Politecnico di Torino

Realizzazione di un banco a pendolo inverso con attuatore pneumatico e controllo mediante PLC e Arduino

Candidato:
Alessandro Marino S231351

Relatori:
Terenziano Raparelli
Luigi Mazza
Federico Colombo
Andrea Trivella

Indice generale

1. Introduzione

Qual è la problematica:

il problema consiste nel tenere in equilibrio un pendolo inverso, che per sua natura è un sistema instabile. Il pendolo è montato su un carrello che può scorrere su una guida lineare, il sistema di controllo avviene tramite un PLC (Programmable Logic Controller) sul quale possono essere utilizzate diverse modalità di programmazione come ad esempio la programmazione Ladder, FBD (che utilizzano controllori PID) e la programmazione Fuzzy che utilizza una logica completamente differente e innovativa.
In una fase successiva è stato previsto l'inserimento di un controllo tramite la scheda programmabile Arduino, dove si può notare la differenza rispetto al controllo tramite PLC.
In questi ultimi anni si stanno affermando sempre di più veicoli per il trasporto di persone che utilizzano sistemi di controllo di questo tipo, Segway ne è un esempio, essendo uno dei primi costruttori a creare veicoli in grado di bilanciarsi e portare carichi su strutture di natura instabile come i veicoli a due ruote affiancate o monoruota.
Alcuni esempi sono i seguenti:

Fig 1: Segway

Nell'immagine a sinistra si può vedere il primo veicolo prodotto da Segway e forse il più conosciuto tra tutti i prodotti di questo marchio.

Si può decidere la direzione di movimento sbilanciandosi in avanti o indietro.

Il sistema di controllo usa un giroscopio per conoscere l'inclinazione, salvo questa differenza, la logica di funzionamento è simile a quella usata sul banco in tesi.

Tramite questo procedimento il veicolo può avanzare e muoversi con facilità.

Questi veicoli si stanno diffondendo per coprire dei brevi tragitti casa-lavoro oppure i tragitti finali.
Da qui la necessità di avere veicoli sempre più piccoli e trasportabili con sé.
Si può pensare ad un utente che per giungere sul luogo di lavoro debba compiere una tratta in automobile ma l'azienda dove lavora si trova ad esempio nel centro città dove con la macchina non può accedere: con l'aiuto di un veicolo del genere, che può riporre comodamente nel bagagliaio dell'automobile, l'utente può percorrere gli ultimi km dal parcheggio al luogo di lavoro in maniera silenziosa, comoda, evitando ingorghi e caos cittadino.

Nell'immagine a destra si vede una monoruota, sempre prodotta da Segway, che sfrutta la stessa logica di quella descritta in precedenza.
Questi veicoli sfruttano dunque tecniche di controllo basate sul pendolo inverso perché devono essere leggeri, compatti e facilmente trasportabili.
Avere una ruota in meno, rispetto ad una bicicletta, comporta lo svantaggio di avere un punto di appoggio in meno che rende il veicolo instabile, ma ha il vantaggio di esser più piccolo e meno ingombrante.

Fig 2: Monoruota

Per il futuro:

Si può pensare ad un futuro dove i robot saranno sempre più presenti nella nostra vita, non solo a livello industriale dove già da un po' di anni trovano importanti applicazioni, ma anche nella vita di tutti i giorni a livello "civile":

Segway ha creato questo robot che si può vedere come un'evoluzione intelligente dei primi sistemi di trasporto prodotti dall'azienda.
La modifica è stata fatta creando un'unità "intelligente" capace di interagire con l'umano in maniera diretta e di "apprendere" giorno dopo giorno.
Oltre alla funzione di robot questo dispositivo può essere usato anche come mezzo di trasporto.

Fig 3: Segway Robot

Come il problema è stato affrontato:

Generalmente il problema del pendolo inverso è affrontato utilizzando delle valvole proporzionali, come quelle utilizzate in questa tesi, ma potrebbero essere usate anche valvole digitali con la logica del PWM.
In rete si trovano molti esempi di pendolo inverso, molti sono gestiti da controllori elettronici come Arduino e generalmente non sono vincolati a stabilizzare il pendolo su un percorso limitato in quanto sono dotati di ruote (è la logica che viene applicata ad esempio su Segway).
Controllare un sistema pendolo inverso comandato tramite un cilindro pneumatico è ancora più difficoltoso rispetto altri sistemi di attuazione, quali motori elettrici rotativi o lineari.
Il sistema pneumatico infatti ha una capacità d'aria, neanche uguale tra le due camere e continuamente variabile in funzione della posizione, che comporta un effetto molla ad ogni cambio di direzione.
Inoltre il sistema è più lento rispetto a progetti simili che utilizzano motori, in quanto ci sono dei tempi di ritardo che sono legati al circuito pneumatico.
La sfida per questa tesi è stata quella di prendere un sistema di natura instabile e renderlo stabile attraverso un controllo raffinato adattato ai componenti utilizzati.
Dei molti esempi trovati sulla rete, quasi nessuno utilizzava componenti pneumatici per movimentare il carrello, questo fa intuire che il compito, già in principio difficile, veniva ulteriormente complicato da questa scelta. Tuttavia, come si vedrà nel seguito di questa tesi, il banco ha dato risultati eccellenti sotto il punto di vista del funzionamento.

2. Teoria del pendolo inverso

Il pendolo inverso è un pendolo che ha il suo centro di massa al di sopra della cerniera, è dunque un sistema instabile di sua natura.
Molte applicazioni del pendolo inverso sono ad un grado di libertà, dunque fissando un asse di rotazione si tenta di fargli mantenere l'equilibrio ad esempio applicando una coppia all'asse stesso. Altre soluzioni, come quella adottata per questa tesi, è far muovere un carrello che supporta la cerniera stessa per far cambiare l'angolo dell'asta.
Il problema del pendolo inverso è un classico problema della dinamica e della teoria di controllo, e viene spesso usato come riferimento per il test di alcuni algoritmi (controllori PID, reti neurali, Fuzzy control ecc).

Si possono dunque scrivere le equazioni secondo Lagrange del pendolo:

$$L=\frac{1}{2}\cdot M\cdot v_1^2+\frac{1}{2}m\cdot v_2^2-m\cdot g\cdot l\cos\theta$$

dove v_1 è la velocità del carrello (indicato con massa M) e v_2 è la velocità della sfera di massa m all'estremità dell'asta. L'asta viene considerata inerte.

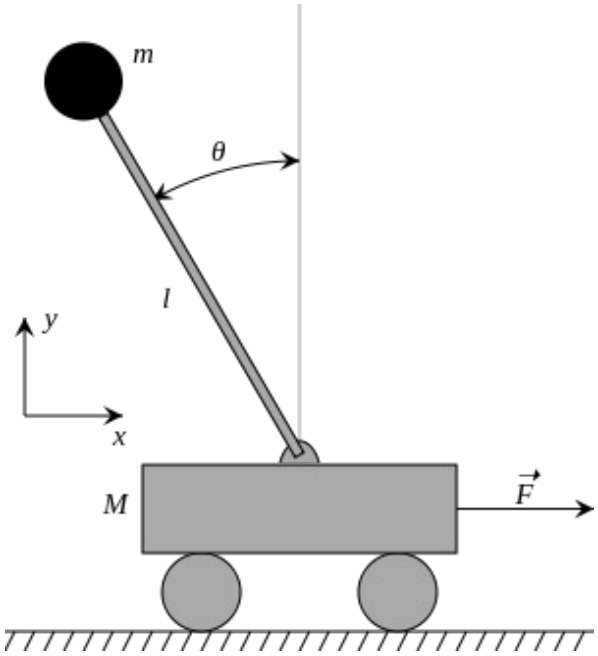


Fig 4: Modello Carrello

$$v_1^2=\left(\frac{dx}{dt}\right)^2=\dot{x}^2$$

$$v_2^2=\left(\frac{d\left(x-l\cdot sen\,\theta\right)}{dt}\right)^2+\left(\frac{d\left(l\cdot\cos\theta\right)}{dt}\right)^2$$

semplificando:

$$v_2^2=\dot{x}^2-2\cdot l\cdot\dot{x}\cdot\dot{\theta}\cdot\cos\theta+l^2\cdot\dot{\theta}^2$$

e la funzione lagrangiana:

$$L=\frac{1}{2}\cdot\left(M+m\right)\cdot\dot{x}^2-m\cdot l\cdot\dot{x}\cdot\dot{\theta}\cdot\cos\theta+\frac{1}{2}\cdot m\cdot l^2\cdot\dot{\theta}^2-m\cdot g\cdot l\cdot\cos\theta$$

le equazioni del moto risultano essere:

$$\frac{d}{dt}\cdot\frac{\delta L}{\delta\dot{x}}-\frac{\delta L}{\delta x}=F$$

$$\frac{d}{dt}\cdot\frac{\delta L}{\delta\dot{\theta}}-\frac{\delta L}{\delta\theta}=0$$

$$\left(M+m\right)\cdot\ddot{x}-m\cdot l\cdot\ddot{\theta}\cdot\cos\theta+m\cdot l\cdot\dot{\theta}^2\cdot sen\,\theta=F$$

$$l\cdot\ddot{\theta}-g\cdot sen\,\theta=\ddot{x}\cdot\cos\theta$$

Queste equazioni non sono lineari ma il controllo dovrebbe mantenere il pendolo in posizione verticale con $\theta\approx 0$

3. Costruzione del banco

Il **banco** è stato realizzato in legno di abete, ha un ingombro massimo di 1,5 metri ed è di semplice realizzazione in quanto è formato da soli 5 pannelli di legno. Questo banco è stato creato in modo tale da favorire l'economicità, la praticità d'uso, la semplicità e la chiarezza del sistema che è montato al di sopra.
Come si può notare sul piano del banco sono montati i seguenti elementi:
-un cilindro pneumatico
-un carrello che scorre su una guida lineare a cui è connesso il pendolo
-un trasduttore di posizione lineare LVDT
-un trasduttore angolare
-quattro valvole proporzionali per il carico/scarico di ciascuna camera del cilindro

Sul pannello frontale si trova uno switch che serve per la selezione del PLC, il "PLC 1" è in questo caso il Rockwell, montato per primo sulla guida DIN, mentre "PLC 2" è il PLC Siemens.
É stata prevista anche una predisposizione di un pulsante di emergenza, che potrà essere collegato con una successiva evoluzione del banco.
Si nota inoltre un connettore per una pulsantiera a filo connessa al PLC Siemens.

Trasduttori:
- trasduttore di posizione LVDT,
- trasduttore di rotazione, potenziometro rotativo

Attuatori:
- valvole proporzionali comandate direttamente in PWM

Commutazione:
- 3 relè a 4 contatti, alimentati dallo switch sul pannello frontale

Controllori:
- PLC Rockwell
- Arduino Mega con scheda/interfaccia di potenza per pilotaggio valvole
- PLC Siemens (utilizzato da un altro studente)

Struttura:
- in legno con profilati in alluminio 20x20 montati in parallelo per l'installazione precisa dei componenti

Alimentazioni:
- all'interno del banco sono disponibili due tipi di alimentatore, un alimentatore per PC Desktop (ATX) e un alimentatore fisso da banco a 24V DC. Questi forniscono diverse tensioni e verranno illustrati in dettaglio successivamente

Il banco è dotato di due maniglie laterali per consentirne il sollevamento, una maniglia frontale con appositi fermi sui lati per consentirne l'apertura a modo di sportello, con cerniera in basso.

L'alimentazione del banco è a 220V AC tramite il cavo multipolare grigio posizionato sul retro.
Il cavo per l'alimentazione va inserito nell'apposito foro dove si trova il connettore dell'alimentatore ATX che provvede, tramite un bypass, anche all'alimentazione del PLC Rockwell.

Alimentazioni dei controllori:

1. PLC Rockwell: Alimentazione 220V AC con cavo di protezione, il modulo centrale è il trasformatore;

2. Arduino si può alimentare in differenti maniere, ad esempio:
   - Alimentazione 12V DC tramite connettore apposito (accetta da 7V DC fino a 20V DC, guardare il datasheet dell'Arduino per avere più informazioni a riguardo);

   -Alimentazione 5V DC stabilizzata, sul fascio di cavi che vanno al connettore principale della scheda di potenza di Arduino, è stato lasciato un cavo "libero" connesso al pin Vin di Arduino, l'alimentazione tramite questo pin deve essere molto stabile e precisa;

   Nota bene: per utilizzare le due precedenti alimentazioni, fornite direttamente dal trasformatore ATX installato nel banco, bisogna stabilizzare il trasformatore ATX collegando all'uscita 5V DC (cavo rosso +5V, cavo nero GND) un carico che assorba almeno 1A. Con un carico sul 5V si noterà un incremento della tensione sul cavo giallo da 11.8V a 12.2V circa, questo è normale ed è dovuto alla stabilizzazione.
   Se questo non viene fatto, Arduino può avere problemi o risposte totalmente inattese, nello specifico mi è capitato che il banco con l'alimentazione tramite USB funzionava correttamente, mentre con l'alimentazione tramite spinotto 12V non funzionava correttamente (ampie oscillazioni del carrello rispetto all'altra alimentazione).

   -Alimentazione tramite USB, utilizzabile quando Arduino è collegato al PC per la programmazione, oppure senza PC è disponibile un trasformatore 5V DC a spina con uscita USB

3. PLC Siemens: Alimentazione 24V DC.

P.s:
Arduino quando non è alimentato viene visto come un carico da parte dei sensori collegati agli ingressi analogici.
Se questo succede tutti gli altri controllori del banco non funzionano perché i valori che leggono sono differenti dalle reali condizioni.
Affinché tutto funzioni, Arduino deve essere alimentato anche durante il controllo con i PLC, oppure se non si vuole che rimanga alimentato, si può disconnettere completamente dal banco tramite il connettore bianco a 6 poli della scheda di potenza.

Fasi costruttive del banco:

Conviene partire dalla fine per avere un'immagine completa del banco, che finito risulta essere così assemblato:

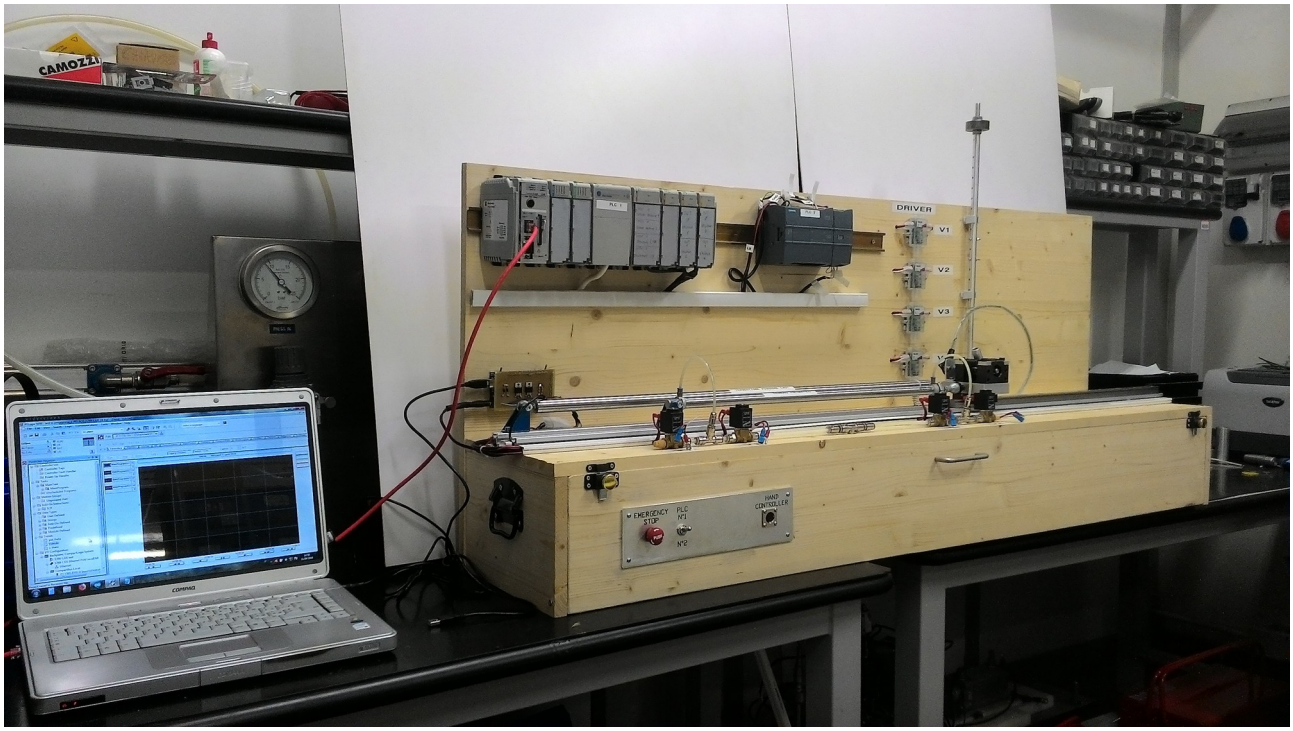


Fig 5: Banco Terminato

Il tutto è stato assemblato partendo dalle tavole di legno, che una volta unite in un unico blocco, sono diventate la struttura portante del sistema:

Sulla **struttura** sono stati aggiunti due profilati di alluminio per il montaggio dei componenti, come ad esempio il trasduttore di posizione **LVDT**, il **cilindro pneumatico** e la guida che supporta il carrello a ricircolo di sfere.

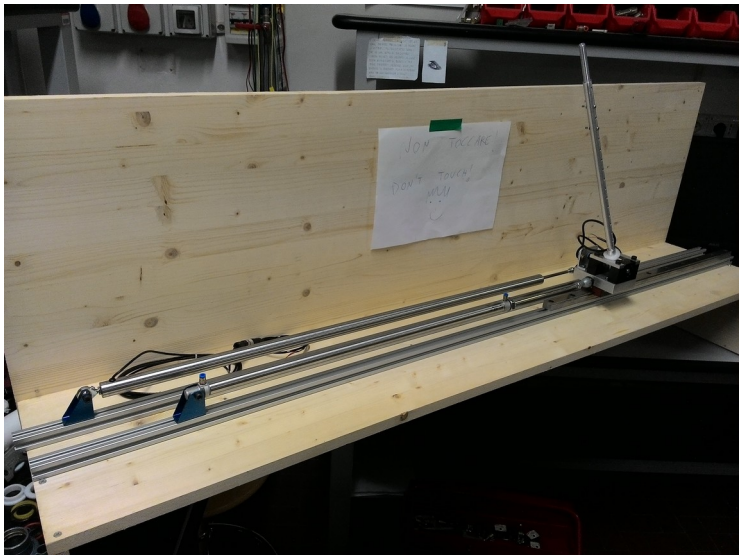


Fig 6: Inizio costruzione banco

Il **carrello** è stato, come il resto del banco, disegnato sul software SolidWorks, il quale ha permesso di avere una visione completa del progetto già prima di iniziare l'assemblaggio.
La parte inferiore del carrello è composta dalla guida a ricircolo di sfere Bosch, che permette di avere uno scorrimento molto fluido quasi senza attriti.

Fig 7: Carrello su guida

La **guida a ricircolo di sfere** ha degli appositi ingrassatori, attraverso il quale è permessa la lubrificazione interna.
La corsa disponibile è di 50 cm.

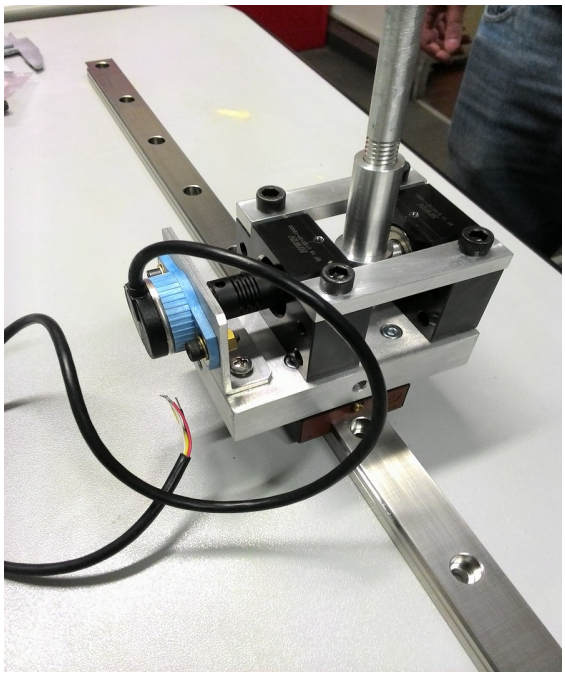


Fig 8: Dettagli carrello

Come si può notare dall'immagine a fianco, l'albero del pendolo è supportato da due cuscinetti a sfere che permettono una rotazione molto fluida.

Collegato all'albero del pendolo si trova un **giunto meccanico** che permette di recuperare qualche eventuale errore di inclinazione tra gli assi del **potenziometro** rotativo e dell'albero del pendolo.

Il potenziometro (in blu) permette la lettura dell'angolo dell'asta.

Sul banco è presente anche un **trasduttore di posizione** lineare, ovvero un **LVDT** (Linear Variable Displacement Transducer) che è collegato tramite degli snodi sferici sia al banco sia al carrello.

L'uscita di questo trasduttore è compresa tra 0 e 10V.

Fig 9: LVDT

Si è proceduto con l'installazione della **guida DIN** sulla quale sono stati montati i due PLC.

A sinistra si può vedere il **PLC Rockwell**, mentre più a destra si vede il **PLC Siemens**.

I profilati in alluminio sono utili per garantire l'allineamento dei componenti come cilindro pneumatico e guida del carrello.
Per sicurezza è stato montato un giunto sferico anche tra l'asta del cilindro pneumatico e il carrello.

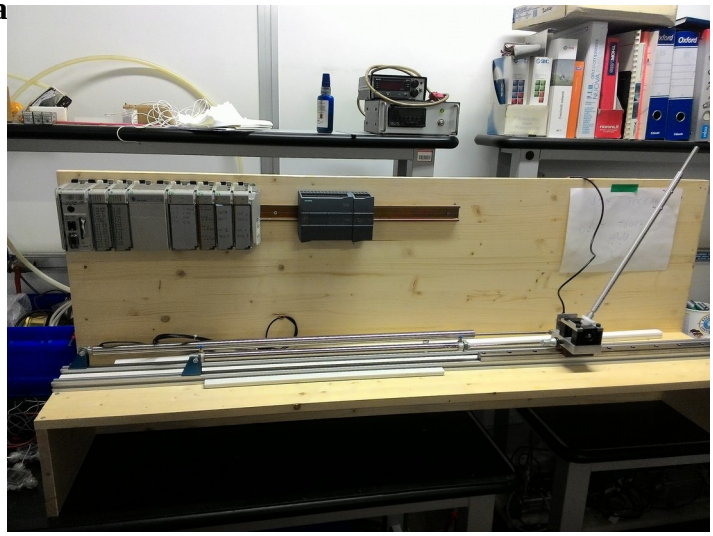


Fig 10: Installazione PLC e guida DIN

La fase successiva è stata quella di inserire le **valvole** sul banco con tutti i collegamenti pneumatici.

Dopodiché, avendo tutti i componenti sul banco, è stato possibile posizionare le canaline elettriche per predisporre l'impianto elettrico.

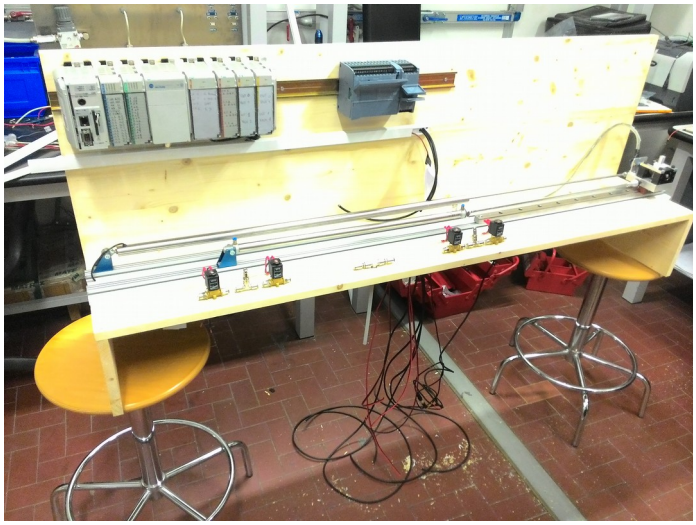


Fig 11: Installazione impianto elettrico

Al di sotto del banco è stato ricavato lo spazio per l'installazione dei due **trasformatori e** della **scatola di commutazione**.

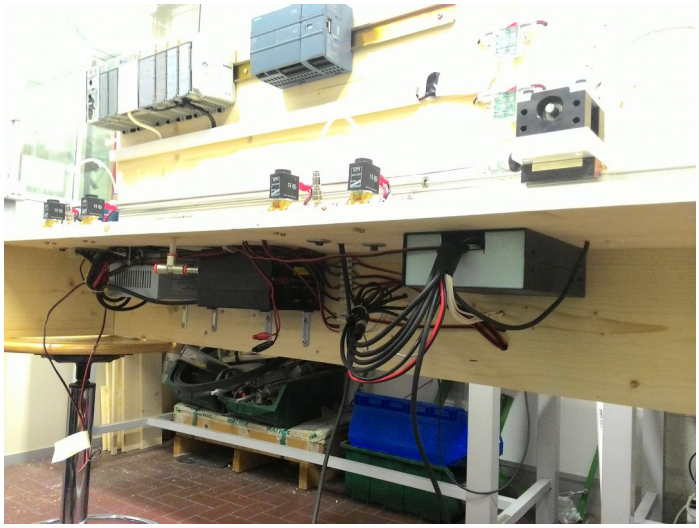


Fig 12: Dettagli impianto

Il **pannello** in legno **frontale** funziona da sportello, può essere abbassato per consentire le operazioni sull'impianto.
Tutti i cavi sono stati etichettati e si può far riferimento agli schemi elettrici in tesi per operare agevolmente sul banco.

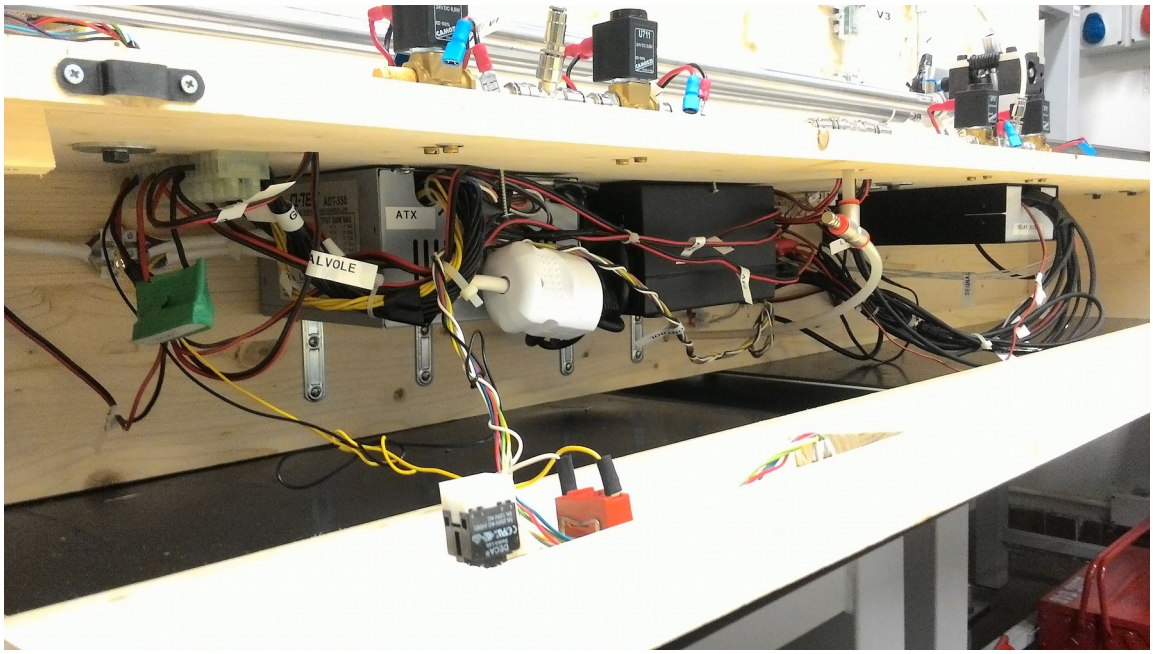


Fig 13: Vista frontale del banco

Dettaglio della **scatola relè**, con tutti i cavi di ingresso e uscita.

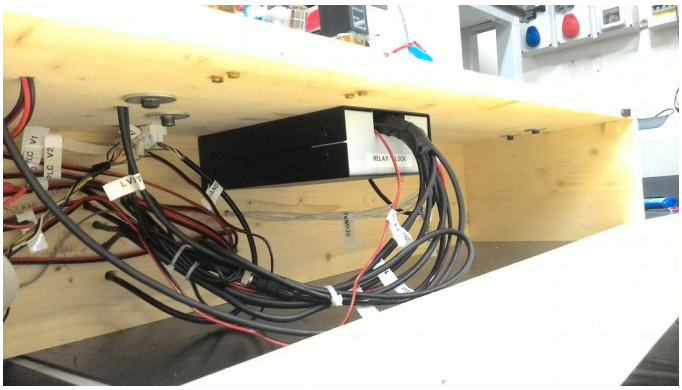


Fig 14: Scatola relè

Scheda **Arduino** collegata al banco tramite l'interfaccia di potenza sulla quale sono stati inseriti i MOSFET.

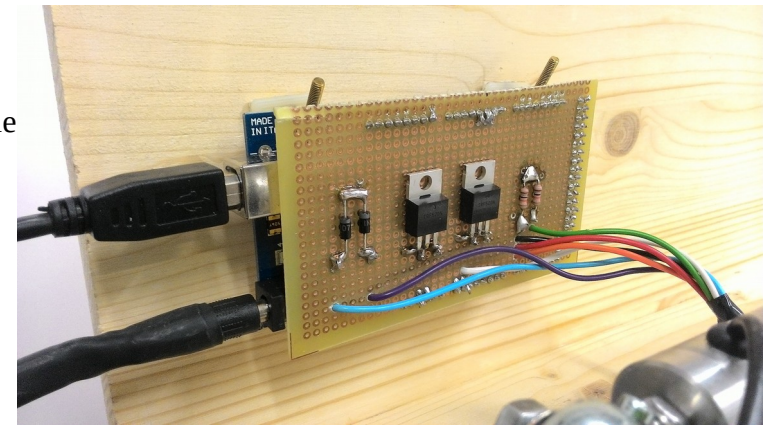


Fig 15: Arduino e scheda di potenza con MOSFET

Dettaglio delle valvole,
il solenoide può essere rimosso dal corpo valvola svitando la ghiera in plastica posta al di sopra.

Si nota la struttura della valvola al cui interno risiede l'otturatore.

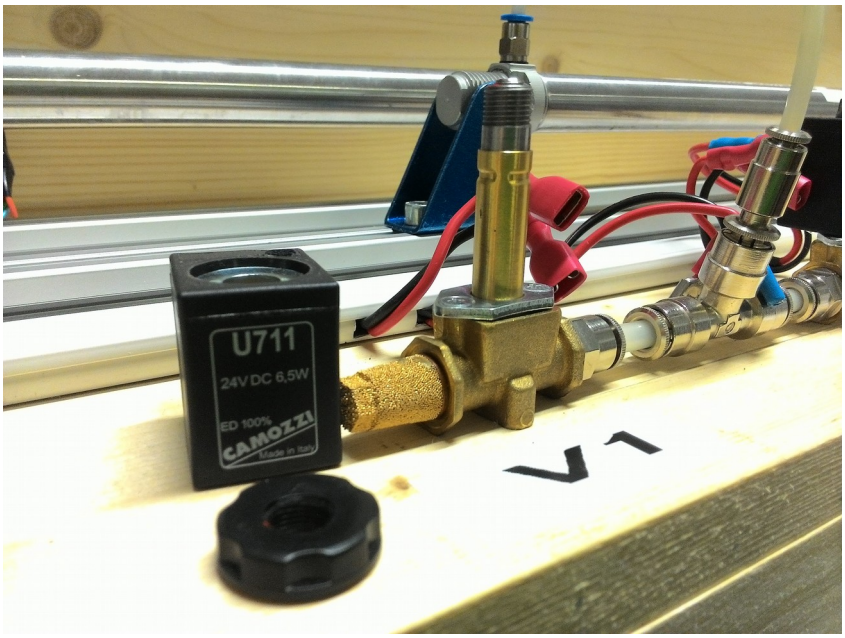


Fig 16: Dettaglio valvola

Dettaglio del carrello, si può notare lo **snodo sferico** che collega il cilindro pneumatico alla base del carrello stesso.

Notare anche i cavi che vanno alle valvole, ora sono connessi i cavi con copri-faston rossi, dedicati ai PLC.

A fianco di questi, altri due cavi che sono riservati al comando con Arduino e che bisogna connettere quando si vuole utilizzare quest'ultimo.

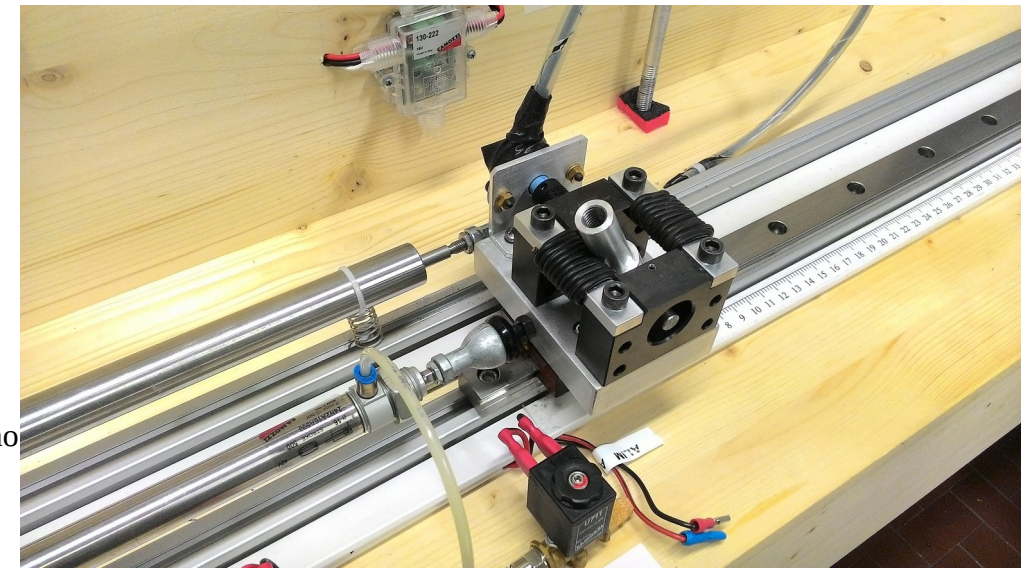


Fig 17: Dettagli carrello

Nelle immagini successive verranno illustrate le connessioni del PLC Rockwell, con i colori dei cavi utilizzati per i vari segnali.

Il modulo 3, ovvero quello aperto, ha le seguenti connessioni:

- Cavo giallo: canale 0 (ovvero Vin 0+), è connesso al sensore di pressione 1 (non usato);
- Cavo grigio: canale 1 (Vin 1+), è connesso al sensore di pressione 2 (non usato);
- Cavo verde: canale 2 (Vin 2+), è connesso al trasduttore di posizione LVDT;
- Cavo bianco: canale 3 (Vin 3+), è connesso al potenziometro che rileva l'inclinazione dell'asta;
- Cavo rosa: ground comune.

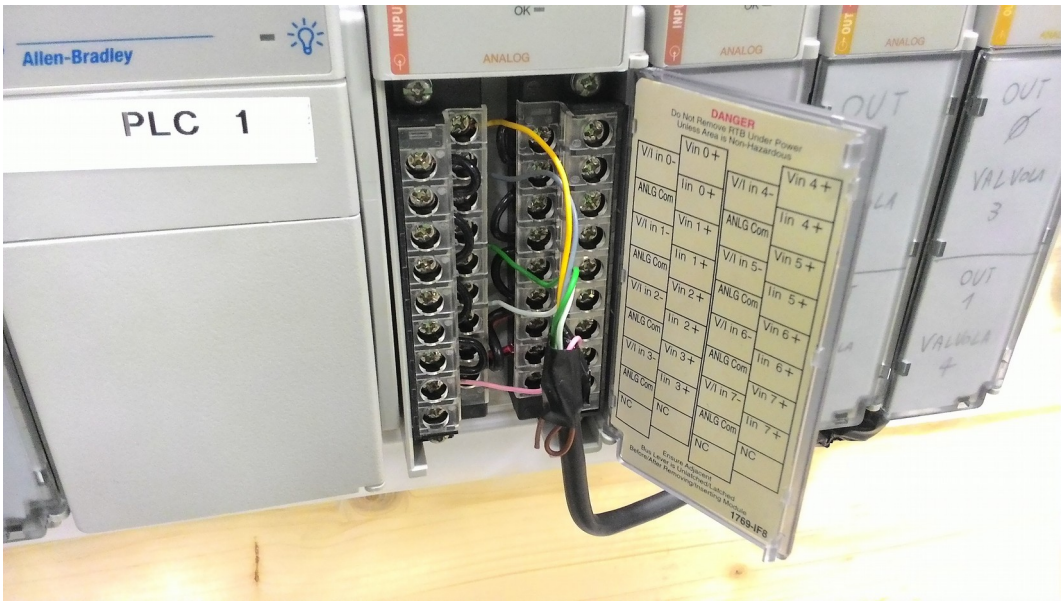


Fig 18: Modulo 3

Modulo 5, ha le seguenti connessioni:

- Cavo giallo: canale 0 (V out 0+), è collegato al driver della valvola V1;
- Cavo verde: canale 1 (V out 1+), è collegato al driver della valvola V2;
- Cavo nero: ground.

Fig 19: Modulo 5

Modulo 6, ha le seguenti connessioni:

- Cavo grigio: canale 0 (V out 0+) collegato al driver della valvola V3;
- Cavo bianco: canale 1 (V out 1+) collegato al driver della valvola V4.
- Cavo nero: ground.

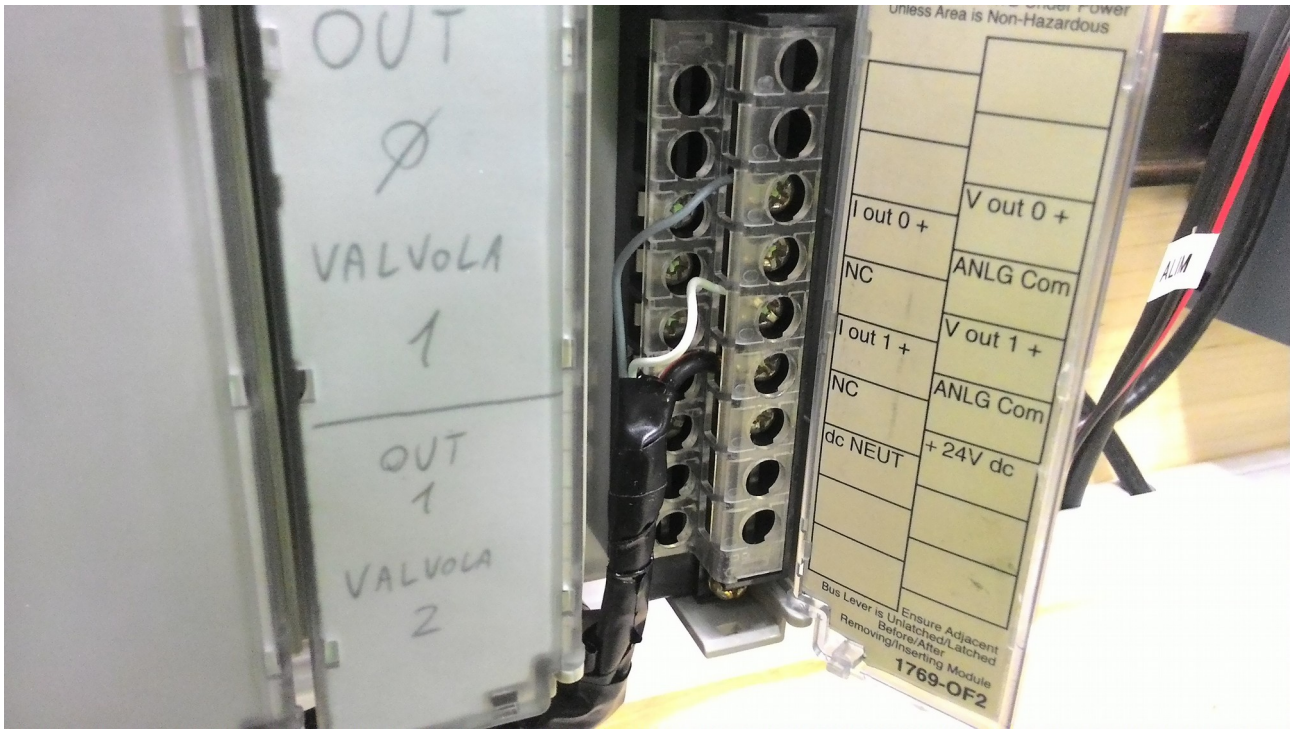


Fig 20: Modulo 6

Il banco terminato:

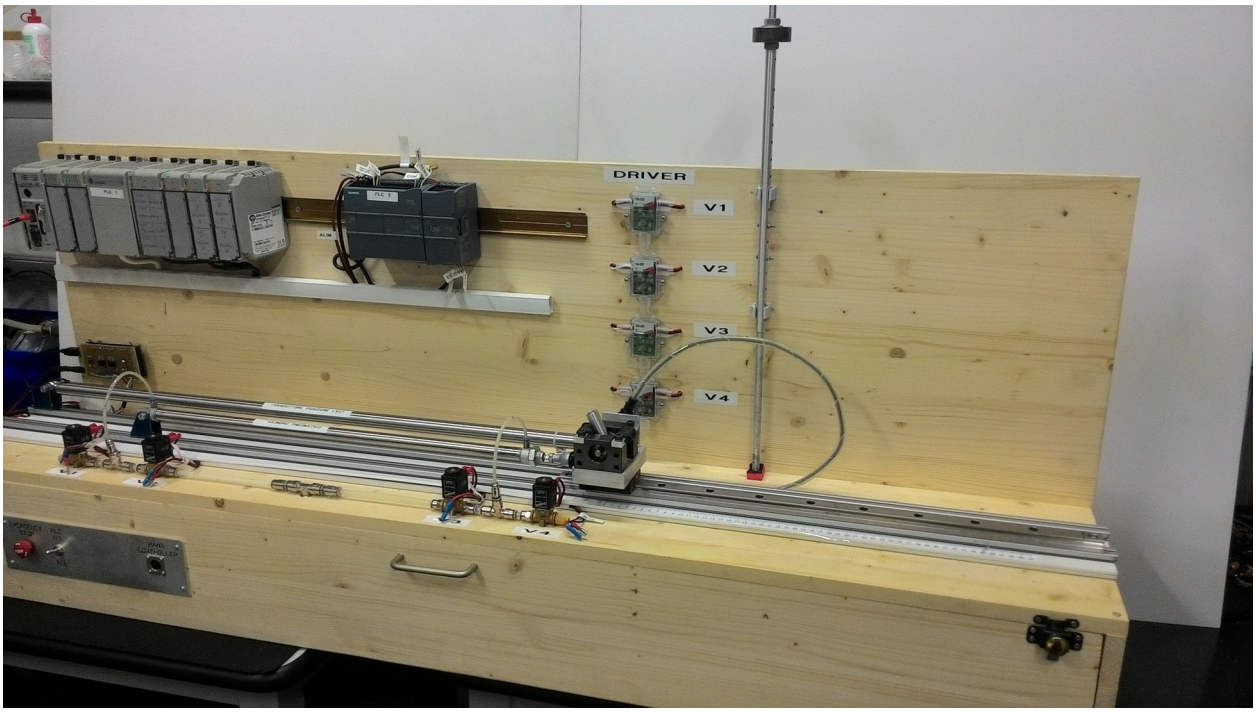


Fig 21: Banco ultimato

4. Il circuito pneumatico

Il circuito pneumatico del banco è relativamente semplice, ed è composto dai seguenti componenti:

- Valvole proporzionali ad otturatore, comandate direttamente in PWM;
- Cilindro pneumatico;
- Silenziatori allo scarico delle valvole;
- Tubi e T di connessione.

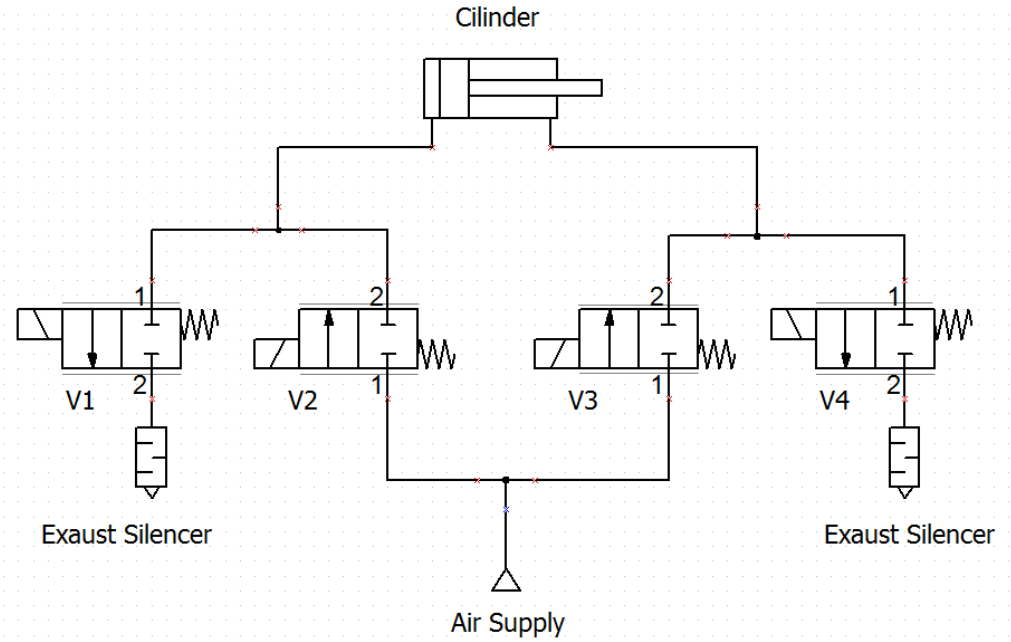


Fig 22: Circuito pneumatico

Funzionamento:

Le quattro valvole sono collegate alle due camere del cilindro pneumatico, queste vengono azionate tramite un comando in PWM.
Si possono identificare le valvole di scarico (V1 e V4) e quelle di carico (V2 e V3).
La camera posteriore del cilindro è collegata alle valvole V1 e V2, mentre quella anteriore è collegata alle valvole V3 e V4.
La rete di aria compressa non deve superare i 7 bar, in quanto oltre questo valore le valvole lasciano fluire dell'aria siccome la molla non riesce a contrastare la pressione al di sotto dell'otturatore stesso.

- Movimentazione del cilindro verso destra: bisogna mettere in pressione la camera posteriore e scaricare la camera anteriore, dunque vengono azionate le valvole V2 e V4, mentre le valvole V1 e V3 rimangono chiuse.
- Movimentazione del cilindro verso sinistra: bisogna mettere in pressione la camera anteriore e scaricare la camera posteriore, si azionano le valvole V1 e V3, mentre V2 e V4 rimangono chiuse.

5. L’impianto elettrico del banco

L’impianto elettrico prevede tutte le connessioni atte a relazionare controllori, sensori e attuatori. È un impianto che trova posto nella parte inferiore del banco stesso e ci sono due principali schemi di cui tenere conto:

- Impianto generale con linee alimentazioni a 3 tensioni;
- Schema di switching, ovvero la parte che consente la commutazione rapida del comando tra un PLC e l’altro

Schema generale:

Si può subito vedere che sono presenti le tensioni 24V, 12V e 5V, ovviamente tutte DC.

- L’alimentazione 5V: questa tensione alimenta solo il sensore di rotazione (potenziometro rotativo) ed è prelevata direttamente dall’alimentatore ATX, il quale fornisce anche la tensione di 12V. Da notare, come detto anche altre volte, che sulla linea 5V si deve inserire un carico resistivo con assorbimento da almeno 1A, che ha solo il compito di stabilizzare le tensioni in uscita dall’alimentatore ATX. Qui non è rappresentato questo componente perché può essere facilmente connesso/sconnesso da un morsetto a vite all’uscita dell’ATX.
  Il colore dei cavi 5V in uscita dall’ATX è il rosso.

- Alimentazione 12V: questa alimentazione è usata per alimentare principalmente l’Arduino, si preleva sempre dai morsetti a vite posti sotto il banco; il connettore di Arduino è facilmente individuabile in quanto è un jack coassiale.

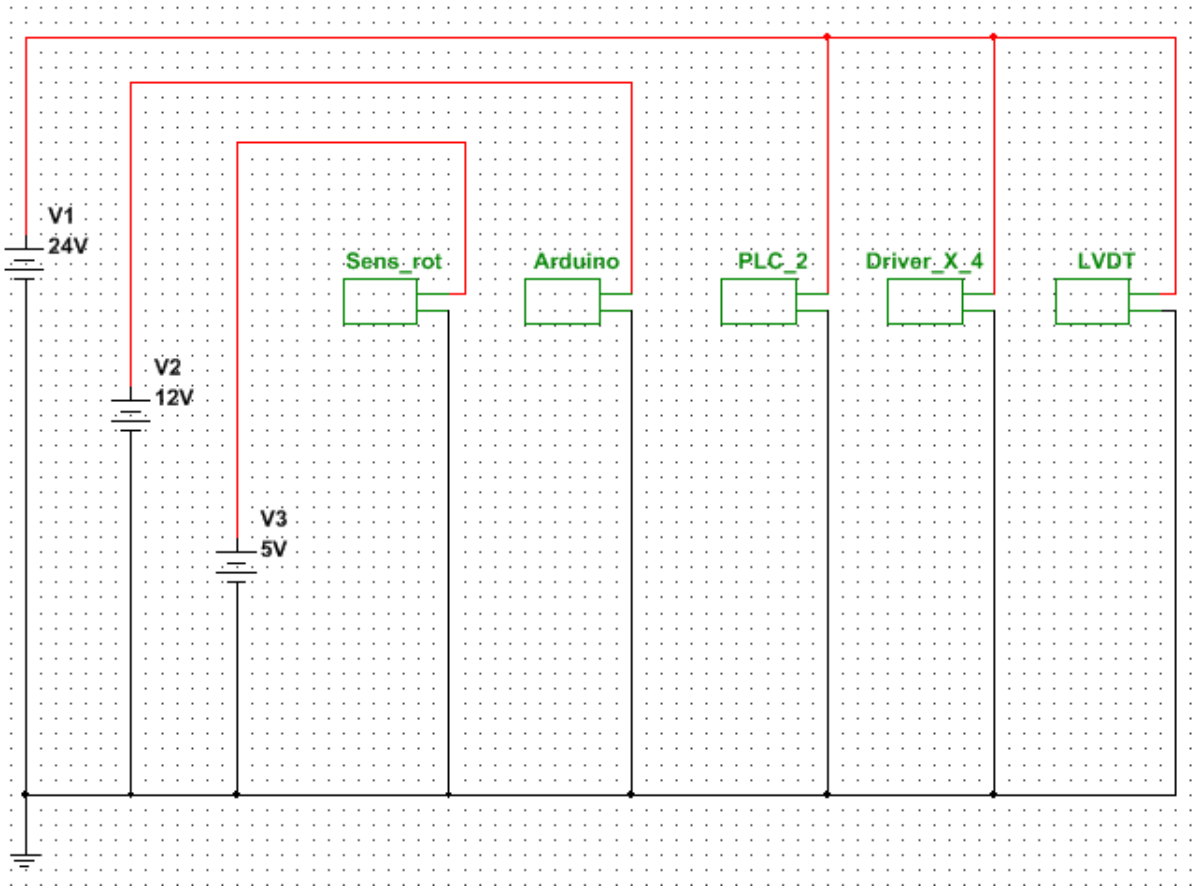


Fig 23: Schema alimentazioni

Da notare il ground comune a tutte le alimentazioni, anche di alimentatori diversi come quello che fornisce il 24V.

- Alimentazione 24V: questa alimentazione è la più importante in quanto ha il compito di alimentare il PLC 2 (Siemens), i 4 driver valvole e il sensore LVDT. È una tensione molto stabile perché fornita da un alimentatore stabilizzato ad una sola uscita.

La parte di switching:

Con J sono indicati i singoli cavi di segnale, i ground sono stati connetti tutti insieme per avere un riferimento comune.
Come si può vedere dallo schema seguente, un blocco relè multicontatto viene azionato da uno switch presente sul pannello frontale del banco.
Questo relè ha un'alimentazione di 24V.

In posizione di riposo, quindi switch aperto e solenoide relè non eccitato, i cavi da J1 a J10 sono collegati agli ingressi del PLC 1 (Rockwell) sui rispettivi moduli I/O.

Azionando lo switch (S), la bobina del relè viene eccitata,dirottando i segnali al PLC 2 (Siemens).

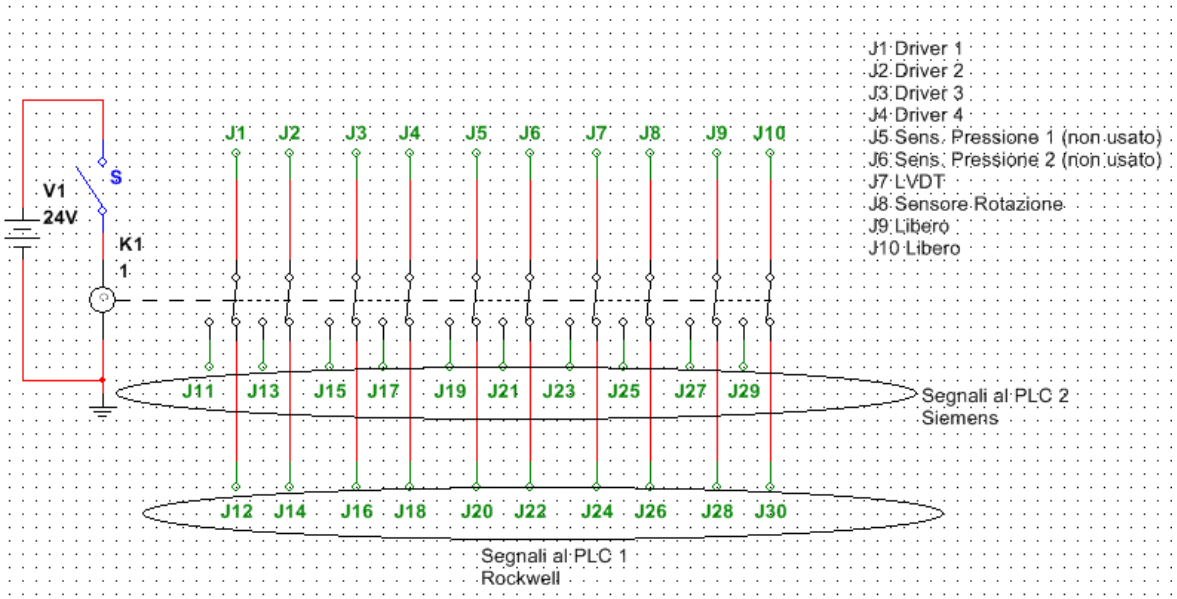


Fig 24: Schema blocco switching

Da notare che non sono stati utilizzati i sensori di pressione, ma il cablaggio è già predisposto se in un futuro volessero venir aggiunti.
Ci sono anche due contatti liberi che possono essere utili per azionare accessori in futuro.
Questo sistema permette un notevole risparmio di tempo quando bisogna cambiare controllore, dando la possibilità durante un'esercitazione, di valutare le risposte con due diversi controllori.
Il banco è stato pensato per un uso didattico, tutto è stato predisposto per essere il più intuitivo possibile, pulito ma funzionale allo stesso tempo.
Tutti i cavi e componenti sono stati etichettati per facilitarne la comprensione.
Il controllo tramite Arduino invece richiede la disconnessione fisica dei faston originari sulle valvole con quelli riservati ad Arduino stesso, questo perché questo controllore è stato aggiunto per ultimo su un sistema con una logica diversa (infatti i PLC usano i driver, mentre Arduino no).

6. Settaggio del software RSLogix 5000 e configurazione del PLC

Il PLC utilizzato è il modello **Compaq Logix 1769-L32E**, il quale richiede esclusivamente il software **RSLogix 5000**, nello specifico la **versione 19.10** utilizzabile con Windows XP e Windows 7.
Installato il software, si procede con l'apertura dello stesso, si sceglie "New Project" dopodichè viene visualizzata una schermata dove si deve scegliere il PLC in uso:

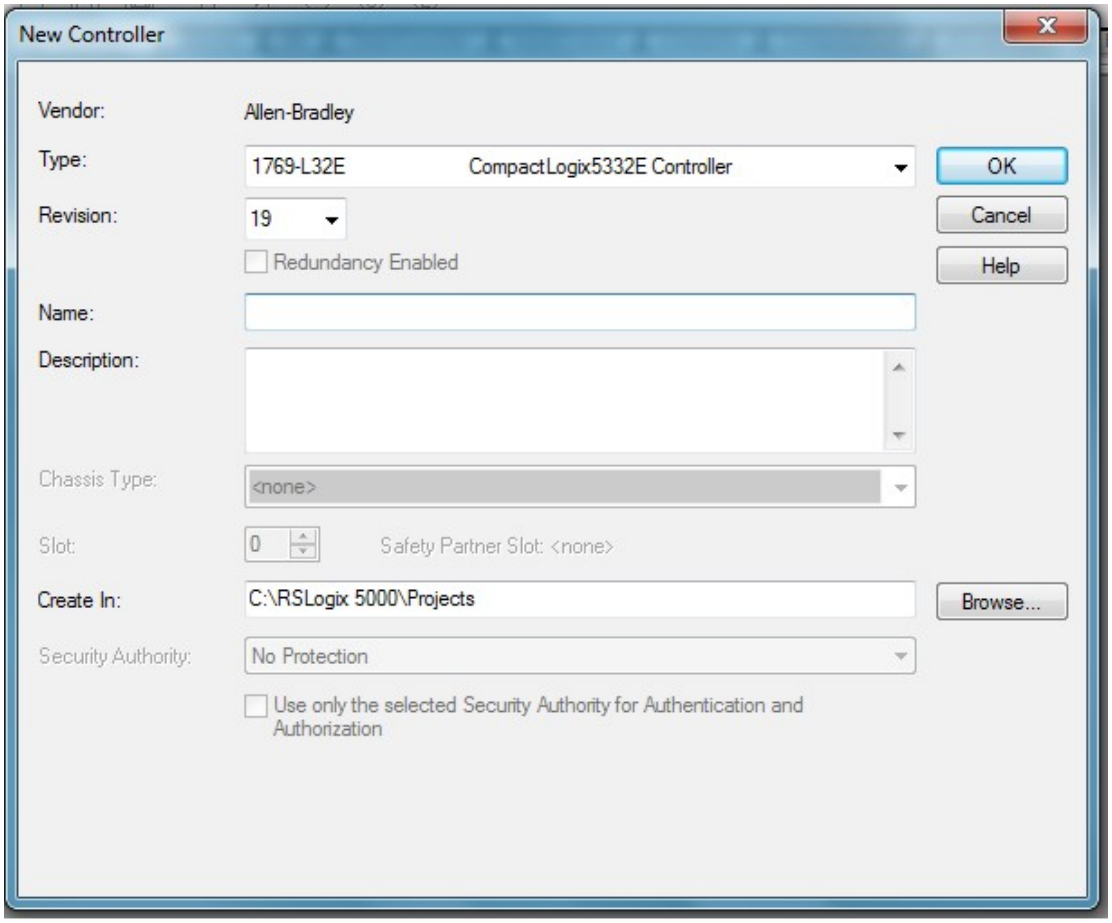


Fig 25: Scheda selezione PLC

Avviato il programma si può scegliere il tipo di connessione al PLC, le più semplici da utilizzare sono la connessione Ethernet oppure la connessione Seriale.

Per velocità è facilità di configurazione è consigliabile optare per la connessione Ethernet, che tuttavia ha qualche particolarità:

- ## Connessione diretta tramite cavo Ethernet standard

Possibile solo se il computer ha integrato uno Switch interno, che consente di connettere due dispositivi che sono sullo stesso livello. Il cavo Ethernet collega direttamente il PC e il PLC;

Connessione tramite Switch e cavo Ethernet standard

Se il computer è datato, oppure non ha Switch interno, si può optare per questa modalità per metterli in comunicazione;

Connessione tramite cavo Ethernet incrociato

questa connessione di differenzia dalla prima solo per il tipo di cavo Ethernet, in sostanza si ha un cavo con le seguenti connessioni sugli spinotti:

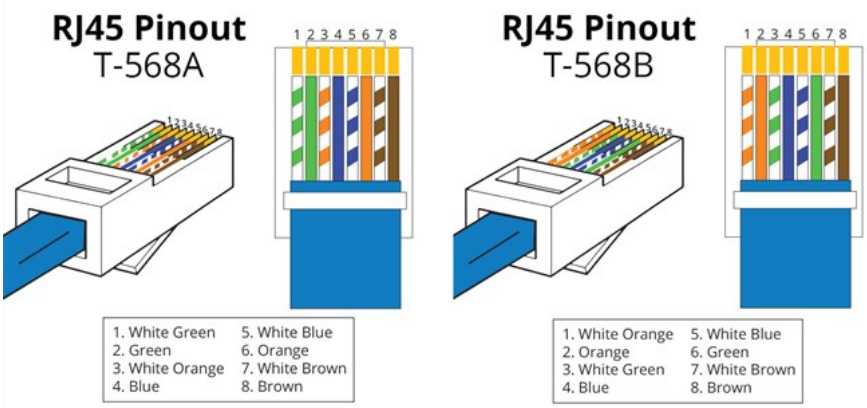


Fig 26: Connettori Ethernet

Come si può vedere dall'immagine si può creare un Ethernet incrociato prendendo un cavo comune e collegando un estremo in configurazione T-568A e l'altro estremo in configurazione T-568B.

Fatto questo si deve connettere il cavo con il PC e il PLC, si apre la schermata delle connessioni del computer e si imposta un indirizzo IP statico, ad esempio 192.168.1.150 (ma può essere in un range di 192.168.1.100-255).

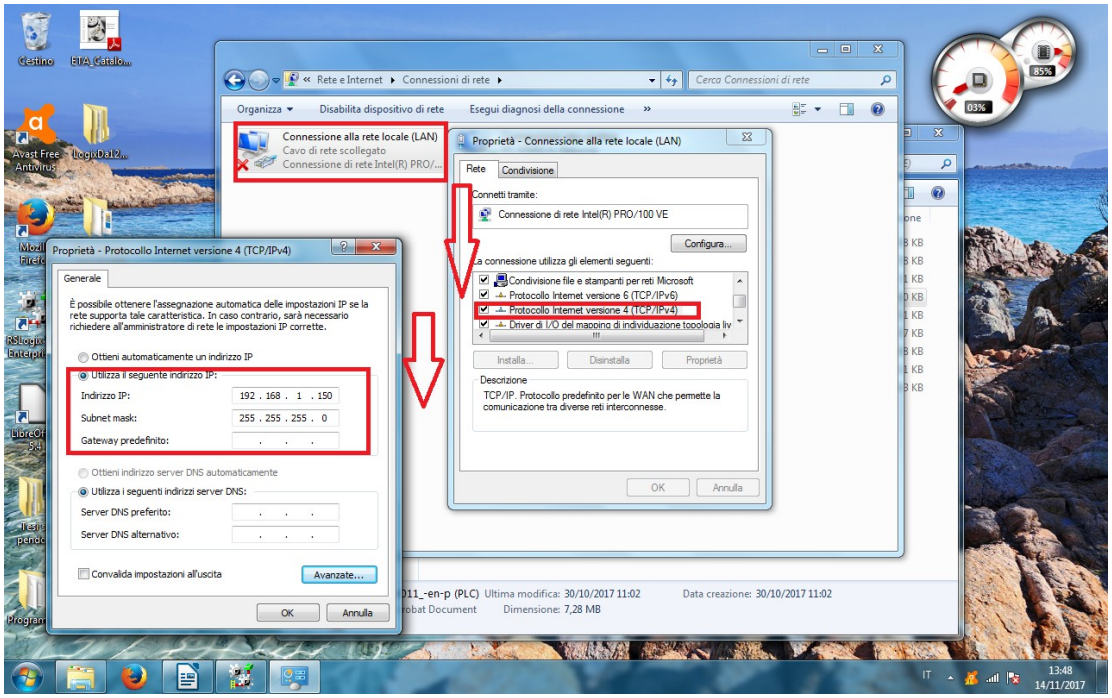


Fig 27: Procedura per configurazione

se tutto è andato a buon fine si vedrà il led della porta Ethernet acceso.

Configurazione dei moduli del PLC

Ora si può procedere alla configurazione dei moduli connessi al PLC, in particolare abbiamo sei moduli che iniziano tutti con 1769.

Nella lista del “Controller Organizer”, c’è la voce “CompaqBus Local” e per aggiungere i moduli basta fare clic con il tasto destro del mouse, selezionare “New Module” e andare a cercare i moduli fisicamente connessi al PLC.

Nel caso il primo modulo è un modulo di ingresso digitale, 1769-IQ16 che ha appunto 16 ingressi come si può intuire dalla sigla.
Importante dare un nome e selezionare il numero corretto dello slot a questo modulo così da ricordarsi a cosa serve e dov’è messo, soprattutto nella definizione del modulo bisogna fare clic su “change” per modificare la voce “Electronic Keyring” che verrà impostata su “Disable Keyring”.
Questa procedura è utile per il riconoscimento e il corretto funzionamento dei moduli e verrà effettuata alla stessa maniera anche sui restanti moduli.

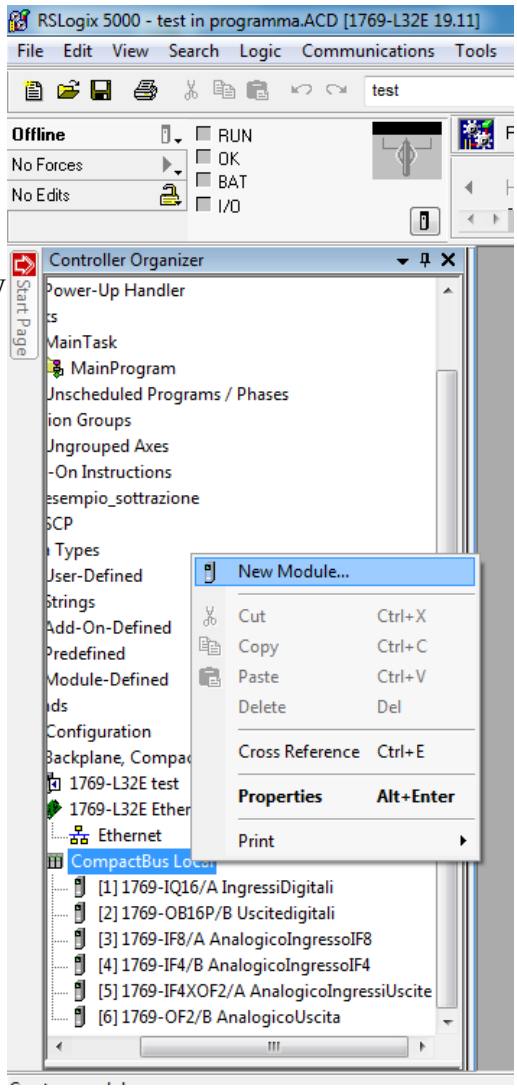


Fig 28: Configurazione moduli

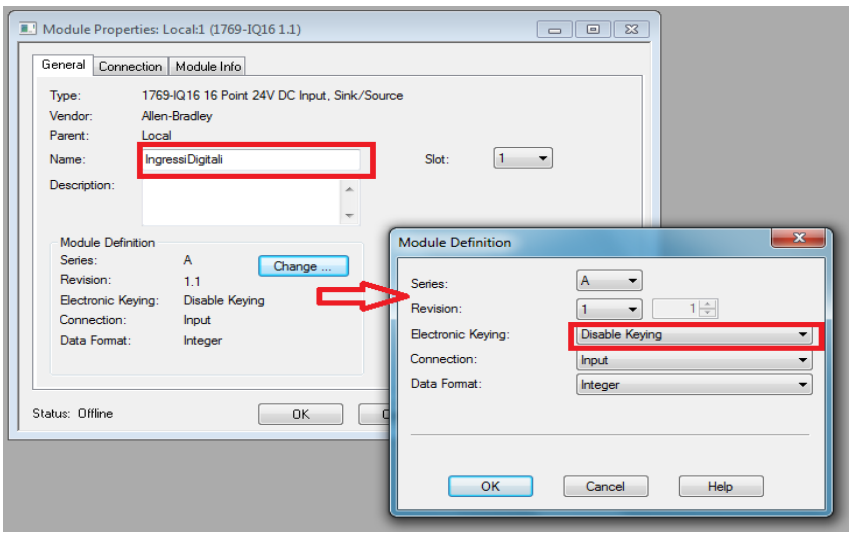


Fig 29: Impostazioni del singolo modulo

Dalla schermata principale cliccare ora la voce “Communications”, successivamente cliccare su “Who Active” e selezionare la scheda che riguarda il PLC;
a questo punto si può cliccare “Go Online” per avviare lo scambio dati tra PC e PLC.

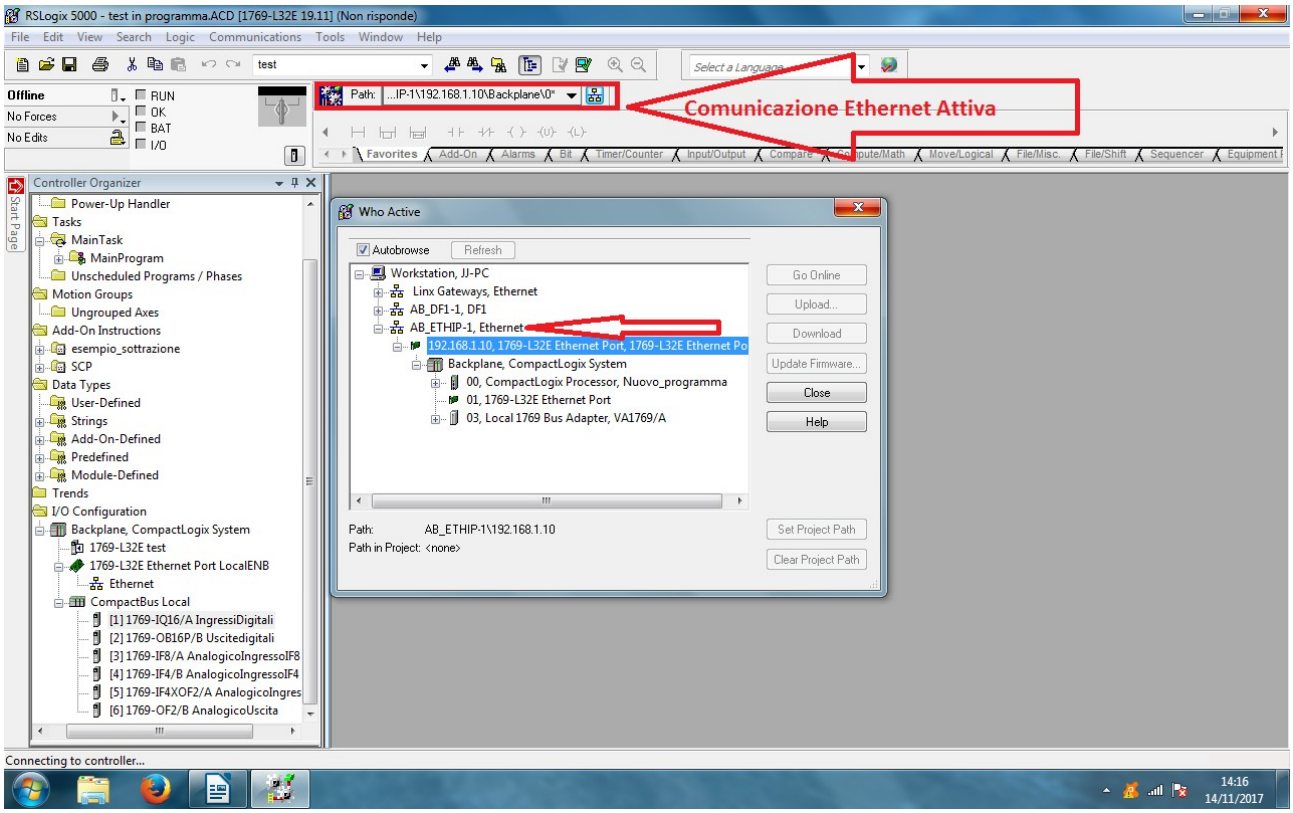


Fig 30: Attivazione comunicazione

7. La modulazione in PWM

PWM e Duty Cycle:

PWM è l'acronimo di Pulse-Width-Modulation, è una tecnica di modulazione usata per comandare dispositivi digitali.
È principalmente usata per il controllo di dispositivi elettrici, come ad esempio i motori, valvole digitali ecc.

Il valor medio di tensione e corrente fornito è controllato collegando il carico attraverso impulsi ripetuti ON e OFF nel quale si controlla la durata attraverso il rapporto di T on e T.
Idealmente più la frequenza è alta più il risultato si avvicina ad un controllo analogico, ma bisogna sempre prendere in considerazione l'applicazione.

Il termine duty cycle è proporzionale al periodo di ON, si ha infatti che $duty\,cycle = \frac{T_{on}}{T}$, ad esempio un duty cycle basso corrisponde ad una potenza fornita bassa, perché l'alimentazione è su OFF per la maggior parte del tempo.
Il duty cycle viene espresso in percentuale, 100% equivale a dire che il carico è sempre alimentato.

Si crea dunque un'onda quadra, data proprio dal continuo cambiamento tra ON e OFF durante il periodo.

Variando il duty cycle si varia la porzione del tempo di ON rispetto al periodo della portante.

Con Arduino il duty cycle può essere effettuato tramite la funzione analogWrite() e i valori di tensione in uscita sono compresi tra 0 volt e 5 volt.

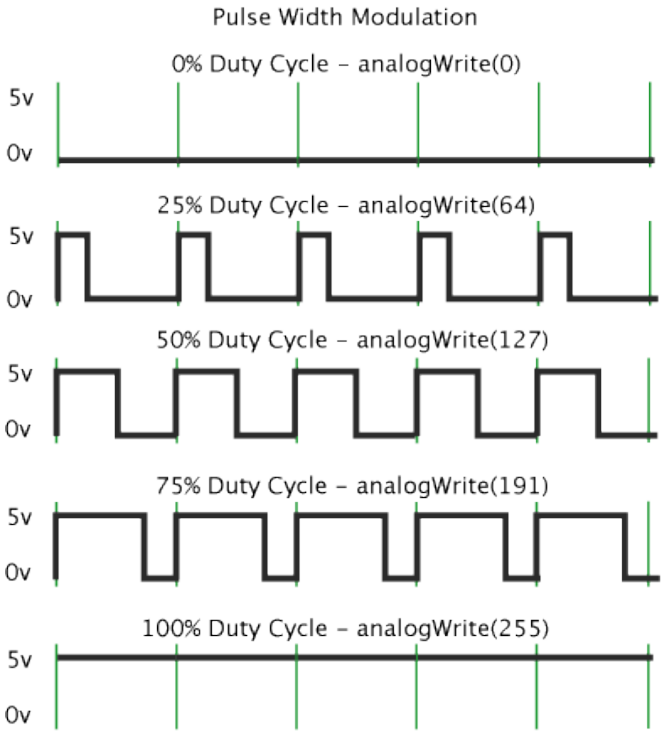


Fig 31: Percentuali di PWM e livelli di tensione

Elettrovalvole:

Esistono due tipi di elettrovalvole, digitali e proporzionali, le quali si dividono in:

1. elettrovalvole digitali, con comportamento ON/OFF;
2. elettrovalvole proporzionali in anello aperto, comandate in PWM, il cassetto si sposta continuamente a seconda del PWM applicato (le valvole usate sul banco sono di questo tipo);
3. elettrovalvole proporzionali in anello chiuso (è controllata la posizione del cassetto), in cui la portata o la pressione sono proporzionali al segnale di riferimento, che in genere è un segnale in tensione.

Nel banco in tesi sono state utilizzate delle valvole proporzionali in flusso in anello aperto, sfruttando la tecnica del PWM, questo ha dei vantaggi economici rispetto alle valvole proporzionali ad anello chiuso perché queste costano di meno e sono meno ingombranti.

Come funzionano le valvole proporzionali in anello aperto:
Il solenoide è alimentato con un segnale PWM, la corrente media nel solenoide determina la posizione di quest'ultimo.
Come si può vedere dall'immagine seguente, una molla si oppone allo spostamento del cassetto, dunque maggiore è la corrente applicata, più la molla verrà compressa.

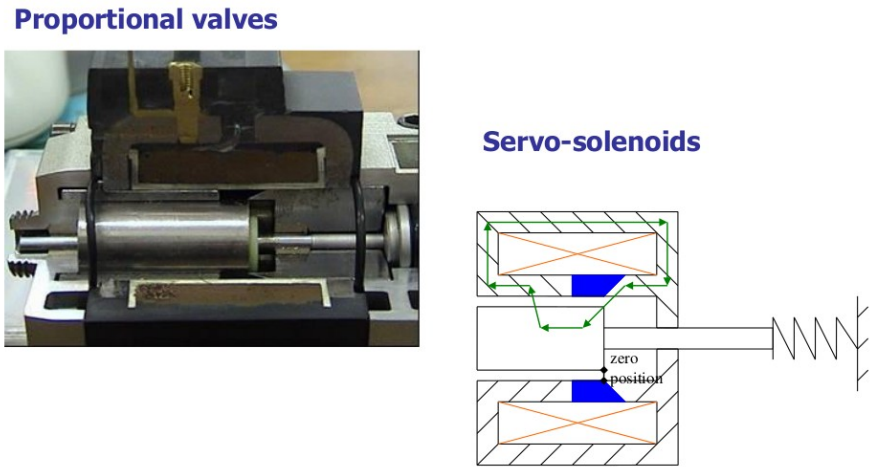


Fig 32: Valvola proporzionale

Il grafico Forza-Spostamento mette in relazione queste due grandezze variando la corrente media nel solenoide.
Le curve infatti sono differenziate dal valore medio di corrente che attraversa il solenoide, che è funzione del PWM applicato.
Si lavora nella regione lineare in quanto c'è una relazione tra lo spostamento del cassetto e la forza della molla.

Si può dire che la forza imposta dalla molla è $F = K \cdot X$ dove K è la costante elastica della molla, mentre F è la forza esercitata dalla molla stessa.

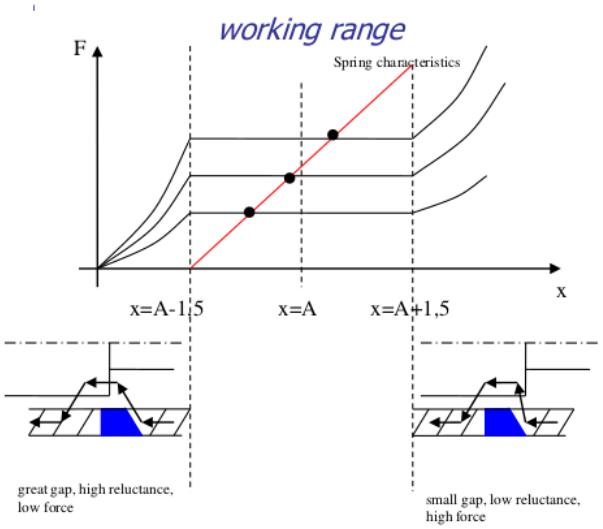


Fig 33: Campo di lavoro, caratteristica forza-spostamento

A regime, quindi con il cassetto fermo, questa forza è anche uguale a quella esercitata dal solenoide.

Volendo le valvole proporzionali potrebbero essere anche sostituite da valvole digitali:
Le elettrovalvole digitali prendono il nome dalla posizione dell'otturatore, che appunto può essere aperto o chiuso. Le elettrovalvole digitali sono anche più robuste rispetto a quelle proporzionali perché eventuali impurità nell'aria non trattata andrebbero a danneggiare la valvola o comunque porterebbero ad un funzionamento non corretto.

Modulazione PWM applicata alle valvole digitali:

A differenza delle valvole proporzionali che utilizzano uno spostamento continuo del cassetto (o un otturatore) agente sulle luci di passaggio per regolare il flusso, le valvole digitali hanno solo due posizioni stabili: apertura completa (modalità ON) e chiusura completa (modalità OFF).
La valvola è costituita da un cassetto o un otturatore scorrevole, sul quale agiscono la forza esercitata da una molla di contrasto (1) e la forza elettromagnetica generata dal solenoide(2).

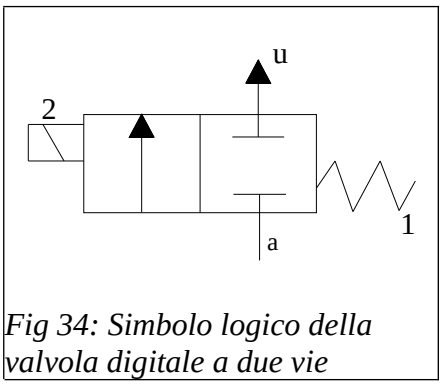


Fig 34: Simbolo logico della valvola digitale a due vie

La molla (1) esercita una forza che tiene chiusa la valvola; quando viene fornito il comando di apertura al solenoide (2), si crea una forza sul cassetto in verso opposto alla molla che vince il precarico e lo fa spostare fino all'apertura.
Per avere un flusso variabile nel tempo, si può utilizzare la modulazione PWM.
La valvola riceve un segnale ad onda quadra di periodo T costante.

Il duty cycle alla valvola è definito come in precedenza, cioè $dc = \frac{t_i}{T}$ dove t_i è la durata variabile dell'impulso inviato alla valvola dal driver o dalla scheda di potenza quando viene utilizzato Arduino.

Variando il duty cycle si ha una variazione di portata media che attraversa la valvola.
Più è piccolo il periodo della portante, tanto più il comportamento della valvola si avvicina al comportamento analogico desiderato, ma stando ben attenti a non eccedere con frequenze troppo elevate (dunque periodi della portante piccoli).
Esiste infatti un limite inferiore del periodo della portante: questo limite è in funzione del tempo di risposta dell'elettrovalvola.

Per avere un corretto funzionamento, la valvola deve compiere l'intero ciclo di apertura-mandata-chiusura restando nel periodo della portante T.
Sul banco sono stati installati dei driver specifici per le valvole utilizzate (proporzionali ad azionamento diretto), che le comandano con una frequenza della portante di 490Hz.
Anche con Arduino la frequenza utilizzata è 490Hz.
Con valvole proporzionali non sussiste il problema descritto in precedenza, perché il solenoide vede sempre una corrente media che lo attraversa e il cassetto (o otturatore nel caso particolare di queste valvole) può assumere posizioni intermedie.

Nell'immagine a fianco è mostrata la relazione tra il duty cycle e la tensione di riferimento V_{rif} .

L'ampiezza del duty cycle viene impostata dalla tensione di riferimento V_{rif} .

Si vede che per ogni istante $t = i \cdot T$ dove i è l'iesimo ciclo.

Facendo il rapporto tra $V_{rif.i}$ e $V_{rif.max}$ si ha il duty cycle relativo al periodo i-esimo T_i

$$dc(T_i) = \frac{V_{rif.i}}{V_{rif.max}}$$

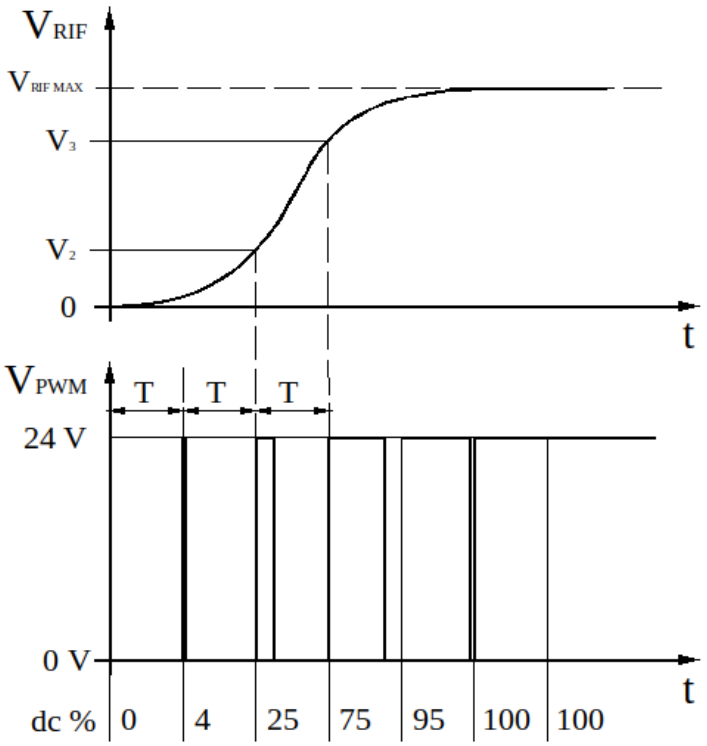


Fig 35: PWM e tensione media di riferimento

Nel caso in tesi, le valvole stesse sono comandate in open loop, ciò equivale a dire che viene inviato un duty cycle ma non viene controllata la posizione del cassetto.

Nelle valvole digitali possono presentarsi le seguenti situazioni:

- mantenendo la frequenza costante e aumentando il DC si ottiene una maggiore portata, cioè la valvola rimane mediamente più aperta.
- mantenendo il duty cycle costante e aumentando la frequenza invece si ha che la zona di non linearità aumenta percentualmente, questo perché le non linearità sono legate ai tempi fisici di apertura e chiusura valvola, che sono fissi, l'inerzia della valvola rimane costante, la forza applicata anche, quindi anche il tempo di apertura non cambierà.
  Aumentando la frequenza si diminuisce il periodo totale del DC e anche il periodo T on.
  Generalmente, aumentando la frequenza fino ad un certo punto il sistema potrebbe diventare più pronto (c'è una frequenza di aggiornamento maggiore) ma dipende con che DC si sta lavorando, comunque sia ci si avvicina sempre più alle zone di non linearità.

Si può dire che esistono 2 tipi di duty cycle:

1. quello vero imposto dal driver o dalla scheda di potenza alla valvola, che è il duty cycle di comando;
2. quello che effettivamente è sull'apertura della valvola, chiamato duty cycle di apertura, ovvero esistono delle non linearità della valvola.

Queste non linearità sono legate al tipo di valvola, alla taglia, alla pressione applicata ecc.
Con duty cycle di comando troppo piccoli la valvola può non riuscire ad aprirsi (generalmente fino a circa il 15%), viceversa la valvola può restare sempre aperta se vengono inviati DC superiori a 85% (ad esempio).
Come si vede nell'immagine successiva gli estremi sono zone non lineari, mentre solo all'interno c'è una regione lineare.

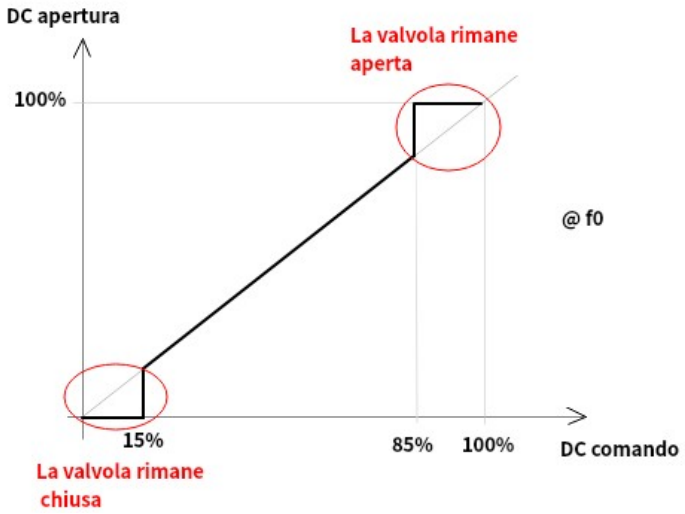


Fig 36: Comparazione DC comando e DC apertura, banda morta piccola

Aumentando la frequenza di duty cycle, ovvero riducendo il periodo della portante, anche il tempo di apertura viene ridotto, ma la valvola ha una dinamica che non può cambiare, perché la massa del cassetto mobile è sempre la stessa, come anche la forza imposta dal solenoide e dalla molla, non cambiano variando il la frequenza della portante. Le zone non lineari quindi aumentano all'aumentare della frequenza della portante.

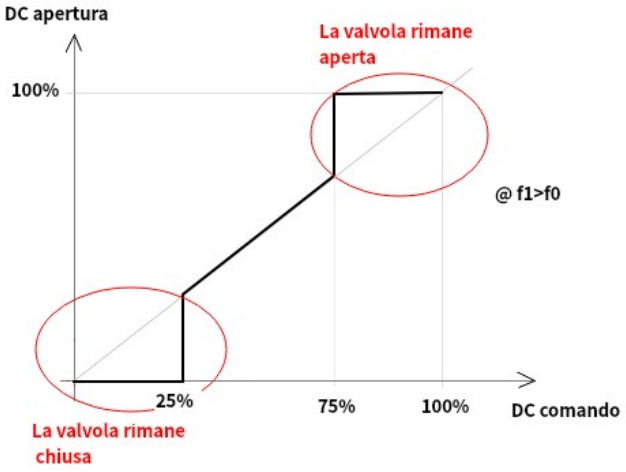


Fig 37: DC comando e DC apertura, banda morta elevata

8. PID

Un PID è un controllo Proporzionale-Integrativo-Derivativo molto usato in ambito nei sistemi di controllo, sia in ambito industriale sia in varie altre applicazioni che richiedono un controllo continuo e modulato.
Il funzionamento di questo sistema si basa sul calcolo di un errore nel tempo che è la differenza tra il setpoint e il feedback, ovvero la variabile di processo misurata **e(t)=Set-Feedback.**
A valle di questo blocco l'errore viene trattato tramite una parte proporzionale, una integrativa e una derivativa, ed è proprio da queste parti che il controllo prende il nome.

Sostanzialmente il controllo applica le correzioni in funzione del setpoint, un esempio molto comune è il cruise control sui veicoli, dove si imposta una determinata velocità di crociera e il sistema la mantiene (o comunque cerca di mantenerla) anche se il veicolo affronta salite o discese che influiscono sulla velocità stessa, che in questo caso è la variabile di processo.

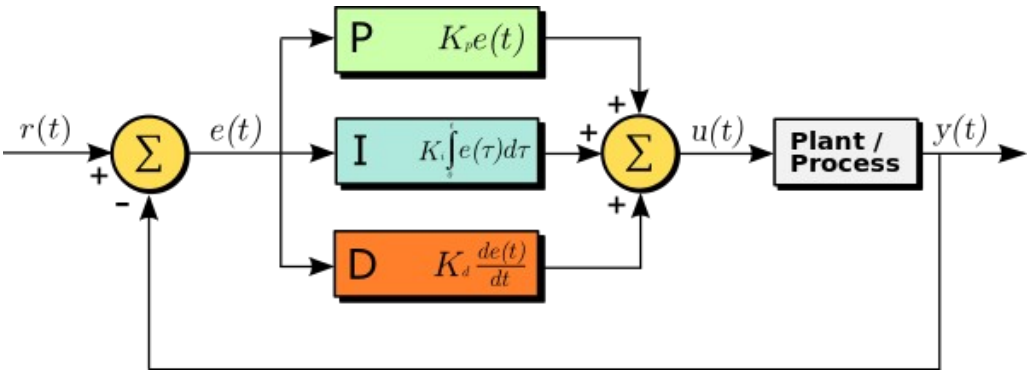


Fig 38: Schema a blocchi di un controllo PID generico

Nell'immagine si vede come al setpoint r(t) viene sottratta la variabile di processo y(t) per generare l'errore e(t).

- Il termine proporzionale è sostanzialmente un'amplificazione proporzionale al valore di Kp dell'errore stesso;
- il termine integrativo serve per eliminare gli errori residui e tenta di eliminarli col passare del tempo dunque a regime, anche questo blocco ha una proporzionalità Ki;
- il termine derivativo valuta il futuro andamento dell'errore valutandone la sua derivata, è anche chiamato controllo anticipatorio, come per gli altri blocchi anche qui c'è una costante di proporzionalità Kd.

Forma matematica:

L'uscita può essere interpretata come $u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)\,dt + K_d \cdot \frac{de(t)}{dt}$

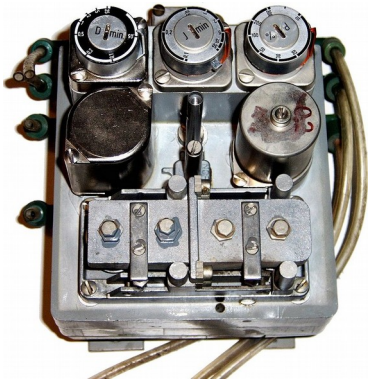
dove Kp, Ki e Kd sono valori non negativi corrispondenti ai termini proporzionale, integrativo e derivativo. Alcuni termini possono essere annullati, ad esempio su sistemi controllati PI si ha un guadagno derivativo nullo per eliminarne l'effetto, così come si ha guadagno integrale nullo in un controllo PD ecc.

Applicabilità del controllo:

L'uso dell'algoritmo PID non garantisce un controllo ottimo del sistema o la sua stabilità, ci possono essere dunque situazioni dove ci sono ampi ritardi di risposta dovuti anche a ritardi precedenti nel loop come la misura della variabile di processo o a ritardi successivi di attuazione (nel caso della tesi questo potrebbe essere il tempo di comando valvole che gestiscono il cilindro pneumatico, la capacità di tubazioni e cilindro stesso che introducono costanti di tempo elevate ecc). Si può in certi casi fare una compensazione per ovviare a tale problema.
La risposta del controllo può essere descritta in termini di velocità di risposta all'errore, dall'overshoot al setpoint e al grado di oscillazione successivo.
Va ricordato che tuttavia il PID è un controllo che è ampiamente utilizzabile e che necessita solo dell'errore e della misura della variabile di processo, mentre non necessita di un modello matematico per funzionare.

Origini del PID:

Non si può stabilire una data precisa per la nascita del controllo PID in quanto è sempre stata un'evoluzione di sistemi atti a raggiungere un certo valore di setpoint, ma è nel 1922 che può essere individuato un primo controllo PID a tre termini sul controllo di direzione di una nave dell'esercito americano, qui si notò dall'esperienza del capitano che serviva anche la "storia" dell'errore, la sua variazione e l'errore istantaneo, per condurre l'imbarcazione nella giusta direzione.
L'ingegner Minorsky si accorse dunque della necessità di avere anche un termine integrativo e uno derivativo per raggiungere la stabilità del sistema.



Nell'immagine a destra si può notare uno dei primi controllori PID pneumatici, i valori dei guadagni sono impostabili tramite gli indicatori posti nella parte superiore.

Fig 39: PID pneumatico, primissimi esemplari

Successivamente, con l'avvento dell'elettronica, è stata possibile la realizzazione di PID fatti tramite amplificatori operazionali, il vantaggio è il basso costo e la facilità di costruzione, nell'immagine si vede che l'errore è trattato da un primo amplificatore operazionale (quello più a sinistra) che è invertente (il piedino del segnale è collegato all'ingresso) e le due resistenze effettuano uno "scaling" del valore di errore per il successivo blocco di regolazione tramite la legge:

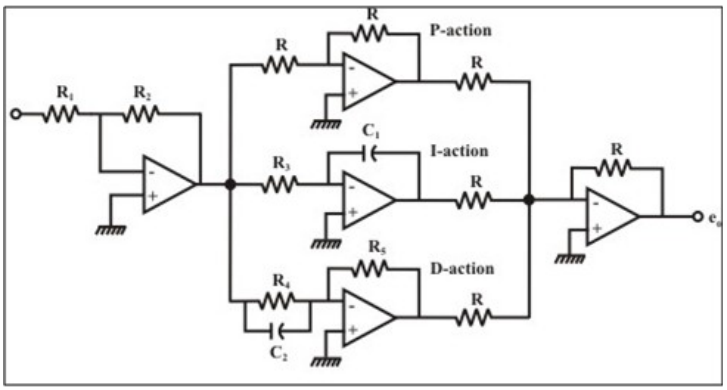


Fig 40: PID con amplificatori operazionali

$$V_{out} = -\left(\frac{R_2}{R_1}\right) \cdot V_i$$

questo è necessario per definire un range di ampiezza dell'errore stesso, successivamente i 3 amplificatori operazionali (proporzionale, integrativo e derivativo) trattano le rispettive componenti d'errore che verranno sommate e invertite nell'ultimo amplificatore operazionale che sarà l'uscita.

Teoria del controllo PID:

Come già anticipato prima, il PID ha 3 termini, i quali sommati formano l'uscita u(t).

$$u(t)=K_p\cdot e(t)+K_i\cdot\int_0^t e(t)\,dt+K_d\cdot\frac{de(t)}{dt}$$

dove

K_p è il guadagno proporzionale,

K_i è il guadagno integrale,

K_d è il guadagno derivativo,

$e(t)=SP-PV(t)$ è l'errore (SP è il setponint e PV(t) è la variabile di processo, ovvero il feedback)

nel dominio di Laplace, questo si traduce come $L(s)=K_p+\frac{K_i}{s}+K_d\cdot s$ dove con "s" è indicata la variabile di Laplace.

Termine proporzionale:

il termine proporzionale produce un valore di uscita che è proporzionale al valore corrente dell'errore. Il contributo proporzionale può essere regolato moltiplicandolo per il guadagno proporzionale K_p che è di fatto costante.

Si ha dunque $P_{out}=K_p\cdot e(t)$

un guadagno proporzionale elevato ha come risultato una grande variazione dell'uscita anche a seguito di una piccola variazione dell'errore. Se il guadagno proporzionale è troppo elevato, il sistema può diventare instabile, viceversa con un piccolo guadagno proporzionale si possono avere grandi variazioni di errore e piccole variazioni sull'uscita, che rendono il contributo poco sensibile all'errore. Se il guadagno proporzionale infatti è troppo piccolo, il sistema può risultare lento nel rispondere ai disturbi.

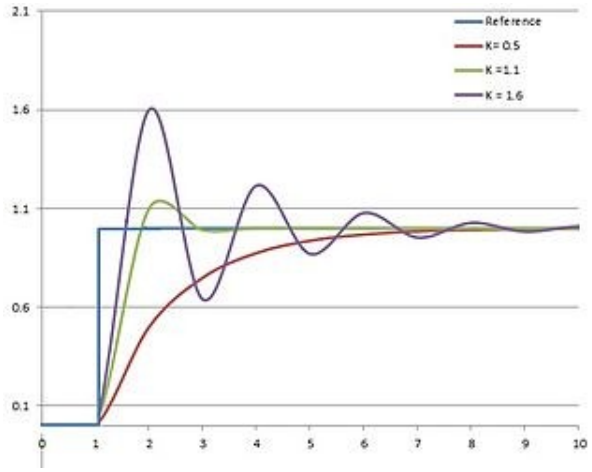


Fig 41: Variazione del termine proporzionale (P)

Termine integrale:

Il contributo della parte integrale può amplificare l'errore e la durata dello stesso, il termine integrale nel PID è la somma dell'errore istantaneo sul tempo e fornisce un offset accumulato che dovrebbe esser stato corretto precedentemente. L'errore accumulato è poi moltiplicato per il guadagno integrale e il tutto viene sommato all'output del controllo.

Ricapitolando: $L_{out}=K_i\cdot\int_o^t e(t)\,dt$

Il termine integrale serve per eliminare l'errore residuo a regime, che è generalmente presente con un controllo puramente proporzionale. Bisogna tenere

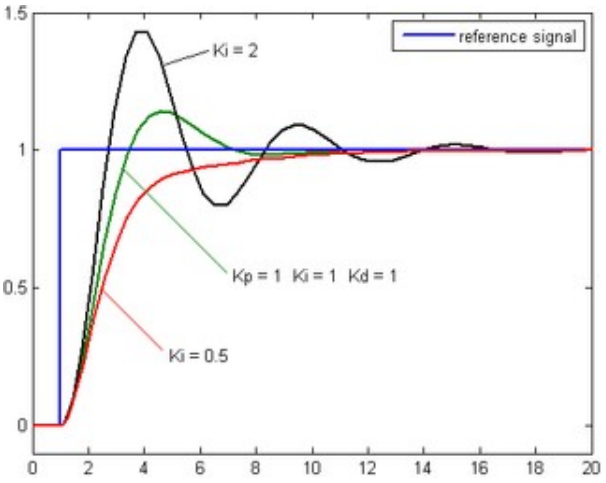


Fig 42: Variazione termine integrale (I)

conto che anche se il termine integrale conta il valore accumulato dell'errore dal passato, può causare degli overshoot attorno al valore di setpoint.

Termine derivativo:

Il termine derivativo è calcolato determinando la variazione dell'errore su un certo periodo, questa variazione viene infine moltiplicata per il guadagno derivativo k_d

$$D_{out} = K_d \cdot \frac{de(t)}{dt}$$

questo contributo predice il comportamento del sistema e incrementa la stabilità del sistema.

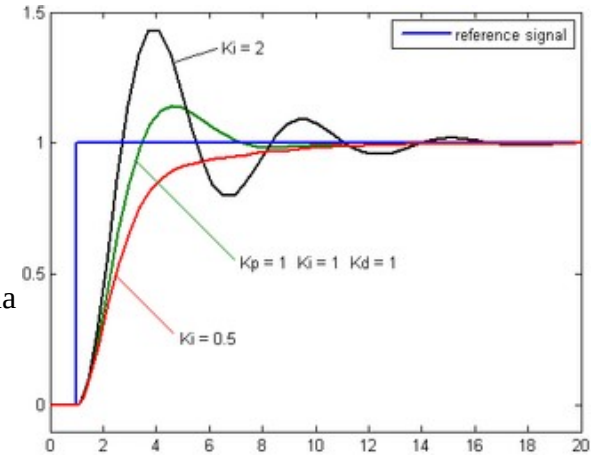


Fig 43: Variazione termine derivativo (D)

Loop tuning:

Con tuning si vuole indicare le modifiche ai parametri di controllo (guadagni del proporzionale, integrativo e derivativo) che portano ad avere un sistema con risposta ottimale. La stabilità è un requisito base, ma a seconda del sistema si possono avere diversi comportamenti e a volte i requisiti possono essere in conflitto.
Molti processi possono avere dei gradi di non linearità e può capitare che in certi campi, i parametri utilizzati vanno bene, mentre in altri campi questi valori di tuning sono errati e non mantengono la stabilità del sistema; questo può essere corretto facendo dei "gain scheduling" ovvero mappando differenti parametri in differenti regioni dove opera il sistema.

Stabilità:

Se i parametri del pid (Kp, Ki, Kd) non sono corretti, il sistema può essere instabile, molto spesso questa instabilità è causata da un guadagno eccessivo, particolarmente se il sistema presenta ritardi significativi.
In linea teorica il sistema deve arrivare al valore di set impostato il prima possibile e non oscillare (nel caso in tesi, il pendolo se perturbato dovrebbe recuperare sia la posizione centrale de carrello mantenendo così l'asta in posizione verticale).
Matematicamente, considerando un classico anello, si ha la seguente funzione di trasferimento

$H(s) = \frac{K(s) \cdot G(s)}{1 + K(s) \cdot G(s)}$ dove K(s) è la funzione di trasferimento del pid, mentre G(s) è la funzione di trasferimento della retroazione. Il sistema è instabile se la funzione di closed loop diverge, questo accade quando $K(s) \cdot G(s) = -1$ ma tipicamente questo succede quando $|K(s) \cdot G(s)| = 1$ con la fase a 180°.

Comportamento ottimale:
il comportamento ottimale può essere stabilito in due condizioni:

1. si imposta un set fisso e si disturba il sistema con un input esterno, il sistema deve tornare al set il più velocemente possibile e (idealmente) senza oscillare. Nella pratica si preferisce un sistema più veloce a raggiungere il set anche se sono presenti piccole oscillazioni attorno ad esso, che però si smorzano abbastanza velocemente piuttosto che avere un sistema che non oscilla per niente al set ma inevitabilmente rimane più lento nel raggiungerlo.
2. si imposta un set variabile e si nota quanto il sistema è veloce e stabile nel seguire il set.

Metodi di regolazione:

esistono diversi metodi per fare il tuning di un PID, il più utilizzato in generale implica lo sviluppo di modelli, successivamente si scelgono i valori del Kp, Ki e Kd. Tuttavia la via più pratica è fare un tuning manuale vedendo come reagisce il sistema, ma questo può portare a tempi di settaggio lunghi. La scelta del metodo è influenzata anche da quanto il sistema è complesso.

Manual tuning:

Se il sistema è settabile “on line” (come nel caso del pendolo, i valori possono essere cambiati quando il sistema è in modalità “run” e si può apprezzare visivamente questi cambiamenti) si inizia settando Ki e Kd con valori nulli, successivamente si incrementa Kp.
Nell’immagine si nota un Kp troppo basso, non oscilla e ma non si avvicina neanche al set.
Kp viene aumentato fino a quando l’uscita del sistema inizia ad oscillare.

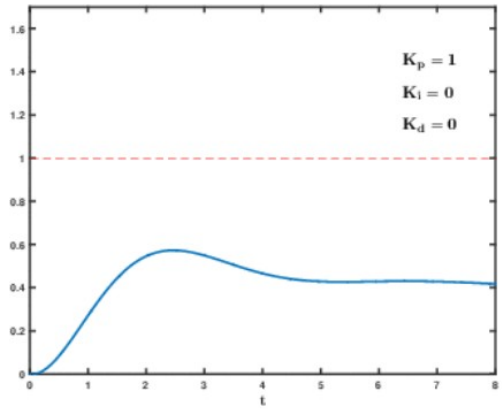


Fig 44: Inizio del tuning

Il valore del Kp dovrebbe essere impostato all’incirca alla metà di questo valore, per avere una “quarter amplitude decay type response”.
In pratica il rapporto tra il valore del secondo overshoot (A2) e del primo overshoot (A1) dev’essere più piccolo di ¼.

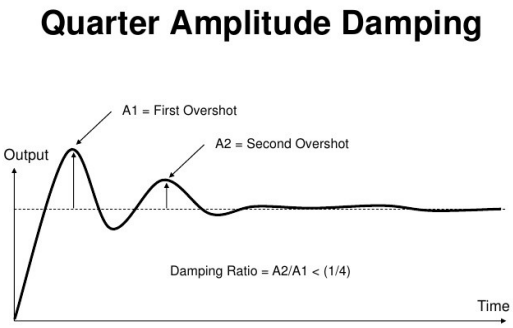


Fig 45: Risposta 1/4 di ampiezza

Aumentando il Kp si arriva al set, il sistema oscilla un po’ ma rimane l’errore residuo rispetto al set impostato.

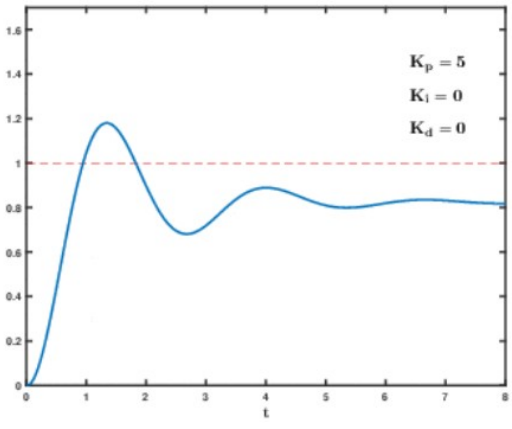


Fig 46: Aumento Kp

Dopodiché viene incrementato il valore di Ki fino a quando l'offset è corretto in un tempo sufficiente per il processo, va ricordato tuttavia che un valore di Ki troppo alto causa instabilità.

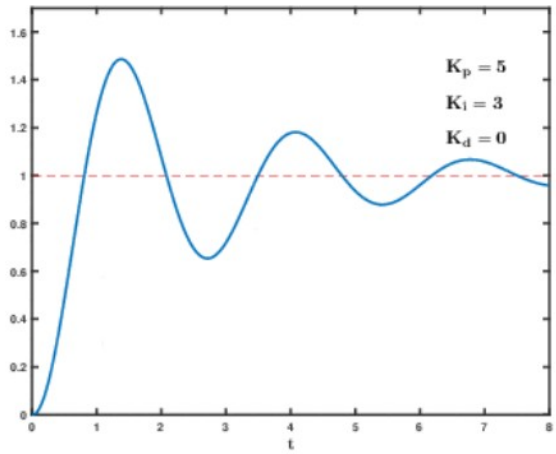


Fig 47: Aumento Ki

Infine si incrementa il valore di Kd, se richiesto, fino a quando l'anello è abbastanza veloce a raggiungere il valore di riferimento dopo esser stato disturbato dall'esterno. Il valore di Kd, se troppo elevato, causa una risposta eccessiva/overshoot.

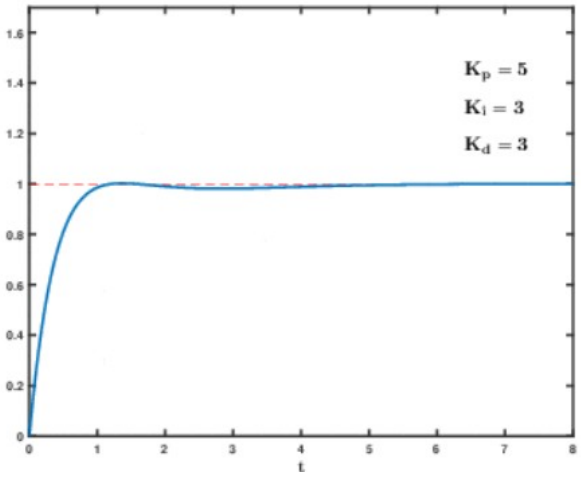


Fig 48: Aumento Kd

Un anello PID veloce non oscilla eccessivamente quando raggiunge il setpoint, tuttavia accettare il livello di oscillazione attorno al setpoint è anche in funzione del sistema che si sta considerando. Se invece il sistema non accetta overshoot, risulta quindi una risposta sovrasmorzata che necessita un valore di Kp più piccolo del valore che precedentemente causava l'oscillazione.

Effetto dell'incremento dei parametri

Parametro	**Tempo di salita**	**Overshoot**	**Errore a regime**	**Stabilità**
Kp	Diminuisce	Aumenta	Poca variazione	Diminuisce
Ki	Diminuisce	Aumenta	Eliminato	Diminuisce
Kd	Poca variazione	Diminuisce	Nessun effetto (in teoria)	Aumenta se Kd è piccolo

Metodo di Ziegler-Nichols:

Questo metodo è stato introdotto negli anni ‘40, come per il metodo precedente si inizia impostando il guadagno integrale e il guadagno derivativo a zero, successivamente il guadagno proporzionale viene incrementato fino al “guadagno ultimo” K_u , raggiunto il quale l’uscita del sistema inizia ad oscillare. Questo guadagno viene associato al suo periodo T_u e i due valori saranno utilizzati come segue:

Controllo	Kp	Ki	Kd
P	$0{,}50 \cdot K_u$	/	/
PI	$0{,}45 \cdot K_u$	$0{,}54 \cdot \frac{K_u}{T_u}$	/
PID	$0{,}60 \cdot K_u$	$1{,}2 \cdot \frac{K_u}{T_u}$	$\frac{3}{40} \cdot K_u \cdot T_u$

Quando applicati ad un PID standard, i parametri temporali T_i e T_d sono dipendenti unicamente dal periodo di oscillazione T_u .
Generalmente si può scrivere secondo Laplace che l’uscita è

$$L(s) = K_p + \frac{K_i}{s} + K_d \cdot s = K_p \cdot \left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right)$$

avendo dunque $T_i = \frac{K_p}{K_i}$ e $T_d = \frac{K_d}{K_p}$.
Nel dominio del tempo l’uscita può essere scritta come:

$$u(t) = K_p \cdot [e(t) + \frac{1}{T_i} \cdot \int_o^t e(t) \cdot dt + T_d \cdot \frac{de(t)}{dt}]$$

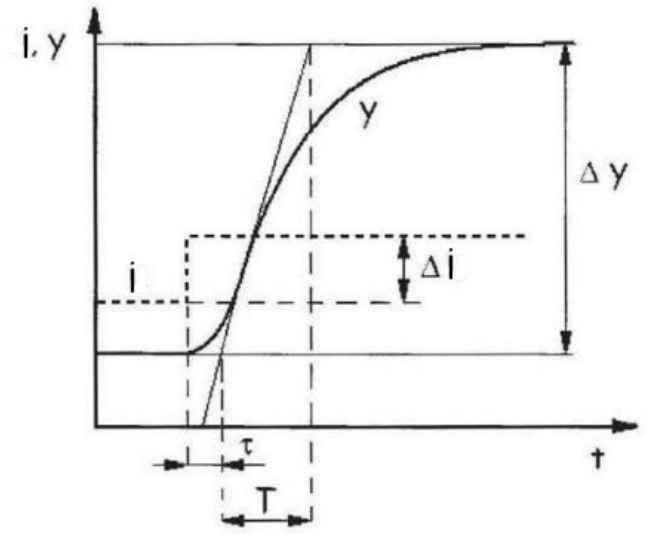
L’applicazione del metodo è anche in funzione del sistema considerato, funziona molto bene nel respingere i disturbi; tuttavia avendo overshoot può essere inappropriato per certi sistemi.

Metodo di Ziegler-Nichols in anello aperto:

In questo metodo, come si intuisce dal nome, il sistema è in anello aperto.

- viene applicato al sistema un gradino di intensità Δi
- si traccia la tangente del transitorio nel punto di flesso, determinando la costante di tempo T, il guadagno statico $K = \frac{\Delta y}{\Delta i}$ e il ritardo τ , dove Δy è la variazione dell’uscita
- con i parametri ricavati si usa la tabella per determinare il guadagno proporzionale e delle costanti di tempo nei vari casi del controllo.

Il parametro R che si trova in tabella corrisponde a $R = \frac{K}{T}$.
Questo metodo è semplice da utilizzare ma non è applicabile a quei sistemi dove l’imposizione di un gradino porta all’instabilità.
Il rapporto ritardo su costante di tempo $\frac{\tau}{T}$ influenza molto i parametri di controllo, bisogna dunque dedicare particolare attenzione alla rilevazione del ritardo. Ovviamente, affinché il sistema sia in campo lineare, il gradino applicato all’ingresso non deve essere né troppo piccolo né troppo grande.



	Kp	Ti	Td
P	$\frac{1}{\tau \cdot R}$	/	/
PI	$\frac{0,9}{\tau \cdot R}$	$3 \cdot \tau$	/
PID	$\frac{1,2}{\tau \cdot R}$	$2 \cdot \tau$	$0,5 \cdot \tau$

Fig 49: Grandezze del metodo Ziegler-Nichols

Metodo “Relay”:

Pubblicato nel 1984 da Karl Astrom e Tore Hagglund, il metodo utilizza, secondo la terminologia inglese il “bang-bang control”, misurando l’oscillazione risultante dell’output.
L’uscita è settata tra due valori della variabile controllata, che sarebbe l’uscita del PID, (da questo prende il nome il metodo, appunto come uno switch in due posizioni), questi valori devono essere opportunamente scelti in modo tale che il processo raggiunga il setpoint e lo attraversi.
Non è necessario tuttavia che i valori siano lo 0 e il 100%, facendo una scelta accurata si possono evitare pericolose oscillazioni del sistema stesso.
Quando la variabile di processo è sotto il setpoint, l’uscita è settata al valore alto; quando la variabile di processo supera il setpoint, l’uscita è portata al valore basso il più velocemente possibile. Idealmente, la forma dell’output è un’onda quadra.
Il periodo e l’ampiezza dell’oscillazione risultante sono misurate, e utilizzate per calcolare il guadagno e il periodo che sono inseriti nel metodo di Ziegler-Nichols.
In modo particolare, il periodo ultimo T_u , è assunto essere uguale al periodo osservato, e il guadagno ultimo si può ricavare come $K_u = \frac{4 \cdot b}{\pi \cdot a}$ dove il termine “a” è l’ampiezza dell’oscillazione della variabile di processo, mentre il termine “b” è l’ampiezza dell’uscita del controllo che l’ha causata.
Questo metodo tuttavia ha molte varianti.

Software per il tuning:
Quando possibile non si usano i metodi manuali descritti in precedenza, ma si usano software capaci di fornire risultati consistenti in minor tempo. Questi pacchetti software raccolgono i dati conoscendo un certo modello di sistema e suggeriscono secondo calcoli il tuning ottimale.
Matematicamente il tuning dell’anello PID introduce un impulso nel sistema, successivamente usa la risposta del sistema per ottenere i valori del PID. In sistemi dove il tempo di risposta è dell’ordine dei minuti, questa procedura è raccomandata, perché manualmente l’operazione potrebbe richiedere giorni per trovare i valori PID ai quali il sistema è stabile.
Come esempio nel caso in tesi, il software Allen-Bradley, contiene un PID auto tuning se il programma viene scritto in FBD.
Anche Arduino ha delle librerie che permettono l’auto tuning dei parametri.

Limitazioni del controllo PID:

Il controllo PID può essere applicabile in molte situazioni, in molte funziona bene ma in altre potrebbe avere prestazioni non soddisfacenti. La difficoltà principale con un controllo PID semplice è di avere un sistema controllato con un feedback con parametri costanti, non si ha dunque la conoscenza diretta del processo che comporta ritardi compromettendo in certi casi le prestazioni.
I controllori PID, quando usati da soli, possono portare a prestazioni scarse se i guadagni sono ridotti in modo tale che il sistema non ha overshoot e oscillazioni attorno al setpoint. I PID hanno difficoltà con la presenza di non linearità del sistema, perché rispondono sempre allo stesso modo.
Un miglioramento può essere fatto attraverso il feed-forward con la conoscenza del sistema, usando il PID per controllare solo l'errore.
I PID possono essere usati anche facendo dei "gain scheduling" cioè usando differenti parametri in base alla zona di lavoro e alle prestazioni.
Un controllo PID può essere migliorato anche agendo sulla lettura dei valori, sul tempo di campionamento, aumentando precisione e accuratezza o utilizzando più controlli PID in cascata.

Feed-Forward:

Con il feed-forward il disturbo è misurato e conteggiato prima che vada ad influenzare il sistema. La difficoltà è quella di misurare il disturbo, soprattutto nel caso di molteplici disturbi che possono influenzare il sistema.
Ci sono 3 tipi di controllo:

(a) controllo del sistema in anello aperto
(b) controllo del sistema tramite feed-forward
(c) controllo del sistema tramite feedback

Nel primo caso (a) si imposta un input, e tramite una relazione viene determinato un output che verrà modificato dal disturbo. In questo caso non si ha nessun controllo sul disturbo che agisce direttamente sul sistema.

Nel secondo caso (b) il disturbo viene letto dal feed-forward e viene sommato all'input tramite un nodo, in questo modo l'output del sistema può variare, in maniera programmata per agire contro il disturbo, anche se l'input fornito è fisso, si ha dunque controllo su cosa succede.

Il caso (c) è il classico sistema con retroazione dove si controlla l'uscita che viene successivamente comparata con l'input fornito.

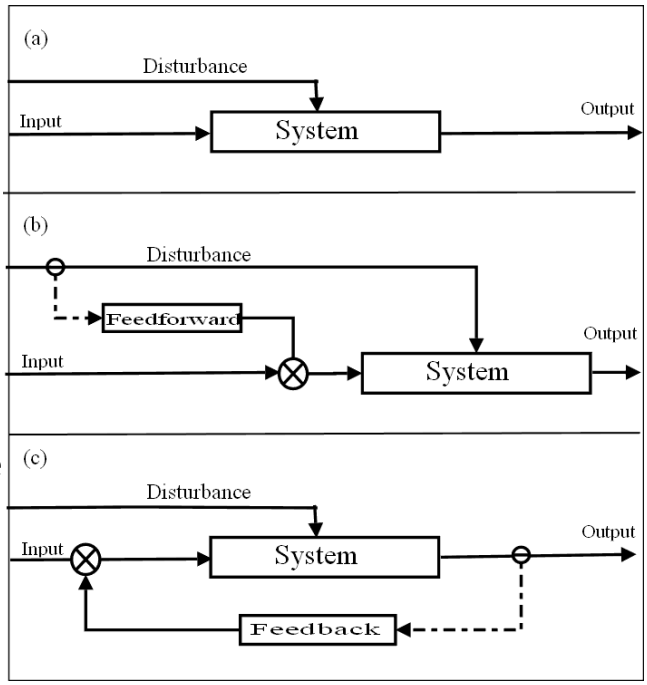


Fig 50: Vari tipi di controllo

I benefici del feed-forward possono essere tali da giustificarne i costi/tempi per implementarlo:

- accuratezza migliorata
- consumo energetico minore rispetto ad altri controlli
- minori costi di manutenzione rispetto altri controlli
- elevata affidabilità
- riduzione isteresi

Windup dell’integrale:

Come visto in precedenza, il termine integrale integra l’errore nel tempo, seguendo questo comportamento il termine può accumulare eccessivamente portando l’uscita del PID oltre il massimo valore in uscita (windup). Ponendo il caso che il regolatore sia capace di fornire il 100% in uscita, il termine integrale accumula superando il 100% e continua ad accumulare fino a quando l’errore ha lo stesso segno. Con errore di segno opposto l’integrale inizia a diminuire, iniziando a sottrarre dal valore precedentemente raggiunto. Se questo valore era oltre il 100% ad esempio, anche con errore negativo, ci vuole un certo tempo prima che l’uscita del regolatore diminuisca al di sotto del 100%. Ovviamente questo non è accettabile, si può ricorrere a limitazioni dell’uscita fornita dall’integrale che si chiamano appunto Anti-Windup.
Si può dunque:

- prevenire l’accumulo dell’integrale al di sopra o al di sotto di predeterminate bande;
- disabilitare l’integrazione fino a quando la variabile di processo (feedback) non rientra in una regione controllabile, questo evita di avere errori troppo elevati che farebbero accumulare velocemente l’errore;
- calcolare con segno opposto l’errore nel termine integrale costringendo l’uscita del regolatore in una zona interna alle sue capacità;

PID in cascata:

Può essere vantaggioso mettere due PID in cascata per avere migliori prestazioni.
Si pensi al caso in tesi del pendolo, ci sono due PID rispettivamente per la posizione del carrello e per la rotazione dell’asta.
Questo controllo in cascata funziona sulla base di settare l’input del secondo PID con l’output dato dal primo PID. Si crea così un anello interno ed uno esterno. Il setpoint del secondo PID sarà dunque variabile in funzione del setpoint del primo PID e dal suo feedback.
In maniera del tutto generica, si ha lo schema seguente:

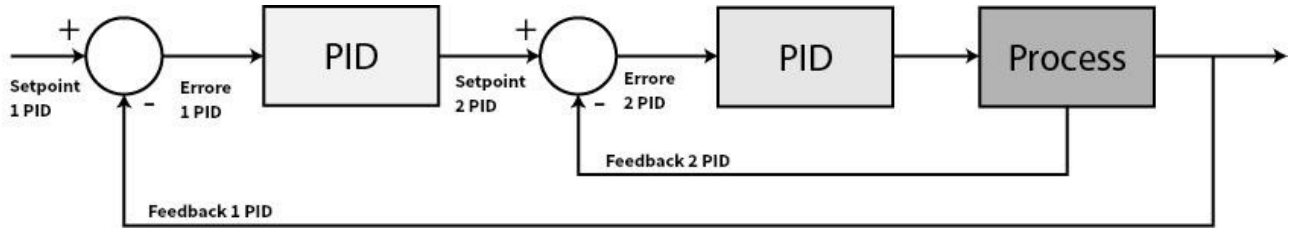


Fig 51: PID in cascata

Da notare che nel caso del pendolo inverso, il setpoint è la posizione, ma l’uscita del regolatore complessivo dei due PID, deve essere un angolo, questo schema sarà analizzato nel dettaglio successivamente.

I guadagni del termine proporzionale, integrativo e derivativo, possono essere molto differenti tra i due PID, soprattutto alcuni termini possono essere anche nulli (ad esempio nel PID posizione del pendolo, esiste solo la parte proporzionale, che per altro è molto piccola).

Forma di Laplace per i PID:

Con Laplace il PID può essere rappresentato come: $G(s)=K_p+\frac{K_i}{s}+K_d\cdot s=\frac{K_d\cdot s^2+K_p\cdot s+K_i}{s}$

Conoscendo la forma di Laplace e avendo la funzione di trasferimento del sistema controllato, risulta facile determinale la funzione di closed-loop del sistema.

Sviluppando:

$$G(s)=K_p+\frac{K_i}{s}+K_d\cdot s=\frac{K_d\cdot s^2+K_p\cdot s+K_i}{s}=K_d\frac{s^2+\frac{K_p}{K_d}\cdot s+\frac{K_i}{K_d}}{s}$$

Si definisce:

$$H(s)=\frac{1}{s^2+2\cdot\zeta\cdot\omega_0\cdot s+\omega_0^{\,2}} \quad \text{dove} \quad \frac{K_p}{K_d}=2\cdot\zeta\cdot\omega_0$$

e assemblando i termini: $G(s)\cdot H(s)=\frac{K_d}{s}$

Generalmente appare utile rimuovere i poli instabili, ma non è questo il caso. La funzione di trasferimento di closed-loop continua ad avere poli instabili.

Schema a blocchi di controllo del banco:

Nel controllo viene impostata una posizione che il carrello deve mantenere, ma si vuole che il sistema pendolo inverso resti sempre in equilibrio. È necessario dunque avere due PID, dunque due anelli, rispettivamente per la posizione e per la rotazione.
Come si può vedere dall'immagine successiva, solo il PID Theta è il responsabile dell'attivazione delle valvole, in quanto il PID X fornisce solo il SET per il PID Theta.
Le valvole, in funzione del PWM fornitogli, inviano un flusso d'aria che ha il compito, attraverso il cilindro pneumatico, di far muovere il carrello.
Il blocco pendolo lega la posizione del carrello con l'inclinazione dell'asta.

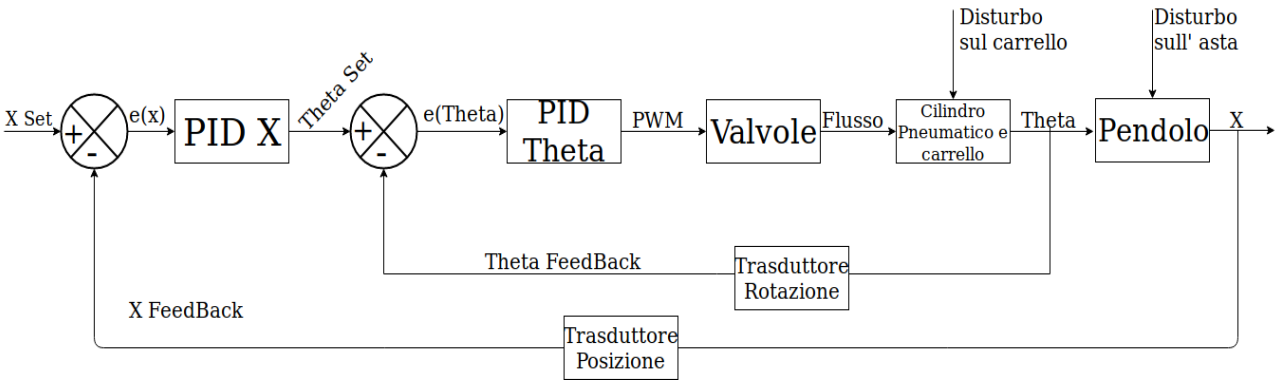


Fig 52: Struttura degli anelli PID per il controllo del banco

9. FUZZY

Un sistema controllato con la logica Fuzzy è un sistema matematico che analizza gli ingressi analogici in termini di variabili logiche che variano tra 0 e 1, ma in questo caso la logica opera con valori continui, d'altronde lo stesso termine "fuzzy" indica la "sfumatura" che può essere assunta tra questi due valori.
Questa logica è ampiamente usata nel controllo di macchine, il vantaggio è che la logica consente un controllo con variabili che sono parzialmente vere (1 è considerato completamente vero) e parzialmente false (0 è considerato falso).
La logica Fuzzy è abbastanza recente, proprio nell'anno 1987 Takeshi Yamakawa ha dimostrato il funzionamento proprio tramite un esperimento con il pendolo inverso, il quale è un classico problema di controllo.
Le variabili in ingresso al sistema sono mappate generando delle funzioni conosciute come "fuzzy sets" e il processo di conversione in valore "crisp" viene chiamato "fuzzificazione".
Facendo una mappa si può dunque comunicare al microcontrollore quali sono le decisioni da eseguire.
I risultati verranno combinati successivamente con valori crisp per la fase di "defuzzificazione" e verranno attivate proporzionalmente le uscite.
I controlli tradizionali, a differenza del Fuzzy, usano modelli matematici basati su equazioni differenziali (il PID ne è un esempio) e questo funziona quando il modello matematico esiste o non è troppo dispendioso dal punto di vista della potenza di calcolo e memoria, ed è proprio in queste situazioni che un controllo Fuzzy può risultare vantaggioso e meno oneroso se usato al posto di un controllo tradizionale.

Controllo Fuzzy in dettaglio:

Concettualmente è un controllo semplice, c'è uno stadio di ingresso, uno di processo e uno di uscita.

- Lo stadio di ingresso legge i valori degli input quali sensori o altri dispositivi e li mappa secondo la funzione di appartenenza;
- Lo stadio di processo invoca le regole e genera un risultato per ognuna, in seguito combina i risultati delle regole;
- Infine, lo stadio di uscita, converte i risultati in uno specifico valore.

Le funzioni di appartenenza sono generalmente triangolari, ma possono essere anche trapezoidali o curve a campana, la forma comunque è meno rilevante rispetto al numero e al posizionamento di queste ultime.
Usualmente si utilizzano dalle 3 alle 7 funzioni e la forma triangolare è quella più utilizzata.

Un esempio su un termostato:

```
IF (temperature is "cold") THEN (heater is "high")
```

questa è una regola che usa il valore di "temperature" come input e genera un risultato in Fuzzy set per "heater" e verrà utilizzata con i risultati delle altre regole per generare il composito crisp di uscita. Ovviamente più grande è il valore di "cold" tanto più alto sarà il valore di "high" del riscaldatore, dunque c'è una sfumatura nella regolazione del sistema tra valori continui.

Metodo del centro di massa:

È il metodo più usato, il baricentro del risultato fornisce il “crisp value”

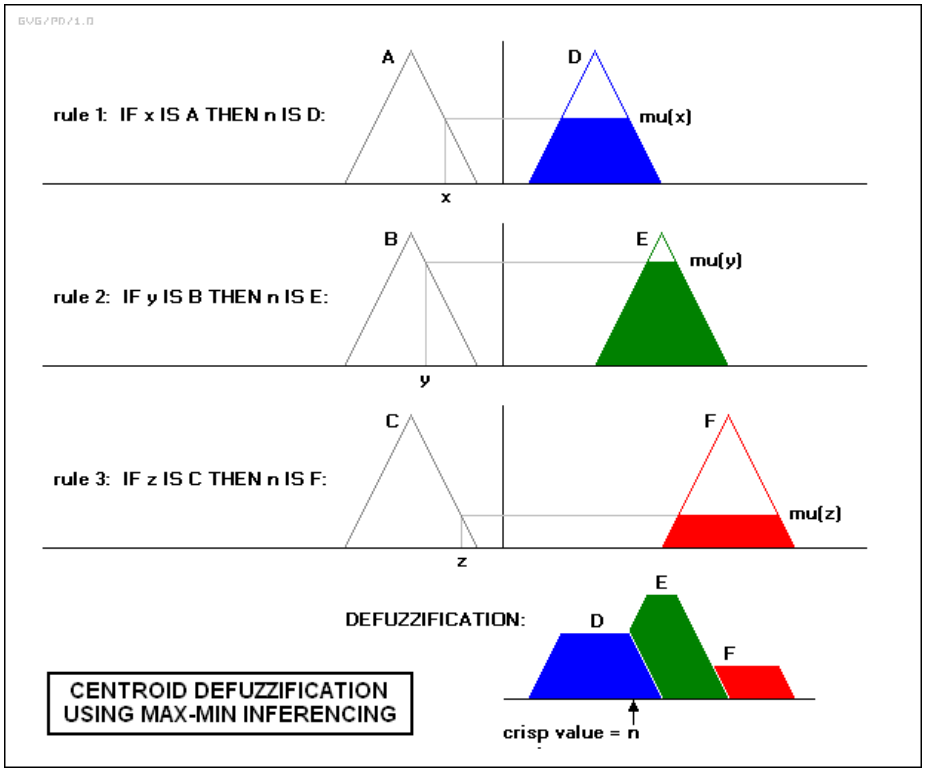


Fig 53: Metodo baricentro

Come si può notare, in questo esempio ci sono 3 regole che vengono associate ed ognuna contribuisce con la sua parte al risultato finale che si ottiene dalla defuzzificazione.
In pratica ogni regola fornisce un risultato come valore vero ad una particolare funzione di appartenenza.

Un altro esempio può essere quello di un controllo di una turbina a vapore:

dove i valori della variabile di uscita sono:

```
N3:   Large negative.
N2:   Medium negative.
N1:   Small negative.
Z:    Zero.
P1:   Small positive.
P2:   Medium positive.
P3:   Large positive.
```

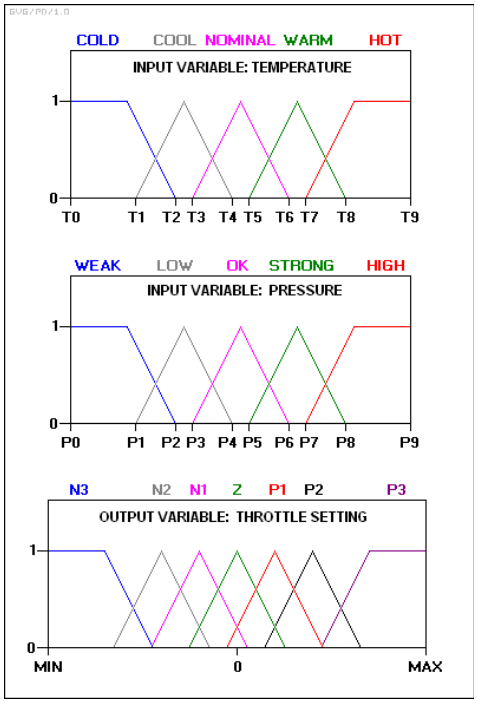


Fig 54: Input e output Fuzzy

E il set di regole è:

```
rule 1:  IF temperature IS cool AND pressure IS weak,
         THEN throttle is P3.

rule 2:  IF temperature IS cool AND pressure IS low,
         THEN throttle is P2.

rule 3:  IF temperature IS cool AND pressure IS ok,
         THEN throttle is Z.

rule 4:  IF temperature IS cool AND pressure IS strong,
         THEN throttle is N2.
```

In pratica il controllo riceve gli ingressi e li mappa in funzioni di appartenenza tramite il valore vero, questa mappatura è inserita successivamente nelle regole. Se la mappatura contiene una relazione “AND” tra ingressi come nell’esempio, il minimo dei due è usato come valore vero combinato, viceversa se viene utilizzata una relazione “OR” viene utilizzato il massimo valore. Il valore vero viene infine defuzzificato:

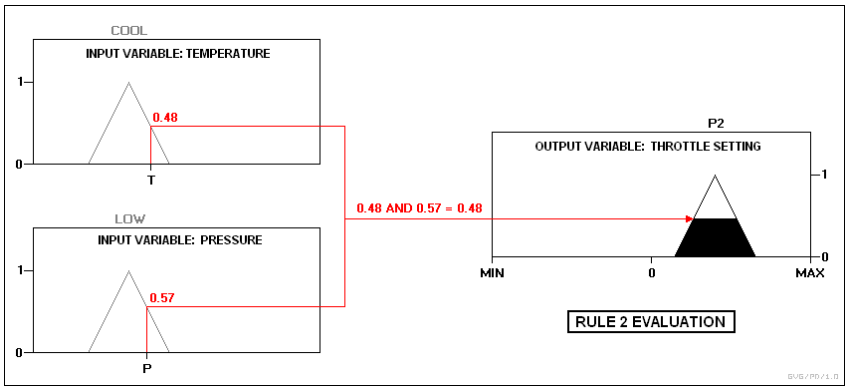


Fig 55: Valutazione regola 2

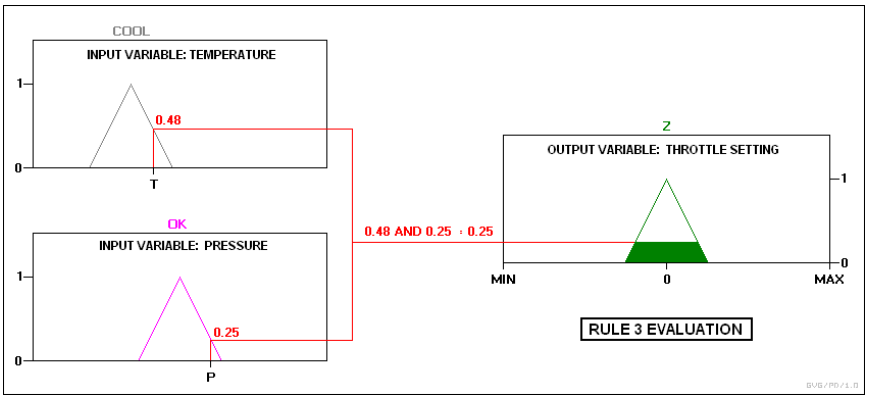


Fig 56: Valutazione regola 3

i due output vengono infine defuzzificati tramite il metodo del centro di massa ottenendo

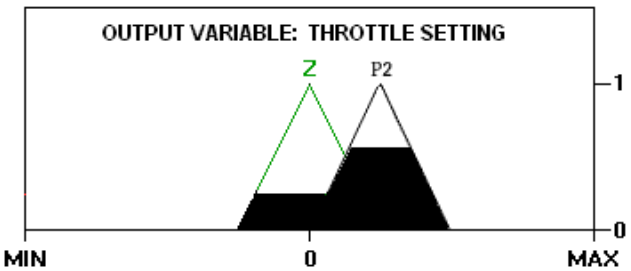


Fig 57: Variabile di output

10. Altri metodi di programmazione e confronto con differenti sistemi

Il problema del pendolo inverso è stato affrontato molte volte e con differenti sistemi, il web è pieno di esempi dove vengono usati, oltre ai PLC, anche schede elettroniche capaci di lavorare direttamente col sistema.
La soluzione più diffusa è con Arduino, una piccola scheda elettronica avente ingressi digitali/analogici e uscite digitali/PWM.

Fig 58: Arduino UNO

Questa scheda può essere programmata attraverso un linguaggio simile al C, con la differenza di alcune istruzioni particolari e della struttura, per Arduino infatti può essere usato solo una programmazione a testo strutturato:

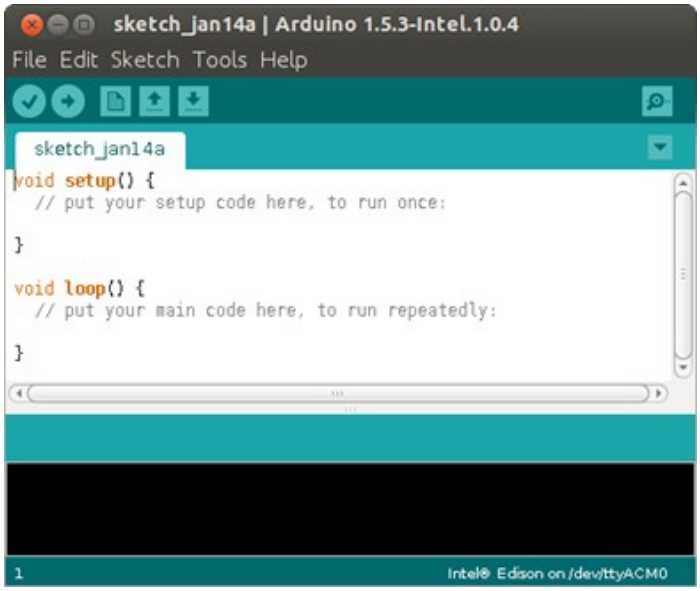


Fig 59: IDE Arduino

Di seguito si nota la realizzazione di questo “Robot” su logica del pendolo inverso, sostanzialmente il controllo agisce sui motori ricevendo il feedback da un giroscopio montato sulla struttura

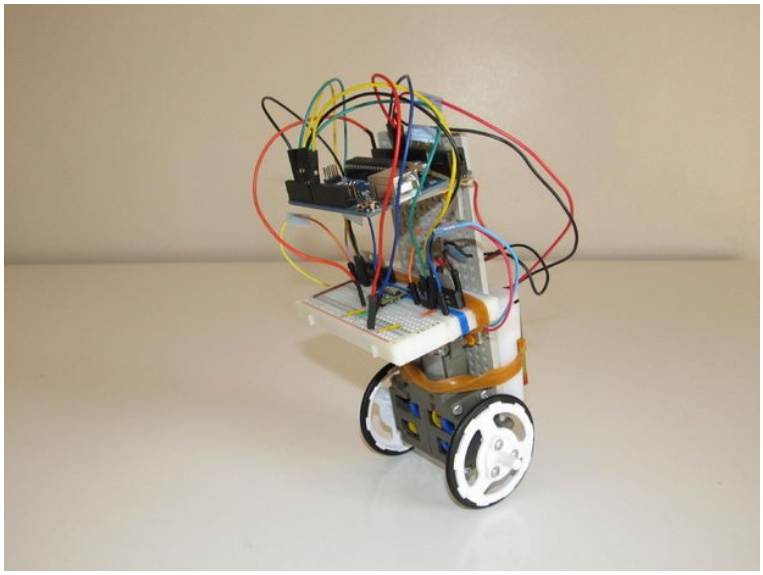


Fig 61: "Robot"

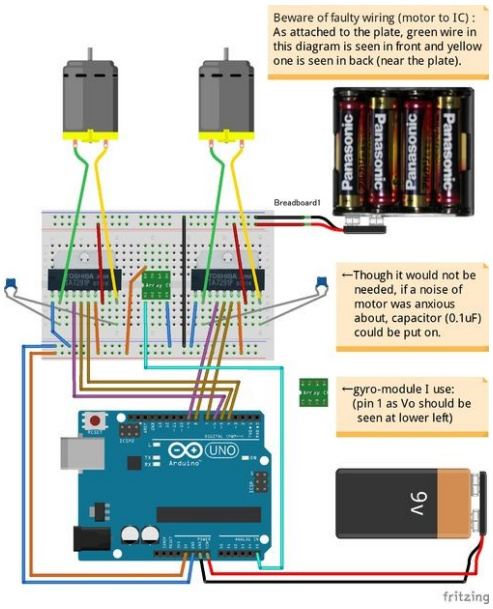


Fig 60: Componenti "Robot"

Questo è un esempio “low cost” di un sistema a pendolo inverso dove viene controllata solo l’inclinazione e la sua derivata, non ci sono infatti vincoli o guide come nel caso del banco in esame.

Questo sistema può essere applicato ad altri oggetti di uso più comune, almeno negli ultimi anni, come il Segway:

La logica di funzionamento è la stessa di quella usata per il pendolo inverso, il sistema tende sempre a mantenere un angolo nullo verticale, per far si che il sistema sia stabile.

Il veicolo viene fatto avanzare spostandolo dalla sua condizione di equilibrio, dunque portandosi in avanti con il peso, un giroscopio rileva la nuova inclinazione e invia un segnale al sistema di controllo, il quale attraverso dei driver attiverà i motori per accelerare facendo tornare il sistema in equilibrio seppur con velocità non nulla.

Quando il sistema è in equilibrio la condizione di velocità viene preservata (non ci sono dunque accelerazioni) e il veicolo procede nella sua marcia.

Fig 62: Segway

Per arrestare il moto bisogna squilibrare nuovamente il sistema, ma questa volta nel verso opposto, per ottenere una accelerazione negativa.

11. Programma Rockwell nel dettaglio

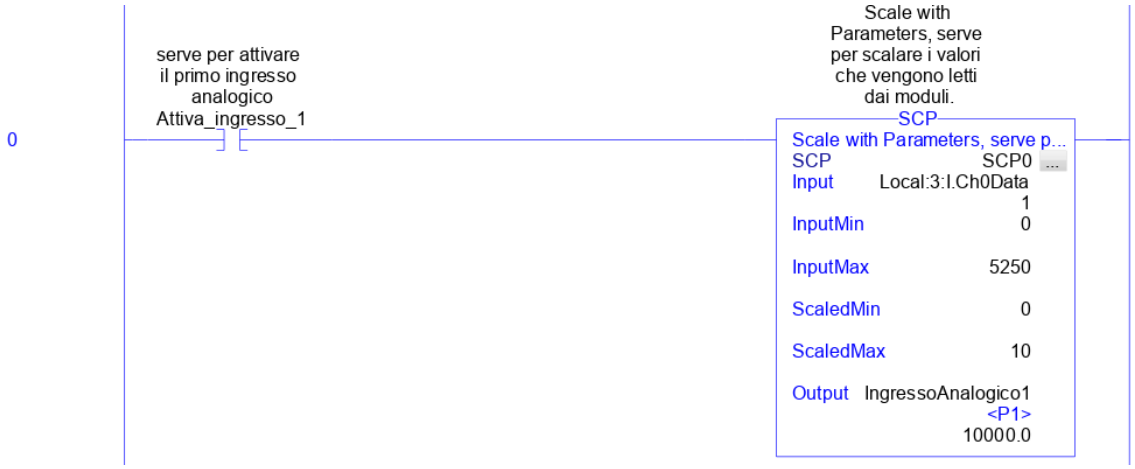
Di seguito viene illustrato il programma scritto tramite il software Logix5000 per il funzionamento del banco. Verrà analizzato ogni singolo rung del ladder ed ogni singolo blocco per facilitare la comprensione del programma stesso.

Il programma ha la funzione di acquisire i parametri, fare le dovute conversioni tra tensioni e valori in bit, introdurre questi valori all'interno dei PID e attivare successivamente le uscite in modo proporzionale.

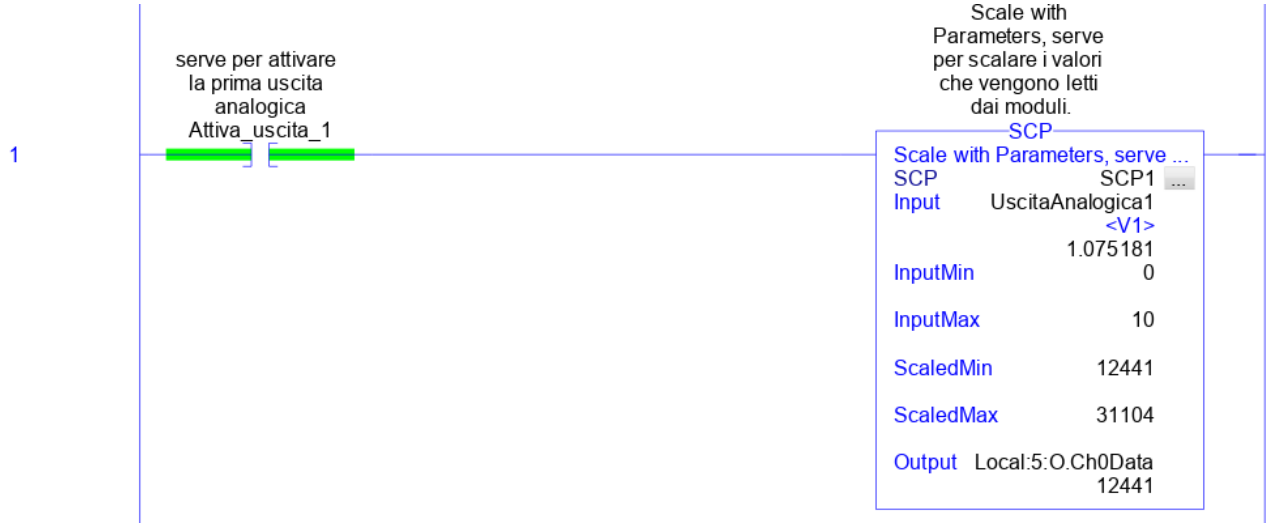
Analizziamo i rung partendo dal primo, in questo caso il rung 0:
come si vede esiste un bit di attivazione chiamato "Attiva_ingresso_1" che va ad attivare il blocco successivo SCP.
Il blocco SCP (Scale with Parameters) serve per convertire il valore di tensione acquisito sul modulo 3, canale 0, in un valore di bit che verranno successivamente utilizzati per effettuare il calcolo. Come si nota le voci del modulo SCP sono le seguenti:

- SCP: nome del blocco SCP, in questo caso è chiamato SCP0;
- Input: individua il modulo e il canale al quale è applicato il segnale, in questo caso modulo 3 Ch 0, si può notare la dicitura Local:3:i.Ch0Data, vuol dire che il modulo 3 è un modulo di input (i) al quale si fa riferimento al canale d'ingresso 0.
- InputMin: valore minimo in bit ricevuti dal modulo, ad una tensione nulla corrisponde il valore in bit=0;
- inputMax: definito nelle impostazioni il valore massimo di tensione per il dato ingresso, ad esempio 5V, si definisce il valore in bit massimo corrispondente a quella tensione. Questo valore si trova sulle tabelle dei rispettivi moduli nel manuale Allen Bradley.
- ScaledMin: Gli "scaled" in questo caso sono valori non esistenti fisicamente, ma comodi per fare i calcoli dopo aver fatto le conversioni, questo valore è come se fosse un valore di tensione nullo (0V) associato ad InputMin (0).
- ScaledMax: valore scalato associato ad InputMax, impostato come 10V.
- Output: è la variabile che viene creata, in questo caso chiamata IngressoAnalogico1 e definita tramite l'alias <P1>

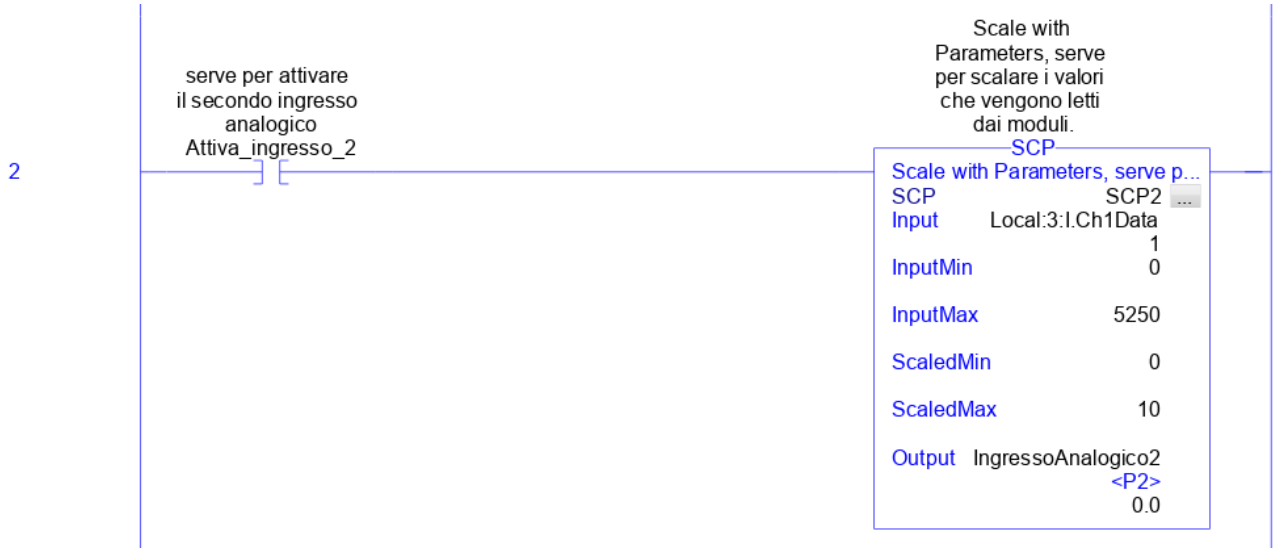
Nota bene: gli ingressi 1 e 2 che verranno spiegati, non sono tuttavia usati, in quanto erano stati pensati come input di pressione per due sensori di pressione piazzati ai capi del cilindro pneumatico. Non essendo stata usata questa informazione, non ci saranno ulteriori rung con calcoli di P1 e P2.



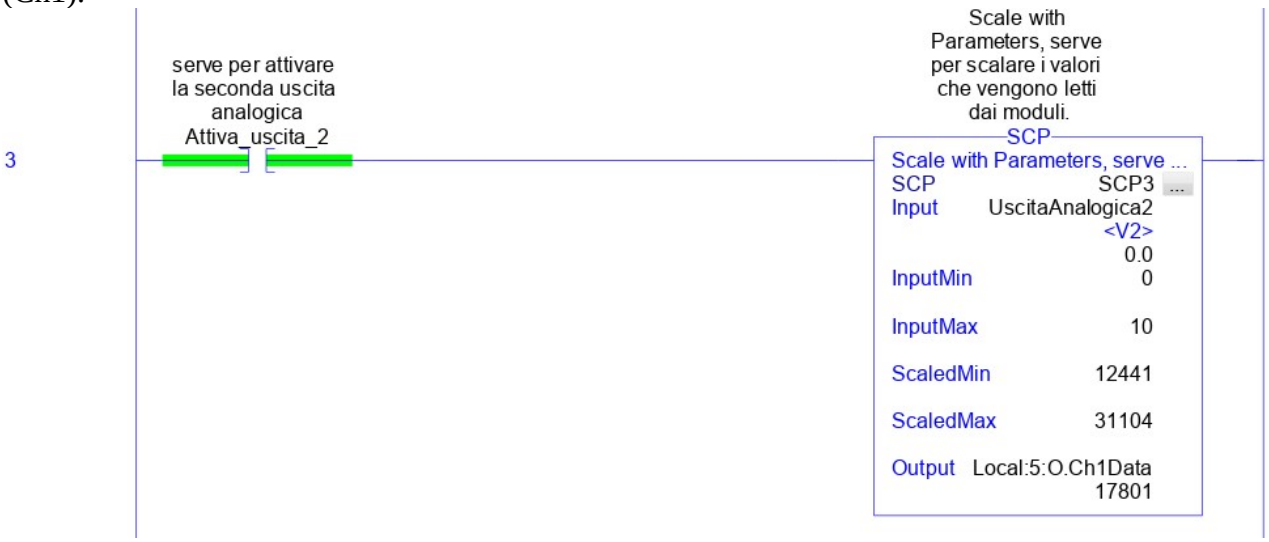
Per quanto riguarda il secondo rung, chiamato **rung 1:**
Qui si può notare un contatto, bit “Attiva_uscita_1”, che attiva il modulo scp posto a valle, questo è un SCP che ha come ingresso una variabile chiamata “UscitaAnalogica1”, anche associata all’alias “V1” e come output è connesso fisicamente il modulo 5, canale 0.
notare nella dicitura dell’output che Local:5:O.Ch0Data identifica un modulo (5 nel caso) di output (lettera O nella sigla) dove la valvola V1 è collegata al canale 0 (Ch0).
In questo blocco si trasformano dei valori di tensione tra 0 e 10V, che sono il risultato di calcolo del PID, in valori in bit tra 12441 e 31104.
Dal manuale si vede che per avere in uscita una tensione di 10V, per il modulo 5, occorre impostare come ScaledMax il valore di 31104, che equivalgono a 10V fisici rilevabili.
Visto che le valvole hanno una banda morta, si preferisce considerare questo fattore facendo lavorare il modulo in un range 4-10V.
Infatti calcolando 0.4*31104=12441 circa.
L’ouput, ovvero la tensione che verrà inviata ai driver delle valvole, dunque sarà compreso tra 4 e 10V, cioè tra 12441 e 31104 bit dei valori scalati. Questo si fa analogamente con gli altri blocchi SCP che comanderanno le valvole.



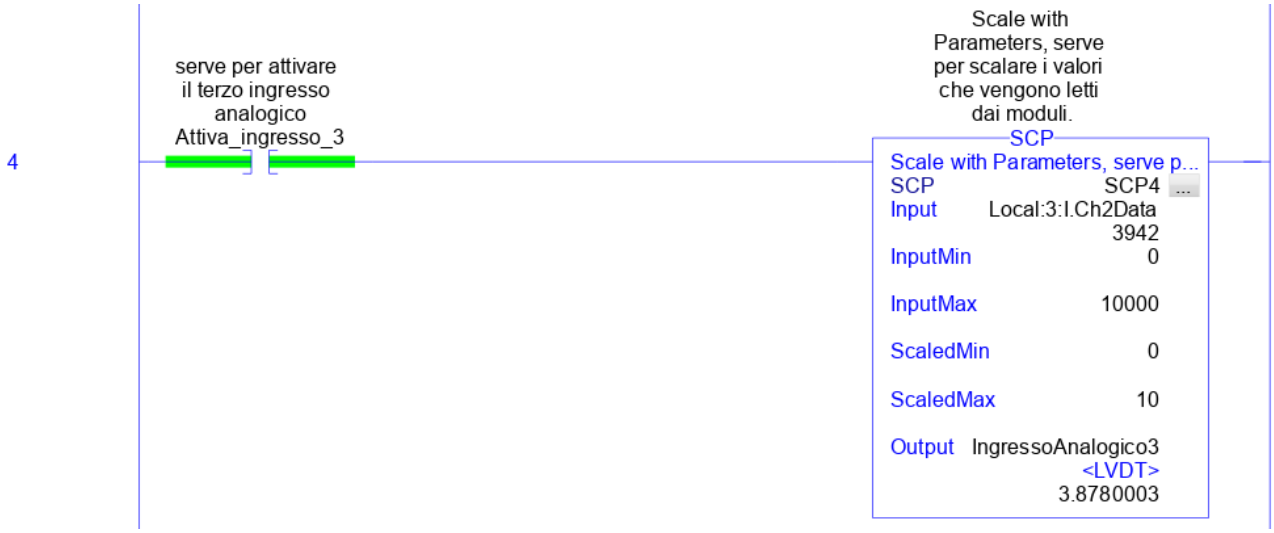
Per il **rung 2** si ha:
Questo rung è simile al primo rung spiegato, sarebbe utilizzabile per rilevare la pressione P2, ma non è utilizzato.



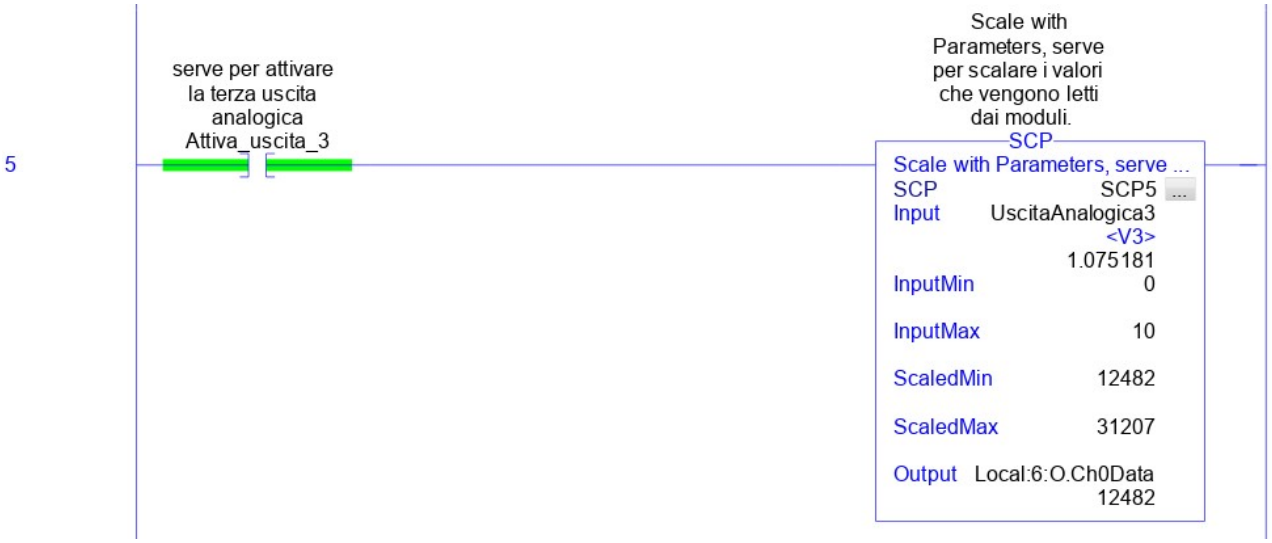
Per il **rung 3:**
è un rung di output, ci sono le stesse considerazioni del rung 1, ma questo è collegato alla valvola V2 (cioè al suo driver). Il modulo di output è il medesimo (modulo 5) ma il canale è il numero 1 (Ch1).



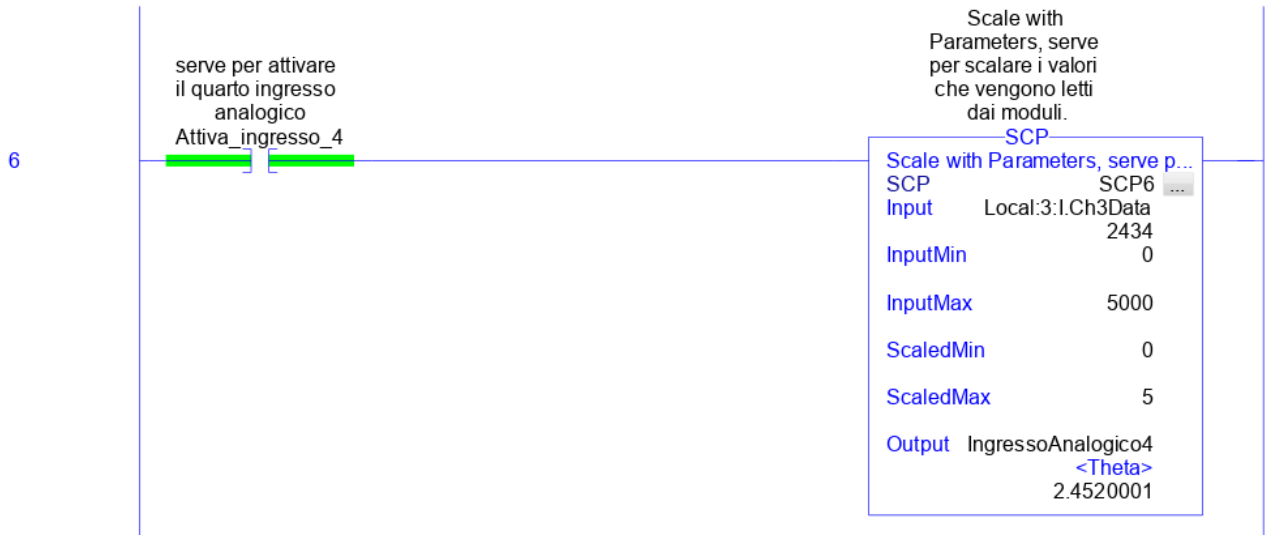
Per il **rung 4:**
Il rung 4 ha sempre il bit di attivazione dell'ingresso 3, che attiva il blocco SCP che acquisisce il valore di tensione del trasduttore LVDT sul modulo 3 canale 2 e lo converte.
Come per gli altri blocchi SCP, i parametri di settaggio bit-volt sono forniti nel manuale.
Questo blocco crea la variabile IngressoAnalogico3, ovvero l'alias di LVDT, che può variare tra 0 e 10V. Vale 0 Volt quando il carrello è completamente a destra, mentre vale 10V quando il carrello è completamente a sinistra.



Per il **rung 5:**
si ha un SCP per un modulo di uscita, in particolare il modulo 6, il cui canale 0 è collegato al driver della valvola 3.
Questo blocco prende l'input UscitaAnalogica3, alias V3, e lo converte in un valore di tensione tra 4 e 10V considerando come i blocchi precedenti la banda morta delle valvole. Questa tensione viene inviata al driver che comanda la valvola.



Per il **rung 6:**
Su questo rung si vede l'acquisizione del potenziometro rotativo per l'angolo theta.
Si può notare che il potenziometro è installato sul modulo 3, canale 3.
Il blocco SCP crea una variabile, IngressoAnalogico4, ovvero alias Theta, dove è stato impostato il campo di variazione massimo per il modulo tra 0 e 5V. In realtà il sensore darà valori più ristretti di quel campo. 2.5 V circa è la tensione quando l'asta è in verticale, questa tensione sarà utile conoscerla per i calcoli successivi.

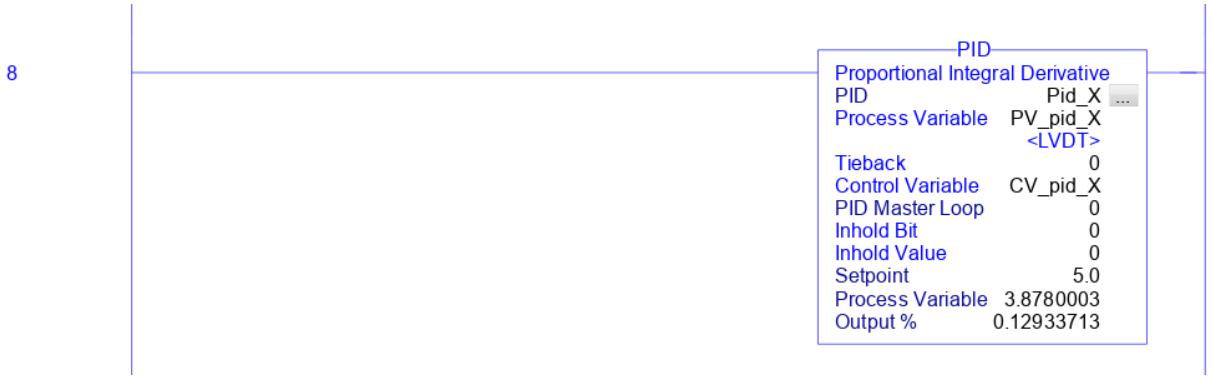


Per il **rung 7:**
Questo è un SCP di uscita, che converte la tensione non reale 0-10V usata internamente al PLC per i calcoli (UscitaAnalogica4, ovvero alias V4), in una tensione fisica tra 4 e 10V che viene inviata al driver 4 della valvola 4.
Come si nota dalla dicitura, il driver 4 è connesso sul modulo 6, al canale 1.
Anche qui si considera ovviamente la banda morta delle valvole.

Con questo rung finiscono i blocchi SCP di acquisizione/attuazione, successivamente ci saranno blocchi per calcoli, blocchi per trasferimento dati e blocchi PID.

Per il **rung 8:**
Questo rung contiene il blocco PID per quanto riguarda la posizione, si può notare la dicitura:

- PID: è il nome del PID, in questo caso chiamato Pid_X;
- Process Variable: è la variabile di processo del PID, qui chiamata PV_pid_X ovvero è il feedback del trasduttore di posizione LVDT, come si può notare dall'alias <LVDT>;
- Tieback, PID Master Loop, Inhold Bit, Inhold Value, devono essere settati su 0 per questo tipo di applicazione e per come è stato creato il ladder. Anche per il PID theta verrà seguita questa regola;
- Control Variable: è la variabile controllata del PID, ovvero la sua uscita, qui chiamata CV_pid_X;
- Setpoint: è il valore di ingresso del PID, questo valore viene impostato manualmente ed è impostato su 5V, ovvero il centro del carrello.
- Process Variable: è la lettura istantanea del feedback in volt, questo valore ovviamente cambia sempre durante il funzionamento perché è la lettura dell'LVDT.
- Output %: è il valore di uscita, in percentuale, del Pid_X.



Per il **rung 9:**
Questo rung contiene due blocchi posti in parallelo, è un rung molto importante ed interessante.

Il blocco CPT (compute):

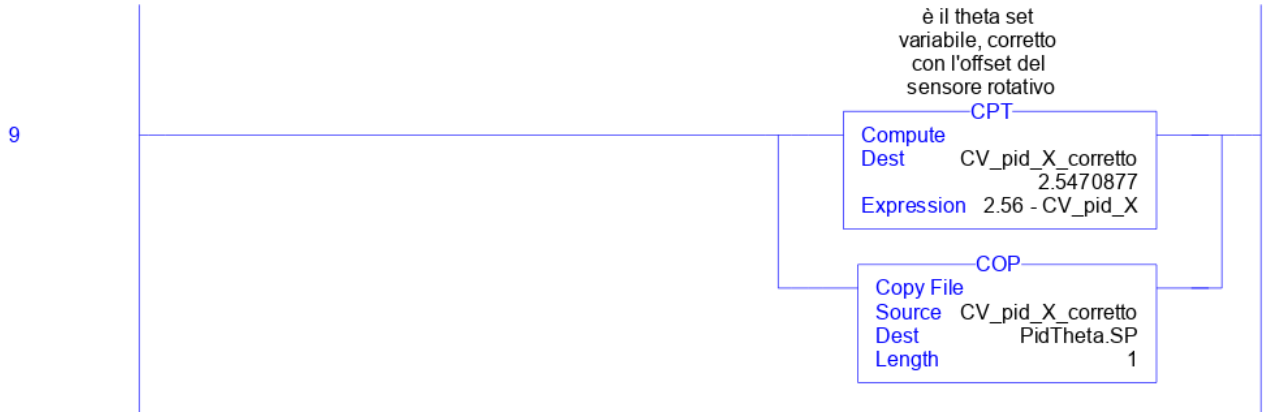
- Dest: è l'uscita del blocco, ovvero il risultato del calcolo;
- Expression: è l'espressione che si vuole calcolare.

è un blocco di calcolo, ha come input il CV_pid_X (che è l'output del Pid_X), ed ha come uscita la nuova variabile CV_pid_X_corretto.
Questo perché va ricordato che la verticale dell'asta vale circa 2.5 V.
Ponendo il caso che il carrello è perfettamente a metà della sua corsa, si ha che il set è uguale al feedback, dunque l'uscita del Pid_X è nulla.
Visto che il sistema è un doppio anello, questa uscita verrebbe inviata come input del PidTheta, ma una tensione nulla non esiste nel range fisico di variazione del potenziometro rotativo, e anche se esistesse non sarebbe comunque la verticale, ecco perché il blocco CPT crea una nuova variabile, appunto CV_pid_X_corretto che è il vero valore di set del PidTheta.
Si ha dunque CV_pid_X_corretto=2.56- CV_pid_X. In questo modo quando il Pid_X ha uscita nulla (dunque CV_pid_X=0), CV_pid_X_corretto vale 2.56V. Fisicamente questo corrisponde ad avere il carrello a metà corsa con un set verticale della rotazione dell'asta.

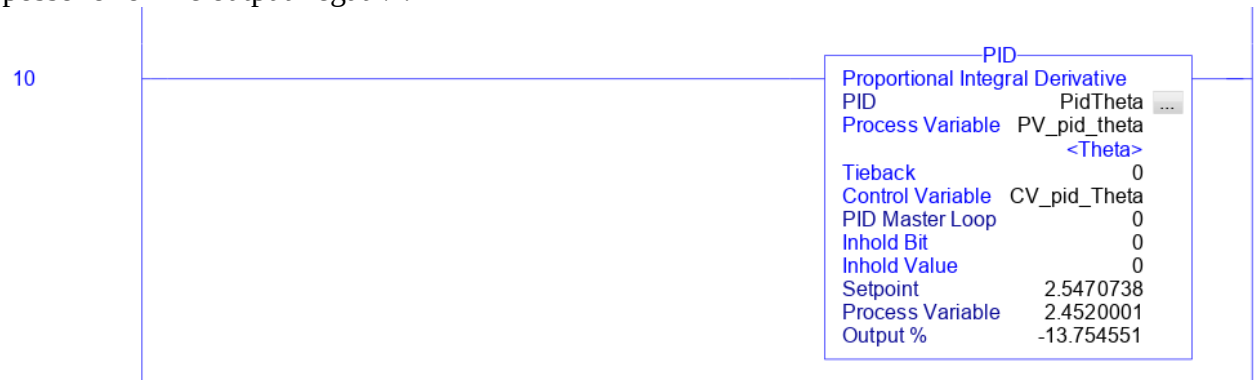
Il blocco COP (copy):

- Source: è l'ingresso del blocco, ovvero la variabile che il blocco prende per trasferirla;
- Dest: è dove quella variabile presa, va inserita;
- Lenght: è il numero di elementi di destinazione da copiare;

questo blocco prende il valore di CV_pid_X_corretto e lo invia come set al PidTheta.
Da notare che l'indirizzo di set, del PID rotazione, è PidTheta.SP, dove l'estensione SP indica appunto SetPoint.

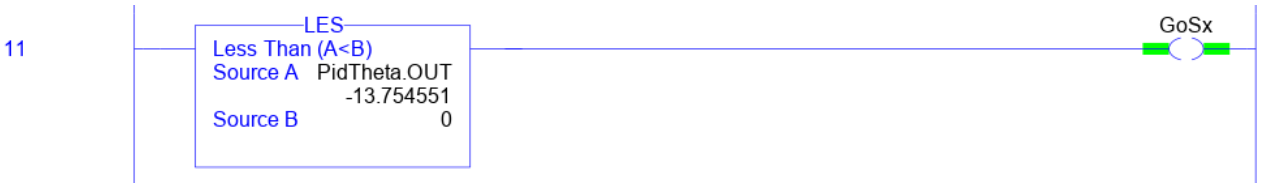


Per il **rung 10:**
Questo rung contiene il PID per la rotazione, chiamato PidTheta.
La variabile di processo, ovvero il feedback, è il valore di tensione proveniente dal potenziometro rotativo collegato all'asta, conosciuto anche con l'alias <Theta>.
Il setpoint è il valore che è stato precedentemente impostato, ovvero CV_pid_X_corretto.
Questo setpoint è continuamente variabile in funzione della posizione del carrello.
Il PID crea una variabile di controllo chiamata Cv_pid_Theta che verrà successivamente usata per azionare le valvole.
Verranno analizzate successivamente le schede di settaggio dei PID, come si vede questi PID possono fornire output negativi.

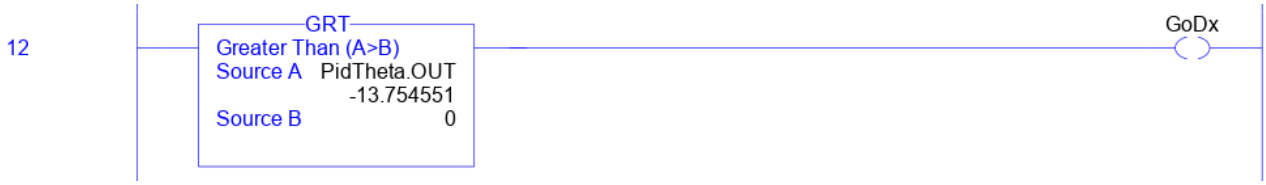


Per il **rung 11:**
Il blocco LES attiva la bobina interna del PLC, chiamata GoSx, quando il valore dell'uscita del PidTheta è negativo. Questo perché secondo la logica di funzionamento, quando l'uscita è negativa il carrello deve spostarsi verso sinistra.
LES è dunque un blocco di comparazione tra due valori:

- Source A: è il valore che dev'essere minore di Source B affinché il blocco attivi gli oggetti che sono posti a valle, in questo caso la bobina GoSx.
  Da notare che Source A corrisponde all'indirizzo dell'uscita del PidTheta, ovvero PidTheta.OUT
- Source B: il valore di soglia, impostato come 0 quando si vuole fare una comparazione tra positivo e negativo.
-



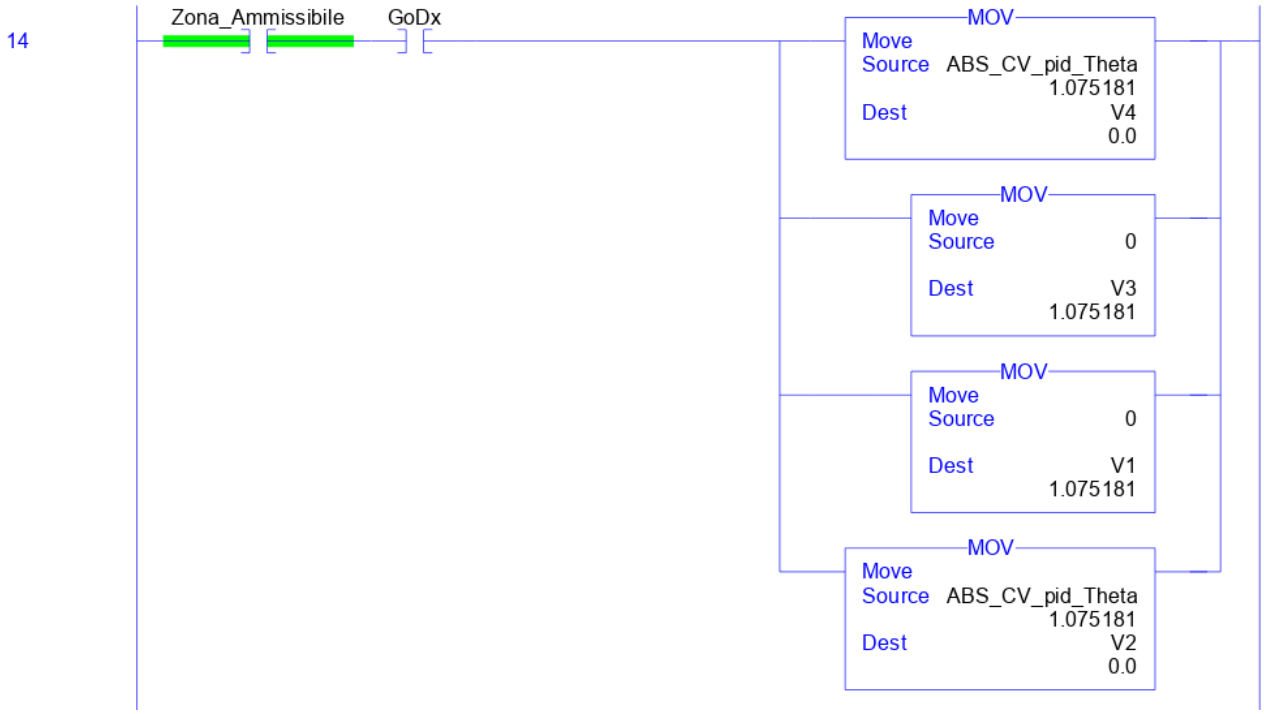
Per il **rung 12:**
Contrariamente a quanto detto prima, se l'uscita del PidTheta è positiva, viene attivata la bobina per la direzione verso destra del carrello. Come si può vedere al momento dello screenshot, la bobina non è attiva perché l'uscita del PidTheta è negativa, quindi non è vera la condizione A>B, con B sempre uguale a zero.



Per il **rung 13:**
Su questo rung si trova il blocco ABS (Absolute) che fa l'operazione valore assoluto del dato in ingresso.
Il blocco ha due voci:

- Source: valore di ingresso, in questo caso l'uscita del PidTheta;
- Dest: valore di uscita, in questo caso crea una nuova variabile chiamata ABS_CV_pid_Theta che verrà utilizzata successivamente.

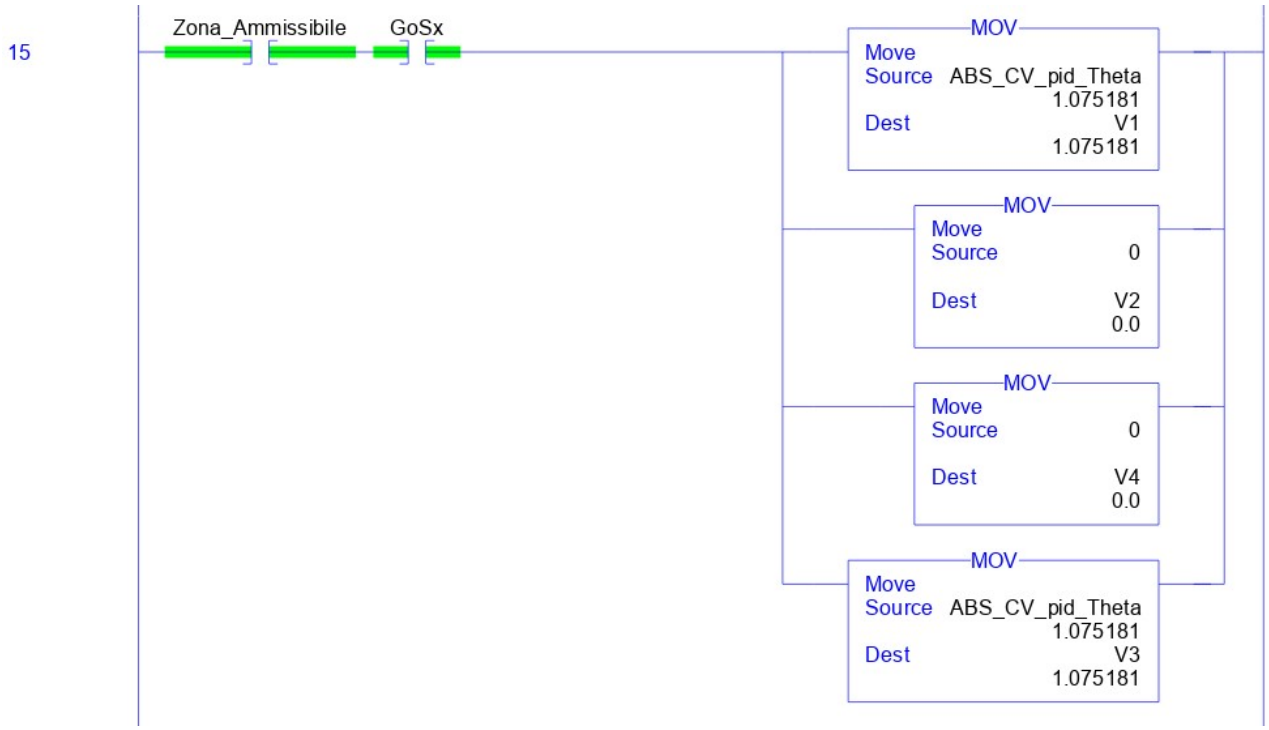
Per il **rung 14:**
Come si vede dall'immagine, il rung è unico, ma a valle ci sono molteplici blocchi posti in parallelo e attivati tutti allo stesso momento.
Zona_Ammissibile è un bit che viene attivato, come si vedrà in seguito, quando il carrello è in certi limiti. Per come è stato impostato ora il range è completo, quindi questo bit è sempre attivo, ovvero chiuso. Potrebbe essere utile per fermare il sistema quando raggiunge gli estremi.
Il bit GoDx rappresenta un'altra condizione da rispettare se si vogliono attivare i blocchi successivi. Come si vede ora il bit non è attivo, dunque i blocchi MOV che sono a valle non vengono azionati.
I blocchi MOV (Move) sono dei blocchi che servono a spostare il valore di una variabile, in un'altra variabile. Le variabili di destinazione sono le variabili di comando delle valvole, chiamate V1, V2, V3, V4.
Come si intuisce da questo rung, essendo il bit GoDx disattivo, vuol dire che il carrello ha il comando per andare verso sinistra.



Per il **rung 15:**
La struttura del rung è simile a quella precedente, cambiano solo i valori nei blocchi MOV e quello del bit della direzione.
Come si vede, qui le condizioni sono entrambe verificate, cioè il carrello è nella zona ammissibile e viene dato il comando per far andare il carrello verso sinistra.
I blocchi MOV svolgono le seguenti operazioni:

- il primo blocco aziona la valvola V1 (che manda a scarico la camera sinistra) secondo il valore assoluto della variabile controllata in uscita dal PidTheta, a questo livello la tensione di comando varia tra 0 e 10V, qui non ci si preoccupa della banda morta perché presa già in considerazione all'inizio con i blocchi SCP;
- il secondo blocco invece impone il valore 0 alla variabile V2 (valvola di carico della camera di sinistra), ciò equivale ad un duty cycle nullo quindi valvola chiusa. In realtà, data la presenza della banda morta tramite i blocchi SCP, le valvole non saranno mai chiuse;
- il terzo blocco "chiude" la valvola V4, che sarebbe la valvola di scarico della camera destra;
- il quarto blocco attiva la valvola V3, che sarebbe la valvola di carico della camera destra, secondo il valore assoluto della variabile controllata in uscita dal PidTheta.

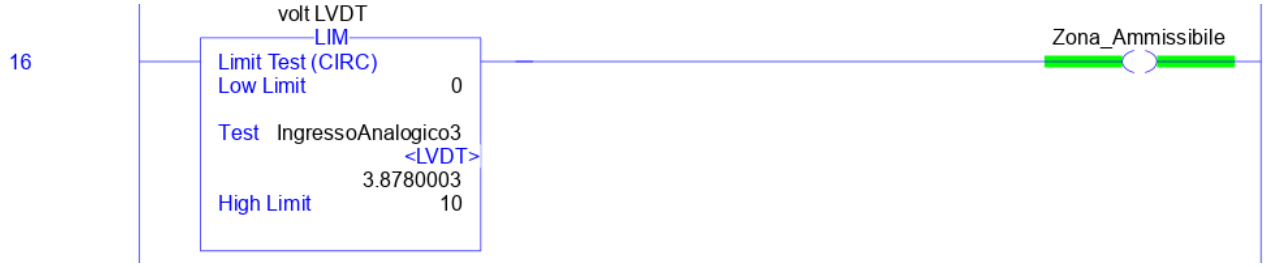
In questa configurazione si mette in pressione la camera destra, si collega all'ambiente la camera sinistra, quindi il carrello si muoverà verso sinistra.
Da notare che le valvole di scarico V1 e di carico V3, vengono modulate con lo stesso valore, ma fisicamente sono collegate su due moduli diversi: V1 è collegata sul modulo 5 canale 0, mentre V2 è collegata sul modulo 6 canale 0 (come si può notare dalle prime pagine di descrizione del programma).
Grazie a questa configurazione è stato anche possibile provare diverse logiche di funzionamento, come ad esempio aprire completamente la valvola di scarico V1 mentre si modula V3 per far andare il carrello verso sinistra, ma questo portava ad eccessive oscillazioni perché la camera si scaricava drasticamente in poco tempo.
Conviene dunque mantenere una certa contropressione per avere movimenti senza eccessive oscillazioni.



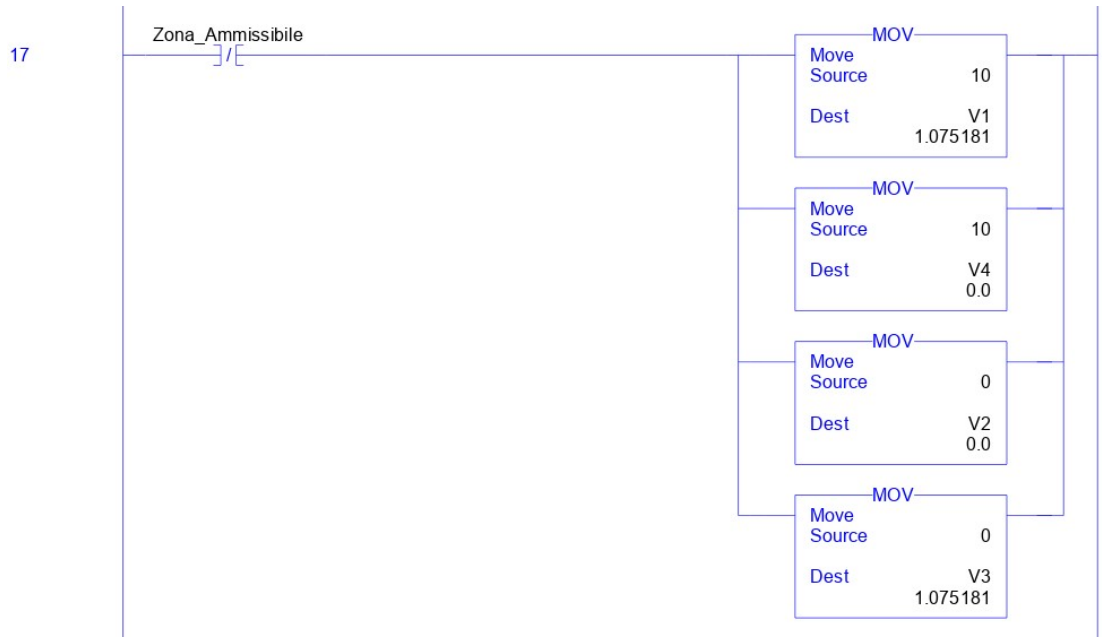
Per il **rung 16:**
Questo rung serve per impostare i limiti lineari di funzionamento del sistema.
Come si può notare il blocco LIM (Limit) compara il valore di Test, con i limiti inferiore e superiore.
Il blocco ha le seguenti caratteristiche:

- Low Limit: limite inferiore, se il valore Test è al di sotto di questo valore, l'uscita del blocco LIM è nulla;
- Test: il valore da controllare, in questo caso viene monitorata la posizione del carrello conoscendo la tensione di uscita dal trasduttore di posizione LVDT
- High Limit: limite superiore, se il valore Test è oltre questo limite, l'uscita del blocco è nulla.

Si capisce dunque che il bit Zona_Ammissibile è attivo, solo quando il valore Test è compreso tra il limite inferiore e quello superiore.
Ora i valori sono impostati come limite inferiore=0V (carrello tutto a destra) e limite superiore=10V (carrello tutto a sinistra), quindi il bit Zona_Ammissibile in questo caso è sempre attivo per ogni posizione assunta dal carrello.



Per il **rung 17:**
Quando si esce dalla zona ammissibile, il bit Zona_Ammissibile viene disattivato, dunque il contatto normalmente chiuso (che era però azionato dalla bobina Zona_Ammissibile, quindi aperto) si richiude, alimentando i blocchi MOV a valle.
I blocchi hanno la funzione di chiudere le valvole di alimentazione e aprire quelle di scarico, interrompendo il movimento del carrello.



Si descrivono ora nel dettaglio le impostazioni dei due PID:
Esistono tre schede importanti, la scheda di tuning, la scheda di configurazione e la scheda di scaling.

Per il **PID X:**
Scheda Tuning:

Si può notare in questa scheda che sono stati impostati i seguenti parametri:

- Setpoint(SP): 5, fa riferimento al centro del carrello, che corrisponde a 5V;
- Proportional Gain (Kp): è il guadagno proporzionale;
- Integral Gain (Ki): è il guadagno integrale;
- Derivative Time (Kd): è il guadagno derivativo.

Gli altri valori non devono essere modificati

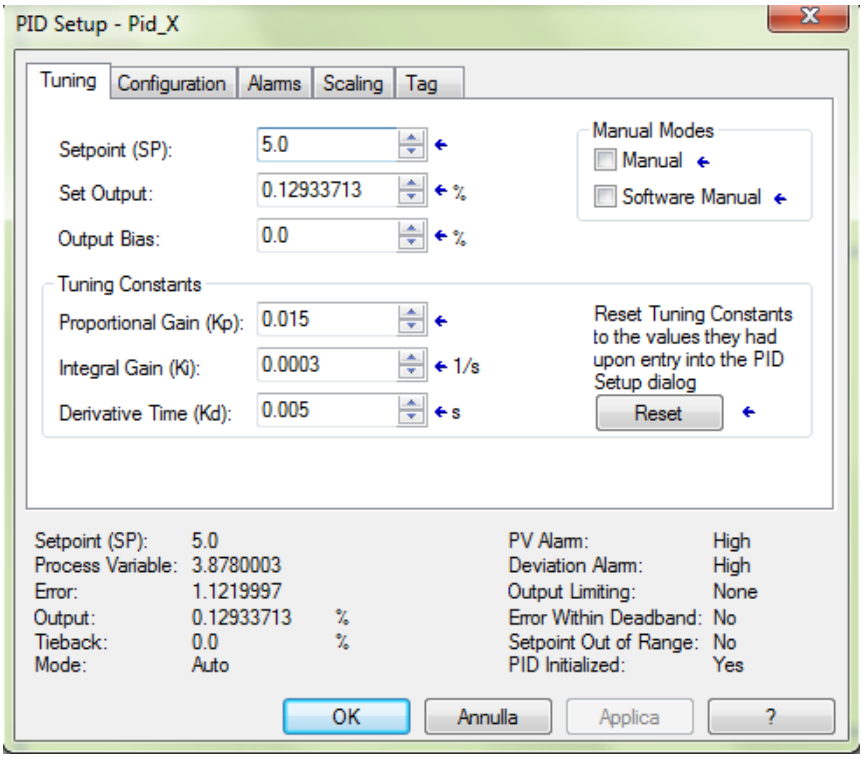


Fig 63: Scheda tuning, PID X

Scheda Configuration:

- PID Equation: settato su indipendente come default, equivale a dire che ogni parte del PID ha il rispettivo guadagno;
- Control Action: definisce l'errore come set-fb;
- Loop Update Time: settato a 1.6 ms, questo tempo influenza la risposta del sistema in quanto è l'aggiornamento dei valori del PID.
- CV High Limit: percentuale di uscita massima che il PID può assumere, in questo caso il 100%;

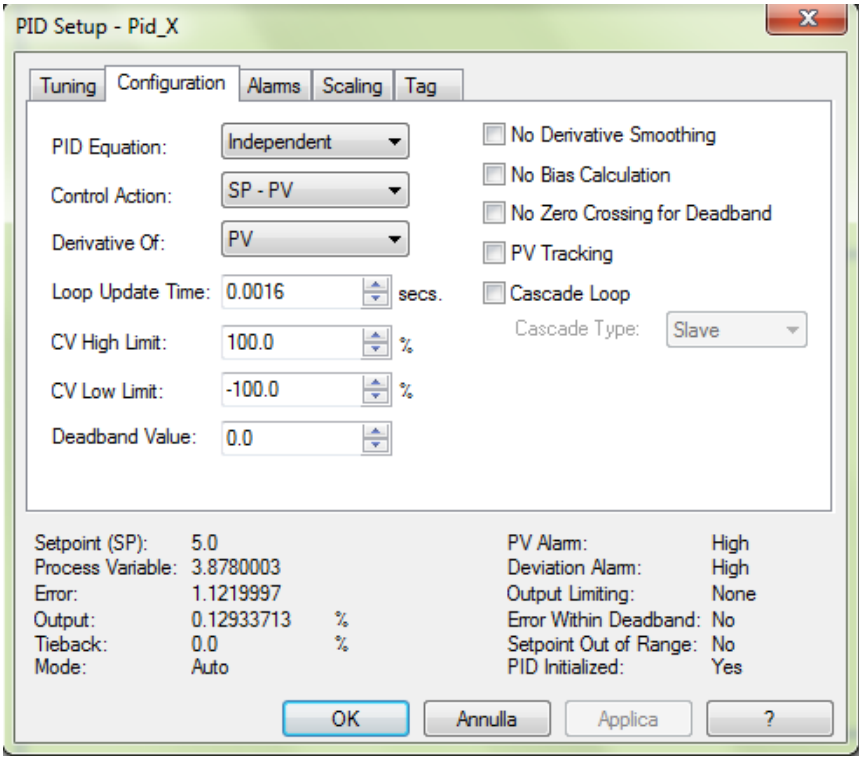


Fig 64: Scheda Configuration, PID X

- CV Low Limit: percentuale di uscita minima che il PID può assumere, da notare che è negativa. Questo è molto importante per ottenere il funzionamento corretto del sistema.

Scheda Scaling:

Nel blocco Process Variable (PV) si definiscono i valori del feedback, ovvero il campo di tensione massimo e minimo in uscita dall'LVDT, in questo caso è tra 0 e 10V.

Visto che sono stati usati i blocchi SCP in precedenza, i valori del blocco a fianco, in unità ingegneristiche, corrispondono.

Nel blocco Control Variable (CV) si definisce il campo di tensione in uscita del PID, questo valore è usato per fare i calcoli.

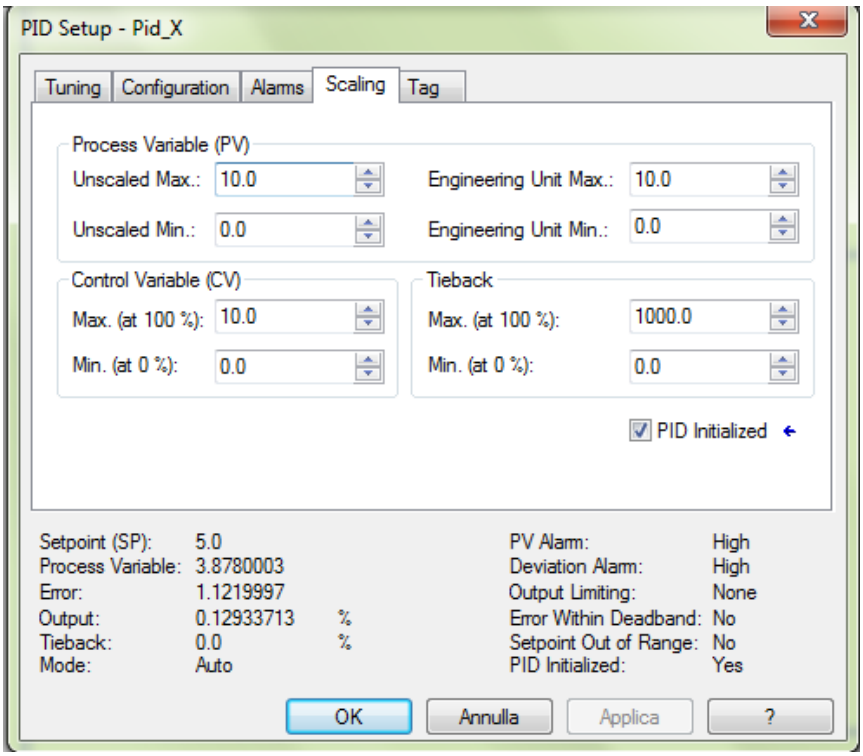


Fig 65: Scheda Scaling, PID X

Per il **PID Theta:**

Scheda Tuning:
in questa scheda NON si deve inserire il Setpoint, in quanto questo valore viene fornito dal PID X dopo esser stato corretto. Su questa scheda si impostano solo i guadagni Kp, Ki e Kd.

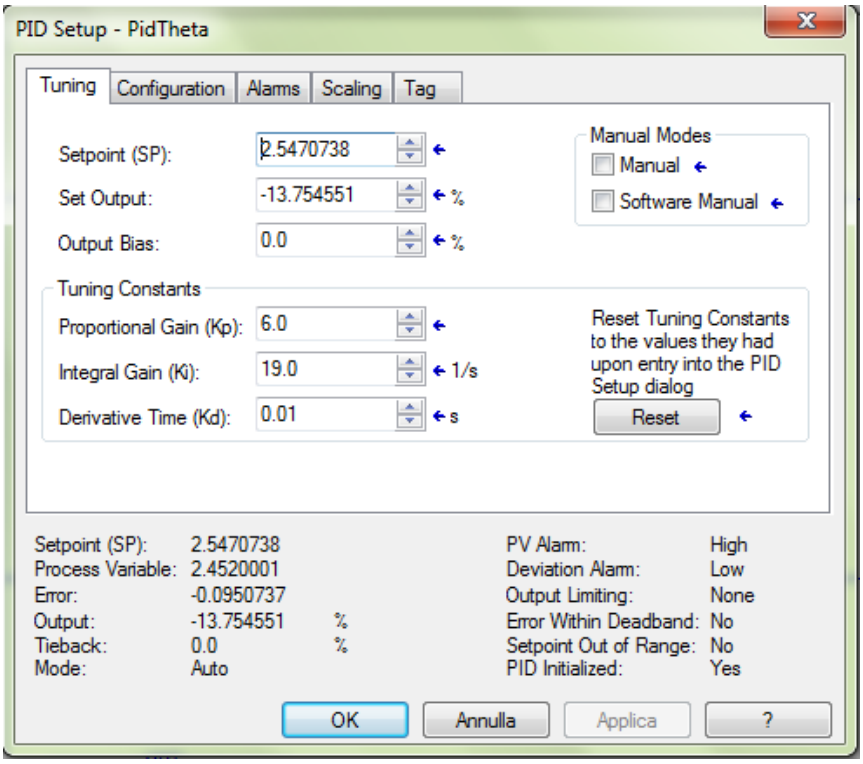


Fig 66: Scheda Tuning, PID Theta

Scheda Configuration:
Anche in questa scheda è stato definito un Loop Update Time di 1.6ms, in quanto è il valore che dava risultati migliori in termini di stabilizzazione del sistema.

L'equazione del PID è impostata come guadagni indipendenti, ma l'errore in questo caso è impostato come feedback-set.

Allo stesso modo del PID posizione, si nota che l'uscita di questo PID può essere anche negativa.

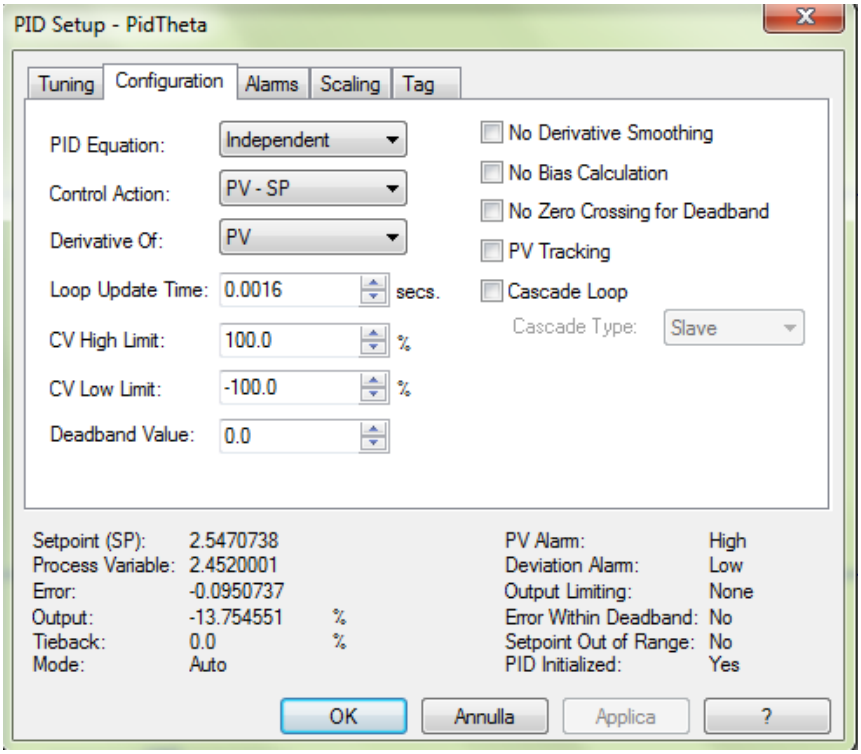


Fig 67: Scheda Configuration, PID Theta

Scheda Scaling:
Nel blocco Process Variable (PV) si inseriscono i valori massimo e minimo di tensione in uscita dal potenziometro rotativo. Con l'asta inclinata verso destra si ha una tensione di circa 3.8V, mentre con l'asta inclinata verso sinistra si ha una tensione di circa 1.3V.

Le unità ingegneristiche rimangono uguali alla variabile di processo, non c'è nessuna scalatura a causa dei blocchi SCP utilizzati in precedenza.

La variabile di controllo varia tra 0 e 10V in quanto questa sarà utile per comandare i driver, attraverso i blocchi SCP.

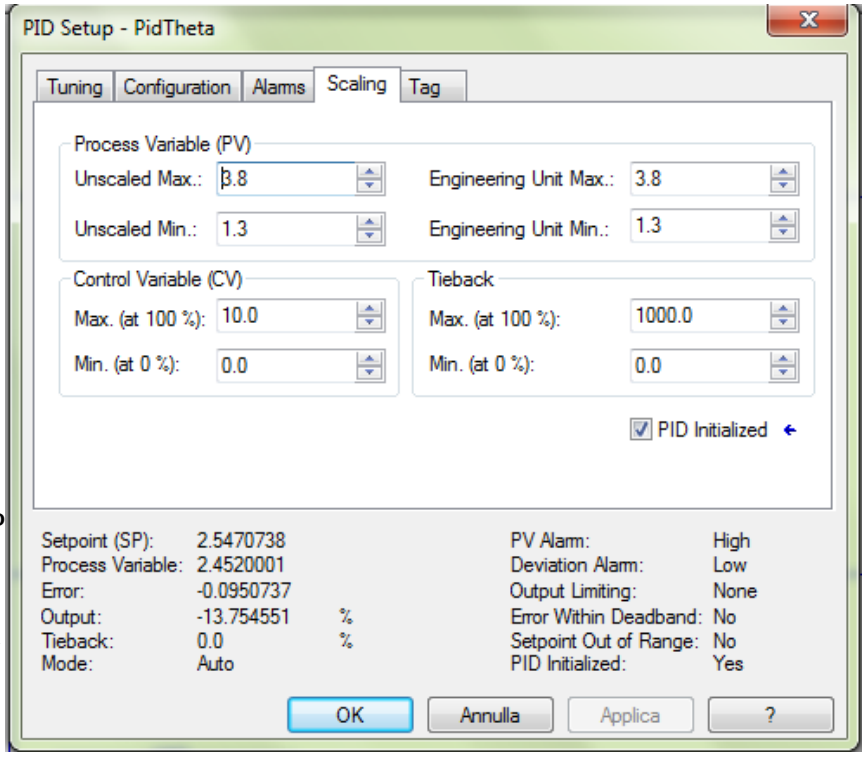


Fig 68: Scheda Scaling, PID Theta

Con queste schede finisce la programmazione effettuata sul PLC.
Per comprendere la funzionalità di queste schede e del PID in sé, sono state fatte diverse prove, inizialmente utilizzando un solo PID sulla rotazione e poi estendendo il tutto al programma completo. Prima di lavorare con il programma del pendolo, il PID è stato anche testato per un controllo pressione avente come disturbo una perdita in un serbatoio.

12. Arduino Mega

Cos’è Arduino:

Arduino è una scheda elettronica open source che comprende un microprocessore e vari ingressi e uscite che possono essere digitali o analogici, il numero e la tipologia dipende dalla scheda presa in considerazione.
Questi input/output possono essere interfacciati con delle schede o con altri circuiti.
La scheda è facilmente programmabile tramite interfaccia USB dal quale si possono caricare programmi scritti con l’apposito software.
Il linguaggio di programmazione si può dire che sia un misto tra il linguaggio C e il C++ con funzioni aggiunte e dedicate agli scopi per cui è stata creata la scheda.
Arduino è nato nel 2003 ad Ivrea, con l’idea di sviluppare una scheda a basso costo che potesse interfacciarsi con sensori e attuatori.

Hardware:

La scheda utilizzata per questo progetto di tesi è un Arduino Mega 2560 Rev.3

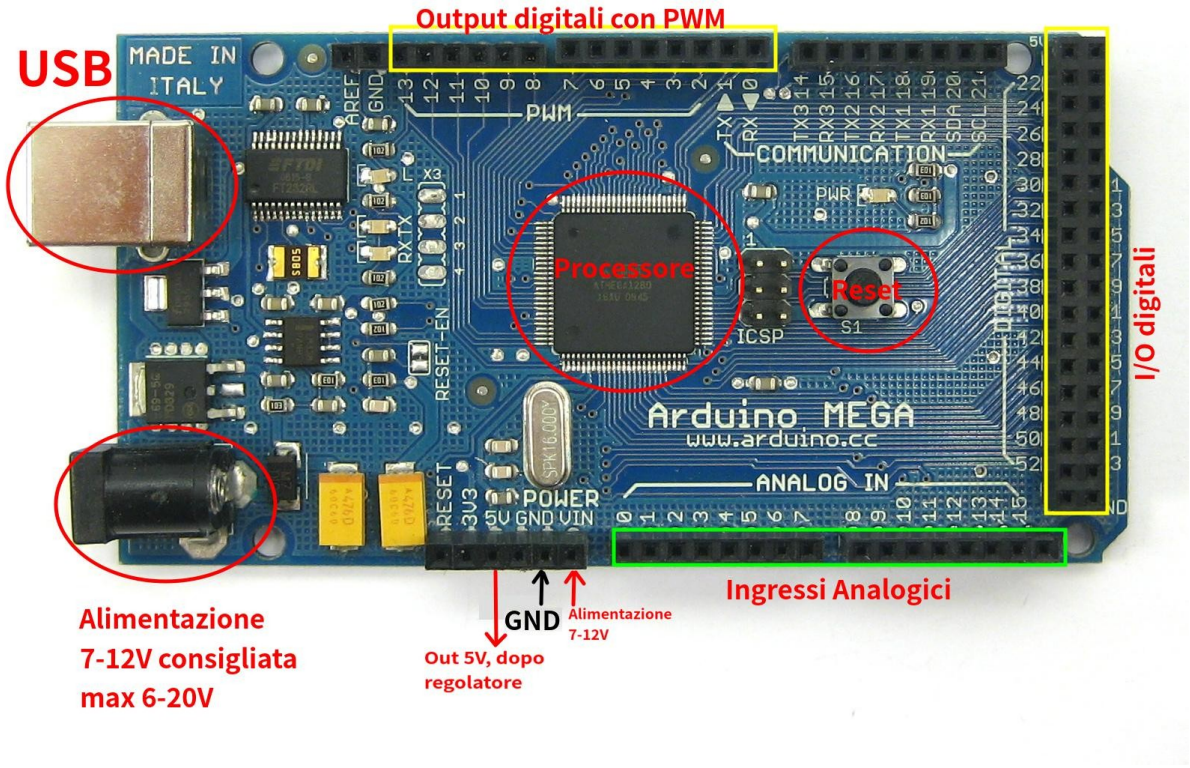


Fig 69: Arduino Mega

La scheda è basata su un processore Atmega2560, ha 54 input/output digitali tramite connettore a pettine, dei quali 15 possono essere utilizzati come uscite PWM.
La scheda ha anche 16 ingressi analogici, un oscillatore al quarzo da 16Mhz, una connessione USB, un power jack per l’alimentazione, un connettore ICSP e un bottone di reset.

L’alimentazione di questa scheda può essere fatta in differenti maniere:

- tramite cavo usb
- tramite power jack
- tramite pin Vin (che in realtà è un parallelo del power jack)
- con 5V sul pin 5V, questa tensione deve essere pulitissima e stabile in quanto bypassa il regolatore di tensione ed è direttamente connessa al processore, generalmente il pin 5V è usato come uscita, ma in casi particolari la sua funzione può cambiare.

La scheda Arduino Mega è compatibile anche con altre schede di espansione per altre versioni di Arduino.
La scheda utilizzata ha le seguenti caratteristiche:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

La scheda contiene anche un fusibile resettabile che protegge la porta USB del computer in caso di cortocircuito o sovracorrente, se più di 500mA sono assorbiti dalla porta, il fusibile interrompe la connessione fino a quando il problema non è stato risolto.
Come si nota dalla tabella, i limiti minimo e massimo di tensione dove la scheda può operare sono tra 6V e 20V DC, se l’alimentazione è sotto 7V i pin 5V possono fornire meno tensione del loro valore nominale e diventare instabili, viceversa se si usano più di 12V il regolatore di tensione può surriscaldarsi rovinando la scheda. Si consiglia dunque di usare tra 7 e 12V DC di tensione di alimentazione.

I pin in dettaglio:

- Vin: è il pin per il quale si può alimentare la scheda tramite un alimentatore esterno. Se l’alimentazione è fornita dal connettore jack questo pin, essendo in parallelo, può essere usato anche per estrarre la tensione per altri dispositivi.
- 5V: è un pin di output con 5V DC provenienti dal regolatore interno di Arduino, solo per alcune applicazioni può essere usato come alimentazione, per cui bisogna fornire 5V DC stabilizzati, in quanto questo pin bypassa il regolatore di tensione interno ad arduino.

- 3,3V: è un pin che fornisce la tensione di 3,3V ed è dopo il regolatore di tensione, certe schede lavorano con livelli logici 0-1 tra 0V e 3.3V, dunque può essere utile avere questa tensione disponibile sulla scheda.

- GND: è il ground, cioè lo 0V di riferimento.
- IOREF: è un pin che fornisce una tensione di riferimento con il quale il microcontrollore opera. Una scheda configurata propriamente può leggere la tensione del pin IOREF e scegliere l'alimentazione appropriata o azionare dei convertitori di tensione per lavorare da 5V a 3.3V

Memoria:

L'ATmega 2560 ha 256kB di memoria flash per il codice, dei quali 8kB sono usati per il bootloader; 8kB di SRAM e 4kB di EEPROM che può essere letta e scritta tramite la EEPROM library.

Input/Output:
ognuno dei 54 I/O può essere usato come input o output, usando le funzioni pinMode(), digitalWrite(), e digitalRead(). Questi operano a 5V. Ogni pin può fornire o ricevere fino a 20mA ed ha una resistenza di pull-up interna, disconnessa per default, tra 20 e 50kohm. In ogni modo non devono essere superati i 40mA di corrente per non avere danni permanenti al microcontrollore.

Certi pin hanno funzioni speciali:
- Serial: 0 è un pin di ricezione (RX) e 1 è un pin di trasmissione (TX); Serial 1: 19 (RX) e 18 (TX); Serial 2: 17 (RX) e 16 (TX); Serial 3: 15 (RX) e 14 (TX). Questi pin sono usati per ricevere e trasmettere dati TTL seriali. I pin 0 e 1 sono anche connessi con i corrispondenti pin dell'ATmega16U2 USB-to-TTL serial chip-
- Interrupts esterni: 2(interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), e 21 (interrupt 2). Questi pin possono essere configurati per innescare un interrupt ad un livello basso, ad una salita o ad una discesa, o ad un cambiamento di livello logico. Vedere la funzione attachInterrupt() per maggiori dettagli.
- PWM: da 2 a 13 e da 44 a 46. Forniscono un output PWM a 8 bit con la funzione analogWrite().
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS); questi pin supportano la comunicazione SPI utilizzando l'apposita libreria.
- Led 13: è un led installato sulla scheda connesso al pin digitale numero 13, quando il livello logico è alto questo led si accende, può essere molto utile per il controllo veloce di certi cicli o funzioni inserendo la voce digitalWrite(13,HIGH); nel codice.
- TWI: 20(SDA) e 21(SCL) spmp dei pin che supportano la comunicazione TWI usando la libreria Wire.

La scheda Arduino Mega 2560 ha anche 16 ingressi analogici, ognuno dei quali fornisce una risoluzione in 10 bit (1024 valori). Per default misurano il valore di tensione applicato tra 0 e 5V, ma è possibile cambiare i limiti usando il pin AREF con la funzione dedicata analogReference().
Ci sono altri pin sulla scheda:
- AREF: è un pin di riferimento tensione per gli ingressi analogici;
- Reset: è il pin attraverso il quale si può estendere un pulsante di reset, che collegando questo pin con il GND resetta il microcontrollore.

13. Programma Arduino nel dettaglio

Di seguito viene illustrato il programma scritto tramite il software Arduino per il funzionamento del banco. Verrà analizzata ogni singola riga di codice o ogni singolo blocco per facilitare la comprensione del programma stesso.

La struttura della programmazione di Arduino si divide in tre parti:

- una parte all'inizio del codice
- una parte di void Setup()
- una parte di void Loop()

P.s: il testo che invece viene scritto dopo due slash "//" è visto come commento e non influenza il programma, questo commento vale solo sulla singola riga, quando un commento è lungo e si va a capo bisogna scrivere nuovamente "//" per avere anche quella parte di commento.
Quando invece bisogna descrivere brevemente cosa fa un codice, si può inserire un testo tra due simbologie come nell'esempio:

- /* testo di esempio */

in questa maniera anche se si va a capo Arduino continuerà ad interpretare il tutto come commento.

Analizziamo il codice:

Questa funzione importa la libreria del PID nel programma di Arduino, si possono dunque utilizzare le varie funzioni del PID che si vedranno successivamente

```
//inclusione librerie
#include <PID_v1.h>
```

Su questo blocco invece vengono definiti gli ingressi e le uscite di Arduino, la componente #define infatti associa al nome l'ingresso o l'uscita fisica della scheda elettronica. In generale associa un nome ad un valore costante prima che il programma sia compilato. La componente #define non occupa memoria dell'hardware, in quanto il compilatore rimpiazzerà il riferimento con la costante al momento della compilazione.

```
//inclusione ingressi/uscite
#define PIN_INPUT_LVDT A0
#define PIN_INPUT_THETA A1
#define PIN_OUTPUT_SX 3 //uscite comandate da analogWrite, quindi sono già in pwm
#define PIN_OUTPUT_DX 2
```

In questo blocco si definiscono definiscono delle variabili: double è un floating a doppia precisione, è utilizzato questo perché con il PID successivamente vengono usate variabili dello stesso tipo.
Come si vede si definisce in tensione il centro del carrello (Arduino accetta tensioni tra 0 e 5V, il trasduttore LVDT ha un'uscita tra 0 e 10V, per questo motivo è stato introdotto il partitore. Arduino dunque riceve un segnale di posizione tra 0 e 5V e qui si definisce la posizione centrale.

```
double centro_carrello = 2.5;
```

Questi tre valori seguenti sono da controllare in caso il banco viene modificato o spostato, si può pensare ad esempio ad un aggiustaggio dell'asta del pendolo rimuovendo il collegamento meccanico tra essa e il potenziometro rotativo, la tensione in uscita dal potenziometro quando l'asta è in posizione verticale potrebbe dunque cambiare. Per come è assemblato ora, si definisce una tensione di uscita dal potenziometro pari a 2.56V quando l'asta è in posizione verticale, una

tensione di 1,43V quando l'asta è appoggiata verso sinistra, e una tensione di 3.69V quando l'asta è appoggiata verso destra.

```
double verticale = 2.566; //sono i volt quando l'asta è in verticale
double lettura_pot_sx = 1.43; //volt in uscita dal sensore quando il pendolo è a sinistra
double lettura_pot_dx = 3.69; //volt in uscita dal sensore quando il pendolo è a destra
```

Alla stessa maniera vengono definite variabili a doppia precisione, per quanto riguarda la rotazione dell'asta, le seguenti:

```
//variabili theta
double Setpoint_theta, Input_theta, Output_pid_theta;
double Error_theta;
double Theta_verticale;
```

Queste sono variabili di calcolo che verranno analizzate successivamente nel programma, e servono per il funzionamento dello stesso oltre che essere utili per un'eventuale acquisizione.

Analogalmente per quanto fatto per le variabili di rotazione, si definiscono le variabili sullo spostamento lineare del carrello:

```
//variabili lvdt
double Setpoint_lvdt, Input_lvdt, Output_pid_lvdt, Output_pid_lvdt_corretto;
double Error_lvdt;
```

Anche queste possono essere plottate o registrate durante il funzionamento.

Di seguito vengono definiti i parametri di settaggio per il PID della rotazione, chiamato myPIDtheta:

Dalla libreria utilizzata, vengono definite in precedenza variabili double sia i guadagni, sia gli input ed output del pid.
Dopo alcune prove è stato riscontrato un funzionamento ottimale con i seguenti parametri del PID theta:
Kp=8
Ki=25
Kd=0.5

```
//setting pid theta
double Kp_theta = 8, Ki_theta = 25, Kd_theta = 0.5; //valori dei guadagni
```

da notare che nella definizione precedente, tutti i parametri sulla linea sono definiti come double, ma questi sono separati tra loro tramite una virgola, mentre alla fine della riga dev'esserci il punto e virgola come chiusura. Allo stesso risultato si poteva arrivare definendo ogni parametro come double, ma su righe diverse e tutte chiuse da un punto e virgola.

Si definisce anche una banda morta della valvola, ovvero il valore per il quale, data la pressione e la frequenza di duty cycle, la valvola risponde correttamente ed ha una apertura lineare in proporzione al duty cycle applicato.

```
double dead_band = 90;
```

Il valore di banda morta è definito come PWM applicato alla valvola quando l'uscita dal PID è nulla, ovvero secondo la precedente definizione si ha $\frac{90}{255} \approx 35\%$ di banda morta.

Il valore 255 deriva dal fatto che le uscite hanno risoluzione di 8 bit, dunque $2^8=256$ valori.
È un valore molto grande, ma questo numero è stato trovato facendo differenti prove con differenti pressioni. Il banco ha una pressione di alimentazione che dev'essere inferiore a 7 bar (altrimenti le valvole, anche se non alimentate, non riescono ad impedire l'efflusso di aria).
Facendo varie considerazioni e pensando che il banco può essere utilizzato anche con pressioni di 5 o 6 bar, questo valore di banda morta definito a codice ha sempre una buona risposta.

Di seguito vengono definiti i parametri di settaggio per il PID della posizione, chiamato myPIDlvdt: è stato riscontrato un funzionamento ottimale con i seguenti parametri del pid x:

```
//setting pid lvdt POSITON
double Kp_lvdt = 0.015, Ki_lvdt = 0.001, Kd_lvdt = 0.005;
```

Questa è la struttura del PID per la rotazione, come si nota la funzione PID è composta da 3 parti:

- la prima parte indica che si tratta di una funzione PID
- la seconda parte è il nome del PID, in questo caso "myPIDtheta"
- la terza parte è il contenuto delle parentesi tonde, che sono i valori in ingresso e in uscita utilizzati dalla funzione

```
PID myPIDtheta(&Input_theta, &Output_pid_theta, &Setpoint_theta, Kp_theta,
Ki_theta, Kd_theta, DIRECT);
```

DIRECT: definisce l'errore come set-feedback;
questa modalità può essere cambiata anche con REVERSE: feedback-set

Generalmente si usa la modalità DIRECT e quando ci sono errori negativi l'output del PID può diventare negativo, va ricordato tuttavia che un'uscita negativa viene fornita solo se precedentemente impostata (si vedrà successivamente come impostarla).

Per il PID posizione la struttura è simile, cambia ovviamente il nome del PID che ora è myPIDlvdt e le variabili di input/output

```
PID myPIDlvdt(&Input_lvdt, &Output_pid_lvdt, &Setpoint_lvdt, Kp_lvdt, Ki_lvdt,
Kd_lvdt, DIRECT);
```

Va ricordato che nel codice, l'istruzione PID, compone una riga unica, andando a capo si genererebbe un errore.

Questo blocco compone il setup del codice, ovvero quella parte di codice che viene letta solo una volta all'accensione dell'Arduino:

```
void setup() {

}
```

Analizzando l'interno della funzione setup, ovvero la parte che sta' tra le parentesi graffe, si trova:

```
 Serial.begin(9600);
```

serve per inizializzare la trasmissione seriale tra Arduino e il PC in fase di RUN, questo è utile quando si vogliono registrare dati con le istruzioni poste alla fine del programma, attivabili togliendo il commento dalla riga desiderata.

```
 pinMode(PIN_OUTPUT_SX, OUTPUT);
 pinMode(PIN_OUTPUT_DX, OUTPUT);
```

La funzione pinMode() definisce una certa variabile con la funzione che dovrà svolgere, in questo caso abbiamo che i connettori fisici dell'Arduino, precedentemente associati ai rispettivi nomi quali PIN_OUTPUT_SX e PIN_OUTPUT_DX, sono pin digitali di uscita.
Questi pin sono collegati ai rispettivi gate dei MOSFET che gestiscono lo spostamento verso sinistra e verso destra.

All'inizio del codice era stato definito centro_carrello=2.5 e si era detto che equivaleva alla tensione in volt, al pin analogico di ingresso di Arduino, quando il carrello era a metà corsa; questo valore tuttavia non può essere utilizzato in volt e dev'essere convertito su 1024 valori.
Si ricorda infatti che Arduino è in grado di leggere valori analogici di tensione e campionarli su $2^{10}=1024$ valori.

```
Setpoint_lvdt = centro_carrello / 5 * 1023;
```

Il valore di Setpoint_lvdt verrà acquisito dal PID di posizione e confrontato con gli altri valori come il feedback.
Da notare che il valore di set di centro carrello è scritto come costante nel codice (2.5 volt), ma nulla impediva di prendere un potenziometro, fare una lettura dello stesso e inviarla come set di posizione del carrello. In questa maniera si poteva cambiare di continuo la posizione lineare di funzionamento.

Allo stesso modo si converte su 1024 valori il valore di tensione quando l'asta è in posizione verticale, nel calcolo seguente si considerano anche le letture a sinistra e a destra dell'asta, permettendo di avere variazione tra 0 (quando l'asta è appoggiata verso sinistra) a 1023 (quando l'asta è appoggiata verso destra). Questo permette di utilizzare a pieno tutto il campo.
"verticale" è quel valore di tensione in uscita dal potenziometro rotativo quando l'asta è appunto in verticale, corrisponde a circa 2.56V.

```
Theta_verticale = (verticale - lettura_pot_sx) * 1023 / (lettura_pot_dx - lettura_pot_sx);
```

Le voci seguenti riguardano i PID, sono voci molto importanti perché da queste dipende il funzionamento o no del sistema:

Per il PID della rotazione:
Questa voce permette di settare il PID, con "AUTOMATIC" si intende che il PID è azionato e funzionante. Viceversa se in una parte di codice non vi fosse la necessità di utilizzare il pid, questo potrebbe essere spento con myPID.SetMode(MANUAL);
Nel caso in tesi il pid è sempre attivo, quindi questa funzione non viene toccata.

```
myPIDtheta.SetMode(AUTOMATIC);
```

Molta attenzione va dedicata a questa funzione, con il quale si definiscono i campi di lavoro del PID stesso. Senza questa funzione il PID avrebbe un'uscita di default solo positiva, quindi tra 0 e 254 valori.
Tra le parentesi si vedono i valori minimi e massimi che l'uscita del PID può assumere, aggiungendo il campo negativo si hanno dunque uscite negative utilissime per semplificare il programma e avere continuità di funzionamento.

```
myPIDtheta.SetOutputLimits(-255, 255); //il pid può avere uscita negativa
```

Altra voce importantissima è la seguente, con il quale viene impostato il tempo di valutazione dell'algoritmo del PID. Di default è 200ms, ma questo tempo non va bene per l'applicazione in esame in quanto risulta essere troppo lento e il pendolo non riesce a stare in equilibrio, questo a parità di ogni altra condizione:

```
myPIDtheta.SetSampleTime(2);
```

Per il PID della posizione:
Allo stesso modo di come si sono definite le impostazioni per il pid della rotazioni, si definiscono le impostazioni per il PID della posizione.
Anche qui abbiamo un pid che può avere uscita negativa e un tempo di calcolo di 2ms per il PID.

```
myPIDlvdt.SetMode(AUTOMATIC);
myPIDlvdt.SetOutputLimits(-255, 255); //il pid può avere uscita negativa
myPIDlvdt.SetSampleTime(2);
```

Viene ora analizzato il cuore del codice, ovvero la parte che continua a ciclare e viene letta più e più volte durante il funzionamento.
Come si intuisce dal nome della funzione (loop), tutto quello che sta' all'interno viene continuamente letto, calcolato, verificato.

```
void loop(){
}
```

Analizzando l'interno della funzione loop, ovvero la parte che sta' tra le parentesi graffe, si trova:

La definizione di Input, ovvero la lettura tramite la funzione analogRead() del valore di tensione esistente sul "PIN_INPUT_LVDT" definito in precedenza come ingresso fisico A0, convertendola su 1024 valori, questa lettura è appunto l'input per il PID della posizione.

```
Input_lvdt = analogRead(PIN_INPUT_LVDT);
```

Si definisce anche l'errore sulla posizione, chiamato Error_lvdt, che altro non è che il set-feedback. Questo errore in realtà è già calcolato internamente al PID, qui è riportato per un eventuale plottaggio nella fase successiva, ovviamente questo errore può assumere anche valori negativi.

```
Error_lvdt=Setpoint_lvdt-Input_lvdt;
```

Di seguito si può vedere la funzione con il quale il PID fa i calcoli, su questa linea infatti vengono aggiornati i valori di uscita che verranno successivamente applicati tramite il PWM.
L'uscita è sempre su 8 bit, quindi 256 valori.

```
myPIDlvdt.Compute();

//l'uscita del pid è su 256 valori, per fare i calcoli bisogna spalmarlo su
1024 valori, siccome il pid ha anche uscita negativa, i campi minimi sono
negativi
```

Va ricordato che l'uscita del PID posizione in realtà è l'input del PID rotazione (quello chiamato come myPIDtheta), quindi i valori in ingresso a quest'ultimo PID devono essere su 10 bit, ovvero

1024 valori. Per rendere comparabili i valori e poter fare i calcoli, serve spalmare questi 8 bit di uscita del PID posizione su 10 bit di ingresso del PID rotazione, tramite la seguente funzione:

```
Output_pid_lvdt_corretto = map(Output_pid_lvdt, -255, 255, -1023, 1023);
```

Output_pid_lvdt_corretto è una variabile in 10 bit (ovvero 1024 valori) e la funzione che la genera è la funzione map().
Questa funzione prende il valore di Output_pid_lvdt, che aveva variazioni tra -255 e 255, e lo riporta su un campo più esteso, cioè tra -1023 e 1023.
La funzione map() lavora solo con numeri interi.

Quando l'errore sulla posizione è nullo, l'uscita del PID posizione (myPIDlvdt) è nulla, il setpoint del PID rotazione dunque deve essere (verticale-lettura_pot_sx)*1023/(lettura_pot_dx-lettura_pot_sx)=511.5 su 1024 valori, dove verticale è il valore in volt (2.56V) in uscita dal potenziometro rotativo quando l'asta è in verticale.

```
Setpoint_theta = Theta_verticale – Output_pid_lvdt_corretto;
```

Viene definito come input del PID rotazione la tensione letta tramite la funzione analogRead() sul "PIN_INPUT_THETA" definito in precedenza come ingresso fisico A1, convertendola su 1024 valori. Questa lettura è l'input per il PID rotazione (myPIDtheta).

```
Input_theta = analogRead(PIN_INPUT_THETA);
```

Anche qui viene definito un errore con la dicitura classica di set-feedback:

```
Error_theta=Setpoint_theta-Input_theta;
```

Viene invocata ora la funzione di calcolo del PID rotazione

```
myPIDtheta.Compute();
```

Di seguito ci sono le condizioni che regolano il funzionamento del pendolo, esiste una sola condizione "if" sul segno dell'output del PID rotazione. Se l'output è positivo si vuole che il carrello vada verso destra, viceversa con output negativo si vuole che il carrello vada verso sinistra.

Analizzando il caso di uscita PID positiva:

```
if (Output_pid_theta >= 0) {
```

Si annulla il movimento in direzione opposta imponendo un duty cycle nullo al pin, che tramite il mosfet, comanda le valvole nella direzione opposta (in questo caso di annulla ogni azionamento che porterebbe il carrello a muoversi verso destra). Tramite questa istruzione la valvola è a tutti gli effetti chiusa perché viene inviato il valore 0 al pin digitale.
Si ricorda che la funzione analogWrite() è la funzione che genera il PWM (data la scheda il PWM ha una frequenza di 490Hz che è la stessa frequenza utilizzata dai driver associati alle valvole utilizzati con il PLC.

```
analogWrite(PIN_OUTPUT_DX, 0);//annulla la direzione opposta
```

Con la stessa funzione si attiva la direzione verso sinistra del carrello, considerando la banda morta. La funzione map() interna alla funzione analogWrite() trasforma la variabile "Output_pid_theta" dai

valori originari (tra 0 e 255), ai valori con banda morta (tra dead_band e 255). Come si nota il valore che viene ricevuto dalle valvole è sempre tra la banda morta e 255, vale banda morta quando l'uscita del PID è nulla, mentre vale 255 quando l'uscita del PID è massima.
Riassumendo: analogWrite riceve un valore modificato, che tiene conto della banda morta, e aggiorna la sua uscita in PWM sul pin digitale definito in precedenza (in questo caso il "PIN_OUTPUT_SX" corrisponde al pin digitale numero 3).

```
analogWrite(PIN_OUTPUT_SX, map(Output_pid_theta, 0, 255, dead_band, 255));
}
```

Se invece l'uscita del PID è negativa:

```
else {
```

Si annulla la direzione verso sinistra perché ora c'è la necessità di andare verso destra con il carrello

```
analogWrite(PIN_OUTPUT_SX, 0);//annulla la direzione opposta
```

e si attiva la direzione verso destra, da notare che su questa riga ci sono tre funzioni annidate, ovvero analogWrite() contiene la funzione map() che a sua volta contiene la funzione abs().
Le prime due funzioni sono state spiegate per il caso precedente e non c'è nessuna differenza nalla logica di funzionamento.
La novità qui è la funzione map(): questa funzione fa il valore assoluto del valore "Output_pid_theta" perché in questo caso è negativo.
La funzione analogWrite() può ricevere valori solo positivi perché deve "scrivere" su dei pin fisici di Arduino che hanno un valore di tensione che varia da 0 a 5 volt con un duty cycle anch'esso positivo.

```
analogWrite(PIN_OUTPUT_DX, map(abs(Output_pid_theta), 0, 255, dead_band, 255));
}
```

Ci sono ancora delle funzioni che possono essere utili per capire il programma, sia leggendo i valori in uscita quando si sposta il carrello/asta a mano (con tubi dell'aria compressa staccati) oppure durante il funzionamento.
C'è da notare che queste funzioni comportano un ritardo, dunque la risposta del banco cambia a seconda di quante funzioni Serial.print() sono attive.
La funzione delay() comporta un'attesa durante il loop, questa è ovviamente commentata durante il funzionamento ma può essere utile togliere il commento e vedere come cambiano i parametri in fase di setting, anche numericamente, senza avere migliaia di valori stampati sul monitor al secondo. Il numero all'interno della parentesi è il tempo in ms di attesa, nell'esempio delay(500); implica un'attesa di 500ms.
Nel software Arduino si possono aprire due interessanti strumenti:

1. monitor seriale: permette di leggere i valori tramite numeri
2. plotter seriale: permette di visualizzare i valori tramite grafici

Alcuni valori che potrebbe essere interessante stampare sono i seguenti:

Per stampare la tensione di set del carrello, che è sempre compresa tra 0 e 5V:

```
//Serial.println(centro_carrello);
```

Per stampare il theta set variabile in tensione:

```
 //Serial.println(Setpoint_theta*((lettura_pot_dx-lettura_pot_sx)/1023)+lettura_pot_sx);
```

Per stampare la tensione dell'lvdt al pin di Arduino, sarebbe il feedback sulla posizione:

```
 //Serial.println(Input_lvdt/1023*5);
```

Per stampare la tensione del potenziometro rotativo al pin di arduino, sarebbe il feedback della rotazione:

```
 //Serial.println(Input_theta*((lettura_pot_dx-lettura_pot_sx)/1023)+lettura_pot_sx);
```

Per stampare l'errore sulla posizione in 1024 valori:

```
 //Serial.println(Error_lvdt);
```

Per stampare l'errore sulla rotazione in 1024 valori:

```
 //Serial.println(Error_theta);
```

Per stampare l'uscita del PID posizione su 255 valori:

```
 //Serial.println(Output_pid_lvdt);
```

Per stampare l'uscita del PID rotazione su 255 valori:

```
 //Serial.println(Output_pid_theta);

 //delay(500);
}
```

Eventualmente si può pensare di impostare certi parametri durante il funzionamento tramite la comunicazione seriale, ad esempio questo è stato utilizzato per impostare la posizione di set per generare una funzione a scalino.
Variando il set si vede come risponde il sistema.

Nella programmazione questo può essere fatto come:

```
void loop(){
 if (Serial.available()) {
    centro_carrello = Serial.read()-48; //dovrebbe essere un char per stampare il valore corretto, siccome è double si toglie il 48
  }
  //lvdt collegato tramite partitore, ingresso tra 0 e 5 volt al pin, su 1024 valori
  Setpoint_lvdt = centro_carrello / 5 * 1023;
……..
}
```

Le modifiche sono semplici: bisogna solo portare il calcolo del Setpoint_lvdt nel loop e aggiungere, sempre nel loop, la parte di ricezione del valore.

La condizione “if” serve per verificare se è stato passato un valore tramite il seriale, ovvero la funzione Serial.available() è quella che riconosce quando un numero viene digitato e inviato tramite l’apposita finestra. Se un valore è presente, viene scritto come variabile sulla riga successiva.

Serial.read() è una funzione che legge il valore passato tramite il terminale e lo riporta alla variabile assegnata nel codice, in questo caso centro_carrello.

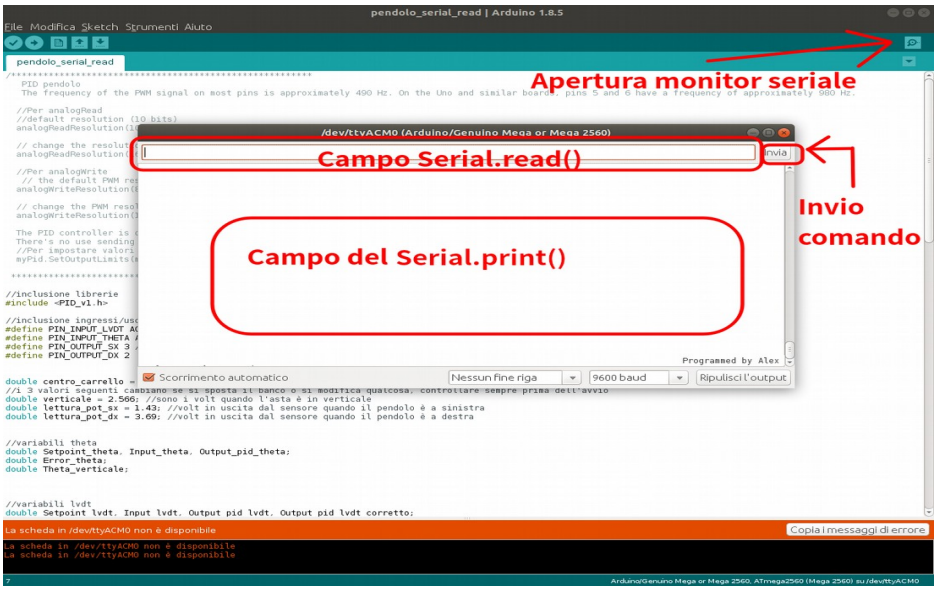


Fig 70: Finestra seriale Arduino

14. MOSFET

Il MOSFET, acronimo di Metal-oxide-semiconductor field-effect transistor è un tipo di transistor ad effetto di campo (FET), fabbricato controllando l'ossidazione del silicio.
Questo componente ha un gate isolato, tramite il quale si applica tensione per portarlo in conduzione.
La capacità di cambiare conduttività a seconda del livello di tensione applicato è molto utile per creare degli amplificatori o switch.
Il vantaggio del MOSFET è che non richiede quasi nessuna corrente di input per controllare il carico, ma richiede solo un livello di tensione. Questo può essere utile quando si hanno dispositivi in grado di fornire un riferimento in tensione ma che hanno potenze interne molto basse che non permettono di pilotare carichi.
In un certo senso si può dire che incrementando il livello di tensione applicato al gate del MOSFET si incrementa la conduttività del dispositivo (esiste una caratteristica che cambia da MOSFET a MOSFET ed è funzione della tensione applicata al gate ma anche della tensione esistente tra drain e source.

Come funziona:

- Vth è la tensione di treshold, ovvero la tensione che applicata al gate porta in conduzione il MOSFET;
- Vgs è la tensione applicata tra gate e source;

Con lo switch aperto, il transistor non conduce dunque il led è spento perché non passa corrente tra drain e source.
La resistenza tra gate e source è una resistenza di pull-down, ovvero connette il gate attraverso di sé al source, dando il riferimento di zero.
Quando lo switch viene attivato, il gate è collegato con il polo positivo della batteria e sulla resistenza di pull-down si crea una tensione uguale a quella dell'alimentazione (essendo in parallelo ad essa). Il MOSFET passa dunque in conduzione e il led si accende.

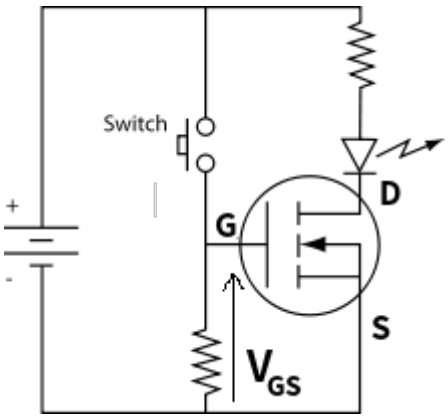


Fig 71: Pilotaggio led attraverso MOSFET

Una caratteristica molto importante per il mosfet è quella riportata qui a destra.
La caratteristica ha sull'asse delle ordinate la corrente del drain, ovvero la corrente del carico da comandare, mentre ha sull'asse delle ascisse la corrente tra drain e source.
Si possono dunque notare due regioni, separate dalla curva in rosso, a sinistra si ha la regione lineare, detta anche zona ohmica, mentre a destra si ha la regione di saturazione. Le curve sono distinte invece da Vgs-Vth.

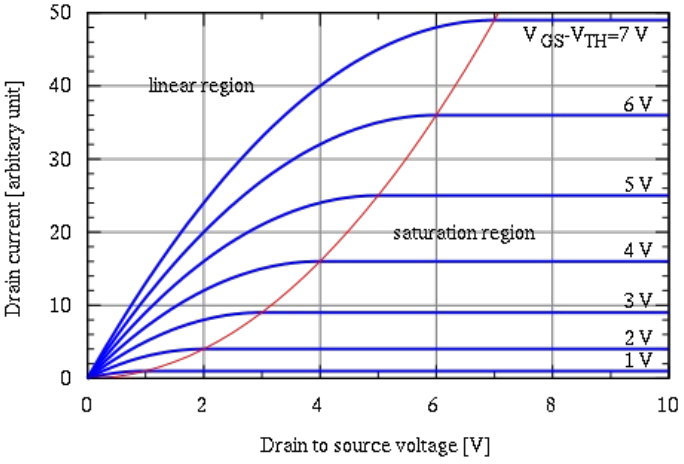


Fig 72: Caratteristica MOSFET

Riassumendo si ha:

- zona interdizione quando Vgs<Vth;
- zona lineare quando Vgs>Vth e Vds<(Vgs-Vth);
- zona di saturazione quando Vgs>Vth e Vds>(Vgs-Vth);

Nella zona lineare il MOSFET è in conduzione ed opera come se fosse un resistore, controllato dalla tensioni di gate e dalle tensioni Vgs e Vth.
Nella zona di saturazione il MOSFET è in conduzione ma qui la regione non è più ohmica, dunque la corrente può variare senza cambiare la tensione Vgs, il comportamento è dunque come uno switch.

L'applicazione del mosfet IFR520 al banco:
Nel banco l'alimentazione delle valvole è a 24V, dunque la tensione Vds=24V.
Si nota subito che questa tensione è ben sopportata dal MOSFET che può comandare carichi fino a 100V.
La tensione di comando viene fornita tramite Arduino, ed ha un valore tra 0 e 5V. Anche qui la tensione è nel campo perché il MOSFET sopporta una tensione Vgs +-20V.

Si riporta di seguito la tabella con i valori massimi di questo MOSFET:

ABSOLUTE MAXIMUM RATINGS (T_C = 25 °C, unless otherwise noted)					
PARAMETER			**SYMBOL**	**LIMIT**	**UNIT**
Drain-Source Voltage			V_{DS}	100	V
Gate-Source Voltage			V_{GS}	± 20	
Continuous Drain Current	V_{GS} at 10 V	T_C = 25 °C	I_D	9.2	A
		T_C = 100 °C		6.5	
Pulsed Drain Current[a]			I_{DM}	37	
Linear Derating Factor				0.40	W/°C
Single Pulse Avalanche Energy[b]			E_{AS}	200	mJ
Repetitive Avalanche Current[a]			I_{AR}	9.2	A
Repetitive Avalanche Energy[a]			E_{AR}	6.0	mJ
Maximum Power Dissipation	T_C = 25 °C		P_D	60	W
Peak Diode Recovery dV/dt[c]			dV/dt	5.5	V/ns
Operating Junction and Storage Temperature Range			T_J, T_{stg}	- 55 to + 175	°C
Soldering Recommendations (Peak Temperature)	for 10 s			300[d]	
Mounting Torque	6-32 or M3 screw			10	lbf · in
				1.1	N · m

Fig 73: Valori massimi MOSFET

Data la configurazione del banco, si può prendere la caratteristica del MOSFET (Id, Vds) entrando con la tensione del carico di 24V, individuando la curva Vgs=5V, e ricavando la corrente che il MOSFET può erogare, che è di 2A.

Con un rapido calcolo: $P = V \cdot I = 24 \cdot 2 = 48\,W$ si vede che la potenza erogabile di 48W è molto maggiore della potenza effettivamente richiesta dalle valvole (sono comandate due valvole da 6.5W ciascuna, per un carico totale al MOSFET di 13W).

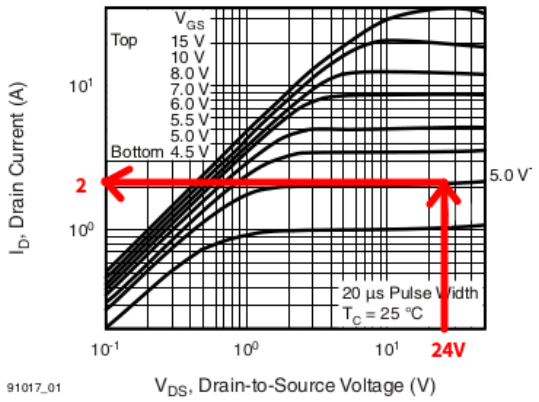


Fig. 1 - Typical Output Characteristics, T_C = 25 °C

L'applicazione risulta dunque verificata 13W < 48W.

Data la bassa potenza non c'è neanche la necessità di installare dei dissipatori termici in quanto le temperature raggiunte dal MOSFET durante il funzionamento sono molto basse.

15. Scheda di potenza con MOSFET

La scheda di potenza è composta da:

- 2 MOSFET IRF520, indicati con M1 e M2 sullo schema;
- 2 Diodi 1N4007, indicati con D1 e D2;
- 2 Resistori da 10kOhm 1/4W, indicati con R1 e R2;
- 1 connettore a 6 poli, i quali pin sono indicati con sigle da J1 a J6;

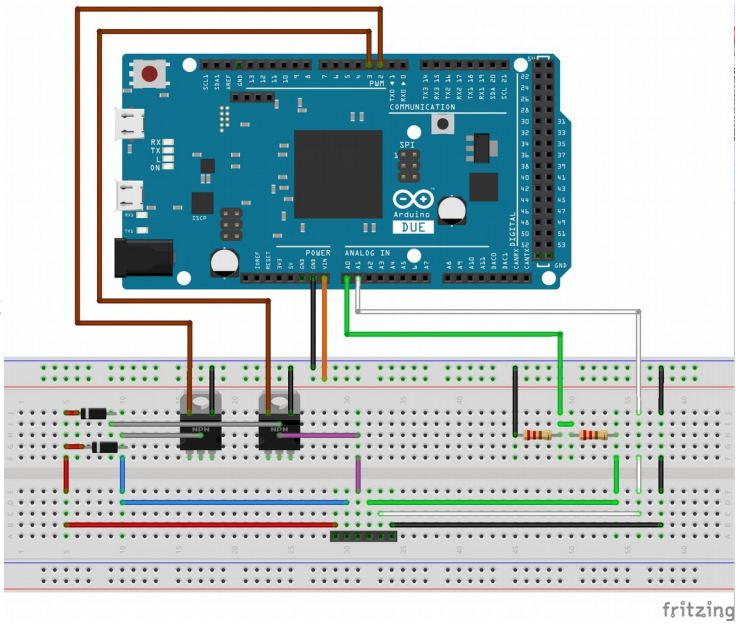


Fig 74: Circuito di potenza su breadbord e connessioni con Arduino

L'alimentazione a 24V DC è collegata alle valvole tramite il pin J6 del connettore, dunque tutte e quattro le valvole hanno contemporaneamente la tensione positiva applicata, quello che viene comandato in realtà è il loro negativo, tramite i rispettivi pin J4 e J5. Questo è dovuto al fatto che sono stati utilizzati dei MOSFET di tipo N.
Le valvole sono connesse in parallelo due a due, ciò vuol dire che la valvola V1 è collegata con la valvola V3, mentre la valvola V2 è collegata con la valvola V4. Questo semplifica il circuito e funziona bene con la logica adottata, in quanto per comandare il cilindro in una determinata direzione serve comandare una valvola in carico e l'altra in scarico, ma queste vengono comandate con lo stesso PWM, quindi va bene che siano connesse elettricamente in parallelo.

I diodi D1 e D2 sono diodi di protezione, in quanto sono in parallelo al carico elettrico (le valvole in questo caso). Questi diodi proteggono i MOSFET che sono messi a valle, che interrompono quindi una corrente induttiva. In realtà è una precauzione che si prende quando ci sono dei transistor a comandare il tutto, ma secondo il datasheet, i MOSFET sono già progettati per comandare carichi induttivi. L'esperienza tuttavia suggerisce questa piccola aggiunta al circuito che altro non fa che scaricare l'energia residua dell'induttanza all'apertura del circuito sui diodi stessi, rendendo meno pesante il compito assegnato ai MOSFET.

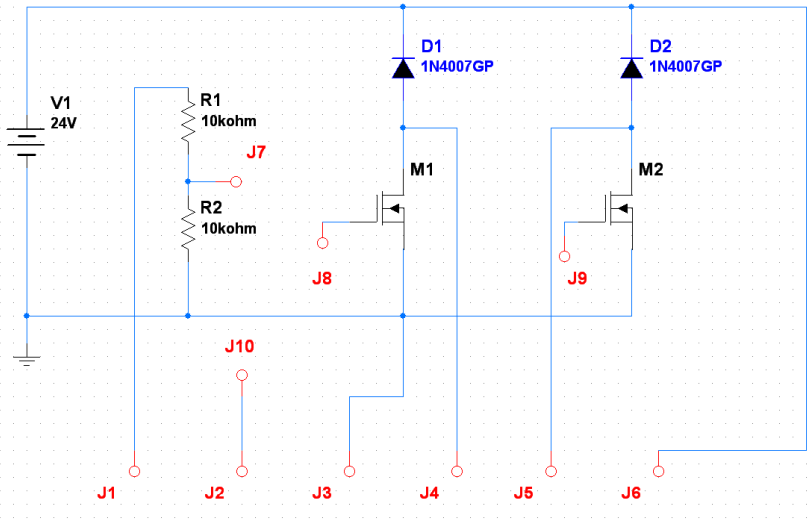


Fig 75: Circuito di potenza

Sulla scheda è anche presente un partitore di tensione: l'ingresso J1 sul connettore è collegato al trasduttore di posizione LVDT, che ha un'uscita compresa tra 0V (quando il carrello è tutto a destra) e 10V (quando il carrello è tutto a sinistra).

Arduino può accettare sugli ingressi analogici tensioni fino a 5V, quindi tramite un partitore avente due resistenze uguali da 10kOhm, si divide la tensione.
Nel programma di Arduino dunque la posizione centrale sarà a 2,5V e non a 5V come nel PLC.
Il pin J7, dopo il partitore, è connesso all'ingresso analogico A0 di Arduino.
Il pin J2 è connesso direttamente al pin J10 che altro non è che il pin analogico A1 di Arduino, questo è il segnale che viene ricevuto dal potenziometro rotativo che in ogni condizione ha un'uscita inferiore a 5V.
Il pin J3 è collegato con il GND, va ricordato che tutti i Ground del sistema sono connessi insieme per avere lo stesso riferimento.
I pin J8 e J9 sono di comando per i due MOSFET, sono collegati rispettivamente ai pin digitali 2 e 3 di Arduino.

Il funzionamento:

I MOSFET, essendo collegati ai pid digitali di Arduino, ricevono sul Gate una tensione tra 0 e 5V; questo segnale è accompagnato da una corrente bassissima, ancora inferiore a quella di cui Arduino è capace (si ricorda che Arduino è limitato a 20mA come corrente di uscita dal pin).

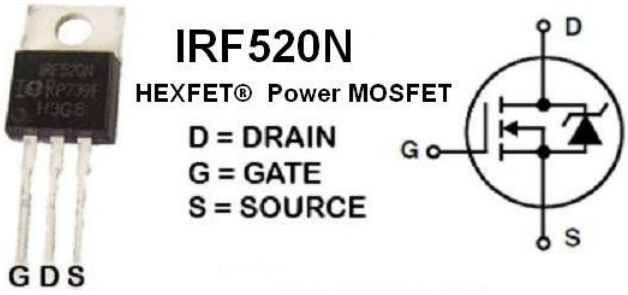


Fig 76: Pinout MOSFET

Quando il MOSFET è attivato con il livello logico alto (+5V) viene messo in collegamento il Drain con il Source, facendo passare la corrente attraverso le due valvole comandate dal MOSFET.
Qui si apprezza il fatto che il diodo sul ramo è installato con il catodo verso il positivo, dunque normalmente è un circuito aperto e non assorbe corrente.
Quando il MOSFET viene disattivato con il livello logico basso (0V) il circuito viene aperto e la corrente attraverso le valvole interrotta, l'energia induttiva si può ora scariare attraverso il diodo posto in parallelo con le valvole.
Date le basse potenze in gioco (si parla di 6.5W per valvola, quindi 13W che il singolo MOSFET deve comandare) non c'è bisogno di installare un dissipatore sul MOSFET stesso in quanto le temperature raggiunte sono trascurabili.

Qual è il vantaggio:

Il MOSFET è la vera interfaccia di potenza tra Arduino e la valvola e fornisce i seguenti vantaggi:

- La tensione massima in uscita da Arduino è 5V, mentre le valvole vanno pilotate con 24V, i MOSFET riescono dunque a mettere in "comunicazione" questi due differenti sistemi;
- La corrente massima di Arduino è 20mA, che non basta per comandare le valvole, dunque anche in questo caso il MOSFET dà la possibilità di "amplificare" la corrente di comando;
- È un componente piccolo e poco costoso, facile da usare e molto veloce nella commutazione, adatto dunque al PWM.

Assemblaggio della scheda:

Al di sotto della scheda sono stati fatti passare i cavi per le connessioni e si vedono anche i connettori a pettine utili per il collegamento diretto con Arduino.

La scheda è stata pensata in prima battuta per essere resistente ed esteticamente bella.
Facendo passare i cavi in questo modo infatti non c'è pericolo che le saldature si distacchino anche se vengono tirati.

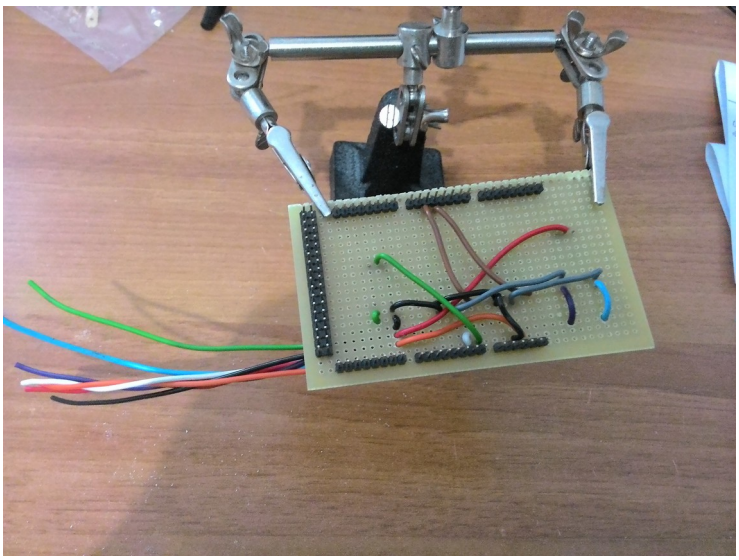


Fig 77: Assemblaggio, parte posteriore

Sulla faccia anteriore invece si possono vedere, da sinistra verso destra:

- i diodi di protezione dei MOSFET;
- i MOSFET;
- i resistori che formano il partitore di tensione.

La scheda inoltre può essere facilmente collegata al banco tramite il connettore a 6 poli.

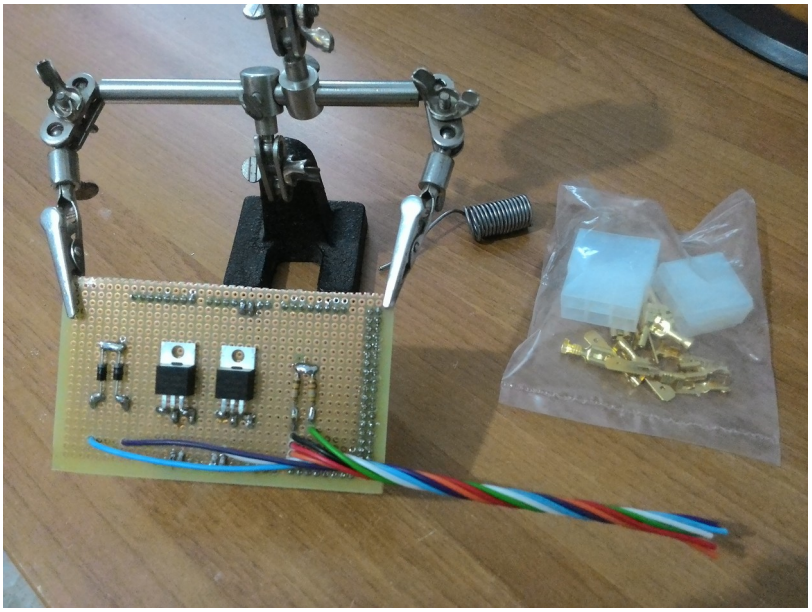


Fig 78: Assemblaggio, parte anteriore

16. Confronto del controllo tra PLC Rockwell e Arduino

Facendo delle acquisizioni durante il funzionamento, si può rilevare l'andamento della posizione e della rotazione dell'asta nel tempo.
Queste prove sono state svolte sia senza disturbare il sistema, sia disturbandolo con forze esterne o cambi di set.

PLC Rockwell:

Di seguito si può notare l'andamento su 30 secondi del controllo tramite PLC, il pendolo è tenuto in equilibrio ma è presente un'oscillazione di circa 10 cm del carrello ed una conseguente oscillazione con ampiezza picco picco di circa 7°.

La sinusoide che si genera ha anche un periodo vicino al secondo.

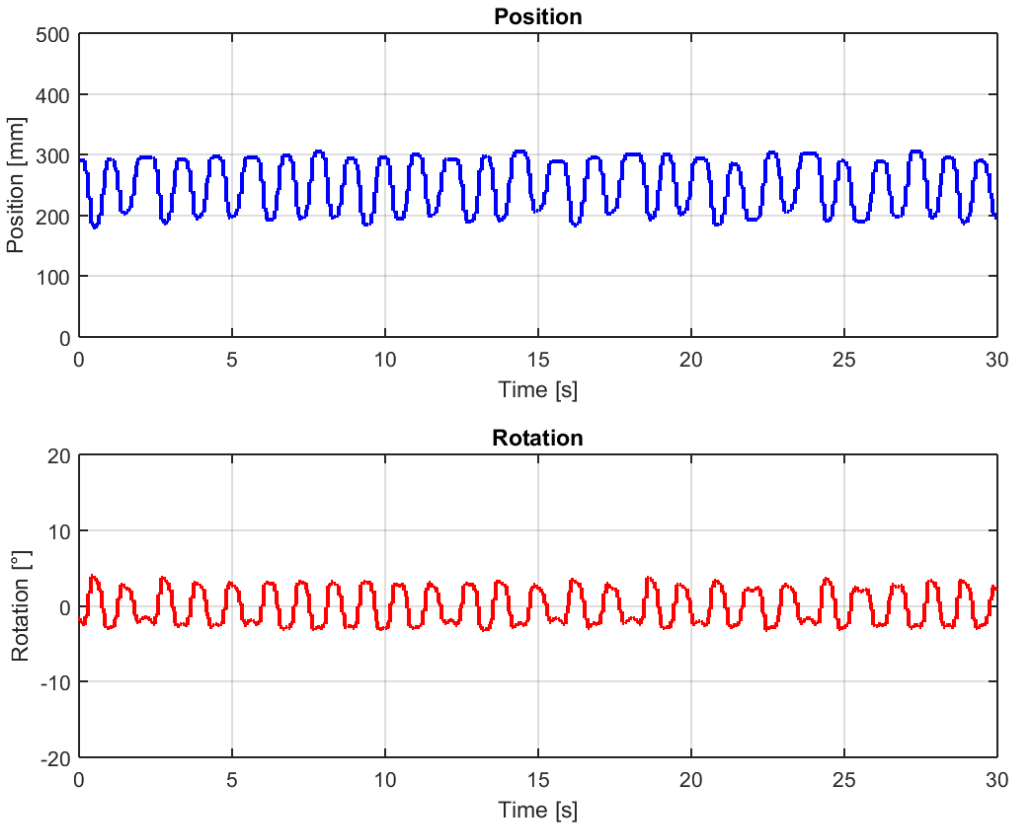


Fig 79: Grafico del funzionamento normale, Rockwell, con guadagni posizione kp=0,015; ki=0,003; kd=0,005; guadagni rotazione kp=6; ki=19; kd=0,01

Si intuisce già da questo andamento che il controllo è al limite e che probabilmente basterebbe un disturbo esterno per squilibrarlo.

Provando a disturbare il sistema con una forza esterna, in questo caso una forza che si opponeva in verso al movimento del carrello, si ha la divergenza del sistema verso l'instabilità.

Si vede che dopo il disturbo, il sistema ha oscillazioni sempre più ampie, caratterizzati da momenti incerti (tratti dove lo spostamento è costante al variare del tempo) dove il carrello si arresta per un certo periodo per avere un successivo spostamento veloce e ancora più ampio di volta in volta.
Chiaramente l'entità della sollecitazione fornita dall'esterno è stata più grande di quella che il sistema, controllato con il PLC, possa permettere.
Interessante è la parte finale, verso i 18 secondi, si vede che il carrello raggiunge la battuta di destra, successivamente l'asta del pendolo viene sbilanciata verso destra rimbalzando sulla battuta fino a fermarsi dopo qualche secondo di oscillazione.
Si nota che il carrello, prima di arrivare alla battuta si ferma un attimo, ma l'asta continua la sua rotazione verso destra, facendo sì che il carrello debba muoversi ulteriormente per recuperare, ma purtroppo la corsa è finita.

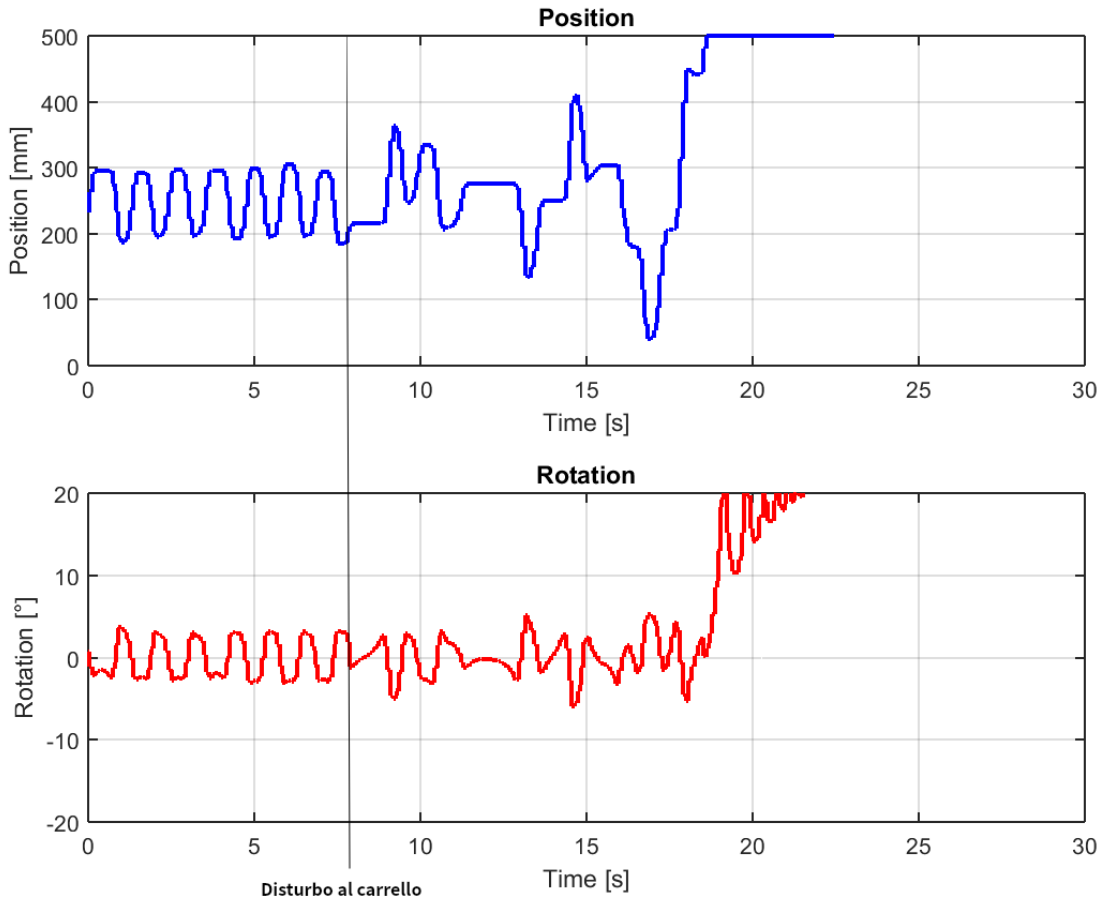


Fig 80: Grafico del funzionamento con disturbo al carrello, Rockwell

Per quanto riguarda il PLC, la conclusione è che il controllo è al limite per questa applicazione.
I valori dei guadagni sono stati scelti per ottenere i migliori risultati, ma purtroppo il controllo continua a far oscillare molto il sistema e non resiste a disturbi esterni.
Il PLC è stato impostato al massimo delle sue capacità, riducendo i tempi di lettura e scrittura dei moduli ad 1ms.
Guardando il campionamento fatto a 10ms, si notano dei valori uguali su certi blocchi, questo porta a pensare che benché i moduli siano stati settati al massimo, il processore non sia abbastanza veloce. Questo comporta anche un controllo limitato e dalle scarse prestazioni.

Arduino:

Con Arduino il risultato è molto diverso, la scheda permette prestazioni superiori sia in termini di stabilità sia in termini di risposta al disturbo.
L'oscillazione attorno al set della posizione è molto ridotta, come anche l'oscillazione attorno alla posizione verticale dell'asta.

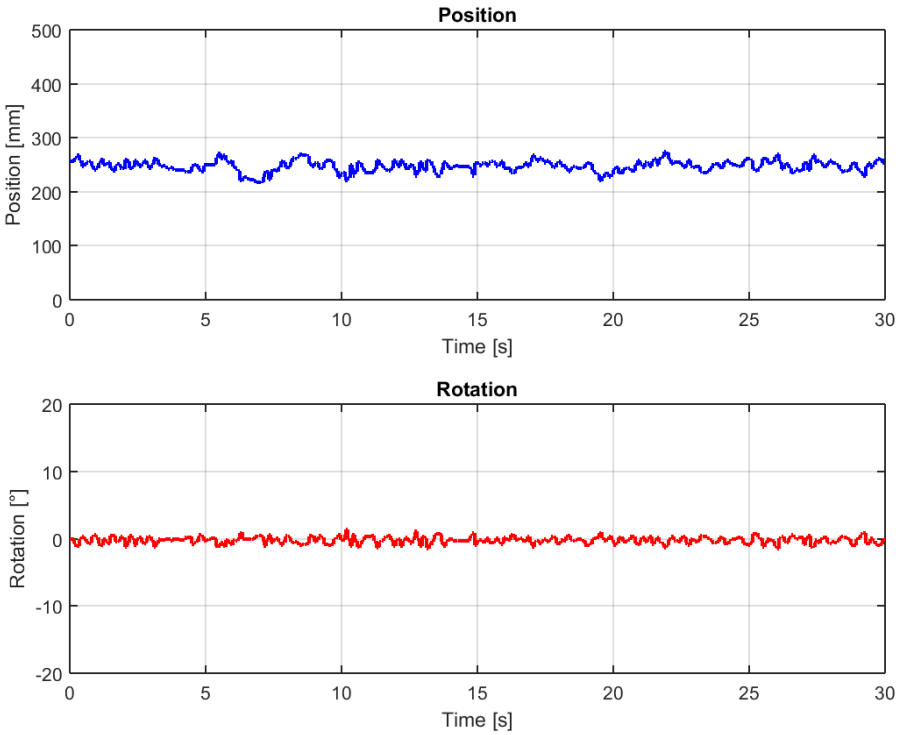


Fig 81: Grafico del funzionamento normale, Arduino, con guadagni posizione kp=0,015; ki=0,001; kd=0,005; guadagni rotazione kp=8; ki=25; kd=0,5

Facendo uno zoom si vede che l'ampiezza picco picco di oscillazione del carrello è di circa 2cm,

mentre l'ampiezza picco picco della rotazione è circa 2°.

Già questo è indice di un controllo più performante rispetto a quello del PLC, ma conviene analizzare anche le risposte ai disturbi sul carrello e sull'asta:

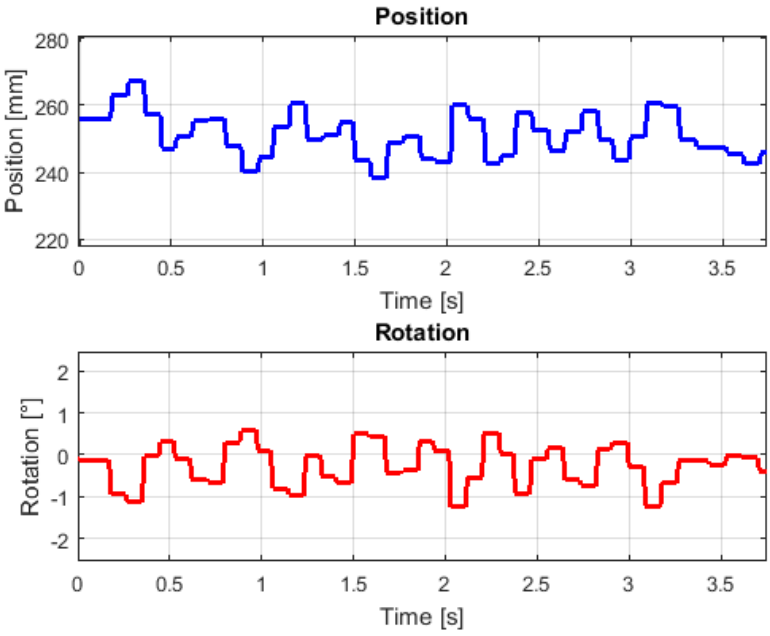


Fig 82: Zoom del grafico precedente, funzionamento normale

Nell’immagine a fianco si può vedere la risposta del sistema quando si disturba il carrello.
Ci sono delle oscillazioni ma di entità limitata ed il sistema recupera molto velocemente.

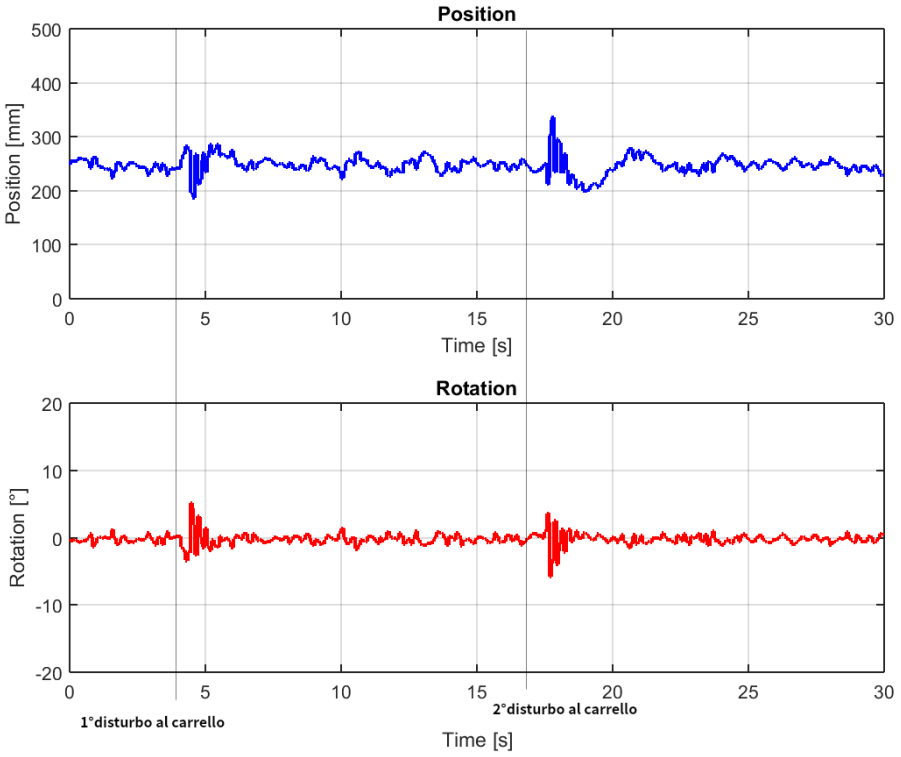


Fig 83: Grafico con due disturbi al carrello, Arduino

Il disturbo all’asta invece è più risentito dal sistema, infatti c’è bisogno di un’escursione maggiore del carrello per recuperare.

Si può notare questa differenza anche in termini temporali, il sistema ci mette di più per tornare alla condizione esistente prima del disturbo.

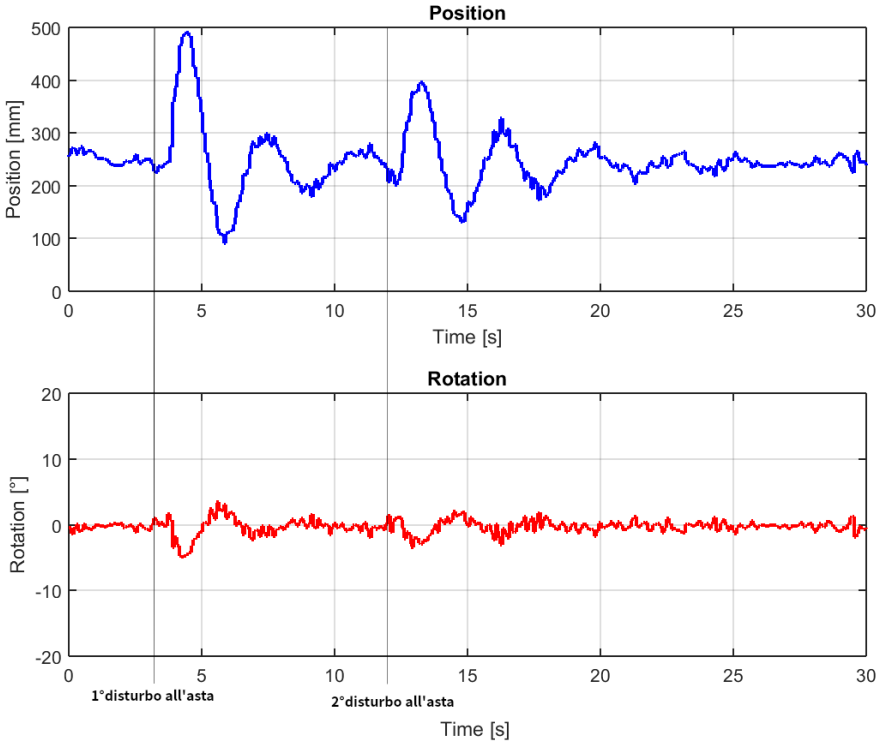


Fig 84: Grafico con due disturbi all’asta, Arduino

I disturbi sono stati accettati molto bene dal sistema, si prova ora a cambiare set con uno scalino su differenti valori.

I set sono stati cambiati utilizzando entrambe le direzioni, così da poter analizzare la risposta del sistema in condizioni di variazioni simili ma con verso di spostamento differente.
Questo può essere interessante in quanto la risposta non sarà uguale data la diversa geometria del cilindro pneumatico, che nella camera anteriore contiene lo stelo, dunque il volume è ridotto rispetto alla camera posteriore. Un volume ridotto comporta una superficie ridotta sullo stantuffo, che a sua volta comporta una forza minore di quella esercitata dalla camera posteriore.

Di seguito si può vedere il grafico dove il set, impostato inizialmente a 10cm, viene portato a 30cm di colpo tramite un set a scalino (a circa 10 secondi) .
È una variazione molto violenta che porta ad una sovraelongazione iniziale di 12cm oltre il valore di set (vengono raggiunti dunque 30cm+12cm=42cm), ma dopo qualche secondo il sistema si stabilizza nuovamente (a circa 17 secondi).

Dopo il set si nota che anche l'angolo dell'asta inizia ovviamente ad oscillare, raggiungendo un massimo di 5°, ma torna stabile dopo qualche secondo.

La pressione utilizzata per questa prova è di 6.8 bar.

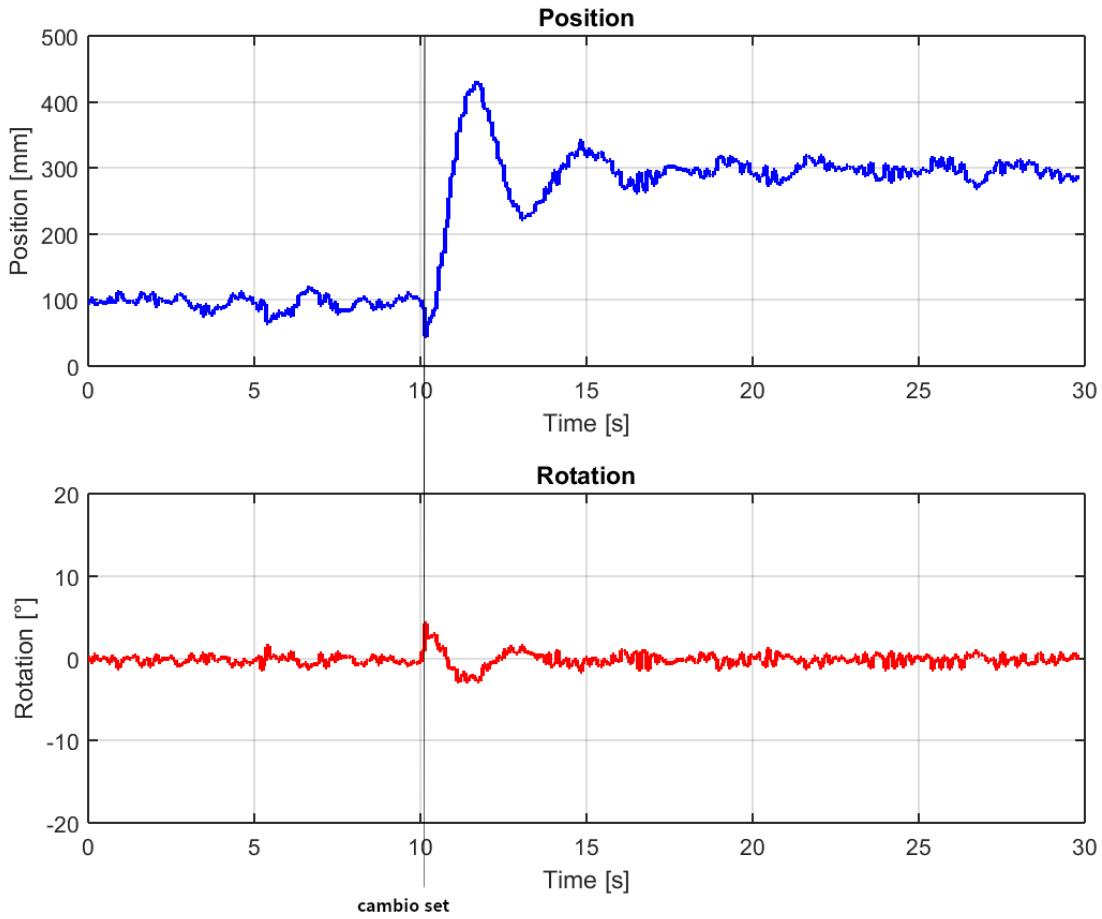


Fig 85: Grafico con cambiamento di set da 10cm a 30cm, il carrello si sposta verso destra

Partendo dal centro del carrello, ovvero a 25cm, viene imposto il set a 40cm.
Si può vedere che, seppur al limite, il sistema recupera e si stabilizza successivamente.

Il carrello si muove verso destra, viene dunque utilizzata la camera posteriore del cilindro

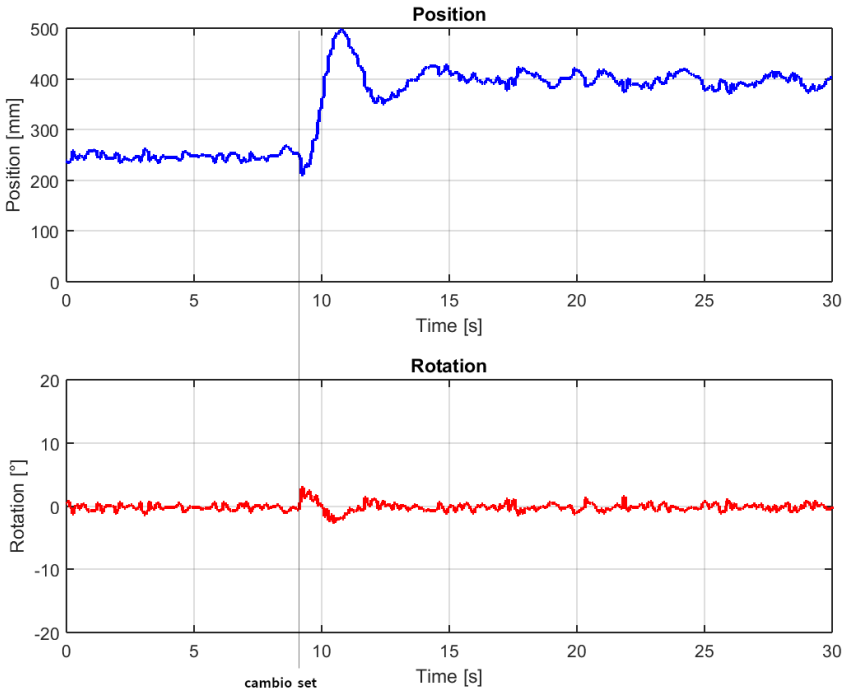


Fig 86: Grafico con cambiamento di set da 25cm a 40cm

Partendo dal centro e impostando la stessa ampiezza di variazione del set (si passa da 25cm a 10cm), ma in direzione opposta, si vede che il sistema non ce la fa.
Il carrello arriva a fine corsa e l'asta rimbalza.

Qui si nota molto bene il risultato della differente geometria del cilindro pneumatico.

La camera anteriore ha meno superficie di spinta, si genera una minor forza che porta ad una minore accelerazione del carrello

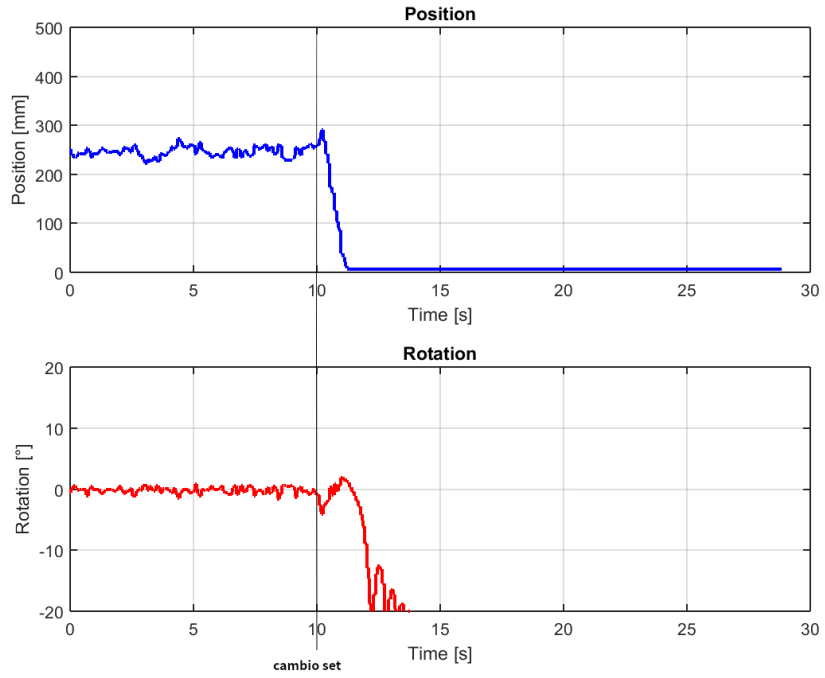


Fig 87: Grafico con cambiamento di set da 25cm a 10cm, pressione 6.8 bar

Per confermare ciò che è stato appena detto, si può provare ad aumentare la pressione del sistema, portandosi da 6.8 bar a 7.5 bar, cioè oltre il valore consentito dalle valvole.

L'aumento della pressione oltre il valore limite comporta la non chiusura totale delle valvole anche quando non sono alimentate, questo perché la molla non riesce a tener chiuso l'otturatore e un flusso d'aria riesce a fuoriuscire.

Questa prova è fatta solamente per vedere se il sistema recupera rispetto alla prova con variazione di set tra gli stessi valori.

I risultati ottenuti confermano il pensiero precedente, cioè la maggior pressione riesce ad esercitare più forza, che a sua volta comporta una maggior accelerazione del carrello.

La maggior accelerazione permette al pendolo di trovare nuovamente una posizione di equilibrio senza andare a sbattere con il carrello a fondo corsa.

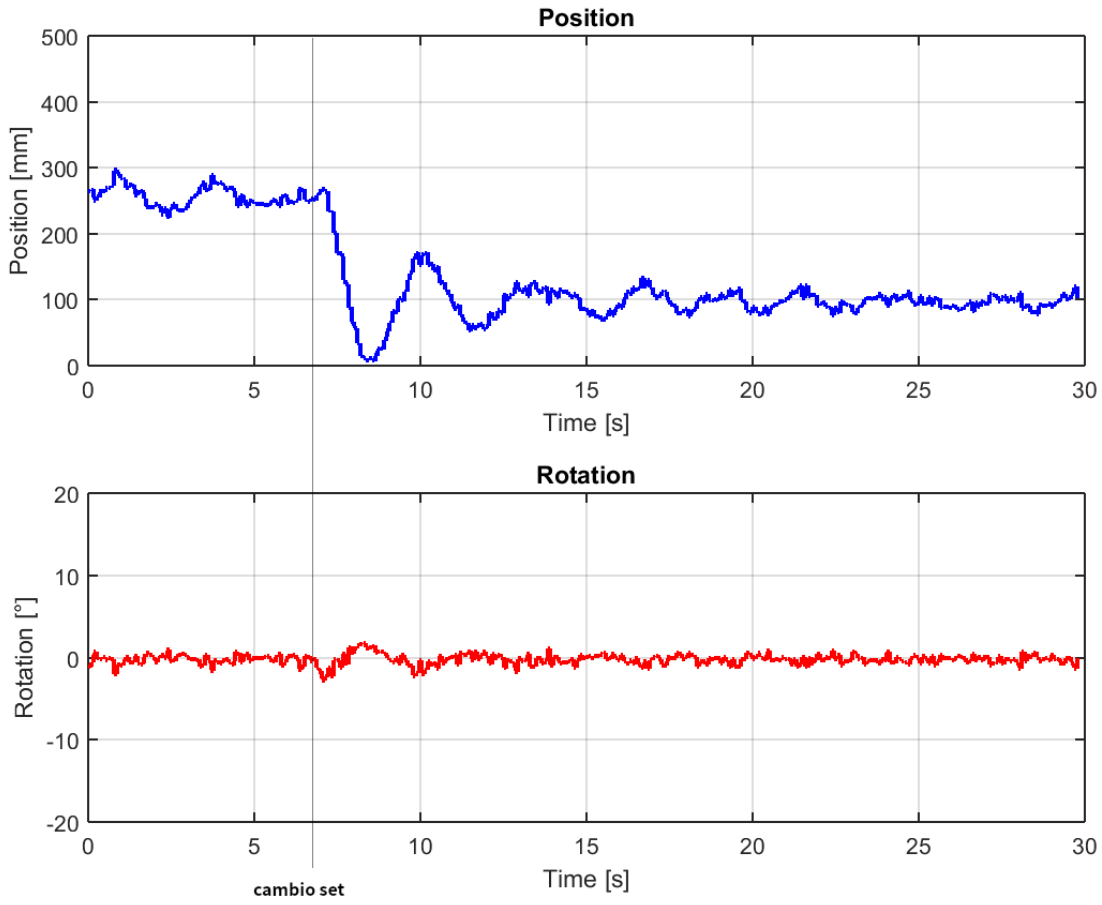


Fig 88: Grafico con cambiamento di set da 25cm a 10cm, pressione 7.5bar

Le variazioni di set, anche se importanti, comportano variazioni di rotazione comunque ridotte, perché il sistema vuole sempre mantenere l'equilibrio del pendolo.

L’effetto della differente geometria può esser notato anche su variazioni ridotte del set.

Qui ad esempio si sposta il set dal centro del carrello a 30cm, la variazione è abbastanza lineare e viene eseguita in poco tempo.

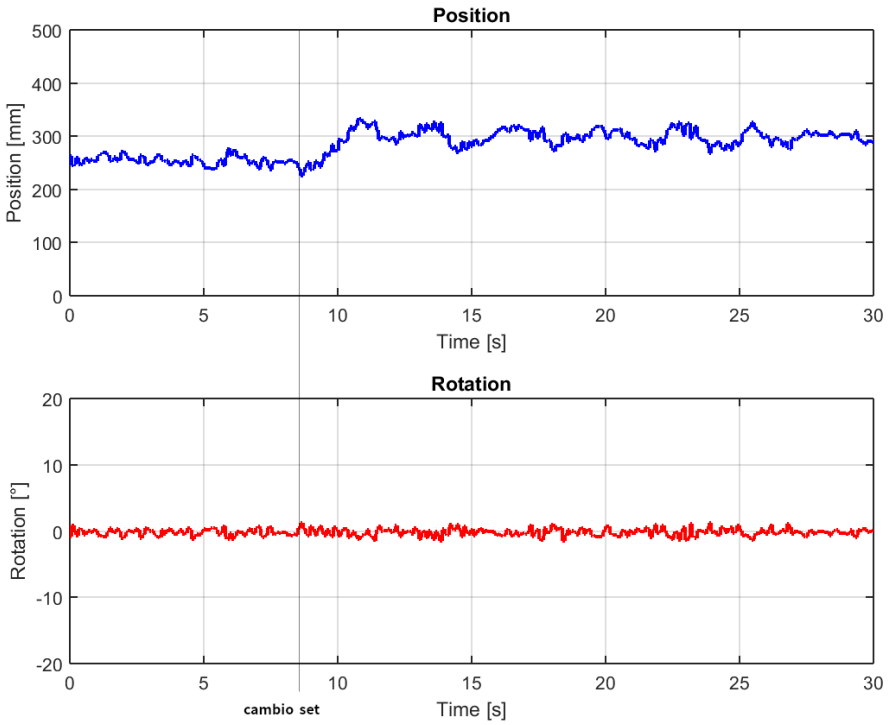


Fig 89: Grafico con cambiamento di set da 25cm a 30cm

Partendo nuovamente dal centro del carrello, si imposta questa volta un set di 20cm.

La risposta sembra essere un po’ più lenta rispetto alla precedente.

Si nota che le oscillazioni a regime sono un po’ più piccole rispetto al caso precedente perché i volumi di lavoro sono differenti rispetto a quelli precedenti.

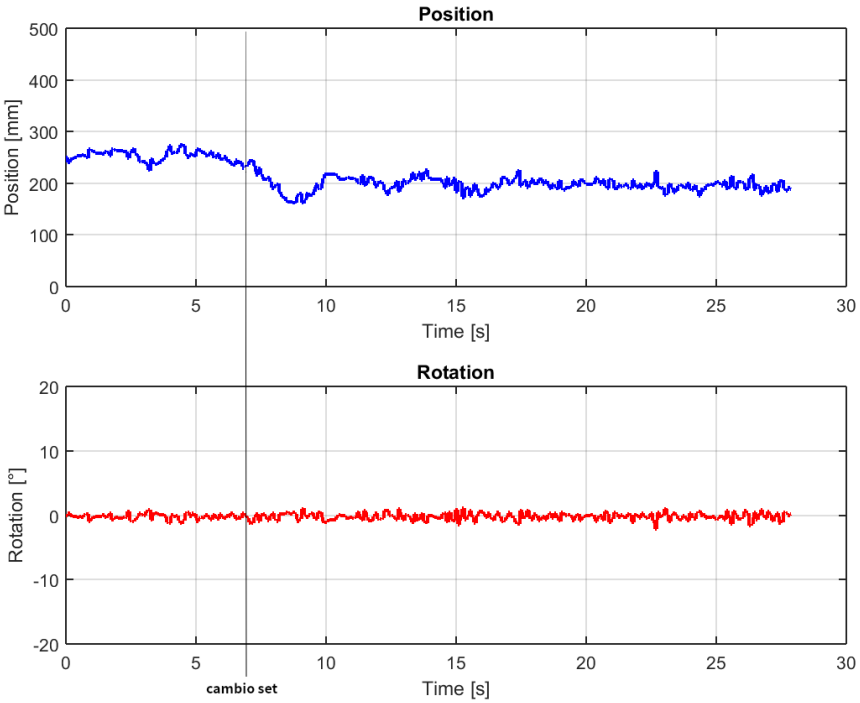


Fig 90: Grafico con cambiamento di set da 25cm a 20cm

Si può pensare di far seguire al carrello un set sinusoidale, che abbia come centro 25cm e un'ampiezza di di 10cm.
Il set sarà dunque una sinusoide con valore minimo a 15cm e valore massimo a 35cm.

Le variazioni sono importanti ma il sistema ha un certo tempo per elaborarle, soprattutto sono variazioni continue nel tempo che spostano il set.

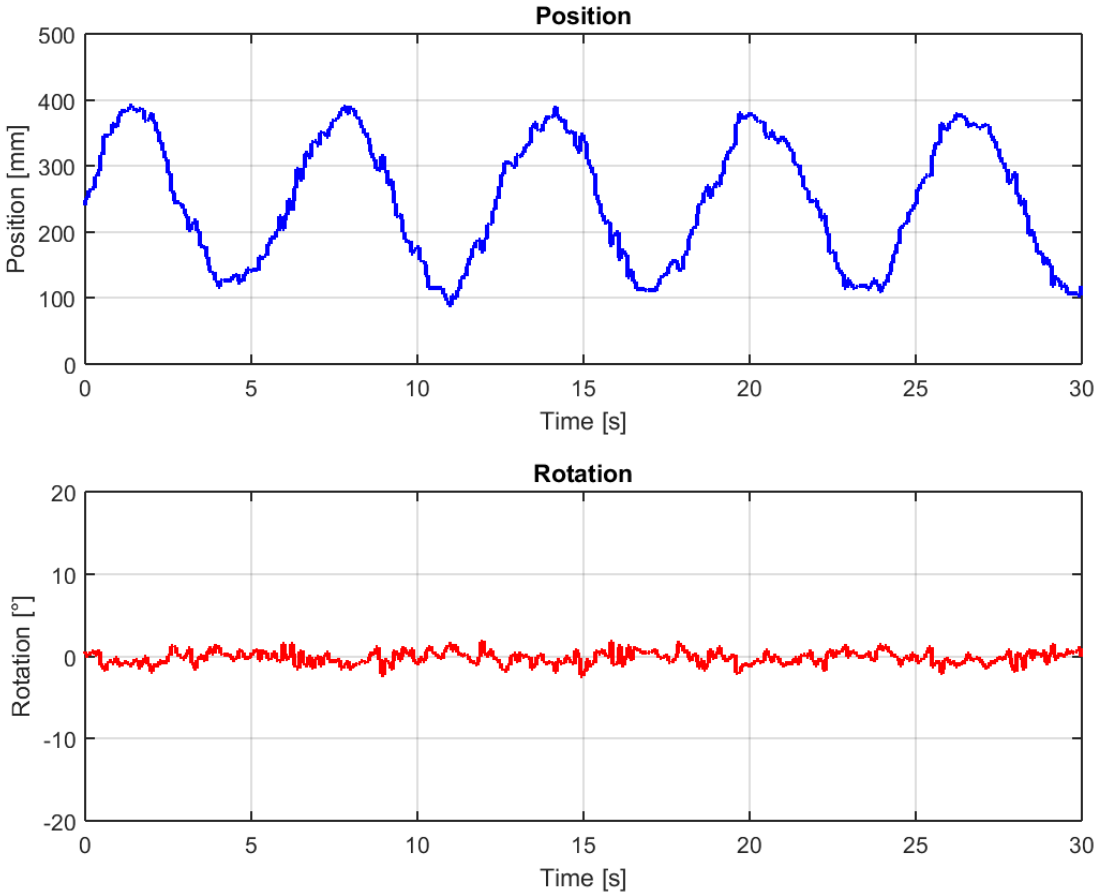


Fig 91: Grafico con set sinusoidale

Il sistema continua a rispondere bene e anche le variazioni di rotazione sono nell'intorno di qualche grado.

17. Acquisizioni aggiuntive per il confronto della variazione dei parametri

Le seguenti prove sono state condotte ad una pressione di 6 bar, è interessante notare come la variazione di singoli parametri (proporzionale, integrativo e derivativo), porta a risposte completamente diverse del sistema, giustificando la scelta dei parametri ottimali descritti in precedenza.
I parametri che verranno variati sono relativi all'anello della theta, in quanto è l'anello che genera il comando finale per le valvole ed è quello "più vicino" in termini di controllo. Si è scelta questa modalità in quanto fare un confronto con la variazioni di sei parametri (tre parametri per anello posizione e altri tre per l'anello rotazione) portava ad avere molte combinazioni possibili, troppe a dire il vero per capire l'effetto delle singole variazioni. I parametri dell'anello posizione dunque rimarranno i seguenti in tutte le prove: kp=0,015, ki=0,001, kd=0,005.

La **prima rilevazione** interessante da rappresentare è la seguente, condotta con i guadagni standard (ovvero sull'anello theta kp=8, ki=25, kd=0,5)

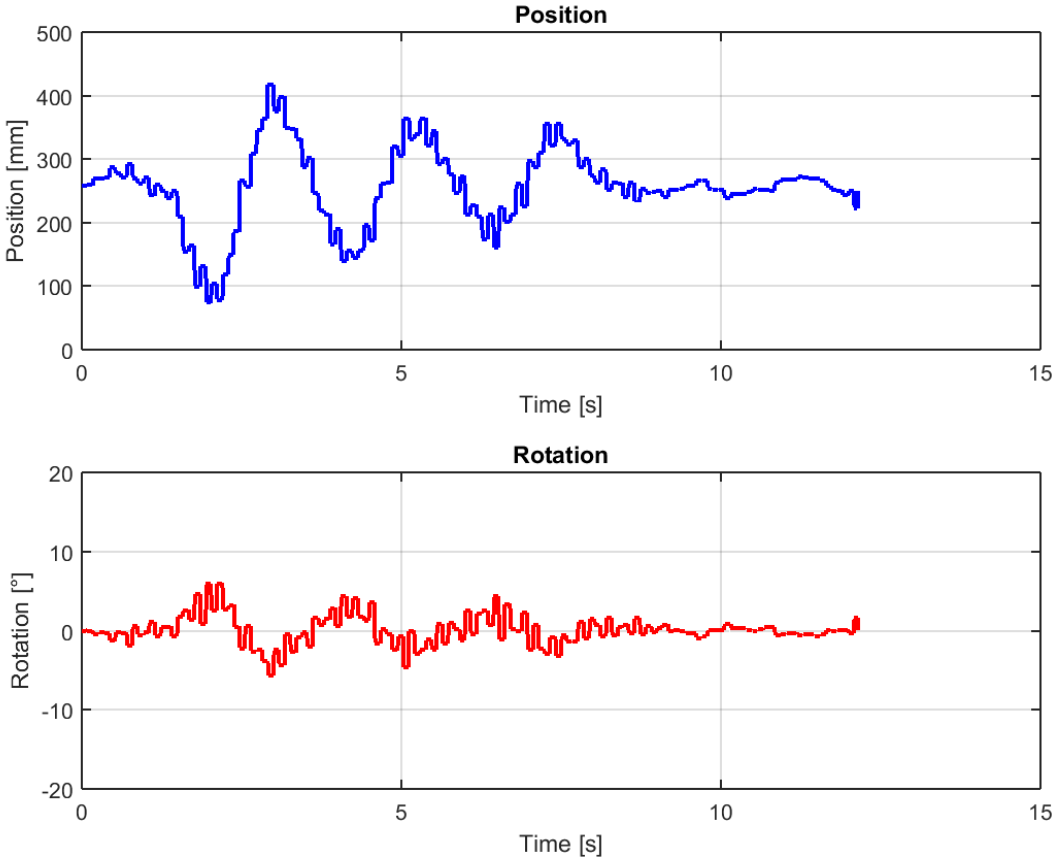


Fig 92: Prova con guadagni standard

Questa acquisizione è stata fatta posizionando il carrello a centro corsa, tenendo il pendolo in posizione verticale e avviando l'acquisizione quando Arduino inizia il controllo, come si può vedere c'è un transitorio dove il sistema di stabilizza (questo perché manualmente il centro carrello come la verticale del pendolo non erano perfette, quindi il sistema di controllo ha calcolato spostamenti

iniziali non nulli) e via via riduce le oscillazioni sia di corsa che di rotazione. Questo è il transitorio di un sistema stabile per definizione.
Con il sistema stabilizzato si è registrata la posizione e la rotazione, in questa rilevazione non ci sono disturbi meccanici al sistema, si nota dunque un certo mantenimento della posizione e della rotazione.

Anche per questa **seconda prova** i guadagni utilizzati sono quelli standard:

Il sistema a regime non viene disturbato e si nota una certa stabilità.

L'oscillazione massima del carrello, ad una pressione di alimentazione di 6 bar è circa 80mm picco picco

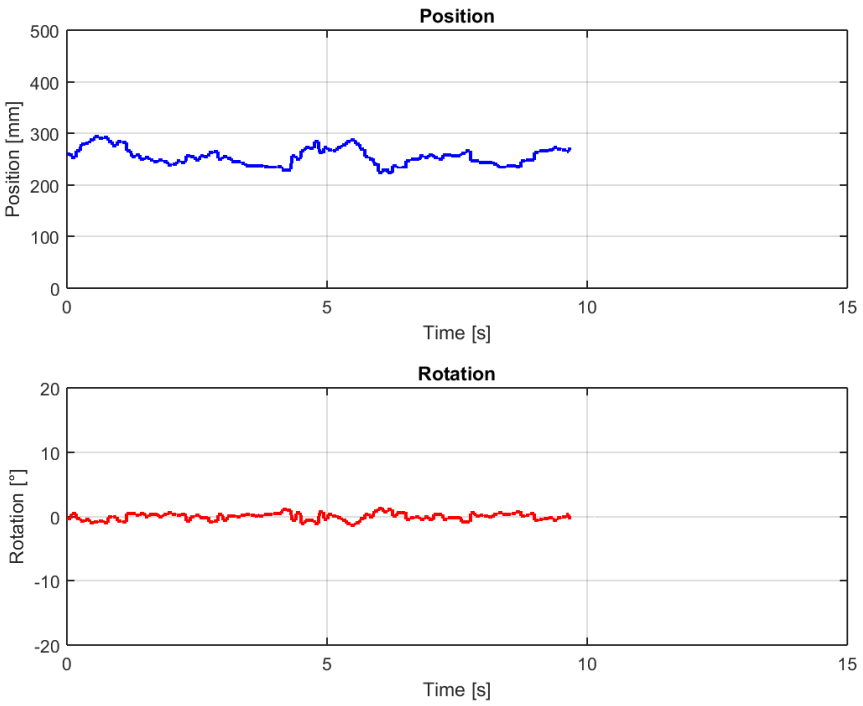


Fig 93: Guadagni standard, dopo transitorio

Rispetto alle prove effettuate a 6,8 bar si può vedere la differenza, ovvero il sistema, a 6 bar, è meno preciso e le oscillazioni sono un po' più grandi.

La **terza prova** è stata effettuata dimezzando il guadagno proporzionale (ora kp=4), mantenendo gli altri guadagni costanti, qui è interessante notare che seppur il sistema sia stato fatto partire in condizioni simili alle partenze precedenti, dopo un transitorio il sistema non riesce a mantenere l'equilibrio e risulta dunque instabile in quanto le oscillazioni, sia di posizione che di rotazione, si amplificano col passare del tempo.

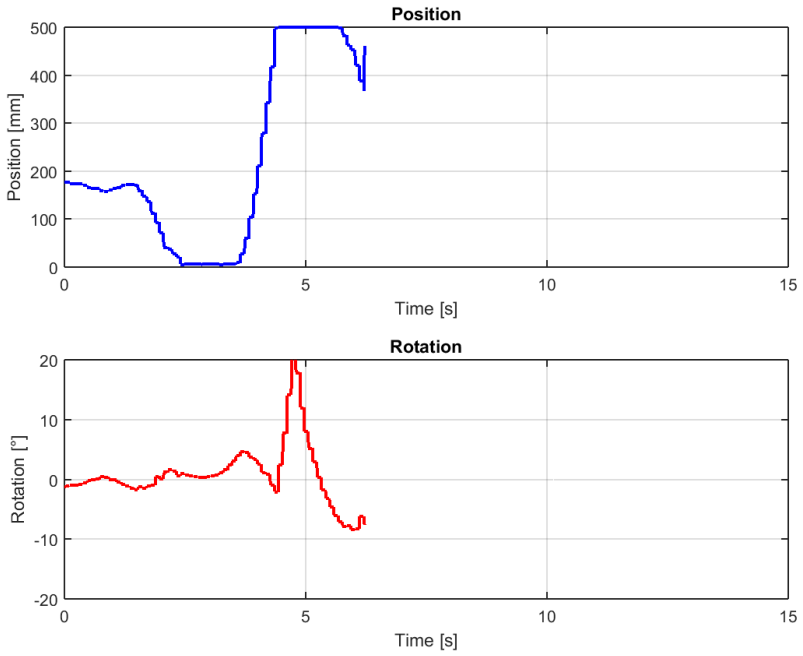


Fig 94: Guadagno proporzionale dimezzato

Per la **quarta prova** si è raddoppiato il guadagno proporzionale (ora kp=16) e gli altri parametri sono stati mantenuti costanti:

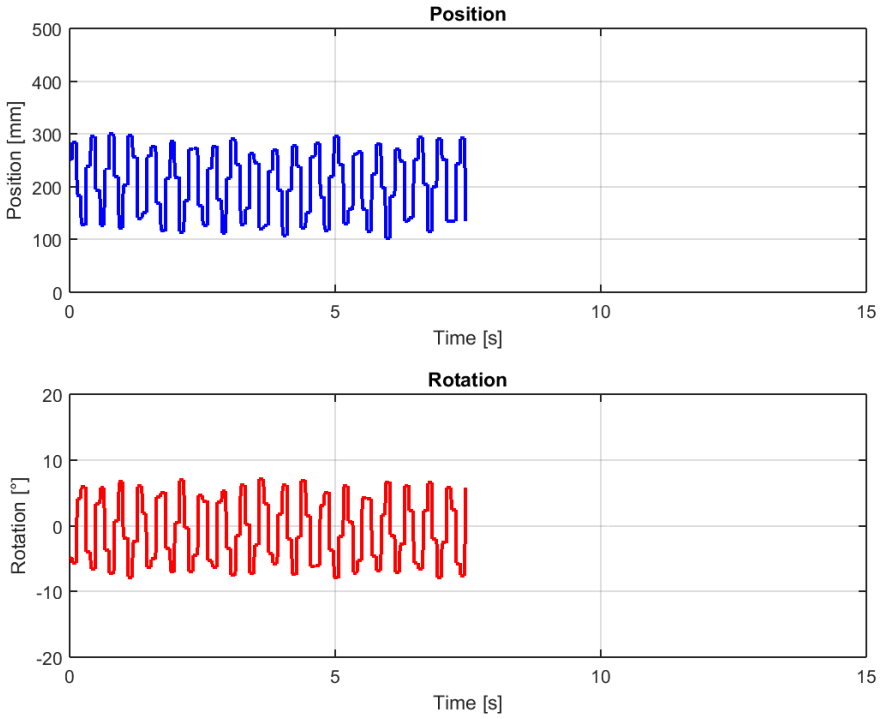


Fig 95: Guadagno proporzionale raddoppiato

Si nota che il sistema è nervoso, oscilla attorno alla posizione di set ma le oscillazioni sono troppo elevate, dunque questo guadagno proporzionale impostato è troppo alto.

Nella **quinta prova** è stato raddoppiato il guadagno integrale (ora ki=50), portando gli altri guadagni ai valori standard:

Anche qui si vede che le oscillazioni sono comunque più ampie rispetto alle oscillazioni ottenute con guadagni standard, il periodo dell'oscillazione tuttavia è diventato più grande rispetto al caso precedente dove si era raddoppiato il kp.

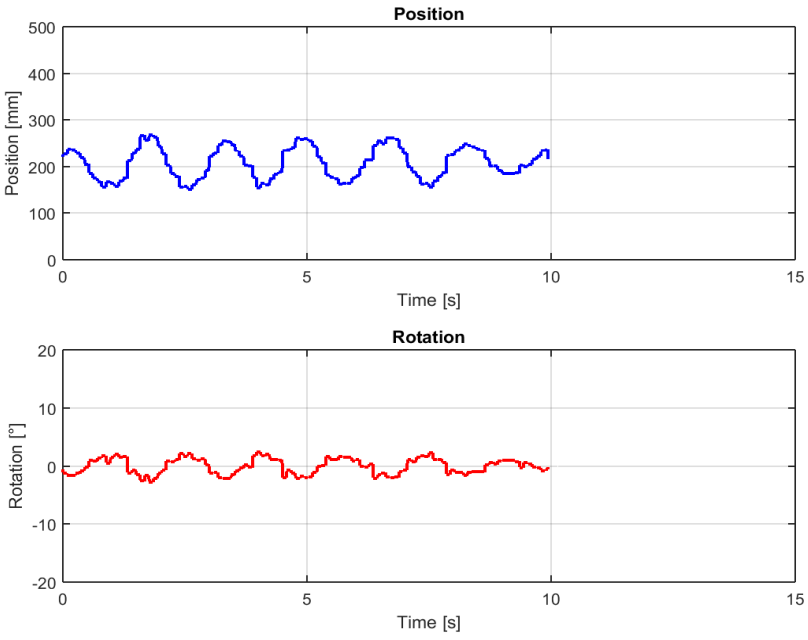


Fig 96: Guadagno integrale raddoppiato

Per la **sesta prova**, non sono stati variati i parametri dal caso precedente, ma si è introdotto un disturbo sull'asta per vedere come il sistema rispondeva:

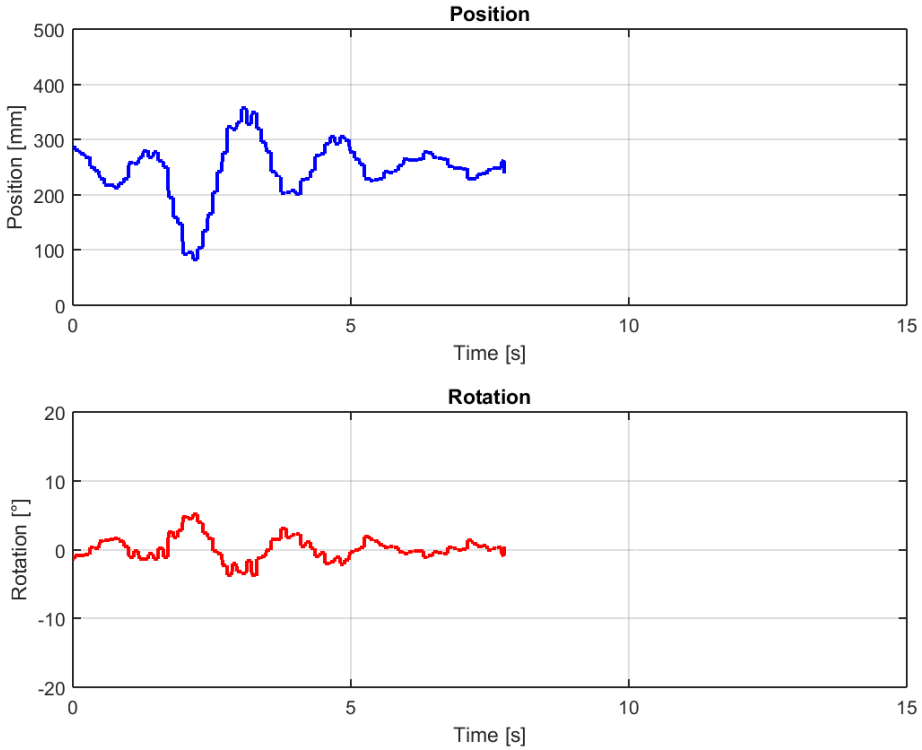


Fig 97: ki=50, disturbando l'asta

Si può notare qui il disturbo a 2 secondi circa (essendo un disturbo all'asta si guarda quando vi è una variazione importante sulla rotazione rispetto al trend precedente, lì è dove è stato applicato il disturbo). Il sistema torna ad essere stabile ma le oscillazioni a transitorio finito risultano essere più ampie rispetto al sistema con parametri standard.

La **settima prova** invece è stata condotta con un guadagno integrativo (ki=10), senza disturbo:

Si può notare come il sistema, seppur con poche oscillazioni, finito il transitorio non è ancora arrivato alla posizione di set del centro carrello di 250mm.

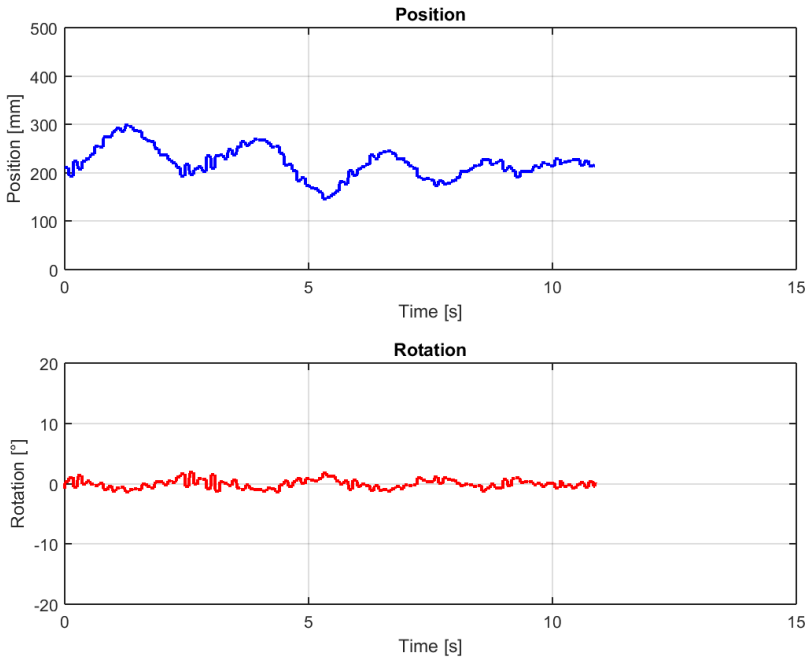


Fig 98: Guadagno integrale ridotto

L’ottava prova è stata effettuata con un disturbo del sistema sull’asta, mantenendo i parametri della settima prova. Come si può notare dal grafico sottostante, a circa mezzo secondo si nota una variazione importante sulla rotazione e di conseguenza una variazione sulla posizione, con successiva stabilizzazione del sistema.

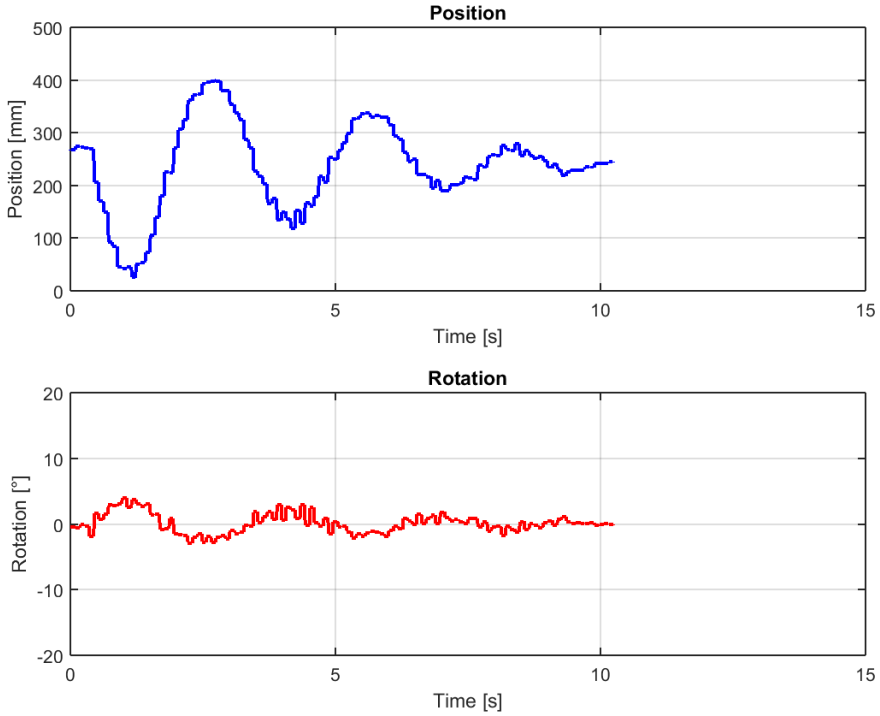


Fig 99: ki=10, disturbando l'asta

Per la **9^ prova** sono stati impostati i parametri standard (kp=8, ki=25, kd=0,5) disturbando il sistema con un piccolo impulso sull’asta a circa 2 secondi:

Il sistema risponde bene smorzando le oscillazioni abbastanza velocemente per arrivare infine ad una condizione stabile.

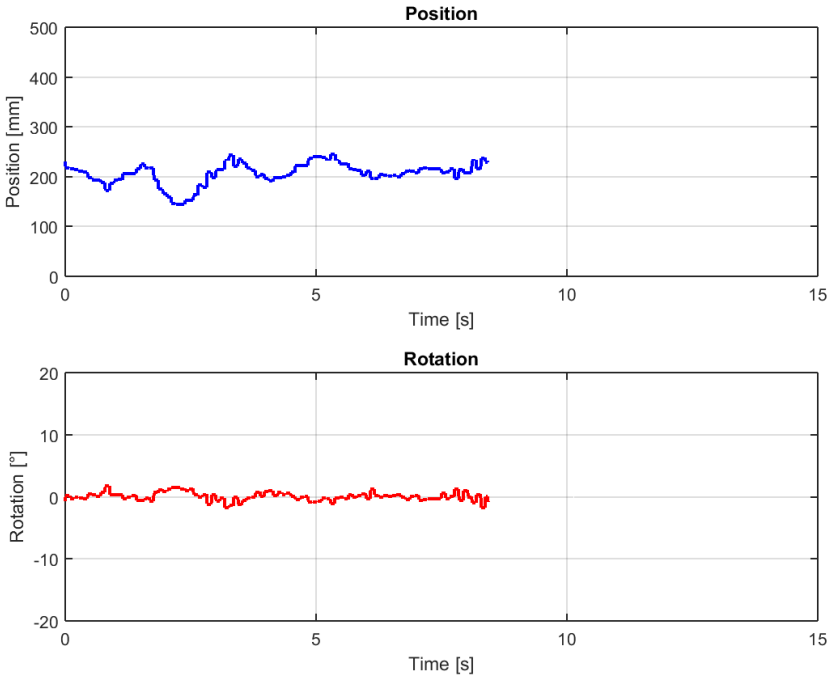


Fig 100: Guadagni standard, disturbo piccolo all'asta

La **10^ prova** si differenzia dalla prima per l'entità del disturbo, qui infatti il disturbo a circa 2 secondi è così importante che il sistema sfrutta quasi tutta la corsa disponibile per recuperare:

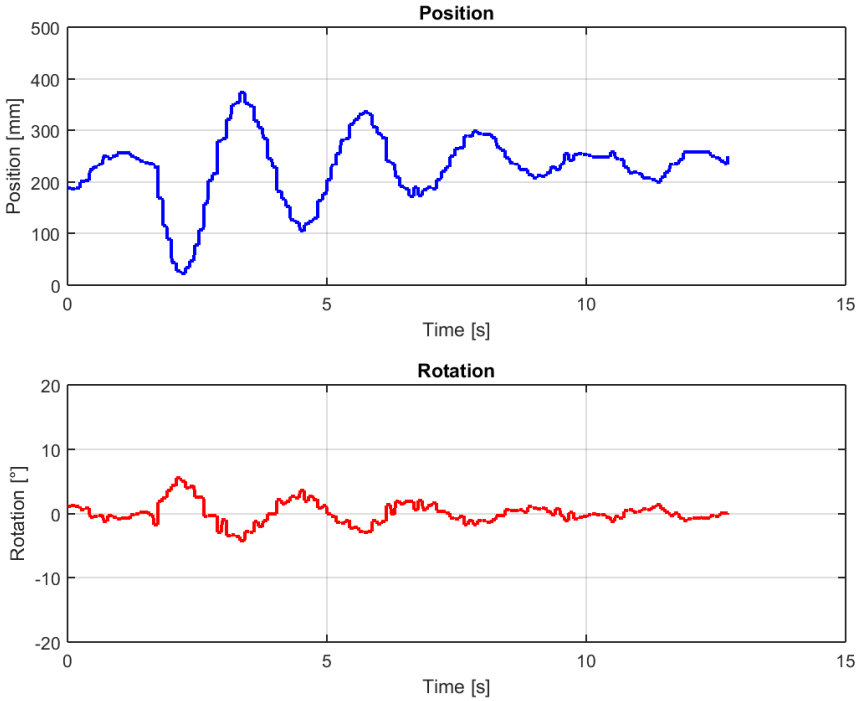


Fig 101: Guadagni standard, grande disturbo all'asta

L'11^ prova è stata effettuata aggiungendo un ulteriore peso di 250g sull'asta (complessivamente ora il peso è di 500g in punta all'asta), come si può vedere la situazione non cambia di molto rispetto al peso standard di 250g, si nota un'escursione leggermente superiore sulla posizione.
Il sistema di controllo è molto sensibile a disturbi elettrici esterni, in laboratorio infatti la rete elettrica condivisa ha variazioni sulla tensione di alimentazione che si ripercuotono su disturbi nel feedback quindi si hanno risposte un po' diverse nel tempo, come si può notare verso 6 secondi.

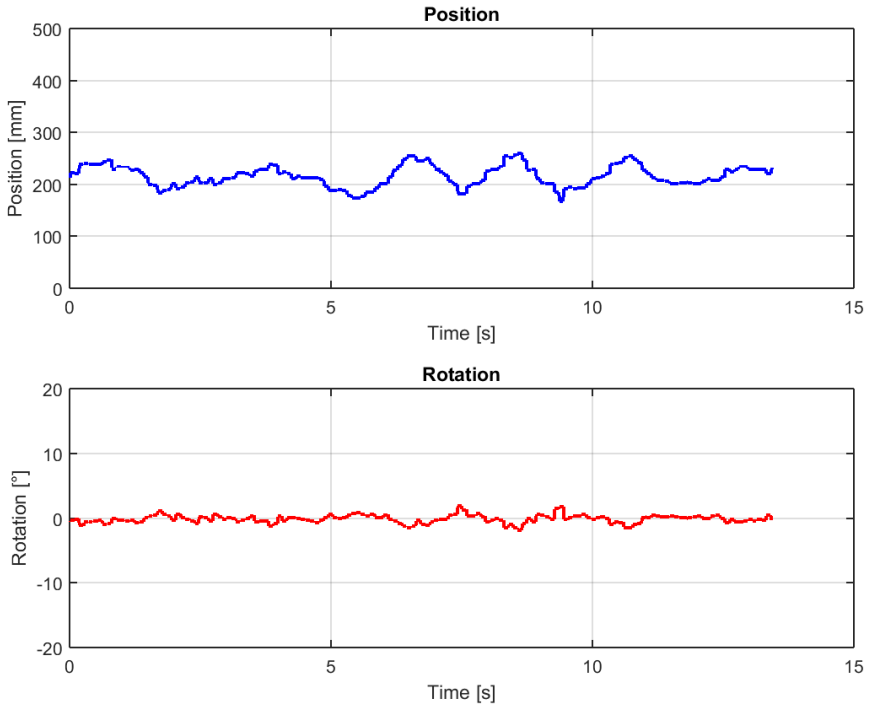


Fig 102: Massa 500g, guadagni standard

Nella **12^ prova** è stato disturbato il sistema rispetto alla prova n.11, ovvero guadagni standard e peso totale di 500g. Il disturbo è applicato a circa 2 secondi.

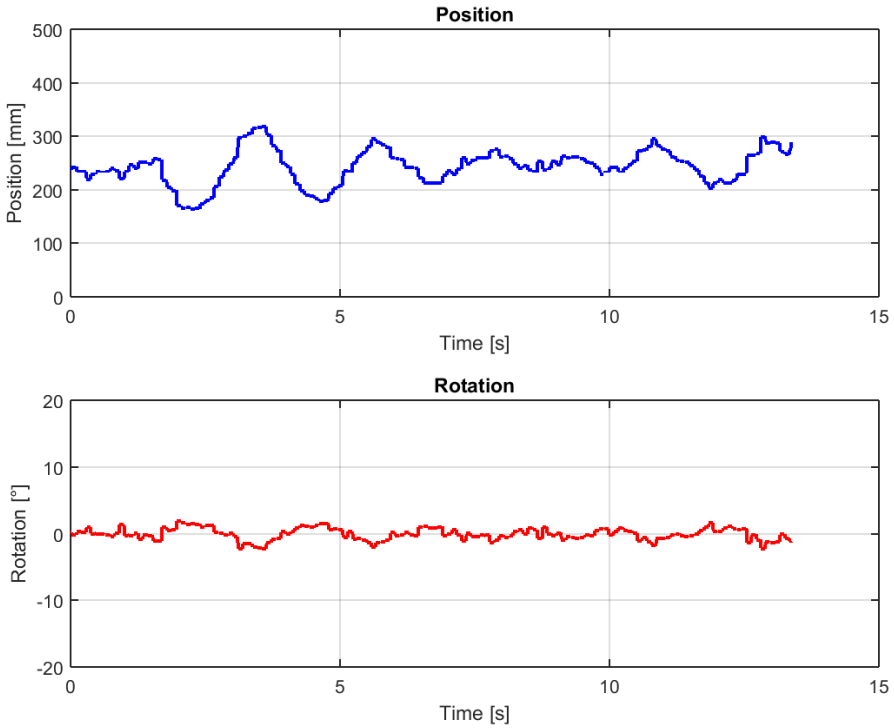


Fig 103: m=500g, con piccolo disturbo all'asta

La **13^ prova** è come la 12^ a differenza del fatto che il disturbo applicato ha intensità maggiore, come si può notare rispetto alla prova n.10, lo stesso disturbo influenza meno il sistema, questo è dovuto alla massa, essendo più importante l'accelerazione risultante sul sistema è minore, dunque il sistema di controllo riesce a recuperare con una corsa minore.

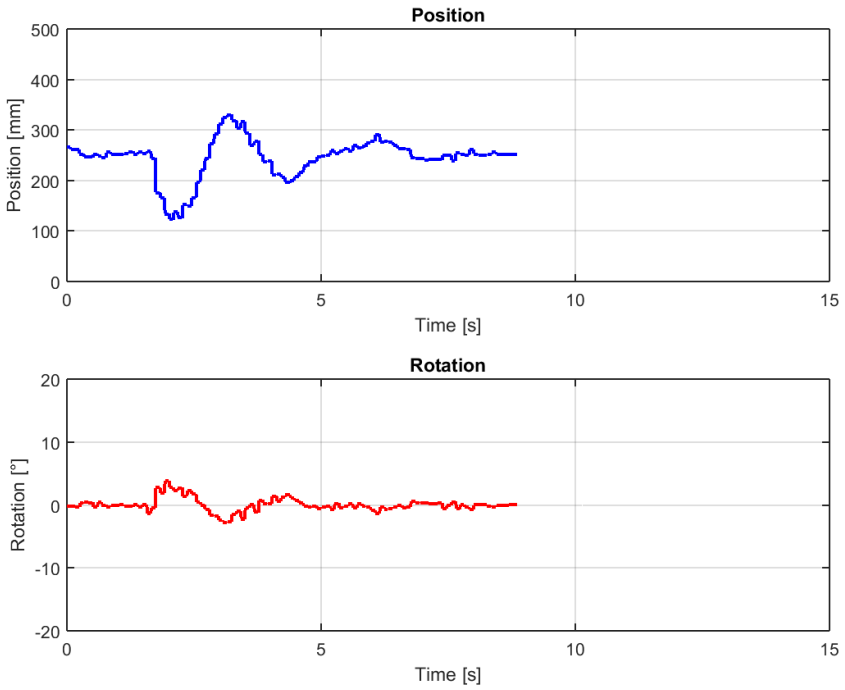


Fig 104: m=500g, grande disturbo all'asta

Per la **14^ prova** (guadagni standard e peso standard) è stato impostato un set variabile sinusoidale, nel programma definito come centro_carrello = 2.5 + sin(millis() / 1000);
dove millis() è una funzione di Arduino che conta i millisecondi

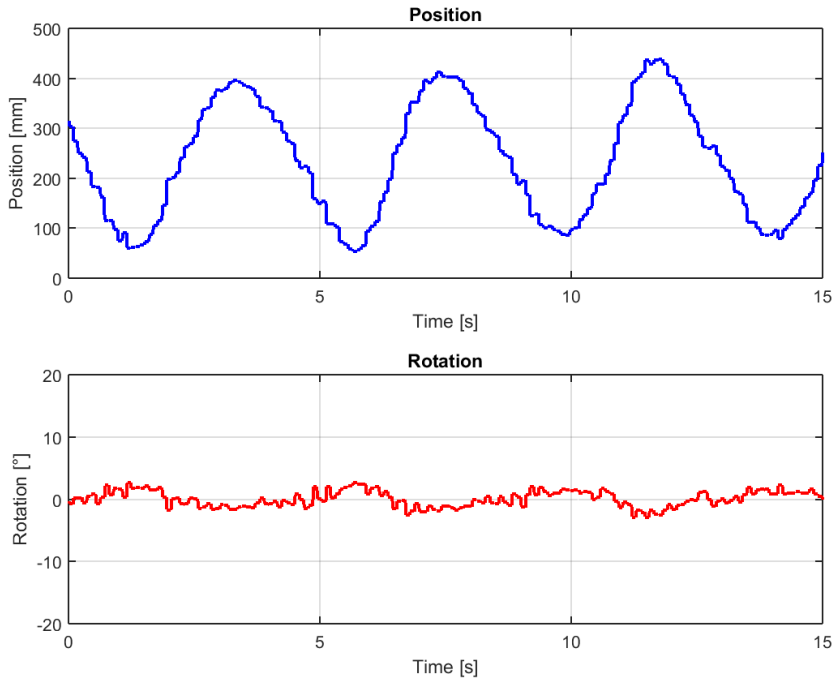


Fig 105: Set sinusoidale

Si può notare che il periodo di oscillazione è circa 4 secondi.
Nonostante questo set sinusoidale il sistema riesce a mantenere l'asta in equilibrio, intorno alla posizione verticale.

Si può notare che il set varia come $250 \pm 100\,mm$ quindi l'estremo superiore di set sarebbe 350mm mentre quello inferiore di 150mm.
Data la natura meccanica del sistema, tenendo conto anche delle inerzie e del controllo stesso, si va oltre questi valori, toccando sia i 100mm inferiori sia i 400 superiori.

Si può notare anche una tendenza crescente della sinusoide come angolazione al variare del tempo, questo è dovuto che il sistema raggiunge poco per volta il centro di oscillazione a 250mm.
Verosimilmente se l'acquisizione fosse andata avanti, si avrebbero le stesse ampiezze di oscillazione nell'intorno dei 250mm.
Questa non simmetria del transitorio è dovuta all'istante iniziale quando il pendolo viene rilasciato a mano, pur facendo attenzione, non si riesce mai ad avere la perfetta posizione, sia per la X sia per la Theta.

La **15^ prova** è stata effettuata cambiando set, ovvero raddoppiando la frequenza di oscillazione e dividendo la funzione seno per 2.
ovvero si ha centro_carrello = 2.5 + 0,5*sin(2*(millis() / 1000));
il risultato è il seguente:

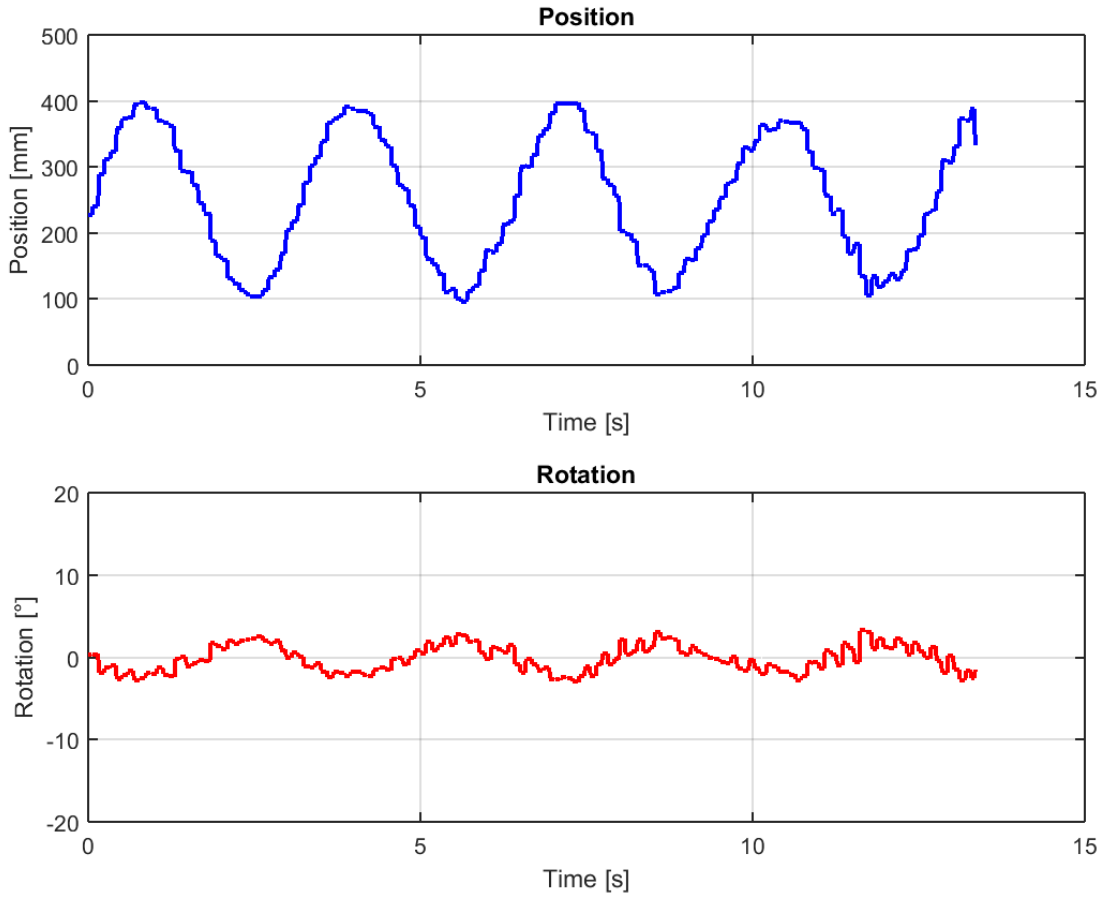


Fig 106: Periodo raddoppiato, ampiezza dimezzata

Si vede che il periodo della sinusoide è diminuito, ma a differenza della prova precedente è stato necessario dimezzare la funzione seno in quanto il sistema essendo più veloce aveva bisogno di più corsa per recuperare.
Se così non fosse stato fatto, il sistema non sarebbe riuscito a recuperare e alla prima sinusoide sarebbe andato a sbattere su uno dei due estremi del carrello.
Questo segna la banda passante del sistema, ovvero un periodo minore di 3 secondi è difficilmente mantenibile.

In conclusione:

Dai risultati ottenuti si può dire che Arduino sia più performante rispetto al PLC Rockwell.
C'è da ricordare che il PLC utilizzato è ormai datato e probabilmente con un altro PLC più moderno i risultati ottenuti sarebbero stati migliori.

18. Riferimenti

Pendolo inverso su ruote:
http://www.instructables.com/id/A-Simple-and-Very-Easy-Inverted-Pendulum-Balancing/

Segway:
http://www.segway.it/
https://it.wikipedia.org/wiki/Segway

Pendolo inverso:
https://en.wikipedia.org/wiki/Inverted_pendulum

Fuzzy:
https://en.wikipedia.org/wiki/Fuzzy_control_system

PID:
https://en.wikipedia.org/wiki/PID_controller

Arduino:
https://en.wikipedia.org/wiki/Arduino

Arduino Mega:
https://www.arduino.cc/en/Main/arduinoBoardMega2560/

Per i riferimenti e la programmazione tramite Arduino:
https://www.arduino.cc/reference

PWM:
https://en.wikipedia.org/wiki/Pulse-width_modulation
https://www.arduino.cc/en/Tutorial/PWM
Appunti lezioni Hydraulic servosystems – Massimo Sorli

MOSFET:
https://en.wikipedia.org/wiki/MOSFET

IRF520 PDF:
https://www.google.it/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwj89trNsrTaAhVLuxQKHRq6BRUQFggzMAA&url=https%3A%2F%2Fwww.vishay.com%2Fdocs%2F91017%2F91017.pdf&usg=AOvVaw1oDLFAy5LcWThKFazhx187

19. Ringraziamenti

Questo lavoro di Tesi ha richiesto molto tempo e molta dedizione, come giustamente dev'essere.

La sfida è stata impegnativa, in quanto un pendolo inverso con attuatore pneumatico è un sistema abbastanza raro che non si vede ovunque, il web è pieno di esempi sul pendolo inverso, ma pochissimi utilizzano questo tipo di attuazione e ancor meno sono quelli che ottengono risultati positivi.

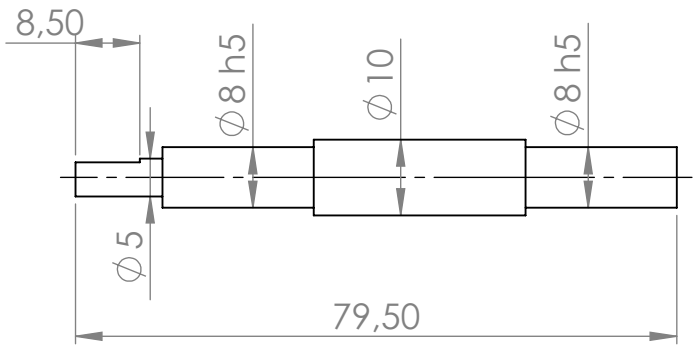
Ringrazio i miei Relatori, sempre disponibili e attenti.

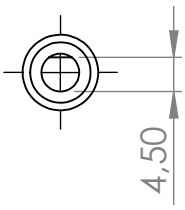
Un ringraziamento speciale anche per il Sig. Luca Garzone, tecnico Rockwell, che si è reso disponibile più volte nel chiarirmi dubbi riguardo alla programmazione e l'utilizzo del software, soprattutto nei primi periodi.

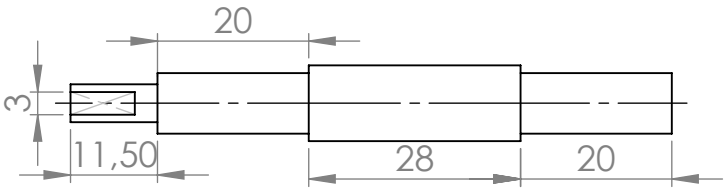
Si ringraziano gli amici, in particolare il "Doc" Elia Mameli, che mi ha aiutato nell'elaborazione dei grafici e tutti gli altri che con piccole frasi mi hanno sempre sostenuto.

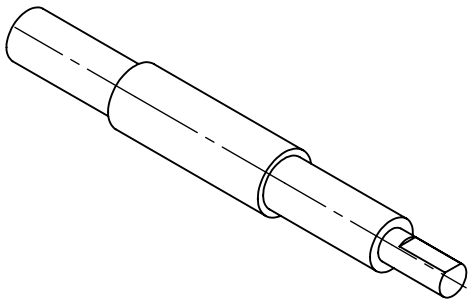
Un enorme ringraziamento va anche a mia Mamma, che mi ha sempre supportato (e anche sopportato!!) in tutti questi anni universitari; et merci à toi aussi, Papà, que depuis le ciel tu me regardes toujours, je sais que t'es là!

20. Allegati

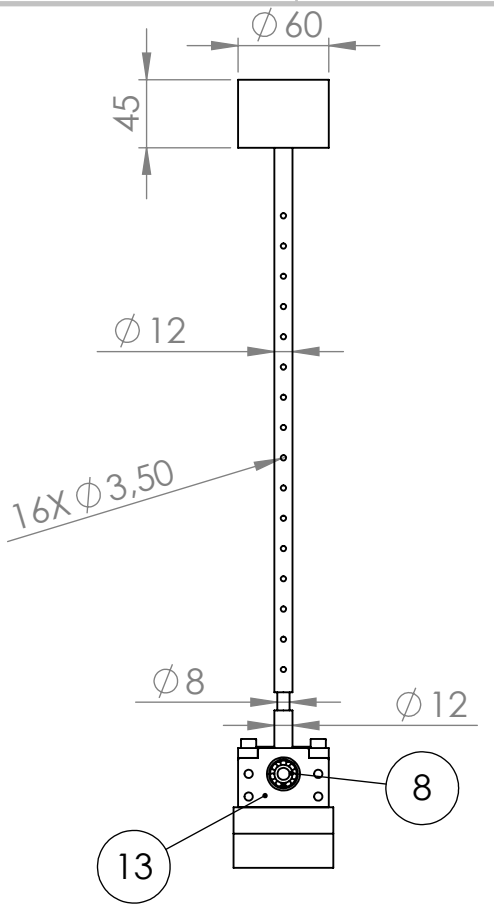


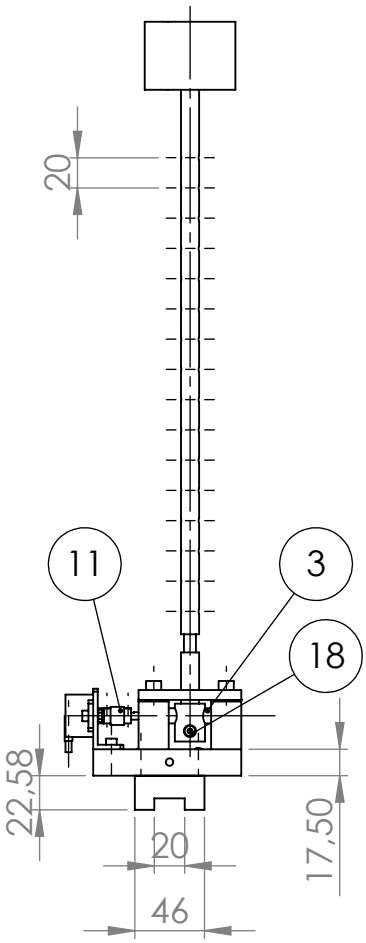


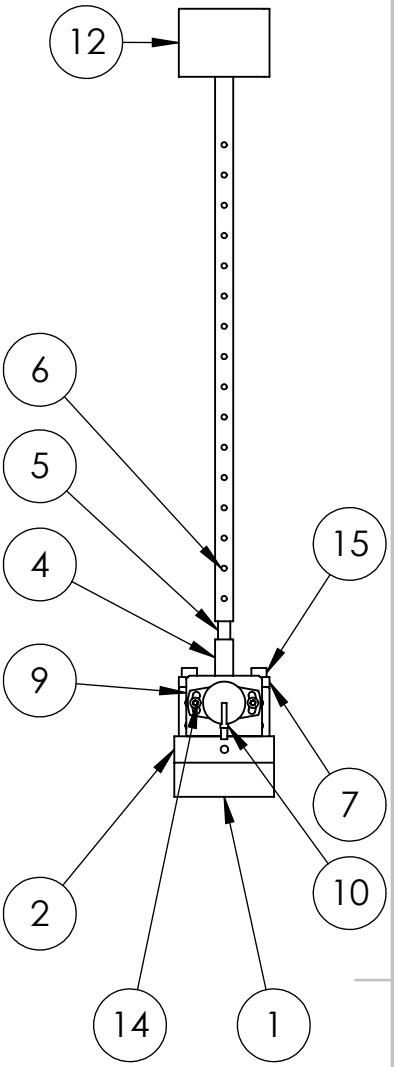


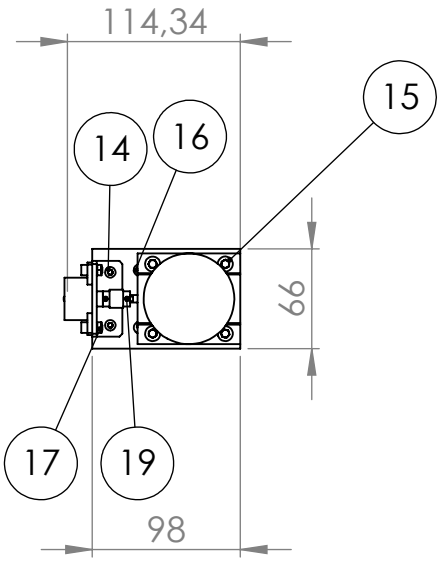


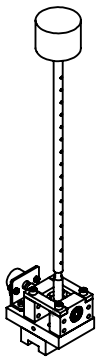
POLITECNICO DI TORINO	FINITURA: Ra 3.2		Oggetto: Banco Pendolo Inverso		ISO 2768 m-K
			Allievo: Alessandro Marino		Matricola: S231351
	NOME	FIRMA	DATA CREAZIONE:	Tav n. 1	Descrizione
DISEG.					Albero Pendolo
VERIF.			LAVORAZIONE		
APPR.					
FABB.					
Qual.			MATERIALE:	N. DISEGNO	A4
			Acciaio al carbonio semplice		
Modellazione Funzionale			PESO [kg]:	SCALA:1:1	FOGLIO 1 DI 1











Num. articolo	Num. parte	Quantità
1	Pattino	1
2	PiastraCollegamento	1
3	Albero2	1
4	Montante_Ver1	1
5	Pendolo_Sotto	1
6	Pendolo_Sopra	1
7	StopPendolo	2
8	cuscinetto	2
9	Staffa_sensore	1
10	SRH280DP	1
11	Giuntino	1
12	massa	1
13	Supporto_cuscinetto	2
14	ISO 4762 M4 x 10 - 10N	4
15	ISO 4762 M6 x 55 - 24N	4
16	ISO 7045 - M4 x 20 - Z - 20N	4
17	Hexagon Nut ISO 4032 - M4 - W - N	2
18	DIN 7984 - M4 x 14 --- 11.9N	1
19	DIN 913 - M3 x 3-N	2

POLITECNICO DI TORINO

FINITURA:

Oggetto: Banco Pendolo Inverso | ISO 2768 m-K

Allievo: Alessandro Marino | Matricola: S231351

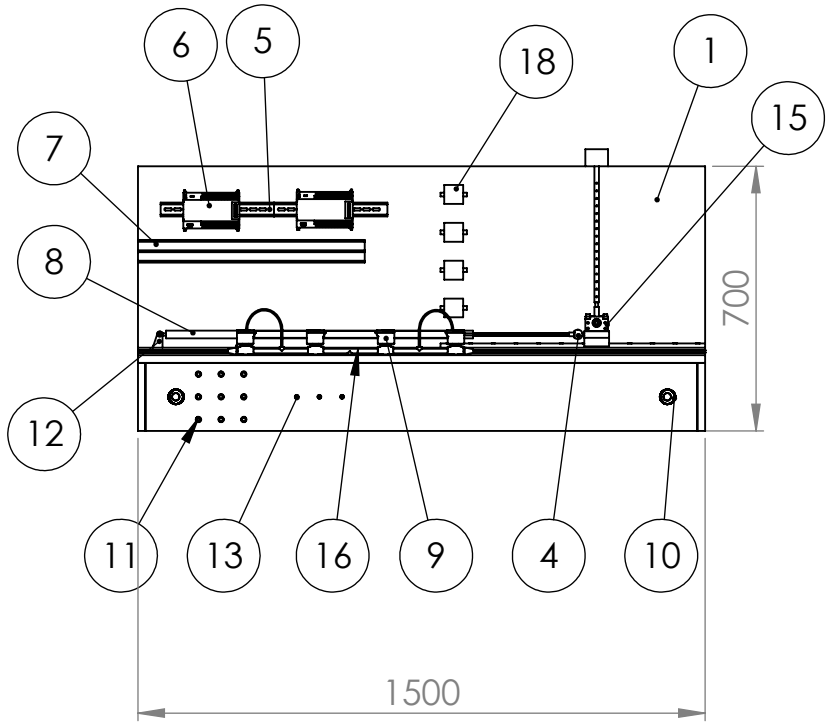
	NOME	FIRMA	DATA CREAZIONE:	Tav n. 2
DISEG.				
VERIF.			LAVORAZIONE	
APPR.				
FABB.				
Qual.			MATERIALE:	

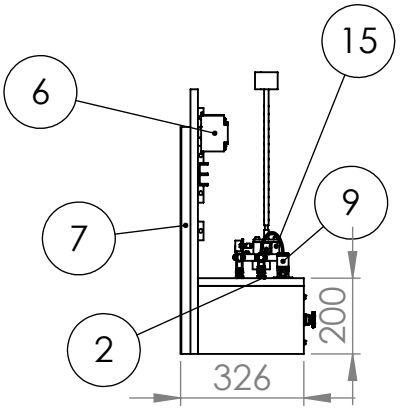
Descrizione

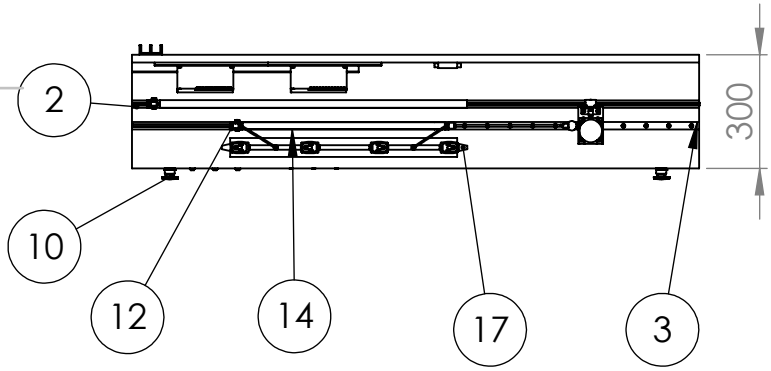
Assieme Carrello

N. DISEGNO | A4

Modellazione Funzionale | PESO [kg]: | SCALA:1:5 | FOGLIO 1 DI 1







Num. articolo	Num. parte	Quantità
1	Struttura principale	1
2	Profilato 20X20	2
3	Guida carrellino	1
4	Snodo_Cilindro_camozzi	1
5	Dinrail	2
6	PLC	2
7	Canalina	2
8	LVDT	1
9	Valvole_assieme	1
10	Stop Push Button	2
11	Push Button	9
12	Supporto_Camozzi_LVDT	2
13	10 turn potentiometer	3
14	Cilindro Pneumatico	1
15	AssiemeCarrello	1
16	Raccordo T	3
17	silenziatore	2
18	driver	4

POLITECNICO DI TORINO

FINITURA:

Oggetto: Banco Pendolo Inverso | ISO 2768 m-K

Allievo: Alessandro Marino | Matricola: S231351

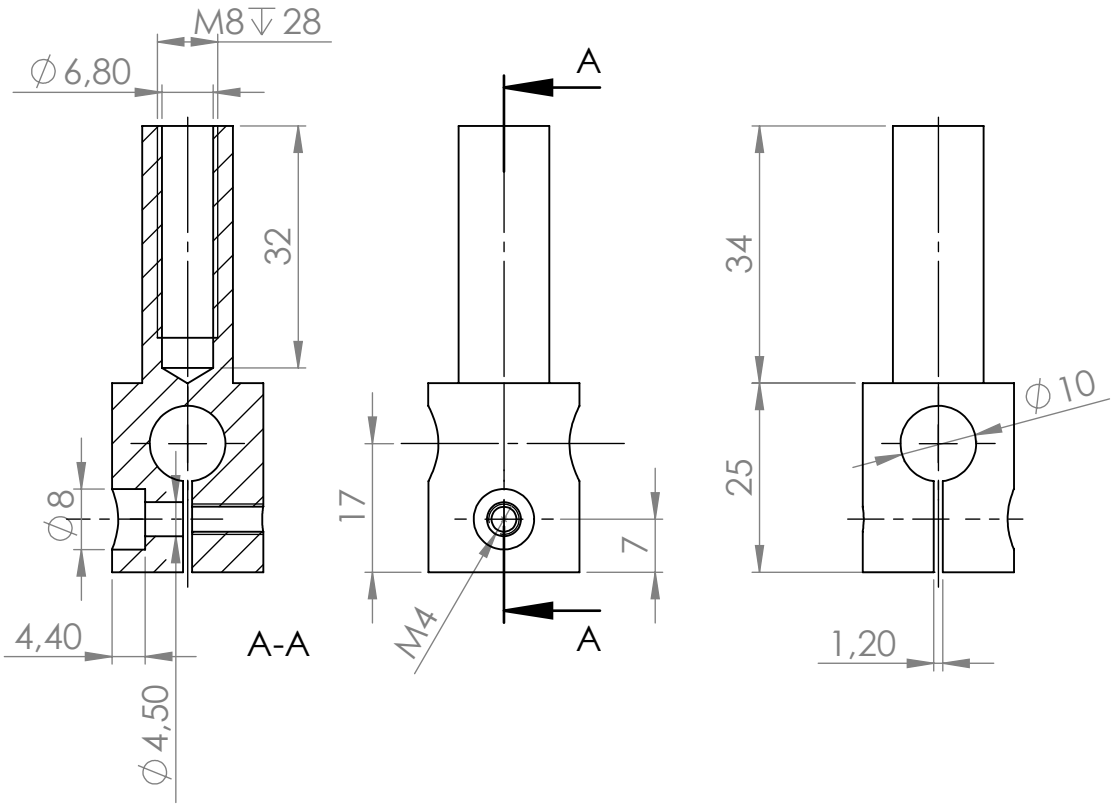
	NOME	FIRMA	DATA CREAZIONE:	Tav n.3
DISEG.				
VERIF.				LAVORAZIONE
APPR.				
FABB.				
Qual.			MATERIALE:	

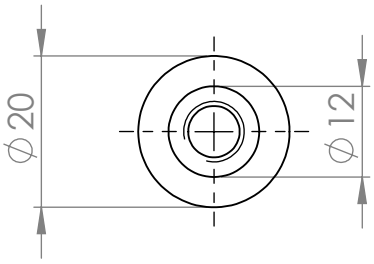
Descrizione

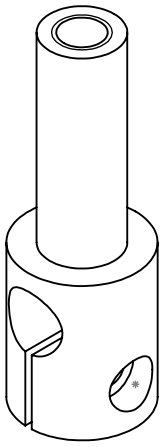
Assieme Banco

N. DISEGNO | A4

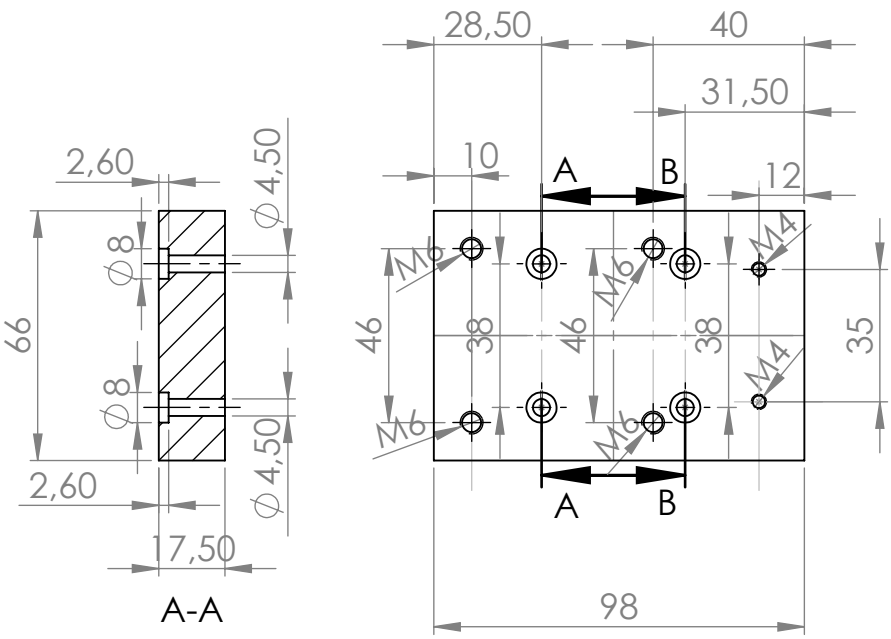
Modellazione Funzionale | PESO [kg]: | SCALA:1:20 | FOGLIO 1 DI 1

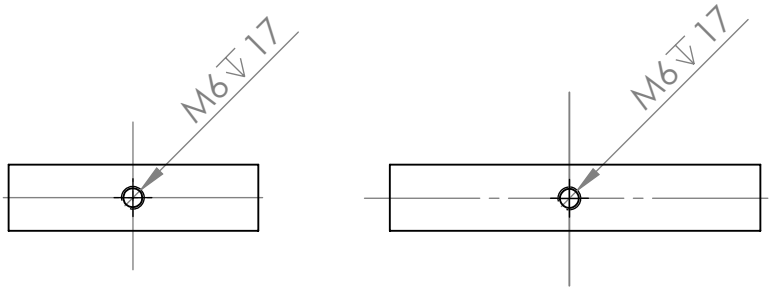


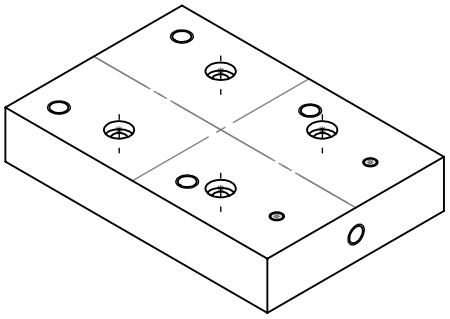




POLITECNICO DI TORINO	FINITURA: Ra 3.2		Oggetto: Banco Pendolo Inverso		ISO 2768 m-K
			Allievo: Alessandro Marino		Matricola: S231351
	NOME	FIRMA	DATA CREAZIONE:	Tav n.4	Descrizione
DISEG.					Montante Asta
VERIF.			LAVORAZIONE		
APPR.					
FABB.					
Qual.			MATERIALE: Alluminio		N. DISEGNO A4
Modellazione Funzionale			PESO [kg]:	SCALA:1:1	FOGLIO 1 DI 1

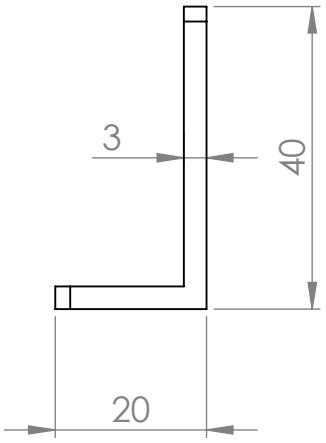


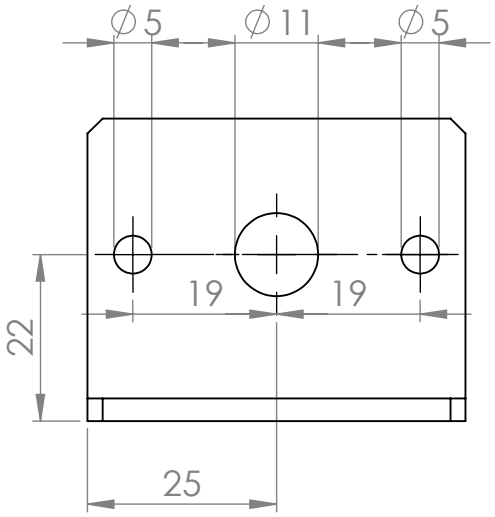


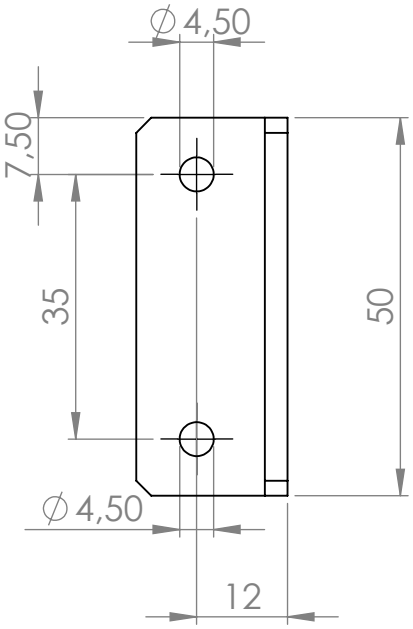


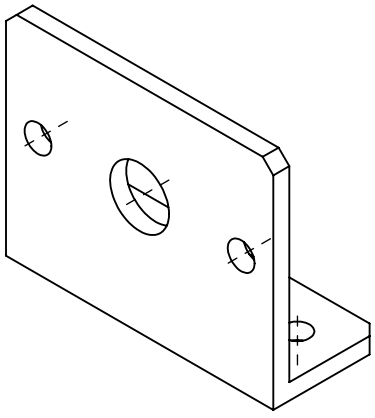
Fori filettati passanti

POLITECNICO DI TORINO	FINITURA: Ra 3.2		Oggetto: Banco pendolo inverso	ISO 2768 m-K	
			Allievo: Alessandro Marino	Matricola: S231351	
	NOME	FIRMA	DATA CREAZIONE:	Tav n. 5	Descrizione: Piastra collegamento
DISEG.					
VERIF.			LAVORAZIONE		
APPR.					
FABB.					
Qual.			MATERIALE: Alluminio	N. DISEGNO	A4
Modellazione Funzionale			PESO [kg]:	SCALA:1:2	FOGLIO 1 DI 1





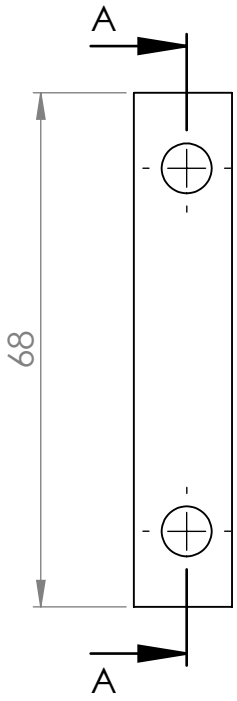


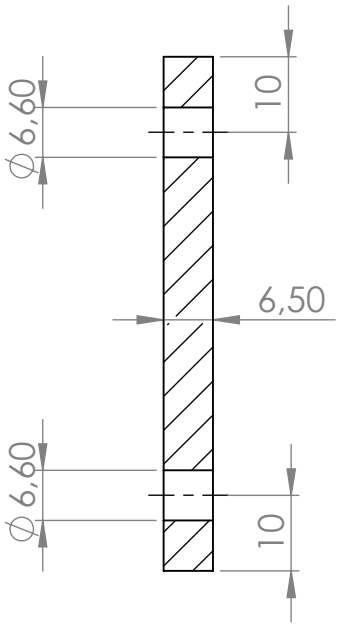


Smussi non quotati 2x45°

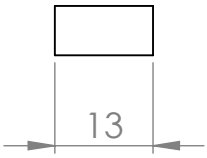
POLITECNICO DI TORINO	FINITURA: Ra 3.2		Oggetto: Banco Pendolo Inverso		ISO 2768 m-K
			Allievo: Alessandro Marino		Matricola: S231351

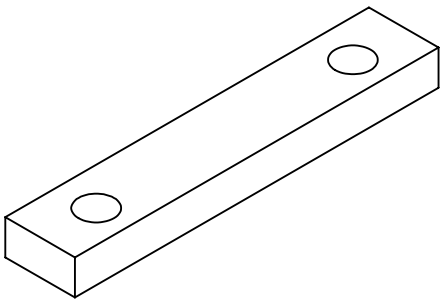
	NOME	FIRMA	DATA CREAZIONE:	Tav n. 6	Descrizione
DISEG.					Staffa Sensore
VERIF.				LAVORAZIONE	
APPR.					
FABB.					
Qual.			MATERIALE: Alluminio		N. DISEGNO A4
Modellazione Funzionale			PESO [kg]:		SCALA:1:1 FOGLIO 1 DI 1



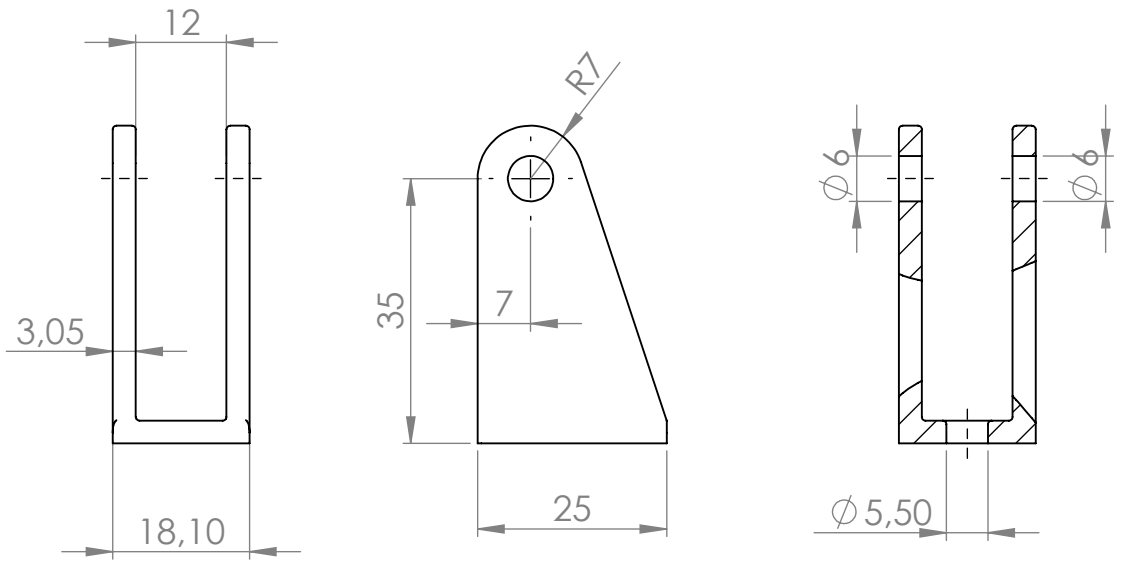


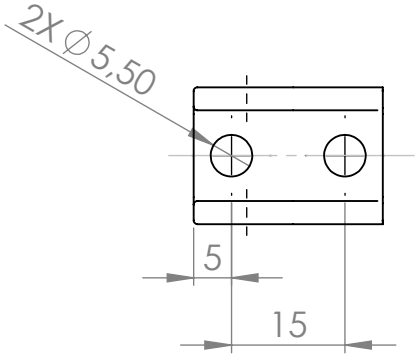
A-A

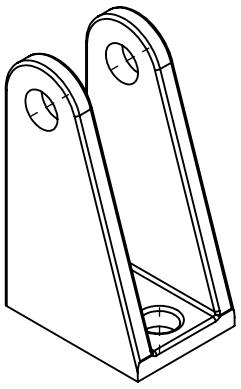




POLITECNICO DI TORINO	FINITURA: Ra 3.2		Oggetto: Banco Pendolo Inverso		ISO 2768 m-K
			Allievo: Alessandro Marino		Matricola: S231351
	NOME	FIRMA	DATA CREAZIONE:	Tav n. 7	Descrizione
DISEG.					Stop Pendolo
VERIF.				LAVORAZIONE	
APPR.					
FABB.					
Qual.			MATERIALE: Alluminio		N. DISEGNO A4
Modellazione Funzionale			PESO [kg]:		SCALA:1:1 FOGLIO 1 DI 1







Raccordi non quotati R0,5

POLITECNICO DI TORINO	FINITURA: Ra 3.2		Oggetto: Banco pendolo inverso	ISO 2768 m-K	
			Allievo: Alessandro Marino	Matricola: S231351	
	NOME	FIRMA	DATA CREAZIONE:	Tav n. 8	Descrizione
DISEG.					Supporto per LVDT e cilindro
VERIF.			LAVORAZIONE		
APPR.					
FABB.					
Qual.			MATERIALE: Alluminio	N. DISEGNO	A4
Modellazione Funzionale			PESO [kg]:	SCALA:1:1	FOGLIO 1 DI 1

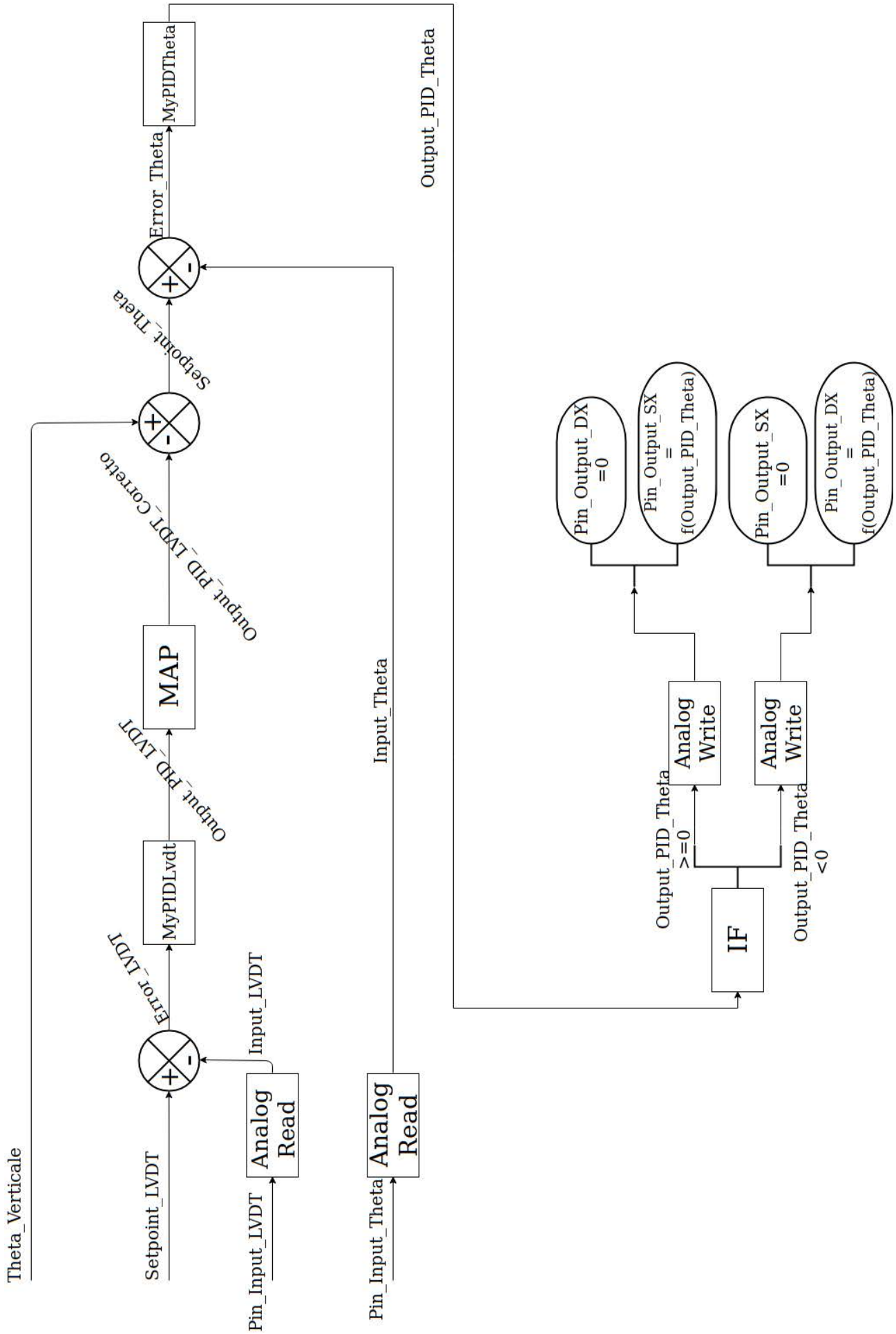
Programma Arduino

```
/*******************************************************
 PID pendolo
 The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and
similar boards, pins 5 and 6 have a frequency of approximately 980 Hz.

 //Per analogRead
 //default resolution (10 bits)
 analogReadResolution(10);

 // change the resolution to 16 bits
 analogReadResolution(16);

 //Per analogWrite
 // the default PWM resolution
 analogWriteResolution(8);

 // change the PWM resolution to 12 bit resolution is only supported on the DUE
 analogWriteResolution(12);

 The PID controller is designed to vary its output within a given range. By default
this range is 0-255: the arduino PWM range.
 There's no use sending 300, 400, or 500 to the PWM. Depending on the application
though, a different range may be desired.
 //Per impostare valori minimi e massimi
 myPid.SetOutputLimits(min, max)

 *******************************************************/

//inclusione librerie
#include <PID_v1.h>

//inclusione ingressi/uscite
#define PIN_INPUT_LVDT A0
#define PIN_INPUT_THETA A1
#define PIN_OUTPUT_SX 3 //uscite comandate da analogWrite, quindi sono già in pwm
#define PIN_OUTPUT_DX 2


double centro_carrello = 2.5; //sono i volt dopo il partitore dell'lvdt, quando il
carrello è in posizione centrale
//i 3 valori seguenti cambiano se si sposta il banco o si modifica qualcosa,
controllare sempre prima dell'avvio
double verticale = 2.566; //sono i volt quando l'asta è in verticale
double lettura_pot_sx = 1.43; //volt in uscita dal sensore quando il pendolo è a
sinistra
double lettura_pot_dx = 3.69; //volt in uscita dal sensore quando il pendolo è a
destra


//variabili theta
double Setpoint_theta, Input_theta, Output_pid_theta;
double Error_theta;
double Theta_verticale;



//variabili lvdt
double Setpoint_lvdt, Input_lvdt, Output_pid_lvdt, Output_pid_lvdt_corretto;
double Error_lvdt;

//setting pid theta
double Kp_theta = 8, Ki_theta = 25, Kd_theta = 0.5; //valori dei guadagni
double dead_band = 90; //a causa della banda morta, le valvole iniziano ad aprirsi
con un pwm non nullo, questo è il pwm su 255 valori @ apertura
```

```
//setting pid lvdt POSITON
double Kp_lvdt = 0.015, Ki_lvdt = 0.001, Kd_lvdt = 0.005;

PID myPIDtheta(&Input_theta, &Output_pid_theta, &Setpoint_theta, Kp_theta, Ki_theta,
Kd_theta, DIRECT);
PID myPIDlvdt(&Input_lvdt, &Output_pid_lvdt, &Setpoint_lvdt, Kp_lvdt, Ki_lvdt,
Kd_lvdt, DIRECT);


void setup() {
 // initialize serial communication at 9600 bits per second:
 Serial.begin(9600);

 pinMode(PIN_OUTPUT_SX, OUTPUT);
 pinMode(PIN_OUTPUT_DX, OUTPUT);

 //lvdt collegato tramite partitore, ingresso tra 0 e 5 volt al pin, su 1024 valori
 Setpoint_lvdt = centro_carrello / 5 * 1023;

 //contiene il valore della tensione in uscita dal potenziometro quando verticale
(verticale) e lo suddivide su 1024 valori, considerando lo span
 Theta_verticale = (verticale - lettura_pot_sx) * 1023 / (lettura_pot_dx -
lettura_pot_sx);

 //accensione dei pids
 myPIDtheta.SetMode(AUTOMATIC);
 myPIDtheta.SetOutputLimits(-255, 255); //il pid può avere uscita negativa
 myPIDtheta.SetSampleTime(2);
 myPIDlvdt.SetMode(AUTOMATIC);
 myPIDlvdt.SetOutputLimits(-255, 255); //il pid può avere uscita negativa
 myPIDlvdt.SetSampleTime(2);
}

void loop()
{
 Input_lvdt = analogRead(PIN_INPUT_LVDT);
 Error_lvdt=Setpoint_lvdt-Input_lvdt;
 myPIDlvdt.Compute();

 //l'uscita del pid è su 256 valori, per fare i calcoli bisogna spalmarlo su 1024
valori, siccome il pid ha anche uscita negativa, i campi minimi sono negativi
 //con errore x nullo, output pid x nullo, il setpoint theta deve essere (verticale-
lettura_pot_sx)*1023/(lettura_pot_dx-lettura_pot_sx)=511.5 su 1024 valori
 Output_pid_lvdt_corretto = map(Output_pid_lvdt, -255, 255, -1023, 1023);
 Setpoint_theta = Theta_verticale - Output_pid_lvdt_corretto;
 Input_theta = analogRead(PIN_INPUT_THETA);
 Error_theta=Setpoint_theta-Input_theta;
 myPIDtheta.Compute();

 if (Output_pid_theta >= 0) {
 analogWrite(PIN_OUTPUT_DX, 0);//annulla la direzione opposta
 //con output pid positivo, il carrello deve andare verso sinistra
 analogWrite(PIN_OUTPUT_SX, map(Output_pid_theta, 0, 255, dead_band, 255));
 }

 else {
 analogWrite(PIN_OUTPUT_SX, 0);//annulla la direzione opposta
 //con output pid negativo, il carrello deve andare verso destra,abs perchè output
pid negativo
 analogWrite(PIN_OUTPUT_DX, map(abs(Output_pid_theta), 0, 255, dead_band, 255));
 }
```

```
 //per leggere i valori in fase di setting, scollegare aria compressa e muovere il
carrello a mano, usare il delay
 //togliere il commento alle parti interessanti da graficare e aprire il plotter
seriale da Strumenti per visualizzare l'andamento

 //Serial.println(centro_carrello); //stampa la tensione di set del carrello
 //Serial.println(Setpoint_theta*((lettura_pot_dx-
lettura_pot_sx)/1023)+lettura_pot_sx); //stampa il theta set variabile in tensione

 //Serial.println(Input_lvdt/1023*5); //stampa la tensione dell'lvdt al pin di
arduino, 2.5 volt è al centro del carrello
 //Serial.println(Input_theta*((lettura_pot_dx-lettura_pot_sx)/1023)+lettura_pot_sx);
//stampa la tensione del potenziometro rotativo al pin di arduino

 //Serial.println(Error_lvdt); //stampa l'errore sulla x in 1024 valori
 //Serial.println(Error_theta); //stampa l'errore sulla rotazione in 1024 valori

 //Serial.println(Output_pid_lvdt);//stampa l'uscita su 255 valori del pid x
 //Serial.println(Output_pid_theta);//stampa l'uscita su 255 valori del pid theta

 //delay(500);
}
```

Schema a blocchi programma Arduino



0

serve per attivare il primo ingresso analogico Attiva_ingresso_1

Scale with Parameters, serve per scalare i valori che ven ono letti dai moduli.

SCP
Scale with Parameters, serve p...
SCP SCP0 ...
Input Local:3:I.Ch0Data
1
InputMin 0
InputMax 5250
ScaledMin 0
ScaledMax 10
Output IngressoAnalogico1
<P1>
10000.0

1

serve per attivare la prima uscita analogica Attiva_uscita_1

Scale with Parameters, serve per scalare i valori che ven ono letti dai moduli.

SCP
Scale with Parameters, serve ...
SCP SCP1 ...
Input UscitaAnalogica1
<V1>
1.075181
InputMin 0
InputMax 10
ScaledMin 12441
ScaledMax 31104
Output Local:5:O.Ch0Data
12441

2

serve per attivare il secondo ingresso analogico Attiva_ingresso_2

Scale with Parameters, serve per scalare i valori che ven ono letti dai moduli.

SCP
Scale with Parameters, serve p...
SCP SCP2 ...
Input Local:3:I.Ch1Data
1
InputMin 0
InputMax 5250
ScaledMin 0
ScaledMax 10
Output IngressoAnalogico2
<P2>
0.0

3

serve per attivare la seconda uscita analogica
Attiva_uscita_2

Scale with Parameters, serve per scalare i valori che ven ono letti dai moduli.

SCP
Scale with Parameters, serve ...
SCP SCP3 ...
Input UscitaAnalogica2 <V2> 0.0
InputMin 0
InputMax 10
ScaledMin 12441
ScaledMax 31104
Output Local:5:O.Ch1Data 17801

4

serve per attivare il terzo ingresso analogico
Attiva_ingresso_3

Scale with Parameters, serve per scalare i valori che ven ono letti dai moduli.

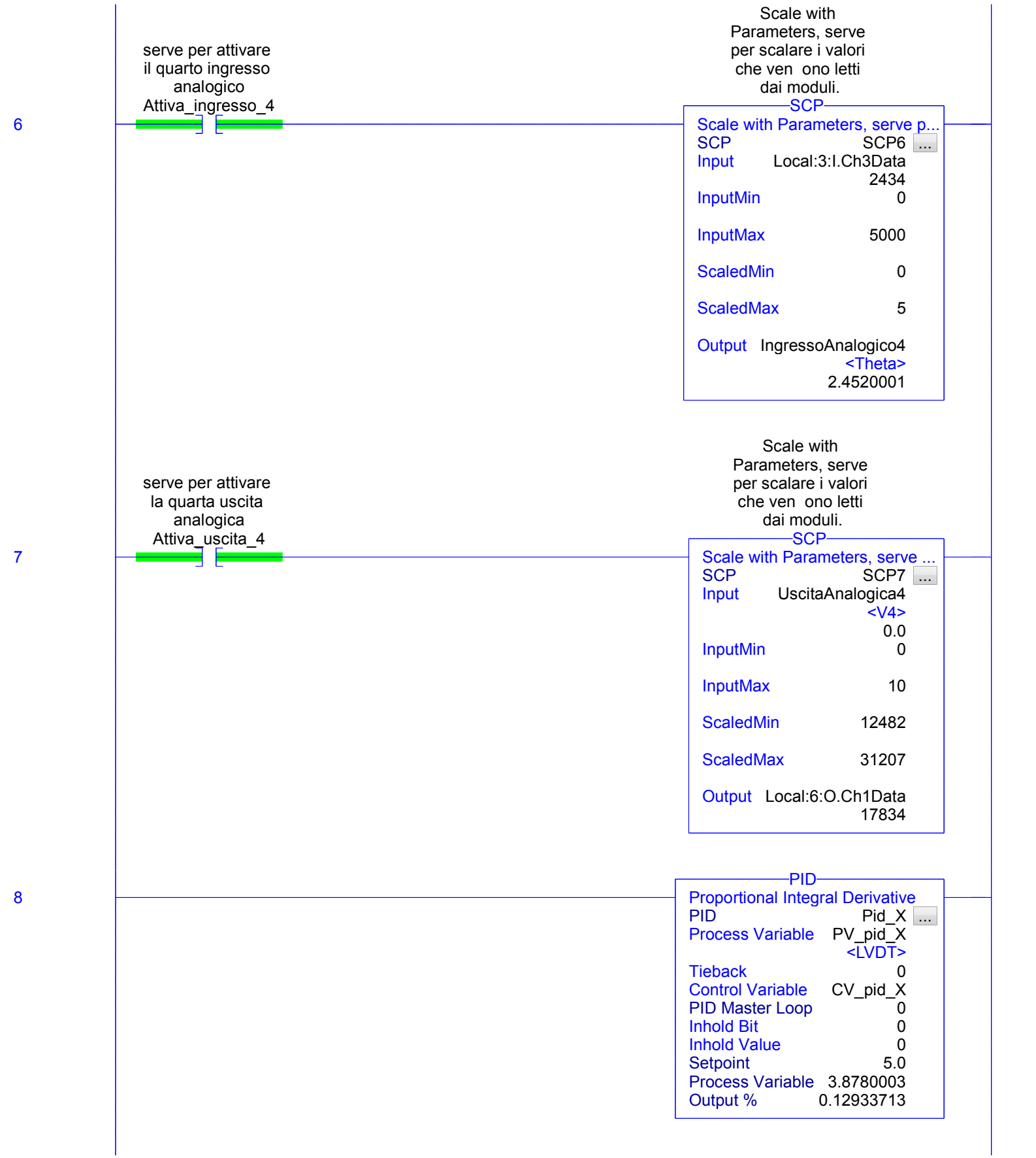
SCP
Scale with Parameters, serve p...
SCP SCP4 ...
Input Local:3:I.Ch2Data 3942
InputMin 0
InputMax 10000
ScaledMin 0
ScaledMax 10
Output IngressoAnalogico3 <LVDT> 3.8780003

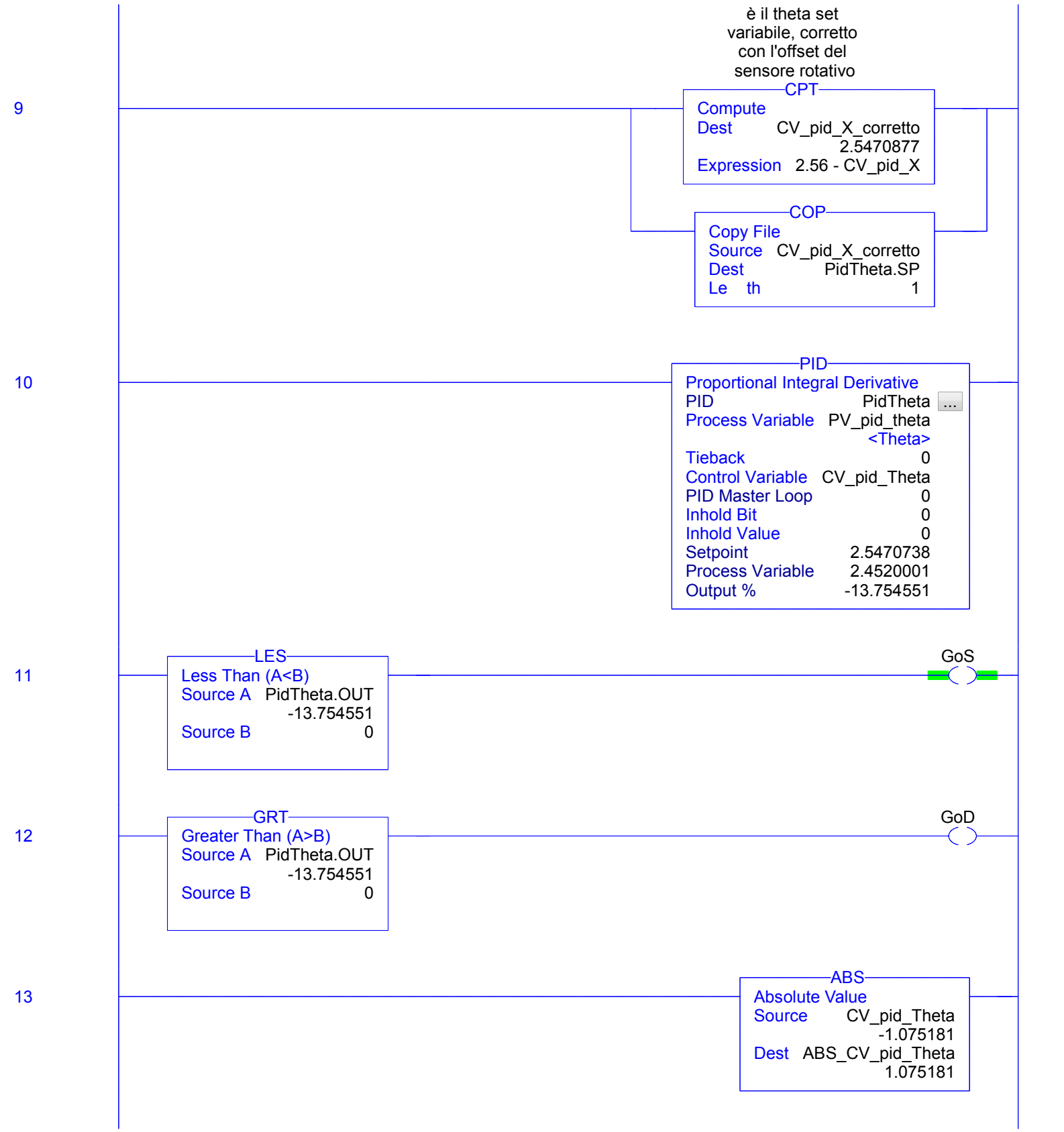
5

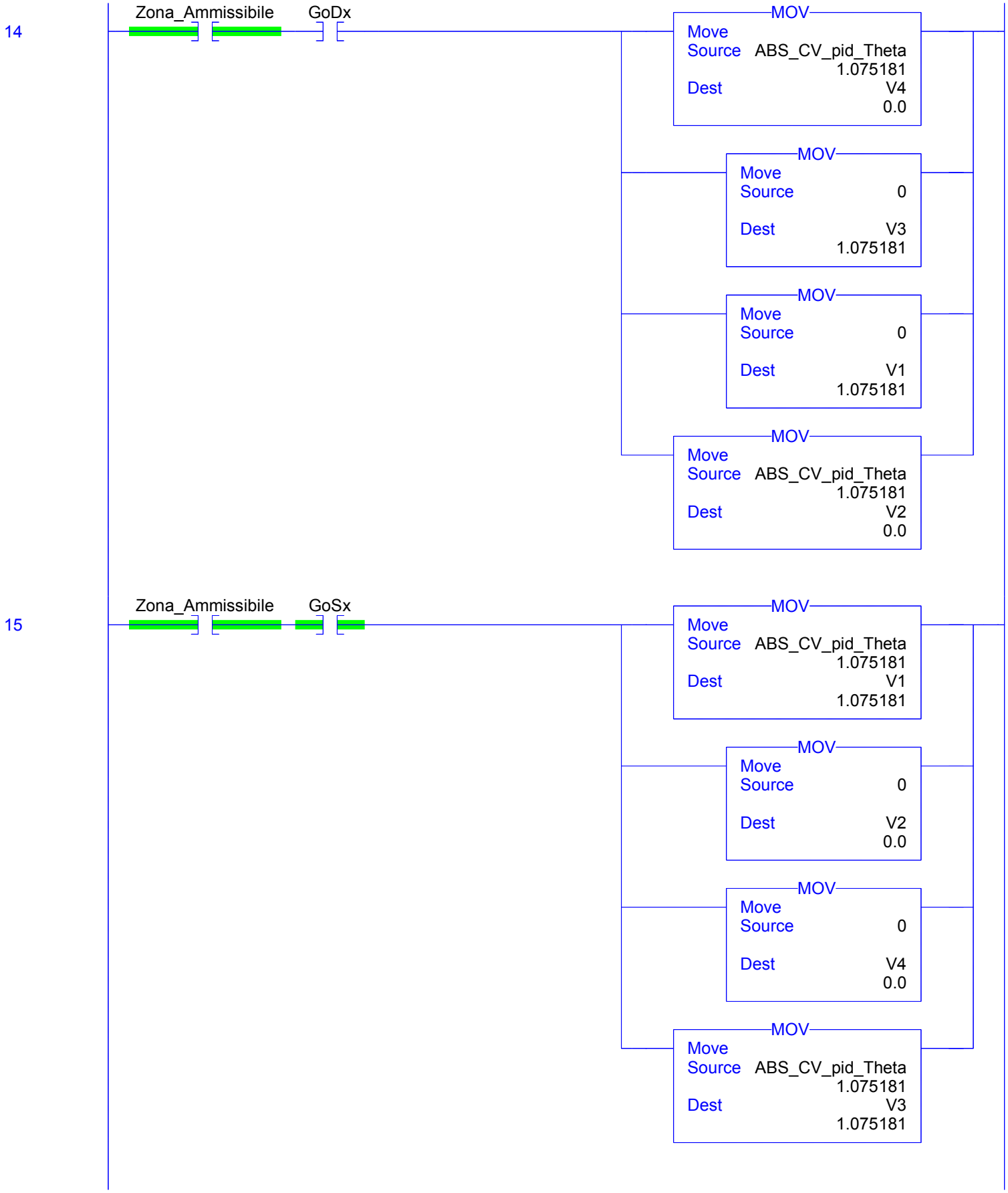
serve per attivare la terza uscita analogica
Attiva_uscita_3

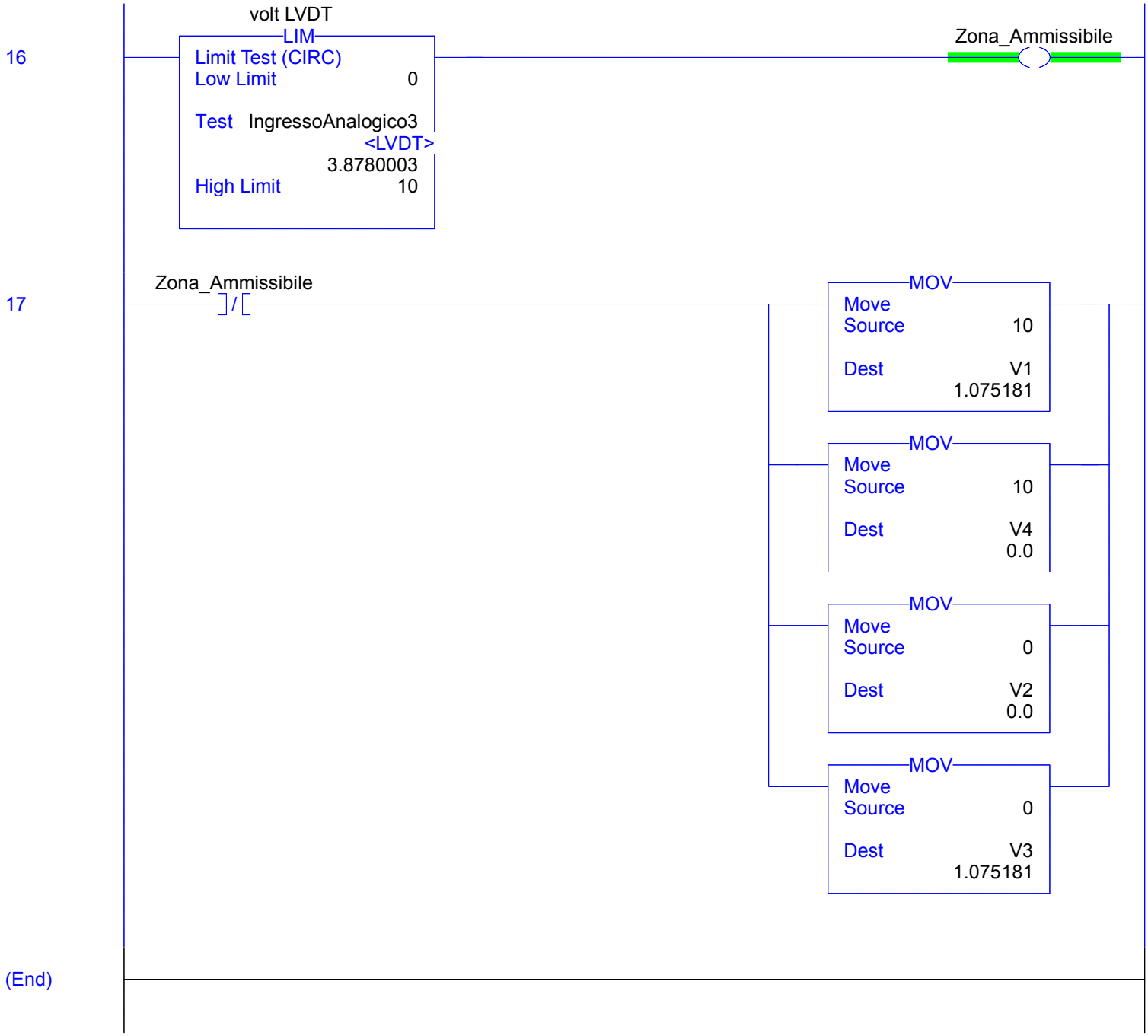
Scale with Parameters, serve per scalare i valori che ven ono letti dai moduli.

SCP
Scale with Parameters, serve ...
SCP SCP5 ...
Input UscitaAnalogica3 <V3> 1.075181
InputMin 0
InputMax 10
ScaledMin 12482
ScaledMax 31207
Output Local:6:O.Ch0Data 12482

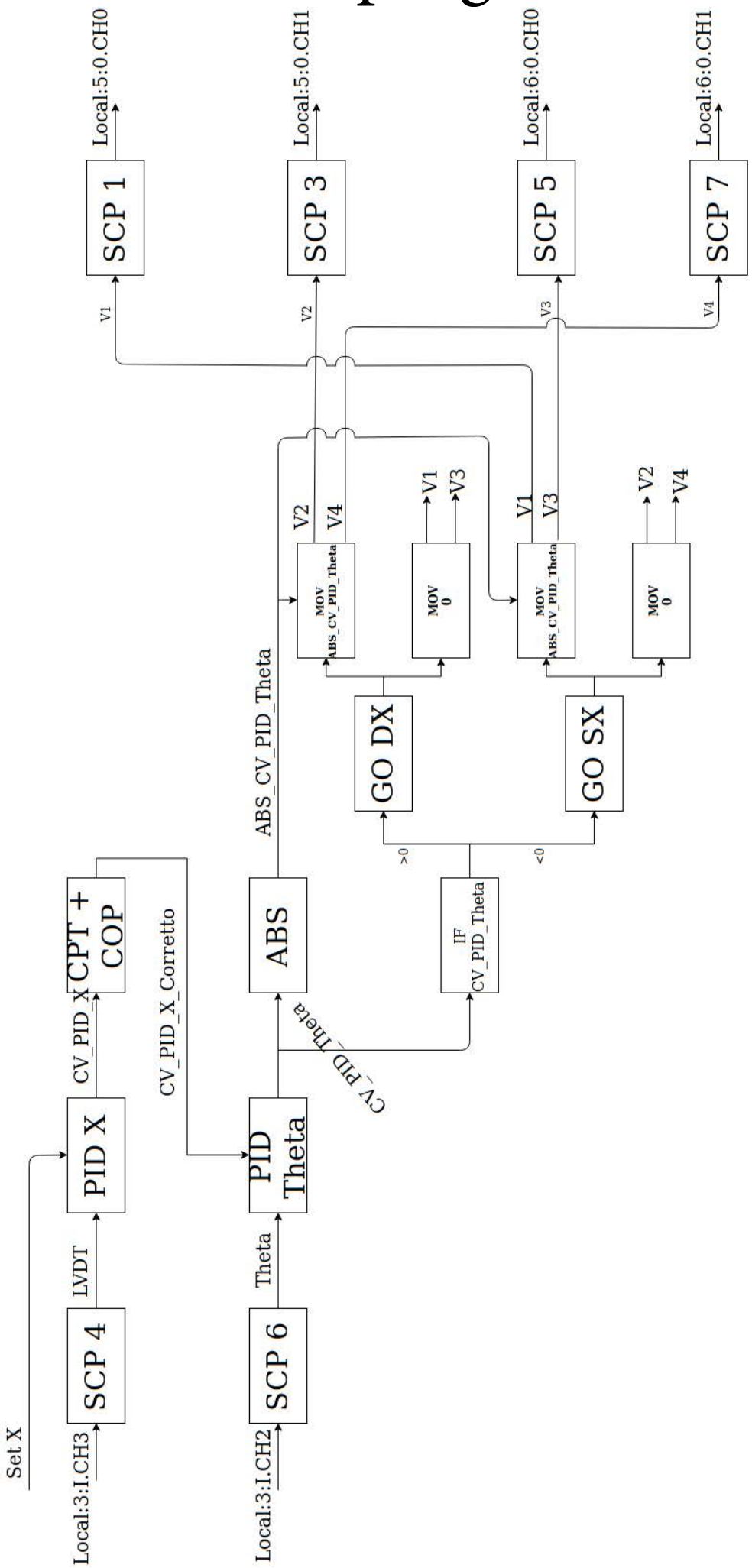

6
serve per attivare il quarto ingresso analogico Attiva_ingresso_4
Scale with Parameters, serve per scalare i valori che ven ono letti dai moduli.
SCP
Scale with Parameters, serve p...
SCP SCP6 ...
Input Local:3:I.Ch3Data 2434
InputMin 0
InputMax 5000
ScaledMin 0
ScaledMax 5
Output IngressoAnalogico4 <Theta> 2.4520001
7
serve per attivare la quarta uscita analogica Attiva_uscita_4
Scale with Parameters, serve per scalare i valori che ven ono letti dai moduli.
SCP
Scale with Parameters, serve ...
SCP SCP7 ...
Input UscitaAnalogica4 <V4> 0.0
InputMin 0
InputMax 10
ScaledMin 12482
ScaledMax 31207
Output Local:6:O.Ch1Data 17834
8
PID
Proportional Integral Derivative
PID Pid_X ...
Process Variable PV_pid_X <LVDT>
Tieback 0
Control Variable CV_pid_X
PID Master Loop 0
Inhold Bit 0
Inhold Value 0
Setpoint 5.0
Process Variable 3.8780003
Output % 0.12933713


è il theta set
variabile, corretto
con l'offset del
sensore rotativo
9
CPT
Compute
Dest CV_pid_X_corretto
2.5470877
Expression 2.56 - CV_pid_X
COP
Copy File
Source CV_pid_X_corretto
Dest PidTheta.SP
Le th 1
10
PID
Proportional Integral Derivative
PID PidTheta
Process Variable PV_pid_theta
<Theta>
Tieback 0
Control Variable CV_pid_Theta
PID Master Loop 0
Inhold Bit 0
Inhold Value 0
Setpoint 2.5470738
Process Variable 2.4520001
Output % -13.754551
11
LES
Less Than (A<B)
Source A PidTheta.OUT
-13.754551
Source B 0
GoS
12
GRT
Greater Than (A>B)
Source A PidTheta.OUT
-13.754551
Source B 0
GoD
13
ABS
Absolute Value
Source CV_pid_Theta
-1.075181
Dest ABS_CV_pid_Theta
1.075181


14
Zona_Ammissibile
GoDx
MOV
Move
Source ABS_CV_pid_Theta
1.075181
Dest V4
0.0
MOV
Move
Source 0
Dest V3
1.075181
MOV
Move
Source 0
Dest V1
1.075181
MOV
Move
Source ABS_CV_pid_Theta
1.075181
Dest V2
0.0
15
Zona_Ammissibile
GoSx
MOV
Move
Source ABS_CV_pid_Theta
1.075181
Dest V1
1.075181
MOV
Move
Source 0
Dest V2
0.0
MOV
Move
Source 0
Dest V4
0.0
MOV
Move
Source ABS_CV_pid_Theta
1.075181
Dest V3
1.075181


16
volt LVDT
LIM
Limit Test (CIRC)
Low Limit 0
Test IngressoAnalogico3 <LVDT> 3.8780003
High Limit 10
Zona_Ammissibile
17
Zona_Ammissibile
MOV
Move
Source 10
Dest V1 1.075181
MOV
Move
Source 10
Dest V4 0.0
MOV
Move
Source 0
Dest V2 0.0
MOV
Move
Source 0
Dest V3 1.075181
(End)

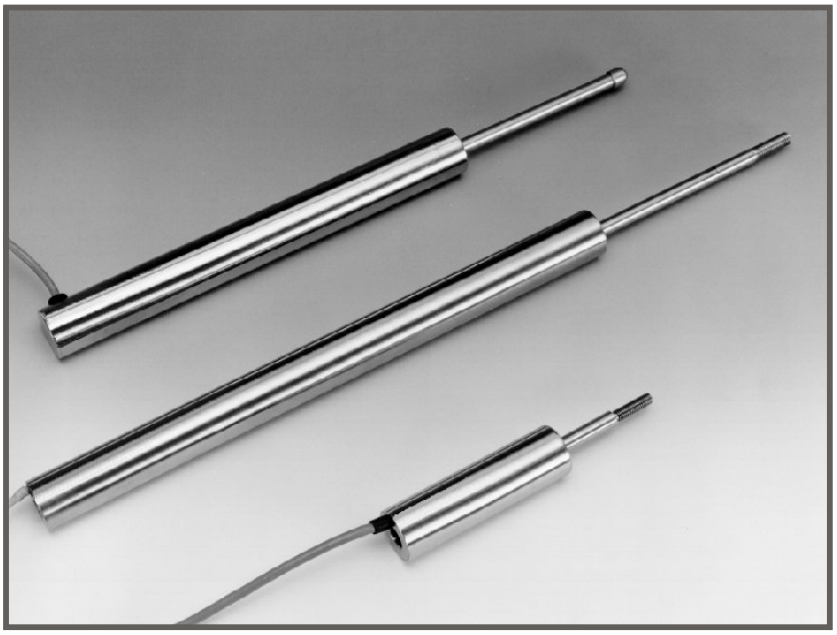
Schema a blocchi programma PLC



Standard Series LVDT

DISPLACEMENT TRANSDUCER

AML/E Series



- **Ranges ±0.5mm to ±500mm**
- **Stainless Steel Construction**
- **Simple Installation**
- **Versatile Packaging, giving many Standard Mounting Options**
- **Wide Variety of Different Outputs; mVac, 0-5Vdc, 0-10Vdc, 4-20mA, ±2.5Vdc**
- **Ideally Suited for OEM Applications.**
- **3 YEAR WARRANTY**

Options Available

Longer cable lengths available on request.

±0.25% Accuracy (excludes Spring Loaded units)

Higher temperature versions (consult factory).

Custom design versions available (consult factory)

DESCRIPTION

The AML/E series of Standard LVDT Displacement Transducers can be AC or DC powered and are widely used in OEM and general purpose applications such as material testing machines, automotive/aerospace test rigs and actuators, etc.

They are supplied in a variety of packaging formats, enabling engineers to select quickly and precisely, the product required for a particular application.

The AML/E is supported with a versatile range of instrumentation to enable engineers to implement the sensor with the minimum of fuss within a system. Supporting instrumentation includes trip amplifiers, indicators, PC interfaces, rack systems, etc.

Transducer Specialists...

APPLIED MEASUREMENTS LIMITED

3 MERCURY HOUSE - CALLEVA PARK - ALDERMASTON - BERKSHIRE - RG7 8PN - UK

Tel: (+44) 0118 981 7339 Fax: (+44) 0118 981 9121 email: info@appmeas.co.uk Internet: www.appmeas.co.uk

SPECIFICATION

CHARACTERISTICS	AML/E---	AML/EJ---	AML/EU---	AML/EU--- -10	AML/EI---	AML/ED---	UNITS
Stroke Measurement Range:	±0.5, ±2.5, ±5, ±10, ±12.5, ±15, ±25, ±50, ±75, ±100, ±125, ±150, ±175, ±200, ±250 ±300, ±400, ±500 (maximum stroke is ±100 for Sprung Loaded Core & Extension - Option S)						**millimetres**
Signal Output:	See Table Below		0-5volt	0-10volt	4-20mA	±2.5volt	
No. of Wires	6	4	3	3	3	4	
Supply Voltage (unregulated):	2 to 5Vrms @ 1 to 5kHz		10-24Vdc	14-24Vdc	14-24Vdc	12Vdc regulated	
Supply Current:	-		35mA @ 15V	35mA @ 15V	-	35mA @ 12V	
Max. Loop Resistance:	-		-	-	300 @ 30V	-	**ohms**
Max. Output Sink Current:	-		0.5	1	-	0.1	**milliamps**
Non-Linearity:	<0.50						**±% Stroke Range**
Repeatability:	<0.10						**±% Stroke Range**
Output Bandwidth:	100		100	100	100	100	**Hz**
Output Ripple:	-		30mV max.	30mV max.	0.1% @ 20mA	30mV max.	
Operating Temperature Range:	AML/E & EJ: -30 to +85 Std. / -30 to +150 Opt.		0 to +70 on DC/DC models				**°C**
Zero Temperature Coefficient:	<0.020		<0.010				**±%Stroke Range/°C**
Span Temperature Coefficient:	<0.020		<0.030				**±%Stroke Range/°C**
Vibration Resistance:	20g up to 2kHz						
Shock Resistance:	1000g for 10milliseconds						
Construction Materials:	Body & Extension Rod: 303 St/Steel, Core: 416 St/Steel, Cable Gland: Nickel-Plated Brass,						
	Spring: 316 St/Steel, Rod-End Bearings: Mild Steel						
Connections:	2 metre screened PVC*, axial exit. For radial exit request option C. (*High-Temp Version = PTFE)						
Environmental Sealing:	IP54						

Note: On DC output version (0Vdc / 4mA) is given with the core in the extended / outwards position. This can be reversed if required, please request **Option Y** on your order.

Dimensions for AC Units with Radial Cable exit (AML/E & AML/EJ) only

Note - add 10mm to length of sensor if axial cable option is selected

	Core Extension STANDARD		Core Extension OPTIONS X & G	Core Extension OPTION S	Core Extension OPTION R	AML/E & EJ Output Sensitivity @3kHz (mV/V)
STROKE (mm)	**BL**	**CL**	**PL**	**SL**	**EL**	
±0.5	25	15	25	50	111	50
±2.5	32	15	32	57	118	90
±5	73	29	73	98	159	80
±10	77	35	77	98	159	280
±12.5	92	35	92	117	178	300
±15	120	50	120	145	206	230
±25	160	76	160	185	246	240
±50	246	115	246	271	332	320
±75	320	138	320	345	406	350
±100	377	140	377	345	463	190
±125	435	152	435	n/a	521	300
±150	512	165	512	n/a	598	330
±175	563	180	563	n/a	649	310
±200	628	185	628	n/a	714	300
±250	750	170	750	n/a	836	350
±300	850	185	850	n/a	936	400
±400	1100	250	1100	n/a	1186	460
±500	1350	314	1350	n/a	1436	390

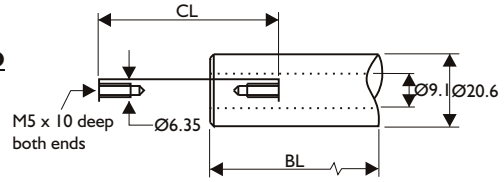
Dimensions for DC units only

(Models: AML/EU, AML/EU-10, AML/EI & AML/ED)

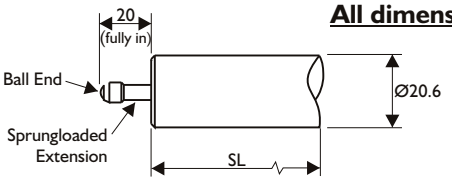
	Core Extension STANDARD		Core Extension OPTIONS X & G	Core Extension OPTION S	Core Extension OPTION R
STROKE (mm)	**BL**	**CL**	**PL**	**SL**	**EL**
±0.5	75	15	75	100	151
±2.5	82	15	82	107	158
±5	123	29	123	148	199
±10	123	35	123	148	199
±12.5	142	35	142	167	218
±15	170	50	170	195	246
±25	210	76	210	235	286
±50	296	115	296	321	372
±75	370	138	370	395	446
±100	427	140	427	395	503
±125	485	152	485	n/a	561
±150	562	165	562	n/a	638
±175	613	180	613	n/a	689
±200	678	185	678	n/a	754
±250	800	170	800	n/a	876
±300	900	185	900	n/a	976
±400	1150	250	1150	n/a	1226
±500	1400	314	1400	n/a	1476

All dimensions in mm

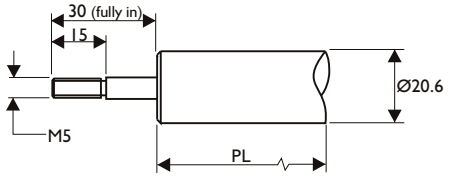
STANDARD
Core only



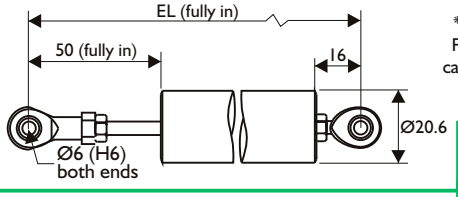
OPTION S
Sprung Loaded Core & Extension



OPTION X
Plain Core & Extension
OPTION G
Guided Core & Extension



OPTION R
Guided Core* & Extension with Rod End Bearings



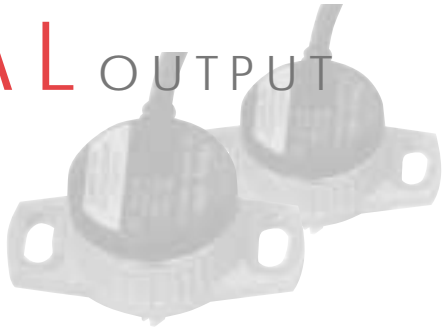
*Core is not captive. Please consult sales if captive core is required

APPLIED MEASUREMENTS LIMITED

Issue 02/12

Continuous product development may result in minor changes to published specifications.

SRH280DP DUAL OUTPUT
contactless rotary sensor



PERFORMANCE

ELECTRICAL

Measurement range	°	20 to 360 in 1° increments
Supply voltage	Vdc	9 to 30 (unregulated) and 5 ±0.5 (regulated)
Over voltage protection	Vdc	Up to 40 (-40 to +60°C)
Maximum supply current	mA	<25
Reverse polarity protection		Yes
Short circuit protection		
Output to GND		Yes
Output to supply		In 5V regulated mode only
Power-on settlement time	S	<1
Resolution	%	0.025 of measurement range (12 bit)
Non-linearity*	%	<±0.4
Temperature coefficient	ppm/°C	<±30 in 5V supply mode; <±90 in 9-30V supply mode

* Non-linearity is measured using the least-squares method on a computerised calibration system

Analog Output (order code A1, A4) – see graph on page 31

Voltage output range		
9-30V supply	Vdc	Absolute voltage, 0.5 to 4.5 (A1) or 0.1 to 4.9 (A4) over measurement range (±3%)
5V supply	Vdc	Ratiometric output voltage - 10 to 90% (A1) or 2 to 98% (A4) of Vs over measurement range (±1%)
Monotonic range	Vdc	0.25 (5%) and 4.75 (95%) nominal (A1)
	Vdc	0.05 (1%) and 4.95 (99%) nominal (A4)
Load resistance	Ω	10k minimum (resistive to GND)
Output noise	mVrms	<1
Input/output delay	mS	<2

PWM Output (order code Pn) – see output characteristics on page 31

PWM frequency	Hz	244 (P1); 500 (P2); or 1000 (P3) ±20% over temperature range
PWM levels 9-30V supply	Vdc	0 and 5 nominal (±3%)
5V supply	Vdc	0 and Vs (±1%)
Duty cycle	%	10 to 90 over measurement range
Monotonic range	%	5 and 95 nominal
Load resistance	Ω	10k minimum (resistive to GND)
Rise/fall time	µS	<15

MECHANICAL

Mechanical angle	°	360, continuous
Operating torque - maximum		
sealed shaft IP68	g-cm	120
unsealed shaft IP50	g-cm	100
Shaft velocity maximum	°/sec	3600
Weight	g	<35
Mounting		Use 2 x M4 socket head cap screws and M4 washer - maximum tightening torque 2Nm
Phasing		When shaft flat (or shaft ident mark) is facing toward the cable exit, output is at mid travel. The sensor housing allows for ±10° adjustment via the mounting flange slots.

ENVIRONMENTAL

Protection class		IP68 (to 2m depth for 1 hour) or IP50
Life		20 million operations (10 x 10^6 cycles) of ±75° Sensing element life is essentially infinite (contactless); the SRH280DP life figure refers to the operating shaft seal. Mechanical load (axial and radial) on the shaft should also be considered.
Dither life		Contactless - no degradation due to shaft dither
Operational temperature†	°C	-40 to +140 (5V supply) -40 to +135.7 (9V supply) Derate upper temperature limit by 1.7°C for every 1V increase in supply: e.g. -40 to +100 @30V
Storage temperature	°C	-55 to +140
Vibration		BS EN 60068-2-64:1995 Sec 8.4 (31.4gn rms) 20 to 2000Hz Random
Shock		3m drop onto concrete
EMC Immunity level		BS EN 61000-4-3:1999, to 100V/m, 80MHz to 1GHz and 1.4GHz to 2.7GHz (2004/108/EC)

† See Maximum Operating Temperature – derating graph on page 30.
If the maximum operating temperature is exceeded, the voltage regulator will shut down to protect the device from overheating

OPTIONS

Measurement range (angle)		Select from 20° to 360° in 1° increments (factory programmed) for each output channel
Output		Analog voltage (An) or PWM (Pn)
Output direction		Both clockwise, both anticlockwise or one CW, one ACW
Shaft style		D section, sprung shaft (S) or 2.4mm blade shaft (H)
Shaft sealing		IP50 or IP68
Cable length	m	0.2 or 0.5
Custom housing		Synchro mount style with ball race bearings - ask our technical sales team for details
OEM options		Outputs can be programmed to provide: non linear laws; switch outputs; clamp voltages; different output phasing CH1/CH2; faster input/output delay; extended analog range; and output mapping for potentiometer replacements

AVAILABILITY

All standard configurations can be supplied rapidly from the factory - check with your local supplier for more details

ORDERING CODES

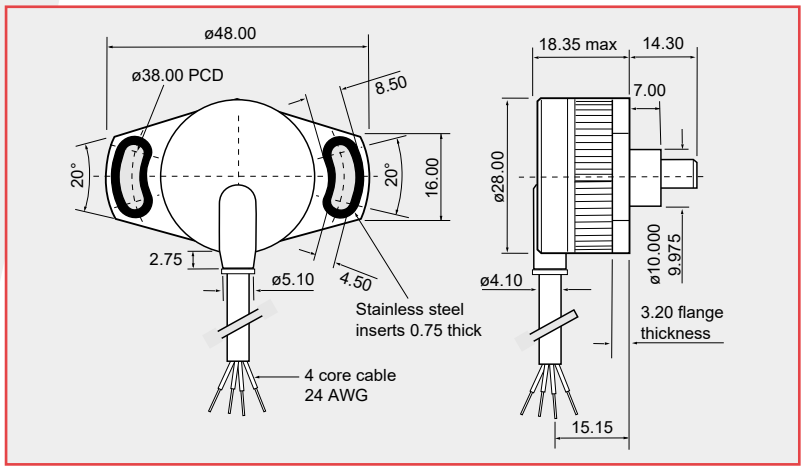
SRH280DP/...../...../...../...../...../...../.....

Measurement range	CH1 = angle in °
Measurement range	CH2 = angle in °
Output	A1 = Analog 0.5-4.5Vdc A4 = Analog 0.1-4.9Vdc P1 = PWM, 244 Hz P2 = PWM, 500 Hz P3 = PWM, 1000 Hz
Direction	3 = Both clockwise 4 = Both anticlockwise 5 = CH1 CW; CH2 ACW
Shaft style	D = D shaft S = Sprung shaft H = 2.4mm blade shaft
Shaft sealing	50 = IP50 68 = IP68
Cable length	P2 = 0.2m P5 = 0.5m

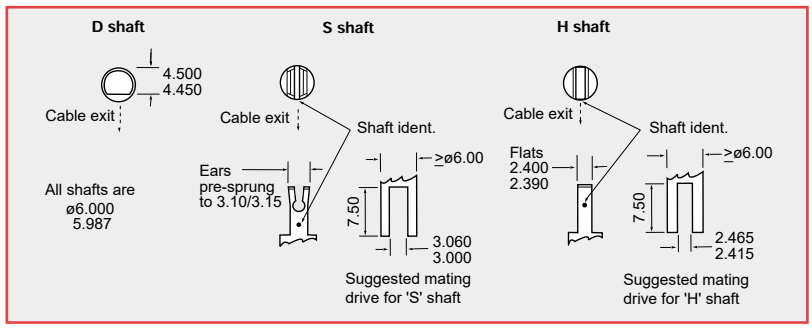
SRH280DP

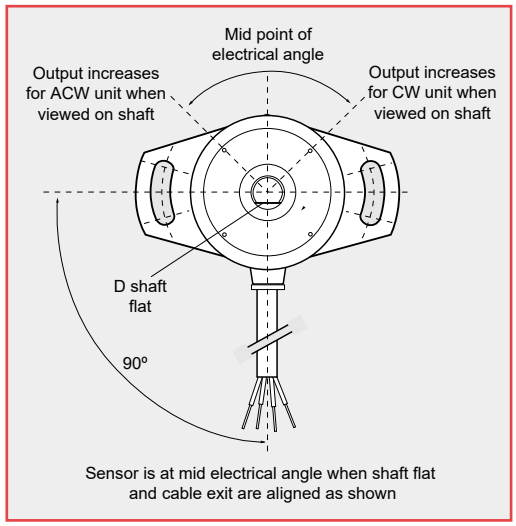
DIMENSIONS

Note: drawings not to scale



SHAFT OPTIONS





ELECTRICAL CONNECTIONS

200 or 500mm of 4-core cable: FDR-25 sheathed, with 55A spec (24AWG) cores

Cable colour	Description
Red	+V Supply
Yellow	Output 1
White	Output 2
Black	0V Supply (GND)

Output increases with CW or ACW rotation viewed on shaft - depending on selected order code.

When connecting the sensor, care should be taken with the correct connections. The sensor is provided with reverse polarity protection and short circuit protection between outputs (Yellow & White) to GND (Black), **but if the outputs (Yellow & White) are connected to the supply this will result in device failure.**