Università Politecnica delle Marche

Relazione Automazione Industriale

Ticchiarelli Davide, Ciotti Niccolò, Angelini Riccardo, Renzi Luca

A.A 2022/2023

Indice

1 Introduzione

1.1 Hardware

Il processo industriale da controllare a livello hardware è composto da più parti:

- **Pistone**: immette i pezzi sul nastro trasportatore
- **Nastro trasportatore**: trasporta i pezzi nella giostra
- **Giostra**: divisa in otto postazioni ognuna con un attuatore o un sensore:
  - ***Postazione 0***: postazione **buffer** per l'ingresso dei pezzi nella giostra
  - ***Postazione 1***: postazione con un **sensore di colore** utilizzato per riconoscere il colore dei pezzi
  - ***Postazione 2***: postazione con un **trapano** che trapana i pezzi
  - ***Postazione 3***: postazione con un **tastatore** che misura l'altezza dei pezzi
  - ***Postazione 4***: postazione con un **pennarello** che colora i pezzi
  - ***Postazione 5***: postazione con un **pistone** che scarta i pezzi in un **buffer illimitato**
  - ***Postazione 6***: postazione con un **pistone** che scarta i pezzi in un **buffer/baia limitato ad un pezzo**
  - ***Postazione 7***: postazione **buffer** dove non viene effettuata nessuna operazione

1.2 Software

1.2.1 TwinCat

Per il progetto è stato utilizzato il software TwinCAT 2 della casa produttrice BECKHOFF.
Tale software permette di interfacciare e configurare il PLC con il Computer attraverso il SYSTEM MANAGER e di programmare, simulare e "debuggare" applicazioni e programmi sviluppati per controllare processi industriali attraverso il PLC CONTROL

1.2.2 Linguaggi di programmazione utilizzati

Per la programmazione sono stati utilizzati più tipi di linguaggi.
Il principale è stato l'**SFC** (*Sequential Function Chart*) ovvero un tipo di linguaggio grafico per i PLC che si basa sull'utilizzo di stati e transizioni. I vantaggi principali sono:

- espressività e leggibilità: è un linguaggio grafico quindi sintetico e non ambiguo
- portabilità: progettazione e implementazione sono slegate tra loro e possono essere realizzate da persone diverse, in tempi diversi, con linguaggi diversi su PLC di produttori diversi
- manutenibilità: è possibile modificare i passi singolarmente senza alterare il resto dell'automatismo

Oltre all'SFC all'interno di alcuni stati e transizioni sono stati utilizzati come linguaggi il **Ladder** e l' **ST**.

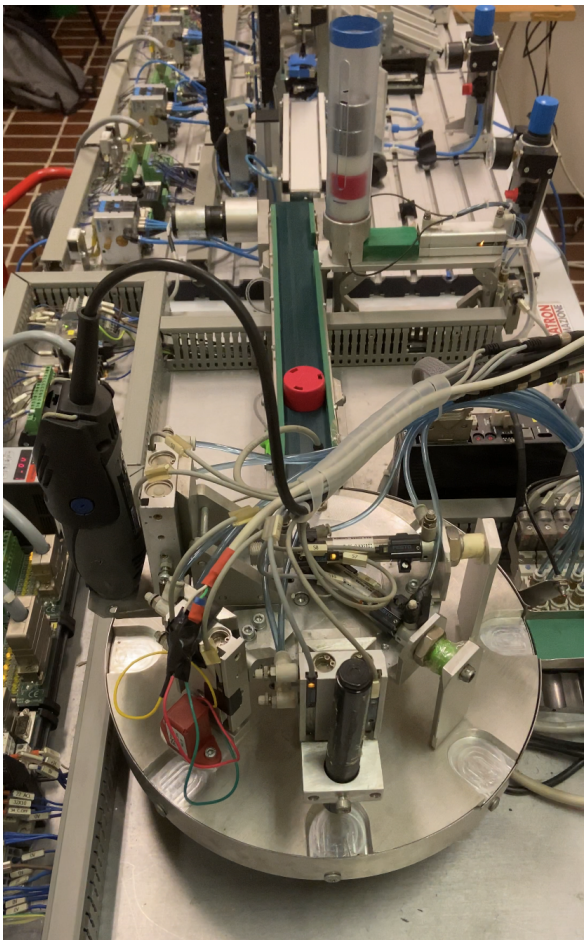
Il **Ladder** è stato il primo linguaggio utilizzato per la programmazione dei PLC e si basa su simboli di provenienza "elettrica" come i contatti e le bobine (coil) e si sviluppa in linee orizzontali chiamate "rung".

L'**ST** (*Structured Text*) è un linguaggio più simile ai linguaggi di programmazione classici per computer come il Pascal, il C, il Cobol e così via.
I linguaggi di questo tipo permettono di scrivere le istruzioni in un modo più simile al nostro modo di pensare e parlare, per questo sono molto efficaci per svolgere certe funzioni.

1.3 Obiettivo

L'obiettivo del progetto è quello di effettuare lavorazioni automatiche sui pezzi inseriti in base alle loro caratteristiche (colore e altezza). In particolare l'impianto deve:

- Trapanare i pezzi di colore rosso
- Colorare i pezzi non rossi alti
- Scartare i pezzi non rossi
- Spingere nella baia i pezzi di colore rosso



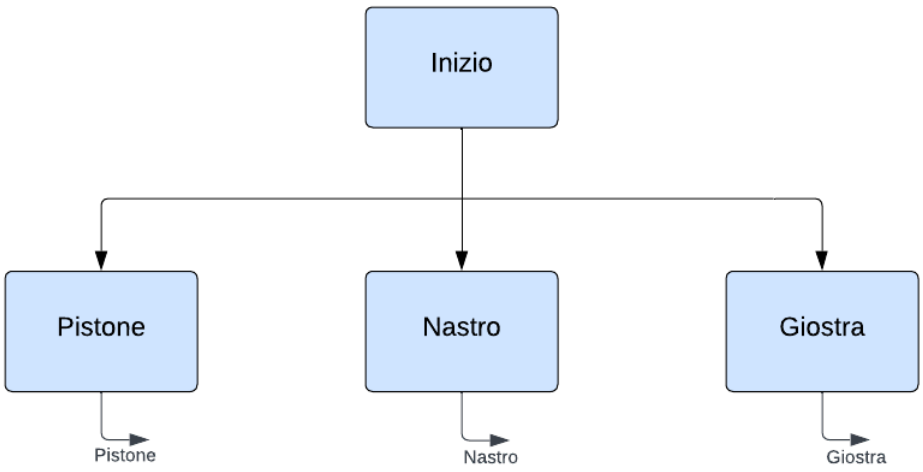
2 SFC

2.1 Struttura generale

Il programma è suddiviso in 3 gruppi:

- Nastro
- Pistone
- Giostra

La struttura può essere descritta dalla seguente figura:



Utilizzando questa struttura i gruppi vengono eseguiti singolarmente in parallelo e in questo in modo che possono lavorare in maniera indipendente gli uni dagli altri.

2.2 Variabili

Avendo utilizzato diversi tipi di linguaggi è stato necessario dover utilizzare anche diversi tipi di variabili. I principali tipi di variabili utilizzate sono state: Int, Bool, Ton, Time e Array.

Oltre alle variabili standard appena elencate abbiamo creato due struct e due enum.
Gli enum sono stati utilizzati per assegnare alla variabile Colore tre possibili valori: colore indefinito, rosso o altro; e ad Altezza i valori: alto o basso.

```
TYPE COLORE: (nienteC,rosso, altro);
END_TYPE
```

```
TYPE ALTEZZA: (nienteA,basso, alto);
END_TYPE
```

Inoltre sono state create due struct: POSTO e PEZZO con all'interno le proprie caratteristiche.

```
TYPE PEZZO :
STRUCT
    Colore : COLORE;
    Altezza : ALTEZZA;
    Trapanato:BOOL:=FALSE;
    Tastato:BOOL:=FALSE;
END_STRUCT
END_TYPE
```

```
TYPE POSTO :
STRUCT
    Presenza :BOOL:=FALSE;
    Pezzo :PEZZO;
END_STRUCT
END_TYPE
```

Queste variabili possono essere divise in base al loro utilizzo in tre gruppi:

2.2.1 Variabili Pistone

```
(*Variabili Pistone*)
    (*Ingressi*)
    pezzo_magazzino AT%IX0.0:BOOL; (*sensore che segnala la presenza di un pezzo nel magazzino*)
    pistone_avanti AT%IX0.2:BOOL; (*sensore che segnala se il pistone è avanti*)
    pistone_indietro AT%IX0.4:BOOL; (*sensore che segnala se il pistone è indietro*)
    N_pezzi_spinti: INT:=0; (*variabile che incrementa di 1 quando viene spinto un pezzo sul nastro*)
    (*Uscite*)
    pistone_on AT%QX0.2:BOOL; (*attuatore del pistone*)

    T_pistone: TON; (*timer pistone di 1.5 s*)
    mytimepistone: TIME;
    timer_pistone: BOOL;
    reset_timer_pistone: BOOL;
```

2.2.2 Variabili Nastro

```
(*Variabili Nastro*)
    (*Ingressi*)
    fine_corsa_nastro AT%IX0.6:BOOL; (*sensore che segnala il passaggio di un pezzo a fine nastro*)
    (*Uscite*)
    nastro_on AT%QX0.0:BOOL; (*attuatore del nastro*)

    T_nastro: TON; (*timer nastro di 1s*)
    mytimenastro: TIME;
    timer_nastro: BOOL;
    reset_timer_nastro: BOOL;
```

2.2.3 Variabili Giostra

```
(*Variabili Giostra*)
    Giostra: ARRAY[0..7] OF POSTO;(*array di 8 posti*)
     i: INT :=0; (*indice che incrementa ad ogni movimento della giostra e usato per lavorare con l'array*)
    giostra_busy AT %IX6.1: BOOL; (*sensore che segnala il movimento della giostra*)
    movimento_giostra AT %QX3.4: BOOL; (*attuatore giostra che la muove di 1 posto*)

    T_giostra: TON; (*timer giostra di 1s*)
    mytimegiostra: TIME;
    timer_giostra: BOOL;
    reset_timer_giostra: BOOL;

    sensore_colore AT %IX2.1: BOOL; (*sensore che segnala i pezzi di colore rosso*)

    trapano_basso AT %IX2.5: BOOL ; (*sensore che segnala se il trapano è basso*)
    trapano_alto AT %IX2.3: BOOL; (*ensore che segnala se il trapano è alto*)
    trapano_on AT %QX3.7:BOOL; (*attuatore per l'attivazione della punta del trapano*)
    trapano_down AT %QX7.0: BOOL; (*attuatore per fare scendere o salire il trapano*)

     T_trapano: TON; (*timer trapano di 1s*)
    mytimetrapano: TIME;
    timer_trapano: BOOL;
    reset_timer_trapano: BOOL;

    tastatore_basso AT %IX5.3: BOOL; (*sensore che segnala se il tastatore è basso*)
    tastatore_alto AT %IX2.7: BOOL; (*sensore che segnala se il tastatore è alto*)
    tastatore_down AT %QX3.6: BOOL; (*attuatore tastatore*)
    tastatore_lettura AT%I*: WORD; (*fornisce la lettura dell'altezza dei pezzi*)
     myInt : INT;

    T_tastatore: TON; (*timer tastatore di 1s*)
    mytimetastatore: TIME;
    timer_tastatore: BOOL;
    reset_timer_tastatore: BOOL;

pennarello_alto AT %IX5.4: BOOL; (*sensore che segnala se il pennarello è alto*)
pennarello_basso AT %IX5.5: BOOL; (*sensore che segnala se il pennarello è basso*)
pennarello_down AT %QX3.3: BOOL; (*attuatore pennarello*)

 T_pennarello: TON;(*timer pennarello di 1s*)
mytimepennarello: TIME;
timer_pennarello: BOOL;
reset_timer_pennarello: BOOL;

pistone_PT5_avanti AT %IX5.0: BOOL; (*sensore che segnala se il pistone 5 è avanti*)
pistone_PT5_indietro AT %IX5.1: BOOL; (*sensore che segnala se il pistone 5 è indietro*)
pistone_PT5_on AT %QX3.1:BOOL ; (*attuatore pistone 5*)

pistone_PT6_avanti AT %IX5.7: BOOL;  (*sensore che segnala se il pistone 6 è avanti*)
pistone_PT6_indietro AT %IX5.6: BOOL; (*sensore che segnala se il pistone 6 è indietro*)
pistone_PT6_on AT%QX3.5: BOOL; (*attuatore pistone 6*)
pezzo_baia AT %IX5.2: BOOL; (*sensore che segnala la presenza di un pezzo sulla baia*)
```

2.2.4 Altre Variabili

```
(*Variabili utilizzate nel FOR per la verificare alcune condizioni *)

    j: INT; (*Indice  FOR 1_posto_true*)
    posto_true: BOOL; (*Condizione di ALMENO 1 posto con presenza a TRUE*)

    k: INT; (*Indice  FOR solo_pezzi_rossi*)
    solo_pezzi_rossi: BOOL; (*Condizione SOLO pezzi rossi e  baia occupata*)

    h: INT; (*Indice FOR giostra_piena*)
    giostra_piena: BOOL; (*Condizione TUTTI i pezzi con presenza TRUE*)

    p: INT; (*Indice FOR giostra_vuota*)
    giostra_vuota: BOOL; (*Condizione se ho TUTTI i pezzi con presenza FALSE*) (*forse ridondante*)

    blocco_giostra: BOOL; (*Condizione per bloccare la giostra se ho un pezzo rosso in PT6 e la  baia è già occupata*)
```

2.3 Funzionamento e diagrammi SFC

2.3.1 Pistone

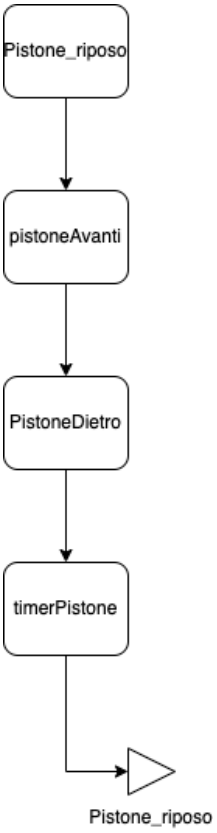
Sono stati implementati due possibili utilizzi del pistone:

- **Pistone V1** (progettoSFC-V1): in questa implementazione la logica utilizzata è abbastanza semplice e lineare.

  Si inizia dallo stato di riposo del pistone dal quale se è presente un pezzo in magazzino, il sensore del pistone indietro è TRUE, NPezziSpinti = 0 e NON è presente un pezzo nel primo posto della giostra (nella postazione P1) allora si passa allo stato successivo dove il pistone spinge il pezzo e tramite una **exit action** la variabile NPezziSpinti incrementa di uno. Una volta spinto il pezzo il pistone ritorna nella posizione di riposo e riparte il ciclo.

- **Pistone V2** (progettoSFC-V2): in questa implementazione la logica è simile alla precedente ma abbiamo inserito un timer ed modificato/aggiunto alcune condizioni, infatti, appena il pistone spinge il pezzo e torna indietro si attiva un timer di 1.5 secondi e una volta raggiunto il tempo prefissato si torna alla posizione di riposo e riparte il ciclo.

  Tutto ciò permette di immettere pezzi nel nastro ogni 1.5 secondi, inoltre, la logica prevede un'altra condizione che fa in modo che nel nastro non siano presenti più di 3 pezzi contemporaneamente.



2.3.2 Nastro

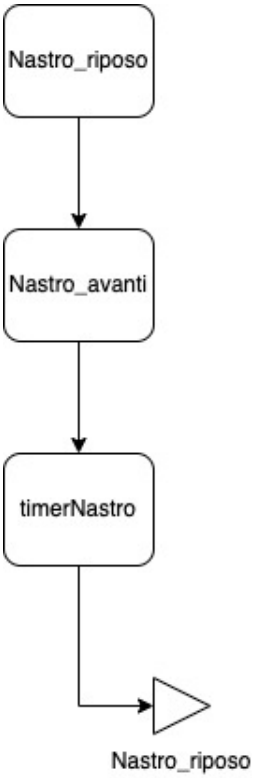
Anche per il nastro sono stati implementati due possibili utilizzi:

- **Nastro V1** (progettoSFC-V1): in questa implementazione la logica è semplice e lineare.

  Si inizia dallo stato "Nastro_riposo" che indica il nastro fermo, non appena il NPezziSpinti è maggiore di zero, la giostra è ferma e non è presente un pezzo nel primo posto della giostra (nella postazione P1) si entra nello stato "Nastro_avanti" e il nastro si avvia, inoltre, in questo stato è presente una **entry action** dove si controlla attraverso un ciclo FOR se la giostra è piena. Dallo stato "Nastro_avanti" se la giostra è piena oppure è in movimento si torna allo stato iniziale di riposo e, quindi, il nastro si ferma e il ciclo può ripartire.

- **Nastro V2** (progettoSFC-V2): in questa implementazione la logica è simile alla precedente ma abbiamo inserito un timer, infatti, dallo stato "Nastro_avanti" appena il pezzo attraversa il fine corsa si entra nello stato "timerNastro" nel quali si attiva il timer nastro di 1 secondo e se la giostra è piena o se è in movimento oppure se il timer del nastro è vero si torna nella posizione di riposo, il nastro si ferma e, quindi, riparte il ciclo.

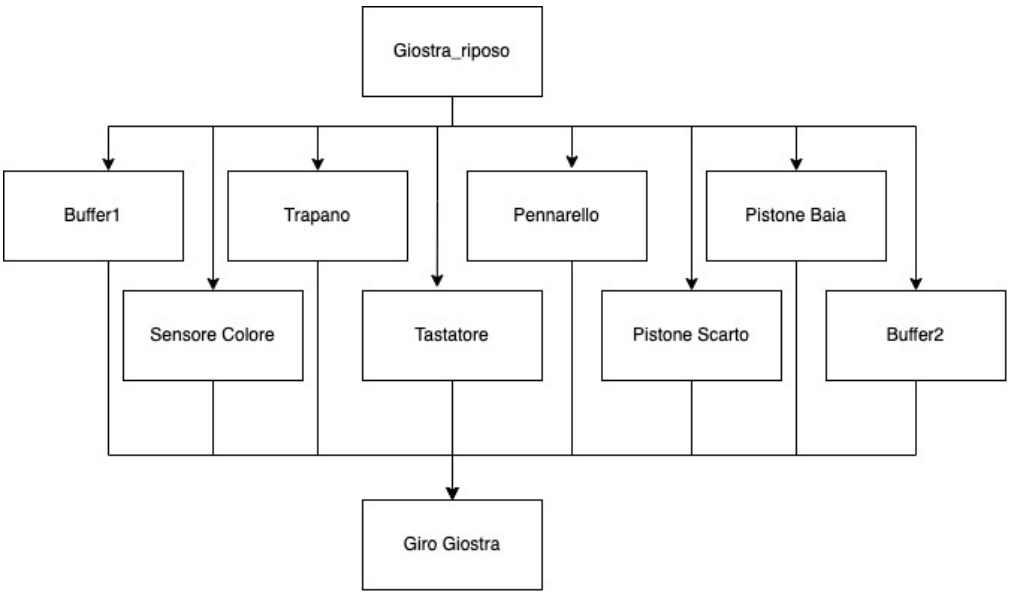
  Tutto ciò permette di avere più pezzi contemporaneamente nel nastro.



2.3.3 Giostra

La logica della giostra è stata pensata in modo tale da garantire la gestione indipendente di ogni postazione dalle altre, permettendo ad ogni attuatore di lavorare in parallelo e solo quando tutte le "operazioni" in ogni postazione si sono concluse la giostra può girare.
Ogni ramo ha una struttura generica composta uno stato di inizio (riposo), uno o più stati che si occupano delle operazioni da eseguire e uno stato di fine (attesa).

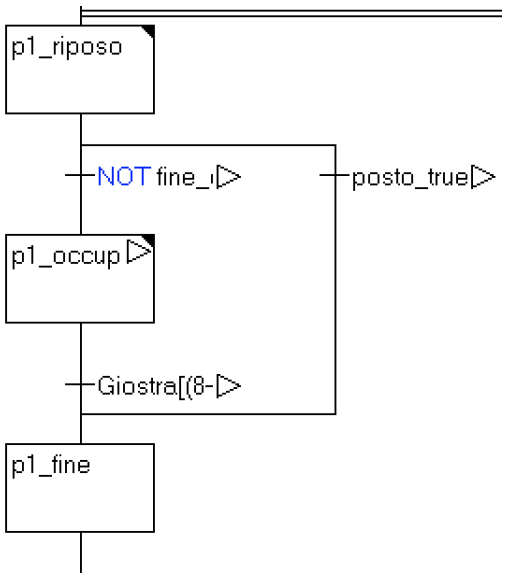
La logica può essere descritta dal seguente schema:



Postazione 0: Buffer p1
La postazione 0 è la postazione per l'ingresso dei pezzi nella giostra. Si parte dallo stato "p1_riposo" dove attraverso un ciclo FOR si controlla se c'è almeno un pezzo all'interno della giostra con presenza a TRUE e da questo stato si hanno due possibilità

- se un pezzo attraversa il fine corsa viene settata la sua presenza a true e decrementa di 1 la variabile NpezziSpinti, successivamente se si ha la presenza di un pezzo si entra nello stato "p1_fine" e si attendono le altre postazioni.
- se, invece, sono presenti già pezzi nella giostra e non si immettono nuovi pezzi si arriva direttamente allo stato "p1_fine" e, quindi non si setta la presenza di nessun nuovo pezzo.

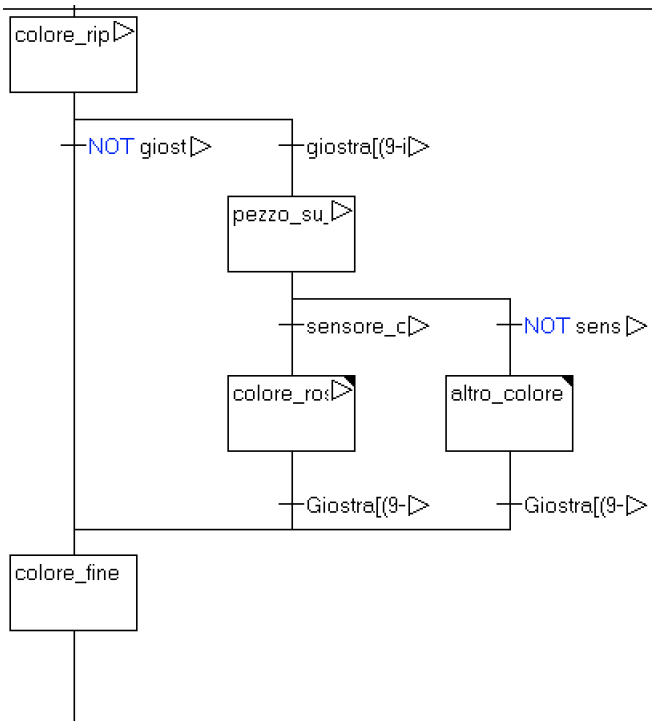
In ogni caso si raggiunge lo stato "p1_fine" e si attende la fine della lavorazione nelle altre postazioni.



Postazione 1: Sensore Colore
La postazione 1 è la postazione utilizzata per rilevare il colore dei pezzi presenti nella giostra.
Si parte dallo stato "colore_riposo" dal quale si hanno due possibilità:

- Se il pezzo non è presente si entra subito nello stato "colore_fine" e, quindi, non viene settato nessun colore.
- Se il pezzo è presente allora si entra nello stato "pezzo_su_nastro" dal quale abbiamo 2 ulteriori possibilità:
  - se il sensore colore è alto si entra nello stato "colore_rosso" dove si imposta il colore del pezzo a rosso.
  - se, invece, il sensore colore è basso si entra nello stato "altro_colore" dove si imposta il colore ad "altro".

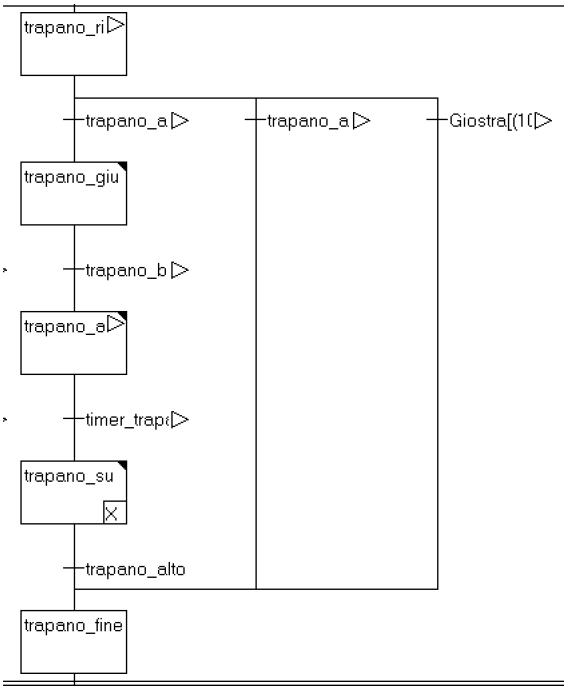
In ogni caso si raggiunge lo stato "colore_fine" e si attende la fine della lavorazione nelle altre postazioni.



Postazione 2: Trapano
La postazione 2 è la postazione utilizzata per trapanare i pezzi rossi.
Si parte dallo stato "trapano_riposo" dal quale si hanno 3 possibilità:

- Se il valore del sensore "trapano_alto" è alto, il pezzo presente è di colore rosso, non è già stato trapanato e la giostra non è in movimento allora il trapano si abbassa, inizia la lavorazione della durata di 1 secondo e, inoltre, a fine lavorazione nella **exit action** dello stato "trapano_su" viene settata la variabile Trapanato (del pezzo appena lavorato) a TRUE.
- Se il pezzo non è presente o non è di colore rosso allora non viene lavorato.
- oppure, se il pezzo è già stato trapanato anche in questo caso non viene lavorato.

  In ogni caso si raggiunge lo stato "trapano_fine" e si attende la fine della lavorazione nelle altre postazioni.

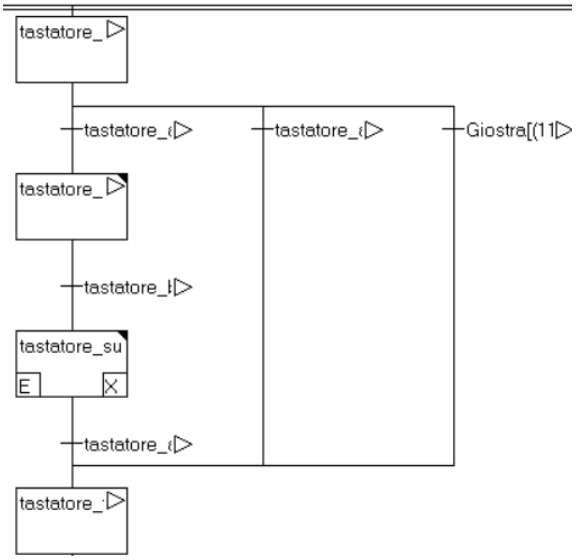


Postazione 3: Tastatore
La postazione 3 è la postazione utilizzata per rilevare l'altezza dei pezzi presenti nella giostra tramite l'ausilio di un tastatore.
Si parte dallo stato "tastatore_riposo" dal quale si hanno 3 possibilità:

- Se il pezzo non è presente si entra subito nello stato "tastatore_fine" e, quindi, non viene settata nessuna altezza.
- Se il pezzo è presente ma è già stato tastato anche in questo caso si entra subito nello stato "tastatore_fine" e, quindi, non viene settata nessuna altezza.
- se il pezzo è presente, non è stato tastato e la giostra non è in movimento allora viene tastato e utilizzando un IF in base al valore della variabile tastatore_lettura viene settata l'altezza del pezzo, abbiamo 2 possibilità:
  - se tastatore_lettura (convertito nell'ST presente nell'entry action tramite la funzione WORD TO INT) è superiore di 30000 si setta l'altezza ad alto.
  - se tastatore_lettura (convertito nell'ST presente nell'entry action tramite la funzione WORD TO INT) è inferiore a 30000 si setta l'altezza a basso.

  Una volta tastato il pezzo, inoltre, tramite l'**exit action** dello stato "tastatore_su" viene settata la variabile Tastato (del pezzo appena lavorato) a TRUE.

In ogni caso si raggiunge lo stato "tastatore_fine" e si attende la fine della lavorazione nelle altre postazioni.

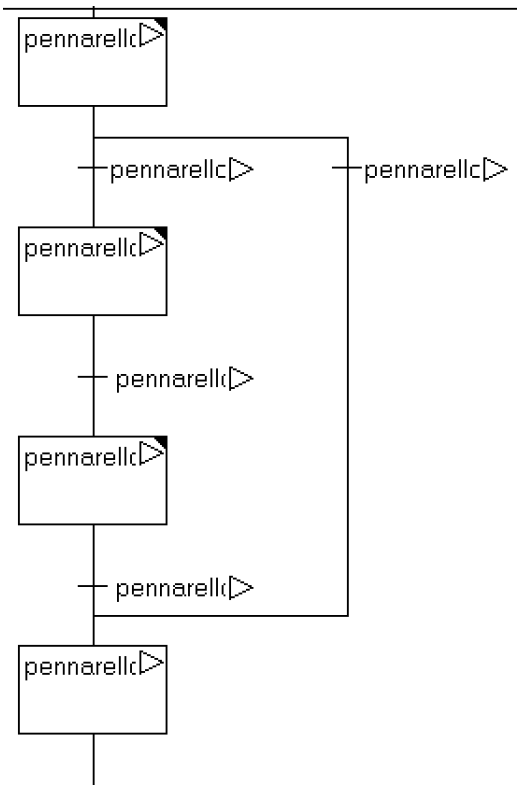


Postazione 4: Pennarello
La postazione 4 è la postazione utilizzata per colore i pezzi presenti nella giostra tramite l'ausilio di un pennarello.

Si parte dallo stato "pennarello_riposo" dal quale si hanno 2 possibilità:

- Se il pezzo è il pezzo è presente, di colore = altro e di altezza = alto, il sensore pennarello_alto è TRUE e la giostra non è in movimento allora il pennarello si abbassa e inizia la lavorazione di 1 secondo e una volta passato il tempo prefissato si entra nello stato "pennarello_fine".
- Se, invece, non c'è un pezzo o il colore è rosso o l'altezza è uguale a basso si entra direttamente nello stato "pennarello_fine" saltando tutta la lavorazione.

In ogni caso si raggiunge lo stato "pennarello_fine" e si attende la fine della lavorazione nelle altre postazioni.



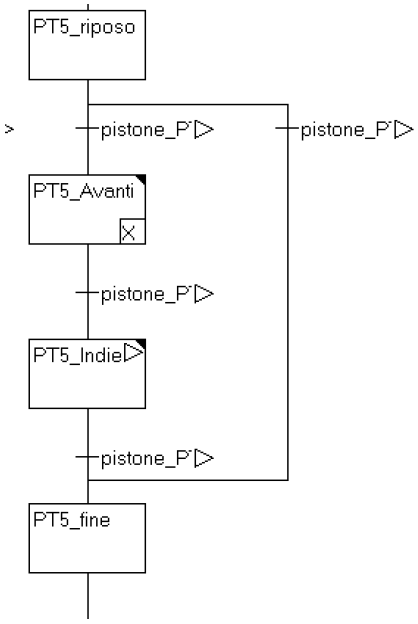
Postazione 5: Pistone Scarto
La postazione 5 è la postazione utilizzata per spingere i pezzi di colore "altro" in un "buffer illimitato".

La logica del pistone di scarto è simile alla logica del pistone che immette pezzi sul nastro con la differenza che una volta "scartato" il pezzo vengono eseguiti una serie di settaggi:

- Presenza = FALSE.
- Tastato = FALSE.
- Colore = nienteC.
- Altezza = nienteA.

In questa postazione l'unica condizione per cui non si scartano pezzi è che non ci siano pezzi o se presenti siano di colore rosso.

In ogni caso si raggiunge lo stato "PT5_fine" e si attende la fine della lavorazione nelle altre postazioni.



Postazione 6: Pistone Baia
La postazione 6 è la postazione utilizzata per spingere i pezzi di colore "rosso" in un "buffer limitato".

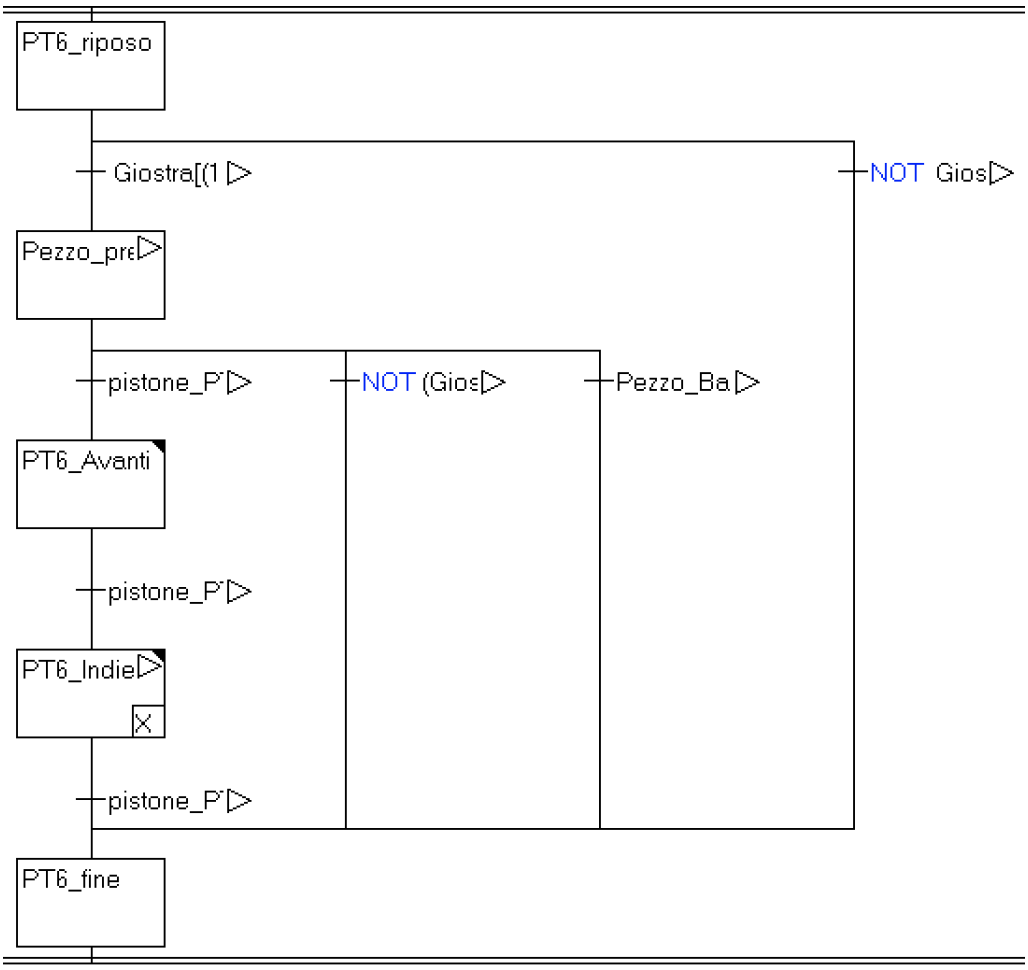
La logica del pistone di scarto è simile alla logica del pistone della postazione precedente ma in questo caso il buffer è limitato ad 1 posto.

La condizione di non lavorazione per questa postazione è se il pezzo non è presente o il pezzo non è di colore rosso o un pezzo rosso è nella baia.
Se, invece, il pezzo è presente, è rosso e non c'è nessun pezzo nella baia allora il pistone lo spinge su quest'ultima.

A fine lavorazione, come nella postazione precedente, si hanno una serie di settaggi:

- Presenza=FALSE.
- Trapanato=FALSE.
- Tastato=FALSE.
- Colore=nienteC.
- Altezza=nienteA.

In ogni caso si raggiunge lo stato "PT6_fine" e si attende la fine della lavorazione nelle altre postazioni.

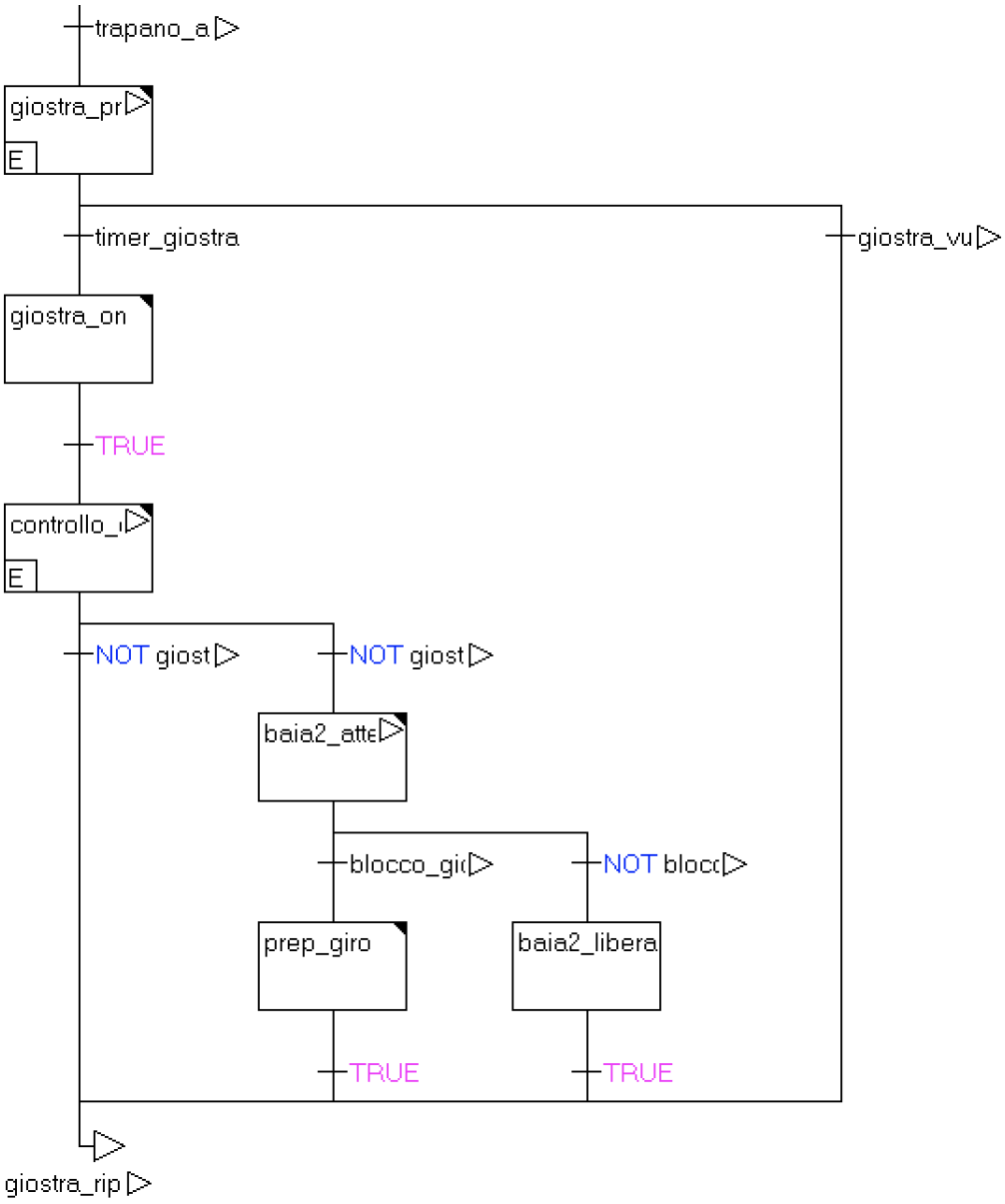


Postazione 7: Buffer 2
Questa postazione è un buffer come la postazione 1 dove c'è solo uno stato di riposo dove non avvengono azioni poichè non sono presenti attuatori.

Giro Giostra
Una volta che per ogni ramo di ogni attuatore si raggiunge lo stato do fine per effettuare un giro della giostra avviene un controllo di tutti i sensori degli attuatori(trapano_alto, tastatore_alto, ecc...) e in questo modo si ha la certezza che tutti i motori abbiano finito la loro lavorazione da qui si entra nello stato "giostra_pronta" si attiva un timer di 1 secondo, viene controllato tramite ciclo FOR se il vettore dei posti è vuoto e si hanno due possibilità:

- se la giostra è vuota e, quindi, la condizione "giostra_vuota" è vera si attiva la transizione che rimanda allo stato "giostra_riposo" saltando tutta la parte che permette alla giostra di muoversi.
- Se la giostra non è vuota, gira incrementando l'indice "i" di 1 e, tramite l'ausilio di un IF e di un ciclo FOR, avviene un secondo controllo:
  - se la baia è occupata si controlla se si hanno solo pezzi rossi, se è vera la giostra continua a girare senza utilizzare attuatori poiché i pezzi sono già stati lavorati.
  - se la baia non è occupata si controlla se ho almeno un pezzo non rosso. Nel caso in cui si hanno solo pezzi rossi ci sono due casi:
    - Se ci sono solo pezzi rossi e non ho nessun pezzo nel nastro la giostra si blocca davanti alla baia nell'attesa che si liberi.
    - Se ci sono solo pezzi rossi ma vengono immessi uno o più pezzi nel nastro allora si procedono con le relative lavorazioni e si ritorna nel caso precedente.

  In ogni caso si ritorna allo stato "giostra_riposo" in modo tale da poter lavorare in modo ciclico.


trapano_a
giostra_pr
E
timer_giostra
giostra_vu
giostra_on
TRUE
controllo_
E
NOT giost
NOT giost
baia2_atte
blocco_gi
NOT bloc
prep_giro
baia2_libera
TRUE
TRUE
giostra_rip

3 Conclusioni

Durante lo sviluppo del progetto abbiamo individuato due diversi modi per implementare l'utilizzo del nastro e del pistone e su come interfacciarli con la giostra, entrambi sono simili come logica ma hanno prestazioni diverse:

- il primo (progettoSFC-V1) con **prestazioni inferiori** e con una **maggiore sicurezza** poichè permette di immettere 1 solo pezzo alla volta sul nastro aumentando quindi i tempi di riempimento della giostra e di lavorazione di tutti i pezzi disponibili.
- il secondo (progettoSFC-V2) con **prestazioni superiori** ma una **minore sicurezza** (però sempre accettabile) poichè permette di immettere più pezzi alla volta sul nastro diminuendo quindi i tempi di riempimento della giostra e di lavorazione di tutti i pezzi disponibili.

In ogni caso dalle prove eseguite risulta che sono state soddisfatte tutte le specifiche e in particolare non si riscontrano situazioni di non sicurezza o malfunzionamenti in cui gli attuatori si attivano quando non dovrebbero anche potrebbe accadere che alcuni pezzi incastrino nel nastro trasportatore o sul pistone ma sbloccandoli manualmente dovrebbe funzionare tutto correttamente.