Laboratorio - Esercizio con Testo Strutturato

Benedetta Vitale ed Emilio Meroni

18 maggio 2024

Indice

1 Itroduzione

Il sistema preso in considerazione è una linea di produzione a 5 stazioni [figura 1]. Il pezzo inizialmente viene posizionato sopra un pallet dalla stazione 1; in seguito subisce diverse lavorazioni da parte delle stazioni: 2, 3 e 5 (saldatura, foratura e avvitatura); infine la stazione 5 esegue un controllo di qualità per poi scaricare il pezzo in un contenitore.

Le stazioni saranno di due principali tipologie:

- **Temporizzate**, stazioni: 2, 3 e 4; esse finiranno l'azione allo scadere di un tempo determinato.

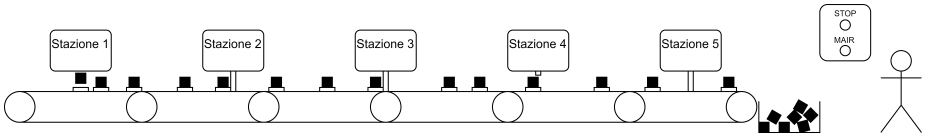

Figura 1: Schema di funzionamento

- **A evento**, stazioni: 1 e 5; le quali, a seguito di un evento, "presenza pezzo" (stazione 1) oppure "controllo qualità eseguito" (stazione 5), termineranno l'operazione.

Inoltre si deve gestire la manutenzione, effettuata ogni 10 pz lavorati.

2 Assunzioni

Le assunzioni con cui abbiamo lavorato sono:

1. Le fotocellule STAZ_# (con # il numero della stazione) saranno pari a *FALSE* se il pezzo non è presente; mentre rimarranno a *TRUE* per tutta la durata della lavorazione, quindi finché il pezzo non abbandona le stazioni.

2. L'uscita dalla stazione, da parte di un pezzo, non implica l'accensione della stazione successiva. Questo si traduce nel nostro codice come: quando una stazione si disattiva, la stazione successiva non si accende automaticamente, ma lo si dovrà fare manualmente attivando la fotocellula. Questa scelta è dovuta al fatto che alcune stazioni sono più lente di altre, generando così delle "code"; questo causerebbe difficoltà nella gestione del programma qualora dovesse succedere che una stazione, per esempio STAZ_3, è ancora in lavorazione mentre la precedente, STAZ_2, finisce di lavorare, con l'effetto di perdere virtualmente un pezzo.

   Questa assunzione ha comportato l'aggiunta: dell'assunzione numero 3 e dell'utilizzo di un contatore in più (spiegato meglio nella sezione 5).

3. I pezzi possono entrare solo dalla prima stazione e uscire dall'ultima, questa assunzione, assieme al secondo contatore, verrà utilizzata per gestire la manutezione.

3 Funzione principale

La sruttura della funzione principale, *PROGRAM_CYCLIC*, viene descritta dalla figura 2. In particolare abbiamo individuato nel sistema tre situazioni diverse:

- **Stato di Fermo**: Tutti gli azionamenti sono spenti, anche se ci sono pezzi sul nastro.

- **Impianto Acceso**: Le stazioni si accendono se è presente un pezzo nella loro zona di lavoro.

- **Stato di Manutenzione**: Tutte le stazioni sono spente perché è richiesta la manutenzione.

Il codice lo si può trovare nella sezione 6.

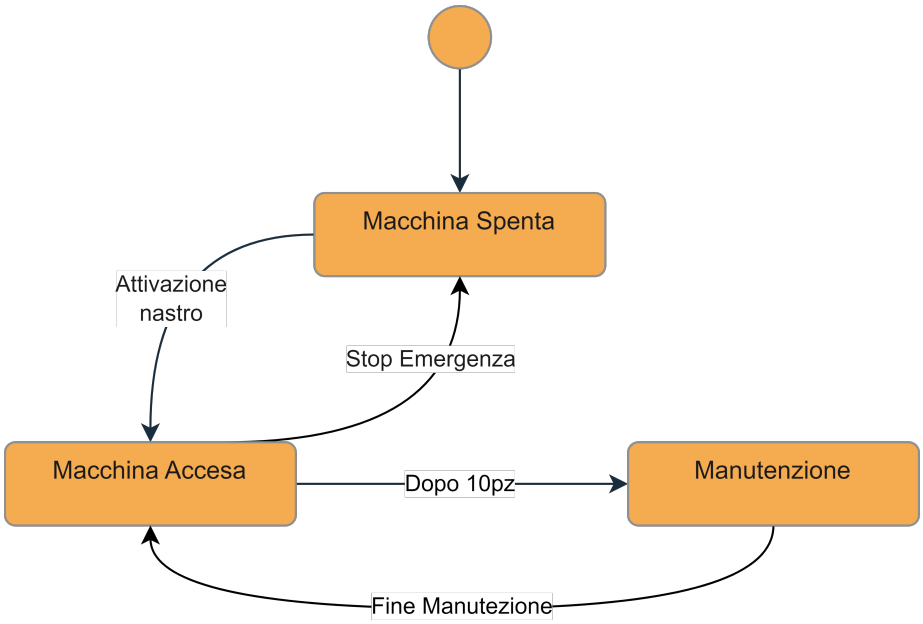

Figura 2: *PROGRAM_CYCLIC*

4 Funzione gestione delle stazioni

Per la gestione delle singole stazioni, lo schema di funzionamento è descritto dallo schema 3. Essa all'ingresso attiverà la stazione (se non l'ha già fatto) e, in base alla tipologia di stazione (temporizzata o a evento), eseguirà il controllo: se la stazione si deve spegnere o no.
Gli input e gli output della funzione sono descritti nella tabella 1.

Nome	**Tipologia**	**Descrizione**
TIPO	Input	Tipologia di stazione: a evento 0 e temporizzata 1
TEMPO	Input	Il tempo di attivazione, nel caso di stazione temporizzata
TRIGGER	Input	Il trigger che disattiva la stazione, nel caso di stazione a evento
AZIONE	Input e Output	L'azione che svolge la stazione

Tabella 1: Descrizione ingressi e uscite della funzione che gestisce le stazioni

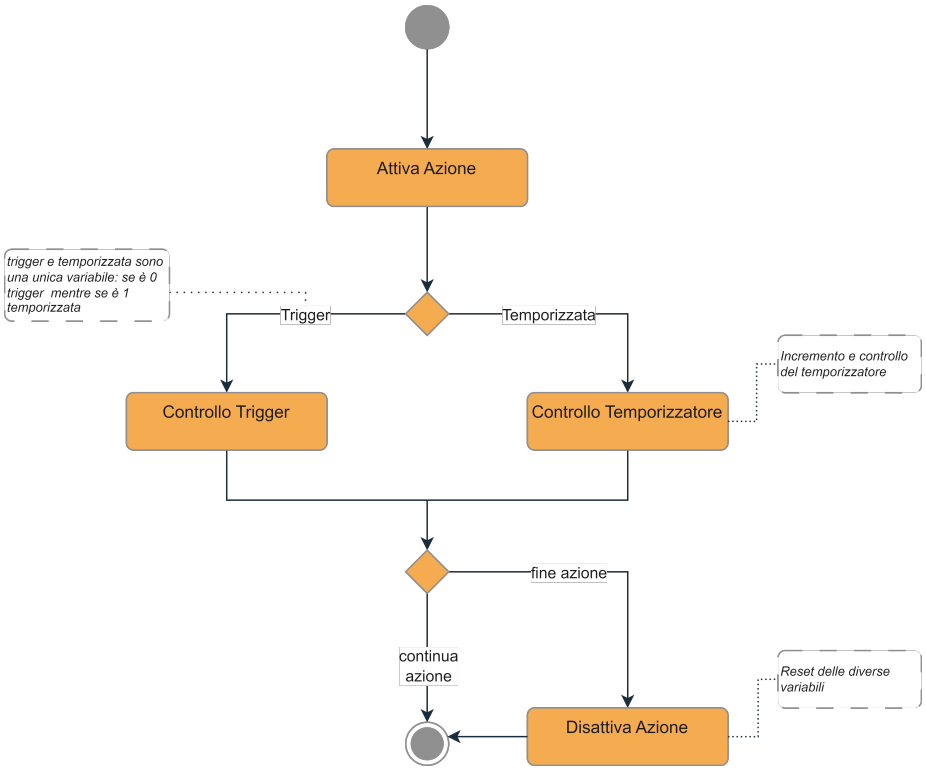

Figura 3: Funzione *GESTIONE_STAZIONE*

5 Problematiche

La probelmatica principale che abbiamo riscontrato durante lo svolgimento è stata nella gestione della manutenzione. Il problema è dovuto al fatto che le stazioni, per nostra assunzione, non lavorano strettamente in modo sequenziale, o meglio, nella realtà se un pezzo esce da una stazione allora esso sicuramente entrerà nella successiva (il fatto che esca non implica l'ingresso nell'altra, magari ci sono altri pezzi in coda). Quindi, come spiegato in precedenza, l'ingresso si deve fare manualmente (anche perché in questo modo le stazioni lavorano in modo totalmente indipendente tra di esse).

La difficoltà è stata nel contare quanti pezzi sono passati all'ultima stazione, problema che inizialmente abbiamo risolto con un contatore alla fine; ma con l'obbiettivo (posto da noi) di avere la linea scarica quando si va in manutenzione, dovevamo bloccare i pezzi a monte, quindi, il contatore lo abbiamo spostato alla prima stazione. Ciò comportava il problema di capire quando entrare nello stato di manutenzione; supporre che la linea era scarica quando tutte le stazioni erano disattivate non era giusto, magari qualche pezzo stava transitando ancora

da una stazione all'altra. Quindi abbiamo risolto inserendo: un sencondo contatore all'uscita dell'ultima stazione e l'assunzione numero: 3.

Ricapitolando: il primo contatore blocca l'ingresso dell'undicesimo pezzo e il secondo contatore, posto all'ultima stazione il quale segnala che sono passati tutti i pezzi (linea scarica), gestisce il cambio di stato in manutenzione.

6 Codice

Structured Text : C:\Users\emili\OneDrive - unibg.it\UNI\Anno III\Automazione Industriale\GitHub\Laboratorio-di-Automazione-Industriale\ST\Logical\Cosberg\Cyclic.st

```
PROGRAM _CYCLIC

    CASE STATO OF
        STATO_SPENTO: // attendiamo l'avvio
            // attivazione del nastro trasportatore
            IF NASTRO_ON THEN
                STATO := STATO_ACCESO;
                NASTRO_ON:= FALSE;
            END_IF;

        STATO_ACCESO:

            // In caso di emergenza spegnamo tutto senza pensarci due volte e cambiamo stato
            IF STOP_EMERGENZA THEN
                ALIMENTAZIONE := 0;
                SALDATURA_ON := 0;
                FORATURA_ON := 0;
                AVVITATURA_ON := 0;
                QUALITY_CHECK := 0;
                // cambiamo stato
                STATO := STATO_SPENTO;
            ELSE  // se non c'è emergenza lavoriamo normalmente
                // se è presente un pezzo in staz_1
                IF STAZ_1 AND C_MAN_IN < CILCI_MANUTENZIONE THEN
                    // chiamo la funzione per la stazione 1
                    GESTIONE_STAZ_1(TIPO := TIPO_TRIGGER , TEMPO :=  T#0s , TRIGGER :=  PRESENZA_PEZZO, AZIONE := ALIMENTAZIONE);
                    // se si verifica l'edge negativo disattivo la stazione e conto il pezzo
                    IF EDGENEG(ALIMENTAZIONE) THEN
                        PRESENZA_PEZZO := FALSE;
                        STAZ_1 := FALSE;
                        C_MAN_IN := C_MAN_IN + 1;
                    END_IF;
                END_IF;
                // se è presente un pezzo in staz_2
                IF STAZ_2 THEN
                    // chiamo la funzione per la stazione 2
                    GESTIONE_STAZ_2(TIPO := TIPO_TEMPO , TEMPO :=  T#5s , TRIGGER :=  FALSE , AZIONE := SALDATURA_ON);
                    // se si verifica l'edge negativo disattivo la stazione
                    STAZ_2 := NOT EDGENEG(SALDATURA_ON);
                END_IF;
                // se è presente un pezzo in staz_3
                IF STAZ_3 THEN
                    // chiamo la funzione per la stazione 3
                    GESTIONE_STAZ_3(TIPO := TIPO_TEMPO , TEMPO :=  T#3s , TRIGGER := FALSE, AZIONE := FORATURA_ON);
                    // se si verifica l'edge negativo disattivo la stazione
                    STAZ_3 := NOT EDGENEG(FORATURA_ON);
                END_IF;
                // se è presente un pezzo in staz_4
                IF STAZ_4 THEN
                    // chiamo la funzione per la stazione 4
                    GESTIONE_STAZ_4(TIPO := TIPO_TEMPO , TEMPO :=  T#4s , TRIGGER :=  FALSE, AZIONE := AVVITATURA_ON);
                    // se si verifica l'edge negativo disattivo la stazione
                    STAZ_4 := NOT EDGENEG(AVVITATURA_ON);
                END_IF;
                // se è presente un pezzo in staz_5
                IF STAZ_5 THEN
                    // chiamo la funzione per la stazione 5
                    GESTIONE_STAZ_5(TIPO := TIPO_TRIGGER , TEMPO :=  T#0s , TRIGGER :=  QUALITY_CHECK_DONE, AZIONE := QUALITY_CHECK);
                    // se si verifica l'edge negativo disattivo la stazione e incremento il contatore finale
                    IF EDGENEG(QUALITY_CHECK) THEN
                        QUALITY_CHECK_DONE := FALSE;
                        STAZ_5 := FALSE;
                        C_MAN_OUT := C_MAN_OUT + 1;
                    END_IF;
                END_IF;
                // verifichiamo se l'ultimo pezzo è uscito dalla stazione
                IF C_MAN_OUT >= CILCI_MANUTENZIONE THEN
                        STATO := STATO_MANUTENZIONE;
                END_IF;
            END_IF;

        STATO_MANUTENZIONE:
            MAI := TRUE;
            IF MAIR THEN
                C_MAN_IN := 0;
                C_MAN_OUT := 0;
                MAI := FALSE;
                MAIR := FALSE;
                STATO := STATO_ACCESO;

            END_IF;
    END_CASE;
END_PROGRAM
```

Structured Text : C:\Users\emili\OneDrive - unibg.it\UNI\Anno III\Automazione Industriale\GitHub\Laboratorio-di-Automazione-Industriale\ST\Logical\Cosberg\GESTIONE_STAZIONE.st

```
1
2   (* function block per la gestione delle singole stazioni *)
3   FUNCTION_BLOCK GESTIONE_STAZIONE
4
5       AZIONE:= TRUE;
6
7
8       CASE TIPO OF
9           TIPO_TRIGGER:
10              // se è presente il fornte positivo di trigger
11              IF EDGEPOS(TRIGGER) THEN
12                  // segnalo che è avvenuto l'evento
13                  EVENTO:= TRUE;
14              END_IF;
15
16          TIPO_TEMPO:
17              //incremento del temporizzatore
18              TEMPORIZZATORE := TEMPORIZZATORE + DELTA_T;
19              // verifica superamento del tempo
20              IF TEMPORIZZATORE >= TEMPO THEN
21                  EVENTO:= TRUE;
22              END_IF;
23      END_CASE;
24
25      // gestione azioni se si verifica l'evento
26      IF EVENTO THEN
27          AZIONE:= FALSE;
28          TEMPORIZZATORE:= T#0s;
29          TRIGGER:= FALSE;
30          EVENTO := FALSE;
31      END_IF;
32
33  END_FUNCTION_BLOCK
34
35
```