UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA
DELL'AUTOMAZIONE

A.A. 2012/2013

Tesi di Laurea

Programmazione di PLC Omron per la gestione di Sistemi produttivi

RELATORE

Ing. Francesco Martinelli

CANDIDATO

Marco Cesaretta

Alla mia famiglia,

mio padre, mia madre e mio fratello

Indice

Ringraziamenti

Un ringraziamento particolare ai miei compagni di corso, un ottimo gruppo studio ma soprattutto una cerchia di amici.

Introduzione

Il termine automazione identifica la tecnologia che usa sistemi di controllo (come circuiti logici o elaboratori) per gestire macchine e processi, riducendo la necessità dell'intervento umano. Si utilizza per l'esecuzione di operazioni ripetitive o complesse, ma anche ove si richieda sicurezza o certezza dell'azione o semplicemente maggiore comodità ed autonomia. A partire dalla rivoluzione industriale sono stati richiesti, agli ingegneri e alle industrie degli enormi investimenti nell'automazione: la macchina a vapore prima, il motore a scoppio e l' elettronica dopo, hanno permesso il raggiungimento di notevoli progressi tecnologici.

Questo lavoro si inserisce in un contesto di automazione industriale, ed in particolar modo all'interno di stabilimenti produttivi. E' proprio all'interno delle fabbriche e delle catene di montaggio che il PLC occupa il maggiore impiego. Con l'evoluzione dei mercati nasce contemporaneamente la necessità di ottimizzare la capienza dei magazzini e la quantità di pezzi prodotti.

Al fine di ottenere una completa conoscenza del PLC, usato non solo come strumento di controllo per macchine, ma anche come gestore della catena di montaggio, si è voluto implementare su di esso leggi di controllo ottimo per la gestione di siti produttivi con diverse caratteristiche. Fase importante del progetto, alla base dell'evoluzione della tecnologia moderna, è stata la comunicazione tra dispositivi di diversa natura. Infatti i processi sono stati simulati su MatLab e controllati on-line tramite

il PLC.

In questa tesi viene esposto il lungo lavoro di ricerca effettuato. L'elaborato si suddivide in cinque capitoli, illustrando nel primo capitolo le caratteristiche principali del PLC utilizzato, non da sostituirsi al manuale, ma un'ottima guida generale per chi muove i primi passi all'interno del mondo dei Programmable Logical Controller. Nel secondo è stata approfondita la comunicazione seriale dei PLC Omron, nonché la struttura dei segnali e la loro interpretazione. Nel terzo e quarto capitolo si analizzano i sistemi produttivi oggetto dello studio, in particolar modo nel quarto capitolo vengono analizzati i controlli e i risultati ottenuti dalle simulazioni, con un accenno di come potrebbe essere continuato questo progetto nel quinto capitolo.

Capitolo 1

PLC

Il PLC è un dispositivo programmabile per il controllo di un processo o un sistema. Il sistema di controllo rivece in ingresso una serie di segnali e comanda, in base ad una elaborazione dei segnali di ingresso e allo stato attuale, le uscite di comando (attuatori). L'elaborazione dei segnali viene fatta in base ad un programma contenuto nell'area di memoria programma del PLC.

1.1 Introduzione

Nasce come elemento sostitutivo della logica cablata elettronica e dei quadri di controllo a relè e si è qualificato da tempo come elemento insostituibile nell'automazione di fabbrica. Acronimo di Programmable Logic Controller è l'elemento base del sistema di controllo di macchine e processi industriali. Un sistema di controllo e di misura di un processo industriale può essere descritto come un insieme di dispositivi interconnessi e comunicanti tra loro attraverso una o più reti di comunicazione.

A differenza dei normali pc, è progettato per lavorare in ambienti difficili, quali sono quelli industriali, dove è facile trovare alte temperature, elevato grado di umidità, disturbi elettrici, vibrazioni e sostanze aggressive, mentre proprio come i pc è dotato di particolari dispositivi con i quali è in grado di dialogare (pulsanti, sensori, azionamenti ed apparecchiature elettriche in generale).

La logica di controllo nei PLC è realizzata tramite una serie di istruzioni che costituiscono il programma (software) che viene installato nella memoria di programma presente nell'apparecchiatura (hardware) durante la fase di programmazione.

Quando successivamente il PLC è chiamato a realizzare l'automazione eseguendo il programma, le informazione che provengono dall'impianto arrivano sugli ingressi (input) e vengono lette; quindi, in relazione alle istruzioni presenti nel programma, sono attivate le uscite (output).

1.2 Struttura di un PLC

In prima approssimazione i PLC possono suddividersi in due grandi categorie: **compatti** e **modulari**. I primi sono praticamente monoblocco, in cui tutte le parti neces-

sarie per il funzionamento sono incluse in un unico case. Questi riducono la flessibilità del PLC che è permessa solo tramite l'aggiunta di un'espansione I/O monoblocco. I secondi invece sono realizzati in funzione delle reali necessità dell'utente, tramite blocchi unimodulari che vengono assemblati tra loro, i quali svolgono funzioni dedicate. Qualunque sia la struttura, in un PLC si possono distinguere i seguenti blocchi funzionali:

- CPU
- Memoria
- Interfaccia I/O
- Interfaccia di comunicazione

1.2.1 CPU

E' il *cervello* dove avvengono le elaborazioni matematiche e la gestione del PLC stesso. In questo modulo sono presenti i microprocessori, che possono essere uno o più, ed una certa quantità di memoria.

La differenziazione tra le varie CPU avviene tramite la valutazione di determinate caratteristiche quali: la quantità di memoria di programma, la quantità di memoria dati, il numero e il tipo di funzioni integrate disponibili e il numero di interfacce gestibili. Il microprocessore dialoga con le altre parti del PLC mediante collegamenti elettrici che prendono il nome di bus dati, bus indirizzo e bus di controllo. In particolare il *bus dati* consente di trasferire i dati tra le varie parti che compongono il PLC,

il *bus indirizzi* consente di individuare una determinata locazione di memoria nella quali si vuole scrivere o leggere, ed infine il *bus di controllo* consente di trasportare i segnali di controllo necessari per il funzionamento di tutto il PLC.

1.2.2 Memoria

Contiene tutte le informazioni utili alla CPU per poter lavorare e si divide in due grandi Aree: *Memoria Dati* e *Memoria Programma*. La memoria programma contiene la sequenza di istruzioni di programma, mentre la memoria dati contiene i dati da elaborare.

Questa memoria è normalmente di tipo RAM ed è volatile, realizzata con tecnologia a semiconduttore e può essere sia usata in scrittura che in lettura. Viene quindi usata una batteria tampone per mantenere le informazioni memorizzate quando viene a mancare l'alimentazione principale. Contrariamente, la memoria di tipo ROM può essere solamente letta e non è volatile bensì ritentiva (non perde il suo contenuto al mancare dell'alimentazione elettrica). Le memorie ROM vengono utilizzate dai costruttori per memorizzare il software di base (sistema operativo) che determina le caratteristiche del PLC. Tale software, per altro non modificabile, è indispensabile per il funzionamento del PLC; è sulla base di questo software che il programmatore realizza i propri applicativi. Alcuni tipi di PLC presentano la possibilità di scrivere su di una memoria Flash (memory card) di tipo non volatile, spesso viene utilizzata per memorizzare dati, parametri, programmi, commenti, simboli, ecc... e non necessita di batteria tampone per il salvataggio dei dati.

1.2.3 Interfaccia I/O

L'interfaccia Input è quella parte di Hardware che permette il collegamento del PLC con parti esterne per poterne valutare lo stato e le condizioni operative come sensori, trasduttori, contatti, ecc..., mentre quella Output permette di gestire dispositivi in grado di comandare attuatori come ad esempio contattori, elettrovalvole, azionamenti elettronici, ecc...

Questa interfaccia permette in pratica di convertire, nei moduli di input, il segnale fisico (elettrico) in un segnale logico (bit), mentre realizza la trasformazione opposta nei moduli di output; i segnali logici viaggiano all'interno del PLC sul bus dati.

1.2.4 Interfaccia di comunicazione

Con l'evoluzione della tecnologia è nata sempre più l'esigenza di far comunicare tra loro dispositivi di diversa natura. Ad oggi il PLC permette di comunicare con terze parti sia via etere che via cavo usando due tra gli standard di comunicazione più noti ed impiegati: comunicazione seriale e TCP/IP. Nella maggior parte delle CPU è presente una porta seriale standard, inoltre possono essere aggiunti dei moduli dedicati alla comunicazione sia RS232 che Ethernet.

1.3 Logica di controllo

L'elaborazione dei dati di un PLC avviene in modo sequenziale secondo l'ordine delle istruzioni all'interno del programma. L'attuazione (segnale fisico sull'uscita, rilevamento dell'ingresso) avviene in momenti precisi e non durante l'elaborazione del

programma. Possiamo considerare la logica di controllo di un PLC suddivisa in tre momenti:

1. Acquisizione dei dati dai sensori
2. Elaborazione del programma
3. Attuazione delle uscite

Questo tipo di funzionamento implica l'impossibilità di rilevare, se non con speciali funzioni o moduli, la variazione di un segnale che avviene durante l'elaborazione del programma.

Nella **prima fase** viene acquisita, in un preciso istante, la condizione degli ingressi (fotografia degli ingressi fisici) e fatta una copia nell'area di memoria di I/O. Nella **seconda fase** vengono elaborati gli ingressi e modificate le uscite, solo all'interno dell'area di memoria del PLC a loro dedicata, in modo sequenziale secondo l'esecuzione del programma.

La **terza fase** è una copia dell'area di memoria, che contiene le uscite, nel buffer delle schede di uscita (attuazione del segnale logico elaborato o rinfresco degli I/O). Le uscite a loro volta alimentano questi dispositivi (contattori ed elettrovalvole) che comanderanno gli attuatori (motori elettrici e cilindri pneumatici).

L'immagine del processo è lo stato logico dell'area di memoria corrispondente agli I/O fisici. In particolare la fotografia degli ingressi ad ogni acquisizione e le uscite elaborate prima dell'attuazione.

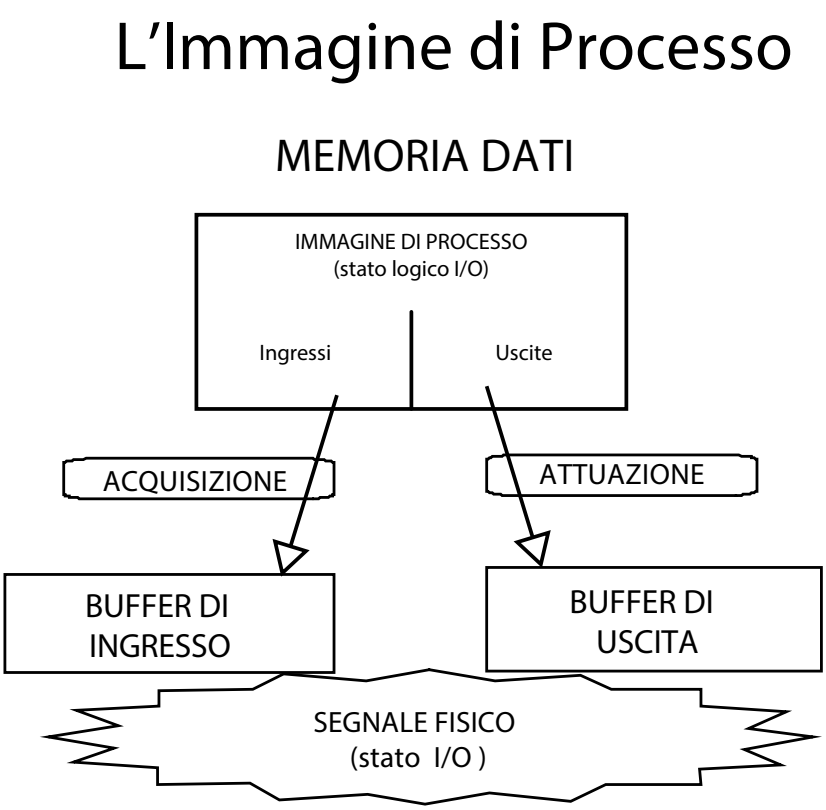


Figura 1.1: Elaborazione dei dati da parte di un PLC

Il segnale fisico gestito dall'interfaccia di I/O viene memorizzato in un'area di buffer per poter essere disponibile al momento del rinfresco degli ingressi e delle uscite(I/O refresh).

Viene definito **tempo di scansione** il tempo che intercorre tra un I/O refresh ed il successivo. Risulta quindi essere composta dalla somma dei seguenti tempi:

- tempo di elaborazione
- tempo di I/O refresh
- tempo di gestione dei processi comuni

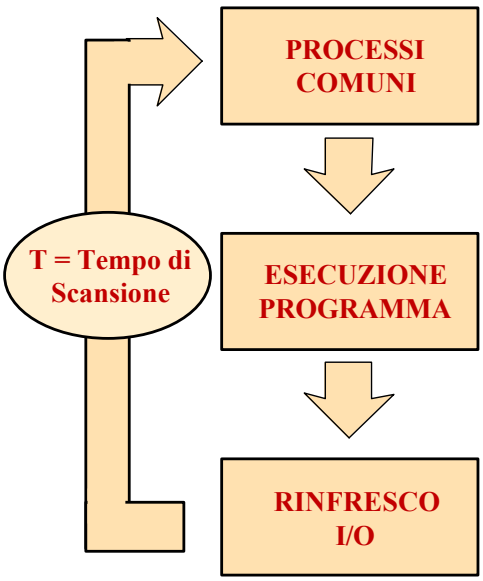


Figura 1.2: Tempo di scansione

- tempo di gestione delle periferiche

Il segnale elettrico interpretato dall'interfaccia di I/O del PLC viene letto ad ogni scansione e non in modo continuato.

Ciò implica che un segnale che ha una variazione più breve di un tempo di scansione potrebbe non essere interpretato.

Per avere la sicurezza che un segnale fisico sia rilevato dal PLC, cioè che nella memoria del PLC si percepisca la variazione di un ingresso, è necessario che il tempo in cui il segnale è presente fisicamente sull'ingresso sia superiore al tempo si scansione.

Il segnale fisico e il segnale logico del PLC sono coincidenti tra loro solo nel momento del rinfresco degli I/O. Dopo il rinfresco, durante l'esecuzione del programma. il segnale logico varia secondo la sequenza delle istruzioni e il segnale fisico non viene

rilevato.

1.4 Codifica dei dati

L'area di memoria di un PLC è costituita da un insieme di celle composte da 16 elementi di memoria elementare che possono assumere il valore 0 oppure 1 (ON, OFF).

Chiamiamo l'elemento di memoria elementare **bit**, la composizione di 4 bit affiancati **nibble(digit)**, 8 bit **byte** ed infine 16 bit, ovvero un'intera cella di memoria **word**.

Il contenuto di ogni word può essere interpretato in modo da avere una corrispondenza di tipo fisico. Un numero, un segnale analogico, un carattere alfanumerico o una configurazione di segnali elettrici ON/OFF, sono memorizzati nel PLC in aree di memoria formata tutte da 16 bit affiancati.

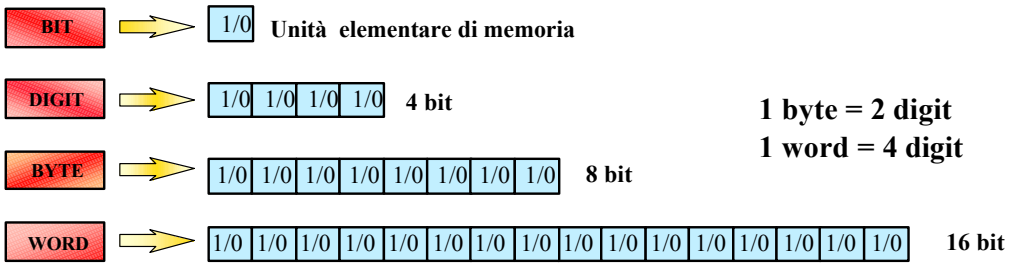


Figura 1.3: Celle di memoria e suddivisione

La corretta interpretazione del contenuto dell'area di memoria permette di "ricostruire" l'informazione che si vuole.

1.5 Ingressi e uscite digitali e analogiche

Il PLC gestisce informazioni lette dai sensori e le elabora per fornire comandi per gli attuatori.

Ogni segnale rilevabile contiene una serie di informazioni che poi possono essere utilizzate all'interno del programma per elaborare le uscite da attuare.

La comunicazione del PLC con il mondo esterno avviene principalmente scambiando informazioni attraverso segnali fisici (elettrici) che vengono interpretati dai due dispositivi in comunicazione allo stesso modo. I segnali elettrici possono essere di tipo binario (ON/OFF), analogico (segnale in tensione o in corrente), ad impulsi (conteggi veloci o uscite a treno di impulsi), o di interfacciamento seriale con livello fisico RS232C, RS422A, RS458.

Nello scambio di informazioni tra PLC e altri dispositivi la prima cosa da verificare è che i due dispositivi "possano parlare" tra loro con lo stesso linguaggio e con lo stesso tipo di segnale fisico.

1.5.1 Segnale digitale - in

Il segnale digitale è un segnale di tipo ON/OFF che nella memoria dei PLC viene interpretato come modifica di un bit 0/1 di un canale di memoria.

Ad ogni segnale digitale viene unicamente assegnata un'area di memoria del PLC in cui viene trasferita l'informazione.

Il livello fisico del segnale digitale è un segnale in tensione. L'interfaccia fisica di ingresso del PLC deve essere in grado di rilevare il segnale.

Quando il contatto collegato ad un determinato ingresso è **aperto**, il livello di tensione sul relativo morsetto ad un determinato ingresso è uguale a **0V** (stato logico

0); viceversa, quando il contatto è **chiuso**, il livello di tensione è uguale al valore previsto per quel modulo, per esempio **24V** (stato logico **1**).

Normalmente, i segnali digitali rilevabili sono segnali in tensione a $5V$, $12V$ o $24V$ in corrente continua (DC), $10 \div 16V$, $115V$ o $230V$ in corrente alternata (AC).

In Corrente continua è necessario avere informazioni sulla logica del segnale (se il segnala è di tipo PNP o di tipo NPN), ovvero se il comune è, rispettivamente, negato (logica negativa) o positivo (logica positiva). Alcuni moduli sono in grado di lavorare con entrambe le polarità.

Nei moduli di ingresso, al fine di garantire un funzionamento sicuro del PLC, è di particolare importanza la creazione dell'isolamento galvanico tra i circuiti interni e i dispositivi posti sull'impianto.

Tale separazione elettrica viene realizzata inserendo nei moduli di ingresso degli accoppiatori ottici o fotoaccoppiatori che, mediante l'uso di uno o due diodi LED e di un fototransistor, inseriti in un unico contenitore, consentono di trasferire l'informazione logica tra i due circuiti mediante la luce, ma anche di garantire l'isolamento elettrico, che può arrivare ad alcune migliaia di volt.

Mentre non si pongono problemi per i collegamenti ad apparecchiature con contatti elettromeccanici, in quanto non polarizzati, particolare attenzione richiedono invece i sensori (interruttori di prossimità) induttivi, capacitivi, ultrasonici e le fotocellule aventi un'uscita statica (a transistor) dato che necessitano di collegamenti omologhi; ad esempio, i sensori con uscita di tipo NPN potranno essere collegati solo ad ingressi di PLC tipo NPN.

Gli ingressi digitali possono essere utilizzati anche per inserire valori numerici mediante l'uso di preselettori che utilizzano segnali elettrici codificati, ad esempio mediante il codice BCD.

Per soddisfare le esigenze di collegamento con i più svariati dispositivi di input, i costruttori hanno reso disponibili sul mercato una gamma di moduli con un numero di ingressi che varia normalmente da 8 a 32, con valori, polarità e tipi di tensione indicati precedentemente.

1.5.2 Segnale digitale - out

Il comando digitale di un'uscita dal PLC deve essere tale da poter essere rilevato dal dispositivo collegato e deve essere adatto al tipo d'ingresso del dispositivo da comandare (contattore, elettrovalvola, lampada di segnalazione, ecc...) che a sua volta può agire sull'attuatore vero e proprio (motore elettrico, cilindro pneumatico, ecc..).

Ogni tipo di uscita ha le sue caratteristiche fisiche. La scelta adeguata dell'uscita permette di migliorare l'interfacciamento e la comunicazione. Le uscite di tipo digitale possono essere a relè, a transistor, a Triac e a MOSFET.

Nel caso di uscite a relè, il PLC comanda un relè che ha un contatto di uscita che può sopportare correnti di una certa entità e segnali in tensione sia continua sia alternata. Un'uscita a relè ha un numero massimo di operazioni possibili, dovuto alla vita del contatto meccanico, e una frequenza massima di commutazione limitata.

Qualora il contatto del relè alimenti un carico ohmico-induttivo,come la bobina di un'elettrovalvola o di un contattore, è necessario collegare in parallelo al carico stesso un gruppo soppressore di disturbi costituito da un gruppo RC o un varistore se si lavora in corrente continua o alternata, oppure un diodo se si lavora in corrente continua. In questo modo si limitano le sovratensioni che nascono all'apertura del contatto che danneggiano il contatto stesso e generano disturbi elettromagnetici.

Qualora l'uscita sia a transistor (a collettore aperto), il PLC comanda un transistor di uscita. L'entità della corrente sopportabile è dell'ordine delle centinaia di mA (normalmente $500mA$). Il transistor può sopportare solo tensioni continue e polarizzate. La scelta del tipo di uscita a transistor (NPN o PNP) deve essere fatta in base al tipo di ingresso che si deve comandare (NPN o PNP).

Sono disponibili, inoltre, uscite statiche a Triac, che supportano segnali solo in tensione alternata (da 100V a 240V AC) e uscite a MOSFET realizzate usando dei relè statici che sono in grado di lavorare con tensioni sia continue sia alternate.

Le uscite a relè consentono di realizzare il disaccoppiamento elettrico tra i circuiti interni e quelli esterni al PLC mediante il relè stesso, mentre per le uscite a semiconduttore è necessario l'inserimento nel circuito di un'uscita di un fotoaccoppiatore, in modo analogo a quanto visto per i moduli di ingresso.

Anche in questo caso i costruttori hanno reso disponibili sul mercato una gamma di moduli con un numero di uscite che varia normalmente da 8 a 32, con valori, polarità e tipi di tensione indicati precedentemente per i moduli di ingresso.

1.5.3 Segnale analogico

Il segnale analogico fornisce un'informazione di tipo complesso. Il livello di tensione o di corrente in ingresso viene convertito in un valore digitale (valore scalarizzato) in un canale (word) del PLC in base alla risoluzione della scheda, mediante un apposito circuito denominato convertitore analogico/digitale DAC (A/D), così come un valore contenuto in un canale del PLC viene convertito in un valore di tensione o corrente in uscita mediante un circuito denominato convertitore digitale/analogico ADC (D/A).

La risoluzione, la precisione e il tempo di aggiornamento del valore convertito della scheda sono le caratteristiche che identificano la bontà dell'interfaccia.

Più è alta la risoluzione, espressa in bit, maggiore è la sensibilità con cui viene rilevato il segnale analogico; ad esempio, su un segnale analogico $0 \div 10V$, una risoluzione di 13 bit ($2^{13} = 8129 punti$) permette di rilevare la variazione del segnale con una sensibilità di $1,2mV$.

La seconda caratteristica è la precisione, espressa in percentuale del valore di fondo scala, che indica l'errore massimo di conversione della scheda dovuto alla tolleranza dei componenti.

La terza caratteristica fondamentale è il tempo di aggiornamento del valore convertito espresso in μs o ms. Questa caratteristica indica la capacità di inseguire una variazione del segnale di ingresso.

In seguito ad una variazione di segnale, maggiore è la velocità di aggiornamento, minore è l'errore tra il valore convertito e il valore del segnale.

Quarta caratteristica identificativa della qualità della scheda riguarda i livelli di segnali con cui può interfacciarsi.
Normalmente i segnali sono in tensione o in corrente (continua):

- segnali in tensione $0 \div +10V$, $-10 \div +10V$, $0 \div +5V$, $-5 \div +5V$
- segnali in corrente $0 \div +20mA$ o $4 \div +20mA$

Sono previsti a catalogo moduli di ingresso e di uscita analogici in numero variabile da 2 a 10. La presenza di più ingressi o uscite viene praticamente realizzata inserendo nel modulo analogico di ingresso un circuito denominato multiplexer (MUX) che

commuta in successione, e uno alla volta, gli ingressi su un unico convertitore A/D; in questo modo si riesce a realizzare un modulo più economico rispetto ad uno che avrebbe un convertitore per ogni ingresso.

Analogamente, nei moduli di uscita analogici viene impiegato un circuito demultiplexer (DX) che fornisce alle uscite il segnale elettrico proveniente dal convertitore D/A.

Sono altresì disponibili moduli, per la misura di temperatura, in grado di lavorare con termoresistenze e termocoppie.

La scheda di **ingresso analogica** converte un valore di tensione o corrente in ingresso in un valore scritto in un'area di memoria del PLC. L'**uscita analogica** è una interfaccia che converte un valore scritto in un'area di memoria del PLC in un valore di tensione o corrente di uscita.

1.6 Moduli speciali

Oltre ai moduli ON/OFF e analogici, visti in precedenza, i PLC possono essere dotati di moduli speciali, molto spesso con un proprio microprocessore dedicato, per risolvere particolari problemi di automazione.

Ad esempio, è possibile trovare nei cataloghi dei costruttori i seguenti moduli:

- modulo conteggio veloce;
- modulo con uscita a treno di impulsi;
- modulo controllo assi;

- modulo di comunicazione;
- modulo master per bus di campo o network (DeviceNet, Profibus, Ethernet, ecc...);

Un segnale di tipo impulsivo normalmente viene utilizzato per comunicare con un azionamento o un encoder. Le informazioni ricavabili o trasferibili da un segnale impulsivo sono:

- frequenza tra un impulso ed il successivo;
- numero di impulsi;
- sfasamento tra le fasi di un segnale;
- segnale di reset.

Queste informazioni permettono di gestire dei posizionamenti completi. Tutte le informazioni vengono registrate in aree di memoria dedicate del PLC.

Il contatore veloce è l'interfaccia per l'encoder.

L'uscita a treno di impulsi è l'interfaccia per l'attuatore (servo motore o motore passo-passo), il tipo di segnale fisico è normalmente dello stesso tipo di quello di un encoder.

L'informazione è fornita su tre segnali: fase A, fase B e fase Z. Le fasi A e B danno informazioni sulla velocità di rotazione, spazio e direzione di rotazione. La fase Z, invece, indica quando l'asse dell'encoder compie un giro completo e normalmente

viene utilizzato per la ricerca dell'origine in sistemi di movimentazione o per realizzare il conteggio dei giri effettuati da un albero (contagiri).

1.7 Moduli di comunicazione

La trasmissione dei dati può avvenire, in generale, in modo parallelo o seriale. In campo industriale normalmente viene preferita la trasmissione seriale, che pur essendo più lenta rispetto a quella parallela, consente la comunicazione anche a distanze elevate. Di solito viene utilizzata la trasmissione seriale denominata asincrona in quanto risulta essere più economica; infatti tale trasmissione necessita di circuiti e software di gestione meno complessi e costosi.

Una trasmissione seriale è caratterizzata dalla velocità di trasmissione e ricezione dei dati ed ha come unità di misura il bps (bit per second)che rappresenta il numero di cambiamenti di stato logico del segnale nell'unità di tempo.

La comunicazione seriale permette uno scambio di informazioni tra dispositivi. Perché sia possibile la comunicazione tra due dispositivi, è necessario che le porte di comunicazione abbiano lo stesso supporto fisico (normalmente lo standard RS232C o RS422/RS485), lo stesso formato dei dati e, infine, che la comunicazione avvenga con lo stesso protocollo. Il protocollo rappresenta quell'insieme di regole necessarie per lo scambio di informazioni.

1.7.1 RS232C

Questo standard si applica al trasferimento dei dati seriali fino a 19200 bps, ad una distanza massima di circa $15 \div 20$ metri. Questo limite è però in funzione delle

caratteristiche elettriche del cavo e della velocità di trasmissione, e può essere spesso superato senza problemi. Lo standard RS232C supporta solo una comunicazione punto a punto; è caratterizzata dall'accettare segnali elettrici compresi tra $+12V$ e $-12V$ e, infine, da circuiti elettrici che possono essere facilmente disturbati da campi magnetici. La comunicazione in RS232C ha, nella configurazione minima, come supporto fisico un doppino (2 fili) più il conduttore di riferimento.

1.7.2 RS422

Lo standard RS232C definisce velocità trasmissive limitate. Questa lentezza è dovuta all'utilizzo di una tecnica trasmissiva dei segnali sbilanciata, cioè con un unico riferimento comune a $0V$. E' possibile coprire distanze maggiori a velocità superiori facendo uso della trasmissione bilanciata come nel caso dell'interfaccia RS422. Con essa, ogni circuito di interfaccia è composto da due fili, e su essi il segnale è pilotato in controfase.

La RS422 prevede soltanto $0,4V$ ($-0,2V$ di differenza tra i due conduttori per lo stato logico 1 e $+0,2V$ per lo 0). In questo modo è possibile anche ridurre l'impedenza di carico a 100Ω. La distanza di comunicazione in funzione della velocità di trasmissione in una rete RS422 è maggiore rispetto alla RS232, sotto ogni punto di vista. E' possibile raggiungere distanze maggiori con maggiori velocità. La comunicazione in RS422 può essere sia di tipo punto punto sia di tipo multipunto. Un'interfaccia di questo genere permette un collegamento di tipo master-slave in cui il dispositivo master gestisce la comunicazione. La comunicazione in RS422 ha come supporto fisico una coppia di doppini intrecciati (4 fili).

1.7.3 RS485

Lo standard RS485 è equivalente a livello di segnale all'interfaccia RS422. La comunicazione è di tipo half duplex. Il supporto fisico è un doppino intrecciato (2 fili). Il messaggio di send e receive viaggia sullo stesso supporto con un segnale di abilitazione alla trasmissione o ricezione per la scelta del tipo di messaggio. Un' interfaccia di questo genere permette un collegamento di tipo master-slave in cui il dispositivo master gestisce la comunicazione.

Dal punto di vista fisico, la trasmissione dei segnali avviene mediante l'uso di un doppino telefonico o cavo twistato che ha la caratteristica di essere economico e facile da installare; purtroppo non è in grado di garantire elevate velocità di trasmissione (fino ad un massimo di circa 1 Mbps).

Il collegamento può essere realizzato anche con un cavo coassiale che però risulta meno economico del doppino telefonico e, inoltre, di più complessa installazione; d'altra parte, consente una maggiore velocità di trasmissione (fino a circa 10 Mbps). Infine, è possibile impiegare le fibre ottiche che, ad un costo in assoluto più elevato, consentono le migliori prestazioni in termini di velocità (circa $2,5$ Gbps), distanze e immunità dai disturbi elettromagnetici.

1.8 Moduli per reti di PLC

Il PLC può gestire, oltre che interfacce di segnali locali, anche una serie di interfacce remote. Parlare di interfacce remote vuol dire poter connettere il PLC in rete.

La rete è un sistema di controllo distribuito in cui sono presenti una o più unità centrali alle quali sono collegati un certo numero di I/O remoti. Gli I/O possono

essere interfacce (moduli) per sensori (ingressi) o per attuatori (uscite).

La comunicazione in una rete può essere suddivisa in diversi livelli, a seconda della quantità di dati da scambiare e della velocità con cui i dati vengono scambiati.

Reti di I/O distribuiti: è una rete ad alta velocità per I/O digitali o analogici. Normalmente la comunicazione è di tipo master-slave. Lo scambio di informazioni è semplificato per poter essere il più rapido possibile. Le informazioni tra il master e un singolo slave sono ridotte al minimo.

Reti di I/O con scambio di informazioni: una rete di questo tipo normalmente può gestire scambio di dati, oltre che a livello di I/O, anche con altri controllori. Aumentando la quantità di informazioni che si può scambiare, se si aumenta la distanza, viene invece ridotta la velocità di scambio dei dati.

Reti di I/O con scambio di informazioni tra PLC e PC: è il livello di scambio dati di alto livello. Vengono scambiate solo informazioni senza interfaccia fisica di I/O, come nel caso delle reti Ethernet.

Sono disponibili sul mercato apparecchiature master collegate con moduli I/O che consentono di effettuare un controllo distribuito dell'impianto.

I moduli I/O remoti DeviceNet offrono la possibilità che sia il dispositivo a fornire informazioni al modulo master secondo le soglie impostate nel modulo, permettendo di snellire la parte del programma relativa alla diagnostica. Consentono di semplificare le operazioni di manutenzione in quanto, in caso di guasto, è possibile avere lo storico degli allarmi avvenuti sul modulo oppure lo stato del sistema al momento dell'allarme.

1.9 OMRON CJ1M

L'elaborato in oggetto si riferisce in particolar modo al PLC Omron con CPU CJ1 (Figura 1.4) che è il dispositivo con il quale si è realizzato l'intero progetto. Da questo punto le nozioni che andremo discutere riguarderanno specificatamente questo PLC anche se in linea generale valgono per ogni tipo di PLC.

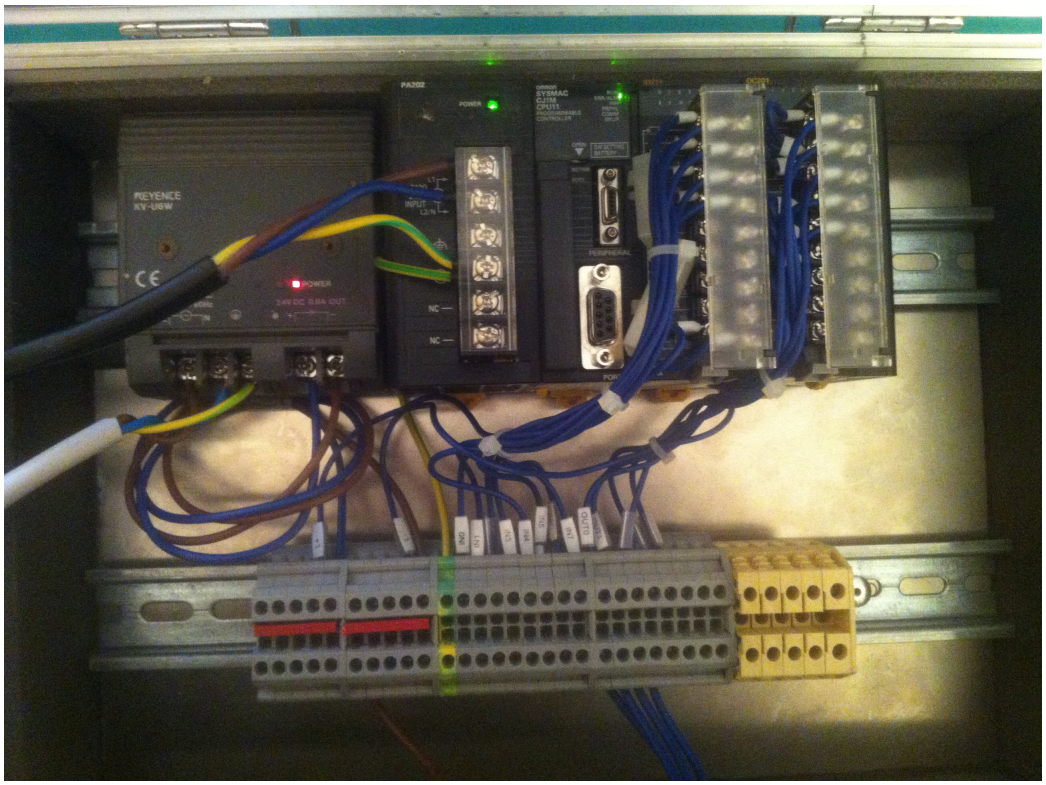


Figura 1.4: PLC Omron CJ1M utilizzato

1.9.1 Aree di memoria

Come già discusso in precedenza, alla base di un PLC ci sono le area di memoria. Tutto ruota intorno ad esse. Vediamo ora come sono suddivise e come possono e devono essere utilizzate per poter eseguire un programma:

1. **Area CIO (Core I/O area)** 6144 word, nessun prefisso. Area di memoria di I/O, moduli speciali, schede di comunicazione, Area non ritentiva, viene azzerata ad ogni avvio del PLC. L'accesso può avvenire a bit o a canale.

2. **Area HR (Holding area)** 512 word, prefisso H. Area ritentiva gestibile a bit e a canale. Memorizzazione permanente di dati di programma.

3. **Area W (Work area)** 512 word, prefisso W. Può essere utilizzata per registri temporanei o bit di appoggio; viene usata tipicamente per l'allocazione automatica. Accesso a bit e a canale. Non ritentiva.

4. **Area A (Ausiliary area)** 960 word di cui 449 a sola lettura, prefisso A. Contiene informazioni per la diagnostica, lo stato e la configurazione del PLC.

5. **Area TR (Temporary Relay)** 16 bit, prefisso TR. I bit temporanei vengono utilizzati per memorizzare le condizioni di esecuzione ON/OFF nelle diramazioni del programma.

6. **Area Temporizzatori (Timer area)** 4096 word, prefisso T. Viene simulato il funzionamento di temporizzatori. L'area è condivisa per tutte le istruzioni di temporizzazione, tranne quelle che non necessitano del numero del timer.

7. **Area Contatori (Timer area)** 4096 word, prefisso C. Viene simulato il funzionamento di contatori. L'area è condivisa per tutte le istruzioni di conteggio ed è in comune con l'area Temporizzatori.

8. **Area DM (Data Memory)** 32768 word, prefisso D. Area di memoria ritentiva per la memorizzazione dei dati. Viene utilizzata anche per la memorizzazione della configurazione dei moduli speciali.

1.9.2 Task

Per una migliore organizzazione del programma questi PLC hanno la possibilità di strutturare il programma in task secondo macro funzioni.

I task vengono eseguiti in modo sequenziale ed è possibile attivare o disattivare i task ciclici all'interno del programma.

La disattivazione di un task di programma con numero inferiore a quello che si sta eseguendo avverrà nella scansione successiva del programma.

Il programma può essere suddiviso in 32 task ciclici e 256 task ad interrupt.

All'interno dei task possono essere utilizzate subroutine (1024 in totale). Gli interrupt, siano essi hardware o software, sono gestiti da task dedicati.

L'esecuzione dei task è sequenziale. La memoria del PLC è comune a tutti i task.

1.9.3 Allocazione I/O

L'allocazione dei canali CIO per i moduli di I/O standard procede in modo consecutivo seguendo l'ordine di installazione, sia per gli ingressi sia per le uscite, a partire dal

canale CIO 0000.

Al contrario i moduli speciali vengono indirizzati con 2 switch rotativi (decine e unità) presenti sul frontale del modulo. Ogni modulo può occupare una o più posizioni logiche.

Se il modulo speciale occupa più posizioni logiche, il successivo dovrà essere indirizzato in un'area libera.

Ad esempio, se si prende un modulo speciale che occupa 5 posizioni logiche e che è indirizzato (con gli switch rotativi) al valore 0, il modulo occuperà anche gli indirizzi 1,2,3 e 4.

Quindi, un successivo modulo speciale non potrà essere impostato con un indirizzo già utilizzato e dovrà essere impostato con indirizzo (switch rotativo) 5.

1.9.4 Tabella degli I/O

La configurazione hardware dei moduli del CJ1 può essere memorizzata in una tabella detta appunto *Tabella di I/O*.

All'accensione, se la tabella è stata registrata, il sistema verifica la configurazione attuale con quella registrata; in caso di mancata corrispondenza, viene generato un errore.

Se è stato tolto, aggiunto o spostato un modulo, il sistema fornisce la segnalazione di allarme non fatale "I/O VERIFY ERROR", mentre se la variazione di configurazione è costituita dallo scambio di posizione tra un modulo di ingresso e uno di uscita viene fornita la segnalazione di allarme fatale "I/O SET ERROR" e l'esecuzione del programma si arresta.

La tabella di I/O può essere creata, letta, verificata e cancellata. Se la tabella è stata cancellata, il sistema all'accensione non esegue alcun controllo.

1.9.5 Codifica delle informazioni

Il PLC è in grado di riconoscere, come si è detto, informazioni elementari, dette bit rappresentabili con 0 e 1, e segue la logica del sistema binario.

Per poter colloquiare con un PLC noi possiamo utilizzare numeri, lettere e vari simboli (es. punteggiatura).

Affinché il PLC sia in grado di riconoscere i simboli che gli vengono inviati, essi devono essere tradotti in una serie di 0/1, è necessario cioè codificarli, in modo che ogni simbolo abbia un codice diverso da tutti gli altri. Un codice necessita per il suo utilizzo di un certo numero di bit. Qunado si trasforma un valore numerico dalla forma decimale in uno ainn forma binaria si effettua una conversione, si fa cioè corrispondere un numero decimale ad un gruppo di bit. Questo insieme di bit viene chiamato *parola (word)*, il numero di bit costituisce la lunghezza della parola.

La combinazione di un certo numero di bit costituisce in generale ciò che viene chiamata una word (4,8,16,32,64 bit, dipende dal tipo di apparecchiatura), come caso particolare una combialnzione di 8 bit viene chiamata Byte (il cui simbolo è una "B" maiuscola)

Capitolo 2

Comunicazione Seriale PLC

Spesso nasce la necessità di far comunicare dispositivi e software di natura differente. In questo caso l'obiettivo è quello di riuscire a creare un interfaccia MatLab-PLC, che riesca ad interpretare dati e comandare le uscite.

2.1 Host Link

I dispositivi Omron CJ1 nel modulo CPU (modulo base) sono dotati di una porta proprietaria Omron (Peripheral port) e di una porta RS232C. La prima è dedicata alla programmazione: permette il trasferimento della configurazione, dei dati e del programma da CX-Programmer nell'area di memoria programma dedicata o inversamente si può trasferire la configurazione presente sul PLC al PC. La seconda, la RS232C, permette la comunicazione con terze parti che utilizzano lo stesso standard.

Lo standard che useremo in per questo progetto è proprio RS232C, il quale sarà descritto più nel dettaglio nel paragrafo 2.5.

Per poter utilizzare la RS232C occorre una particolare piedinatura che è illustrata in Figura 2.1.

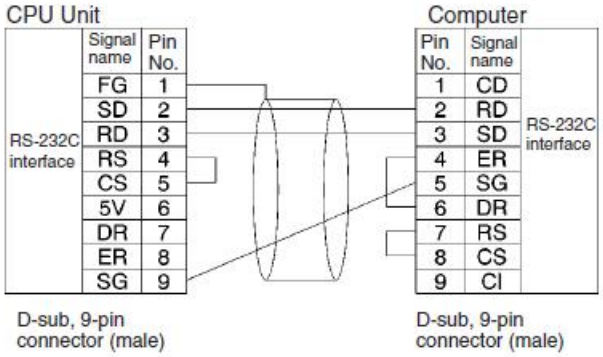


Figura 2.1: Cablaggio cavo Seriale

La porta seriale può essere utilizzata come semplice scambio di dati, ad esempio lettura di informazioni da un codice a barre, o come invio/ricezione di comandi.

Il nostro scopo è quello di comandare il PLC da MatLab per cui utilizzeremo la modalità HostLink per l'invio di comandi.

I PLC CJ1 possono ricevere due tipi di comandi di comunicazione:

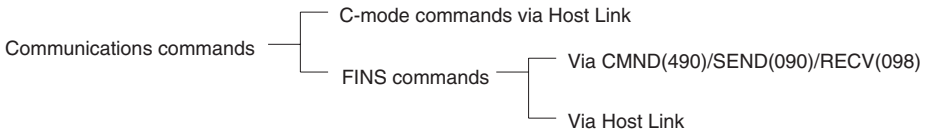


Figura 2.2: Comunications command

C-mode Commands I comandi C-mode sono specifici per la comunicazione in modalità HostLink. Essi sono emessi da un computer host e inviati a una CPU-unit. I dispositivi che possono essere collegati per le comunicazioni seriali sono la CPU, un' unità di comunicazione seriale e una scheda di comunicazione seriale.

FINS Commands I comandi FINS sono messaggi di servizio e non dipendono da un particolare percorso di trasmissione. Possono essere usati per diverse reti di comunicazione (es. Controller Link, Ethernet, ecc...) e per la comunicazione seriale in modalità HostLink. Comandi FINS possono essere emessi da una Unità CPU, da moduli speciali I/O o computer host e possono anche essere inviati ad uno di questi. I comandi specifici che possono essere inviati dipendono dalla destinazione.

Questo elaborato tratta comandi C-mode spediti alla CPU di un CJ1M da un PC connesso in modalità Host Link.

2.2 C-mode Commands

I comandi C-mode (Host Link) formano un sistema di comando/risposta per le comunicazioni seriali (Host Link Mode) per eseguire diverse operazioni di controllo tra una CPU e un computer host collegato direttamente ad esso. Queste operazioni includono la lettura e la scrittura nella memoria I/O, cambiamenti dei modi operativi, l'esecuzione forzata di impostazioni di set e reset, e così via.

Esistono 2 tipi di formato Host Link: collegamento 1:1 dove l' host pc è collegato con un solo PLC, o 1:N dove il PC è connesso con una rete di PLC collegati tra loro.

Diversamente dai comandi FINS, i C-mode possono essere recepiti solo una CPU, e non possono essere utilizzati come messaggi di servizio al di fuori della rete locale. Non possono neanche essere utilizzati per funzioni quali operazioni di file.

I C-mode possono essere inviati da un Host computer al quale possono essere connessi fino a 32 PLC. La loro identificazione è permessa dall'assegnazione da parte dell'Unità Host Link di una numerazione che va da 0 a 31.

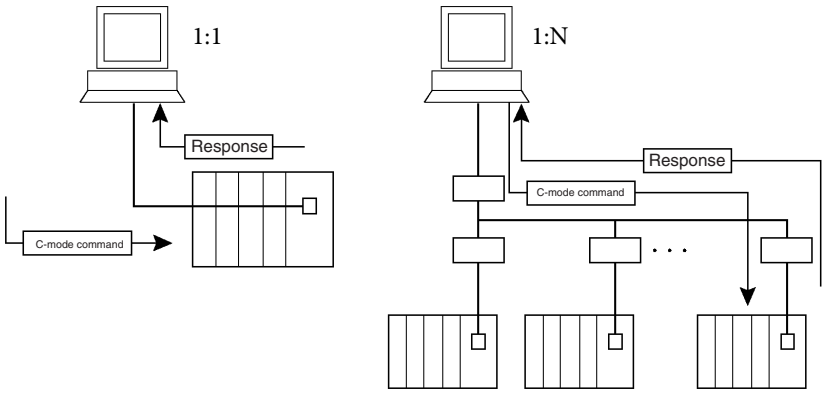


Figura 2.3: Host Link 1:1 e 1:N

La lunghezza di una singola unità di comando o risposta è chiamato **"frame"**. Ogni frame contiene un massimo di 131 caratteri, i quali sono inviati e ricevuti in ASCII.

Può essere trasferito un massimo di 30 word di dati con il primo frame e un massimo di 31 word con i successivi se i comandi di lettura o scrittura di dati appartengono alla Input/Output Memory. Quando si superano le 30 word di dati, il processo di trasferimento si suddivide in una trasmissione multipla con 30 word nella prima e 31 nelle successive. Il formato del frame per i comandi Host Link inviati da un Host computer e in risposta dal PLC sono illustrati nella sezione successiva.

2.3 Formati Comando/Risposta

Come già specificato nel paragrafo precedente un frame non deve superare i 131 caratteri di lunghezza per essere inviato in un'unica stringa di comando. In questi casi si parla quindi di *single-frame command*.

La struttura di un Single-frame command è illustrata in figura 2.4 e si compone di:

- **@** Deve essere inserita all'inizio del comando;
- **Unit Number** Numero dell'Host Link Unit espresso in BCD $(0 \div 31)$;
- **Header Code** Specifico per ogni tipo di operazione;
- **Text** Parametri specifici per il comando inviato;
- **FCS** *Frame Check Sequence* 2 caratteri di controllo;
- **Termination** *"*CR"* Indicano la fine del comando;

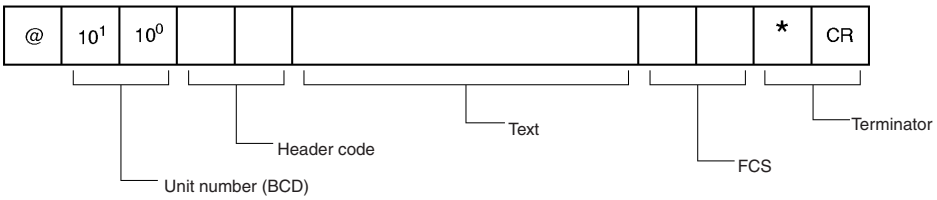


Figura 2.4: Single-Frame command

Allo stesso modo un frame di risposta non deve superare i 131 caratteri per poter essere inviato come singolo comando. Nel caso di una risposta questa stringa prende il nome di *single-frame response* e come illustra in Figura 2.5 è costruita in maniera simile al single-frame command

- **@** Indica l'inizio del comando;
- **Unit Number** Numero dell'Host Link Unit espresso in BCD che ha risposto$(0 \div 31)$;
- **Header Code** Ritorna il tipo di operazione effettuata;

- **End Code** Esito del'operazione effettuata;
- **Text** Presente solo se è stata effettuata una lettura;
- **FCS** *Frame Check Sequence* 2 caratteri di controllo;
- **Termination** *"*CR"* Indicano la fine del comando;

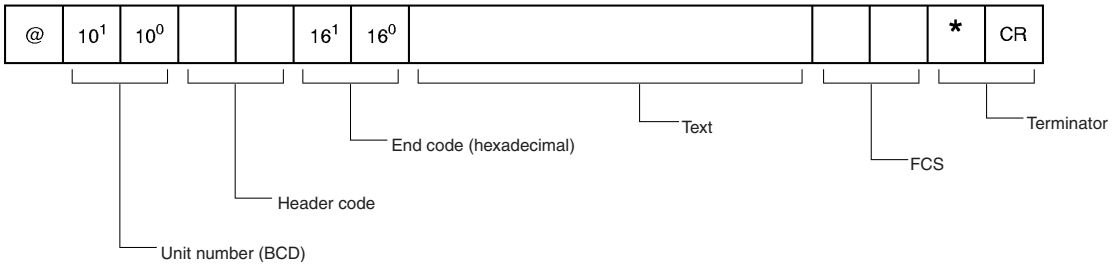


Figura 2.5: Single-Frame response

2.4 FCS

Nei paragrafi precedenti abbiamo parlato di *Frame Check Sequence*, vediamo ora più nel dettaglio a cosa serve e come viene calcolato.

Al fine di verificare la correttezza del single-frame ricevuto con quello interpretato viene calcolato sulla stringa un numero (2 caratteri ASCII) prima dell'invio e viene inserito nella stringa che si sta inviando. L'Host Link Unit che riceve la stringa ricalcola l'FCS che secondo lui dovrebbe essere, per la stringa arrivata, e lo confronta con quello inviato. Se i due risultano uguali allora la trasmissione è andata a buon fine, altrimenti in risposta viene mandato un single-frame response che informa l'Host Computer dell'errore di comunicazione.

Il Frame Check Sequence non è altro che un OR ESCLUSIVO (XOR) di tutti i caratteri inviati. Si scompone ogni carattere ASCII inviato nel rispettivo codice esadecimale a 2 cifre. Ogni cifra viene convertita in un binario 4 bit, e viene eseguito lo XOR sequenziale tra i caratteri che compongono la stringa.

Il risultato è un binario a 8 bit che viene convertito in 2 caratteri esadecimali e manipolato in ASCII.

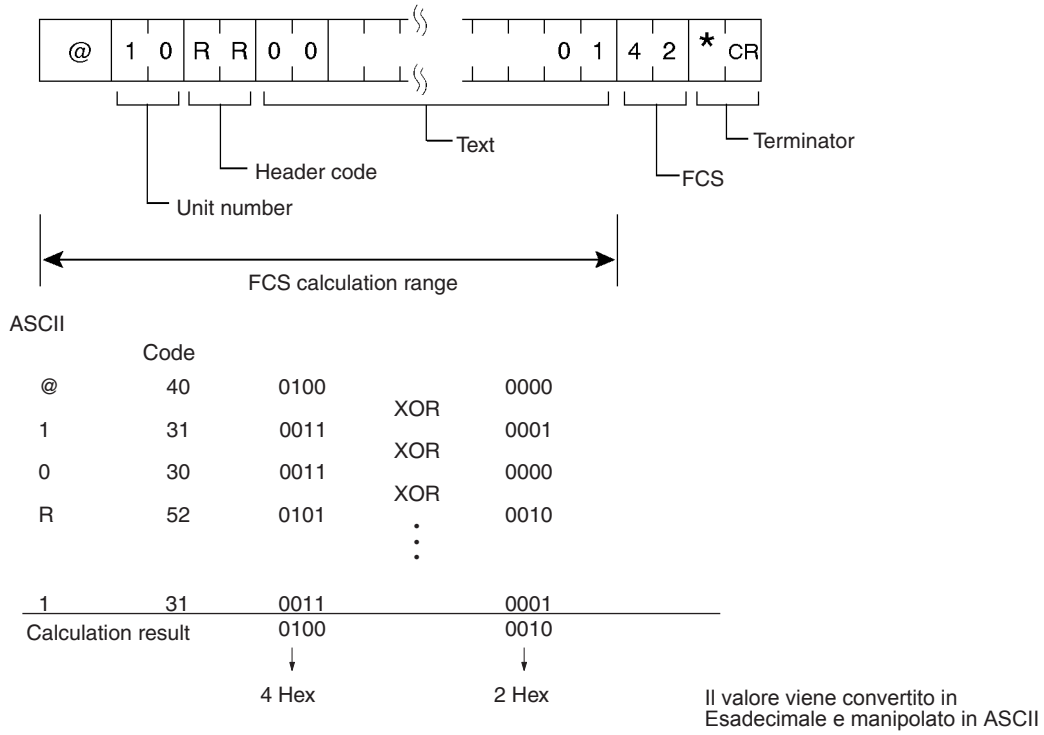


Figura 2.6: Calcolo dell'FCS

2.5 Come è fatto un segnale RS232C

La cosa più semplice per descrivere un segnale RS232 è partire con un esempio.

Nell'immagine 2.7 è visualizzato, in modo idealizzato, cosa appare collegando un oscilloscopio ad un filo su cui transita un segnale RS-232 a 9600 bps del tipo 8n2 (più avanti verrà spiegata questa sigla) rappresentante il valore binario 00110000.

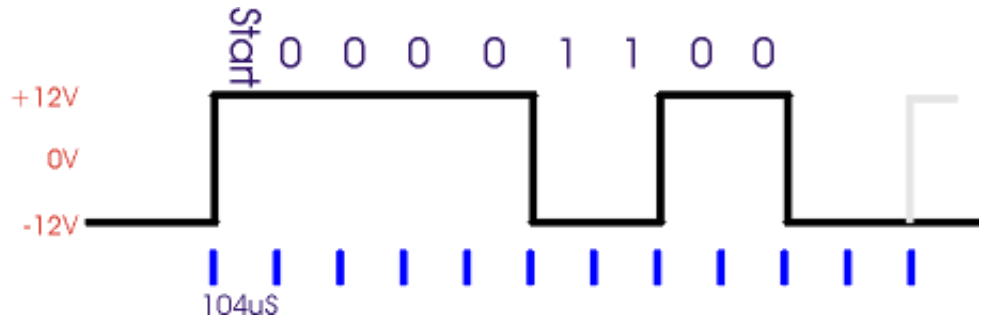


Figura 2.7: RS232

L'ampiezza del segnale è caratterizzata da un valore alto pari a circa $+12V$ ed un valore basso pari a circa ˇ$12V$. Da notare che, nello standard RS-232 un segnale alto rappresenta lo zero logico ed uno basso un uno, come indicato nel disegno e rovesciato rispetto al comune pensare.

A volte un segnale alto ($+12V$, cioè uno zero logico) è indicato come space ed uno basso ($-12V$, uno logico) come mark.

Tutte le transizioni appaiono in corrispondenza di multipli di $104\mu s$, pari ad $\frac{1}{9600}$ cioè ciascun bit dura esattamente l'inverso del baud rate.

La linea si trova inizialmente nello stato di riposo, bassa (nessun dato in transito); la prima transizione da basso in alto indica l'inizio della trasmissione (inizia il bit di start, lungo esattamente $104\mu s$). Segue il bit meno significativo (LSB), dopo altri 104 us il secondo bit, e così via, per otto volte, fino al bit più significativo (MSB).

Da notare che il byte è trasmesso al contrario, cioè va letto da destra verso sinistra. Segue infine un periodo di riposo della linea di almeno $208\mu s$, cioè due bit di stop e quindi (eventualmente) inizia un nuovo pacchetto di bit con un nuovo bit di start (in grigio nel disegno).

Le varianti possibili sono le seguenti:

- Se la trasmissione è più veloce o più lenta, la distanza tra i fronti varia di conseguenza (p.e. a 1200 bps le transizioni avvengono a multipli di $0,833ms$, pari a $\frac{1}{1200}$);
- Invece di trasmettere 8 bit, ne posso trasmettere 6, 7 o anche 9 (ma quest'ultima possibilità non è prevista dalle porte seriali dei normali PC);
- Alla fine è possibile aggiungere un bit di parità, descritto più avanti;
- Alla fine la linea rimane nello stato di riposo per almeno 1 o 1,5 o 2 bit; notare che, se non ho più nulla da trasmettere, il riposo è molto più lungo, ovviamente. Molti sistemi non possono utilizzano 1,5 bit di stop.

In genere il formato del pacchetto trasmesso è indicato da una sigla composta da numeri e cifre, per esempio 8n1 e 7e2:

- La prima cifra indica quanti bit di dati sono trasmessi (nei due esempi rispettivamente 8 e 7);
- La prima lettera il tipo di parità (rispettivamente nessuna ed even-parity, cioè parità pari);
- La seconda cifra il numero di bit di stop (rispettivamente 1 e 2).

Tenendo conto che esiste sempre un solo bit di start, un singolo blocco di bit è quindi, per i due esempi riportati, costituito rispettivamente da 10 (1+8+0+1) e 11 (1+7+1+2) bit. Da notare che di questi bit solo 8 e, rispettivamente, 7 sono effettivamente utili.

Lo standard originale prevede una velocità fino a 20Kbps. Uno standard successivo (RS-562) ha portato il limite a 64Kbps lasciando gli altri parametri elettrici praticamente invariati e rendendo quindi i due standard compatibili a bassa velocità. Nei normali PC le cosiddette interfacce seriali RS-232 arrivano in genere almeno a 115Kbps, 230Kbps o anche più: pur essendo tali valori formalmente al di fuori di ogni standard ufficiale non si hanno particolari problemi di interconnessione.

Una precisazione: trasmettitore e ricevitore devono accordarsi sul modo di trasmettere prima di iniziare la trasmissione stessa, pena l'impossibilità di instaurare la trasmissione o ricevere bit che appaiono casuali. Questa operazione va fatta configurando opportunamente il software e/o modificando manualmente alcuni dip-switch o altri dispositivi hardware.

E' importante garantire il rigoroso rispetto della durata dei singoli bit: infatti non è presente alcun segnale di clock comune a trasmettitore e ricevitore e l'unico elemento di sincronizzazione è dato dal fronte si salita del bit di start. Come linea guida occorre considerare che il campionamento in ricezione è effettuato di norma al centro di ciascun bit: l'errore massimo ammesso è quindi, teoricamente, pari alla durata di mezzo bit (circa il 5% della frequenza di clock, considerando che anche il decimo bit deve essere correttamente sincronizzato). Naturalmente questo limite non tiene conto della difficoltà di riconoscere con precisione il fronte del bit di start (soprattutto su grandi distanze ed in ambiente rumoroso) e della presenza di interferenze intersimboliche tra bit adiacenti: per questo spesso si consiglia caldamente di usare un clock con una

precisione migliore dell'1% imponendo, di fatto, l'uso di oscillatori a quarzo.

Si potrebbe anche ipotizzare un meccanismo che tenta di estrarre il clock dai fronti intermedi ma si tratta nel caso specifico di un lavoro poco utile, visto che la lunghezza del pacchetto è piuttosto breve.

Il bit di parità

Oltre ai bit dei dati (in numero variabile tra 5 ed 9) viene inserito un bit di parità (opzionale) per verificare la correttezza del dato ricevuto. Esistono diversi tipi di parità:

- None: nessun tipo di parità, cioè nessun bit aggiunto;
- Pari (even): il numero di mark (incluso il bit di parità) è sempre pari;
- Dispari (odd): il numero di mark (incluso il bit di parità) è sempre dispari.

L'idea è quella di predeterminare la quantità di 1 (e di conseguenza di 0) da trasmettere, facendo in modo che il loro numero sia sempre pari (o dispari, a secondo della scelta che si vuole fare): così facendo, se durante la trasmissione dovesse accadere un errore su un singolo bit, il ricevitore sarebbe in grado di rilevare l'errore, ma non di correggerlo. Si tratta ovviamente di un protocollo di controllo degli errori elementare e di conseguenza in disuso a favore di altri sistemi basati su codici a ridondanza ciclica (CRC) o altri algoritmi più complessi.

Il bit di parità a volte viene mantenuto sempre ad un livello prestabilito, per esempio in alcuni protocolli usati da macchine industriali. Ciò dà origine ad ulteriori due tipologie di parità, peraltro non molto comuni:

- Mark: il bit di parità vale sempre mark
- Space: il bit di parità vale sempre space

Tali configurazioni sono a volte usate per identificare la tipologia del byte trasmesso, per esempio potrebbe indicare se si tratta di un dato piuttosto che di un indirizzo.

2.6 MatLab - Oggetto Seriale

In MatLab è possibile costruire un oggetto seriale nel quale vengono specificate tutte le caratteristiche della comunicazione seriale che si vuole effettuare.

```
%% Setting PORTA SERIALE
delete(instrfindall);
rs232='com5'
plc=serial(rs232);
set(plc, 'BaudRate', 115200, 'StopBits', 1,'DataBits',8);
set(plc, 'Terminator', 'CR', 'Parity', 'even');
set(plc, 'FlowControl', 'none');
```

Figura 2.8: MatLab - Setting porta seriale

In Figura 2.8 possiamo vedere le caratteristiche specifiche per il nostro tipo di comunicazione.

E' stato settato un baundrate di 115200 *bps* che è il massimo impostabile sull'PLC, con un segnale del tipo $8N1$, identico a quello settato sul PLC.

Come specificato precedentemente il PLC prevede per default che il carattere con cui riconosce il termine di una stringa è il carattare "CR", così impostiamo in MatLab che ogni comando terminerà con il carattere CR senza doverlo specificare ogni volta. Questo è possibile settando il carattere *termination* nell'oggetto seriale.

Capitolo 3

Sistemi produttivi

I processi simulati e controllati saranno processi riguardanti sistemi di produzione con la gestione di magazzini. I sistemi produttivi possono essere suddivisi in diverse categorie: possiamo suddividerli in base alle caratteristiche delle macchine che lo compongono, da qui la distinzione tra sistemi a setup trascurabile e non trascurabile, oppure potremmo distinguerli tra loro in funzione delle caratteristiche dei magazzini e lavorazioni, da qui sistemi pull e sistemi push. Per le nostre simulazioni abbiamo intrecciato le varie caratteristiche ed abbaiamo generato modelli di sistemi di tipo push con tempi di setup trascurabile e sistemi di tipo pull con tempi di setup non trascurabile. Assegnata la struttura e il numero di macchine del sistema, si esaminerà il problema del controllo del sistema di produzione, inteso come individuazione, in base allo stato attuale del sistema e della domanda, dei pezzi che vanno lavorati in ciascuna macchina e il tasso di produzione per ciascuna di esse. I sistemi non saranno approssimati a modelli a flussi continui, e l'implementazione delle leggi di controllo riguarderà sistemi discreti. Il problema della pianificazione della produzione, detto scheduling, verrà affrontato da un punto di vista dinamico: le decisioni vengono prese in base allo stato attuale del sistema e quindi a catena chiusa.

3.1 Push con Setup non trascurabile

In questa sezione considereremo sistemi di produzione con le seguenti caratteristiche:

- M=numero di macchine del sistema;
- P=numero di tipi di parti prodotte;
- d_j=tasso di domanda del pezzo j;
- τ_{pm}=tempo affinché la macchina m lavori il pezzo p;
- δ_{pqm}=tempo di setup della macchina m per passare dalla lavorazione p a quella q.

Inoltre ipotizziamo che questi sistemi siano di tipo *push* ovvero la domanda di tipo i può essere pensata come pezzi di tipo i in arrivo nel sistema che richiedono una lavorazione (oppure come clienti che richiedono un servizio). In questo caso pertanto il sistema non può produrre o lavorare pezzi che non gli sono ancora arrivati. Questo corrisponde ad avere contenuti dei buffers $x_i(t) \geq 0$ per ogni t (quindi mai negativi) e una produzione cumulativa $y_i(t)$ in $(0, t)$ non superiore alla domanda cumulativa td_i, nel caso di una domanda d_i costante.

Per questi tipi di sistemi la dinamica di aggiornamento è data da:

$$\frac{dx_j}{dt} = d_j - u_j(t)$$

e x_j non può diventare negativo.

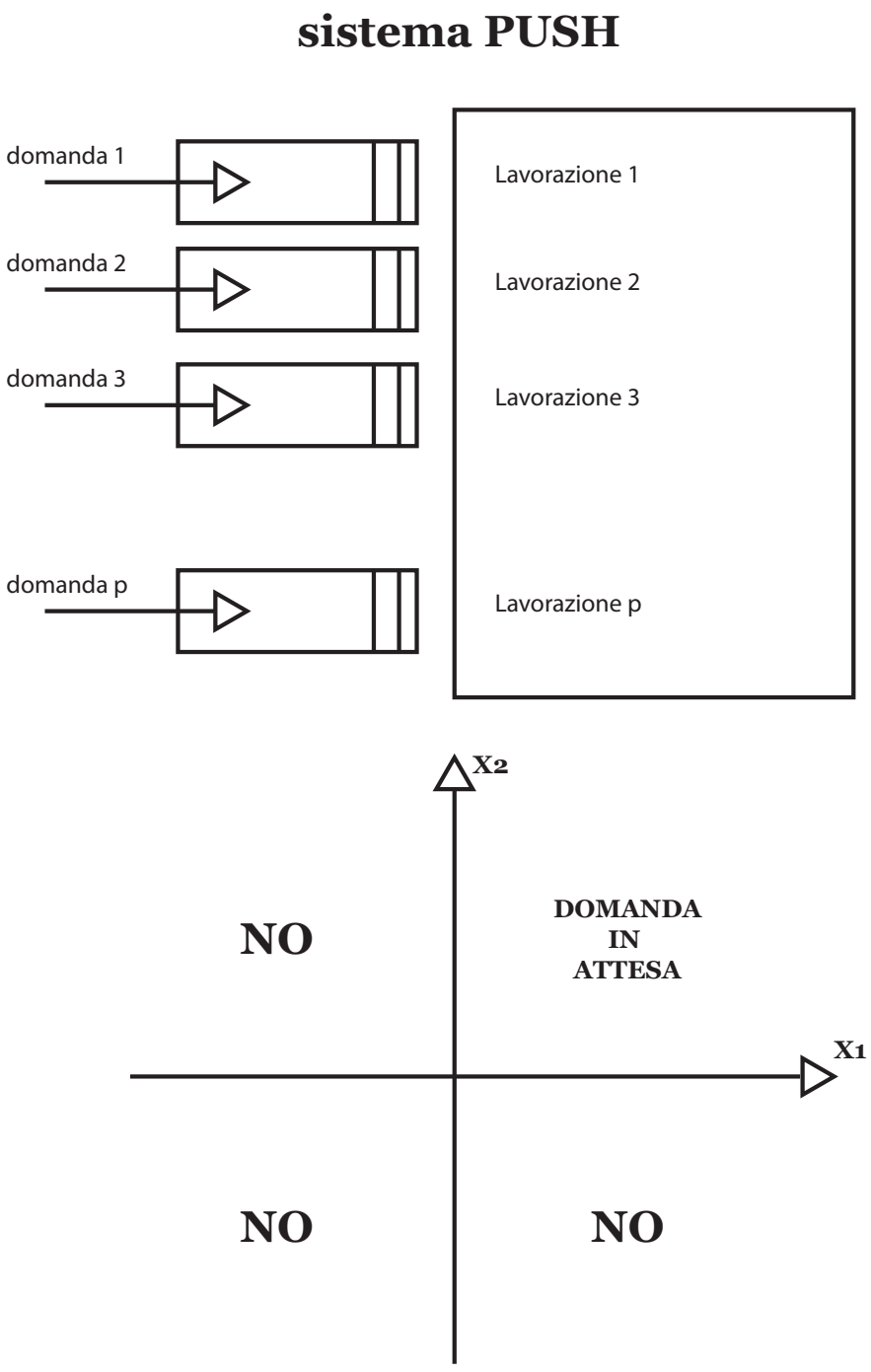


Figura 3.1: Descrizione sistemi Push

Il carico di lavoro della stazione m di questo sistema è rappresentato dalla frazione oraria che la macchina m deve lavorare e lo definiamo come:

$$\rho_m = \sum_{j=1}^{P} d_j \sum_i \tau_{jm}$$

Supponendo che ogni stazione abbia una sola macchina, il carico di lavoro ρ_m rappresenta il lavoro espresso in $\frac{ore}{ora}$, ovvero la percentuale di tempo che la macchina m deve lavorare per soddisfare tutta la domanda.

Questo tipo di sistema di produzione è stabile se e solo se il carico di lavoro ρ_m è inferiore ad 1 per ogni macchina.

$$STABILE \iff \rho_m < 1 \; \forall \; m \, \epsilon \, M$$

Si noti che per la stabilizzabilità il carico di lavoro deve essere strettamente minore dell'unità e non già minore o uguale. La mancanza dell'uguaglianza è dovuta al fatto che i tempi di setup sono trascurabili e quindi ad ogni macchina deve avanzare un po' di tempo per eseguire i setup.

Stabilità

Per i sistemi di produzione del tipo qui considerato la stabilità può essere definita in due diversi modi, equivalenti tra loro se il tempo di trasporto del sistema è limitato:

- se $x_{im}(t)$ è il contenuto del buffer della parte i alla macchina m, allora il sistema è stabile se il contenuto dei buffer di tutte le macchine (ovvero la somma di questi) rimane sempre limitato, cioè se esiste una costante finita M_{tot} tale che: $\sum_{j,m} x_{jm}(t) < M_{tot}\ \forall\ t$ detta anche **Limitatezza dei buffer**.
- se $y_j(t)$ è la produzione cumulativa di pezzi di tipo j, il sistema è stabile se $\forall\ j$ è possibile trovare una costante finita M_j tale che, $\forall\ j$ prodotta $t * d_j - Mj \leq y_j(t)$. Se considero un modello produttivo in cui non si produce oltre la domanda ossia non si accumulano scorte (come potrebbe accadere nel caso in cui il sistema che stiamo studiando sia piuttosto una rete di servizi in cui i clienti vengono processati solo una volta arrivati), complessivamente il sistema è stabile se $\forall\ t$ e j vale una condizione del tipo $t * d_j - Mj \leq y_j(t) \leq t * d_j$.

Quando vale la condizione $\rho_m < 1$ è sempre possibile determinare delle politiche che stabilizzano il sistema ed inoltre:

- sono applicabili in modo distribuito, cioè ogni macchina basa le sue scelte solo sul contenuto dei suoi buffers, senza necessità di avere altre informazioni sul resto del sistema;
- la stabilità non dipende dal sistema di trasporto (purché il ritardo massimo di trasporto sia finito) e quindi il problema del trasporto e quello dello scheduling sono disaccoppiati.

Nelle nostre simulazioni si prendono in considerazione sistemi costituiti da una sola macchina, per cui la definizione di stabilità diventa $\sum_{i=1}^{P} x_i < M_{tot}$, per cui la condizione di stabilità diventa:

$$\rho = \sum_{j-1}^{P} d_j \tau_j < 1$$

Carico di lavoro

Una volta stabilito che il sistema è stabilizzabile è importante valutare il carico di lavoro della macchina, cioè il tempo che la macchina deve lavorare in ciascuna unità di tempo per smaltire la domanda. Chiaramente in un'unità di tempo la macchina non può lavorare più dell'unità di tempo stessa, quindi se non ci fossero tempi di setup potremmo accontentarci di avere un $\rho \leq 1$.

La presenza dei tempi di setup richiede che alla macchina rimanga un po' di tempo da dedicare ai setup e quindi la condizione $\rho < 1$ è necessaria. E' evidente però che se la macchina esegue i setup troppo spesso invece di lavorare, i suoi buffer aumentano illimitatamente. Necessitiamo quindi di porci la domanda di quale sia la frequenza giusta con la quale vengono eseguiti i setup. Intuitivamente possiamo subito dire che la frequenza con la quale si passa da una lavorazione di un pezzo ad un altro di tipo diverso per una politica stabile è sempre limitata.

La quantità di lavoro arrivata in un intervallo $[0, T]$ è ρT ed è la quantità di tempo che la macchina deve lavorare nell'arco di tempo $[0, T]$ per smaltire tutta la domanda arrivata in tale intervallo di tempo. Quindi, $(1 - \rho)T$ è il tempo di inattività della macchina. Questo tempo può essere sfruttato per eseguire il setup. Sia allora n_T il numero di setup che la macchina effettua nell'intervallo $[0, T]$. La frequenza massima di setup, definita come numero di setup sull'intervallo di osservazione è quella che si ha quando tutto il tempo libero $(1 - \rho)T$ della macchina è dedicato ai setup:

$$f = \frac{n_T}{T} \leq \frac{1-\rho)T}{\delta T} = \frac{(1-\rho)}{\delta}$$

Il limite massimo, ovvero la frequenza di setup massima è $\frac{(1-\rho)}{\delta}$. Per le politiche non-idling la frequenza di setup massima a regime coincide sempre con tale valore.

3.2 Pull con Setup trascurabile

Un modello di sistema, per certi versi più generale è quello *pull*. Nei sistemi pull i buffers sono convezionalmente rappresentati a valle della macchina e possono assumere un contenuto positivo o negativo, col seguente significato:

- **buffer POSITIVO** rappresenta scorte in eccedenza di beni già prodotti
- **buffer NEGATIVO** rappresenta domande in attesa

L'evoluzione dei buffers per i sistemi pull è pari alla differenza tra il tasso di produzione e il tasso di domanda. Per questo sistema l'indice di prestazione è legato a due componenti: una che tiene conto delle domande in attesa e un'altra che tiene conto delle scorte positive. La condizione di stabilità è sempre la stessa:

$$\rho = \sum_{j=1}^{P} d_j \tau_j < 1$$

La giustificazione è banale: la stabilità di un sistema push è legata solo al rischio di non riuscire a soddisfare la domanda, e lo stesso vale per i sistemi pull, in quanto non esiste instabilità dovuta ad un accumulo eccessivo di scorte. Se infatti le scorte tendono a divergere è sufficiente bloccare la produzione.

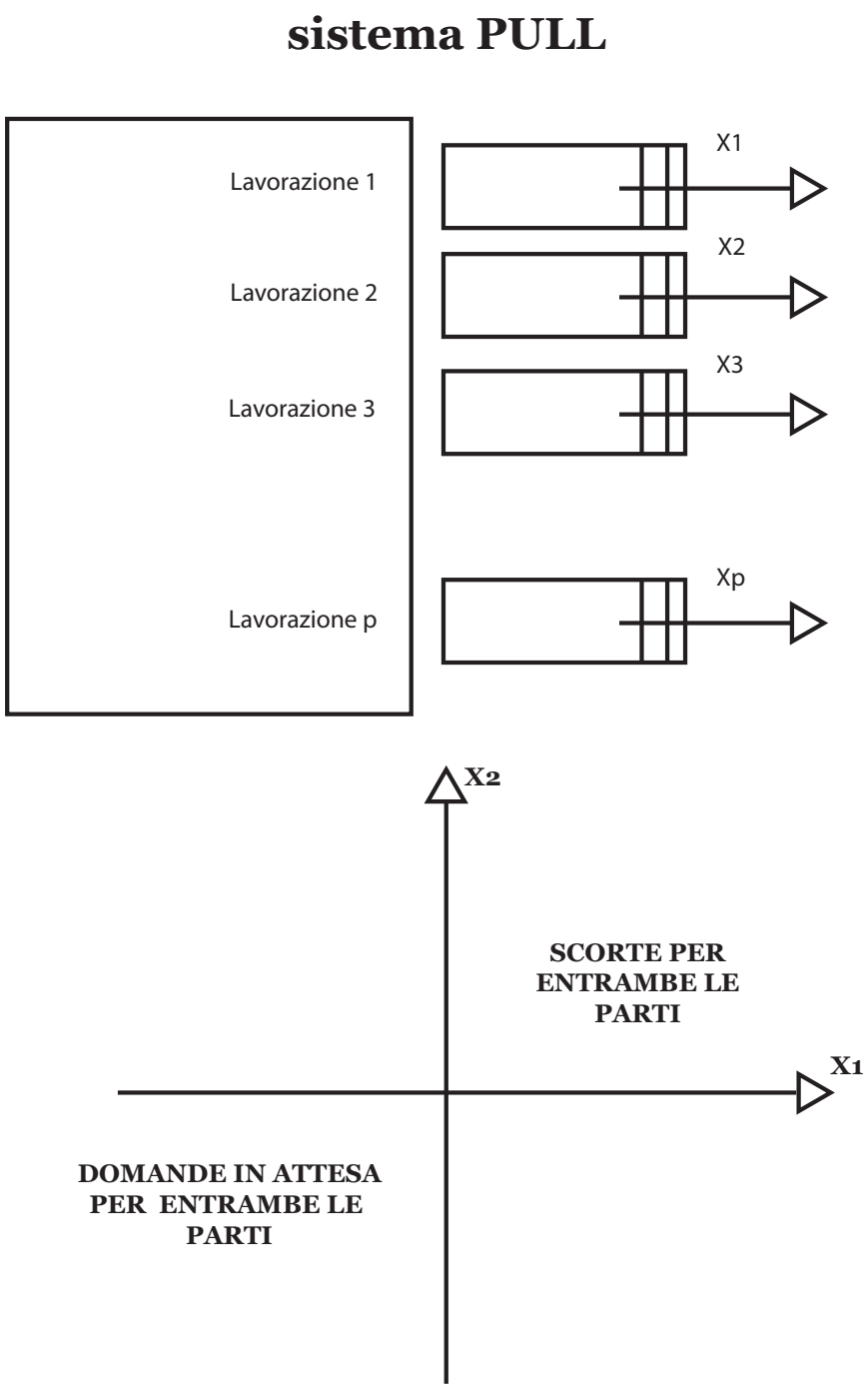


Figura 3.2: Descrizione sistemi Pull

Introduciamo ora il concetto di tempo di setup trascurabile, cioè $\delta_{ij} = 0$, dove δ_{ij} è il tempo necessario per passare dalla produzione di pezzi di tipo i a pezzi di tipo j.

L'ipotesi di tempo di setup trascurabile si traduce nella possibilità di lavorare contemporaneamente più tipi di pezzi, e questa è la differenza sostanziale con i sistemi a tempo di setup non trascurabile, in cui la macchina poteva lavorare un solo tipo di pezzi alla volta. Infatti, se il tempo di setup è nullo, dato un intervallo temporale $[t, t + dt]$ con durata arbitrariamente piccola di dt, è sempre possibile scomporlo in parti di durata $\alpha_i dt$, $\alpha_i \in [0, 1]$ con $\sum_{i=1}^{P} \alpha_i(t) \leq 1$, e lavorare in ciascuna di queste parti i pezzi di tipo i.

La possibilità di lavorare più parti contemporaneamente, cioè la caratteristica che i tempi di setup sono trascurabili, permette (se la capacità produttiva della macchina è abbastanza grande da soddisfare la domanda) di portare e mantenere tutti i buffer a 0, a differenza delle macchine con tempi di setup non trascurabili nei quali solo un buffer alla volta poteva essere portato a 0.

Questo si traduce graficamente nell'avere traiettorie nello spazio di stato convergenti nell'origine per ogni stato iniziale. L'ammisibilità della convergenza è data dalla legge di stabilità.

Nel caso di sistemi pull i buffers non hanno vincoli di segno e la legge che regola l'aggiornamento è data da:

$$\frac{dx_j}{dt} = u_j(t) - d_j$$

.

Capitolo 4

Simulazione

L'idea base è quella di usare un'area di memoria del PLC come area di memoria comune tra MatLab ed il PLC. Con MatLab simuleremo il processo come se fosse una macchina a stati indipendente ed userà l'area di memoria comune per scrivere il valore dello stato e leggere il controllo da utilizzare. Il compito del PLC è quello di usare i dati del processo scritti da MatLab, per calcolare il controllo ottimo istantaneo in base alla legge di controllo implementata su di esso.

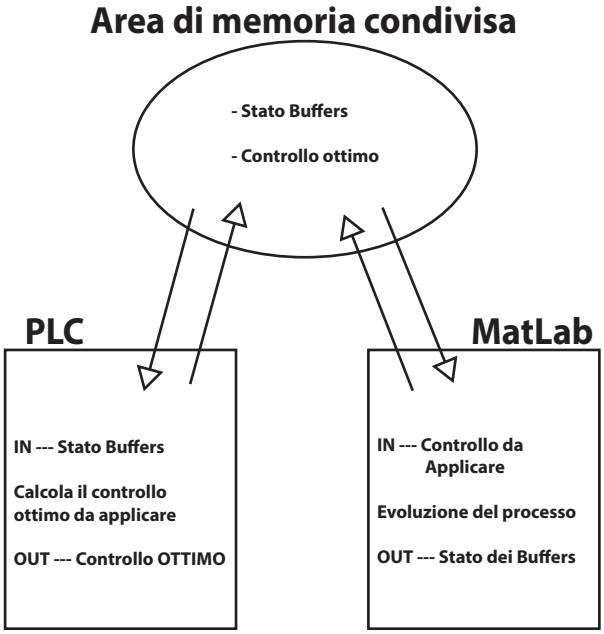


Figura 4.1: Principio di funzionamento della simulazione

4.1 MatLab

Alla base del progetto c'è la corretta esecuzione della trasmissione seriale. MatLab deve essere in grado di poter scrivere e leggere dal PLC, senza alcun vincolo. Inoltre come già visto nel capitolo 2 è possibile anche comandare il PLC.

Per poter fare ciò è necessario innanzitutto impostare il PLC in modalità Monitor, operazione che non sarebbe necessaria se il processo da simulare fosse reale. Distinguiamo quindi le 3 differenti modalità in cui può stare un PLC:

- **Run** Questa modalità viene utilizzata per la normale esecuzione del programma e per il funzionamento standard del sistema. In questa modalità l'indicatore di RUN è acceso. Alcune operazioni sul dispositivo non sono possibili come le modifiche on-line, la forzatura in set e reset, ed il cambiamento dei valori della memoria di I/O, ma sono disponibili altre operazioni come il monitoraggio dello stato di esecuzione del programma (monitoraggio dei programmi e monitoraggio della memoria I/O).

- **Monitor** Questa modalità è utilizzata per effettuare collaudi o altre regolazioni. Mentre il programma è in modalità MONITOR, l'indicatore RUN è acceso, e possono essere eseguite le seguenti operazioni:

  - Modifiche on-line
  - Forzatura di bit in set o reset
  - Modifica dei valori nella memoria I/O

- **Program** L'esecuzione del programma si arresta e l'indicatore RUN non è più acceso. Questa modalità viene utilizzata quando si modifica il programma o quando si fanno altre operazioni di preparazione come ad esempio:
  - Registrazione della tabella I/O
  - Modifica della configurazione del PLC e altre impostazioni
  - Trasferimento e controllo dei programmi

  In questa modalità tutti i task ciclici ed interrupt sono in NON-ESECUZIONE, cioè fermi.

Quindi per poter scrivere e quindi modificare un'area di memoria durante l'esecuzione di un programma è necessario che il PLC si trovi in modalità MONITOR. Impostiamo tale stato tramite comando seriale *@00SC0252*CR* dopodiché saremo abilitati a poter scrivere sulle aree di memoria.

4.2 Controllo sistemi PUSH a tempo di setup non trascurabile

Abbiamo visto nel capitolo 3 cosa si intende e quali sono le caratteristiche principali di un sistema produttivo le cui macchine che lo compongono hanno dei tempi di setup diversi da 0. In particolare ci concentriamo su sistemi costituiti da una sola macchina, per più macchine useremo gli stessi concetti ripetuti per ogni macchina.

La stazione di lavorazione può lavorare p tipi di pezzi a seconda delle richieste. Come decidere quale pezzo far lavorare alla macchina? Per quanto tempo? Per rispondere a queste domande ci si avvale di alcune leggi di controllo, che cercano di ottimizzare i tempi di lavorazione e garantiscono limitatezza dei buffers. Una politica per questo tipo di sistemi di produzione è la **CLB** - *Clear the Largest Buffer*. E' una politica non-idling abbastanza semplice. La politica prevede di lavorare il buffer maggiore fino a portarlo a 0, dopodiché viene scelto il successivo, ma occorre attendere il tempo di setup prima che l'altro pezzo venga lavorato. Così il primo buffer, che era stato portato a 0, aumenta nuovamente. Si crea un sistema ciclico perché non riusciremo mai a tenere tutti i buffer a 0.

4.2.1 Controllo CLB su PLC

Per implementare questo controllo sul PLC è importante tenere presente il fattore di sincronizzazione.

In MatLab, dopo aver inserito tutti i dati caratteristici del sistema (domanda, tasso produzione, numero buffers, ecc..) e della comunicazione seriale (numero porta, baund rate, bit di parità, bit di stop, ecc...) viene eseguito un ciclo, che come già accennato nell'introduzione del seguente paragrafo, legge dal PLC quale buffer lavo-

rare, esegue la lavorazione ed invia al PLC i nuovi valori dei buffers. Il processo in MatLab è scansionato come fosse un sistema discreto dove ogni unità di scansione può rappresentare minuti, ore, giorni a seconda del tipo di simulazione che si vuole ottenere. Basterà quindi adeguare i dati del sistema all'unità di misura prescelta.

Il PLC esegue il programma in maniera ciclica e continua, quindi è necessario fare attenzione che non calcoli sempre il controllo, ma deve essere calcolato solo ogni qualvolta il numero dei pezzi del buffer in lavorazione diventi nullo.

Tra le varie funzioni usate nel programma Ladder del PLC, le più importanti, quelle su cui si basa tutta la logica per l'implementazione della politica CLB c'è l'operazione di *differenziale*. E' un'opzione che può essere usata su quasi tutte le istruzioni del PLC e prevede che l'operazione alla quale è applicata viene eseguita solo una volta quando viene abilitata e non ad ogni ciclo. Usando questa opzione su tutte le operazioni, le quali saranno abilitate da un flag, che sarà settato ON solo quando ci sarà bisogno di variare la lavorazione, abbiamo garantito che il calcolo viene eseguito solo quando necessario.

Il programma è strutturato principalmente su due variabili. Una rappresenta quale parte si sta lavorando attualmente ed una quale parte dovrebbe lavorare secondo la politica di controllo.

Il buffer con contenuto massimo è quello che deve essere lavorato, ma se è lo stesso di ciò che stiamo lavorando non dobbiamo effettuare alcuna modifica.

Come possiamo osservare dalla Figura 4.2 ogni volta che necessitiamo di cambiare il tipo di lavorazione viene alimentato un timer, che rappresenta proprio il tempo di setup e che può essere impostato con tempi diversi per ogni tipo di lavorazione da effettuare.

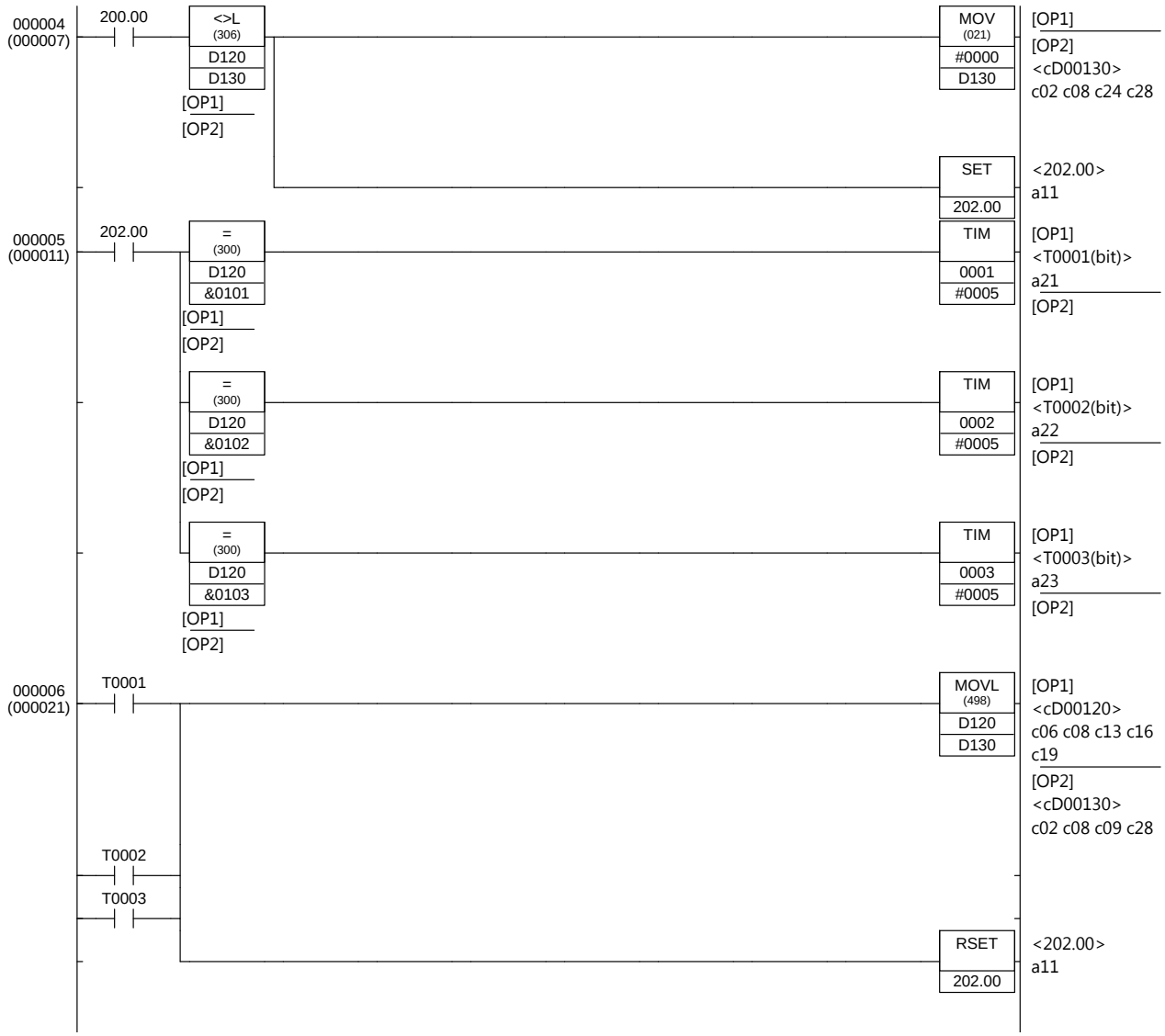


Figura 4.2: CLB - Selezione del buffer da lavorare

Il ritardo dovuto al setup è stato inserito all'interno del PLC allo scopo di aumentare le difficoltà di programmazione anche se è un dato relativo al sistema e quindi avrebbe dovuto essere presente all'interno del processo simulato in MatLab. La nostra macchina a stati continuerà dunque ad evolvere nel tempo, ma durante il tempo di setup il PLC restituisce un valore rispetto al buffer da lavorare che viene interpretato come nullo, quindi nessun buffer viene lavorato per cui in quell'arco temporale (tempo

di setup) lo storage dei magazzini aumenta.

4.2.2 Risultati Simulativi

Vediamo cosa accade per sistemi stabilizzabili di tipo PUSH con tempi di setup non trascurabili.

Domanda costante

Prendiamo in esempio un sistema costituito da 3 tipi di pezzi per cui le domande sono costanti nel tempo.

I dati caratteristici sono:

DOMANDA

- $d_1 = 5$ $[\frac{pezzi}{unita' \ di \ tempo}]$
- $d_2 = 6$ $[\frac{pezzi}{unita' \ di \ tempo}]$
- $d_3 = 6$ $[\frac{pezzi}{unita' \ di \ tempo}]$

TASSO PRODUTTIVO

- $\mu_1 = 20$ $[\frac{pezzi}{unita' \ di \ tempo}]$
- $\mu_2 = 15$ $[\frac{pezzi}{unita' \ di \ tempo}]$
- $\mu_3 = 18$ $[\frac{pezzi}{unita' \ di \ tempo}]$

STATO INIZIALE BUFFERS

- $x_1 = 15$ $[pezzi]$
- $x_2 = 16$ $[pezzi]$

- $x_3 = 4$ $[pezzi]$

Come prima verifica, dobbiamo controllare che un sistema con queste caratteristiche sia stabile. Ricordiamo che

$$STABILE \Longleftrightarrow \rho_m < 1 \; \forall \; m \; \epsilon \; M$$

e che:

$$\rho = \sum_{j-1}^{P} d_j \frac{1}{\mu_j} < \; 1 \; \forall \; t \; \in [0, T]$$

Essendo le domande ed i tassi di produzione costanti nel tempo, ne consegue:

$$\rho = \frac{5}{20} + \frac{6}{15} + \frac{6}{18} = 0.983$$

che rispetta la condizione

$$\rho < 1 \; \forall \; t \; \in [0, T].$$

Nella Figura 4.3 e Figura 4.4 sono mostrati i risultati simulativi.

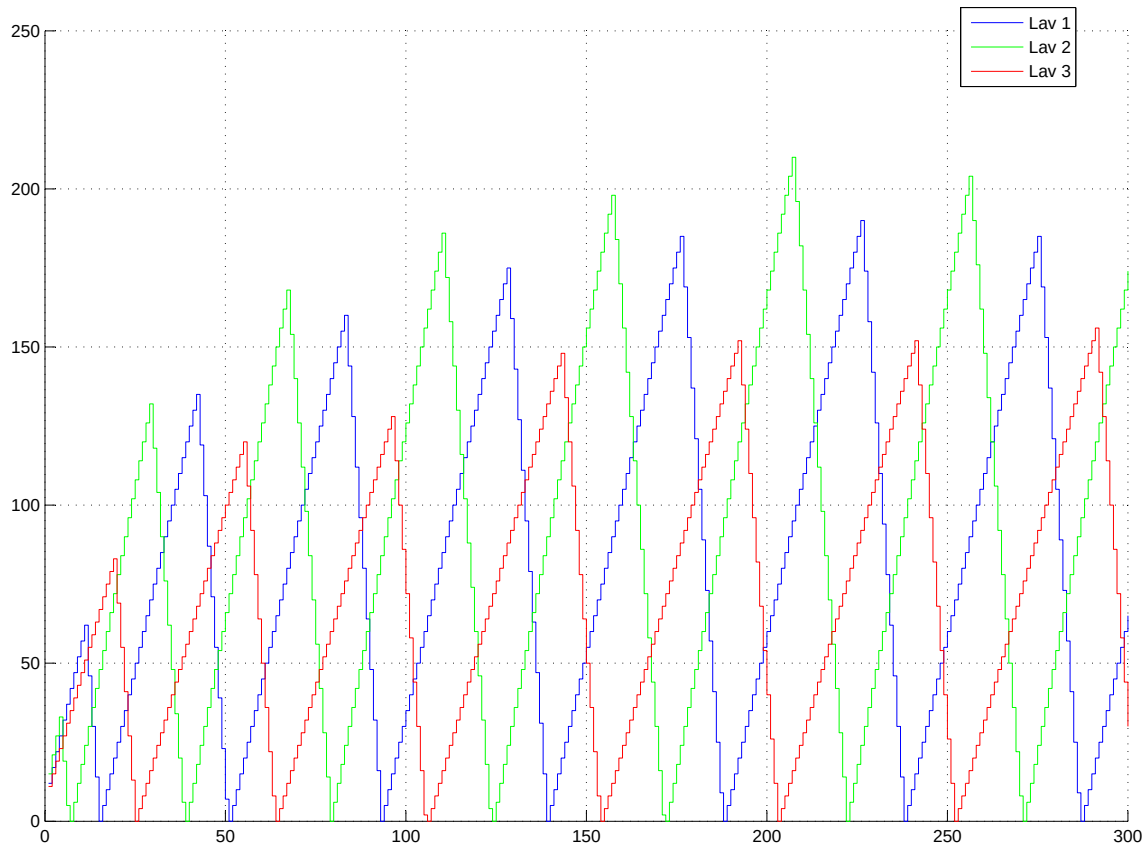


Figura 4.3: CLB - Andamento complessivo dei buffers

Possiamo notare come dopo un transitorio, in cui i buffers crescono, riusciamo a rendere stabile il sistema. Inoltre dalla Figura 4.4 viene evidenziato come durante i tempi di setup tutti e 3 i buffers crescono contemporaneamente. Solo dopo alcuni campioni, uno dei tre, quello che aveva contenuto maggiore al momento della scelta del buffer da lavorare (ovvero quando la lavorazione precedente aveva smaltito tutti i pezzi in magazzino) inizia la discesa.

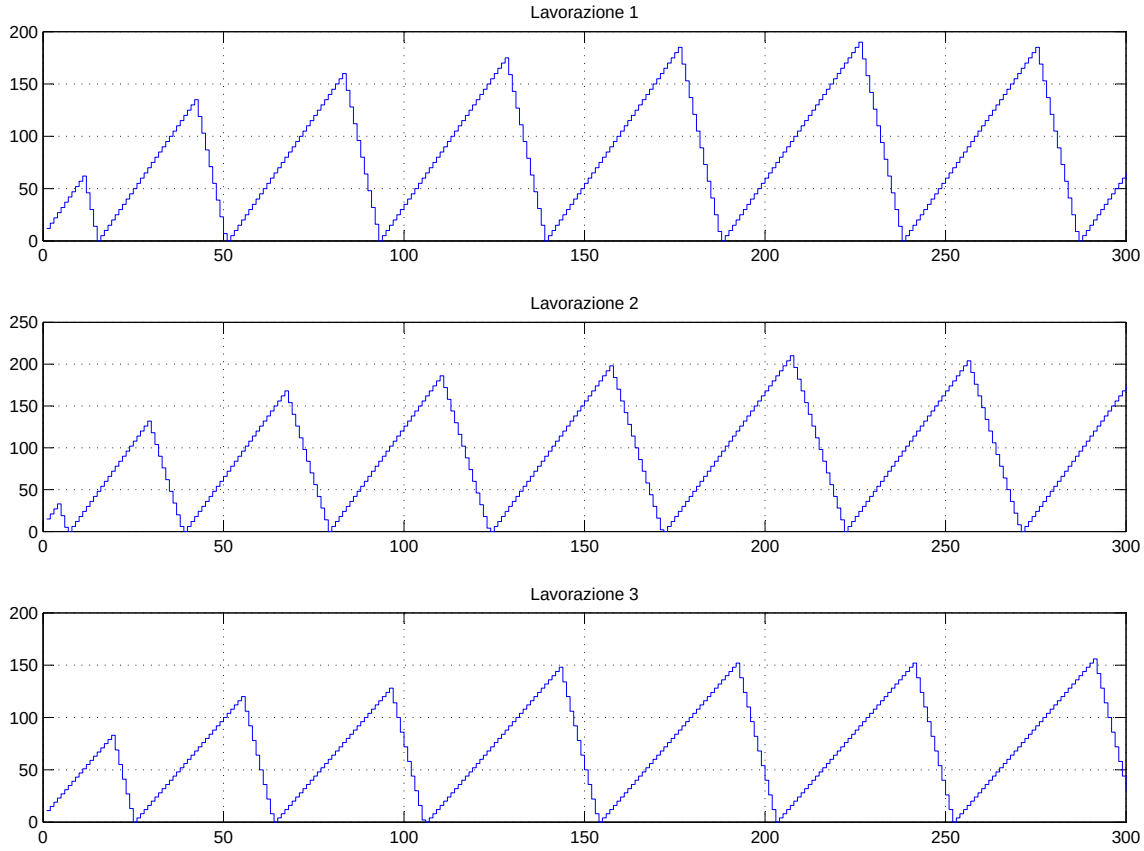


Figura 4.4: CLB - Andamento singoli buffers

Domanda variabile

Analizziamo ora se, per una domanda variabile nel tempo, la politica di controllo CLB riesce ancora a mantenere i buffers limitati.

I dati caratteristici sono:

DOMANDA

- $d_1 = 4\cos((3t) + 9)$ $[\frac{pezzi}{unita' \ di \ tempo}]$
- $d_2 = 6$ $[\frac{pezzi}{unita' \ di \ tempo}]$
- $d_3 = 6sin((7t) + 6)$ $[\frac{pezzi}{unita' \ di \ tempo}]$

TASSO PRODUTTIVO

- $\mu_1 = 35\ [\frac{pezzi}{unita'\ di\ tempo}]$
- $\mu_2 = 38\ [\frac{pezzi}{unita'\ di\ tempo}]$
- $\mu_3 = 28\ [\frac{pezzi}{unita'\ di\ tempo}]$

STATO INIZIALE BUFFERS

- $x_1 = 12\ [pezzi]$
- $x_2 = 15\ [pezzi]$
- $x_3 = 11\ [pezzi]$

Prima cosa da analizzare è se un sistema con queste caratteristiche sia stabilizzabile.

Vediamo quindi in Figura 4.5 l'andamento temporale delle domande. Ricordando dal paragrafo 3.1 la condizione di stabilità per un sistema push, possiamo notare in Figura 4.6 che l'indice ρ è sempre minore di 1, possiamo quindi confermare che ha senso tentare di controllare il sistema in oggetto tramite la politica di controllo CLB.

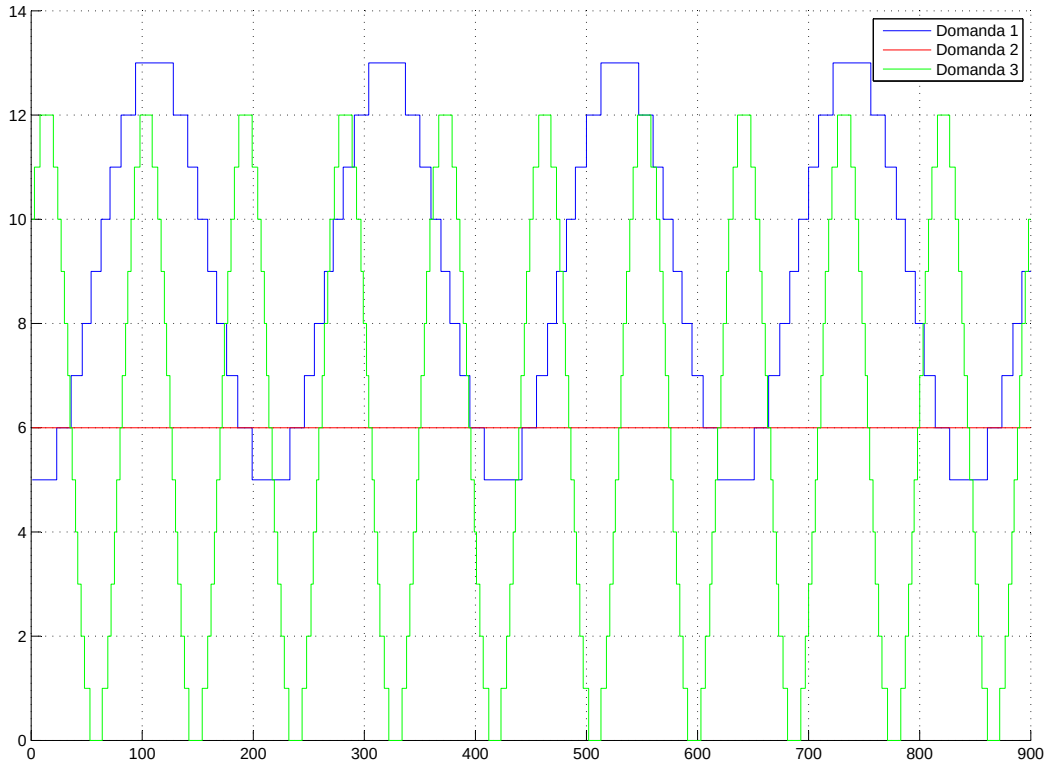


Figura 4.5: CLB - Andamento delle domande variabili nel tempo

A seguito della simulazione notiamo come gli andamenti possano risultare meno uniformi rispetto a simulazioni in cui la domanda è costante, ma riusciamo comunque a dire che i valori dei buffers restano limitati anche se hanno dei valori di punta più alti rispetto alle condizioni precedenti.

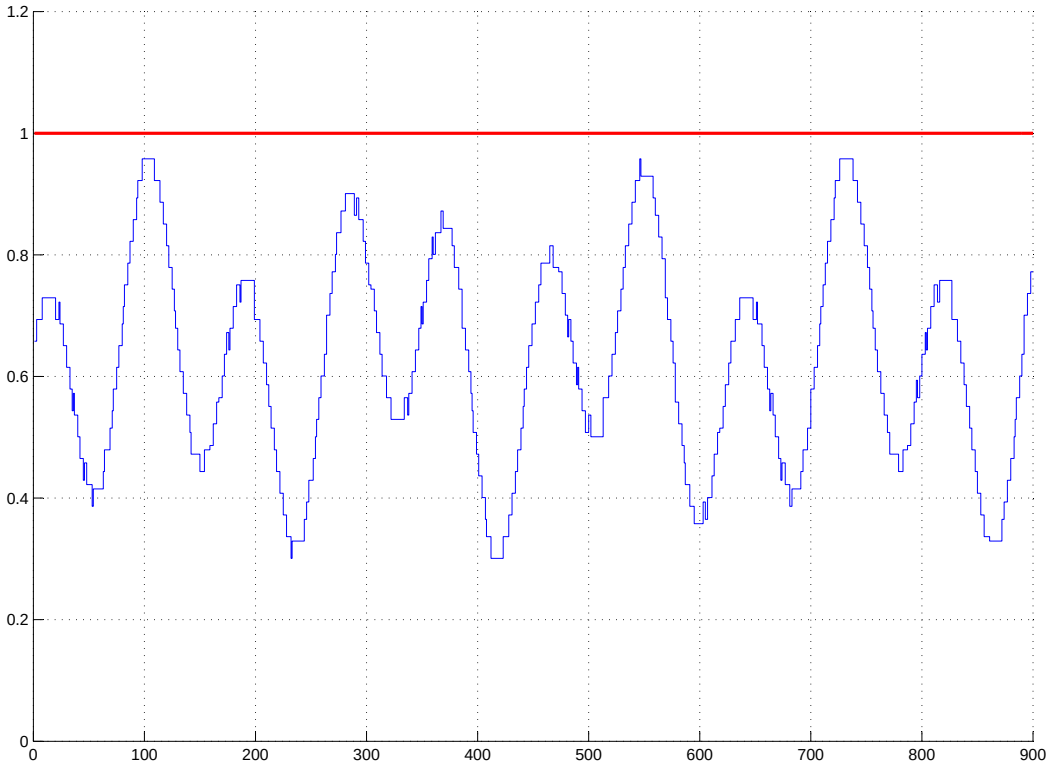


Figura 4.6: CLB - Andamento dell'indice di stabilità

4.3 Controllo sistemi PULL a tempo di setup trascurabile

Si ricorda che in un sistema PULL i buffers possono essere sia negativi sia positivi. La ricerca di una politica stabile è banale e risulta invece interessante derivare una politica che tenga conto di particolari funzioni di costo relative ai buffers. Le politiche che tengono conto delle funzioni di costo e dei loro andamenti sono di due tipi:

- **politiche miopi** che sono politiche ottime per funzioni di costo convesse
- $c\mu$ **estesa** che è una politica miope nel caso lineare.

Concetto principe di questa tipo di controllo è il confronto tra l'andamento delle singole funzioni di costo.

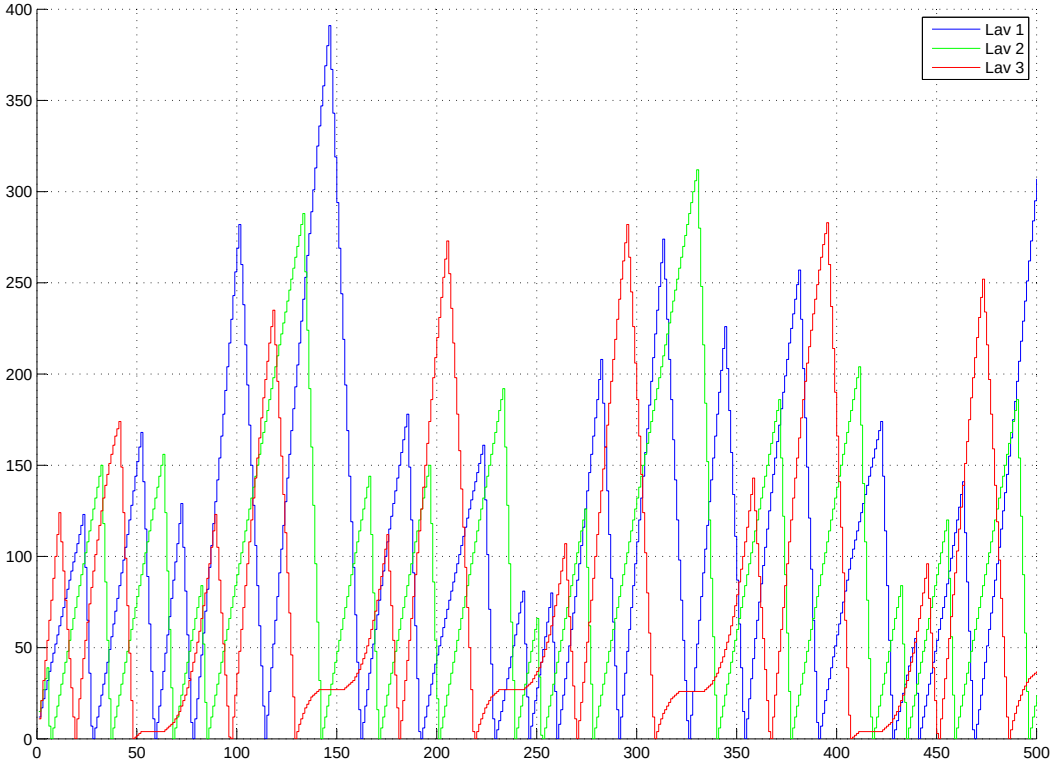


Figura 4.7: CLB - Andamento dei buffers per domande variabili

Analizziamo un sistema costituito da due soli buffers. Viene scelto, come buffer da lavorare, quello la cui derivata della funzione di costo rispetto al buffer, moltiplicata per il rispettivo tasso di produzione è minore. Questo avviene se ci troviamo nel terzo quadrante, situazione più critica, ovvero quando tutti i buffers hanno delle domande in attesa. Quando ci troviamo nel secondo e quarto quadrante avremo un buffer con delle scorte ed uno con domanda in attesa attiva, lavoreremo quindi quello con domanda in attesa.

Se ci trovassimo nel primo quadrante potremmo lasciare la macchina inattiva in quanto abbiamo scorte per tutti i buffers. Appena riusciamo a portare tutti i buffers a zero, se ci trovassimo nel caso fluido (approssimazione a tempo continuo) potremmo lavorare tutti i pezzi contemporaneamente con un tasso produttivo pari alla domanda

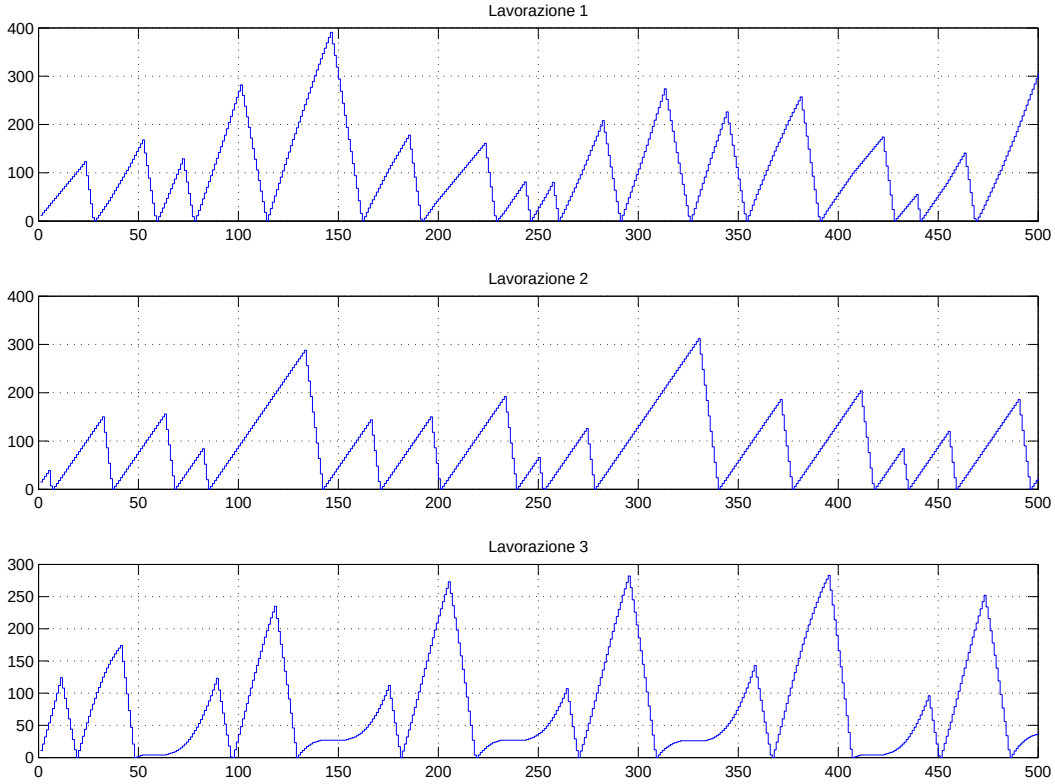


Figura 4.8: CLB - Andamento dei singoli buffers per domande variabili

stessa, cosicché possiamo mantenere nulla la funzione di costo. Poiché abbiamo deciso di simulare i sistemi il più realisticamente possibile, non possiamo fare quest'approssimazione, riusciremo quindi, solo per particolari caratteristiche del sistema e delle condizioni iniziali a portare i buffer a zero, altrimenti nella maggior parte dei casi i valori di x_1 e x_2 oscilleranno intorno ad esso.

4.3.1 Controllo Miope su PLC

Rispetto all'implementazione di un controllo CLB, dove la difficoltà è stata individuata nel fattore di sincronizzazione, ciò che complica le cose in questo tipo di controllo sono i numerosi calcoli da eseguire sul PLC. Non è infatti presente sul CJ1M una funzione che esegua la derivata, per cui dovremmo strutturare una serie di operazioni che eseguano il limite del rapporto incrementale per entrambe le funzioni di costo, e individuare in ogni istante in quale quadrante ci troviamo.

Le funzioni di costo prese in oggetto sono:

$$g(x_1) = x_1^2$$

$$g(x_2) = 2\,|x_2|^3$$

e quindi le rispettive derivate:

$$\frac{\partial g(x_1)}{\partial x_1} = 2x_1$$

$$\frac{\partial g(x_2)}{\partial x_2} = 6x_2^2 * sgn(x_2)$$

4.3.2 Risultati Simulativi

Senza guasti

Ipotizziamo per il primo tipo di simulazione che il sistema con le seguenti caratteristiche non sia soggetto a guasti.

DOMANDA

- $d_1 = 1$
- $d_2 = 1$

TASSO PRODUTTIVO

- $\mu_1 = 2$
- $\mu_2 = 3$

Di un sistema con queste caratteristiche possiamo subito dire che è stabilizzabile Le domande ed i tassi di produzione sono costanti nel tempo, ρ non varierà il suo valore nel tempo per cui basterà calcolarlo una sola volta.

$$\rho = \frac{1}{2} + \frac{1}{3} = 0.833$$

Verificato che il sistema è stabile ($\rho \leq 1$), vediamo una simulazione in cui ci troviamo in un caso critico, ovvero entrambi i buffers sono negativi.

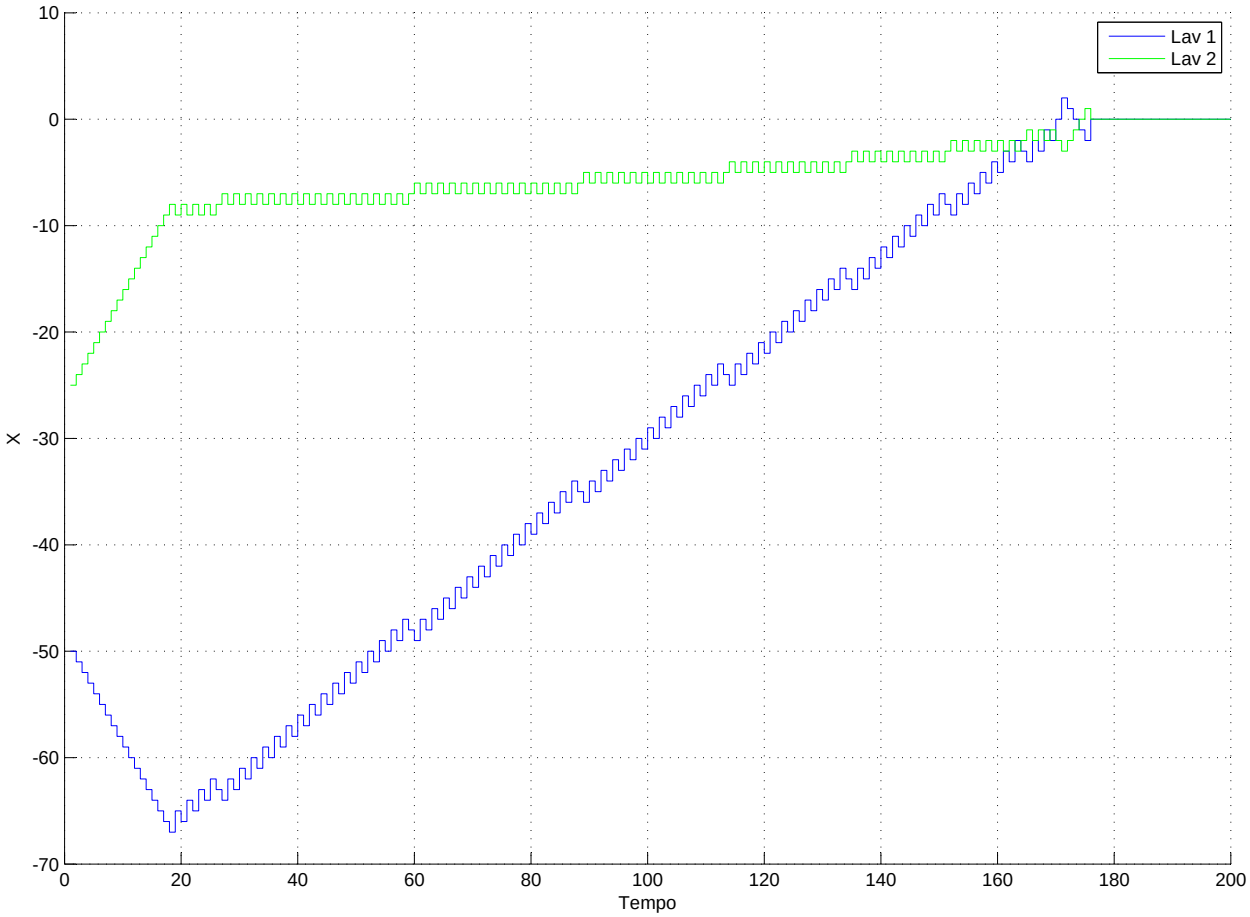


Figura 4.9: Miope - Andamento dei buffers senza guasti

Come possiamo osservare dal diagramma di stato in Figura 4.11 la scelta delle lavorazione segue proprio le curve costruite dal tasso di produzione per la derivata della funzione di costo.

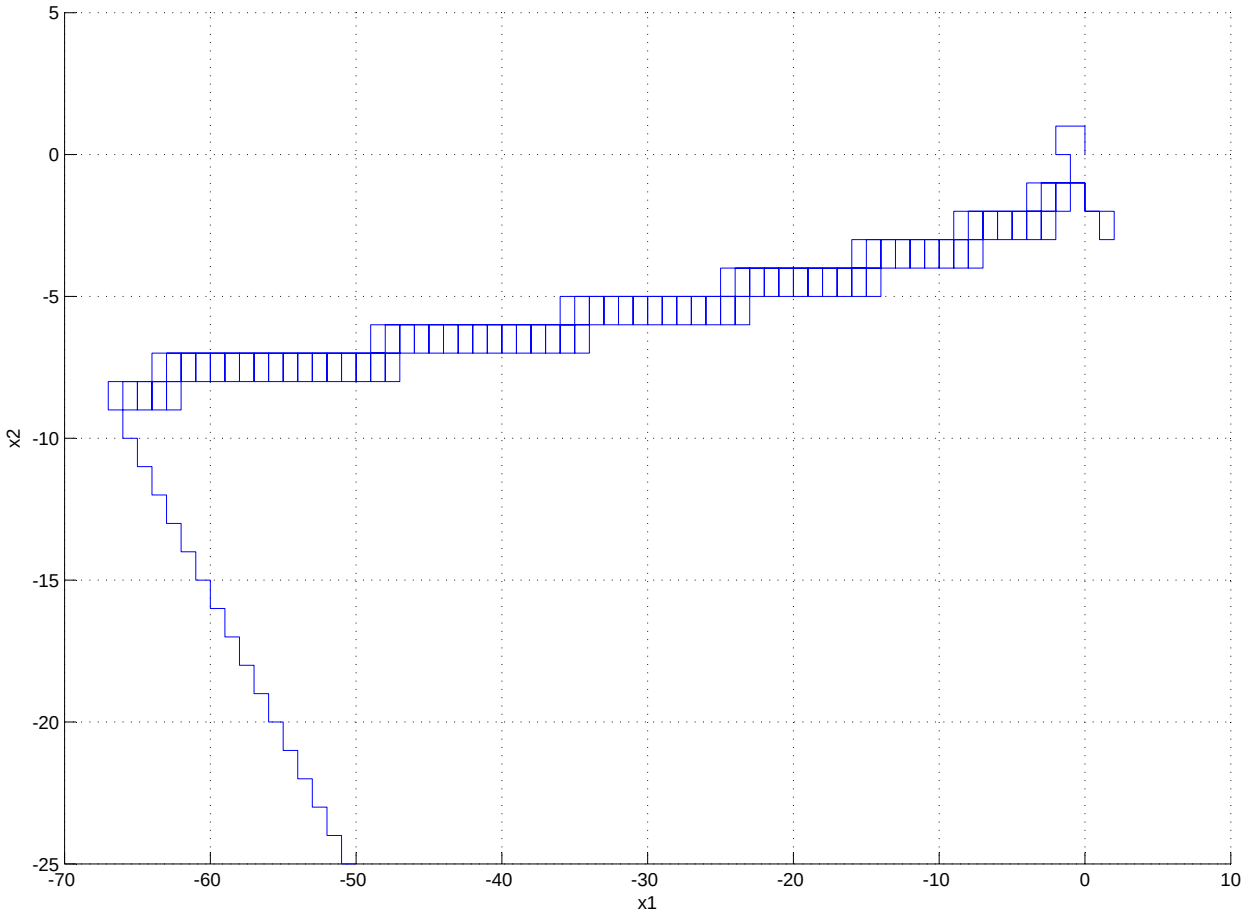


Figura 4.10: Miope - Diagramma di stato Discreto per sistema senza guasti

Infine possiamo (Figura 4.12) come questa politica tende ad avvicinare tra loro le curve dei costi per poi portarle entrambe a zero quasi contemporaneamente. Anche il grafico delle curve delle derivate dei costi per rispettivo tasso di produzione (Figura4.13)ha un andamento simile a quello dei costi, tendono avvicinarsi ed andare a zero contemporaneamente.

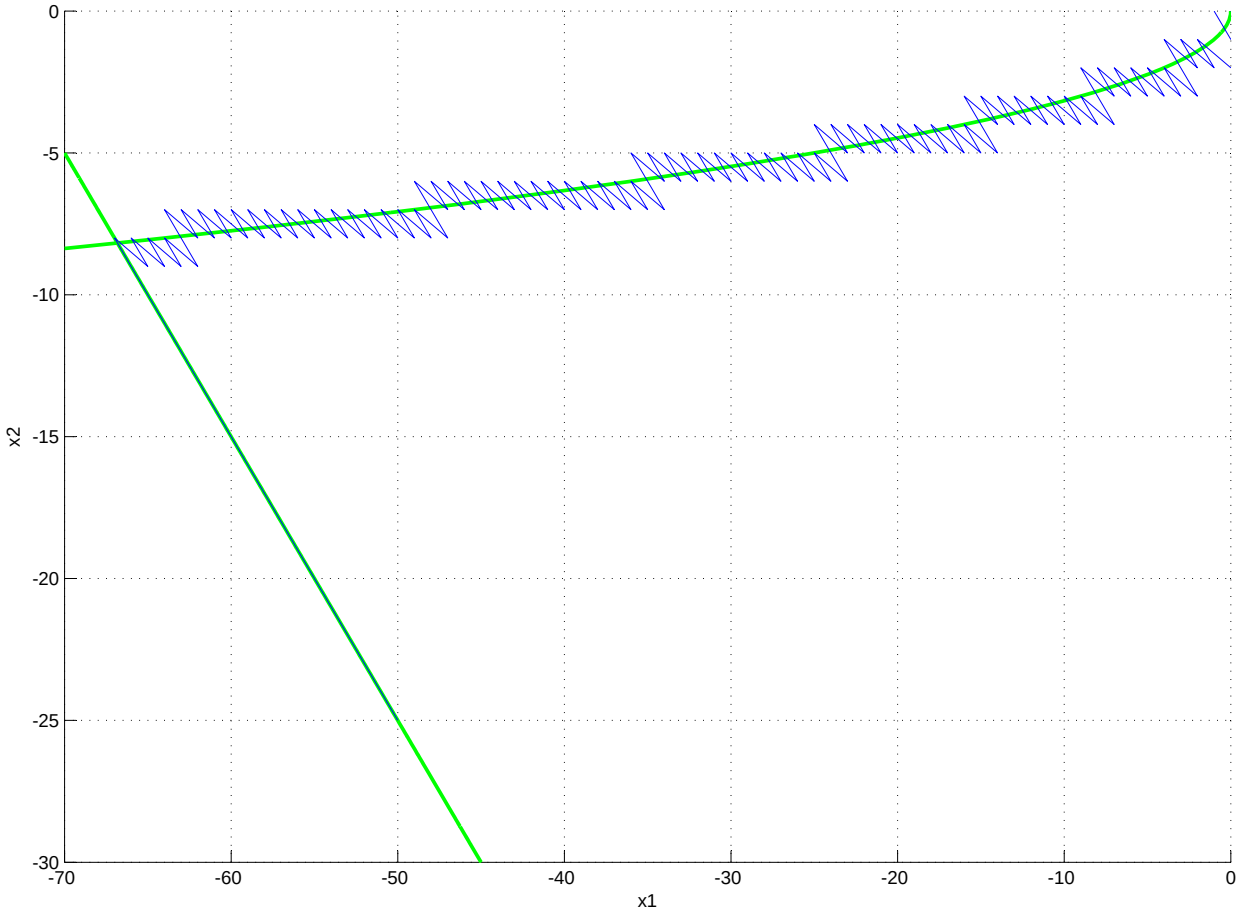


Figura 4.11: Miope - Confronto Diagramma di stato e Andamento dei costi

Parallelamente osserviamo una simulazione in cui non sia validata l'ipotesi di lavorazione simultanea e quindi non è possibile approssimare in parziali lavorazioni. Notiamo come in questo caso (Figura 4.14), la simulazione si avvicini sempre più alla realtà. Il sistema inizia ad oscillare il valore dei buffer attorno allo zero, senza riuscire a rimanerci, in quanto non è possibile lavorare un pezzo per una frazione di lavorazione, o lo si lavora tutto o niente.

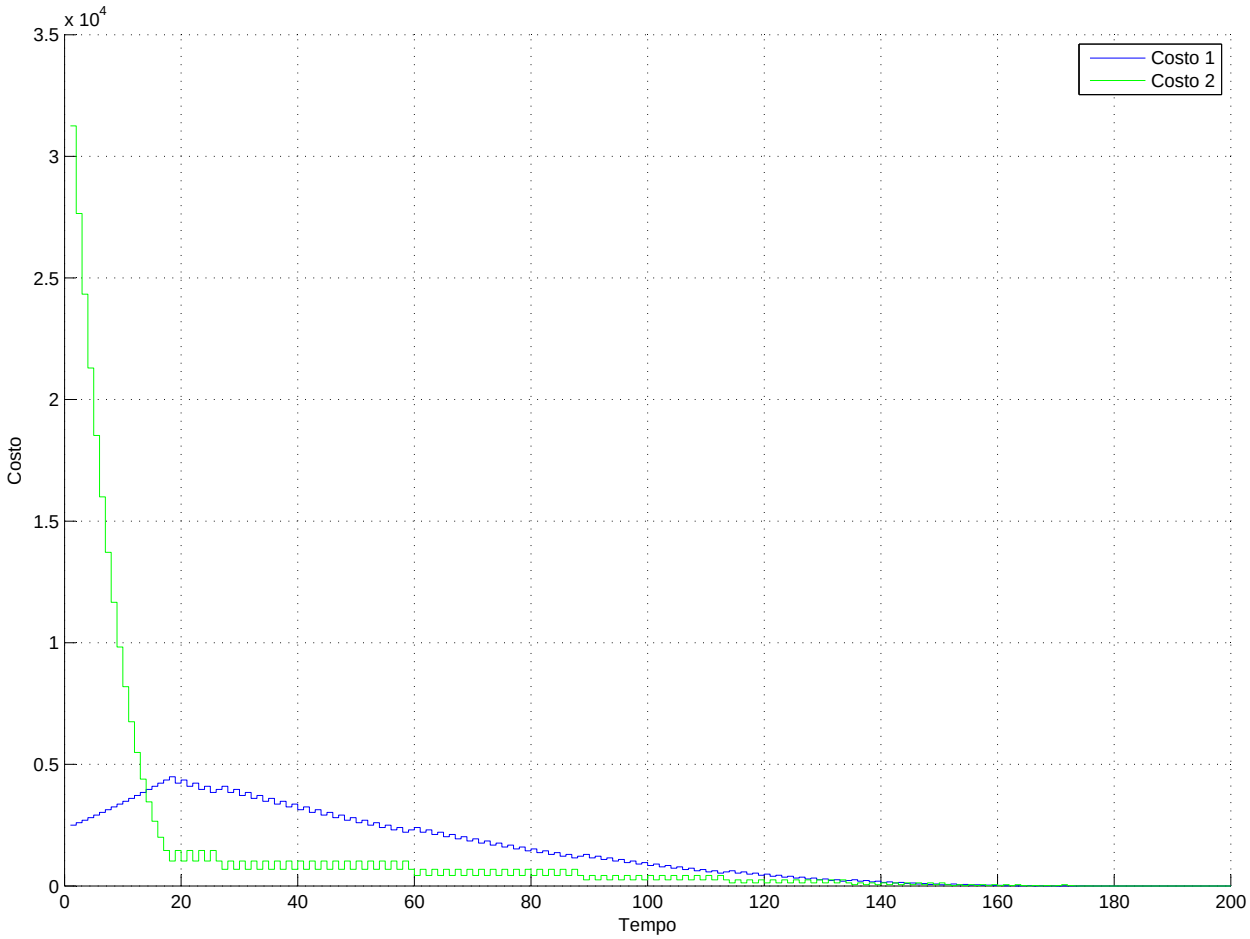


Figura 4.12: Miopi - Andamento dei costi

Con guasti casuali

Ora generiamo sulla nostra macchina a stati dei guasti random che possono incidere sull'attività della macchina in oggetto. Questi guasti fermeranno la macchina per un tempo anche esso random e diverso per ogni guasto. Supponiamo inoltre una frequenza di guasto del $1,5\%$.

I guasti sono stati generati tramite MatLab. Abbiamo creato una funzione che dati in ingresso il tempo di simulazione ed una percentuale di guasto, in maniera random, genera un numero di guasti pari alla percentuale data.

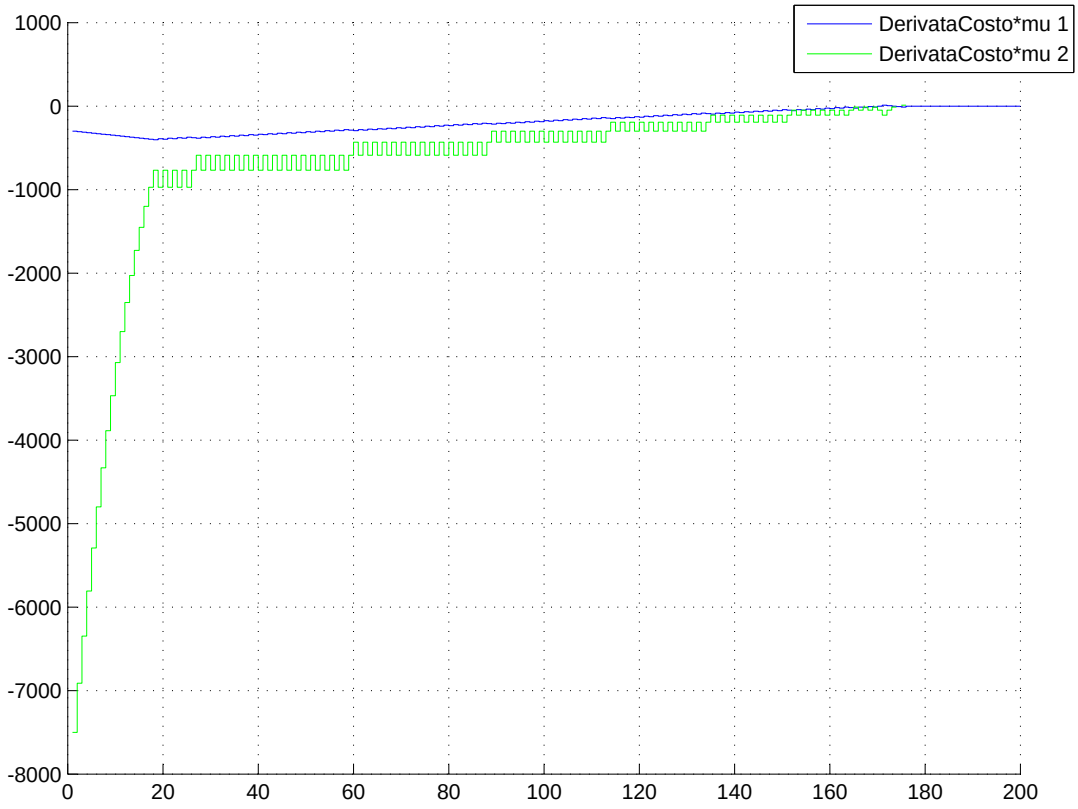


Figura 4.13: Miopi - Andamento delle derivate dei costi per tasso produttivo

La durata di ogni guasto viene stimata anch'essa in modo random. Calcolando il tempo che intercorre tra un guasto ed il successivo, il tempo di guasto non sarà mai superiore a quest'ultimo, così da non far sì che due guasti possano accavallarsi.

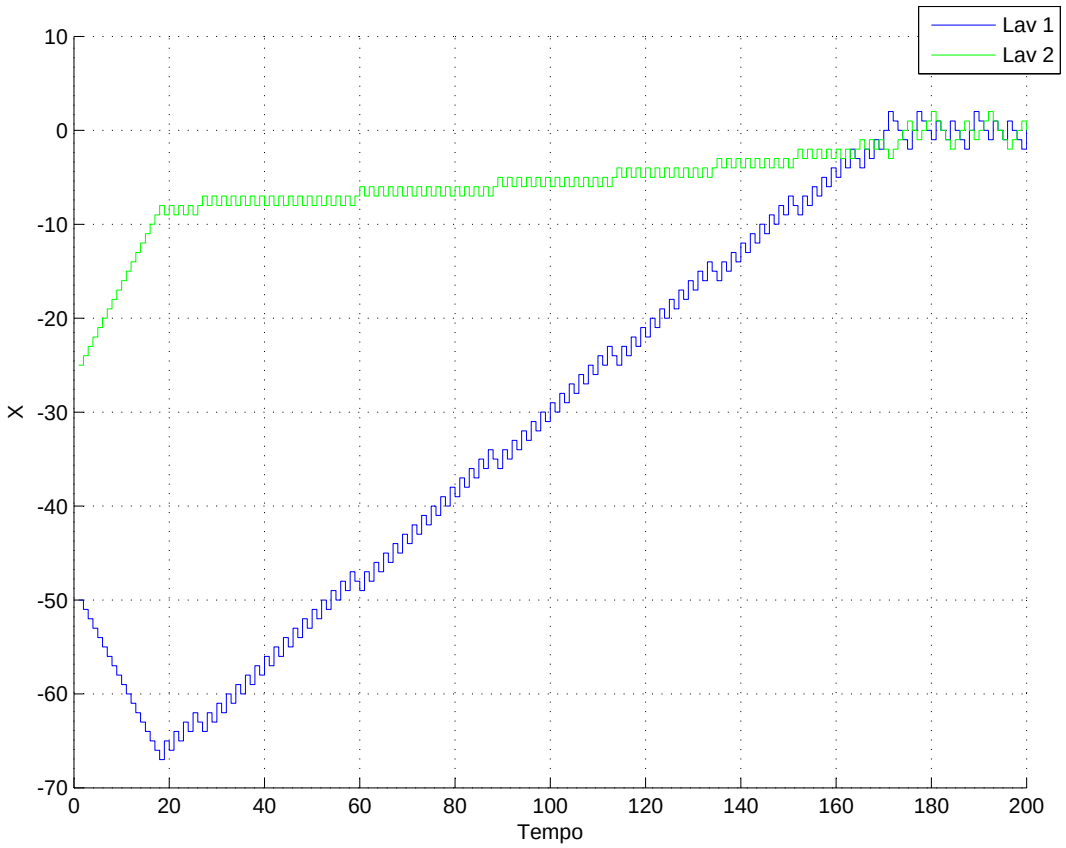


Figura 4.14: Miopi - Andamento buffers senza ipotesi di lavorazione simultanea

Tramite il codice in Figura 4.15 generiamo quindi un array di istanti di guasto che useremo per mettere fuori uso il sistema di controllo e di conseguenza i buffers aumenteranno il loro contenuto secondo il tasso di domanda senza poter smaltire alcun pezzo, in quanto durante gli istanti di guasto i tassi di produzione saranno nulli.

```
ts=400;   % NUMERO DI ITERAZIONI

pg=0.015;  % percentuale di guasto
ng=round(ts*pg);  % quantità di guasti
guasto=(sort(randi([2 ts],ng,1)))';  %definizione dei tempi di guasto
g=zeros(1,ng);
tguasto=zeros(1,1);
tguastomax=15;

for i=2:ng
    g(i-1)=min(guasto(i)-guasto(i-1),tguastomax);
end
g(ng)=min(ts-guasto(ng),tguastomax);

for i=1:ng
    tgs=randi([1 g(i)]-1,1);
    for j=1:tgs
        tguasto(end+1)=guasto(i)+j-1;
    end
end
```

Figura 4.15: MatLab - Codice per generare guasti

Notiamo come in presenza di guasti, il sistema cerchi di portare a 0 i buffers, ma ad ogni guasto i buffers diminuiscono nuovamente aumentando la domanda in attesa.

Inoltre, anche con una percentuale di guasto molto piccola, i costi variano molto creando quindi un danno non trascurabile al sistema produttivo (Figura 4.17). In realtà i guasti o la manutenzione sono una motivazione per cui in pratica abbiamo interesse nel saper come portar a zero in modo ottimo i buffers.

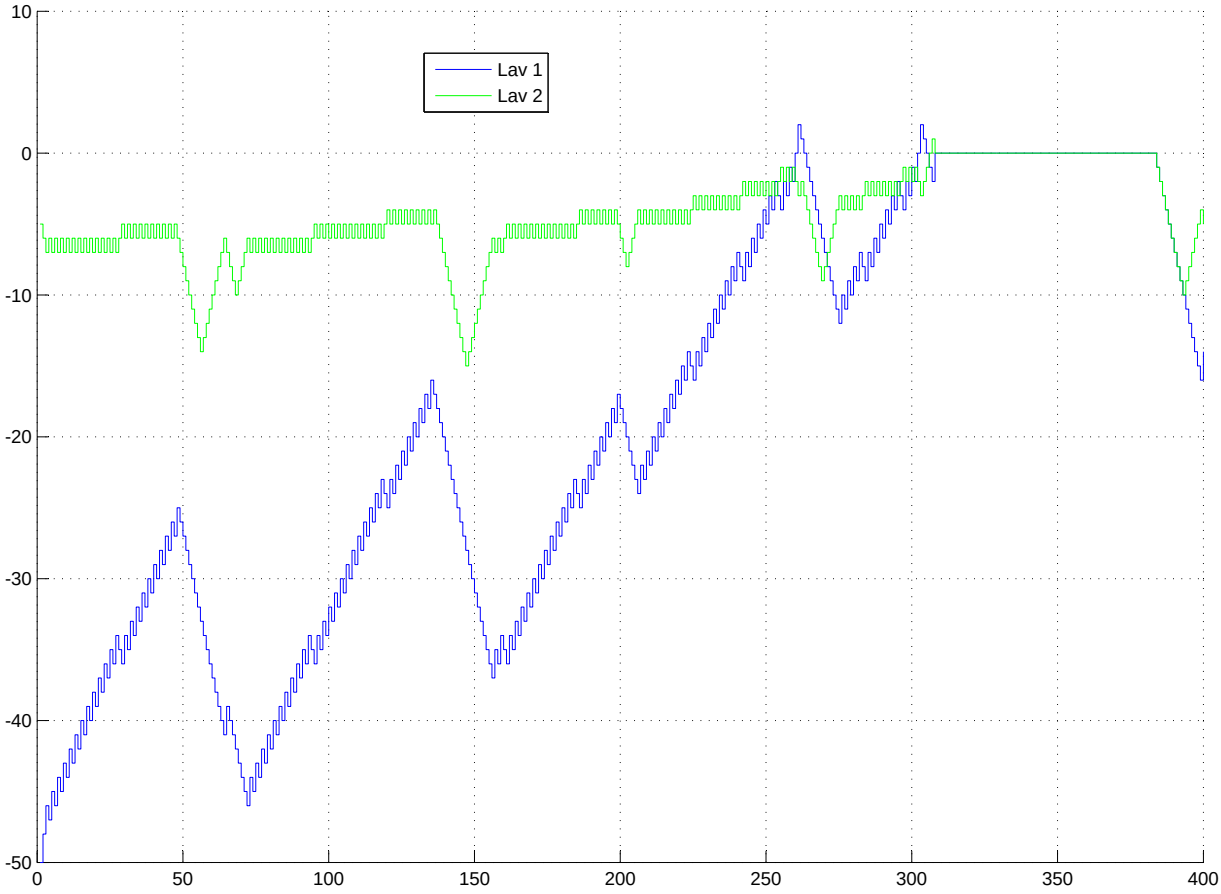


Figura 4.16: Miopi - Andamento dei buffers per sistemi con guasti

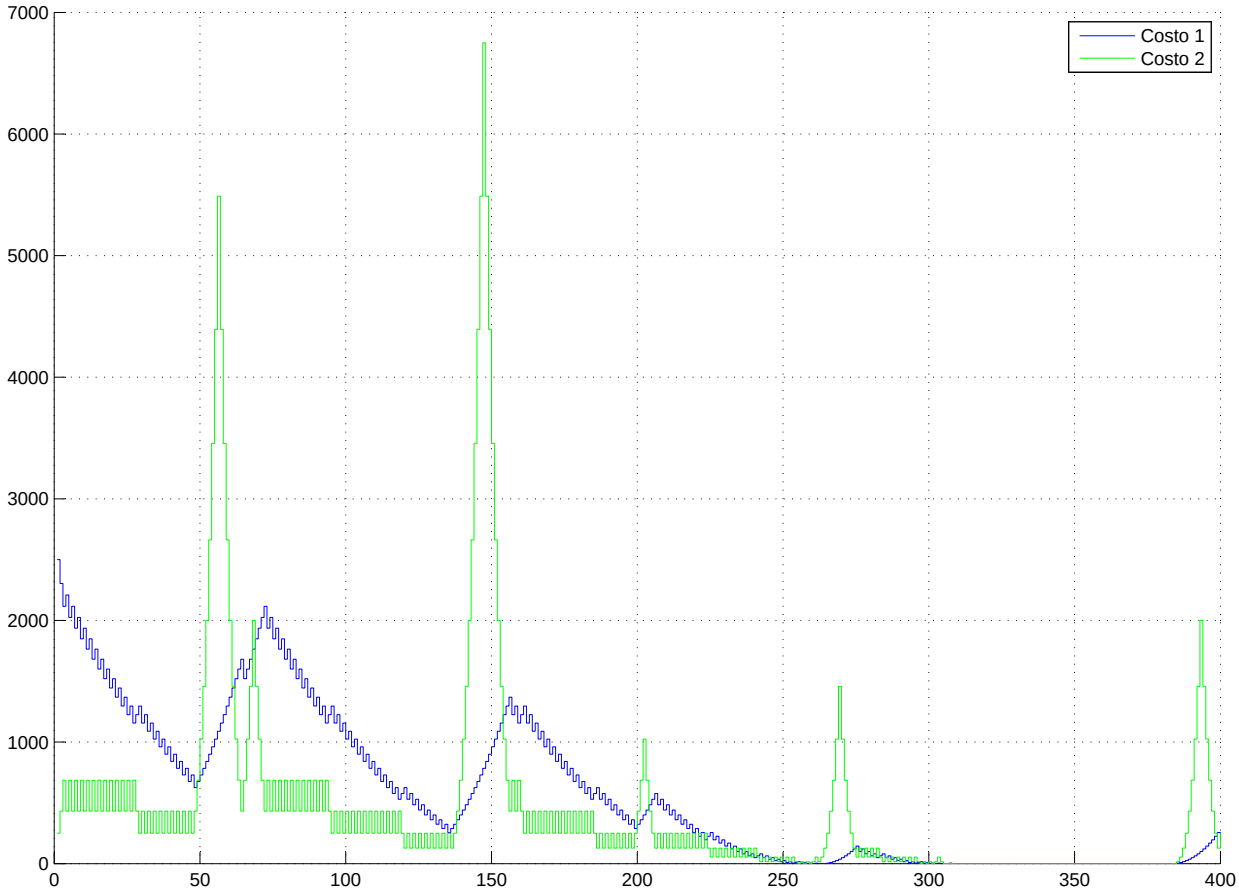


Figura 4.17: Miopi - Andamento dei costi per sistemi con guasti

Capitolo 5

Conclusioni e sviluppi futuri

Quest'elaborato è molto utile per chi inizierà a muovere i primi passi con i PLC, in particolare gli Omron. Non vuole però sostituirsi a manuali e datasheet che sono comunque indispensabili per la buona preparazione e conoscenza di un qualsiasi dispositivo. E' molto utile osservare come tramite una buona ed accurata preparazione sulle tipologie di sistemi comunicativi si possano far dialogare dispositivi di natura diversa, ed in particolar modo come si possano simulare sistemi tramite MatLab e controllarli real-time tramite un dispositivo quale il PLC.

Uno dei possibili sviluppi futuri sarà interno all'università, e permetterà alle prossime classi di studenti di poter utilizzare questi strumenti come base necessaria per la programmazione ed implementazione di nuovi controlli su questo tipo di PLC. Sono state infatti costruite delle funzioni per la costruzione automatica di segnali seriali per le funzioni più utilizzate dal PLC, che saranno messe a disposizione per facilitare chiunque voglia conoscere più a fondo questi dispositivi che regolano l'intero mondo dell'automazione nei settori produttivi.

Un altro possibile lavoro che può seguire è l'analisi di questi sistemi per capire effettivamente gli andamenti delle funzioni di costo e tramite reti neurali poter prevedere le perdite derivanti da possibili guasti aumentando le scorte di magazzino solo

nei periodi che precedono il guasto.

Appendice A
Funzioni MatLab

Scrivere da MatLab su PLC

```
function out=write_PLC(where,header,from_word,what)

% SCRITTURA SU PLC prende in ingresso 2 stringhe di caratteri che sono
% rispettivamente -> header code, from_word ed un int -> what)

at='@00';

n=length(what);

for i=1:n
    if what(i)<0
        what(i)=65536+what(i);
    end
end

whathex=dec2hex(what,4);
[r c]=size(whathex);
what_s='';
whathex=whathex';

for i=1:(r*c)
    what_s=strcat(what_s,whathex(i));
end

comando=strcat(at,header,from_word,what_s);

fcs=fcs_calculation(comando);
termination='*';

comando=strcat(comando,fcs,termination);

fprintf(where,comando);
out= fscanf(where);
```

Leggere da MatLab su PLC numeri interi con segno

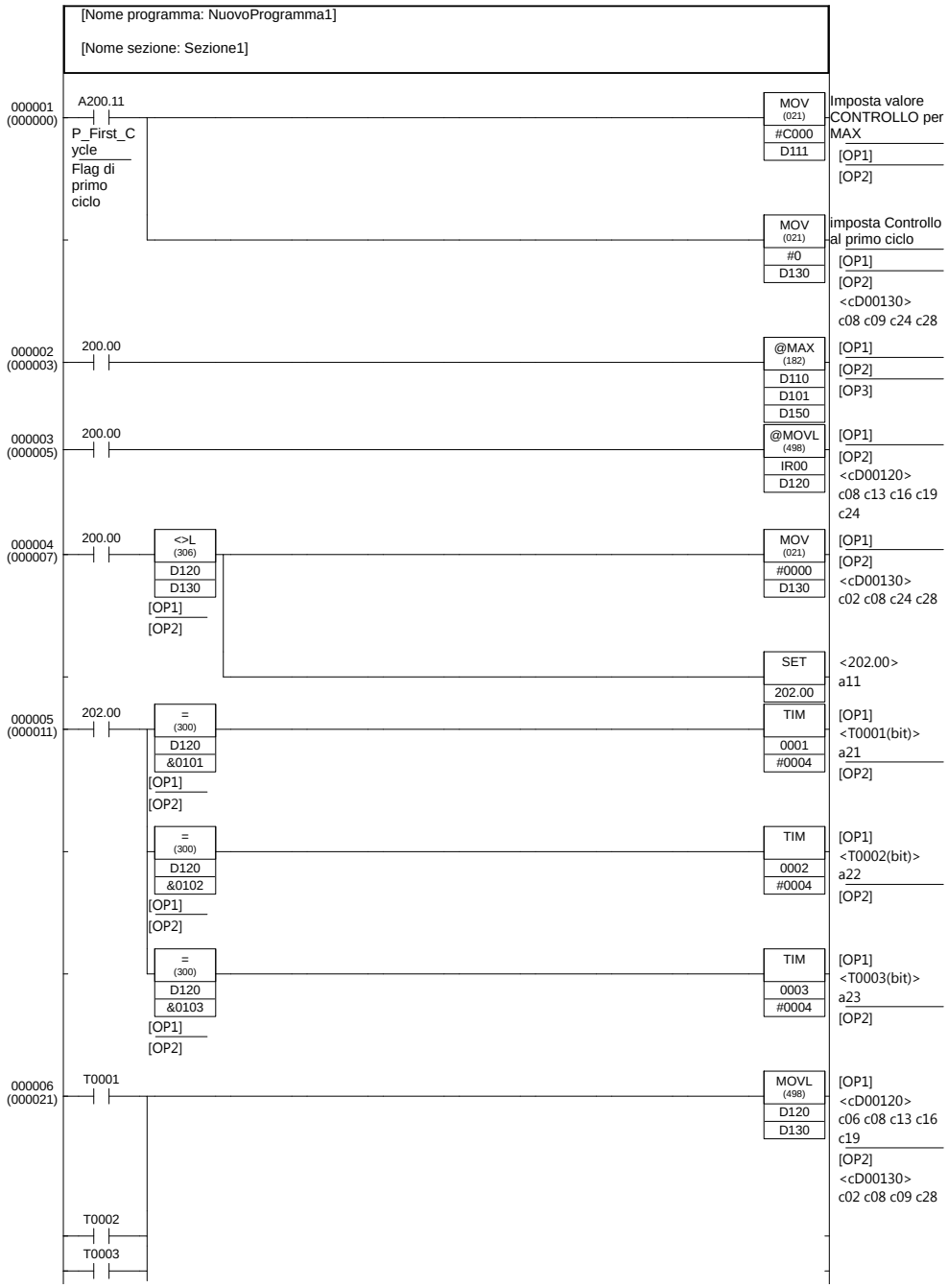
```
function out=read_PLC(where,header,from_word,how_many)

% LETTURA DALL AREA PLC

at='@00';

how_many_s=dec2hex(how_many,4);

comando=strcat(at,header,from_word,how_many_s);

fcs=fcs_calculation(comando);
termination='*';

stato=strcat(comando,fcs,termination);

fprintf(where,stato);
readed= fscanf(where);

readed=readed(8:end-4);

read=zeros(how_many,1);

for i=1:how_many

    read(i,1)=hex2dec(readed(((i*4)-3):(i*4)));

end

out=read;
```

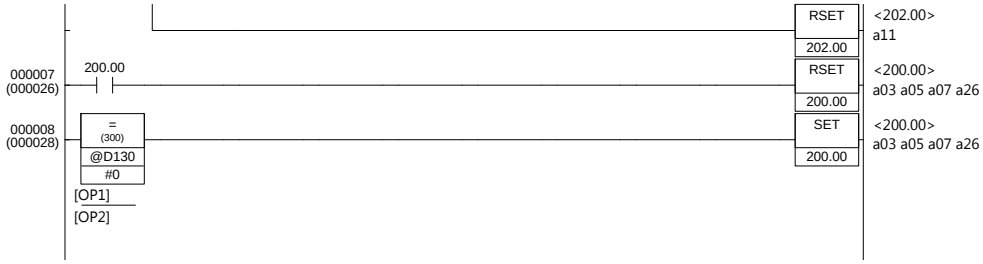
Calcolare FCS

```
function fcs=fcs_calculation(serial_in)

n=length(serial_in);
binary=zeros(n,8);

for i=1:n
    temp=dec2bin(serial_in(i),8);
    for j=1:8
        binary(i,j)=double(temp(j))-48;
    end
end

[r_max c_max]=size(binary);

fcs=binary(1,:);

for c=1:c_max
    for r=1:r_max-1
        fcs(1,c)=xor(fcs(1,c),binary(r+1,c));
    end
end

fcs1=0;
fcs2=0;

fcs1=(fcs(1)*2^3)+(fcs(2)*2^2)+(fcs(3)*2^1)+(fcs(4)*2^0);
fcs2=(fcs(5)*2^3)+(fcs(6)*2^2)+(fcs(7)*2^1)+(fcs(8)*2^0);

fcs1=dec2hex(fcs1);
fcs2=dec2hex(fcs2);

fcs1=num2str(fcs1);
fcs2=num2str(fcs2);

fcs=strcat(fcs1,fcs2);
```

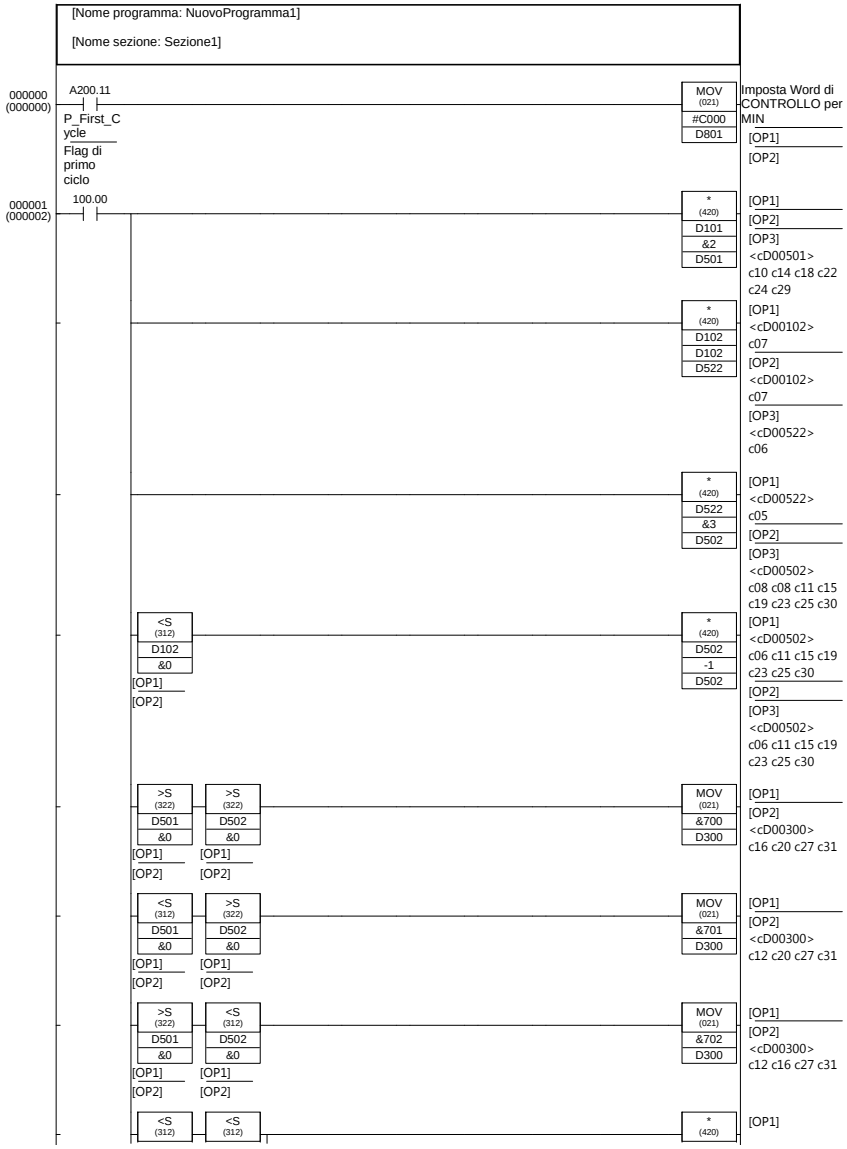
Appendice B
Programmazione PLC

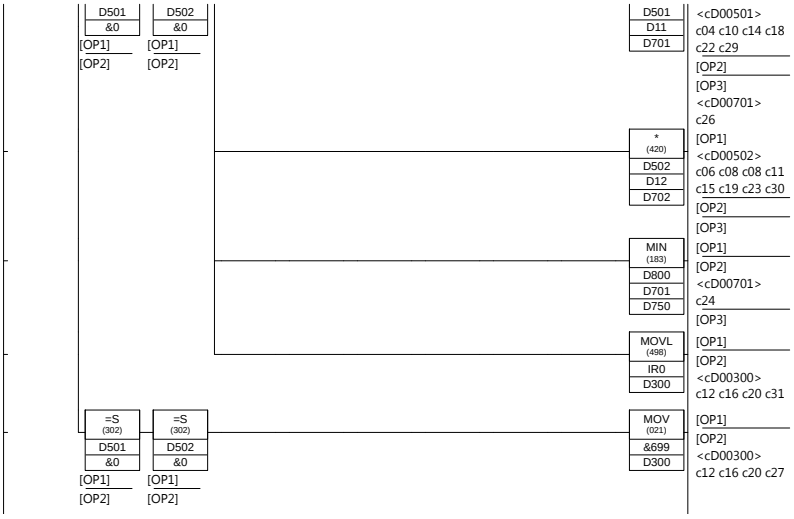
Implementazione CLB





Implementazione Politica Miope




D501
&0
[OP1]
[OP2]
D502
&0
[OP1]
[OP2]
D501
D11
D701
<cD00501>
c04 c10 c14 c18
c22 c29
[OP2]
[OP3]
<cD00701>
c26
*
(420)
D502
D12
D702
[OP1]
<cD00502>
c06 c08 c08 c11
c15 c19 c23 c30
[OP2]
[OP3]
MIN
(183)
D800
D701
D750
[OP1]
[OP2]
<cD00701>
c24
[OP3]
MOVL
(498)
IR0
D300
[OP1]
[OP2]
<cD00300>
c12 c16 c20 c31
=S
(302)
D501
&0
[OP1]
[OP2]
=S
(302)
D502
&0
[OP1]
[OP2]
MOV
(021)
&699
D300
[OP1]
[OP2]
<cD00300>
c12 c16 c20 c27

Elenco delle figure

Bibliografia

[1] Massimo Barezzi, *"PLC Controllori Logici Programmabili"*, Editrice SAN MARCO, 2009.

[2] Francesco Martinelli, *"Problemi di controllo per sistemi di produsione"*, Dispense, 2008.

[3] Omron, *"W342 - Communication Command"*, Reference Manual.

[4] Omron, *"W394 - Programmable Controllers"*, Programming Manual.

[5] Francesco Martinelli, Chang Shu, and James R. Perkins, *"IEEE Transactions on automatic control, Vol.46"*, 8 August 2001.

[6] James R. Perkins and P. R. Kumar, *"IEEE Transactions on automatic control, Vol.34"*, 2 February 1989.

[7] S. Connolly, Y. Dallery, and S. B. Gershwin, *"A real time policy for per-forming setup changes in a manufacturing system, in Proc. 31st IEEE Conf. Decision Control, vol. 1"*, Dec. 1992.

[8] A. Y. Ha, *"Optimal dynamic scheduling policy for a make-to-stock production system, Oper. Res., vol. 45"*, Jan./Feb. 1997.

[9] J. R. Perkins, C. Humes, Jr., and P. R. Kumar, *"Distributed scheduling of flexible manufacturing systems: Stability and performance, IEEE Trans. Robot. Automat., vol. 1"*, Apr. 1994.