Università Politecnica delle Marche

Corso di Laurea
Ingegneria Informatica e dell'Automazione

Automazione Industriale

Relazione
Controllo impianto industriale

A.A 2021/2022

Studenti:
Mecozzi Fabio, S1094003
Pantella Edoardo, S1093665

Introduzione

Controllo impianti industriali

I processi industriali sono di solito processi molto complessi ma nei quali è possibile individuare sottosistemi indipendenti che possono essere controllati indipendentemente l'uno dall'altro. Tuttavia è necessario un lavoro di coordinamento tra i vari sottosistemi al fine di raggiungere l'andamento desiderato dell'impianto industriale che viene effettuato tramite un controllo logico basato sullo stato del sistema e sugli eventi che avvengono nel sistema. Esso viene infatti modellato come un sistema dinamico, non lineare, guidato da eventi e a tempo discreto.

Hardware

Glossario

- **Postazione**: area della giostra su cui può essere posizionato un attuatore o un sensore (esempio: la postazione del trapano è l'area sottostante al trapano). La postazione 1 è l'ingresso della giostra e la numerazione segue il verso antiorario (guardando la giostra dall'alto). L'ultima postazione non ha né attuatori né sensori ma è solamente un buffer.
    - **Ingresso giostra:** postazione 1
    - **Baia giostra:** buffer davanti alla postazione 7
    - **Buffer giostra:** postazione 8
- **Posto:** alloggio per un pezzo nella giostra: il posto uno è quello in ingresso al nastro al momento del reset del sistema e in quel momento corrisponde alla postazione 1. Dopo una rotazione della giostra il posto 1 corrisponde alla postazione 2 (sensore del colore), il posto 2 corrisponde alla postazione 3 (trapano) e così via fino al posto 8 che corrisponde alla postazione 1 (ingresso giostra).

Nota: per una più semplice gestione delle variabili nel software è stata utilizzata una numerazione che va da 0 a 7, quindi nella sezione SOFTWARE verrà utilizzata questa seconda numerazione (ad esempio l'ingresso della giostra verrà indicato come postazione 0)

Impianto

Abbiamo avuto a che fare con un impianto in scala che simula la lavorazione di pezzi tramite un processo lineare: i pezzi vengono caricati in un buffer e spinti su un nastro trasportatore per opera di un pistone; successivamente vengono trasportati in una giostra sulla quale possono essere effettuate diverse operazioni automatiche, nell'ordine:

- Rilevazione colore rosso o diverso da rosso (postazione 2)
- Trapanazione (postazione 3)
- Rilevazione altezza tramite tastatore (postazione 4)
- Colorazione del pezzo tramite pennarello (postazione 5)
- Scarto del pezzo su buffer tramite pistone (postazione 6)
- Scarto del pezzo su baia tramite pistone (postazione 7)

Controllo

Il controllo viene effettuato tramite computer che si interfaccia all'impianto comunicando tramite LAN con un PLC collegato ai sensori e agli attuatori dell'impianto.

Linguaggi di programmazione

Il principale linguaggio di programmazione utilizzato è stato SFC, Sequential Function Chart, ovvero un linguaggio di programmazione grafico per i PLC.
I vantaggi di questo linguaggio di programmazione sono l'elevata leggibilità, l'espressività, essendo sintetico e non ambiguo, e la manutenibilità, essendo infatti possibile modificare i singoli passi senza alterare il resto del programma.
Oltre all'SFC, per alcune piccole parti del programma sono stati usati anche i linguaggi Ladder e ST.
Il Ladder è stato il primo linguaggio di programmazione per i PLC; è un linguaggio a contatti che si articola su varie linee orizzontali dette "rung".
L'ST (Structured Text) invece è un linguaggio di programmazione testuale di alto livello, simile al PASCAL o al C. In esso il codice del programma è composto di espressioni e di istruzioni.

Strumenti software

Lo strumento software utilizzato per interfacciarsi con il PLC è stato il pacchetto Twincat di Beckhoff utile per configurare, programmare, simulare, diagnosticare ed effettuare il debug di applicazioni utente e programmi volti al controllo di processi industriali.

Obiettivo del progetto

L'obiettivo del progetto è quella di effettuare lavorazioni automatiche sui pezzi (tramite attuatori) e selezionare i pezzi che vengono inseriti nell'impianto in base a caratteristiche fisiche degli stessi (colore e altezza) per poi essere scartati nei rispettivi spazi, il tutto seguendo una logica stabilita a priori.
In particolare abbiamo scelto la seguente logica:

- Trapanare i pezzi rossi e scartarli nella baia
- Colorare con pennarello i pezzi bassi
- Scartare nel buffer i pezzi non rossi

Obiettivi qualitativi

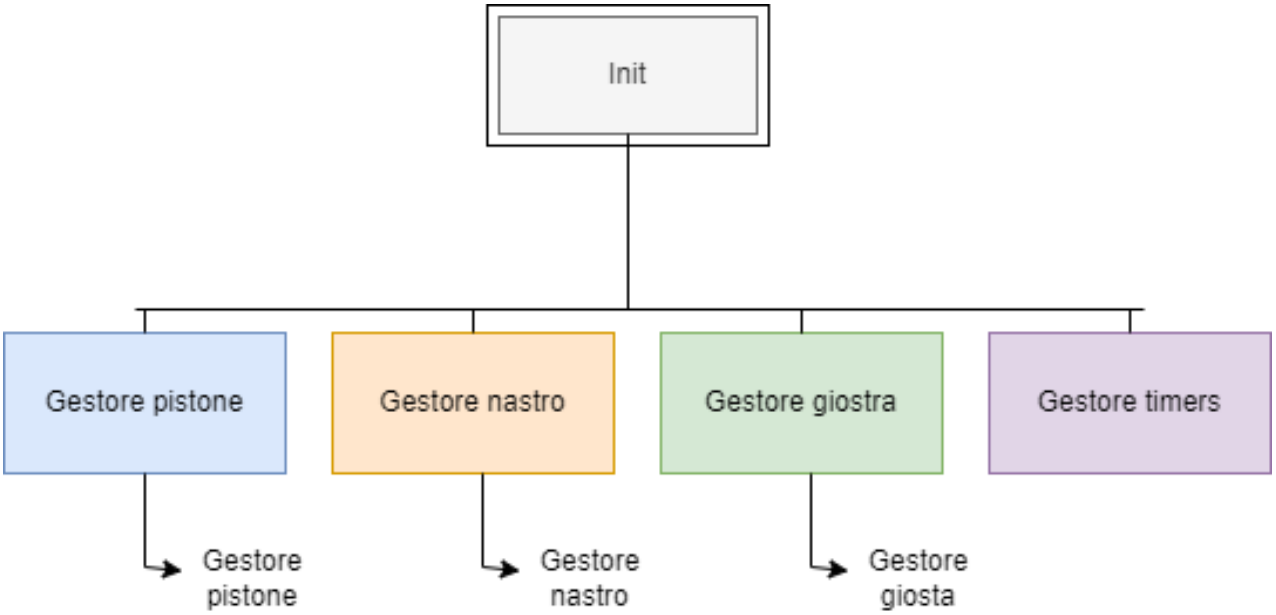
Nello sviluppo del progetto ci siamo attenuti ai seguenti obiettivi per le decisioni di progettazione:

- **Sicurezza**: gli attuatori e altri pezzi del sistema non devono rischiare la rottura durante l'evoluzione del sistema
- **Efficienza**: il throughput del sistema (numero di pezzi lavorati su unità di tempo) deve essere sufficientemente elevato
- **Controllabilità**: evitare che il sistema raggiunga stati non controllabili ovvero stati attraverso i quali esiste una serie di eventi che portano il sistema a non espletare il proprio compito

Software

Il programma si compone di 4 parti principali: data type, variabili, azioni e diagramma sfc.
Per quanto riguarda la struttura principale del diagramma sfc, esso è stato diviso in 5 macrosezioni: init, pistone, nastro, giostra e timer.

```
0001 TYPE ALTEZZA: (nd_A, basso, alto);
0002 END_TYPE
```



Schema generale main

Abbiamo scelto questa configurazione così che, dopo l'inizializzazione, pistone, nastro giostra e timer possono essere eseguiti in parallelo. Perciò i singoli gestori

sono stati realizzati per lavorare in maniera indipendente gli uni dagli altri, fatta eccezione per semplici interfacce realizzate tramite variabili condivise con le quali i componenti si appropriano di risorse che non possono essere utilizzate contemporaneamente (ad esempio l'ingresso della giostra: la giostra non può girare se il nastro sta inserendo un pezzo).

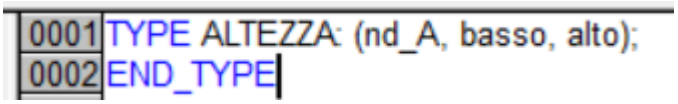
Data type

In questa sezione sono definiti i tipi di dati utili alla gestione dell'impianto.

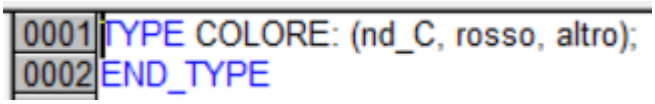
Enum

Gli enum sono costanti definite dall'utente utili per una maggior chiarezza del codice. Sono stati definiti per descrivere le caratteristiche fisiche di un pezzo.

1. Altezza: descrive se il pezzo è più alto o più basso di un certo valore soglia (nd_a indica che l'altezza del pezzo non è ancora stata misurata);
2. Colore: descrive se il pezzo è del colore riconosciuto dal sensore (nd_c indica che il colore del pezzo non è ancora stato rilevato);



Definizione enum "ALTEZZA"

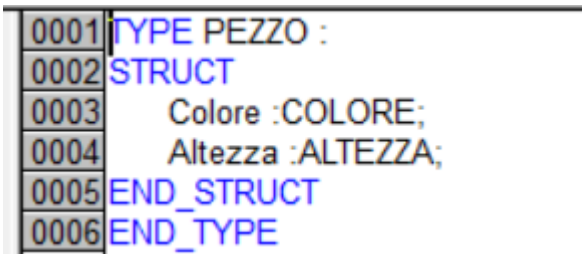


Definizione enum "COLORE"

Struct

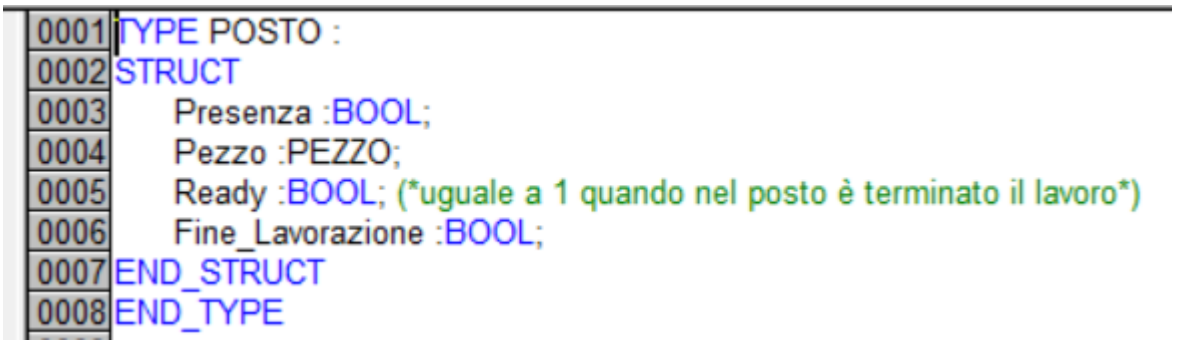
Le strutture dati sono definite dall'utente per accorpare variabili logicamente correlate al fine di una più semplice gestione. Sono stati definiti per descrivere lo stato delle postazioni della giostra.

1. Pezzo: descrive le caratteristiche di un pezzo nella giostra (sfruttando gli enum precedentemente descritti);
2. Posto: descrive le caratteristiche di un posto della giostra.



```
0001 TYPE PEZZO :
0002 STRUCT
0003     Colore :COLORE;
0004     Altezza :ALTEZZA;
0005 END_STRUCT
0006 END_TYPE
```

Definizione struct “PEZZO”



```
0001 TYPE POSTO :
0002 STRUCT
0003     Presenza :BOOL;
0004     Pezzo :PEZZO;
0005     Ready :BOOL; (*uguale a 1 quando nel posto è terminato il lavoro*)
0006     Fine_Lavorazione :BOOL;
0007 END_STRUCT
0008 END_TYPE
```

Definizione struct “POSTO”

Variabili

Per l’esecuzione del programma sono state dichiarate delle variabili che possono essere divise in 4 categorie in base alla loro funzione logica:

1. Variabili per il pistone
2. Variabili per il nastro
3. Variabili per la giostra
4. Variabili per le interfacce tra pistone, nastro e giostra

Le variabili per le interfacce sono utilizzate al fine di soddisfare l’obiettivo della sicurezza: i componenti non sono consapevoli dell’interazione reciproca e quindi il mancato controllo di queste interazioni può portare a condizioni di non sicurezza per i vari componenti oppure ad evoluzioni non prevedibili del sistema.
Per le interfacce tra i componenti sono stati utilizzati anche timer perchè esistono eventi non osservabili, quali la presenza di un pezzo nella porzione di nastro davanti al pistone oppure l’avvenuto inserimento di un pezzo all’interno della giostra: se il sistema trascorre una certa quantità di tempo in taluni stati possiamo supporre che questi eventi si siano verificati.

```
0004 (*Variabili Pistone*)
0005     Presenza_Pezzo AT %IX0.0: BOOL;
0006     Pistone_Indietro AT %IX0.4: BOOL;
0007     Pistone_Avanti AT %IX0.2: BOOL;
0008     Pistone_On AT %QX0.2: BOOL;
```

Definizione variabili pistone

```
0010 (*Variabili nastro*)
0011     Nastro_On AT %QX0.0: BOOL;
0012     Fine_Corsa AT %IX0.6: BOOL;
0013     NumPeN: INT:= 0;
```

Definizione variabili nastro

```
0015 (*Variabili gestione timers per non creare conflitti tra nastro pistone e giostra*)
0016 TimerNP: TON;
0017 Timer_GN: TON;
0018 Timer_Nastro: TON;
0019 ResetTimer: BOOL := FALSE;
0020 AttesaNP: TIME:= t#0.6s;
0021 Q: BOOL;
0022 Delay_Pistone: TON;
0023 Ritardo_Pistone: BOOL;
0024
0025
0026 (*Interfaccia Nastro-Giostra*)
0027 InserimentoNG: BOOL:=FALSE; (*Il nastro deve mettere a TRUE quando sta inserendo un pezzo all'interno della giostra e a FALSE quando non lo sta facendo*)
0028 FineInserimentoGN: BOOL;
0029 NReady: BOOL := TRUE;    (*Vero quando il nastro autorizza il pistone a spingere un nuovo pezzo*)
0030 GStart: BOOL := TRUE;    (*Vero quando la giostra è autorizzata a girare*)
```

Definizione variabili interfacce

```
0033 (*variabili giostra*)
0034 Movimento_Giostra AT %QX3.4: BOOL;
0035 Giostra_BUSY AT %IX6.1: BOOL;
0036 Giostra: ARRAY[0..7] OF POSTO; (*Array che descrive lo stato di tutti i posti della giostra*)
0037 TEMP: ARRAY[0..7] OF BOOL; (*Array di supporto per verificare delle condizioni su array di 8 elementi*)
0038 presenza_fine_lav: BOOL; (*Variabile che deve essere settata a TRUE quando tutti i pezzi presenti nella giostra hanno finito la lavorazione oppure non è presente nessun pezzo nella giostra*)
0039 i: INT:=0; (*Contatore per il numero di movimenti della giostra (8 movimenti = un giro)*)
0040 index: INT; (*Variabile di supporto per iterare sull'array*)
0041
0042 Sensore_Colore AT %IX2.1: BOOL;
0043
0044 Trapano_Basso AT %IX2.5: BOOL := FALSE;
0045 TrapanoOn AT %QX3.7:BOOL;
0046 TimerTrapano: TON; (*timer che tiene conto del tempo di lavorazione trascorso per il pezzo sotto al trapano*)
0047 FineTrapanazione: BOOL := FALSE;
0048 Trapano_Alto AT %IX2.3: BOOL;
0049 Trapano_Down AT %QX7.0: BOOL;
0050
0051 Tastatore_Basso AT %IX5.3: BOOL;
0052 Tastatore_Alto AT %IX2.7: BOOL;
0053 Tastatore_Down AT %QX3.6: BOOL := FALSE;
0054 ValTastatore: REAL; (*Valore di output del tastatore*)
0055
0056 Pennarello_Down AT %QX3.3: BOOL;
0057 Pennarello_Alto AT %IX5.4: BOOL;
0058 Pennarello_Basso AT %IX5.5: BOOL;
0059 Fine_Colorazione: BOOL;
0060
0061 Pistone_PT5_On AT %QX3.1:BOOL := FALSE;
0062 Pistone_PT5_Avanti AT %IX5.0: BOOL;
0063 Pistone_PT5_Indietro AT %IX5.1: BOOL;
0064
0065 Pistone_PT6_On AT%QX3.5: BOOL := FALSE;
0066 Pistone_PT6_Avanti AT %IX5.7: BOOL;
0067 Pistone_PT6_Indietro AT %IX5.6: BOOL;
0068 Pezzo_Baia AT %IX5.2: BOOL;
```

Definizione variabili giostra

Array giostra

L'array descrive i posti della giostra con i dati dei relativi pezzi. Il booleano PRESENZA è TRUE quando un pezzo è presente nel corrispondente posto, FALSE altrimenti. Pezzo descrive gli attributi fisici di un pezzo (altezza e colore sono settati su nd_A e nd_C quando gli attributi non sono ancora noti). Ready è TRUE quando l'eventuale lavoro nella corrispondente postazione è terminato, FALSE altrimenti. Fine_Lavorazione è TRUE quando la giostra ha terminato tutte le misurazioni e ha effettuato tutte le lavorazioni.
I dati riguardanti i pezzi arrivano per postazione (ad esempio Sensore_Colore dà l'informazione che il pezzo nella POSTAZIONE 1 è rosso o non rosso); quindi si è utilizzata la seguente formula per passare da un postazione *pn* ad una posizione *ps*:

$$ps = (8 + pn - i) \,\%\, 8$$

dove *i* è il numero di scatti della giostra e *%* è l'operazione modulo, con la condizione che

$$0 \leq i \leq 7$$

quindi quando la variabile i è uguale a 7 e la giostra gira viene impostata a 0.

presenza_fine_lav

presenza_fine_lav è una variabile booleana che è vera quando tutti i pezzi presenti nella giostra hanno finito la lavorazione o quando non ci sono pezzi nella giostra, è falsa altrimenti.

Azioni

Per semplificare la gestione delle variabili sono state create alcune azioni che possono essere attivate/disattivate nei vari stati del diagramma SFC: esse sono state scritte in Ladder, Structured Text o SFC.
Descriviamo di seguito le principali azioni:

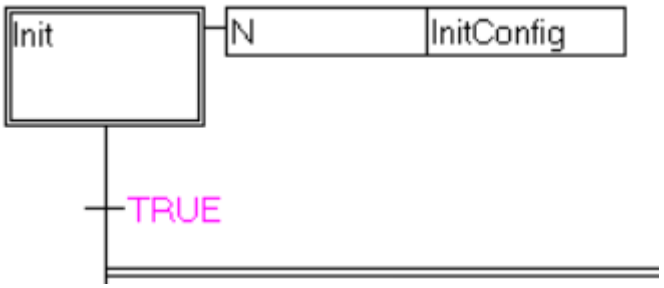
- **AggiungiPezzoAGiostra**: azione da attivare quando un pezzo viene inserito nella postazione 0 della giostra;
- **ResetReady**: azione da attivare quando viene girata la giostra così da permettere alle postazioni di effettuare le eventuali lavorazioni;
- **SetResetTimers**: azione SFC che gestisce la logica di *timerNP* (deve essere sempre attiva);
- **GestoreTimers**: azione che gestisce *Timer_GN*, *TimerTrapano* e, insieme a *GestoreTimers*, anche *TimerNP* (deve essere sempre attiva).

Diagramma SFC

Nel diagramma esistono vari stati il cui nome inizia con “**Dummy**”: essi non sono altro che segnaposti che permettono ai **jump** di saltare in specifiche posizioni del diagramma o di ciclare.

Init

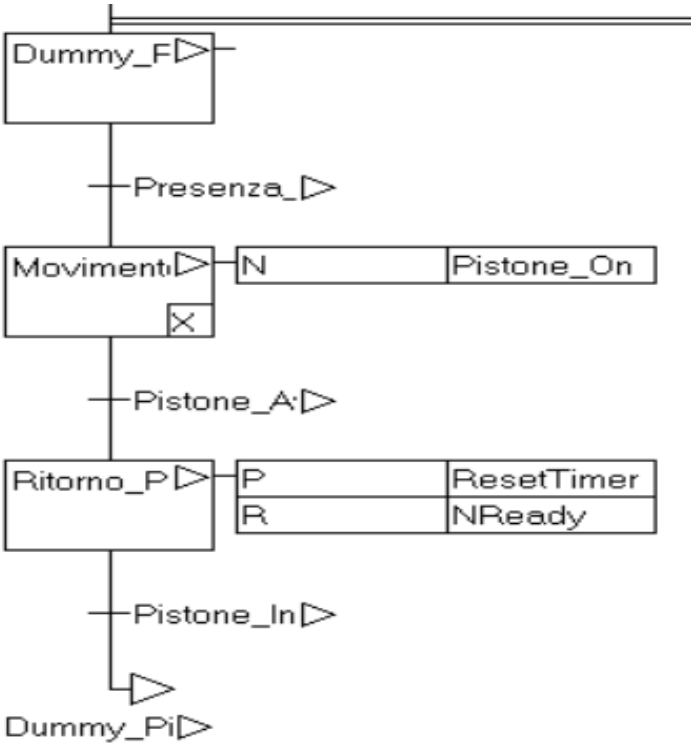
Stato iniziale in cui si inizializzano dei parametri.



Codice Init

Pistone

La logica di gestione del pistone è molto semplice e lineare. Si inizia con il dummy, dopodichè se è presente un pezzo, il nastro è pronto e la variabile booleana Ritardo_Pistone è TRUE allora si passa allo stato successivo e viene attivato il pistone. Quando poi il sensore legge che il pistone è in posizione avanzata, si attiva l’exit action dello stato Movimento_Pistone, la quale incrementa di 1 il numero di pezzi presenti sul nastro (NumPeN), e si passa allo stato successivo. In questo nuovo stato (Ritorno_Pistone) viene attivata l’azione ResetTimer e viene resettata la variabile NReady, per indicare che il nastro non è pronto. Infine, una volta che il sensore legge che il pistone è indietro, si ritorna allo stato dummy.

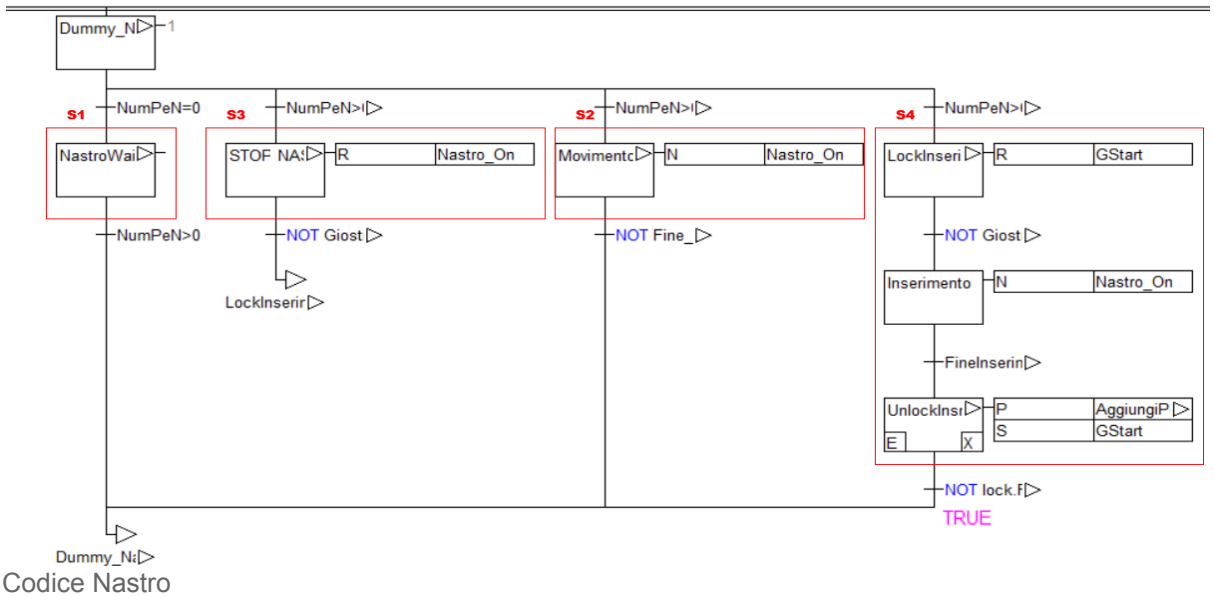


Codice Pistone

Nastro

La **logica di gestione del nastro** può essere riassunta con la seguente tabella, nella quale l'elenco puntato sta a significare una contemporaneità di condizioni e l'elenco numerato una sequenza di azioni.

Stato	Azione	Condizioni di usicta	Stato successivo
S1 • Nessun pezzo presente	Nessuna	Pezzi presenti	Dummy_Nastro
S2 • Pezzi presenti • Nessun pezzo a fine corsa	Nastro acceso	Pezzo arrivato a fine corsa	Dummy_Nastro
S3 • Pezzi presenti • Pezzo a fine corsa • Giostra che gira o pezzo in postazione 0	Nastro spento	• Giostra ferma • Nessun pezzo in postazione 0	S4
S4 • Pezzi presenti • Pezzo a fine corsa • Giostra ferma • Nessun pezzo in postazione 0	1. Giostra bloccata 2. Nastro acceso per un certo lasso di tempo 3. Giostra sbloccata 4. Modifica array Giostra	Azioni completate	Dummy_Nastro

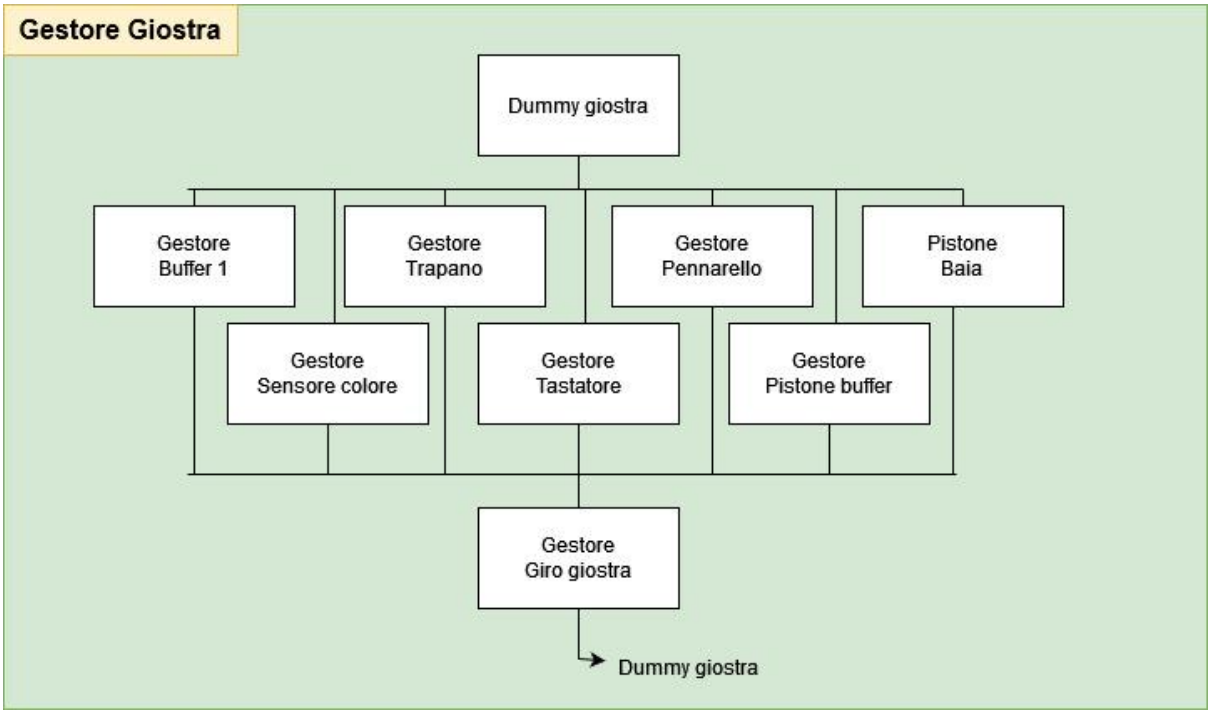


Codice Nastro

Giostra

Applicando lo stesso principio di gestione indipendente utilizzato per controllare pistone, nastro e giostra separatamente, abbiamo suddiviso logicamente a sua volta la giostra in gestori di sottocomponenti indipendenti. La sequenza di azioni svolte dal Gestore Giostra è la seguente:

- vengono attivati in parallelo i gestori delle singole postazioni;
- Quando tutte le postazioni sono arrivate nello stato finale e hanno impostato il posto corrispondente a ready viene attivato il gestore del giro della giostra;
- il Gestire Giro GIostra valuta determinate condizioni ed eventualmente gira la giostra e infine riattiva nuovamente in parallelo tutti i gestori delle postazioni.



Schema della logica della giostra

Ogni postazione ha la seguente struttura:

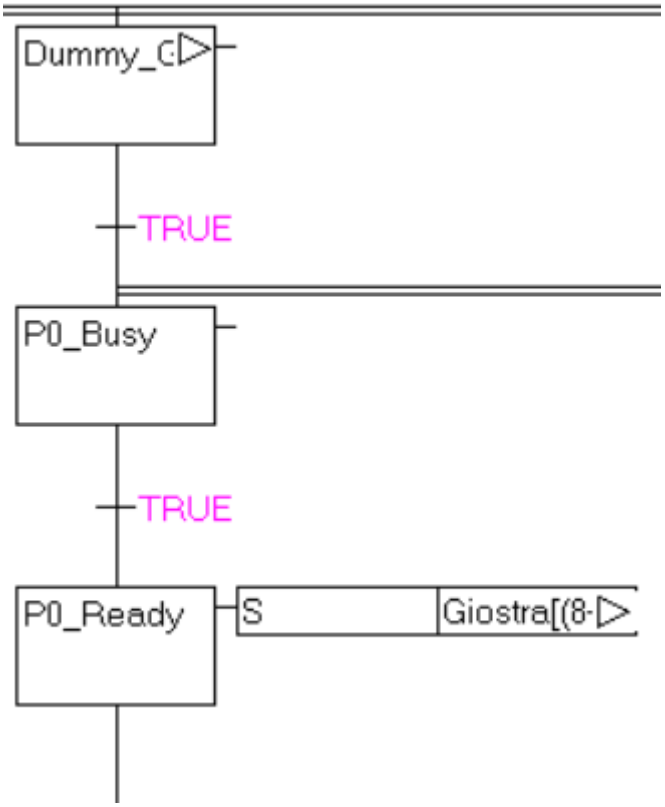
Schema logico postazioni

- Lo stato “Postazione_Busy” indica l’ingresso in una postazione
- “Logica di gestione della postazione” (opzionale) è un sottosistema in cui viene gestita la postazione stessa
- Lo stato “Postazione_Ready” indica che la postazione ha finito di svolgere il suo compito

Le condizioni di non lavorazione per le postazioni da 2 a 5 (sensore colore, trapano, tastatore e pennarello) sono la non presenza del pezzo o la fine lavorazione del pezzo.

Postazione 1: Ingresso

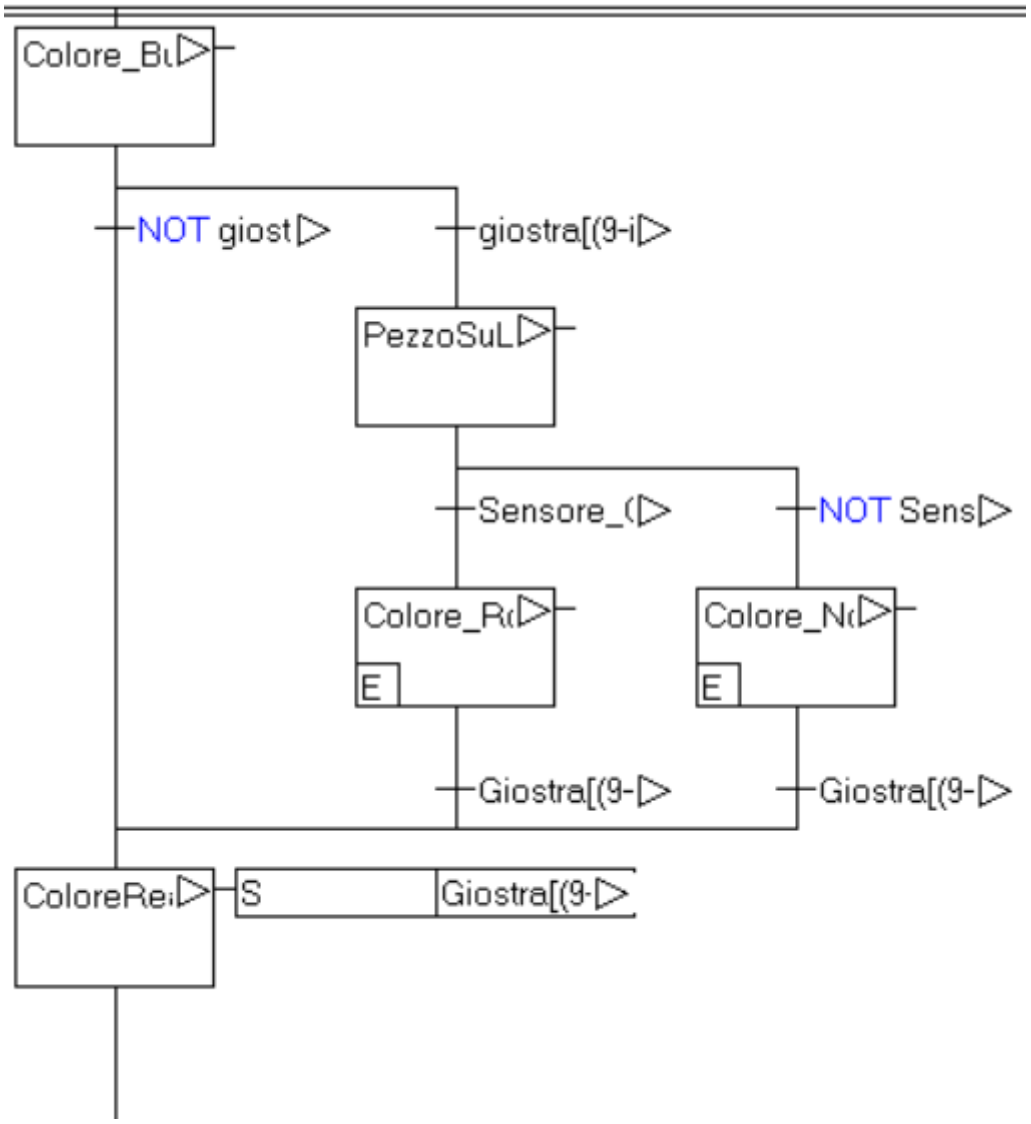
La prima postazione è l’ingresso della giostra; in questa postazione non si effettuano lavorazioni quindi l’unica azione che viene fatta è settare la variabile relativa alla postazione in modo da consentire il giro della giostra.



Codice ingresso giostra

Postazione 2: Sensore colore

La **logica di gestione del sensore colore** semplicemente imposta il colore del pezzo corrispondente a rosso se il valore del sensore è alto, e a non rosso altrimenti.

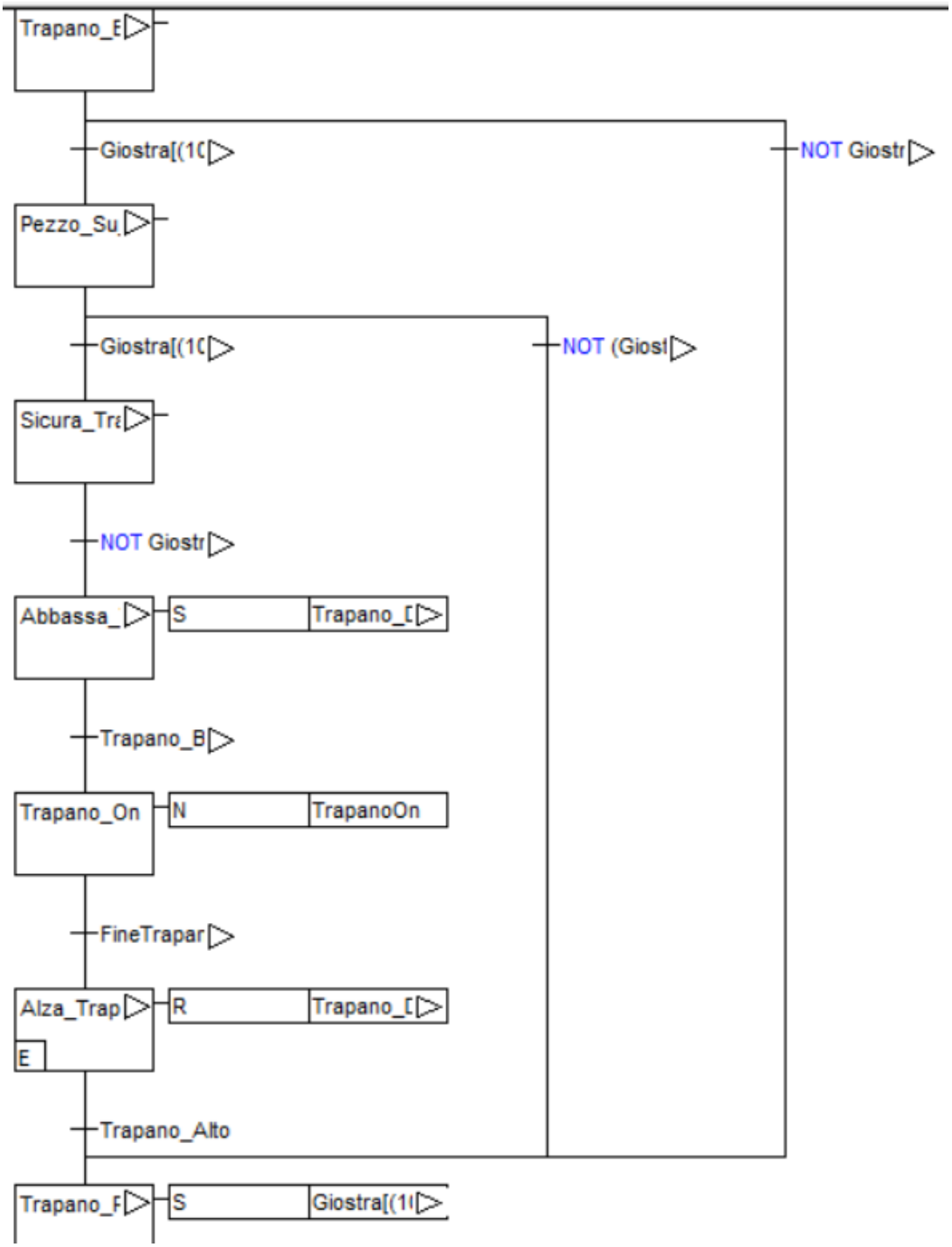


Codice sensore colore

Postazione 3: Trapano

La **logica di gestione del trapano**:

- verifica se il pezzo presente è rosso
- se il pezzo è rosso, si abbassa e lo trapana
- se il pezzo è non rosso, non lo trapana
- infine setta la variabile relativa alla postazione in modo da consentire il movimento della giostra

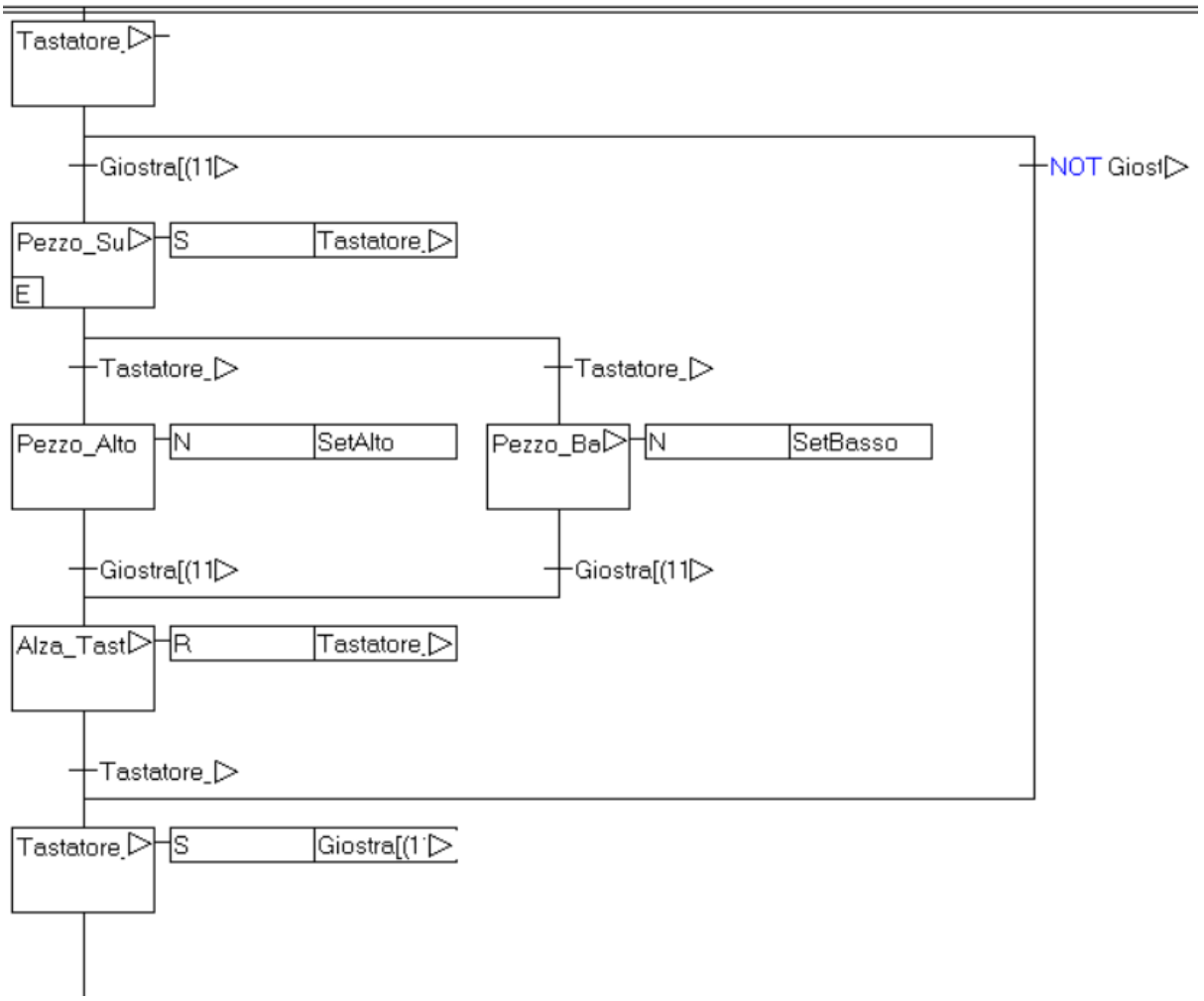


Codice trapano

Postazione 4: Tastatore

La **logica di gestione del tastatore**:

- abbassa il tastatore
- quando il tastatore è arrivato a fine corsa legge il valore di uscita del tastatore
    - Se il valore del tastatore è maggiore del valore soglia allora il pezzo viene registrato come alto
    - Altrimenti il pezzo viene impostato come basso
- quando il valore dell'altezza del pezzo è stato impostato viene rilasciata la variabile che abbassa il tastatore
- quando il tastatore si è alzato la variabile relativa alla postazione passa a ready

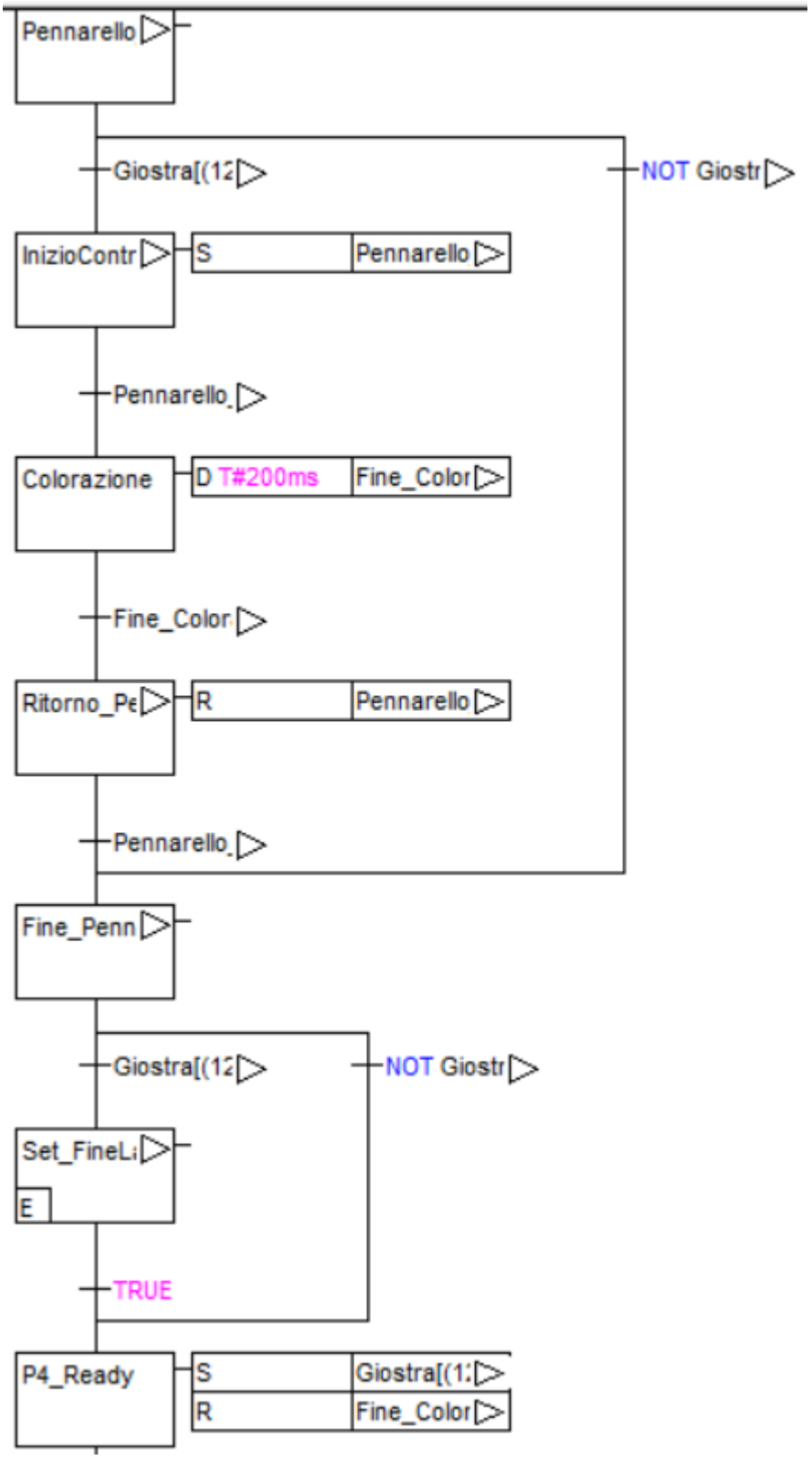


Codice tastatore

Postazione 5: Pennarello

La **logica di gestione del pennarello**:

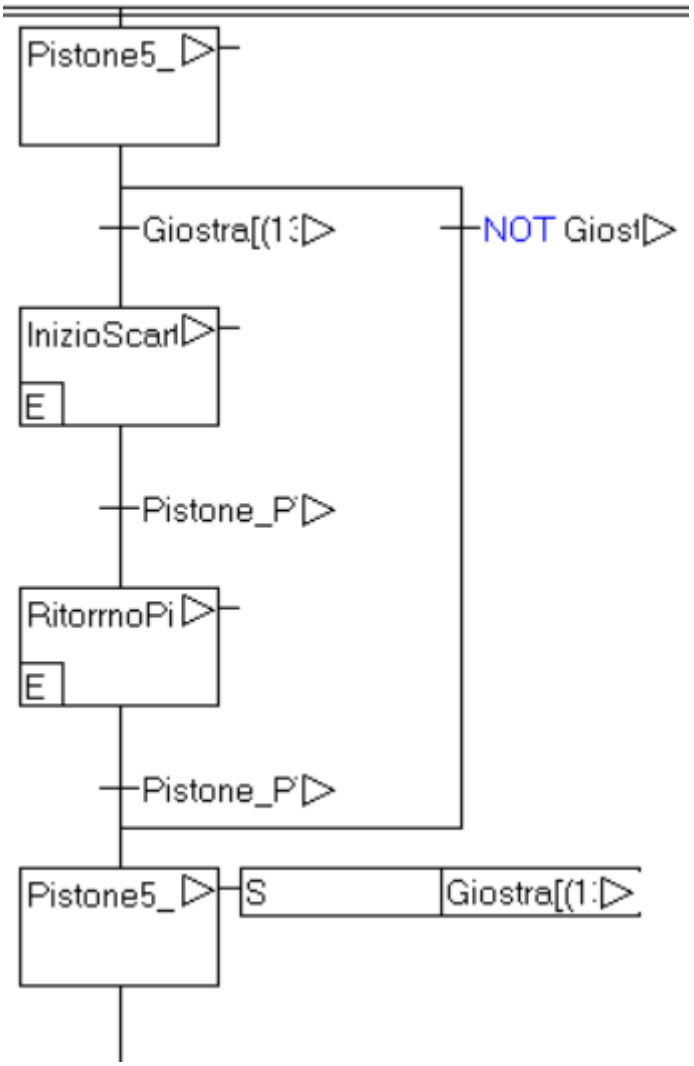
- verifica se il pezzo presente è basso
- se il pezzo è basso, si abbassa e lo colora
- se il pezzo non è basso, non lo colora
- infine setta la variabile relativa alla postazione in modo da consentire il movimento della giostra



Codice pennarello

Postazione 6: Buffer di scarto

La **condizione di non lavorazione** è se non c'è nessun pezzo o se il colore del pezzo è rosso.
La **logica di gestione del pistone di scarto** è molto simile alla gestione del pistone che immette i pezzi nel nastro, ma senza condizioni di attesa per la spinta e con la differenza che in uscita deve aggiornare la variabile giostra per registrare l'espulsione del pezzo.

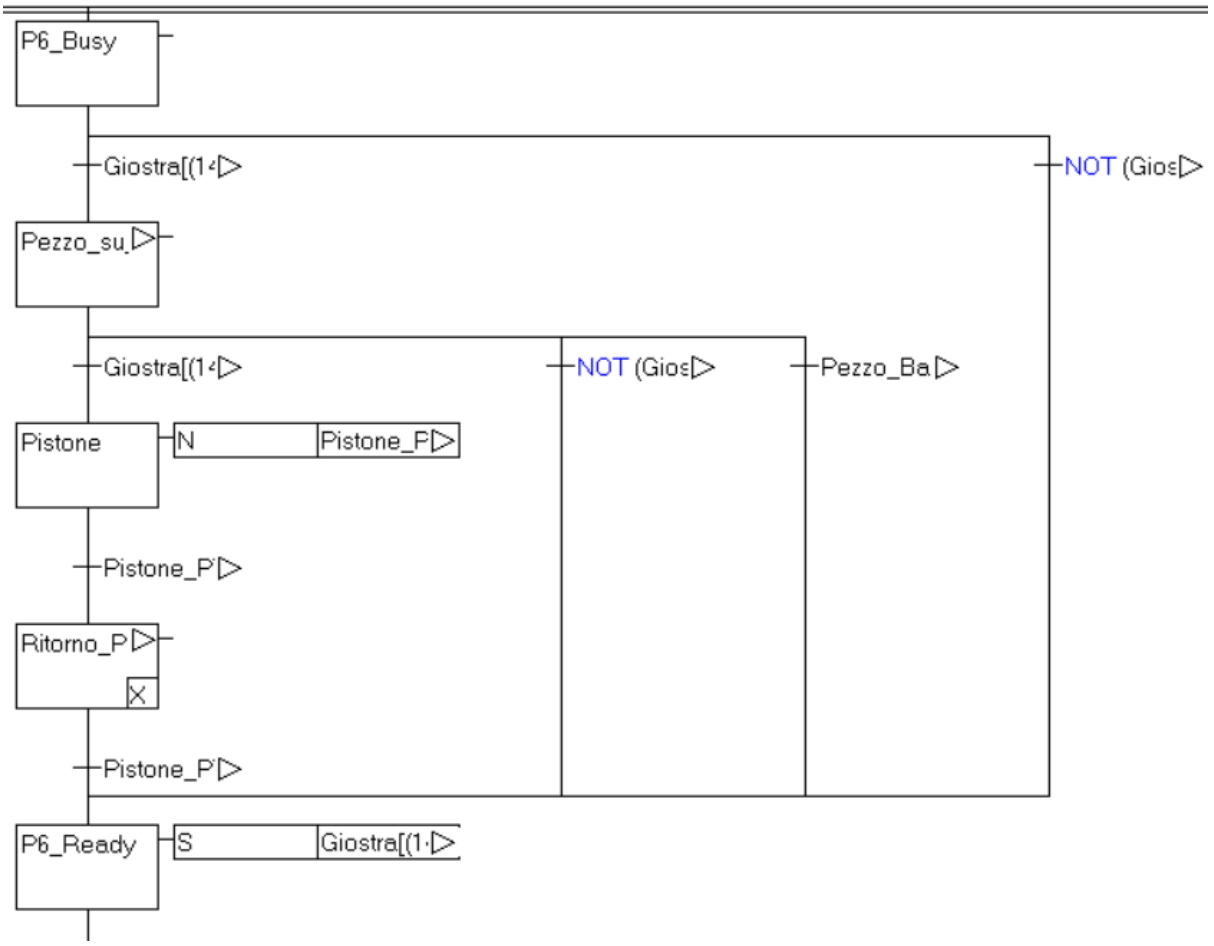


Codice pistone buffer

Postazione 7: Baia

La **condizione di non lavorazione** per questa postazione è se non è presente il pezzo o se è già presente il pezzo nella baia o se il pezzo non è di colore rosso.

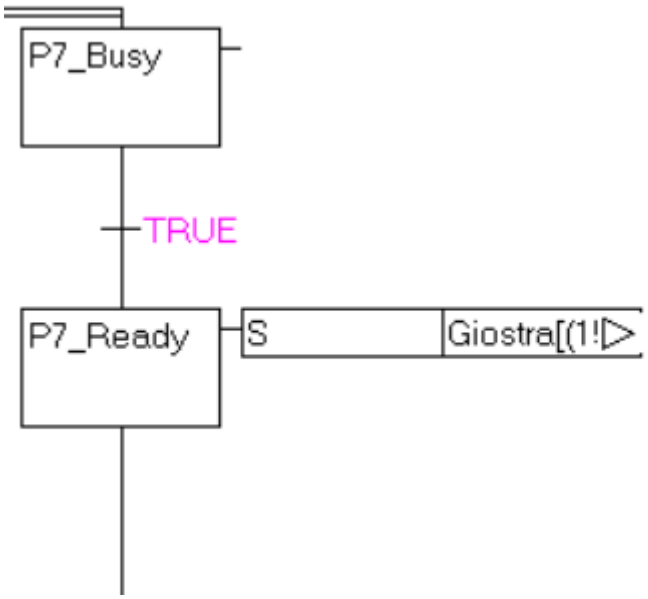
La **logica di gestione del pistone su baia** è molto simile alla gestione del pistone di scarto ma con la differenza che nel pistone precedente il buffer di scarto è stato considerato illimitato mentre qui il buffer (cioè la baia) ha capacità di un pezzo. Questo vuol dire che se è già presente un pezzo sulla baia, il pistone non potrà far uscire il pezzo dalla giostra finché un operatore non avrà tolto il pezzo dalla baia.



Codice pistone baia

Postazione 8: Buffer

La gestione della postazione 8 è uguale alla gestione della postazione 1.



Codice buffer giostra

Giro giostra

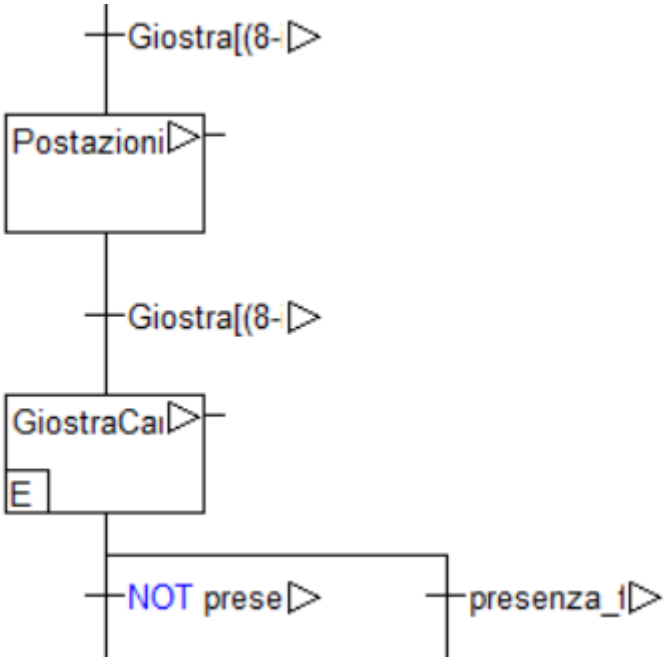
La **logica del giro della giostra** si basa su un primo fondamentale controllo per verificare se tutte le postazioni della giostra sono pronte, ovvero se hanno finito le rispettive lavorazioni.

Abbiamo poi creato due variabili di supporto, TEMP e presenza_fine_lav:

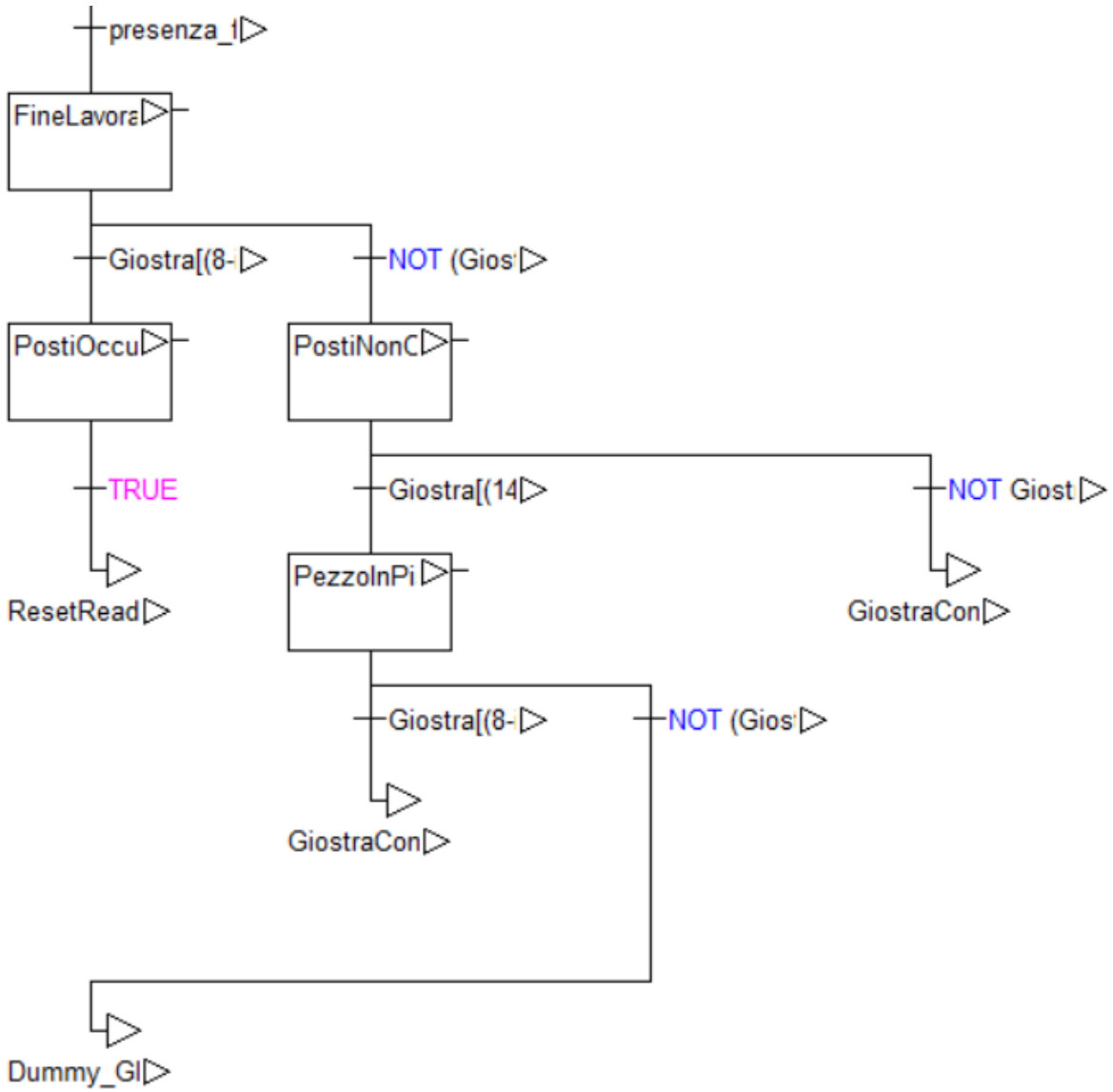
- **TEMP** è un array di booleani che rappresenta le 8 postazioni della giostra; ogni elemento di questo array è settato a TRUE se in quella postazione non è presente un pezzo o se il pezzo presente ha finito la lavorazione, a FALSE altrimenti;
- **presenza_fine_lav** è una variabile booleana che viene settata a TRUE quando tutti i TEMP (da 0 a 7) sono TRUE, ovvero quando tutti i pezzi presenti nella giostra hanno finito la lavorazione o quando non ci sono pezzi nella giostra, a FALSE altrimenti.

Viene poi fatto il controllo su presenza_fine_lav per distinguere due casi:

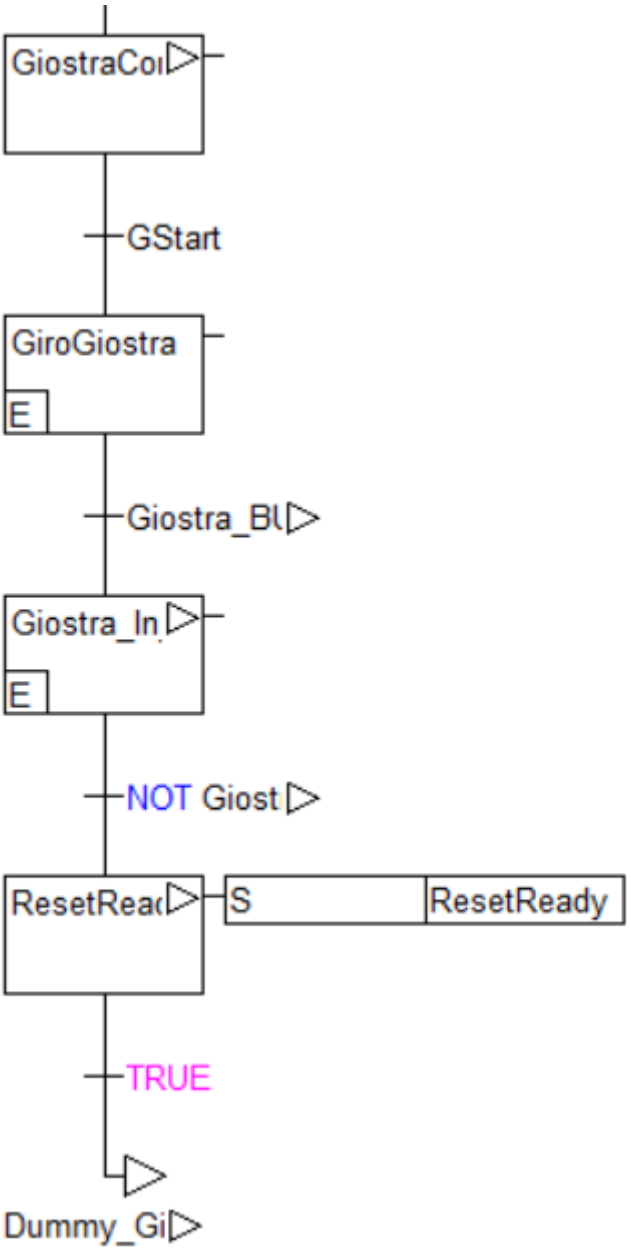
- se presenza_fine_lav è FALSE e la giostra è autorizzata a girare (GStart) la giostra gira, viene aggiornato l'indice che conta i giri della giostra e vengono settate a FALSE tutte le postazioni della giostra;
- se presenza_fine_lav è TRUE si differenziano altri due casi:
    1. se i posti della giostra sono tutti occupati, si resettano tutte le postazioni della giostra e si ritorna a dummy senza girare;
    2. se non tutti i posti della giostra sono occupati, se c'è un pezzo nel pistone della baia e non c'è un pezzo nel posto d'ingresso e il nastro non sta inserendo un pezzo allora si fa girare la giostra, altrimenti si torna a dummy.



Codice giro giostra (1)



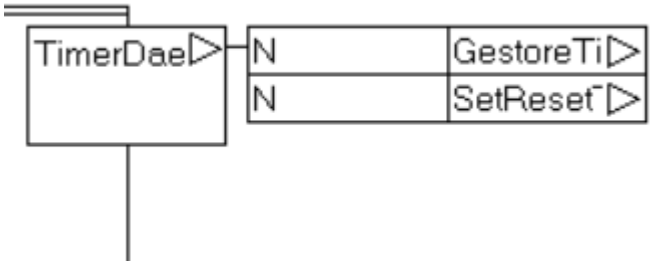
Codice giro giostra (2)



Codice giro giostra (3)

Timer

La gestione dei timer avviene nello stato TimerDeamon, che è sempre attivo e nel quale vengono chiamate due funzioni: GestoreTimers e SetResetTimer.



Codice Timer Deamon

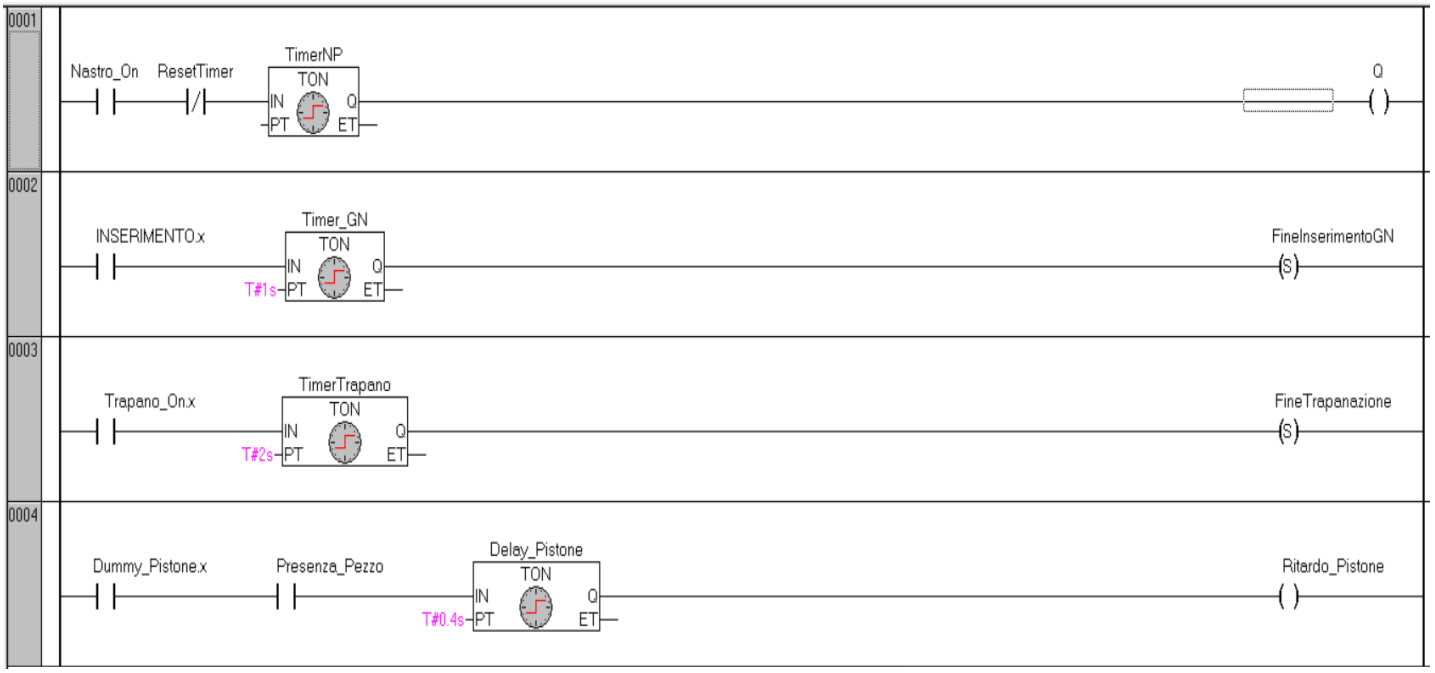
Gestore Timers

Questa funzione si occupa di settare e resettare tutti i timer di interesse per l'applicazione:

- **TimerNP** è utilizzato per conoscere da quanto tempo un pezzo è stato trasportato dal nastro dopo che è stato spinto dal pistone così che un nuovo pezzo non venga spinto immediatamente.
    - Esso è attivato quando il nastro è attivo e la variabile di controllo reset timer (gestita inSet Reset Timer) è falsa.
- **Timer_GN** è utiizzato per per impostare a vero una variabile che rappresenta il fine inserimento quando il nastro sta inserendo un pezzo per più di 1 secondo.
- **TimerTrapano** è utilizzato per per impostare a vero una variabile che rappresenta la fine della lavorazione del trapano quando quest'ultimo è attivo da 2 secondi.
- **Delay_Pistone** è utilizzato per impostare a vero una variabile che indica che c'è un pezzo davanti al pistone e il pistone è fermo da 0.4 secondi.

I timer, come accennato precedentemente, sono utili in quanto, trascorso un certo tempo sotto determinate condizioni, possiamo assumere che determinati eventi non osservabili si siano verificati.
La seguente tabella mette in relazione i timer con i rispettivi eventi non osservabili.

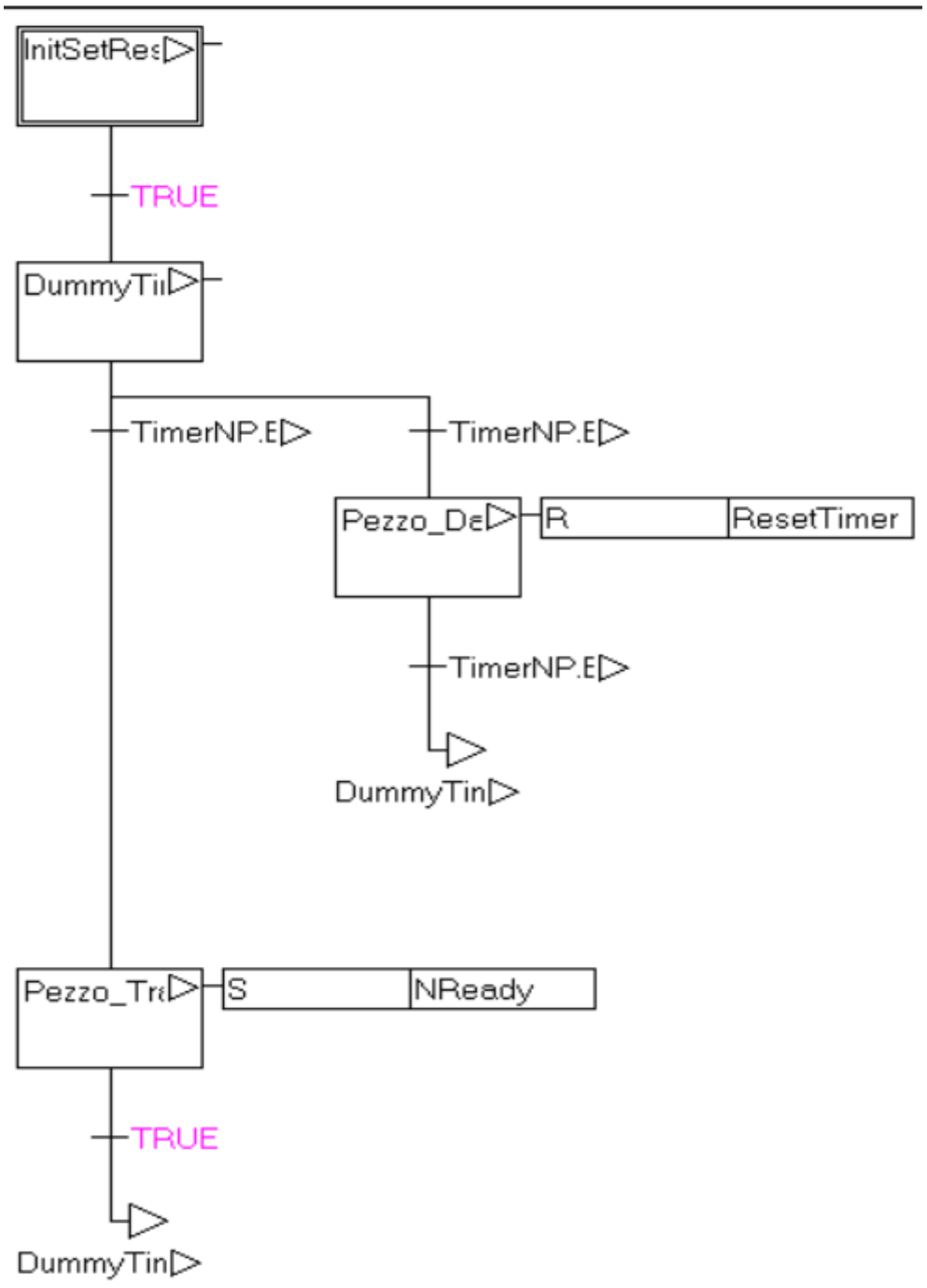
Timer	Evento non osservabile
TimerNP	Il pezzo non è più nella sezione del nastro davanti al pistone
TimerGN	Il pezzo che si trovava davanti al sensore alla fine del nastro è stato inserito nella giostra
TimerTrapano	Il trapano ha finito la lavorazione
Delay_Pistone	Il pezzo sul sensore del pistone è correttamente posizionato



Codice gestore timers

SetResetTimer

Questa funzione SFC gestisce le variabili ResetTimer e NReady.
NReady viene impostato a TRUE (quindi il nastro è pronto a ricevere nuovi pezzi) quando non ci sono pezzi nel nastro oppure quando l'ultimo pezzo inserito è stato trasportato dal nastro per almeno 0.6 secondi (corrispondente ad AttesaNP):
Viceversa ResetTimer viene messo a FALSE quando ci sono pezzi nel nastro e l'ultimo pezzo inserito non è stato trasportato dal nastro per almeno 0.6 secondi.



Codice Set Reset Timer

Risultati

Dalle prove eseguite risulta che gli obiettivi funzionali vengono soddisfatti.
Anche gli obiettivi qualitativi sono soddisfatti, infatti:

- **Sicurezza**: non si presentano situazioni in cui gli attuatori sono attivi quando non dovrebbero, perciò nessun componente rischia di rompersi.
- **Efficienza**: non ci sono tempi morti durante l'esecuzione.
- **Controllabilità**: nonostante ci siano situazioni in cui i pezzi si bloccano meccanicamente nel sistema, anche in questi casi non si verificano blocchi;
    - Ad esempio i pezzi potrebbero incastrarsi nel nastro trasportatore ma sbloccandoli manualmente il sistema continua a svolgere correttamente il suo compito.