Laboratorio di Sistemi Automatici

Selettore colori con IPC Beckhoff

Capolavoro

Autore:
Mascagni Lorenzo Maria

7 giugno 2024

Sommario

Questo documento descrive il progetto di sviluppo di un programma per un IPC (Industrial PC) per una linea di smistamento con rilevamento del colore. Verranno discusse le funzionalità principali, la struttura del programma e i vantaggi derivanti dall'implementazione di questo sistema. Inoltre, saranno fornite riflessioni finali e prospettive future per miglioramenti e aggiornamenti.

Indice

1 Introduzione

1.1 Presentazione del Progetto

Il progetto consiste nello sviluppo di un programma per un IPC della Beckhoff, un dispositivo utilizzato per automatizzare processi industriali. In particolare, si tratta di una linea di smistamento con rilevamento del colore, costituita da un kit della Fisher Technik, utilizzata per la separazione automatica di blocchi colorati. Un nastro trasportatore trasporta componenti geometricamente identici ma di colore diverso a un sensore di colore, dove vengono separati in base al loro colore.

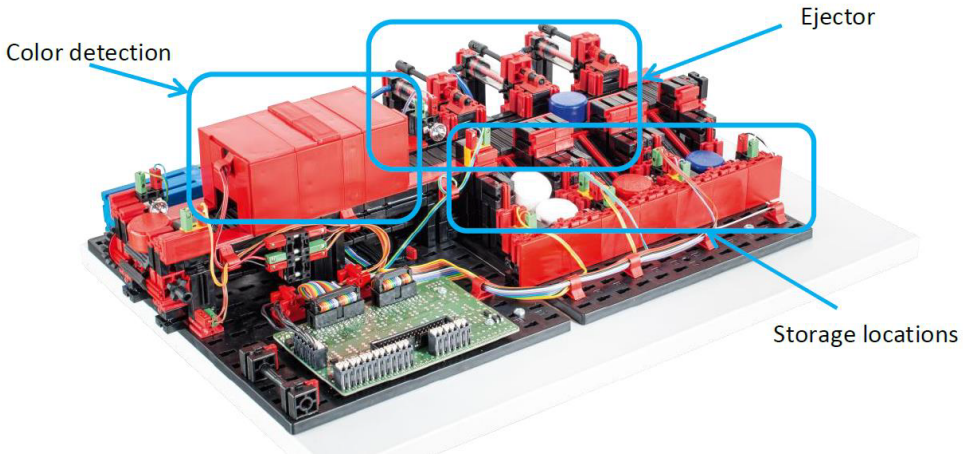


Figura 1: Linea di smistamento con rilevamento del colore

1.2 Obiettivi del Progetto

L'obiettivo principale del progetto è creare un programma per il controllo automatico del processo di smistamento basato sul colore. Una volta fatto ciò sarà possibile implementare nuove funzioni con lo scopo di migliorare il monitoraggio e il controllo del processo.

2 Struttura dell'impianto

2.1 Descrizione Generale

Il programma per IPC sviluppato offre una serie di funzionalità principali per il controllo e l'automazione della linea di smistamento. Il nastro trasportatore è alimentato da un motore a +24VDC, e il percorso di trasporto viene misurato con l'aiuto di un sensore a impulsi. L'espulsione dei pezzi viene gestita da cilindri pneumatici azionati da elettrovalvole, e diverse barriere fotoelettriche monitorano il flusso dei pezzi e il livello di riempimento dei luoghi di stoccaggio.

2.2 Dettagli dell'Impianto

Il progetto è stato concepito come due macchine separate che lavorano insieme in modo coordinato:

- **Sensore di Colore**: Il rilevamento del colore è gestito da un sensore ottico riflettente. Questo sensore è in grado di rilevare il colore dei blocchi in base alla riflessione superficiale, distinguendo i pezzi colorati in base a valori soglia tarati sperimentalmente. È fondamentale che il sensore sia protetto dalla luce ambientale e installato perpendicolarmente alla superficie degli oggetti.
- **Sistema di Espulsione**: L'espulsione dei pezzi è gestita da cilindri pneumatici monostabili, controllati da elettrovalvole a 3/2 vie. Quando un blocco attraversa la barriera luminosa, il sensore di colore determina il cilindro pneumatico da attivare per espellere il blocco verso il magazzino di stoccaggio appropriato. Questo sistema utilizza un sensore a impulsi per garantire che l'espulsione sia indipendente dalla velocità del nastro trasportatore.

Dividendo l'impianto in due macchine separate possiamo quindi semplificare in maniera significativa lo sviluppo del codice.
Per riassumere, l'impianto include:

- Controllo del nastro trasportatore tramite motore a +24VDC.
- Misurazione del percorso di trasporto con sensore a impulsi.
- Rilevamento del colore tramite sensore ottico riflettente.
- Espulsione dei pezzi gestita da cilindri pneumatici azionati da elettrovalvole.
- Monitoraggio del flusso dei pezzi e del livello di riempimento dei luoghi di stoccaggio con barriere fotoelettriche.

3 Codice

3.1 Struttura del Programma

Il programma principale è strutturato in due macchine a stati finiti subordinate che gestiscono singolarmente i due blocchi definiti in precedenza, all'interno di una principale che costituisce la logica principale di questo progetto e gestisce i sistemi di emergenza e ripristino. Ciò pemette una migliore manutenibilità e aggiornamento dell'intero impianto.

3.2 Tecnologie e Strumenti Utilizzati

Sono stati utilizzati i seguenti linguaggi di programmazione e software per lo sviluppo del programma:

- **Ambiente di Sviluppo**: TwinCAT, un potente software di sviluppo per prodotto da Beckhoff, che offre un'ampia gamma di strumenti per la programmazione, simulazione e debug.
- **Linguaggio di Programmazione**: Structured Text (ST), uno dei linguaggi definiti dallo standard IEC 61131-3, che permette di scrivere codice in forma strutturata e leggibile, simile ai linguaggi di programmazione ad alto livello come Pascal.
- **Strumenti di Simulazione**: Funzionalità integrate in TwinCAT per testare il programma in condizioni controllate prima della sua implementazione sul campo.

3.3 Fasi di Sviluppo

Il progetto si è svolto in diverse fasi, ciascuna delle quali ha comportato specifiche attività e risultati. Di seguito una descrizione dettagliata di ciascuna fase:

1. **Pianificazione e Analisi dei Requisiti**
   - **Requisiti Forniti**: Le specifiche del progetto sono state fornite dal mio professore durante una prova di laboratorio di quarta superiore a seguito del corso di programmazione di Beckhoff. Queste specifiche hanno delineato i requisiti funzionali e non funzionali del sistema, fornendo una base chiara per lo sviluppo.
   - **Analisi dei Requisiti**: Una volta ricevute le specifiche, è stata condotta un'analisi dettagliata per comprendere appieno le esigenze del progetto. Questa fase ha incluso la definizione dei criteri di successo.
   - **Documentazione dei Requisiti**: Tutti i requisiti sono stati documentati in modo strutturato, creando una guida di riferimento per le fasi successive dello sviluppo.
2. **Progettazione e Implementazione del Codice**
   - **Progettazione del Sistema**: Basandosi sulle specifiche dei requisiti, è stata creata una progettazione dettagliata del sistema. Questo includeva diagrammi di flusso, schemi elettrici e una suddivisione del programma in moduli e sottoprogrammi.
   - **Sviluppo del Codice in TwinCAT**: Utilizzando TwinCAT come ambiente di sviluppo e Structured Text (ST) come linguaggio di programmazione, è stato sviluppato il codice per il PLC. Il codice è stato scritto in modo modulare per facilitare la manutenzione e gli aggiornamenti futuri.
   - **Integrazione del Sistema**: Una volta completati i singoli moduli, sono stati integrati per formare il sistema completo. Sono stati effettuati test di integrazione per garantire che tutti i moduli funzionassero correttamente insieme.
3. **Testing e Debug**
   - **Testing Unitario**: Ogni modulo è stato testato singolarmente per verificare che funzionasse correttamente. Sono stati utilizzati casi di test specifici per coprire tutte le funzionalità e le condizioni limite.
   - **Simulazione e Test tramite TwinCAT**: Utilizzando la Run Mode di TwinCAT, il programma è stato testato in condizioni controllate per rilevare e correggere eventuali bug prima della messa in opera. Questo ha permesso di verificare il funzionamento del sistema.
   - **Debug e Ottimizzazione**: Durante il testing, sono stati identificati e corretti bug. Sono state inoltre apportate ottimizzazioni per migliorare le prestazioni del sistema.

3.4 Struttura della MSF

La struttura del programma di questo progetto si basa sulle nozioni apprese durante il seminario offerto dalla Beckhoff che ci ha permesso di capire come impostare il codice di una macchina a stati finiti in modo semplice, leggibile e di facile manutenibilità. Per prima cosa gli stati vengono definiti in un tipo dato apposito denominato generalmente `E_STEP`. La macchina a stati finita è composta da tre parti principali:

- **Inizializzazione della Macchina:** Tramite la verifica della variabile di tipo `BOOL`, `bInitialized`, inizializza la macchina nella sua condizione di avvio.
- **Preambolo:** Dove sono presenti controlli e letture di poarametri che devono essere sempre eseguite.
- **Switch-Case:** Tramite uno switch-case che controlla il valore della variabile di stato di tipo `E_STEP`, `eStep`, la macchina esegue una determinata logica contenuta all'interno di un singolo stato.

Il singolo stato è composto da tre parti:

- **Body:** Dove è presente la logica di funzionamento delle uscite.
- **Transizione:** Dove viene gestita la traniszione negli altri stati.
- **Exit Act:** Parte di codice che viene eseguita nella fase di uscita da questo stato.

Un esempio di questa tipologia di struttura è mostrato nell'appendice B

4 Implementazioni Aggiuntive

4.1 Filtraggio del Segnale Analogico

Il rilevamento del colore avviene facendo passare il pezzo sotto il sensore di colore, che produce una tensione analogica variabile tra 0V e +10V sulla linea I4. Per garantire una lettura precisa e stabile dei valori di tensione prodotti dal sensore, si utilizza un algoritmo di filtraggio basato su un filtro passa basso numerico. Questo approccio consente di attenuare i disturbi e le fluttuazioni casuali nel segnale, migliorando l'affidabilità delle misurazioni. L'algoritmo di filtraggio funziona come segue:

- Un Function Block (FB) viene richiamato periodicamente ogni T_c secondi dal Task Main per eseguire la lettura della tensione $v[n]$ fornita dal sensore.
- Il valore letto $v[n]$ viene utilizzato per aggiornare un valor medio $v_{mean}[n]$ mediante un filtro passa basso numerico, secondo la formula:

$$v_{mean}[n] = a \cdot v_{mean}[n-1] + (1-a) \cdot v[n]$$

- La costante a nel filtro determina la costante di tempo T del filtro passa basso, che controlla la velocità di risposta del filtro alle variazioni del segnale. La scelta di a dipende dal livello di attenuazione desiderato e dalla frequenza dei disturbi che si desidera filtrare.

- Il valore iniziale del valor medio $v_{mean}[0]$ può essere assegnato utilizzando una prima lettura della tensione $v[n]$ del sensore durante la fase di inizializzazione del Function Block. Questo assicura che il filtro parta da un valore realistico e rappresentativo del segnale.

- Durante l'operazione normale, il valore $v_{mean}[n]$ viene continuamente aggiornato ad ogni ciclo di lettura, producendo una versione smorzata del segnale di ingresso che riflette più accuratamente il livello di colore del pezzo sotto il sensore.

Il filtro passa basso numerico è essenziale per ridurre l'impatto delle variazioni rapide e dei picchi di rumore che possono essere introdotti da variazioni della luce ambientale o da altre interferenze. In questo modo, il sistema può rilevare con maggiore precisione il colore dei pezzi, migliorando l'accuratezza e l'affidabilità del processo di smistamento. Il codice di questa FB è mostrato nell'appendice B. Il risultato di questo processo è visualizzabile nella seguente immagine.

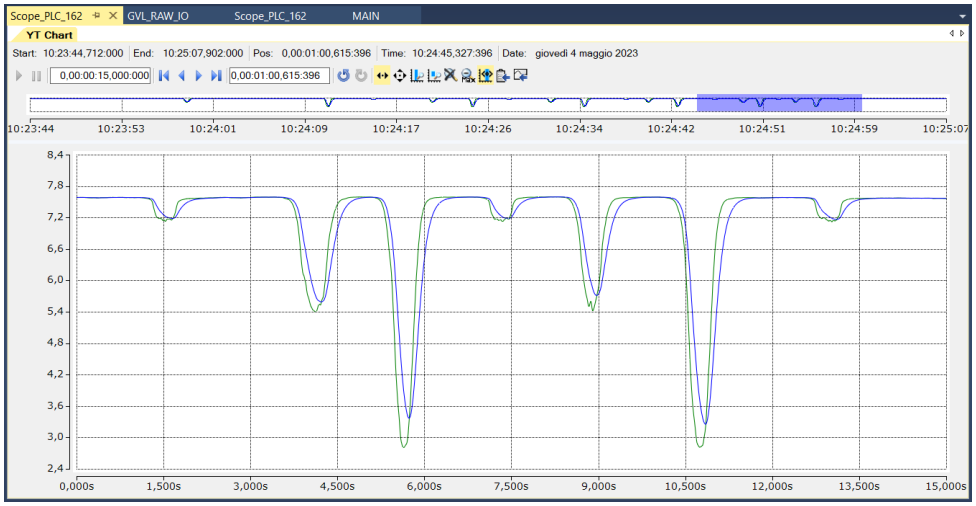


Figura 2: Vista dello Scope del segnale filtrato

Come si può osservare, il segnale filtrato presenta un lieve sfasamento rispetto a quello originale. Tuttavia, questo non rappresenta un problema, poiché il processo di misura avviene a una velocità inferiore.

4.2 HMI (Human-Machine Interface)

Il sistema include un'interfaccia HMI (Human-Machine Interface) sviluppata utilizzando TwinCAT. L'HMI offre un'interfaccia grafica intuitiva che permette agli operatori di monitorare e controllare il processo di smistamento in tempo reale. Le principali funzionalità dell'HMI includono:

- **Visualizzazione dello Stato del Sistema**: L'HMI mostra lo stato attuale della macchina astati finiti principale e delle due subordinate

- **Controllo Manuale**: Gli operatori possono interagire con il sistema, eseguendo comandi manuali per avviare o fermare l'impianto.

- **Settaggio dei Parametri**: L'HMI permette di impostare e regolare vari parametri operativi, come i valori soglia per il rilevamento del colore e i tempi di ritardo per

l'attivazione dei cilindri pneumatici. Questo consente agli operatori di ottimizzare le prestazioni del sistema in base alle esigenze specifiche del processo.

- **Allarmi e Notifiche**: In caso di anomalie o malfunzionamenti, l'HMI fornisce una lista con tutti gli errori di sistema per informare immediatamente gli operatori, facilitando la manutenzione dell'impianto.

Di seguito sono presenti le due pagine principali dell'HMI, ovvero quella per il controllo della macchina e quella per impostare i parametri.

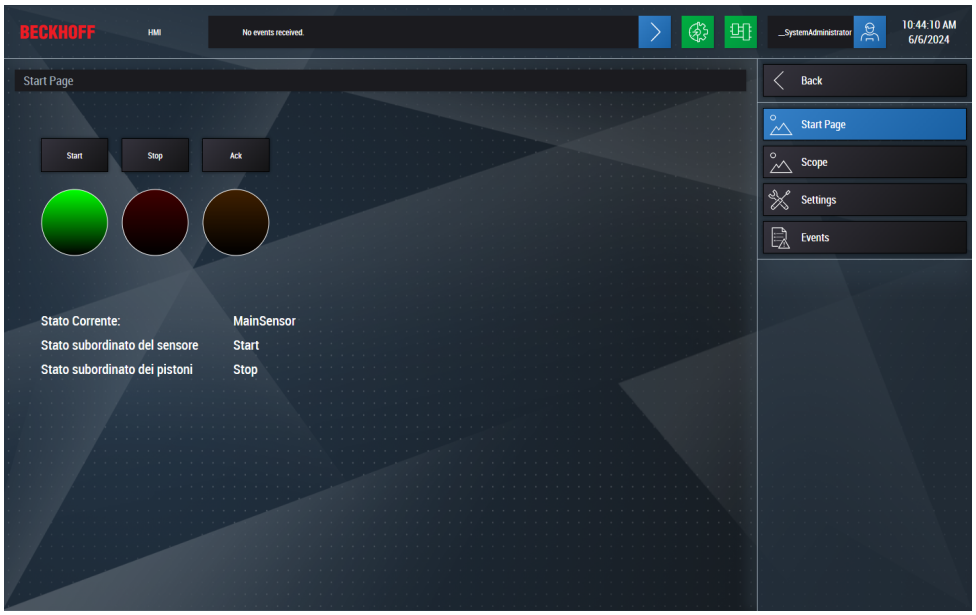


Figura 3: Pagina dell'HMI per il controllo dell'impianto.

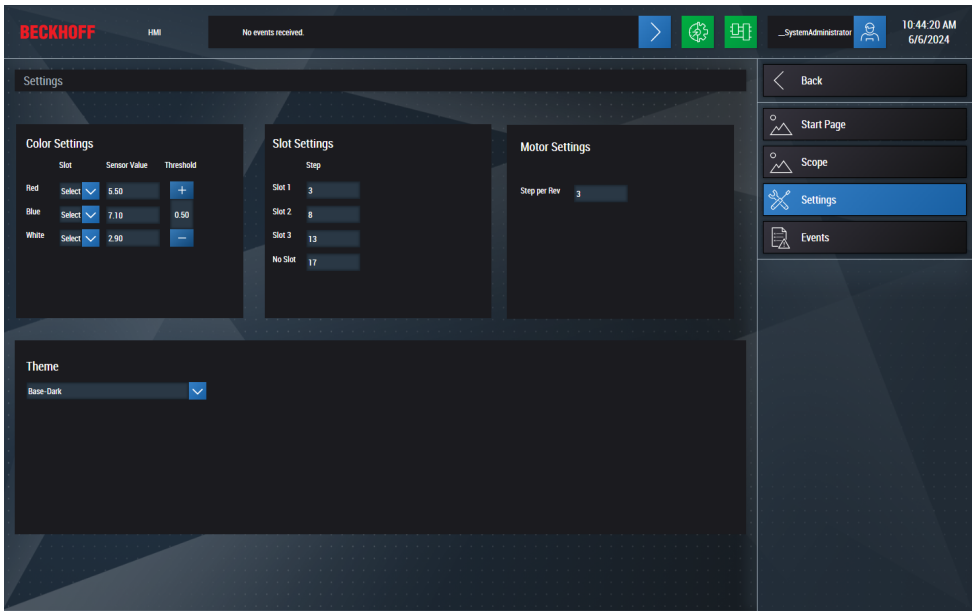


Figura 4: Pagina dell'HMI per impostare i parametri.

4.3 Pubblicazione dei Parametri Tramite MQTT

Per migliorare il monitoraggio e la diagnostica del sistema, è stata implementata una task che pubblica vari parametri operativi su un broker utilizzando il protocollo MQTT. Questa funzionalità consente la trasmissione in tempo reale dei dati operativi a un sistema di monitoraggio remoto: I dati pubblicati includono informazioni sul numero di cicli eseguiti, il valore in tempo reale del sensore per i colori e il numero di giri effettuato dal motore. ùi dati possono essere visualizzati iscrivendosi ai vari topic. nella seguente immagine è possibile vedere una schermata che dimostra la pubblicazione dei parametri.

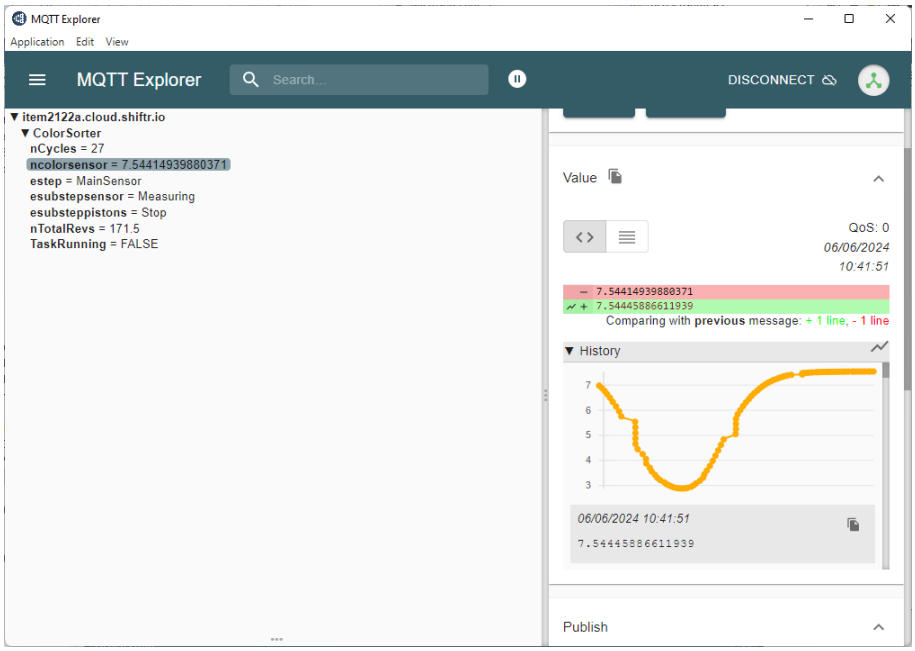


Figura 5: schermata di MQTTExplorer.

Questi dati possono essere monitorati da remoto tramite apposita dashboard realizzata con Node-Red, permettendo una supervisione continua del sistema.

Figura 6: Dashboard per la visualizzazione dei parametri.

5 Conclusione

5.1 Riflessioni Finali

Prima di tutto, l'unione tra il PLC Beckhoff e il kit Fischer Technik ha creato un sistema pratico ed efficiente, rispondendo concretamente alle esigenze di selezione e smistamento dei colori nelle linee di produzione. Questo sistema fornisce una soluzione affidabile e precisa, di grande importanza per le aziende che cercano di ottimizzare i processi produttivi. In secondo luogo, durante lo sviluppo di questo progetto ho avuto l'opportunità di approfondire le mie competenze nell'utilizzo del software TwinCAT, imparando a sfruttare al massimo le sue potenzialità. L'integrazione di funzionalità come HMI (Interfaccia Uomo-Macchina) e MQTT (Protocollo di messaggistica) ha arricchito il mio bagaglio di competenze tecniche, rendendomi più familiare con tecnologie fondamentali nel settore dell'automazione industriale. Infine, considerando l'intero percorso di sviluppo, il successo di questo progetto è stato possibile grazie non solo alle mie competenze tecniche, ma anche alla determinazione nel superare le sfide e al lavoro di squadra. Questa esperienza ha contribuito in modo significativo alla mia crescita professionale, preparandomi ad affrontare con fiducia le sfide future nel campo dell'automazione industriale e dell'ingegneria dei sistemi.

5.2 Futuri aggiornamenti possibili

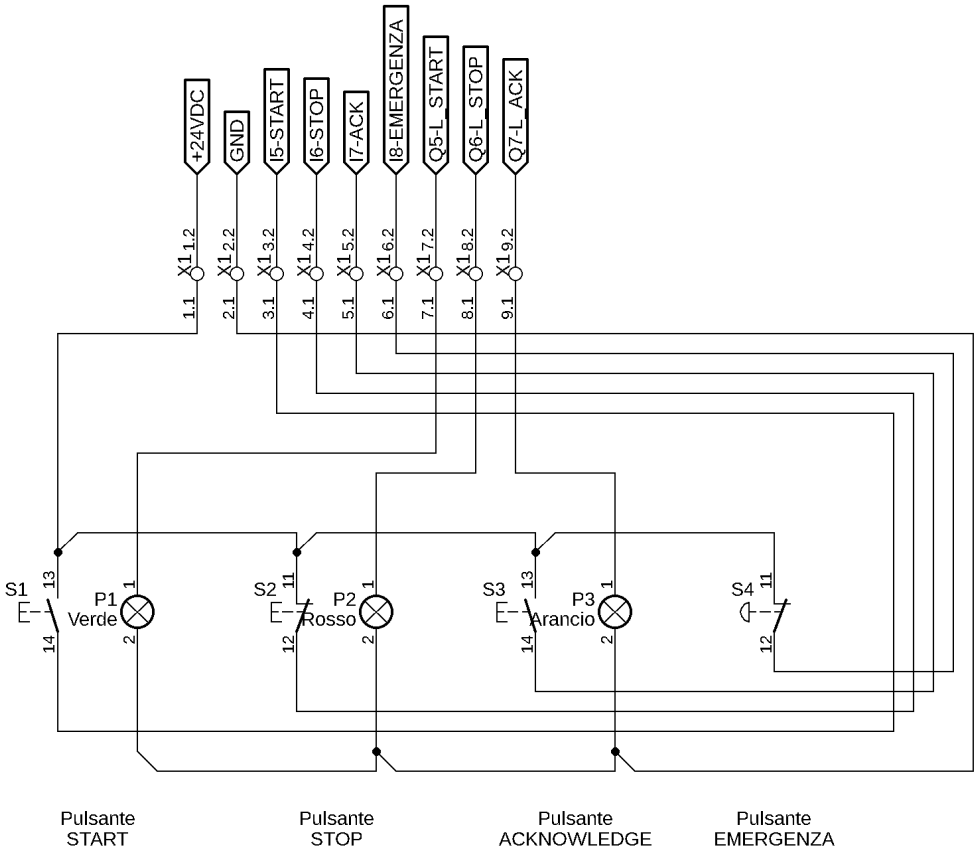
Per quanto riguarda i prossimi sviluppi del progetto, ci sono diverse aree che intendiamo esplorare per migliorare ulteriormente le funzionalità e le prestazioni del sistema. In primo luogo, prevediamo di concentrarci sull'ottimizzazione dell'interfaccia utente sia sull'HMI che sulle dashboard. Questo potrebbe includere un design più intuitivo e user-friendly, con grafici e controlli più chiari e facili da interpretare. L'obiettivo è rendere l'interazione con il sistema ancora più semplice ed efficiente per gli operatori. In secondo luogo, stiamo valutando l'implementazione di un sistema di diagnostica dei problemi più avanzato. Questo consentirebbe di rilevare e risolvere eventuali anomalie o guasti in modo più rapido ed efficace, contribuendo a ridurre i tempi di fermo e migliorare la manutenzione preventiva del sistema. Infine, stiamo considerando una riprogettazione della logica di base che gestisce lo smistamento dei pezzi. Questo potrebbe coinvolgere l'introduzione di algoritmi più sofisticati che consentono lo smistamento di più pezzi contemporaneamente, ottimizzando così i tempi di lavorazione e aumentando la capacità produttiva complessiva del sistema.

5.3 Ringraziamenti e Riconoscimenti

Nei ringraziamenti, desidero esprimere la mia gratitudine a Beckhoff Italia, e in particolare all'ing. Enricomaria Pavan, per la competenza e la capacità comunicativa dimostrate nei due seminari tecnici organizzati negli ultimi due anni scolastici. Ringrazio l'ing. Pavan per aver condiviso framework e modelli di programmazione che hanno costituito le fondamenta indispensabili per lo sviluppo di questo progetto. Inoltre, desidero ringraziare il Prof. Luca D'Amore per avermi concesso l'opportunità di dedicare tempo e risorse allo sviluppo di questo progetto. Il suo sostegno e la sua guida sono stati fondamentali per il successo del lavoro svolto.

A Schemi e Diagrammi

A.1 Pulsantiera di Comando



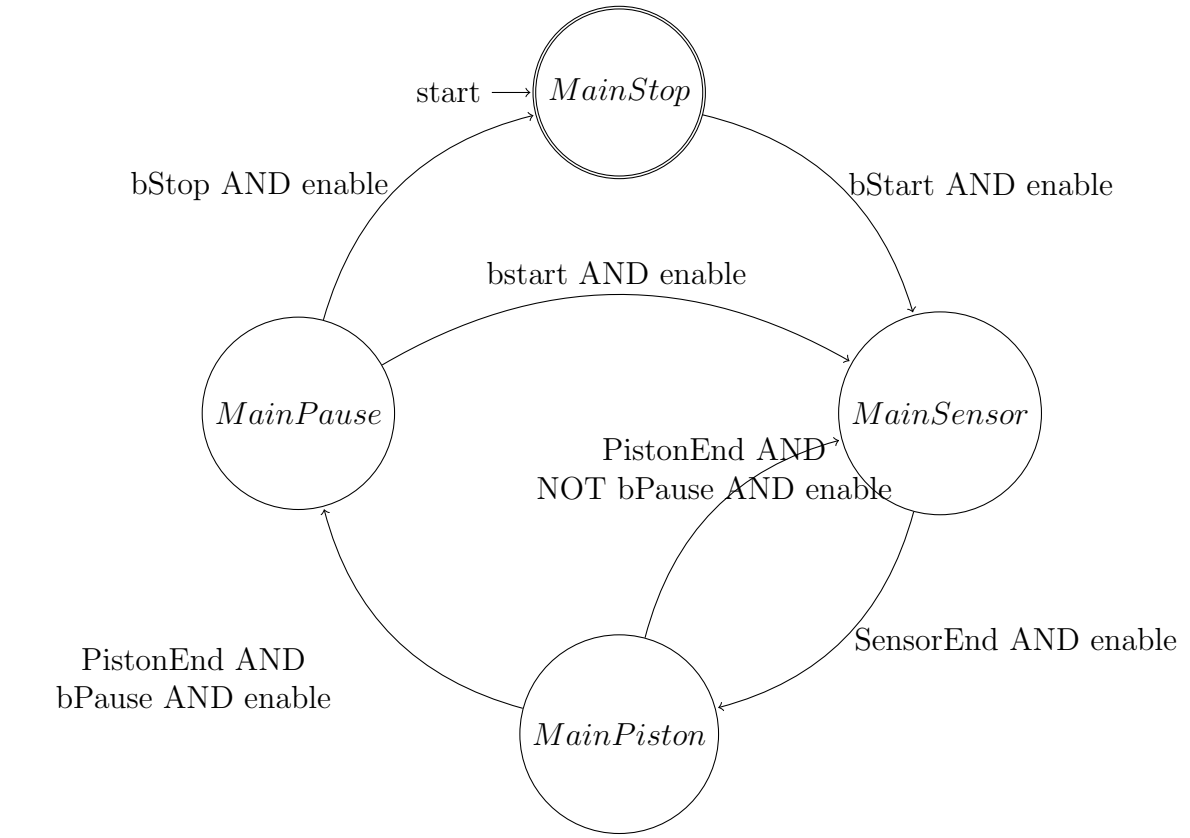
A.2 Mappa degli I/O

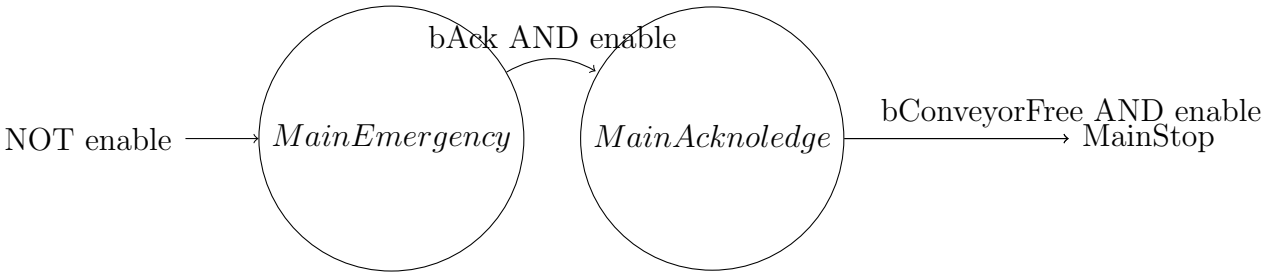
KIT Fischer Technik			Morsettiera IPC	
Terminale	Funzione	Input/Output	Terminale	Input/Output
1	Alimentazione attuatori	24V DC		Bus +24V
2	Alimentazione sensori	24V DC		Bus +24V
3	GND attuatori	GND		Bus GND
4	GND sensori	GND		Bus GND
5	Sensore a impulsi del nastro	I1 (NC)	EL1809	I1
6	Sensore presenza pezzo all'ingresso del selettore colori	I2 (NC)	EL1809	I2
7	Sensore presenza pezzo all'uscita del selettore colori	I3 (NA)	EL1809	I3
9	Uscita analogica sensore colori	I4	EL3004	A1 e il pin 2 a GND
10	Sensore magazzino 1	I5	EL1809	I5
11	Sensore magazzino 2	I6	EL1809	I6
12	Sensore magazzino 3	I7	EL1809	I7
17	Motore avanzamento nastro trasportatore	Q1	EL2809	Q1
18	Compressore	Q2	EL2809	Q2
20	Valvola pistone 1	Q3	EL2809	Q3
21	Valvola pistone 2	Q4	EL2809	Q4
22	Valvola pistone 3	Q5	EL2809	Q5

KIT Fischer Technik			Morsettiera IPC	
Terminale	Funzione	Input/Output	Terminale	Input/Output
	START	I9	EL1809	I9
	STOP	I10	EL1809	I10
	ACK	I11	EL1809	I11
	EMERGENZA	I12	EL1809	I12
	EMERGENZA con chiave	I13	EL1809	I13
	L_START	Q9	EL2809	Q9
	L_STOP	Q10	EL2809	Q10
	L_ACK	Q11	EL2809	Q11

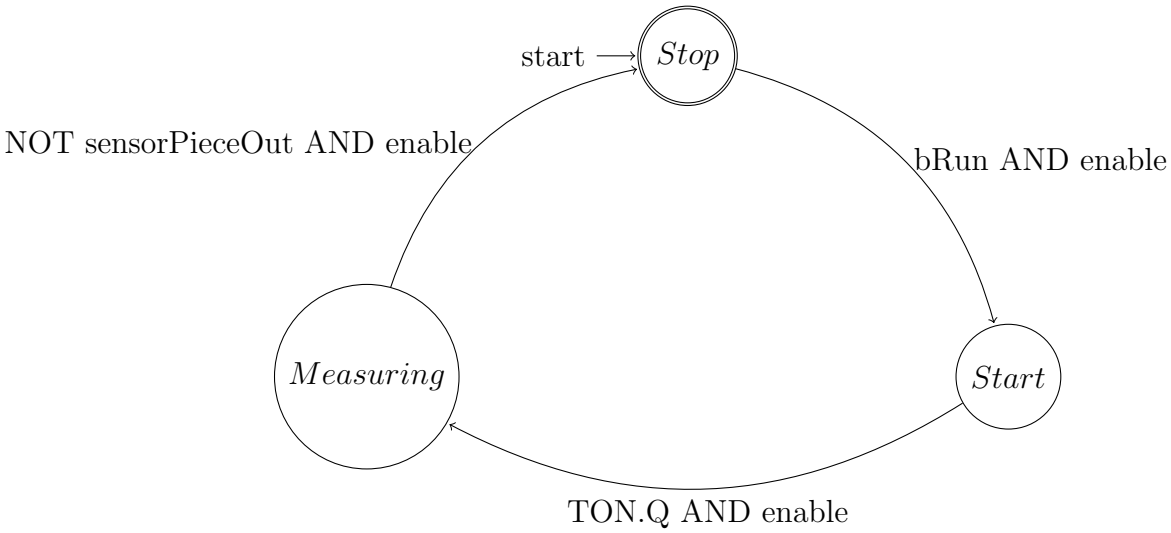
A.3 diagrammi degli stati

A.3.1 MSF della MAIN

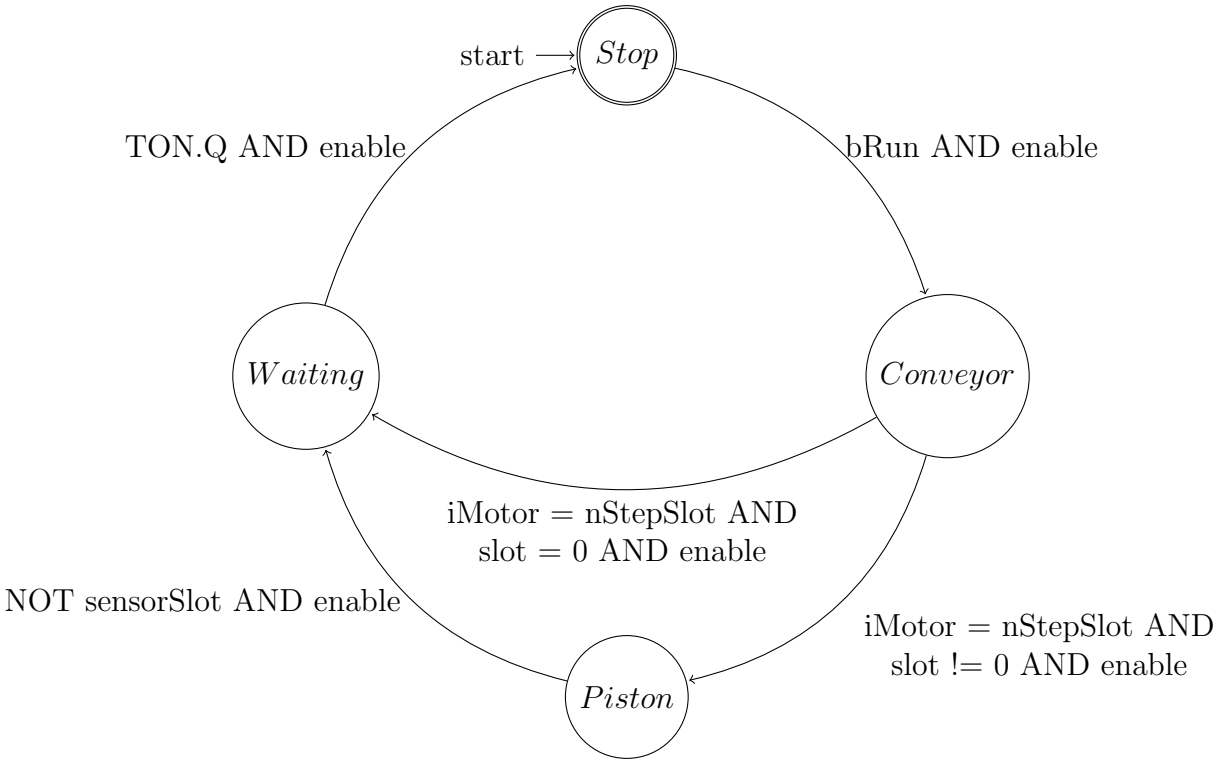




A.3.2 MSF del Blocco Sensore



A.3.3 MSF del Blocco Pistoni



B Codice Esempio

B.1 Struttura della MSF

B.1.1 E_STEP

```
{attribute 'qualified_only'}
{attribute 'strict'}
{attribute 'to_string'}
TYPE E_Step :
(
    (* special steps *)
    NoStep := -32767,  // No step
    Unknown := -32000, // unknown step

    (* Not Enabled or Errors*)
    NeOrError := -990, // Not Enabled or Errors

    (*MSF Steps*)
    //main steps
    MainStop         :=  10,    // stop
    MainSensor       :=  20,    // sensor area
    MainPiston       :=  30,    // piston area
    MainPause        :=  40,    // pause
    MainEmergency    :=  50,    // emergency
    MainAcknowledge  :=  60,    // acknowledge
    //sub-steps
    Stop             :=  90,
    Start            :=  100,
    Measuring        :=  110,
    Processing       :=  120,
    conveyor         :=  130,  // a new piece was placed on the conveyor belt
    waiting          :=  140,  // attesa
    piston           :=  150   // pistone
);
END_TYPE
```

B.1.2 Macchina a Stati Finiti

```
IF NOT bInitialized THEN
    eNewStep := E_Step.NoStep;
    eStep := E_Step.MainStop;
    bEntryAct := FALSE;
    bFbEnd := FALSE;
    bInitialized := TRUE;
    bFbSensorReset := TRUE;
    bFbPistonsReset := TRUE;
ELSE

    (*Preamble*)

    (*Forcing the Not Enabled State*)
    IF NOT bEnable THEN
        eNewStep := E_Step.NeOrError;
    END_IF

    (*Forcing the emergency state*)
    IF NOT stIn.bEmerg THEN
        IF eStep <> E_Step.MainEmergency THEN
            eEmergStep := eStep;
        END_IF
        eNewStep := E_Step.MainEmergency;
    END_IF

    ...

    (*Case*)
    CASE eStep OF

        (*MainStop Step - S0*)
        E_Step.MainStop :

        ...

    END_CASE

    (* elapsed time into the current step *)
    fbTonStep(IN := (eNewstep = E_Step.NoStep), PT := T#500H);
    tStepTime := fbTonStep.ET;
END_IF
```

B.1.3 Struttura dello Stato

```
(*MainStop Step - S0*)
E_Step.MainStop :
    (*Body*)
    bPause := FALSE;

    stOut.bLStart := fbOscillator.q;
    stOut.bLStop := TRUE;
    stOut.bLAck := FALSE;
    stOut.bMotor := FALSE;

    (*Transition - to MainSensor*)
    IF stIn.bStart AND bEnable THEN
        eNewStep := E_Step.MainSensor;
    END_IF

    (*Exit Act*)
    IF eNewStep <> E_Step.NoStep THEN
        IF eNewStep = E_Step.NeOrError THEN
            eErrStep := eStep;
            //nErrId := 16#9999_9999;
        END_IF
    END_IF
```

B.2 FB il Filtraggio del Segnale

```
VAR_INPUT
    VIn : REAL;
END_VAR
VAR_IN_OUT
    VOut : REAL;
END_VAR
VAR
    VPrev : REAL := 0;
    tau : REAL := 50E-3;
    Tc : REAL := 10E-3;
    A : REAL;
END_VAR

A := tau / (tau + Tc);
VOut := A * VPrev + (1 - A) * VIn;
VPrev := VOut;
```