UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea triennale Ingegneria Informatica e dell'Automazione

Sistemi di controllo basati su PLC per l'automazione industriale

PLC based control system for industrial automation

Relatore:

Chiar.mo Prof. Gianluca Ippoliti

Correlatore:

Prof. Giuseppe Orlando

Tesi di Laurea di:

Edoardo Palmoni

A.A. 2022 / 2023

Sommario

Introduzione **3**

1 L'automazione dei processi industriali **4**

1.1 Cos'è un processo industriale 4

1.2 L'automazione industriale 4

2 PLC (Programmable Logic Controller) **6**

2.1 Cos'è un PLC 6

2.2 Cenni storici 6

2.3 Struttura PLC 7

3 Linguaggi di programmazione **9**

3.1 Linguaggio Ladder 10

3.2 Linguaggio SFC 13

3.2.1 Reti di Petri 16

3.2.2 Traduzione SFC in Ladder 19

3.3 Linguaggio FBD 20

3.4 Lista di istruzioni 20

3.5 Testo strutturato 22

4 OpenPLC **23**

5 Esempi sviluppati **24**

5.1 Esempio 1 (Trapano Automatico) 24

5.2 Esempio 2 (Distributore Automatico di Bibite) 28

5.3 Esempio 3 (Sistema Semaforico) 32

6 Conclusione **37**

7 Bibliografia e Sitografia **38**

Introduzione

Con questa tesi si vuole approfondire lo studio, la caratterizzazione e l'implementazione dei sistemi di controllo sviluppati mediante il PLC, dispositivo molto utilizzato in ambito industriale. Progresso ed efficienza sono attributi centrali alla base dello sviluppo aziendale e ciò è dettato anche dall'impiego del PLC, riuscendo ad imporre un importante cambiamento di questo settore che è sempre più in evoluzione.

Il motivo che mi ha spinto ad approfondire questa tematica è stata la curiosità emersa a seguito dello studio di un corso a scelta della facoltà frequentante, trovando proprio nell'efficienza del mondo industriale la possibilità non solo di salvaguardare le condizioni di vita dell'uomo senza risparmiarne il miglioramento della qualità di questa, ma anche quella di tutelare il benessere della Natura.

L'elaborato è articolato in cinque capitoli.
Nel primo capitolo vengono introdotti i principi fondamentali dell'automazione industriale dando una generica presentazione dei processi industriali e del campo dell'automazione industriale.
Nel secondo capitolo si descrive il PLC (*Programmable Logic Controller*), si illustra la storia di questo dispositivo e si presenta una struttura standardizzata di un generico controllore, osservandone poi i diversi tipi.
Nel terzo capitolo si mostrano i cinque diversi linguaggi di programmazione per il controllo dei PLC descrivendoli uno ad uno.
Nel quarto capitolo si presenta il software open source OpenPLC con il quale sono state effettuate delle implementazioni e simulazioni riportate negli esempi del quinto capitolo.
Nella parte conclusiva si esprime un riscontro positivo del dispositivo e si fanno delle considerazioni sui sistemi di tipo SCADA.

1 L’automazione dei processi industriali

1.1 Cos’è un processo industriale

Un processo industriale è una sequenza di attività o operazioni coordinate che trasformano materie prime, energia o informazioni in prodotti finiti o semilavorati. Un’industria è un sistema complesso formato da più processi che cooperano tra di loro. Col passare del tempo il concetto di industria manifatturiera ha cambiato stampo, quello che prima era il cosiddetto lavoro a catena di montaggio, dove gli operatori si ritrovavano ad effettuare lavori ciclici, si sta trasformando in un vero e proprio sistema automatizzato dando ampio spazio alla tecnologia, e dove il lavoro dell’uomo rimane fondamentale per la sorveglianza e la manutenzione.

> “Un sistema automatizzato è un qualcosa che è capace di compiere un certo servizio, prendendo autonomamente le decisioni atte a realizzare quanto è stato preventivamente predisposto, in funzione di determinati input provenienti dalla parte controllata; tale capacità è frutto di una serie di azioni combinate e coordinate degli elementi che compongono il sistema. Un’automazione può essere strutturata con differenti livelli di complessità in modo da controllare una sola macchina, l’insieme delle macchine di un reparto, fino ad arrivare alla fabbrica automatica, dove l’intervento dell’uomo è richiesto in misura molto marginale soltanto per la supervisione del processo.” [1]

Dunque un sistema di produzione è un insieme di macchine, elementi di trasporto e di immagazzinamento organizzato per produrre particolari prodotti.
Del sistema ne farà sicuramente parte il ruolo dell’uomo, volto alla:

- progettazione del sistema stesso,
- manutenzione correttiva (riparare dispositivi che si guastano),
- manutenzione preventiva (mantenere l’impianto in condizioni ottimali di funzionamento),
- riconoscimento nella gestione degli allarmi.

1.2 L’automazione industriale

Per automazione industriale si intende un campo dell'ingegneria e della tecnologia che studia le metodologie e le tecnologie che permettono il controllo dei flussi di energia, di materiali e di informazioni necessari alla realizzazione di processi produttivi.
Sicuramente tra i principali vantaggi possiamo dire che l’automazione industriale contribuisce a:

- migliorare l’efficienza, la precisione e la sicurezza delle operazioni industriali,
- ridurre i sovraccarichi di produzione e di lavoro,
- minimizzare i fermi macchine e la congestione dati IT,
- aumentare l'affidabilità aziendale,
- aumentare la produttività e riduzione dei tempi di produzione,
- ridurre i costi operativi,

[1] Venturi, E., & Ortolani, G. (2018). *Manuale di Elettrotecnica, Elettronica e Automazione.* Milano: Hoepli.(XIV-3)

- migliorare la gestione dei magazzini,
- avere un'ottima flessibilità,
- possibilità di ridurre l'impatto ambientale e di risparmiare energia.[2]

L'automazione industriale impiega ottimamente l'uso di sistemi di controllo, come i PLC (Programmable Logic Controller) e i computer, per gestire processi industriali e ridurre al minimo l'intervento umano.

[2] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

2 PLC (Programmable Logic Controller)

2.1 Cos'è un PLC

Un Programmable Logic Controller (PLC) è un dispositivo elettronico programmabile utilizzato per controllare macchinari e processi industriali. I PLC sono ampiamente impiegati nell'automazione industriale per gestire e monitorare operazioni complesse e sequenze di produzione. Lo sviluppo del controllo di macchine e processi ha avuto una considerevole notorietà con l'avvento del microprocessore, con il quale un unico dispositivo era in grado di gestire una vasta gamma di applicazioni e poterle personalizzare con precisione in maniera che potesse soddisfare i bisogni del richiedente. Inizialmente il controllo dei segnali era condotto diversamente in base al tipo, analogico o digitale. Per il controllo di grandezze fisiche di tipo analogico (temperatura, pressione, umidità ecc.) si implementavano sistemi DCS (*Distribuited Control System*), mentre per valori booleani si optava più per dispositivi chiamati PLC in quanto venivano elaborati dati solamente su singoli bit. Con lo sviluppo delle reti in aggiunta alla vasta produzione e varietà, i PLC combinati all'uso del computer hanno di gran lunga sostituito i sistemi DCS. Una definizione più completa è data dallo standard IEC 1131 sui PLC sviluppata dal Working Group 7 dell'IEC Technical Commitee TC65, che lo definisce come un:

> «*sistema elettronico a funzionamento digitale, destinato all'uso in ambito industriale, che utilizza una memoria programmabile per l'archiviazione interna di istruzioni orientate all'utilizzatore per l'implementazione di funzioni specifiche, come quelle logiche, di sequenziamento, di temporizzazione, di conteggio e di calcolo aritmetico, e per controllare, mediante ingressi ed uscite sia digitali che analogici, vari tipi di macchine e processi*».

Il PLC è un componente che connette l'elettromeccanica all'elettronica e che ha di gran lunga preso il posto della logica cablata in quanto elabora, diagnostica e interviene in modo più evolutivo. La logica programmabile fornisce una connessione tra dispositivi permettendone delle modifiche e rettifiche che la logica cablata non riuscirebbe a dare con tale facilità. In ambito industriale il PLC viene utilizzato per funzionare in ambienti con presenza di disturbi elettrici, con alte temperature, con vibrazioni, e a diverse condizioni quindi che altri dispositivi non riuscirebbero a resistere. [3]

2.2 Cenni storici

Prima dell'avvento dei PLC, la logica di funzionamento nei macchinari elettromeccanici era gestita dai relè, apparecchi tutt'ora utilizzati, che presentavano l'inconveniente per cui dopo molte ore di lavoro si rompevano, inoltre la loro applicazione comportava anche una lunga serie di cablaggi. Nel 1960, l'ingegnere meccanico Richard Dick Morley, avviava una piccola azienda chiamata Bedford Associates lavorando con fabbriche che utilizzavano macchine utensili, alla ricerca di miglioramenti nei processi produttivi. Nel 1968 Morley cominciò a guardarsi intorno e scrisse quelle che furono le prime specifiche per un apparecchio PLC. Per programmare il PLC voleva un linguaggio proprietario e

[3] Venturi, E., & Ortolani, G. (2018). *Manuale di Elettrotecnica, Elettronica e Automazione.* Milano: Hoepli.

naturalmente tutti i diritti su questo linguaggio, il passo successivo fu l'invenzione del linguaggio a logica ladder (LD). Morley si rivolse al suo team e cominciarono a lavorare alla creazione di questo controllore logico programmabile. Il primo PLC aveva un contenitore metallico che assicurava una buona dissipazione di calore e lasciava entrare l'aria al suo interno. Il suo nome fu "modello 084", dato che era l'ottantaquattresimo progetto della Bedford Associates. Gli svantaggi principali del "modello 084" erano lentezza, un po' "ingombrante" e mancava di alcune funzionalità utili. Successivamente venne creato il nuovo modello chiamato "184", il quale possedeva delle migliorie rispetto al precedente. Nacque una nuova azienda chiamata "Modicon", abbreviativo di "Modular Digital Controller", con la quale iniziò la produzione di massa dei PLC. Mentre i primi PLC venivano costruiti, la General Motors Hydramatic Transmission Factory era in cerca di qualcosa per ammodernare le proprie fabbriche, e all'avvenuta conoscenza del progetto della Modicon, fecero un ordine da un milione di dollari per acquistare da essa una serie di unità. Dopo General Motors, anche General Electric ordinò la sua prima partita di PLC da un milione di dollari.[4]

2.3 Struttura PLC

La configurazione minima di un sistema PLC è composta da:

- Armadio (o cestello o rack)
- Modulo processore
- Moduli di I/O
- Modulo alimentatore
- Terminale di programmazione

L'*armadio* assicura la connessione meccanica e il collegamento elettrico. Si tratta di una struttura metallica, e deve essere connesso elettricamente a terra sia per ragioni di sicurezza, sia per schermare meglio i moduli alloggiati.

Il *modulo processore* è costituito essenzialmente da una scheda con uno o più microprocessori. Viene supportato dalla memoria eseguendo i programmi del sistema operativo e quelli sviluppati dall'utente, conservati nella memoria stessa. Controlla e supervisiona tutte le operazioni eseguite all'interno del sistema, attraverso l'esecuzione delle istruzioni contenute nella memoria.

I *moduli I/O* (input/output), sia digitali che analogici, sono delle schede che permettono l'interfacciamento tra la microelettronica del PLC e il mondo esterno. Gli input possono provenire da sensori e interruttori, mentre gli output vanno verso attuatori e dispositivi di controllo. Sono i moduli attraverso cui il PLC comunica con il processo fisico, rilevando eventi e dati dai sensori e comandando azioni agli attuatori.

Il *modulo alimentatore* è una scheda che alimenta tutti gli altri moduli presenti nell'armadio. Viene connesso alla rete di alimentazione elettrica e riesce a fornire una o più tensioni stabilizzate con un massimo di corrente erogabile. Sarà quindi composto da

[4] sito web https://automazione-plc.it/

un trasformatore, un circuito rettificatore, un filtro, un circuito stabilizzatore e da un circuito per la protezione da sovracorrenti o cortocircuiti.

Il *terminale di programmazione* può essere un dispositivo particolare o un semplice personal computer e serve per la programmazione del PLC che non ha, usualmente, dispositivi di interfaccia con l'uomo. Il terminale di programmazione viene utilizzato tramite una porta seriale e/o una rete informatica.[5]

Ci sono diverse tipologie di PLC, potendo quindi soddisfare differenti esigenze in base all'applicazione che ne si deve fare e alle richieste economiche. In linea generale si hanno tre tipi di strutture:

- PLC compatti: presentano una struttura di dimensioni ridotte e comprendono poche connessioni di I/O dando una disponibilità di massimo 10 punti non espandibili. Sono controllori a basso costo, il loro impiego si ha in applicazioni a livello civile e a livello industriale si limitano al comando di semplici macchine.
- PLC espandibili: sono apparecchiature più sviluppate rispetto ai PLC compatti. In genere la configurazione prevede il modulo base equipaggiato con CPU, alimentatore e dei moduli aggiuntivi di espansione di Input/Output fino ad un massimo di circa cento punti.
- PLC modulari: la loro struttura è simile a quella dei PLC espandibili ma con notevoli possibilità di espansione. Presentano più moduli alloggiati in un unico armadio ed ha la disponibilità di connettere fino a migliaia di I/O. Sono le apparecchiature più costose, ma che garantiscono delle prestazioni superiori. Le applicazioni sono di tipo industriale, per le quali, data la flessibilità di questa struttura, è possibile configurare architetture aggiungendo moduli in base alle necessità.[6]

[5] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

[6] Venturi, E., & Ortolani, G. (2018). *Manuale di Elettrotecnica, Elettronica e Automazione.* Milano: Hoepli.

3 Linguaggi di Programmazione

Per eseguire una progettazione di un sistema automatico bisogna seguire determinati passi formando una sorta di algoritmo della programmazione:

- determinare il sistema che si deve gestire ed i rispettivi processi,
- determinare il numero di I/O necessari,
- determinare le caratteristiche che il PLC deve avere,
- effettuare l'indirizzamento degli I/O,
- avere ben chiaro e predisporre mediante uno schema le operazioni che il sistema deve svolgere, anche le situazioni di allarme,
- implementare lo schema precedente in un linguaggio di programmazione editabile,
- inserire il programma nel PLC,
- ricercare eventuali errori di programmazione a livello sintattico o esecutivo,
- generare una simulazione del programma effettuato,
- collegare il PLC all'impianto da controllore e verificarne il corretto funzionamento.

Elementi fondamentali nella programmazione dei PLC sono le variabili, che sono il mezzo per identificare tutti gli oggetti del programma stesso (quindi ingressi, uscite e marker). Nella dichiarazione di una variabile si devono rispettare dei canoni di definizione:

- inserimento del segno percentuale (%),
- inserimento del prefisso di locazione (I, Q, M),
- inserimento del prefisso di taglia (B, W, D, L, niente),
- inserimento di uno o più numeri interi senza segno, separati da un punto (.).

Gli inserimenti vanno eseguiti rispettando la tabella sottostante secondo IEC 1131 (CEI 65-40):

Prefisso	Descrizione
I	Locazione di ingresso
Q	Locazione di uscita
M	Locazione di memoria
X	Taglia di un solo bit
nessuno	Taglia di un solo bit
B	Taglia di un byte (8 bit)
W	Taglia di una parola (16 bit)
D	Taglia di una parola doppia (32 bit)
L	Taglia di una parola lunga (64 bit)

Per esempio, la variabile %IW1.6.4.1 indica che è una parola della zona degli ingressi (I), della taglia di una parola di 16 bit (W), collocata nel primo canale di ingressi (.1), del sesto modulo (.6), del quarto rack (.4) del primo bus di I/O (.1). [7]

[7] Venturi, E., & Ortolani, G. (2018). *Manuale di Elettrotecnica, Elettronica e Automazione.* Milano: Hoepli.

I linguaggi di programmazione utilizzati nella logica programmabile sono molteplici, proponendo un ampio ventaglio di scelta agli operatori per soddisfare le loro necessità. Lo standard IEC 1131 prevede cinque linguaggi di programmazione per i PLC:

- Linguaggio a contatti (LD o Ladder)
- Diagramma funzionale sequenziale (SFC o Sequential Funcional Chart)
- Diagramma a blocchi funzionali (FBD o Function Block Diagram)
- Lista di istruzioni (IL o Instruction List)
- Testo strutturato (ST o Structured Text)

3.1 Linguaggio Ladder

Il linguaggio Ladder (o a contatti) è un linguaggio di tipo grafico ed è stato il primo utilizzato per la programmazione dei PLC in quanto, come detto precedentemente, riusciva a sostituire comodamente i quadri a logica cablata con tecnologia a relè.
Nel linguaggio ladder il programma viene:

- scritto tra due linee verticali, binari di potenza, dette «power rail»;
- articolato in linee orizzontali chiamate «rung», dove ogni rung può contenere contatti, bobine, blocchi funzionali e funzioni;
- ciascun rung deve essere necessariamente connesso al power rail di sinistra, mentre la connessione con il power rail di destra è opzionale;
- i contatti in un rung rappresentano le condizioni logiche da valutare per poter determinare lo stato che deve assumere l'uscita, rappresentata dalla bobina;
- nella lettura del linguaggio, il flusso di energia può andare sempre e solo da sinistra verso destra, senza possibilità di inversione;
- i rung vengono letti dal PLC dall'alto verso il basso, e arrivati all'ultimo si ricomincia dal primo.

Nella programmazione bisogna associare i contatti e le bobine a bit di memoria oppure a nomi simbolici. I bit di memoria possono essere delle variabili interne (marker) o degli indicatori di ingresso o uscita fisici. Lo stato di una bobina può rappresentare anche lo stato di un contatto se questi vengono associati allo stesso bit. La struttura del linguaggio a contatti si basa su alcuni semplici simboli.

I contatti sono in genere associati a dispositivi di ingresso fisicamente connessi al PLC (es. pulsanti, interruttori, finecorsa, sensori di prossimità ecc), ma possono anche essere riferiti ad altri elementi come bobine di uscita, relè interni, locazioni di memoria, blocchi funzione (contatori, temporizzatori ecc.).

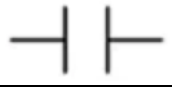
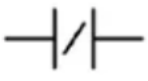
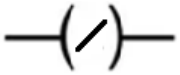
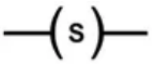
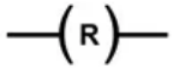
Le bobine, invece, sono in genere associate per indicare le uscite fisiche del PLC, determinando lo stato del dispositivo di uscita corrispondente attraverso il quale viene comandato il relativo attuatore. Possono anche dare un riferimento ai marker (relè o aree di memoria interne al PLC), avendo la funzione di memorizzare il risultato dell'elaborazione di un determinato rung di programma che verrà utilizzata successivamente.

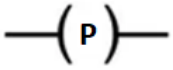
I contatti possono essere di due tipi:

- Contatto Normalmente Aperto: se il bit a esso associato vale 1 (ON) il processo chiuderà il contatto assicurando la continuità logica nel rung, dunque la corrente circola quando il contatto viene azionato. Se il bit vale 0 (OFF) il contatto rimarrà aperto.
- Contatto Normalmente Chiuso: è esattamente il duale del contatto normalmente aperto. Se il bit a esso associato vale 0 (OFF) il processo chiuderà il contatto assicurando la continuità logica nel rung, dunque la corrente circola con il contatto in posizione di riposo. Se il bit vale 1 (ON) il contatto sarà considerato aperto.

Abbiamo inoltre diversi tipi di bobine:

- Bobina (o relè) di uscita: tale istruzione serve per controllare lo stato del bit a essa associato rappresentando un'uscita fisica o un marker interno. Deve essere sempre inserita alla destra del rung e sono il modo più comune con il quale lo si termina. Se le condizioni logiche alla sua sinistra sono verificate il suo stato viene portato a 1 (ON), altrimenti è posto a 0 (OFF).
- Bobina negata: è esattamente il duale della bobina di uscita. Se le condizioni logiche alla sua sinistra sono verificate il suo stato viene portato a 0 (OFF), altrimenti è posto a 1 (ON).
- Bobina di eccitazione (o Latch Coil): la bobina è eccitata nel momento in cui i contatti ai quali è collegata sono chiusi e resta eccitata anche se successivamente i contatti non sono più chiusi (modalità SET). Dunque mantiene lo stato logico 1 (ON) anche quando le condizioni di attivazione cessano. L'unico modo per portare a 0 (OFF) lo stato è quello di usare un'istruzione di RESET.
- Bobina di diseccitazione (o Unlatch Coil): è esattamente il duale della bobina di eccitazione. La bobina si diseccita quando i contatti ai quali è collegata sono chiusi. Resta diseccitata anche se successivamente i contatti non sono più chiusi (modalità RESET). Dunque riporta allo stato logico 0 (OFF) una uscita.
- Bobina a fronte di salita: il bit associato è 1 quando la bobina passa da non alimentata ad alimentata. Resta a 1 per l'intervallo tra due esecuzione consecutive.

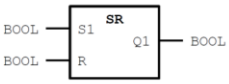
Contatti	Contatto Normalmente Aperto	
	Contatto Normalmente Chiuso	
Bobine	Bobina di uscita	
	Bobina negata	
	Bobina di eccitazione	
	Bobina di diseccitazione	

	Bobina a fronte di salita	

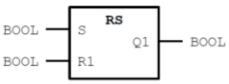
I blocchi funzione sono elementi di programmazione spesso utilizzati nei PLC per organizzare e riutilizzare il codice. I blocchi possono essere predefiniti dal produttore del PLC o definiti dall'utente per eseguire funzioni specifiche. L'utilizzo di blocchi funzione contribuisce a una progettazione del software più modulare (consentono di organizzare il codice in moduli o sotto-programmi indipendenti che svolgono funzioni specifiche), facilitando la comprensione, la manutenzione e il riuso del codice. Si possono creare più copie di blocchi funzionali, ognuno avente un proprio nome identificativo e costituito da variabili interne proprie. Tra i blocchi funzione standard si hanno: i flip-flop, i contatori, i temporizzatori ed i rilevatori di fronte.

- Flip-Flop: sono circuiti sequenziali usati come memoria elementare, dunque l'output non dipende soltanto dallo stato degli input correnti ma anche dallo stato precedente. Nella programmazione ladder si hanno le due configurazioni SR e RS, quindi si può decidere se a dominare è il Set o il Reset.

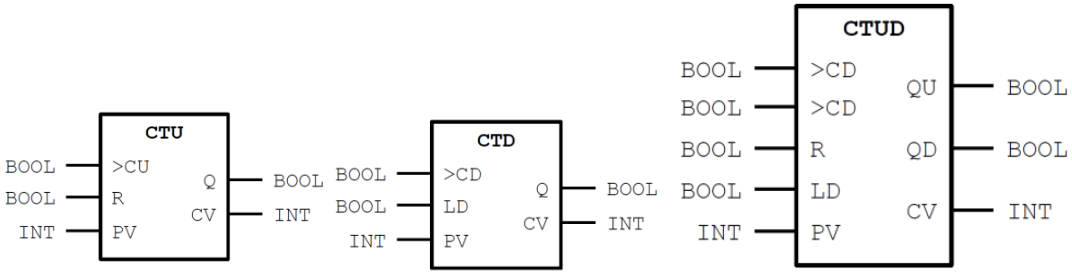
```
FUNCTION_BLOCK SR (* priorità sul SET *)
    VAR_INPUT
        S1,R : BOOL;
    END_VAR
    VAR_OUTPUT
        Q1 : BOOL;
    END_VAR
    Q1 = S1 OR (Q1 AND NOT R)
END_FUNCTION_BLOCK
```




```
FUNCTION_BLOCK RS (* priorità sul RESET *)
    VAR_INPUT
        S,R1 : BOOL;
    END_VAR
    VAR_OUTPUT
        Q1 : BOOL;
    END_VAR
    Q1 = NOT R1 AND (Q1 OR S)
END_FUNCTION_BLOCK
```



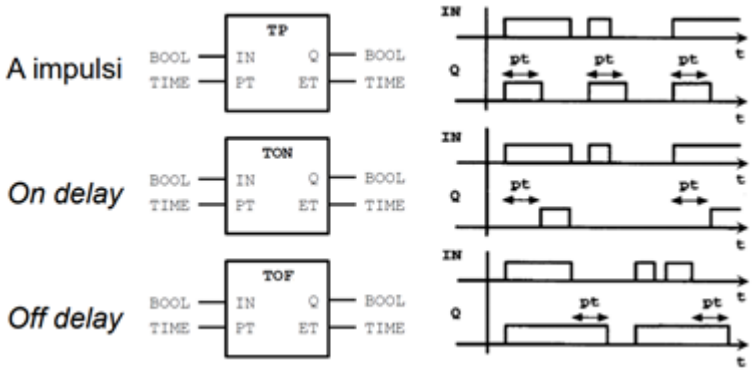
- Contatori: esistono contatori a incremento (CTU), a decremento (CTD) o bidirezionali (CTUD). I segnali da conoscere per la gestione dei contatori sono:
  CU, CD ingressi sui cui fronti di salita si incrementa/decrementa il contatore;
  R è il reset (porta a zero il valore di conteggio);
  PV è il valore di conteggio di ingresso (preset);
  LD carica il contatore con il valore PV;
  Q end of count [CU = PV (incremento); CD = 0 (decremento)];
  QD segnala che il contatore bidirezionale ha raggiunto lo zero;
  QU segnala che il contatore bidirezionale ha raggiunto il valore PV;
  CV è il valore raggiunto dal contatore.
  Per il contatore andrà previsto un valore di conteggio massimo PV. Questo limite può essere aggirato usando più contatori in cascata.



- Temporizzatori: si hanno tre tipi di timer in base all'uso che se ne vuol fare:
  - A impulsi (TP): il timer genera una finestra rettangolare di durata prestabilita quando l'ingresso IN diventa vero;

- On delay (TON): l'uscita diventa vera dopo un certo tempo;
- Off delay (TOF): l'uscita diventa falsa dopo un certo intervallo dallo spegnimento di IN.

Anche in questo caso è possibile aggirare il limite massimo utilizzando più timer in cascata o un contatore in cascata al timer.



- Rilevatori di fronte: possono essere di salita o di discesa.
  - Un rilevatore di fronte di salita (R_TRIG) rileva la transizione di un segnale da un livello basso a un livello alto (da 0 a 1). Questo tipo di rilevatore è spesso utilizzato per catturare l'inizio di un impulso o di un ciclo di clock.
  - Un rilevatore di fronte di discesa (F_TRIG) rileva la transizione di un segnale da un livello alto a un livello basso (da 1 a 0). Questo tipo di rilevatore può essere utilizzato per catturare la fine di un impulso o di un ciclo di clock.

3.2 Linguaggio SFC

Il linguaggio SFC (*Sequential Functional Chart*) è un linguaggio di programmazione grafico utilizzato principalmente nel contesto della programmazione dei PLC. Tale linguaggio è stato progettato per rappresentare il comportamento sequenziale di sistemi ad eventi discreti, cioè di quei sistemi in cui le variazioni dello stato non sono legate al trascorrere del tempo ma all'occorrenza di situazioni particolari. I diagrammi del linguaggio SFC sono strutturati in modo da modellare sequenze di azioni in modo chiaro e intuitivo, e sono composti da passi, transizioni e azioni:

- Passo (o stato): rappresenta le diverse fasi o condizioni del sistema. Il passo può essere definito come una condizione invariante del sistema che può essere modificata solo all'occorrenza di un determinato evento. Ogni passo può essere associato a un'azione o a un insieme di azioni specifiche. Deve essere univocamente contraddistinto da un nome (o da un numero). Per distinguere i passi iniziali si utilizza la doppia cornice.
- Transizione: indica le condizioni che determinano il passaggio da un passo all'altro. Quando una transizione è attivata, il sistema si sposta da uno stato all'altro. La

transizione viene indicata con una barretta e la sigla Tn, dove n è il numero che la identifica.

- ❖ Azione (o arco orientato): rappresenta l'unità fondamentale di lavoro nel diagramma. Le azioni sono associate a uno specifico stato e descrivono l'attività o il compito che deve essere eseguito quando il sistema si trova in quel particolare passo. Hanno il compito di collegare tra di loro le fasi, stabilendone la sequenza. Sono interrotte dalle transizioni, le quali determinano le condizioni da soddisfare per attivare le fasi. L'orientamento degli archi viene indicato con una freccia, ma può essere evitato se l'arco è orientato verso il basso, dunque non abbia condizioni di ambiguità.

Nella composizione dell'SFC si dovrà fare riferimento a delle regole di tipo strutturale e di sviluppo. Due regole importanti per la costruzione di un SFC sono quelle che affermano che:

I. tra due fasi collegate da un arco orientato vi deve essere sempre una transizione;
II. tra due transizioni vi deve essere sempre almeno una fase.

Le regole di evoluzione dell'SFC, invece, affermano che:

1. una transizione si dice *abilitata* se tutte le fasi a monte sono attive. Una transizione si dice *superabile* se è abilitata e la condizione ad essa associata assume il valore "vero";
2. se una transizione è superabile allora viene effettivamente superata: tutte le fasi a monte della transizione vengono disattivate e tutte le fasi a valle vengono attivate.

Se più transizioni diventano superabili nello stesso istante, esse sono tutte superate contemporaneamente.
Nei sistemi di controllo si deve ricorrere molto spesso, per le transizioni di fase, a variabili legate al tempo. Nell'SFC la variabile temporale si indica con la notazione *t/Xn/d* dove:

- *t* identifica la temporizzazione;
- *Xn* identifica il marker di fase la cui attivazione fa partire la temporizzazione;
- *d* è la durata della temporizzazione.

La variabile temporale assume valore 0 all'istante iniziale, e conserva tale valore fino a quando la fase Xn viene attivata. Assume il valore 1 quando è trascorso un tempo d dall'ultima attivazione di Xn, per poi ritornare al valore 0 se la fase si disattiva. [8]
Per quanto riguarda le azioni, come precedentemente descritte, sono associate alle specifiche fasi e descrivono il lavoro che si deve svolgere al passo corrente. Possono esserci più azioni per ogni passo e vengono poste alla destra dei blocchi di fase. Utilizzano dei qualificatori per descrivere al meglio i comportamenti delle azioni da compiere, e questi possono essere:

- N (Normal): L'azione è attiva finché è attivo il passo;
- L (Limited): L'azione viene iniziata quando è attivo il passo. Continua finché il passo viene disattivato o quando è trascorso il tempo impostato;

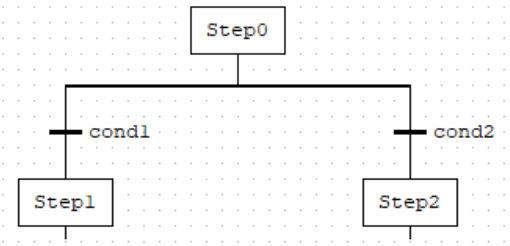
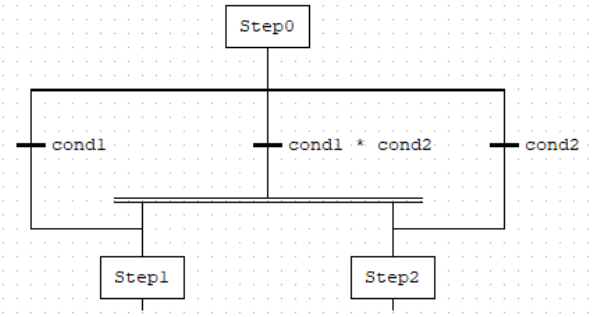
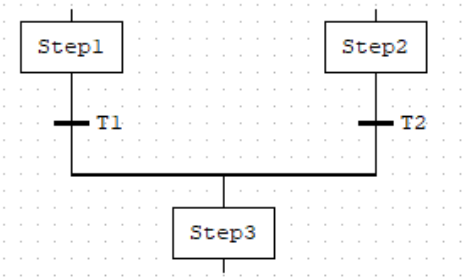
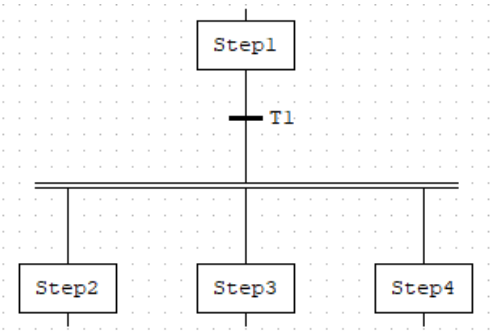
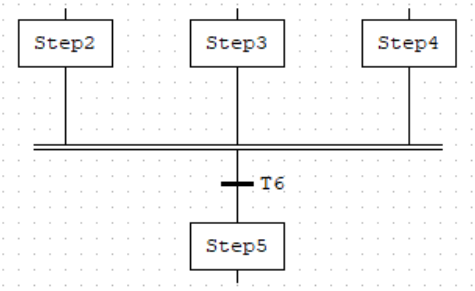
[8] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

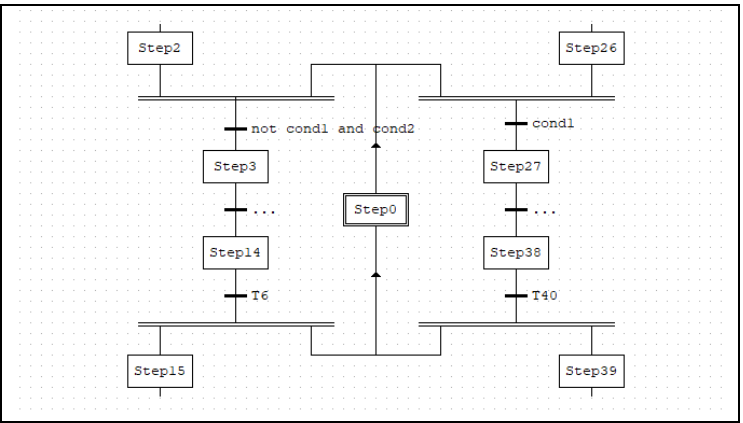
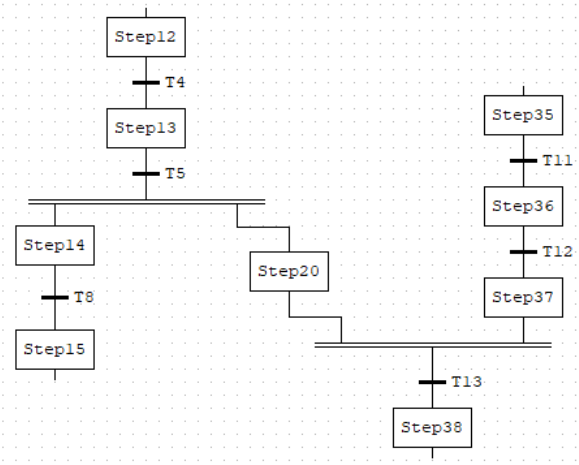
- S (SET): L'azione viene iniziata quando è attivo il passo e continua anche dopo che viene disattivato, finché l'azione non sarà sottoposta a reset;
- R (RESET): L'azione viene disattivata;
- D (Delayed): Quando il passo viene attivato parte un timer di ritardo. Se il passo è ancora attivo dopo il ritardo di tempo, l'azione inizia e continua finché non sarà disattivata;
- P (Pulse): L'azione viene iniziata quando il passo è attivato/disattivato e viene eseguita una volta.

Per i qualificatori L e D è necessario un valore di tempo nel formato TIME (es. L T#5s).[9]
Il Sequential Functional Chart adotta delle strutture classiche per il controllo del flusso elettrico dei blocchi funzionali. Si differenziano in:

- Scelta: quando una fase è seguita da più transizioni. Attivato *Step0*, si hanno due condizioni *cond1* e *cond2* esprimendo la scelta tra due possibilità.
- Scelta non ambigua: la decisione di attivare contemporaneamente le due fasi è chiaramente espressa.
- Convergenza: si ha quando più sequenze terminano nella stessa fase attraverso transizioni diverse.
- Parallelismo: si ha quando una transizione è seguita da più fasi. Se la transizione *T1* diventa superabile saranno attivate le fasi *Step2*, *Step3*, *Step4*.
- Sincronizzazione: si ha quando più fasi precedono la stessa transizione. Condizione necessaria affinché la transizione sia superata è che tutte le sequenze parallele sono terminate.
- Mutua esclusione: viene realizzata tramite la struttura a semaforo. Assume una particolare importanza quando due zone di attività agiscono su una stessa risorsa fisica. Per assicurare mutua esclusione tra le due zone della struttura semaforo (costituita da fase *Step0* e transizioni *not cond1 and cond2*, *cond1*, *T6* e *T40*), le condizioni di ingresso *not cond1 and cond2* e *cond1* devono essere mutuamente esclusive.
- Sincronizzazione locale: si ha quando al raggiungimento di una certa sequenza, si deve attendere il superamento di un'altra sequenza prima di poter procedere oltre. La sequenza di destra, una volta raggiunta la fase *Step37*, deve attendere il superamento della transizione *T5* prima di procedere.[10]

[9] sito web: https://product-help.schneider-electric.com/
[10] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

Tipo	Struttura
Scelta	
Scelta non ambigua	
Convergenza	
Parallelismo	
Sincronizzazione	

Mutua esclusione	
Sincronizzazione locale	

11

3.2.1 Reti di Petri

Il Sequential Functional Chart è un linguaggio derivante direttamente dal formalismo delle Reti di Petri. Le reti di Petri sono un modello di sistemi ad eventi discreti che trae origine dal lavoro di Carl Adam Petri, un ricercatore tedesco. Il modello di rete di Petri più semplice descritto è detto *rete posto/transizione* (o rete P/T). Si tratta di un modello logico che non consente di rappresentare la temporizzazione degli eventi ma solo l'ordine con cui essi si verificano. Una rete di Petri è definita da una quadrupla di matrici N= <P, T, Pre, Post> dove:

- P: Matrice dei nodi posto, di dimensione mx1;
- T: Matrice delle transizioni, di dimensione nx1;
- Pre: Matrice delle marcature per lo scatto, o funzione di ingresso, di dimensione mxn;
- Post: Matrice delle marcature creato dallo scatto, o funzione di uscita, di dimensione mxn.

I posti (p) sono rappresentati da cerchi. Formalmente l'insieme dei posti è indicato con P={p1,p2,p3,...}. Le transizioni (t) sono rappresentate da barre o da rettangoli. Formalmente, l'insieme delle transizioni è indicato con T={t1,t2,t3,...}. Si suppone che P∩T = Ø, cioè posti e transizioni sono insiemi disgiunti, e che PUT ≠ Ø, cioè la rete è costituita da almeno un posto o da una transizione. Le matrici Pre e Post sono composte

[11] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

da numeri interi non negativi e tali possono essere compattate nella matrice di incidenza C seppur perdendo qualche informazione sulla struttura della rete. La matrice di incidenza C : P x T → Z è la matrice mxn definita come: C(p,t)=Post(p,t)−Pre(p,t), quindi ogni elemento della matrice di incidenza è la somma algebrica dei rispettivi elementi nelle matrici Post e Pre. Un elemento negativo è associato a un arco "pre", mentre un elemento positivo è associato a un arco "post".
Il meccanismo alla base del funzionamento delle Reti di Petri si realizza attribuendo dei «gettoni» (tokens) ai posti, per indicare il fatto che la condizione descritta da quel posto è soddisfatta. Nei grafi RP un token è indicato da un pallino posizionato all'interno del cerchio del corrispondente posto e l'assegnamento dei gettoni ai posti definisce una marcatura. Quindi *una marcatura è una funzione M : P → N che assegna a ogni posto un numero intero non negativo di marche (o gettoni).*
Nelle Reti di Petri il processo di transizione di stato è fornito dallo spostamento dei tokens attraverso la rete e dal conseguente cambiamento di stato. Quando una transizione è abilitata si dice che può «scattare», ovvero che può verificarsi. Ciò può succedere se M ≥ Pre(·,t), cioè se ogni posto p ∈ P della rete contiene un numero di token pari o superiore alla colonna della matrice Pre per ogni transizione t relativa.
Fra i modelli ad eventi discreti l'importanza delle reti di Petri è prevalente, infatti tra i vantaggi troviamo:

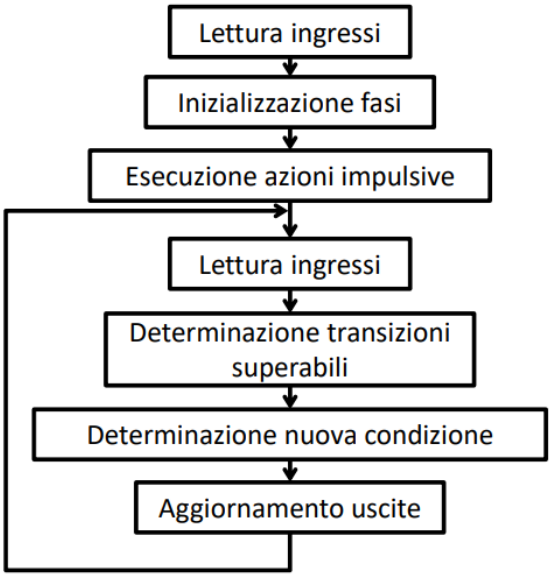
- Le RP sono un formalismo *grafico* e *matematico* allo stesso tempo. Sono di facile comprensione e possono essere usate come strumento di aiuto visuale in fase di progetto e di specifica. Consentono di applicare una vasta gamma di tecniche di analisi.
- Permettono di dare una *rappresentazione compatta* di sistemi con un grande spazio di stato. Spesso una struttura di rete finita può essere usata per descrivere sistemi che hanno un numero infinito di stati.
- Permettono di rappresentare esplicitamente il concetto di *concorrenza*, cioè di attività che possono venire svolte parallelamente.
- Consentono una *rappresentazione modulare*, cioè se un sistema è composto da più sottosistemi che interagiscono fra loro è generalmente possibile rappresentare ciascun sottosistema con una semplice sottorete, poi mediante operatori unire le varie sottoreti per ottenere il modello del sistema complesso.

Le differenze sostanziali tra SFC e Rete di Petri è che nel Sequential Functional Chart lo stato di una fase è booleno, quindi può essere o attivo o disattivo, mentre nelle RP lo stato di una fase è un numero intero. Nell'SFC le transizioni simultaneamente superabili sono superate, mentre nelle Reti di Petri le transizioni superabili sono superate con una sequenza che potrebbe anche non comprenderle tutte. Nell'SFC le condizioni di transizione possono dipendere dallo stato delle singole fasi, mentre nelle RP le condizioni sono solo segnali esterni.[12]

[12] Di Febbraro, A., & Giua, A. (2002). *Sistemi ad eventi discreti.* Genova e Cagliari: McGraw-Hill.

3.2.2 Traduzione SFC in Ladder

A seguito del problema che non tutti i controllori prevedano l'SFC come linguaggio di programmazione, si è sviluppata una procedura per tradurre in linguaggio Ladder un algoritmo progettato in SFC. Tale procedura deve essere il più possibile dettagliata.
Partendo dalle regole di evoluzione precedentemente descritte, si ha la possibilità di realizzare l'SFC tramite i cosiddetti algoritmi di evoluzione, che sono la rappresentazione in forma algoritmica delle regole stesse. Un primo algoritmo di evoluzione prevede una fase di inizializzazione, seguita poi da un ciclo iterativo. Nella fase di inizializzazione si ha la lettura degli ingressi, si attivano le fasi iniziali e si aggiornano le uscite ad esse associate. Nel ciclo iterativo si aggiornano gli ingressi, si determinano quindi le transizioni superabili, si aggiornano le condizioni dell'SFC e si eseguono le uscite. La rappresentazione grafica è riportata al lato mediante uno schema a blocchi. Questo algoritmo è detto *senza ricerca di stabilità*, infatti nel caso di fase non stabile (fase con la transizione di uscita già vera), questo schema commette un errore in quanto eseguirebbe le azioni associate a tutte le fasi intermedie. Nella quasi totalità dei casi questo algoritmo può essere utilizzato senza comportare problemi. Le azioni di lettura degli ingressi e di scrittura delle uscite sono già realizzate dal sistema operativo del PLC in maniera trasparente per l'utente. Ciò che resta da compiere è codificare le operazioni di inizializzazione delle fasi, le esecuzioni delle azioni, la valutazione delle transizioni e l'aggiornamento delle condizioni. Quindi sviluppare le quattro sezioni di cui è composto il programma in linguaggio a contatti che codifica l'algoritmo di evoluzione.



Prima di rappresentare le quattro sezioni in sequenza si fanno le associazioni. Dunque si associa a ogni fase una variabile che funga da marker di fase, con lo scopo di indicare lo stato di attivazione o disattivazione della fase stessa. Tali marker saranno le condizioni per l'esecuzione delle azioni corrispondenti e per la valutazione delle transizioni. Stesso procedimento lo si applica alle transizioni, quindi si associa ad ognuna di esse una variabile che funga da segnalatore della transizione, rappresentandone lo stato.

La *sezione di inizializzazione* deve essere eseguita una sola volta, in essa si settano a 1 i bit di memoria associati ai marker delle fasi iniziali e terminerà con un salto condizionato alla sezione delle azioni.

La *sezione di esecuzione delle azioni* deve eseguire le azioni associate alle fasi attive, quindi all'aggiornamento dell'area di memoria riservata alle uscite. Si deve prevedere un rung di abilitazione per ogni azione continua e si usano i marker delle fasi attive come condizione per eseguire tali azioni.

La *sezione di valutazione delle transizioni* deve valutare la superabilità delle transizioni aggiornando lo stato dei relativi segnalatori. Quindi per ogni transizione è necessario un

rung e le istruzioni sullo stesso devono esprimere la superabilità della singola transizione.

La *sezione di aggiornamento della condizione* deve attivare le fasi a valle e disattivare quelle a monte. Sarà composta da un'istruzione per ogni transizione per disattivare le fasi a monte e un'istruzione per ogni transizione per attivare le fasi a valle.[13]

3.3 Linguaggio FBD

Il Diagramma a Blocchi Funzionali (*Function Block Diagram* o *FBD*) è un linguaggio di programmazione grafico basato su una serie di reti connesse tra loro in grado di descrivere la funzione tra le variabili di ingresso e le variabili di uscita.
La struttura grafica è data da blocchi funzionali e linee di collegamento. Un blocco funzionale ha due caratteristiche principali: la definizione degli ingressi e delle uscite, e un algoritmo che gestisce i valori correnti degli ingressi e delle variabili interne producendone i valori delle uscite. Una funzione è descritta come un insieme di blocchi elementari dove le variabili di uscita dei blocchi sono collegate agli ingressi dei blocchi successivi tramite linee di circolazione del segnale. Il flusso di connessione quindi si ha per mezzo di linee di collegamento. Non è concesso il collegamento in parallelo tra uscite, ma sarà necessario utilizzare l'apposita funzione definita come blocco OR.
Lo studio delle reti seguirà un certo ordine, quindi si ha l'obbligo di completare la valutazione di una prima rete per poi effettuarne la valutazione di un'altra, utilizzante una o più uscite delle reti precedentemente analizzate. Quindi la connessione è orientata e i dati vengono associati dall'estremità sinistra all'estremità destra.

3.4 Lista di istruzioni

Il linguaggio Lista Istruzioni (*Instruction List* o *IL*) rientra nei cosiddetti linguaggi testuali ed è il metodo di programmazione che più si avvicina alla logica con cui lavora il microprocessore.
Tra le caratteristiche principali che la norma associa a questo sistema di programmazione si ha che le istruzioni sono eseguite in modo sequenziale dal controller. Inoltre ciascuna istruzione deve iniziare su una linea e ognuna di esse deve comprendere un operatore seguito da un eventuale modificatore, in aggiunta a uno o più operandi separati da virgole. Facoltativo è l'inserimento dell'etichetta, che deve precedere l'istruzione ed essere seguita dai due punti (:). Può esserci anche un commento per descrivere il codice e deve essere inserito come ultimo elemento della linea e delimitato all'inizio e alla fine con "(*" e "*)".
La semantica degli operatori deve essere *risultato := risultato operatore operando*, cioè il valore dell'espressione viene sostituito da quello ottenuto con l'intervento tra operatore e il rispettivo operando. Secondo la normativa gli operatori standard, i loro eventuali modificatori e gli operatori autorizzati devono seguire delle regole ben definite:

[13] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

Operatore	Modificatori	Operando
LD	N	Pone il risultato corrente uguale all'operando
ST	N	Memorizza il risultato corrente nella locazione dell'operando
S	Solo se il risultato corrente è = 1	Posiziona l'operando a 1 (BOOL)
R	Solo se il risultato corrente è = 1	Riposiziona l'operando a 1 (BOOL)
AND	N, (	AND booleano (BOOL)
&	N, (	AND booleano (BOOL)
OR	N, (	OR booleano (BOOL)
XOR	N, (	OR esclusivo booleano (BOOL)
ADD	(	Addizione
SUB	(	Sottrazione
MUL	(	Moltiplicazione
DIV	(	Divisione
GT	(	Comparazione >
GE	(	Comparazione >=
EQ	(	Comparazione =
NE	(	Comparazione <>
LE	(	Comparazione <=
LT	(	Comparazione <
JMP	C, N	Salto verso l'etichetta (LABEL)
CAL	C, N	Chiamata di un blocco funzionale, seguito da una lista di argomenti tra parentesi (NAME)
RET	C, N	Ritorno di una funzione o un blocco funzionale chiamati
)	C, N	Valutazione di una operazione differita

Per ogni istruzione si ha prima di tutto il caricamento dei valori nell'accumulatore tramite l'operatore LD. Successivamente, l'operazione viene eseguita con il primo parametro tratto dall'accumulatore. Il risultato sarà disponibile nell'accumulatore stesso, dove la memorizzazione si effettua con l'istruzione ST.
In generale il linguaggio IL è adatto per compiti molto specifici quali l'interfacciamento di hardware particolare. Anch'esso, come il LD, è disponibile per tutti i PLC.

3.5 Testo strutturato

Il Testo Strutturato (*Structured Text* o *ST*) è un linguaggio testuale ad alto livello che sviluppa la struttura dalla base dell'informatica. Ha una sintassi simile al linguaggio di programmazione Pascal, dove le istruzioni che vengono utilizzate devono terminare con un punto e virgola (;).
A differenza del linguaggio IL è possibile utilizzare numerose costruzioni per programmare i loop. Le istruzioni utilizzabili sono:

- RETURN: con lo scopo di permettere un'uscita prematura da una funzione o da un blocco funzionale.
- IF: istruzione di selezione che esegue un gruppo di istruzioni solamente se l'espressione booleana associata è 1.
- CASE: istruzione di selezione composta da un'espressione data da una variabile da valutare e da una lista di gruppi di istruzioni differenziati da un'etichetta con uno o più campi di valori interi.
- FOR: istruzione di iterazione, il gruppo di istruzioni associate deve essere eseguito in maniera ripetitiva.
- EXIT: istruzione di iterazione utilizzata per terminare le iterazioni prima che la condizione finale venga soddisfatta.
- WHILE: istruzione che termina l'esecuzione ripetuta di una sequenza di istruzioni (che si estende fino alla parola chiave END_WHILE) fino a quando l'espressione booleana associata è falsa.
- REPEAT: istruzione che determina l'esecuzione ripetuta della sequenza di istruzioni (che si estende fino alla parola chiave UNTIL) fino a quando la condizione booleana associata è vera. [14]

[14] Venturi, E., & Ortolani, G. (2018). *Manuale di Elettrotecnica, Elettronica e Automazione.* Milano: Hoepli.

4 OpenPLC

OpenPLC è un controllore logico programmabile open source basato su un software facile da usare. È il primo PLC open source standardizzato e completamente funzionale, sia nel software che nell'hardware. Il progetto OpenPLC è stato realizzato in conformità con lo standard IEC 61131-3, che definisce l'architettura software di base e i linguaggi di programmazione per i PLC. Il progetto OpenPLC è composto da due parti:

- Runtime: è un software portatile progettato per funzionare dal più piccolo di tutti i microcontrollori ai potenti server nel cloud. Esso è responsabile dell'esecuzione dei programmi PLC creati utilizzando l'editor.
- Editor: è il software che gira sul computer e viene utilizzato per creare i programmi PLC.[15]

Il *Runtime OpenPLC* consente di eseguire programmi PLC creati nell'editor OpenPLC. Il runtime principale ha un server web integrato che consente di configurare diversi parametri. Le micro implementazioni del runtime OpenPLC (ovvero le versioni che vanno sui microcontrollori e sulle schede Arduino) non hanno il server web integrato, ma vengono eseguite direttamente dalla finestra di dialogo di caricamento dell'editor OpenPLC.
L'*Editor OpenPLC* è un editor di codice PLC conforme allo standard IEC 61131-3. Consente di creare, compilare e caricare i programmi nel runtime OpenPLC. L'editor è molto semplice da usare e supporta tutti e cinque i linguaggi definiti nello standard IEC e descritti precedentemente.
I progetti PLC vengono archiviati in cartelle, dove il file di progetto XML principale segue lo schema PLCopen TC6-XML. PLCopen è un'associazione mondiale indipendente da fornitori e prodotti che definisce standard internazionali per vari argomenti relativi alla programmazione dei controlli. Uno standard unificato consente ai fornitori di importare ed esportare programmi PLC creati su piattaforme diverse.

[15] sito web (aggiornato il 05-10-2022): https://autonomylogic.com/docs/openplc-overview/

5 Esempi sviluppati

Mediante il software OpenPLC sono stati svolti alcuni esempi di programmi SFC.
Successivamente sono state fatte le dovute trasformazioni in linguaggio a contatti utilizzando la teoria descritta precedentemente. Di seguito vengono rappresentati e descritti gli esempi.

5.1 Esempio 1 (Trapano Automatico)

Il testo del primo esempio è il seguente:

Il sistema di controllo ha a disposizione i seguenti segnali d'ingesso del tipo digitale:

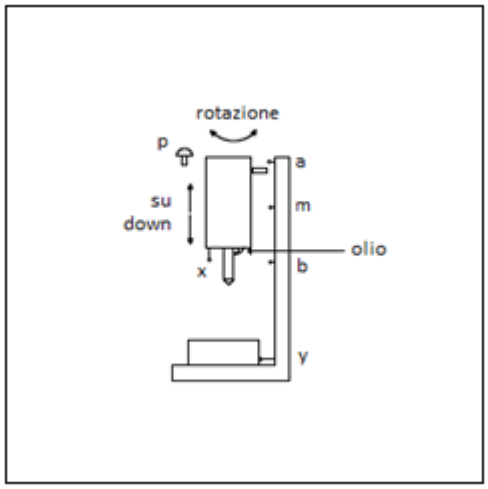
p pulsante per comandare la partenza del ciclo;
a finecorsa della slitta del trapano in alto;
m contatto di media corsa della slitta del trapano;
b finecorsa della slitta del trapano in basso;
x contatto che segnala l'altezza del pezzo;
y contatto che segnala la presenza del pezzo.

I segnali di comando, di tipo digitale, sono:

S comanda la salita del trapano;
D comanda la discesa del trapano;
R attiva la rotazione del trapano;
O attiva l'oliatore.

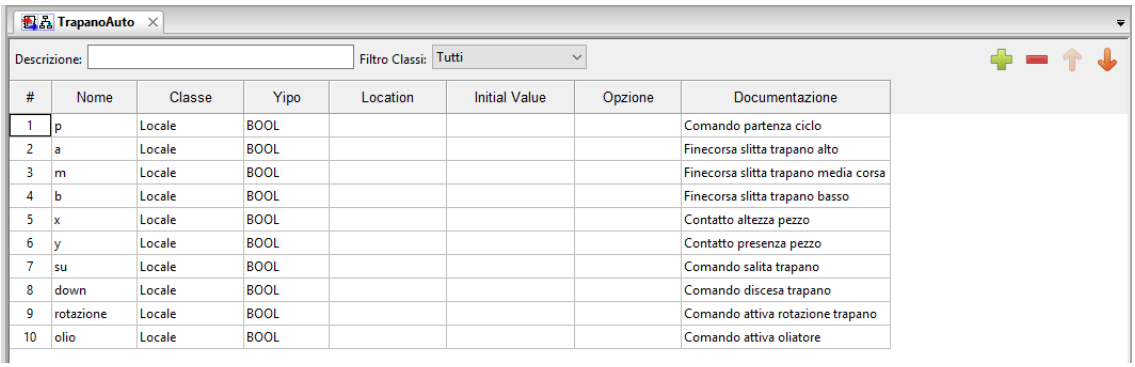
Il contatto di altezza pezzo x segnala se il pezzo è da considerarsi basso, quando diventa vero dopo che lo è diventato il contatto di media corsa m, o alto, quando accade l'inverso. Il ciclo inizia quando l'operatore preme il pulsante di partenza, se è presente un pezzo. Se il pezzo è basso il trapano viene messo in rotazione e fatto scendere fino al finecorsa basso; poi viene fatto risalire fino al finecorsa alto dove viene fermata la rotazione. Se il pezzo è alto, il trapano, messo in rotazione, deve scendere fino al contatto medio e poi risalire: quindi deve essere attivata la pompa oliatrice per 2 secondi; infine il trapano deve scendere fino al finecorsa basso e poi risalire ed essere fermato. Un nuovo ciclo può avviarsi solo se il pezzo trattato viene rimosso e sostituito con un altro.[16]

La rappresentazione grafica del sistema da sviluppare è descritta nella figura in basso.



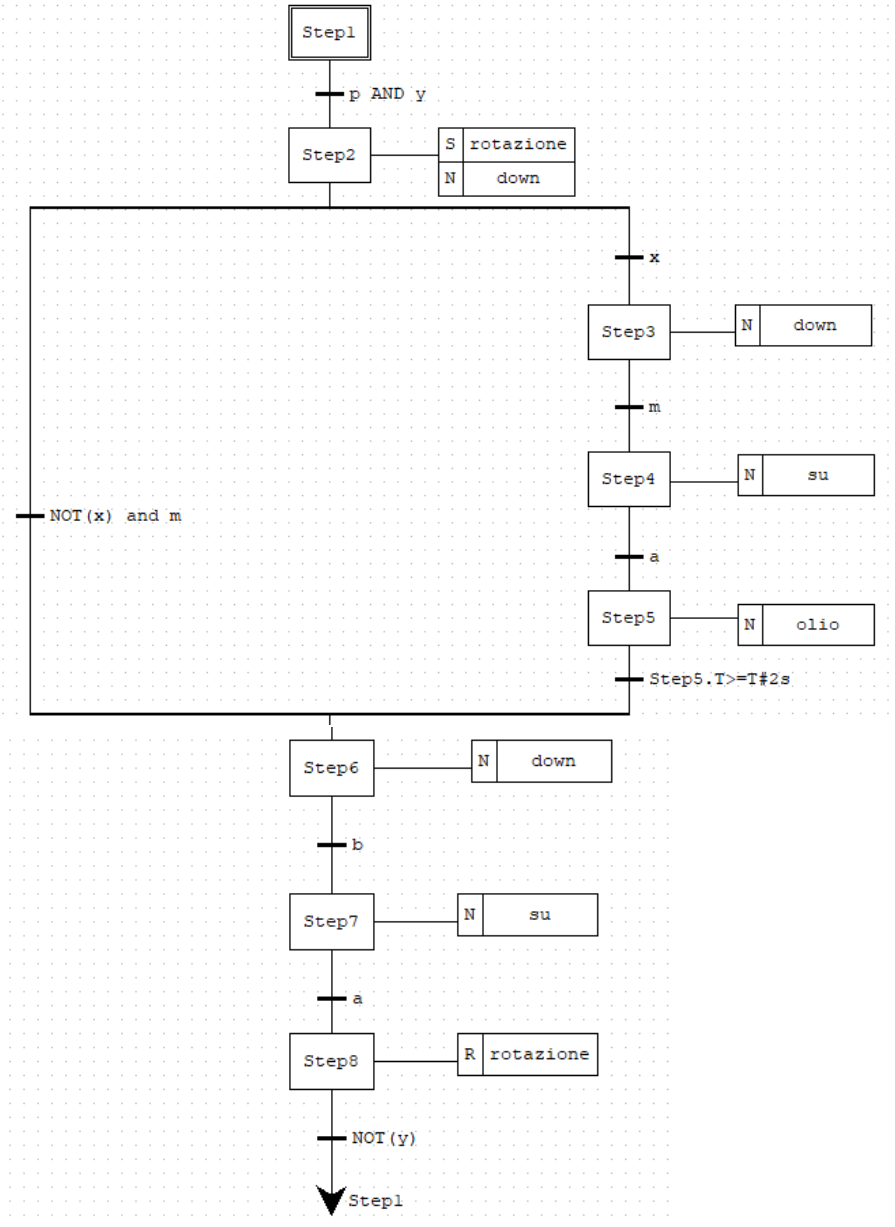
[16] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

Una volta aperto l'editor di OpenPLC e avviato un nuovo programma, si inizia la realizzazione del programma in linguaggio SFC. In primis è necessaria la dichiarazione delle variabili che verranno utilizzate, aggiungendone una piccola descrizione alla destra per facilitare la comprensione.

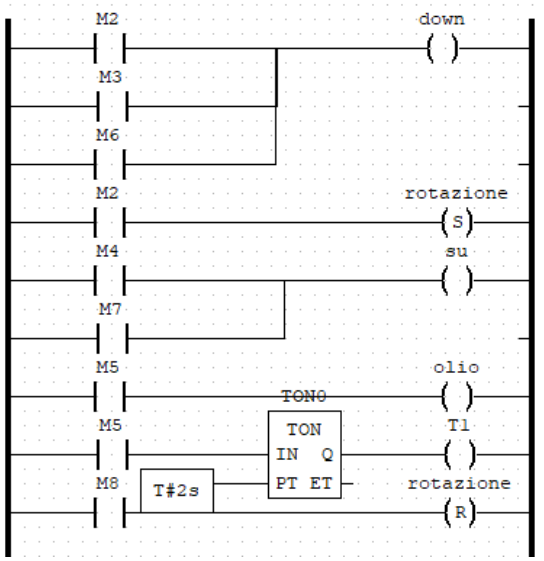


#	Nome	Classe	Yipo	Location	Initial Value	Opzione	Documentazione
1	p	Locale	BOOL				Comando partenza ciclo
2	a	Locale	BOOL				Finecorsa slitta trapano alto
3	m	Locale	BOOL				Finecorsa slitta trapano media corsa
4	b	Locale	BOOL				Finecorsa slitta trapano basso
5	x	Locale	BOOL				Contatto altezza pezzo
6	y	Locale	BOOL				Contatto presenza pezzo
7	su	Locale	BOOL				Comando salita trapano
8	down	Locale	BOOL				Comando discesa trapano
9	rotazione	Locale	BOOL				Comando attiva rotazione trapano
10	olio	Locale	BOOL				Comando attiva oliatore

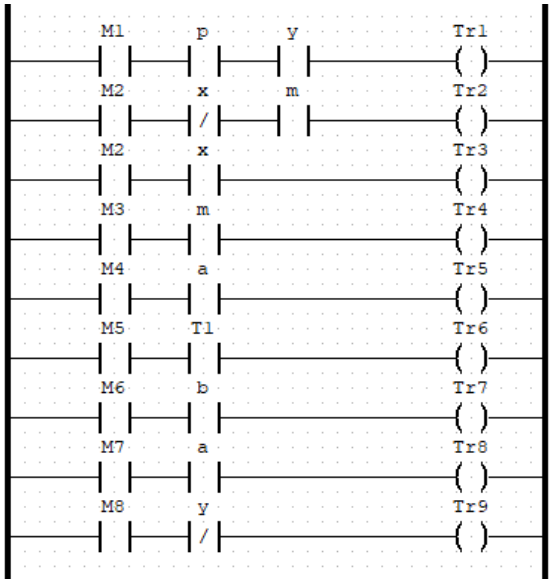
Successivamente si inseriscono gli stati del programma connessi tra loro dalle opportune transizioni. Ogni step dovrà sviluppare determinate azioni. Si arriva dunque allo sviluppo del linguaggio SFC:



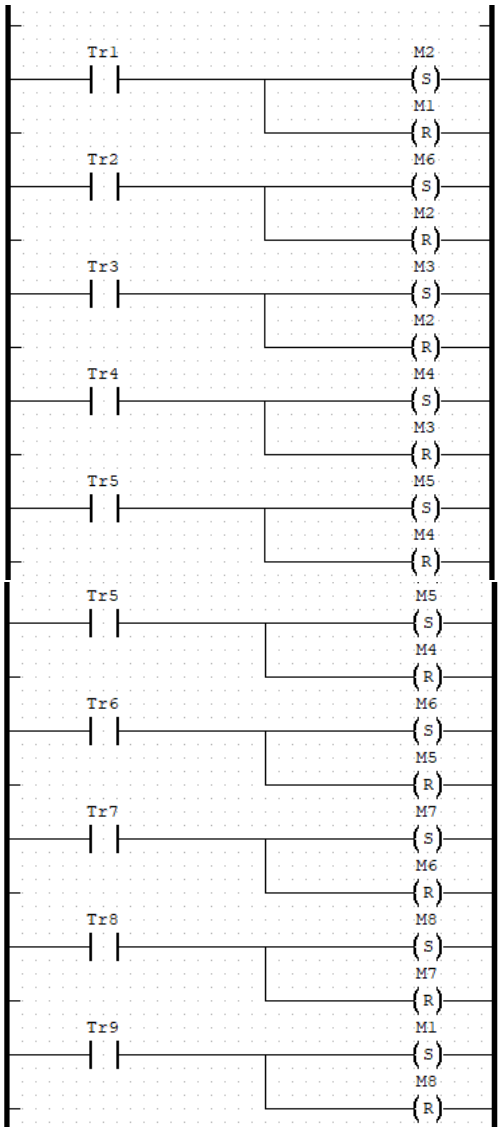
Dopo averne fatta la simulazione in OpenPLC e costatato il corretto funzionamento si può procedere con lo sviluppo della traduzione nel linguaggio a contatti. Come anticipato precedentemente il codice Ladder sarà suddiviso in quattro parti: sezione di inizializzazione, sezione di esecuzione delle azioni, sezione di valutazione delle transizioni e sezione di aggiornamento della condizione.
La rappresentazione grafica della sezione di inizializzazione non è stata possibile in quanto il software con consente jump di programma, ma è stata realizzata nel momento della simulazione. Infatti nella simulazione si setta M1 (marker riferito alla fase Step1), il contatto di partenza ciclo (p) e quello di presenza pezzo (y).
Si prosegue quindi con la sezione di esecuzione delle azioni, quindi per ogni azione si connettono i rispettivi marker di fase:



Nella sezione di valutazione delle transizioni si creano le condizioni logiche per l'attivazione di ogni transizione:



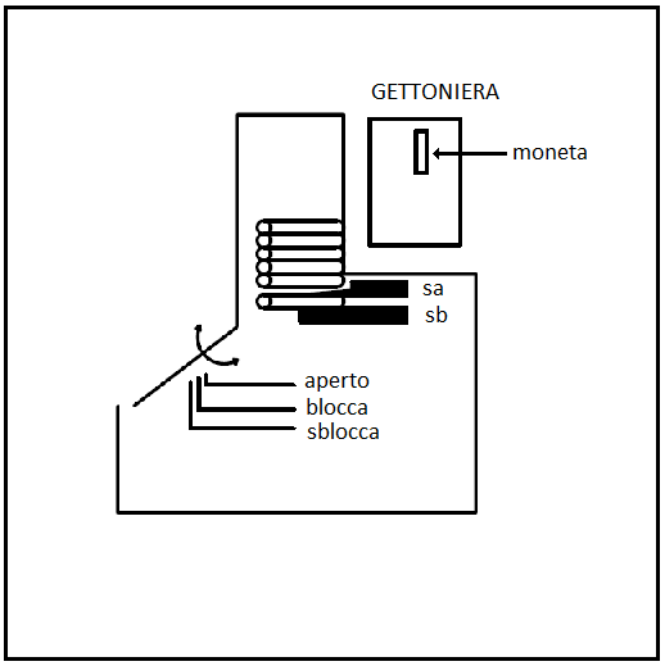
Infine il programma Ladder termina con la sezione di aggiornamento:



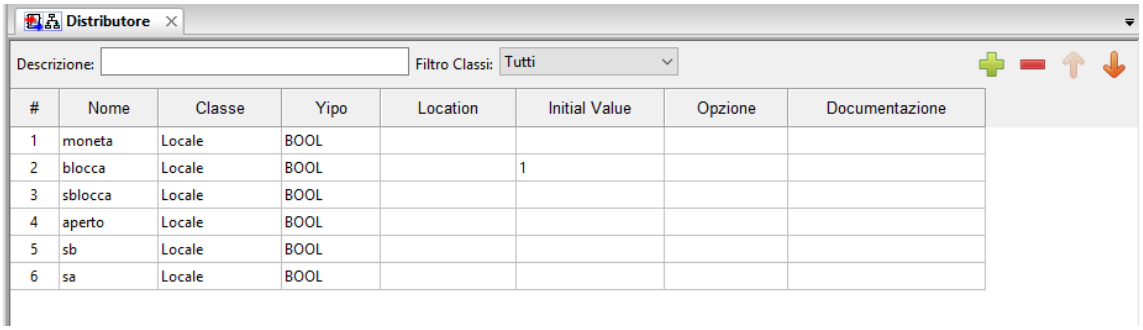
Per concludere si genera la simulazione del programma con il linguaggio a contatti.

5.2 Esempio 2 (Distributore Automatico di Bibite)

Il testo del secondo esempio prende come riferimento un distributore di bibite:
Gli ingressi della parte di controllo sono il segnale MONETA dalla gettoniera e il segnale APERTO che indica che il vano di prelievo è stato aperto. Le uscite sono i segnali BLOCCA e SBLOCCA che agiscono sulla porta del vano di prelievo, e i segnali SA e SB per il comando dei due pistoncini lineari (se il segnale è presente il pistoncino va a destra, altrimenti una molla di richiamo lo riporta a sinistra). Si vuole realizzare l'automatismo in modo che, all'inserimento di una moneta, una sola bibita sia fatta cadere nel vano e sia possibile prelevarla.[17]
La rappresentazione del sistema che si vuole sviluppare è quella riferita alla figura sottostante:



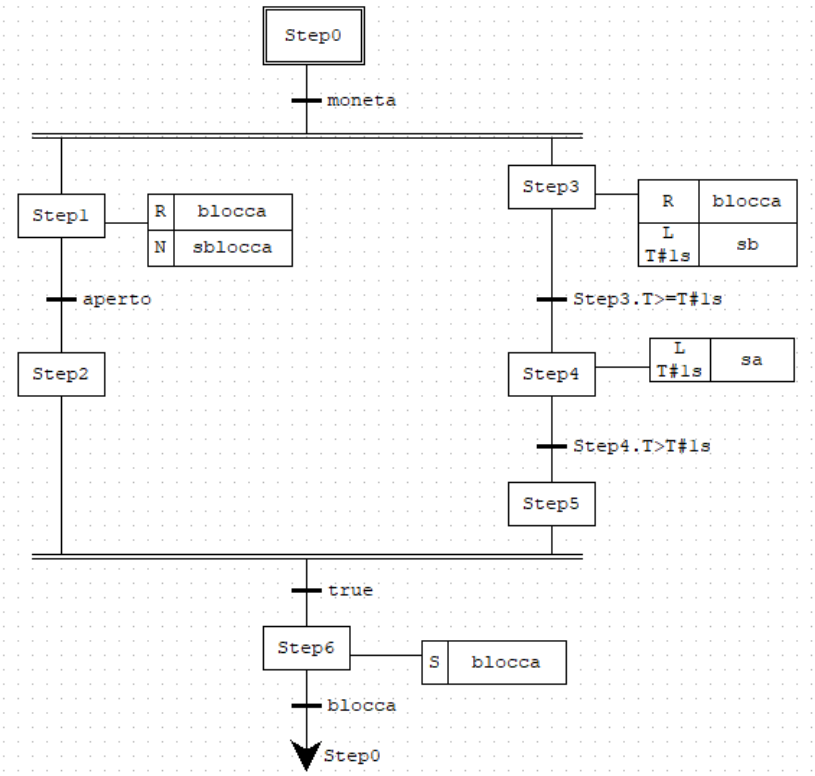
Una volta compreso il sistema da realizzare si può procedere con la programmazione SFC. Dunque si inizializzano le variabili:



#	Nome	Classe	Yipo	Location	Initial Value	Opzione	Documentazione
1	moneta	Locale	BOOL				
2	blocca	Locale	BOOL		1		
3	sblocca	Locale	BOOL				
4	aperto	Locale	BOOL				
5	sb	Locale	BOOL				
6	sa	Locale	BOOL				

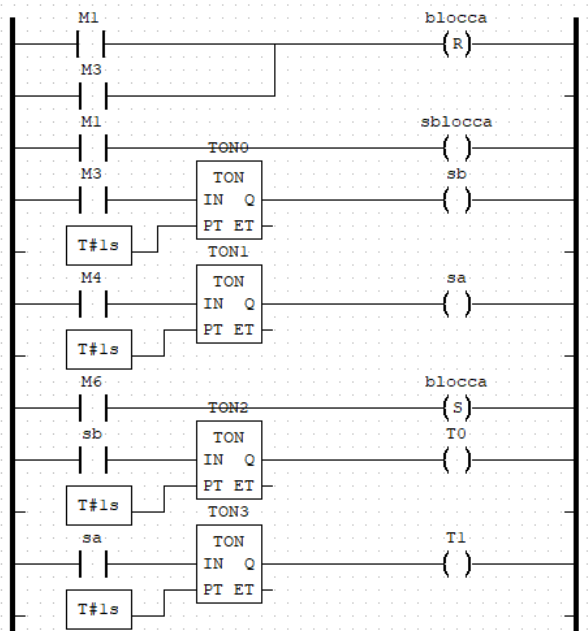
Notiamo che la variabile *blocca* è inizialmente settata ad un valore iniziale pari ad 1, con la funzione di poter tenere il vano di prelievo bloccato già all'inizio del ciclo. Il linguaggio SFC relativo all'esempio può essere del tipo:

[17] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

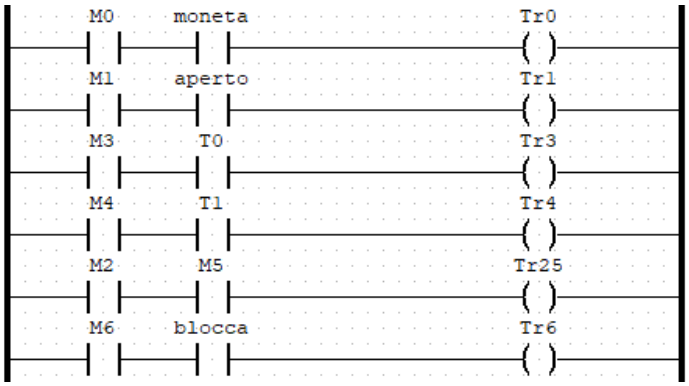


Dunque una volta inserita la moneta nella gettoniera si ha un parallelismo, uno riguardante la lavorazione del vano di prelievo (ramo di sinistra) e l'altro per la gestione della bibita (ramo di destra). La lavorazione del vano di prelievo è molto semplice, nello Step1 si sblocca il vano e tale rimarrà fino a quando non verrà aperto, cioè non verrà prelevato il prodotto. Per quanto riguarda il ramo di destra, si devono gestire le azioni di comando dei pistoncini mediante i segnali sa e sb. Dunque si attiva sb per 1sec, in modo da far cadere la bibita nel vano di prelievo, per poi tornare allo stato iniziale mediante una molla di richiamo. Successivamente si attiva sa per occupare il posto del prodotto che verrà erogato, e quindi poter riavere il sistema pronto per l'attivazione del prossimo ciclo di lavorazione.

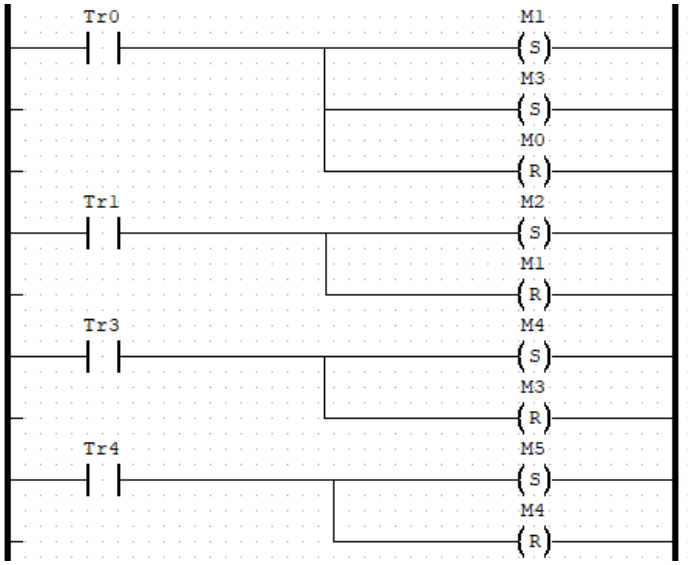
La trasformazione SFC-Ladder è stata effettuata suddividendo il linguaggio a contatti con le quattro sezioni di codice. Non potendo realizzare la sezione di inizializzazione, questa è stata eseguita manualmente al momento della simulazione. Infatti prima dello start si setta M0, marker riferito alla fase Step0, e il segnale moneta della gettoniera. La sezione di esecuzione delle azioni è riportata di seguito:

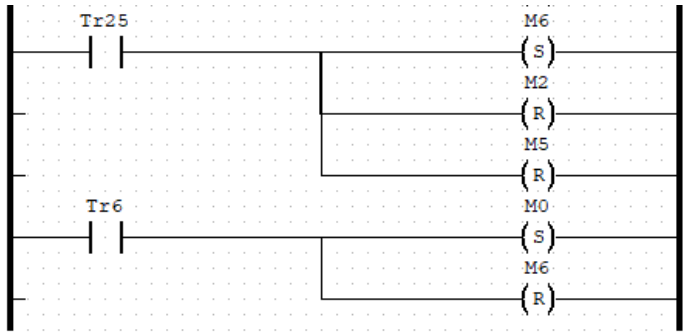


Il codice nel linguaggio a contatti prosegue con la sezione di valutazione delle transizioni riportata di seguito:



Infine il programma termina con l'ultima sezione, quella di aggiornamento della condizione:





Per concludere la realizzazione dell'esempio si genera la simulazione del programma in OpenPLC e si accerta che questo lavora correttamente. Se necessario è possibile aumentare i valori alle variabili temporali per avere uno sviluppo più lento del programma e poterlo seguire con più attenzione.

5.3 Esempio 3 (Sistema Semaforico)

Il testo del terzo esempio prende in considerazione un sistema semaforico:

I segnali di ingresso disponibili, di tipo digitale, sono:

P pulsante richiesta pedonale;

C1 rilevatore presenza continua auto;

C2 rilevatore presenza continua auto.

I comandi disponibili, di tipo digitale, sono:

V1 luce verde del semaforo 1;

G1 luce gialla del semaforo 1;

R1 luce rossa del semaforo 1;

V2 luce verde del semaforo 2;

G2 luce gialla del semaforo 2;

R2 luce rossa del semaforo 2;

Il semaforo 1 è posto su una strada di grande traffico e deve essere normalmente verde.

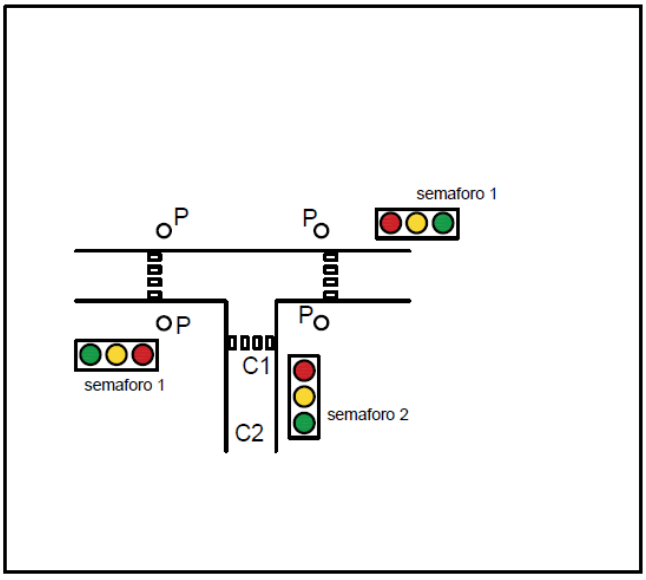
Il ciclo semaforico, descritto più avanti, si deve attivare:

1) *solo se il verde del semaforo 1 è stato presente per almeno 300 secondi;*
2) *immediatamente, se entrambi i rilevatori C1 e C2 segnalano la presenza continua di auto (coda al semaforo 2);*
3) *dopo 30 secondi, se vi è la richiesta da parte di un pedone;*
4) *dopo 240 secondi, se il rilevatore C1 segnala la presenza continua di un auto.*

L'ordine di presentazione delle quattro specifiche è anche quello di priorità che deve essere rispettato nel loro soddisfacimento.

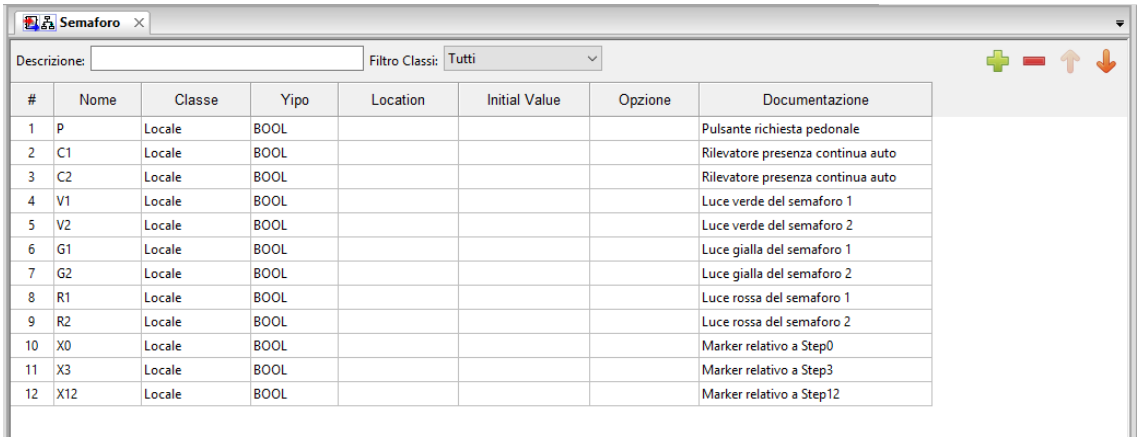
Il ciclo semaforico è quello usuale composto, per ogni semaforo, da una successione verde-giallo-rosso-verde. Il giallo deve durare cinque secondi mentre il verde del semaforo 2 sessanta secondi.[18]

Graficamente si può rappresentare il sistema da gestire prendendo come riferimento la figura di sotto.



[18] Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

Si inizia la programmazione del sistema semaforico con la dichiarazione delle variabili all'interno del software OpenPLC, come anche visto precedentemente:

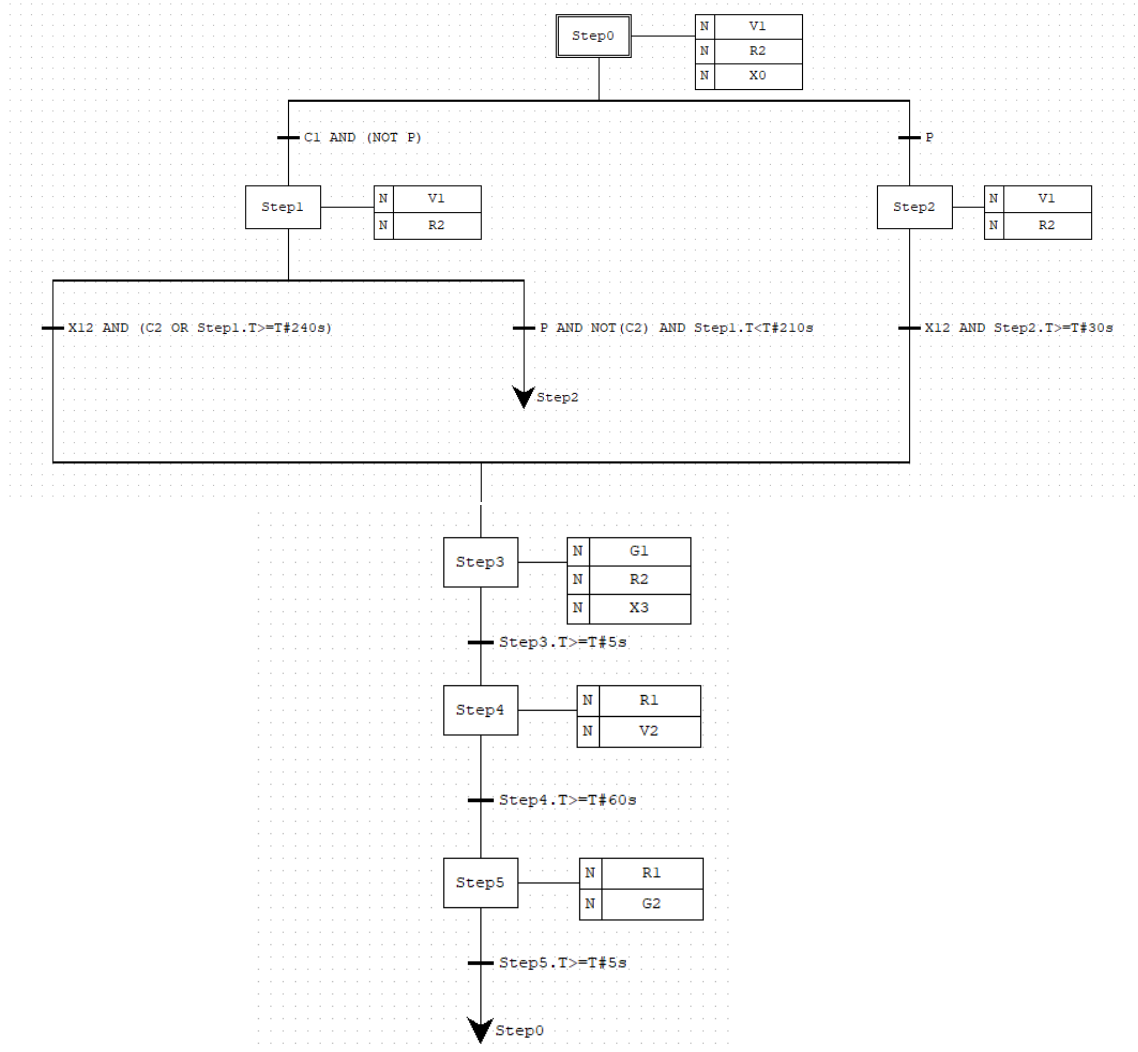


Semaforo

Descrizione: Filtro Classi: Tutti

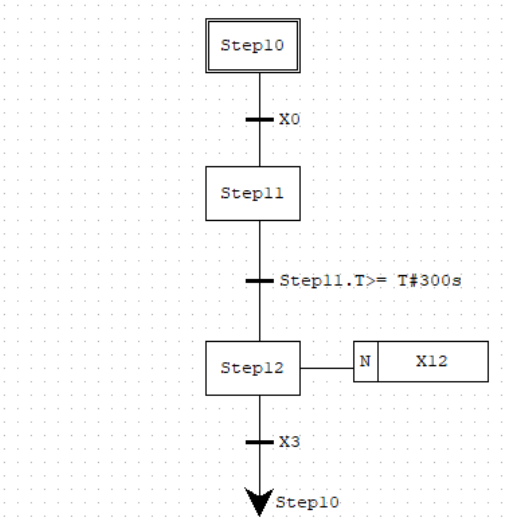
#	Nome	Classe	Yipo	Location	Initial Value	Opzione	Documentazione
1	P	Locale	BOOL				Pulsante richiesta pedonale
2	C1	Locale	BOOL				Rilevatore presenza continua auto
3	C2	Locale	BOOL				Rilevatore presenza continua auto
4	V1	Locale	BOOL				Luce verde del semaforo 1
5	V2	Locale	BOOL				Luce verde del semaforo 2
6	G1	Locale	BOOL				Luce gialla del semaforo 1
7	G2	Locale	BOOL				Luce gialla del semaforo 2
8	R1	Locale	BOOL				Luce rossa del semaforo 1
9	R2	Locale	BOOL				Luce rossa del semaforo 2
10	X0	Locale	BOOL				Marker relativo a Step0
11	X3	Locale	BOOL				Marker relativo a Step3
12	X12	Locale	BOOL				Marker relativo a Step12

Oltre alle variabili fornite dal testo sono stati aggiunti tre marker di fase (X0, X3 e X12) per poter gestire al meglio la programmazione. Dunque si svolge l'esempio con il Sequential Functional Chart, ed in particolare con due SFC che si connettono tra loro. Il primo è il seguente:

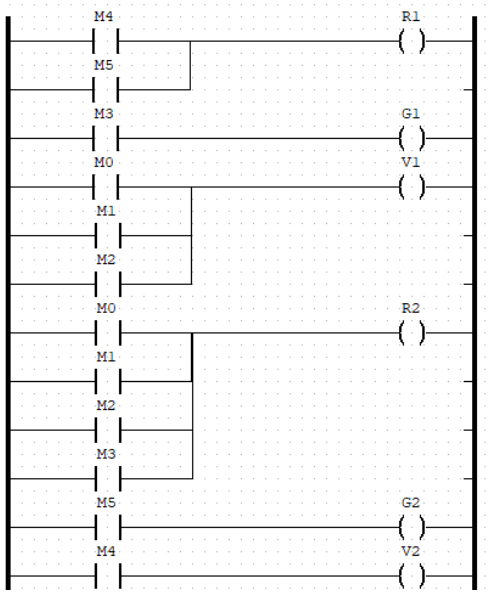


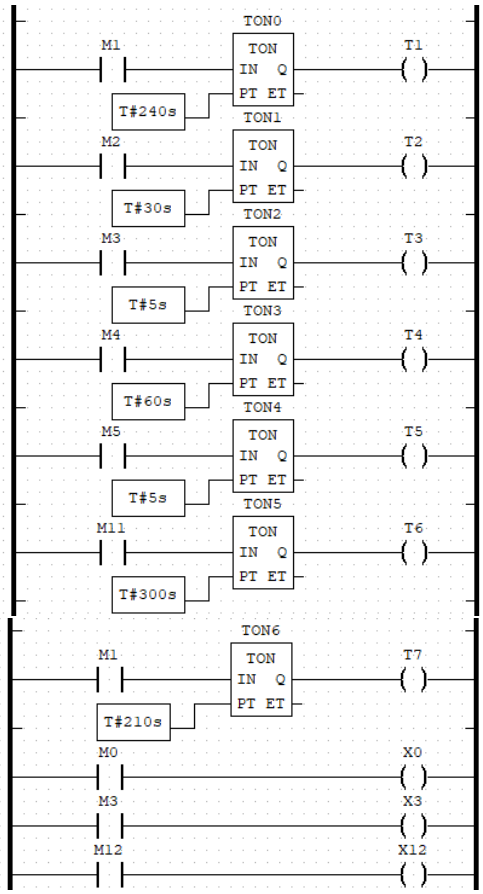
Il primo programma SFC descrive il comportamento del sistema semaforico in base agli eventi che possono avvenire. Dunque nella prima scelta che si incontra, nel ramo di sinistra si svolge il percorso nel caso in cui non ci sia il pulsante di richiesta pedonale

attivo e vi è la presenza continuativa di auto nel primo sensore, nel ramo di destra il caso in cui è attivo il segnale di richiesta pedonale. La convergenza che chiude questo programma SFC porta agli Step3, Step4 e Step5 che descrivono i cambiamenti di stato dei semafori e ne danno una certa durata temporale. Il secondo SFC, invece, serve per la gestione del segnale verde del semaforo 1, quindi serve a soddisfare la specifica 1). Viene rappresentato così:

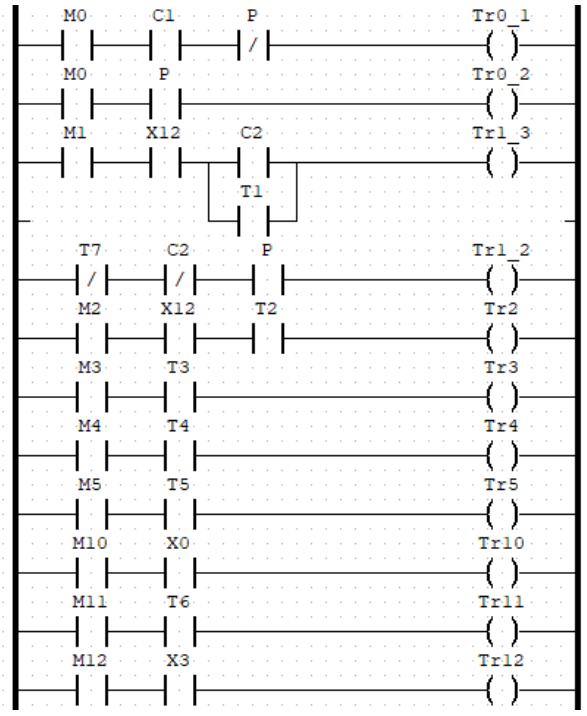


Alla partenza di Step0 si attiva X0 che quindi farà partire il conteggio del secondo SFC, e quindi del segnale verde del semaforo 1. Finito il conteggio si attiva il marker X12 dello Step12 per permettere al programma primario di poter effettuare lo switch dei colori ai semafori. Il marker X3 servirà per concludere il ciclo dell'SFC con stato inziale Step10, verrà attivato quando iniziano i cambiamenti degli stati dei semafori.
Successivamente si effettua la simulazione del programma SFC e, una volta definito il corretto funzionamento si prosegue con la traduzione SFC-Ladder. Anche in questo caso il codice del linguaggio a contatti sarà sviluppato nelle quattro sezioni. La prima, la sezione di inizializzazione, avverrà al momento della simulazione settando M10, il marker riferito alla fase Step10, e successivamente M0, il marker riferito alla fase Step0. Poi la simulazione procederà verso lo Step1 o Step2 in base alle scelte che si vogliono fare. La sezione di esecuzione delle azioni è la seguente:

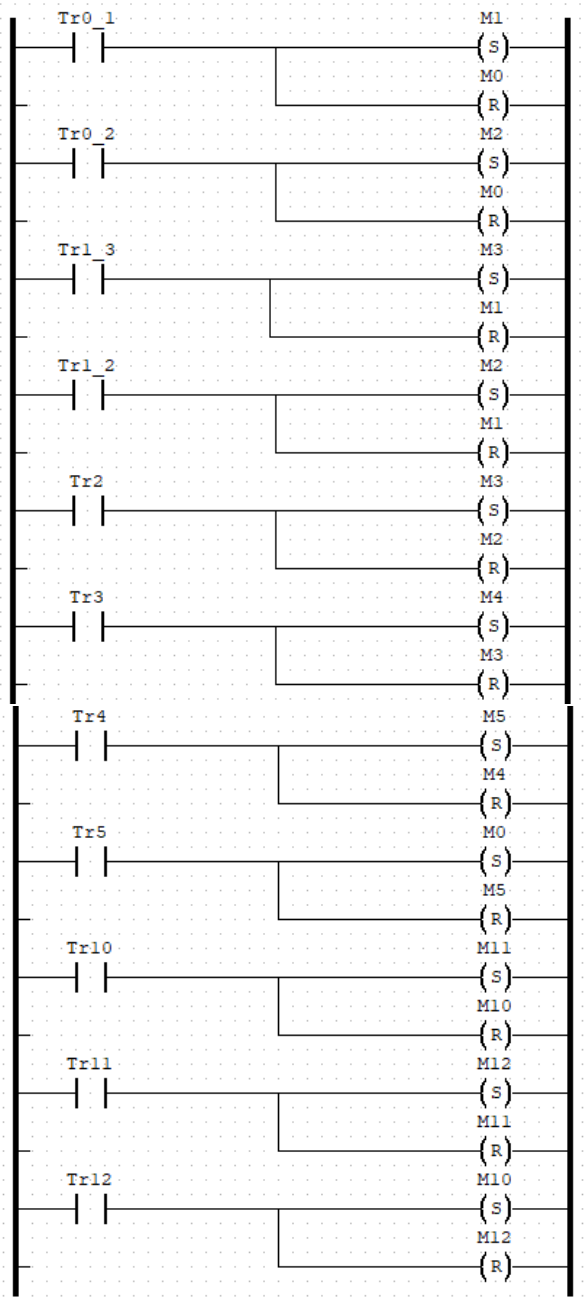




Successivamente si trova la sezione di valutazione delle transizioni:



Il programma si conclude con la sezione di aggiornamento della condizione:



Il terzo esempio si conclude con la simulazione del programma tradotto nel linguaggio a contatti, avente lo stesso procedimento di quello effettuato con il linguaggio SFC.

6 Conclusione

Lo scopo della presente tesi è stato quello di descrivere un sistema di controllo su scala industriale basato sull'utilizzo del PLC. La rappresentazione di un sistema industriale è piuttosto variegata e si diversifica in maniera flessibile in base alle esigenze produttive che si vogliono sviluppare. Alla base di ciò vi è un insieme di elementi comuni che permettono di poter gestire al meglio un certo impianto. Sicuramente l'integrazione del PLC all'interno del settore dell'automazione industriale ha dato numerosi benefici tali da poterlo etichettare come dispositivo centrale per la programmazione di questo settore. Miglioramento dell'efficienza nei processi industriali, qualità e sicurezza sono i parametri chiave della produzione data dall'automazione industriale e sicuramente la flessibilità del PLC ne garantisce un ottimo risultato. Programmare un controllore significa valutare tutte le caratteristiche del sistema, quindi non è solo importante scegliere correttamente il tipo di PLC, ma anche i vari componenti che lo supportano, come ad esempio sensori, macchinari, nastri trasportatori e tutti i devices necessari per garantire la connessione e l'interoperabilità con il controllore stesso. Fondamentale è anche la selezione dei linguaggi di programmazione da adottare per garantire la completezza nel controllo degli step e delle azioni dei singoli devices del sistema stesso. Concludendo si può affermare che per il controllo dei sistemi industriali, al tempo d'oggi e nel futuro prossimo, è necessaria la presenza del PLC essendo un dispositivo molto versatile e dalle ottime prestazioni, dando la possibilità di creare intere reti di comunicazione in grado di monitorare e controllare processi in tempo reale, cioè quello di cui si occupano i sistemi SCADA (*Supervisory Control And Data Aquisition*), sicuramente un tema di rilievo che non è stato approfondito in questa tesi, ma che vede ottime prospettive per i sistemi industriali.

7 Bibliografia e Sitografia

Bibliografia

Chiacchio, P. (1996). *PLC e automazine industriale.* Milano: McGraw-Hill.

Di Febbraro, A., & Giua, A. (2002). *Sistemi ad eventi discreti.* Genova e Cagliari: McGraw-Hill.

Venturi, E., & Ortolani, G. (2018). *Manuale di Elettrotecnica, Elettronica e Automazione.* Milano: Hoepli.

Sitografia

sito web: https://automazione-plc.it/

sito web: https://product-help.schneider-electric.com/

sito web (aggiornato il 05-10-2022): https://autonomylogic.com/docs/openplc-overview/