Politecnico di Torino

Diploma Universitario

in Ingegneria Informatica

Tirocinio

Enrico D'Oro

Progetto e scrittura di applicativi per macchine di collaudo

Stage presso SEICA S.p.A. (luglio – settembre 2018)

Indice

Capitolo 1

Introduzione

Questa introduzione presenta l'azienda Seica dal punto di vista dei prodotti, mercati, soluzioni e servizi, evidenziando il contesto lavorativo nel quale sono stato inserito e in particolare in quale ambito sono state richieste le mie capacità. Vengono mostrati i compiti svolti durante il periodo formativo e le persone che mi hanno guidato e seguito nella realizzazione di un nuovo progetto.

1.1 Informazioni sull'azienda

Fondata nel 1986, Seica S.p.A. è fornitore globale di apparecchiature di test automatico e sistemi di saldatura selettiva, con una base installata di oltre 1900 sistemi in quattro diversi continenti.

1.1.1 Tipo di azienda

Seica si rivolge principalmente a tre gruppi di clienti, militare-aerospaziale, produzione di elettronica industriale e produzione di circuiti stampati, che operano in settori diversi. Tra i clienti operanti nel settore industriale, molti sono *OEM (Original Equipment Manufacturers)* e si rivolgono a mercati quali: automazione industriale, autonica, sicurezza, elettronica di consumo, telecomunicazioni, trasporti, riscaldamento/refrigerazione e medicale.

1.1.2 I prodotti Seica

Seica offre una gamma propria di soluzioni di test che permettono di coprire tutte le richieste che provengono dai settori produttivi. Cinque linee di test, *Pilot*, *Compact*, *Valid*, *Rapid* e *Mini*, una linea di sistemi di saldatura selettiva laser, *Firefly*, e una linea di sistemi per l'ispezione ottica automatica, *Dragonfly*.

Tutti i sistemi Seica hanno una piattaforma hardware e software comune, *VIVA Integrated Platform (VIP).*

1.2 Organizzazione del lavoro

L'azienda è organizzata in tre divisioni interne: *Seica Test Solutions*, *Seica Productronics* e *Seica Smart Buildings & Facilities*.

La divisione cui ho contribuito col mio lavoro è la *Seica Test Solutions*, che si occupa della progettazione, dell'industrializzazione, della produzione e dell'assistenza di tutto ciò che riguarda le tecnologie e i sistemi del test elettrico.

1.3 Palinsesto quotidiano dell'attività lavorativa

L'attività lavorativa quotidiana prevedeva la programmazione di applicazioni utilizzando un linguaggio di programmazione orientato agli oggetti (C#), l'eventuale test di queste applicazioni tramite altrettante applicazioni di emulazione da me programmate secondo le specifiche del cliente e dell'azienda e, nel periodo conclusivo dell'attività formativa, il test definitivo di questi applicativi installandoli sulle macchine di collaudo della Seica.

Inoltre, settimanalmente, erano previste due riunioni: una interna all'azienda, per aggiornamenti riguardanti i diversi progetti in corso, e una in compresenza dell'azienda richiedente il progetto cui ho partecipato, per discutere su aggiornamenti, chiarimenti e migliorie da apportare.

1.4 Le persone

Lo stimolante luogo di lavoro ha portato a confrontarmi con diversi membri del personale Seica. In particolar modo ho lavorato a stretto contatto con un *Software Architect & Developer* dell'azienda, Luca Covolo, che mi ha presentato il progetto cui avrei preso parte, guidandomi nel corso del suo sviluppo.

Ho avuto inoltre l'occasione di partecipare attivamente a una delle conference call tra lo staff Seica che ha lavorato al progetto e il cliente esterno che lo ha richiesto.

Capitolo 2

Presentazione del progetto

In questo capitolo viene presentato il progetto vero e proprio cui ho preso parte, esponendo le richieste del cliente, descrivendone le caratteristiche principali. Conseguentemente viene mostrata la soluzione elaborata da Seica e sviluppatasi nel corso del tirocinio per interfacciare le proprie macchine in modo congruo alle richieste dell'azienda cliente, senza perdere di efficacia.

2.1 Descrizione del progetto

Seica ha inserito le proprie macchine e le proprie competenze all'interno del progetto attenendosi alle richieste del cliente. Il compito di Seica è quello di verificare il corretto assemblaggio e funzionamento delle schede elettroniche prodotte dal cliente, utilizzando delle proprie macchine di collaudo.

Il progetto richiesto dall'azienda cliente integra tecnologie proprie dell'*Industry 4.0* che passa per il concetto di *smart factory*.

2.1.1 Il MES

Caratteristica dell'*Industry 4.0* e uno dei requisiti per il progetto voluto dal cliente è la comunicazione con il *MES (Manufacturing Execution System)* ossia un sistema informatizzato che ha la principale funzione di gestire e controllare la funzione produttiva di un'azienda. Il MES opera in tempo reale per consentire il controllo di più elementi nel processo di produzione.

In questo progetto la comunicazione col MES avviene tramite la suite iTAC. Grazie a questa comunicazione, Seica è in grado di sapere, in tempo reale, che tipo di schede sta collaudando, se sono già state sottoposte a dei test in precedenza e, una volta terminati i test, comunicare lo stato della scheda al cliente.

2.1.2 Specifiche

Su richiesta del cliente, le loro schede devono essere sottoposte a tre tipi diversi di test tramite tre apposite macchine di collaudo Seica. Tuttavia, queste tre macchine devono apparire al cliente come un'isola governabile da un unico computer *supervisor*. Sia il supervisor che le tre macchine di collaudo sono collegate a una rete LAN interna all'azienda, tramite la quale è possibile comunicare con il MES.

La scheda viene inserita nella prima macchina e automaticamente viene eseguito il primo test. Una volta terminato, sempre in autonomia, la scheda passa nella seconda macchina, che esegue il test e fa proseguire la scheda verso la terza e ultima macchina. Questa esegue il suo test e infine espelle la scheda che, automaticamente, grazie al colloquio col MES, viene smistata tra quelle che hanno passato tutti i test o quelle che ne hanno fallito almeno uno.

2.1.3 Le schede nel dettaglio

Le schede elettroniche prodotte dal cliente sono così costituite: si tratta di pannelli elettronici su cui sono montate 6 schede. Il numero di schede tuttavia è indicativo: infatti nulla vieta che il numero di schede possa cambiare nel tempo di produzione dell'azienda cliente. Ciò nonostante, il sistema ideato da Seica e le applicazioni da me sviluppate dovranno continuare a funzionare, qualunque sia il numero di schede montate su un pannello.

Quando vengono eseguiti i test delle macchine Seica, a essere controllate sono le schede, cui verrà assegnato, una volta terminato il test, uno stato che indicherà se la singola scheda ha passato o no il test o se è da scartare. In base allo stato di ciascuna scheda viene definito lo stato totale del pannello facenti parte, che risulterà correttamente montato e prodotto solo se ciascuna delle schede che lo compone avrà passato con successo tutti i test e potrà quindi successivamente essere messo sul mercato.

2.2 Le macchine Seica

Le macchine utilizzate da Seica per soddisfare le richieste del cliente sono tre macchine di collaudo che eseguono tre tipi di test (ICT, Flash, LED) il cui funzionamento è supportato da 5 PLC di cui uno installato a

monte di tutto il processo, uno installato a valle, e i restanti tre sono affiancati alle tre macchine di collaudo.

2.2.1 Le macchine di collaudo

Le tre macchine di collaudo Seica fanno parte della serie *Pilot Next*, in particolare sono tutte modelli *Pilot V8 Next*. Questo modello ottimizza il test di schede elettroniche "a doppio lato", che richiedono stimoli e misure su entrambi i lati contemporaneamente, dove un approccio "a singolo lato" è estremamente limitante in termini di copertura. Altro grande vantaggio di questi modelli è l'uso di *Flying Probe* anziché del letto ad aghi per testare il contatto tra i vari componenti di una scheda. Ciò permette una copertura del test totale su entrambi i lati contemporaneamente e, di conseguenza, una velocità maggiore nell'eseguire i test.

Le tre macchine eseguono, nell'ordine, i seguenti test uno per macchina:

1. **In-Circuit Test (ICT)**: una sonda elettrica esegue un test su una *PCB (Printed Circuit Board)*, controllando cortocircuiti, circuiti aperti, resistenze, capacità, e altre quantità di base che permetteranno di capire se l'assemblaggio dei componenti sulla scheda è stato eseguito correttamente
2. **Flash Test (Flasher)**: installa il firmware fornito dal cliente sulla scheda corrente preparandola al test successivo
3. **LED Test**: viene controllata la presenza dei LED sulla scheda. La scheda viene accesa e un sensore LED misura luminosità, colore, saturazione e spettro di frequenza dei LED, operando nell'intervallo di luce visibile e infrarossa delle lunghezze d'onda

2.2.2. Il software VIVA

Comune alle tre macchine utilizzate (e in generale per tutte le macchine Pilot Next) è la piattaforma software di Seica VIP, ossia VIVA. Questo software permette un approccio al test completamente versatile ed è dimostrato particolarmente bene in questo progetto, permettendo di eseguire tre tipologie di test differenti su tre macchine uguali (Pilot V8 Next).

L'utilizzo di VIVA da parte dell'utente permette inoltre di ridurre drasticamente il tempo di programmazione e di minimizzare errori e

omissioni, assicurando la qualità e l'affidabilità del programma di test finale.

Alla fine di ogni test, VIVA salva in automatico un resoconto di tutte le misure e i test effettuati sui vari componenti di una scheda, permettendo così di verificarne i valori anche parecchio tempo dopo l'esecuzione del test.

2.2.3 I PLC

Un controllore a logica programmabile (in inglese *programmable logic controller* o PLC) è un computer per l'industria specializzato nella gestione o controllo dei processi industriali. Il PLC esegue un programma ed elabora i segnali digitali ed analogici provenienti da sensori e diretti agli attuatori presenti in un impianto industriale.

La presenza di PLC in questo progetto è dunque di fondamentale importanza, in quanto mette in comunicazione tra loro tutte le macchine coinvolte nel processo di collaudo delle schede elettroniche, a partire dal caricatore delle schede (*loader*), passando per le tre macchine di collaudo Pilot V8 Next, per finire con lo scaricatore delle schede (*unloader*), come si può facilmente intuire dall'immagine 2.1.

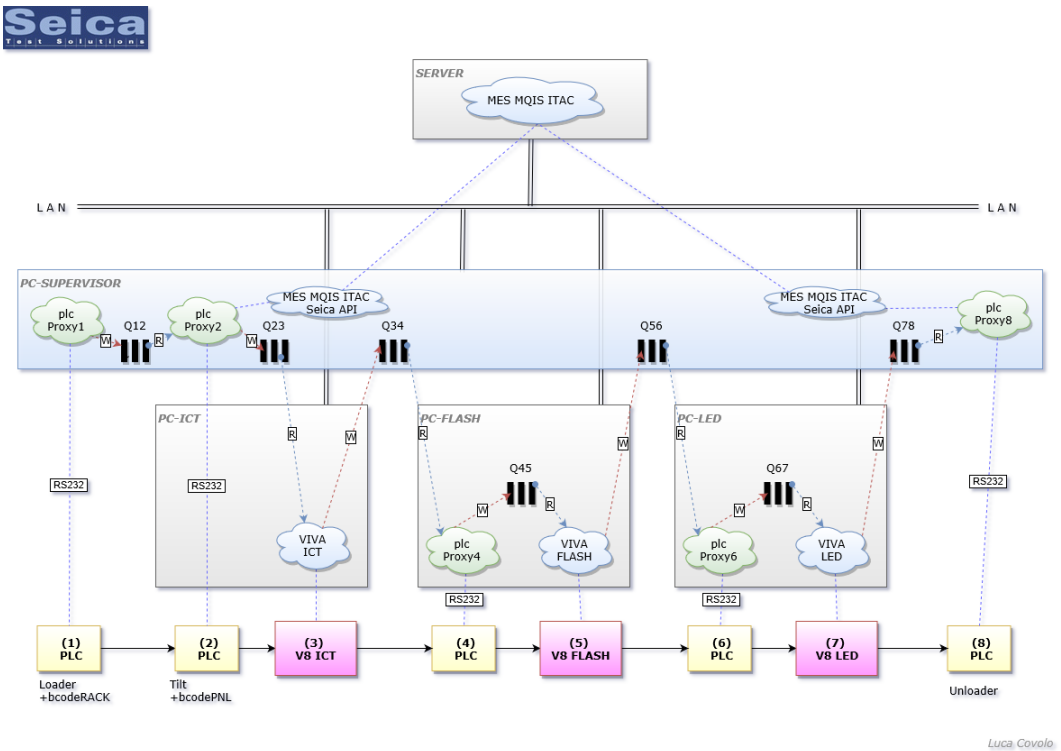

Immagine 2.1 Schema a blocchi del progetto

2.3 La soluzione adottata

La soluzione adottata da Seica per soddisfare le richieste del cliente, "unendo" le tre macchine Pilot V8 Next sotto la guida di un unico computer supervisor, prevede l'uso di 5 PLC, ognuno in comunicazione seriale con la macchina di riferimento.

Internamente, per garantire il passaggio di informazioni tra un'applicazione e l'altra, è stato scelto di utilizzare un sistema di comunicazione basato sulle code di messaggi di Microsoft.

2.3.1 La comunicazione RS-232

Il tipo di comunicazione scelto da Seica tra i PLC e le macchine con cui devono comunicare è quello dello standard EIA R2-232, un'interfaccia seriale a bassa velocità di trasmissione per lo scambio di dati tra dispositivi digitali.

2.3.2 Le code Microsoft

Il sistema di code di messaggi sviluppato da Microsoft (*MSMQ* o *Microsoft Message Queuing*) è un sistema di comunicazione inizialmente distribuito nei suoi sistemi operativi Windows Server, per poi essere successivamente incorporato nelle piattaforme Microsoft. Si tratta essenzialmente di un protocollo di messaggistica che permette ad applicazioni operative su processi separati e distinti di comunicare in maniera sicura e priva di errori.

Proprio per questi motivi è stato scelto di utilizzare MSMQ per registrare le informazioni relative a ciascuna scheda (in particolare il risultato dell'ultimo test cui è stata sottoposta e lo stato generale dell'intero pannello definito da tutti i test eseguiti fino a quel punto su ogni scheda che lo compone) rendendole disponibili per l'elaborazione dell'applicazione successiva.

Capitolo 3

Sviluppo delle applicazioni

Questo capitolo entra nel dettaglio di quello che è stato il mio lavoro all'interno dell'azienda, ossia lo sviluppo di 8 applicazioni software che garantissero l'automazione dell'intero processo di test (a partire dall'inserimento della scheda nella prima macchina di collaudo per mezzo del *loader* fino all'estrazione della stessa dall'isola di macchine per mezzo dell'*unloader*) e lo sviluppo di una libreria C# utilizzata da queste 8 applicazioni per usare il sistema di code di Microsoft, per comunicare con il MES e per elaborare e memorizzare i risultati dei test secondo un formato specifico definito dal cliente.

3.1 M1 – Seica Plc Loader

La prima stazione della linea di produzione è il *Loader*. Il *Loader* è un apparato in cui vengono caricati uno alla volta i 'contenitori' in cui sono stipate le schede (detti *Rack*). Il suo compito è quello di estrarre i pannelli, e caricarli uno dopo l'altro su appositi binari attraverso i quali passeranno nelle stazioni successive.

3.1.1 Il problema dei binari

All'interno della linea di produzione, le schede si spostano da una stazione all'altra scorrendo fra due binari. All'interno di ogni stazione vengono bloccate (senza essere rimosse da essi), analizzate e poi fatte ripartire.

Poiché l'intero sistema nasce per consentire il collaudo automatizzato di schede di diverse dimensioni, questi binari sono regolabili in larghezza, e devono assolutamente essere impostati alla giusta quota prima che una scheda vi transiti. Questo rappresenta la prima delle problematiche legate all'automazione (comune a quasi tutte le stazioni) che abbiamo dovuto affrontare e risolvere in profondità durante lo sviluppo del sistema.

È quindi necessario un sistema per conoscere la larghezza del pannello che sta per essere 'spinto' sui binari, prima che questo avvenga. Poiché all'interno di uno stesso rack sono presenti pannelli aventi tutti la stessa larghezza, il problema è stato risolto (in questa stazione) installando un *lettore di codici a barre* in grado di scandire un codice situato

direttamente sul Rack e riportante la dimensione dei pannelli in esso contenuti.

3.1.2 La ricetta

Affinché anche le stazioni successive possano regolare i rispettivi binari prima dell'arrivo dei pannelli è necessario un metodo di trasmissione delle informazioni relative a ogni pannello attraverso le diverse applicazioni di gestione. A tale scopo si è deciso di utilizzare il sistema MSMQ (*MicroSoft Message Queue*): ogni stazione (eccetto la prima e l'ultima) ha accesso a due code, una in lettura e una in scrittura. Ogni volta che la stazione riceve un pannello in ingresso legge dalla prima coda una stringa in formato XML contenente tutte le informazioni necessarie alla sua elaborazione. Al completamento delle operazioni sul pannello (e prima di 'espellerlo') l'applicazione di gestione scrive sulla seconda coda una versione aggiornata della stringa XML che aveva letto a inizio ciclo. Le due stazioni agli estremi hanno accesso solo a una coda (rispettivamente in scrittura e in lettura).

La stringa XML in questione prende il nome di 'Ricetta' (*Recipe*). Quando viene generata per la prima volta dall'applicazione di gestione del *Loader* ha il seguente formato:

```
<R>
	<Recipe
		ID="{55BF00E2-F915-4EB2-B82F-60D608BCA4B7}"
		M1U="2018-07-21T09:30:10"
		RailWidth="1234"
		BoardsStep="2"
		FirstBoardPosition="1"
	/>
</R>
```

- *Recipe*: è l'elemento principale della ricetta, contiene al suo interno tutti i campi necessari all'elaborazione dei pannelli;
- *ID*: un identificatore univoco generato al momento della creazione della ricetta;
- *M1U*: data e ora di creazione della ricetta;
- *RailWidth*: la larghezza dei binari (ricavata dal codice a barre);

- *BoardsStep*: la distanza tra due pannelli successivi all'interno del rack (ricavata dal codice a barre);
- *FirstBoardPosition*: la posizione della prima scheda all'interno del rack (ricavata dal codice a barre);

3.1.3 L'interfaccia utente

Tutte le applicazioni della linea integrano un'interfaccia utente estremamente semplice, poiché tutte le operazioni sono gestite in completa autonomia dal codice sottostante. Lo scopo principale delle nostre interfacce è quello di avere un riscontro in tempo reale di ciò che sta accadendo nell'applicazione sottostante.

Questa applicazione (come anche le applicazioni siglate M2, M4, M6 e M8) adotta la seguente interfaccia:

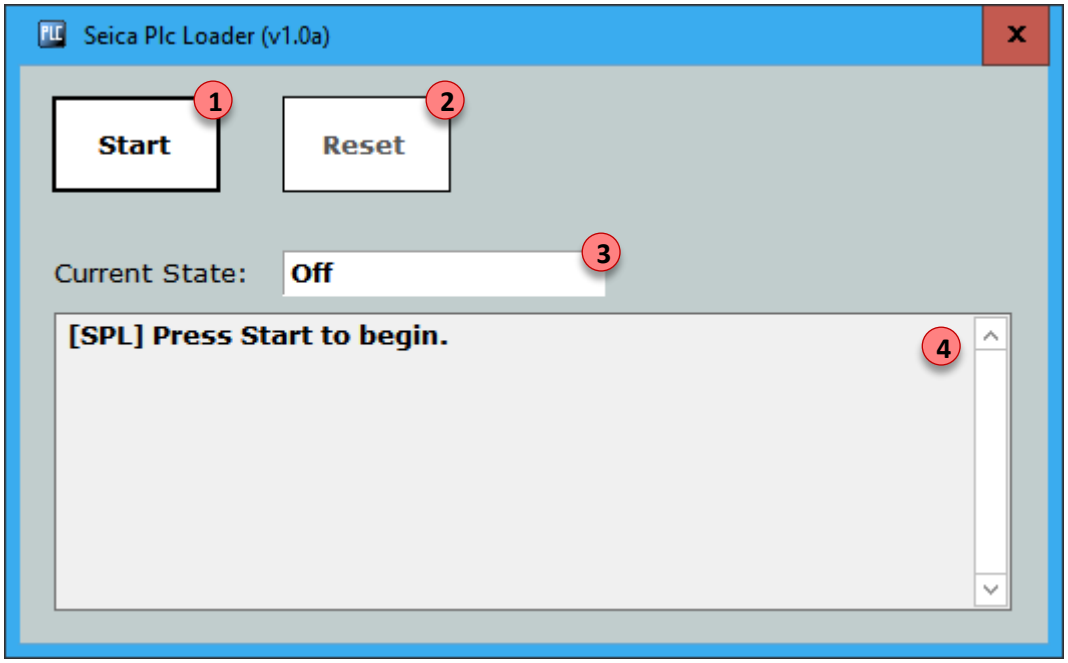

Immagine 3.1 Finestra di avvio 'Seica Plc Loader'

1. *Start*: lancia l'inizializzazione del processo e fa partire il ciclo di produzione;
2. *Reset*: interrompe il processo corrente e cancella le variabili utilizzate;
3. *Barra di stato*: mostra lo stato (vedi paragrafo successivo) del processo;
4. *Finestra di dialogo*: mostra informazioni in tempo reale riguardo alle operazioni che sono eseguite dal processo.

3.1.4 La macchina a stati

Come ognuna delle applicazioni sviluppate per questo progetto, anche *Seica Plc Loader* basa il suo funzionamento su una macchina a stati.

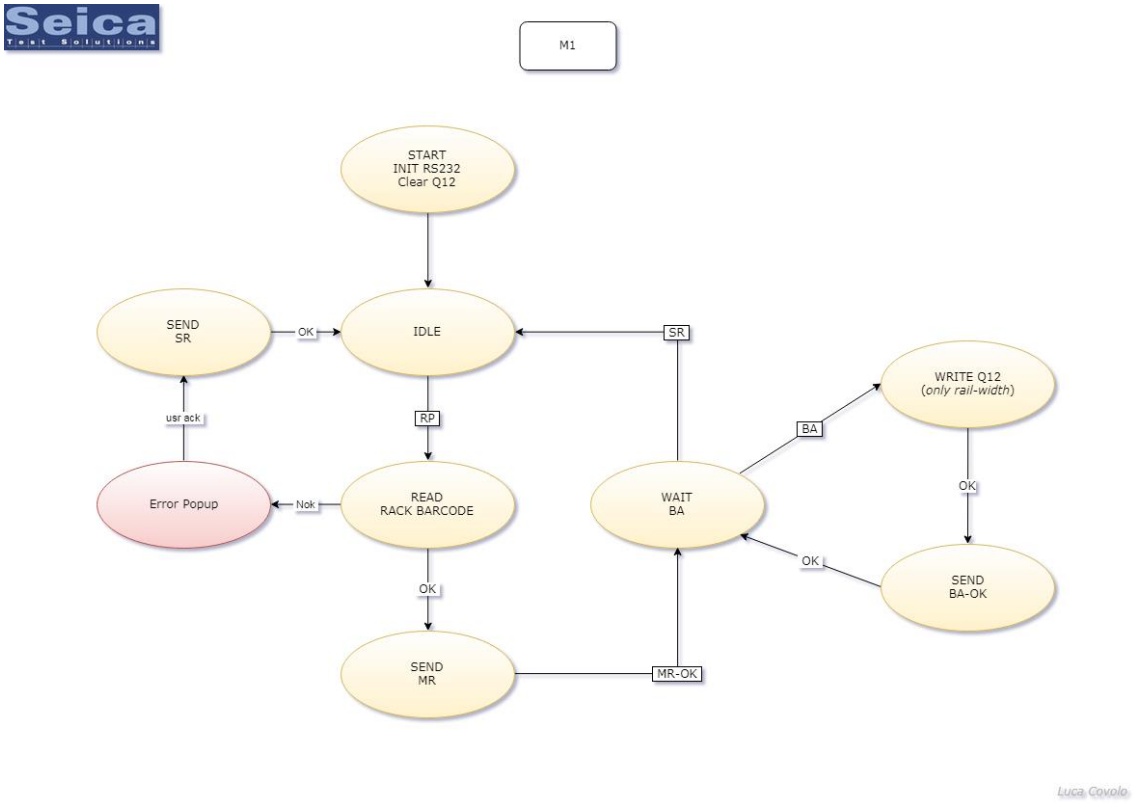

Immagine 3.2 Schema a blocchi M1

Al momento dell'accensione il processo si trova in uno stato 'fittizio' (non riportato negli schemi a blocchi) di '*Off*' a indicare che il processo è pronto ma non è ancora stato attivato.

- *Start*: Alla pressione del pulsante di 'Start' sull'interfaccia l'applicazione entra in questo stato. Qui svolge l'inizializzazione delle due connessioni RS232 tramite seriale (con le quali comunica con il PLC sottostante e con il lettore di codici a barre) e la cancellazione di eventuali messaggi rimasti pendenti sulla coda MSMQ di comunicazione tra questo processo e quello successivo (per chiarezza fare riferimento all'immagine 2.1). Al completamento di queste operazioni si transita allo stato successivo.

- *Idle*: L'applicazione è ora in attesa che venga fisicamente caricato un Rack all'interno del *Loader*. Quando ciò avviene, un sensore avverte il PLC che lo comunica al processo inviando sulla connessione RS232 uno specifico segnale denominato '*RP*' (*Rack Pronto*), il quale fa passare la macchina a stati allo stato successivo.

- *Read Rack Barcode*: Ora che il Rack si trova nella posizione desiderata, è possibile attivare il lettore di codici a barre per ricavare le informazioni sul suo contenuto, necessarie per il proseguimento delle schede all'interno della linea. I comandi di accensione e spegnimento del lettore, come anche il risultato della lettura, vengono tutti trasferiti tramite linea seriale RS232. L'operazione può risultare in una lettura corretta, nel cui caso la macchina a stati procede verso lo stato successivo, o in una fallita lettura, nel cui caso si passa (dopo aver eseguito due ulteriori tentativi di lettura) alla gestione dell'errore.

    - *Error Popup*: nel caso in cui si del tutto impossibile ottenere un codice congruo alle specifiche dalla lettura, viene mostrata una finestra di dialogo all'utente per informarlo dell'errore e invitarlo a procedere allo scarico (manuale o meccanico) del Rack non riconosciuto.

    - *Send SR*: alla conferma dello scarico da parte dell'utente segue l'invio da parte dell'applicazione di un segnale '*SR*' (*Scarico Rack*) al PLC, per informarlo dell'errore in modo che la macchina venga predisposta al rilascio del Rack. Fatto ciò si torna in stato di *Idle*.

- *Send MR*: Quando invece la lettura del codice a barre da un risultato considerato corretto, si procede all'invio al PLC di un segnale '*MR*' (*Machine Ready*) a indicare che l'operazione ha avuto successo e si può passare alla fase successiva di carico delle schede nella linea di produzione. Il PLC risponderà con un segnale di '*MR-OK*' per segnalare la corretta ricezione del messaggio, consentendo al processo di passare allo stato successivo.

- *Wait* BA: Dopo aver inviato il segnale il processo passa in un nuovo stato di attesa: questa volta attende che il *Loader* estragga la prima scheda dal Rack e la collochi in una posizione convenzionale detta 'di uscita', pronta appunto per essere fatta uscire dalla macchina corrente per transitare verso la successiva. Quando ciò avviene il PLC invia un segnale '*BA*' (*Board Available*), alla ricezione del quale la macchina a stati procede verso lo stato successivo.

- *Write Q12*: il passo successivo è quello di creare una 'ricetta' per la scheda in uscita e inserirla nella coda di comunicazione (Q12, poiché mette in comunicazione la macchina in posizione 1 e 2), così che le successive applicazioni conoscano i dati relativi alle schede presenti

in questo Rack, in particolare la larghezza a cui impostare i binari per un corretto transito delle schede tra una stazione e l'altra.

- *Send BA-OK*: dopo aver inviato il pacchetto di informazioni necessario, l'applicazione informa il PLC che è possibile procedere con l'espulsione della scheda verso la macchina successiva, attraverso il segnale di '*BA-OK*'. Su questo punto è richiesta particolare attenzione, in quanto il PLC non è automaticamente autorizzato a spingere la scheda sui binari, poiché essi potrebbero ancora trovarsi ad una larghezza incongruente con quella delle schede del Rack in elaborazione: a questo punto il PLC della prima stazione andrà a comunicare (in modo del tutto trasparente rispetto a *Seica Plc Loader*) con il PLC della stazione seguente tramite segnali elettrici ('fisici'); alla sua richiesta di transito della scheda dovrà seguire una conferma da parte del PLC a valle prima che esso possa effettivamente avvenire. Con questo metodo si evita di far transitare schede di una certa larghezza su binari di larghezza differente. Solo quando la scheda sarà stata fatta uscire e la seguente sarà pronta in uscita il PLC invierà un nuovo segnale '*BA*' al processo.

- *Wait BA (pt.2)*: a questo punto il processo torna in stato di attesa (di avere una nuova scheda pronta in uscita). Questo stato ha un'altra possibile uscita, in seguito alla ricezione di un segnale '*SR*' (*Scarico Rack*) da parte del PLC. Questo avviene quando il Rack si esaurisce e il PLC ne richiede il carico (manuale o automatico) di uno nuovo. A questo punto il primo ciclo di *Seica Plc Loader* può considerarsi concluso.

3.2 M2 – Seica Plc Barcode Reader

La seconda stazione della linea di produzione è il *Barcode Reader*. Il *Barcode Reader* è un convogliatore che porta le schede dalla stazione di carico (la prima) alla prima vera e propria stazione di test (la terza). Inoltre, al suo interno è presente un lettore di codici a barre (da qui il nome dell'applicazione) che ricava, da un codice riportato sul bordo di ogni pannello, importanti informazioni sulle analisi che dovranno essere svolte su di esso.

3.2.1 L'interfaccia utente

L'interfaccia utente di *Seica Plc Barcode Reader* è del tutto identica quella presentata nel capitolo precedente per *Seica Plc Loader* (vedi immagine 3.1).

3.2.2 La macchina a stati

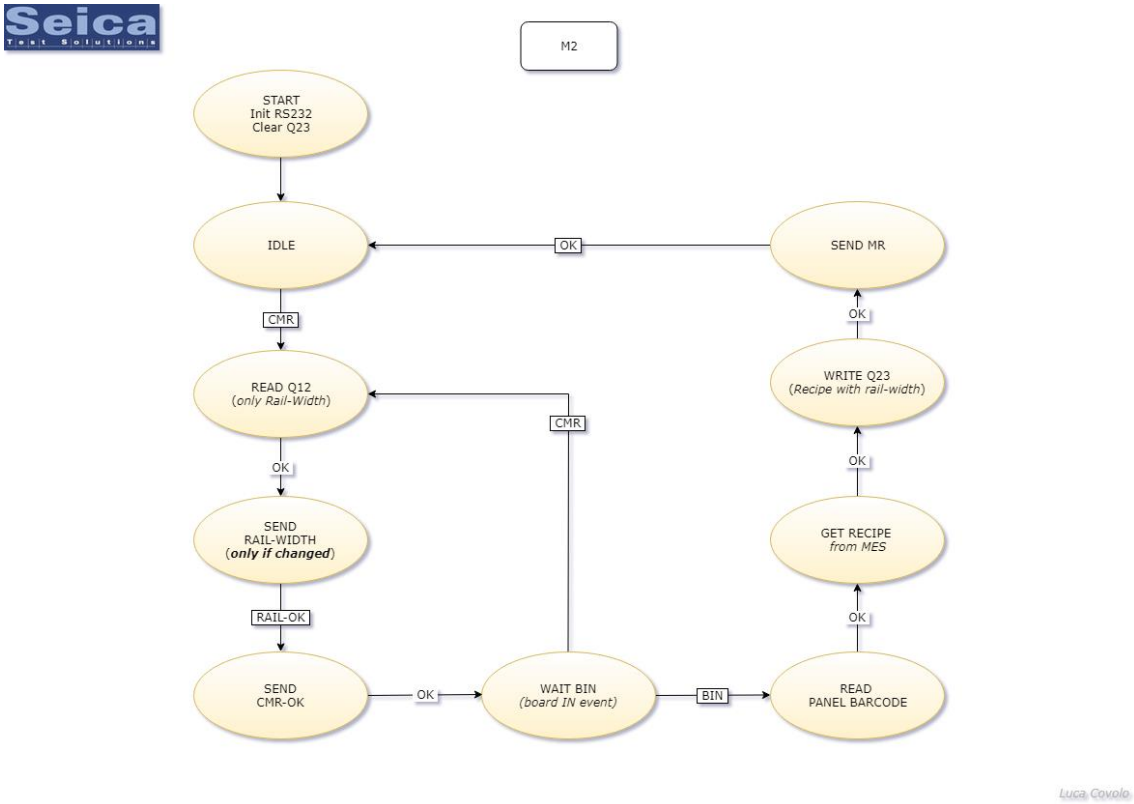

Immagine 3.3 Schema a blocchi M2

Al momento dell'accensione il processo si trova in uno stato 'fittizio' (non riportato negli schemi a blocchi) di '*Off*' a indicare che il processo è pronto ma non è ancora stato attivato.

- *Start*: Alla pressione del pulsante di 'Start' sull'interfaccia l'applicazione entra in questo stato. Qui svolge l'inizializzazione delle due connessioni RS232 tramite seriale (con le quali comunica con il PLC sottostante e con il lettore di codici a barre), la cancellazione di eventuali messaggi rimasti pendenti sulla coda MSMQ di comunicazione tra questo processo e quello successivo (per chiarezza fare riferimento all'immagine 2.1) e l'inizializzazione di una connessione TCP con il '*MES Server*'. Al completamento di queste operazioni si transita allo stato successivo.

- *Idle*: l'applicazione è in attesa di un segnale '*CMR*' (*Consenso al Machine Ready*) dal PLC. Il PLC manda questo segnale verso l'applicazione quando riceve dalla stazione precedente la richiesta di far transitare una scheda sui binari che collegano le due postazioni (l'applicazione deve assicurare la corretta larghezza dei binari e dare appunto un 'consenso' prima che la scheda possa effettivamente transitare).

- *Read Q12*: a questo punto il processo è consapevole del fatto che la stazione precedente è pronta all'invio di una scheda, quindi può andare a leggere dalla coda Q12 il pacchetto (ricetta) di informazioni relativo al pannello pronto in uscita.

- *Send Rail-Width*: una volta acquisiti i dati dalla ricetta, l'applicazione invia al PLC le informazioni riguardanti la larghezza a cui devono essere impostati i binari (solo nel caso in cui essa sia differente da quella già impostata al ciclo precedente) e si mette in attesa. Sarà il PLC che si occuperà dello spostamento fisico dei binari e, una volta terminato, lo comunicherà al processo con un segnale '*Rail-OK*'.

- *Send CMR-OK*: completato il riposizionamento dei binari l'applicazione può dare il 'consenso' al PLC per fare transitare la scheda in uscita dalla prima stazione.

- *Wait BIN*: a questo punto l'applicazione torna in attesa, questa volta perché deve aspettare che la scheda completi il trasferimento fisico da una stazione all'altra e arrivi nella posizione predefinita. Il PLC si accorge dell'arrivo del pannello grazie a un sensore e lo segnala inviando un messaggio di '*BIN*' (*Board IN*). Come si vede dall'immagine 3.3 è presente anche una seconda 'via di uscita' da questo stato: questo perché per varie problematiche legate al funzionamento fisico delle stazioni, il *Loader* potrebbe segnalare una scheda in uscita quando invece essa non è fisicamente presente. In questo caso il processo gestisce il ciclo normalmente, ma arrivati in questo stato ci si aspetterebbe di vedere arrivare la scheda in ingresso, cosa che invece non accade in quanto essa non era fisicamente presente. Succede quindi che il *Loader* passerà al ciclo successivo e invierà una nuova richiesta di transito, che verrà ricevuto mentre l'applicazione si trova in questo stato di attesa. Per gestire questa evenienza, in corrispondenza di tale segnale (nell'immagine 3.3 denominato '*CMR*') il ciclo torna semplicemente

indietro allo stato di *Read Q12* saltando (correttamente) l'elaborazione della scheda non presente.

- *Read Panel Barcode*: quando la scheda si trova ferma in posizione di arrivo all'interno della stazione è possibile leggerne il codice a barre. La lettura avviene come per il *Loader* tramite una comunicazione su linea seriale RS232.

- *Get Recipe*: questa volta il codice a barre non contiene informazioni facilmente decodificabili, ma un numero identificativo univoco per ogni pannello secondo un formato predisposto dal cliente. È quindi necessario comunicare con il *MES Server* per ottenere informazioni utili da questo codice. In questo stato l'applicazione invia tramite connessione TCP il codice a barre appena letto e riceve, in caso di trasmissione con esito positivo, una serie di informazioni riguardanti il contenuto del pannello e il tipo di analisi che dovrà svolgere.

- *Write Q23*: una volta ottenute le informazioni di cui la linea di produzione necessita, viene aggiornata la ricetta e scritta sulla coda successiva (Q23, tramite la quale comunicano la seconda e la terza stazione). La nuova ricetta avrà un formato del tipo:

```
<R>
          <Recipe
               ID="{55BF00E2-F915-4EB2-B82F-60D608BCA4B7}"
               M1U="2018-07-21T09:30:10"
               RailWidth="1234"
               BoardsStep="2"
               FirstBoardPosition="1"

               M2U="2018-07-21T09:31:24"
               ProgramName="<... 11 digit ...>"
               Variant=""
               Barcode="<... 67 digit ...>"
               SkipAllTests="0"
               SkipAllTestsMessage="<... ...>"
               <boards>
                     <brd
                          barcode="<... 31 digit ...>"
                          pos="1"
                          lc=""
                          status="0"
                     />
               </boards>
     />
</R>
```

La prima parte della ricetta rimane identica a come è stata ricevuta dalla stazione precedente, mentre i campi aggiunti hanno il seguente significato:

- *M2U*: data e ora in cui avviene l'elaborazione in questa stazione;
- *ProgramName*: il nome del programma di prova che dovrà essere eseguito nelle stazioni successive sul pannello corrente.

- *Variant*: campo aggiuntivo per utilizzi futuri (attualmente vuoto);
- *Barcode*: il codice a barre di pannello;
- *SkipAllTests*: un numero intero (tra 0 e 1) che indica se il pannello in questione debba saltare tutte le prove (1) oppure, come avviene di norma, eseguirle (0);
- *SkipAllTestsMessage*: una stringa di lunghezza qualsiasi che fornisce una motivazione nel caso in cui il pannello debba saltare tutte le analisi. In caso normale funzionamento rimane vuota;
- *Boards*: questa sezione contiene un elemento per ognuna della sotto-schede in cui è diviso il pannello in analisi (faremo riferimento d'ora in poi a queste come 'schede' e al complesso come 'pannello'). Ogni elemento *brd* rappresenta una scheda e contiene quattro attributi:
    - *Barcode*: il codice della singola scheda (differisce in lunghezza e contenuto da quello del pannello);
    - *Pos*: un intero che indica la posizione della scheda all'interno del pannello (da uno al numero di schede);
    - *Lc*: campo aggiuntivo per utilizzi futuri (attualmente vuoto);
    - *Status*: un intero (tra 0 e 2) che indica lo stato della scheda, indispensabile per le stazioni di prova successive;

Tutte le informazioni aggiunte d questa stazione (ad eccezione del codice a barre del pannello) vengono ricevute dal '*MES Server*'.

- *Send MR*: una volta inviata la nuova ricetta sulla coda in uscita l'applicazione comunica al PLC che è possibile iniziare la sequenza di trasferimento del pannello verso la stazione successiva, inviando un segnale di '*MR*' (*Machine Ready*). Dopo di che torna in attesa di una richiesta di trasferimento dalla stazione precedente (*Idle*).

3.3 M3 – Seica V8 ICT

La terza applicazione della linea è *Seica V8 ICT*. Questa applicazione supporta e gestisce la prima vera e propria macchina di prova che si incontra lungo la linea. Prende infatti il nome dal modello della macchina (V8) e dal tipo di analisi in essa effettuata (ICT, *In Circuit Test*). Come già spiegato in introduzione, ricordiamo che la macchina in questione integra un elaboratore sul quale è in esecuzione un processo fondamentale al funzionamento della stessa (ovvero '*VIVA*', il software su cui si basano tutte le macchine Seica) e con il quale la nostra applicazione andrà ad interloquire.

3.3.1 L'interfaccia utente

L'interfaccia utente di *Seica V8 ICT* è molto semplice: in apertura viene mostrata una prima schermata, riportante alcune informazioni fondamentali perché il processo possa avviarsi correttamente, seguite dal pulsante di 'Start'. Una volta avviato il processo l'interfaccia mostrerà una barra di stato, una finestra di dialogo in cui vengono riportate informazioni riguardanti le operazioni in svolgimento, e un pulsante di 'Reset'.

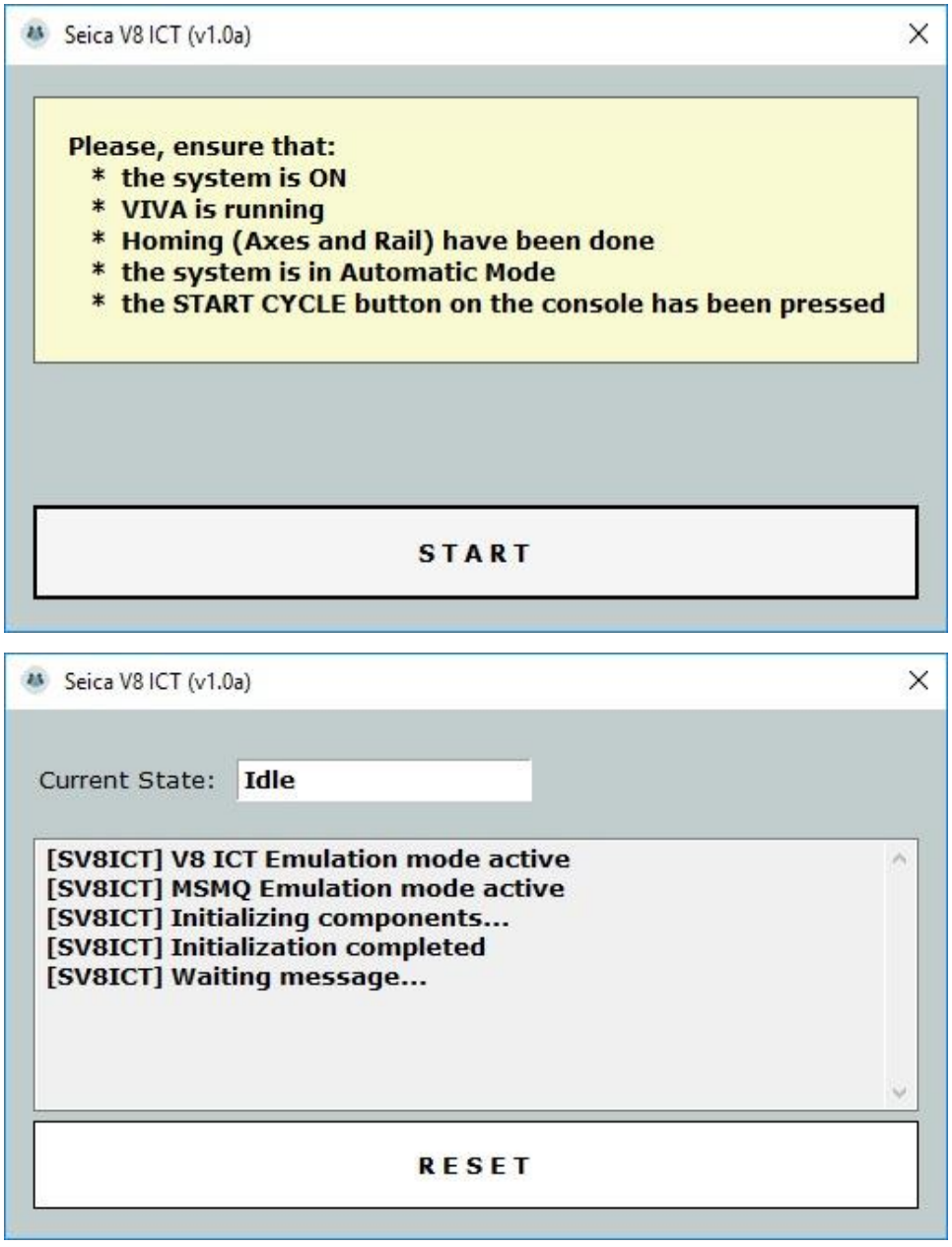

Immagine 3.4 Seica V8 ICT appena aperta (sopra), e durante il ciclo (sotto).

3.3.2 La macchina a stati

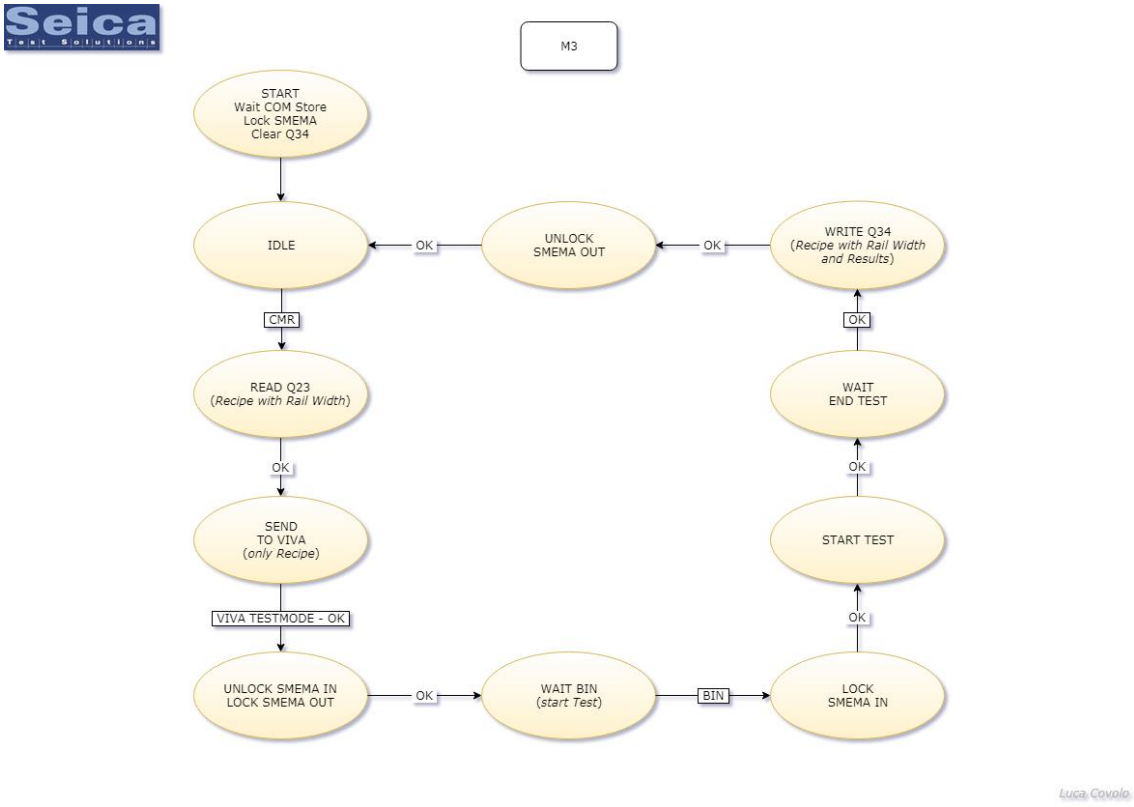

Immagine 3.5 Schema a blocchi M3

Al momento dell'accensione il processo si trova in uno stato 'fittizio' (non riportato negli schemi a blocchi) di '*Off*' a indicare che il processo è pronto ma non è ancora stato attivato.

- *Start*: alla pressione del pulsante di 'Start' sull'interfaccia l'applicazione entra in questo stato, nel quale avviene l'inizializzazione dei vari componenti necessari al processo. In questa fase viene inizializzata e svuotata la coda in uscita (Q34), viene eseguito il controllo dell'esecuzione dei programmi fondamentali al funzionamento della macchina ('VIVA' e un *COM-Server* in grado di interloquire con gli apparati meccanici della stazione) e in seguito vengono bloccati i binari in ingresso e uscita dalla macchina, poiché nessuna scheda può uscire e/o entrare senza autorizzazione da *Seica V8 ICT*. Terminate queste operazioni, viene cambiata la schermata di avvio con quella di esecuzione e si entra in ciclo;

- *Idle*: l'applicazione rimane in stato di attesa fino a quando un segnale elettrico dalla stazione precedente non comunica alla macchina che un pannello è pronto per essere analizzato. In seguito a tale evento

la macchina invia un segnale 'CMR' (*Consenso al Machine Ready*) all'applicazione tramite *COM-Store* e si mette in attesa di conferma. Ricevuto il segnale 'CMR' l'applicazione viene messa al corrente della presenza di un pannello in attesa di elaborazione e può quindi passare allo stato successivo;

- *Read Q23*: a questo punto è possibile andare a prelevare il pacchetto contenente la ricetta del pannello in ingresso dalla coda MSMQ di comunicazione con la stazione precedente.

- *Send to VIVA*: una volta estratto il pacchetto, esso viene elaborato: in particolare in questa stazione è necessario eseguire l'analisi di tutte le schede che hanno come valore di stato '0' (scheda buona) o '1' (scheda da rielaborare), mentre quelle con stato '2' (errore) non devono essere analizzate. Il risultato dell'elaborazione viene trascritto, con un formato specifico, in una stringa che viene inviata al software 'VIVA', in modo che possa selezionare il corretto programma di analisi e importare i dati delle schede in arrivo (nonché dimensionare adeguatamente la larghezza dei binari interni alla stazione);

- *Unlock IN - Lock OUT*: una volta ottenuta la conferma di ricezione del pacchetto precedentemente inviato a 'VIVA', è possibile, attraverso l'utilizzo del *COM-Store*, sbloccare i binari in ingresso della stazione e bloccare (al primo ciclo saranno già bloccati, ma dal secondo potrebbero non esserlo più) i binari in uscita. Fatto ciò inizia il trasferimento fisico del pannello tra le due stazioni;

- *Wait BIN*: l'applicazione entra quindi in attesa che il pannello arrivi nella posizione predisposta all'interno della macchina. Una volta in posizione un sensore invia un segnale 'BIN' (*Board IN*) al processo, che può passare allo stato successivo;

- *Lock IN*: in modo speculare a quanto eseguito nello stato di *Unlock IN*, vengono ora bloccati i binari in ingresso, per prevenire l'entrata di più pannelli nella macchina contemporaneamente;

- *Start Test*: Successivamente viene inviato un comando a 'VIVA' che avvia il processo di analisi vera e propria;

- *Wait End Test*: L'applicazione entra quindi in un altro stato di attesa, in cui aspetta che 'VIVA' le comunichi la fine dell'analisi;

- *Write Q34*: una volta ricevuto il segnale di fine analisi da 'VIVA', l'applicazione aggiorna gli stati delle singole schede (presenti nella ricetta) in base al risultato delle analisi che viene comunicato da 'VIVA'. Viene inoltre eseguita una copia dei file contenenti i risultati dei vari test nel dettaglio (detti *Datalog*) in una cartella predisposta; tali risultati saranno poi recuperati ed elaborati dall'ultima stazione, la quale comunicherà poi gli esiti complessivi al MES. Fatto ciò viene aggiunta una stringa con data e ora dell'elaborazione alla ricetta (simile a quelle che aggiungevano le stazioni precedenti) e si procede alla scrittura di quest'ultima sulla coda in uscita (Q34);

- *Unlock OUT*: a questo punto rimane solo da sbloccare i binari in uscita dalla stazione, in modo che, una volta ricevuto il consenso dalla stazione successiva, il pannello possa transitare e continuare il suo percorso. Fatto ciò il ciclo giunge a termine e l'applicazione ritorna in stato di attesa (*Idle*).

3.4 M4 – Seica Flash Loader

La quarta applicazione della linea è *Seica Flash Loader*. Questa è una delle applicazioni più semplici presenti nella linea, in quanto il suo unico compito è quello di far transitare pannelli e ricette ad essi annesse dalla macchina di analisi precedente (*V8 ICT*) e quella successiva, alla quale deve anche il nome (*V8 FLASH*).

3.4.1 L'interfaccia utente

L'interfaccia utente di *Seica Flash Loader* è del tutto identica quella presentata nel primo capitolo per *Seica Plc Loader* (vedi immagine 3.1).

3.4.2 La macchina a stati

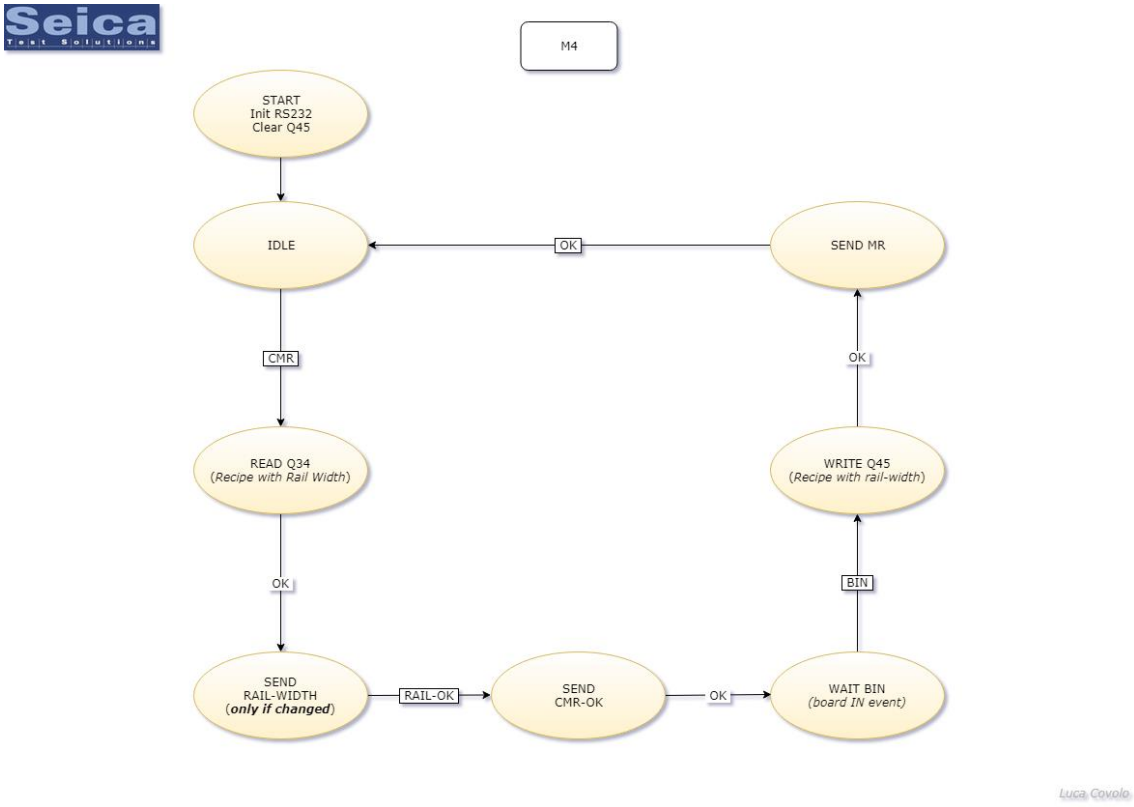

Immagine 3.6 Schema a blocchi M4

Al momento dell'accensione il processo si trova in uno stato 'fittizio' (non riportato negli schemi a blocchi) di '*Off*' a indicare che il processo è pronto ma non è ancora stato attivato.

- *Start*: alla pressione del pulsante di 'Start' sull'interfaccia l'applicazione esegue l'inizializzazione della connessione tramite seriale RS232 con il PLC e la pulizia della coda in uscita (Q45). Una volta terminate queste operazioni entra nello stato successivo;

- *Idle*: durante questo stato l'applicazione si trova in attesa di una richiesta di ingresso da parte di un pannello in uscita dalla stazione precedente. Questa viene segnalata elettricamente al PLC, il quale la propaga verso il processo tramite il segnale 'CMR' (*Consenso al Machine Ready*). Ricevuto questo segnale si entra nello stato successivo;

- *Read Q34*: a questo punto è possibile leggere dalla coda in ingresso (in comunicazione con la stazione precedente) la ricetta relativa al pannello che si sta per ricevere;

- *Send Rail Width*: una volta ottenuta dalla ricetta la larghezza del pannello in ingresso, questa viene comunicata al PLC (tramite uno specifico colloquio), e l'applicazione attende che i binari vengano impostati da esso alla corretta quota. Quando ciò avviene il PLC notifica il processo con un apposito segnale;

- *Send CMR-OK*: ottenuta la conferma di avvenuta regolazione dei binari, è possibile inviare un segnale elettrico alla stazione precedente per informarla della possibilità di trasferire il pannello. Ciò viene fatto dal PLC in seguito alla ricezione del segnale (inviato dall'applicazione) di 'CMR-OK';

- *Wait BIN*: a questo punto il processo entra in uno stato di attesa, in cui aspetta che il pannello arrivi nella posizione predefinita all'interno della stazione. Una volta in posizione, un sensore invierà un 'BIN' segnale all'applicazione, che procederà con le operazioni successive;

- *Write Q45*: una volta che il pannello giunge nella stazione, l'applicazione procede alla scrittura della ricetta sulla coda in uscita, dopo aver aggiunto ad essa un campo riportante data e ora di elaborazione in questa stazione;

- *Send MR*: completato il trasferimento della ricetta, il pannello è già pronto per uscire dalla stazione (operazione che sarà ovviamente subordinata alla ricezione di un consenso da parte della stazione successiva) perciò l'applicazione lo segnala al PLC con un messaggio 'MR' (*Machine Ready*), in seguito al quale quest'ultimo potrà comunicare elettricamente alla stazione successiva la presenza di un pannello pronto per essere trasferito. In seguito a questa operazione il ciclo si conclude, e l'applicazione torna in stato di *Idle*.

3.5 M5 – Seica V8 Flash

La quinta applicazione della linea è *Seica V8 Flash*. Questa è la seconda stazione dove vengono effettivamente eseguite delle analisi sulle schede. Come per *Seica V8 ICT* questa applicazione non colloquierà con un PLC ma bensì direttamente con il software grazie al quale lavorano le macchine di test Seica, ovvero 'VIVA'. Anche questa applicazione prende il nome dal tipo di analisi a cui sono sottoposte le schede al suo interno, che prende il nome di '*Flash Test*'.

3.5.1 L'interfaccia utente

L'interfaccia utente di *Seica Flash Loader* è del tutto identica quella presentata nel terzo capitolo per *Seica V8 ICT* (vedi immagine 3.4).

3.5.2 La macchina a stati

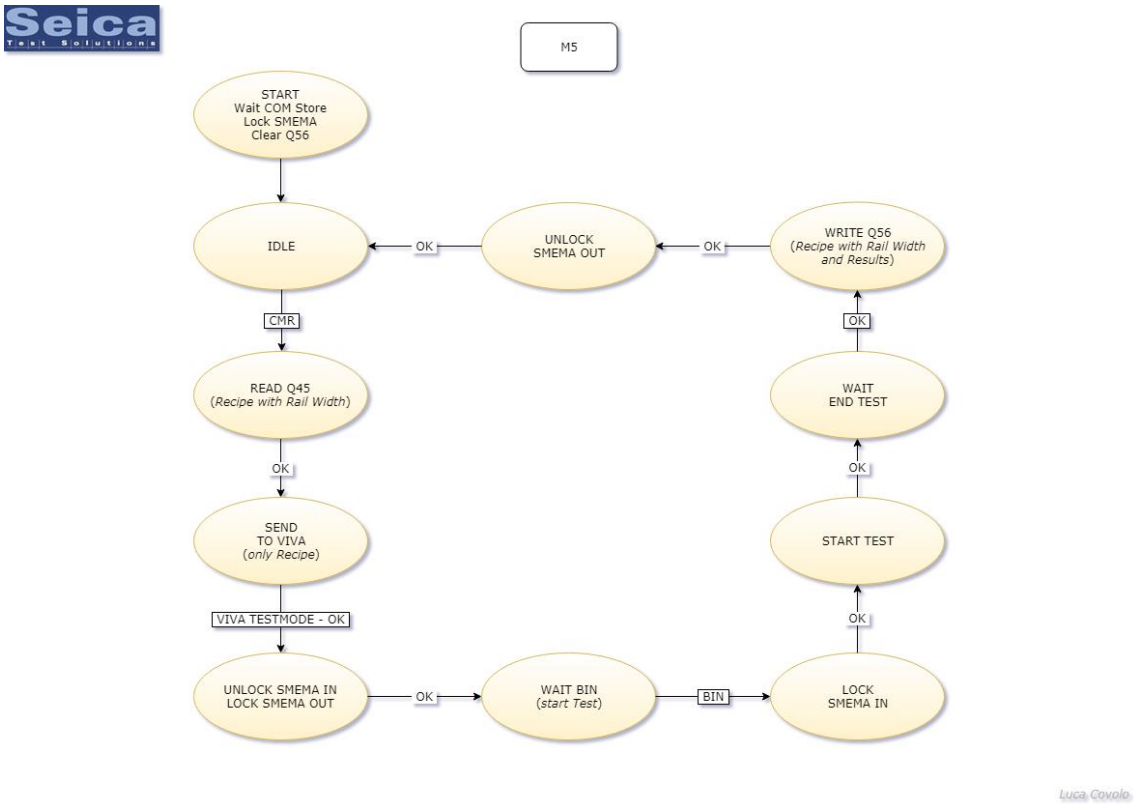

Immagine 3.7 Schema a blocchi M5

Come è possibile dedurre dallo schema 3.7, il ciclo di produzione di questa stazione è del tutto identico a quello della stazione *Seica V8 ICT* (paragrafo 3.3.2), ragion per cui non ne verrà documentata in dettaglio la macchina a stati.

3.6 M6 – Seica LED Loader

La sesta applicazione della linea è *Seica LED Loader*. Questa stazione, decisamente semplice, si occupa del trasferimento dei pannelli e delle ricette annesse dalla macchina di test precedente (*Seica V8 Flash*) alla terza ed ultima stazione di analisi (*Seica V8 LED*) dalla quale deriva il suo nome.

3.6.1 L'interfaccia utente

L'interfaccia utente di *Seica LED Loader* è del tutto identica quella presentata nel primo capitolo per *Seica Plc Loader* (vedi immagine 3.1).

3.6.2 La macchina a stati

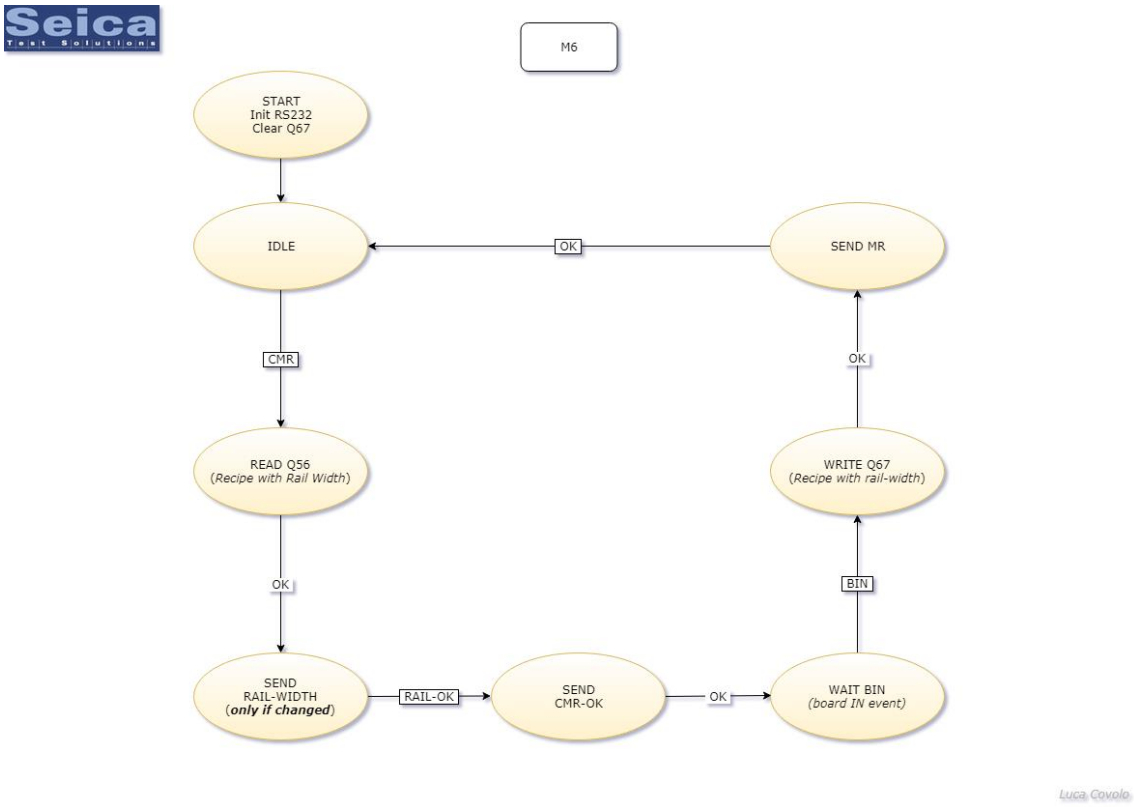

Immagine 3.8 Schema a blocchi M6

Come è possibile dedurre dallo schema 3.8, il ciclo di produzione di questa stazione è del tutto identico a quello della stazione *Seica Flash Loader* (paragrafo 3.4.2), ragion per cui non ne verrà documentata in dettaglio la macchina a stati.

3.7 M7 – Seica V8 LED

La settima applicazione della linea è *Seica V8 LED*. Questa è la terza e ultima stazione dove vengono effettivamente eseguite delle analisi sulle schede. Come per le precedenti, anche questa applicazione non colloquierà con un PLC ma bensì direttamente con il software grazie al quale lavorano le macchine di test Seica, ovvero 'VIVA'. Anche questa applicazione prende il nome dal tipo di analisi a cui sono sottoposte le schede al suo interno, che prende il nome di '*Test LED*', poiché si tratta

di una analisi della luminosità dei canali RGB dei vari componenti a LED presenti sui pannelli.

3.7.1 L'interfaccia utente

L'interfaccia utente di *Seica LED Loader* è del tutto identica quella presentata nel terzo capitolo per *Seica V8 ICT* (vedi immagine 3.4).

3.7.2 La macchina a stati

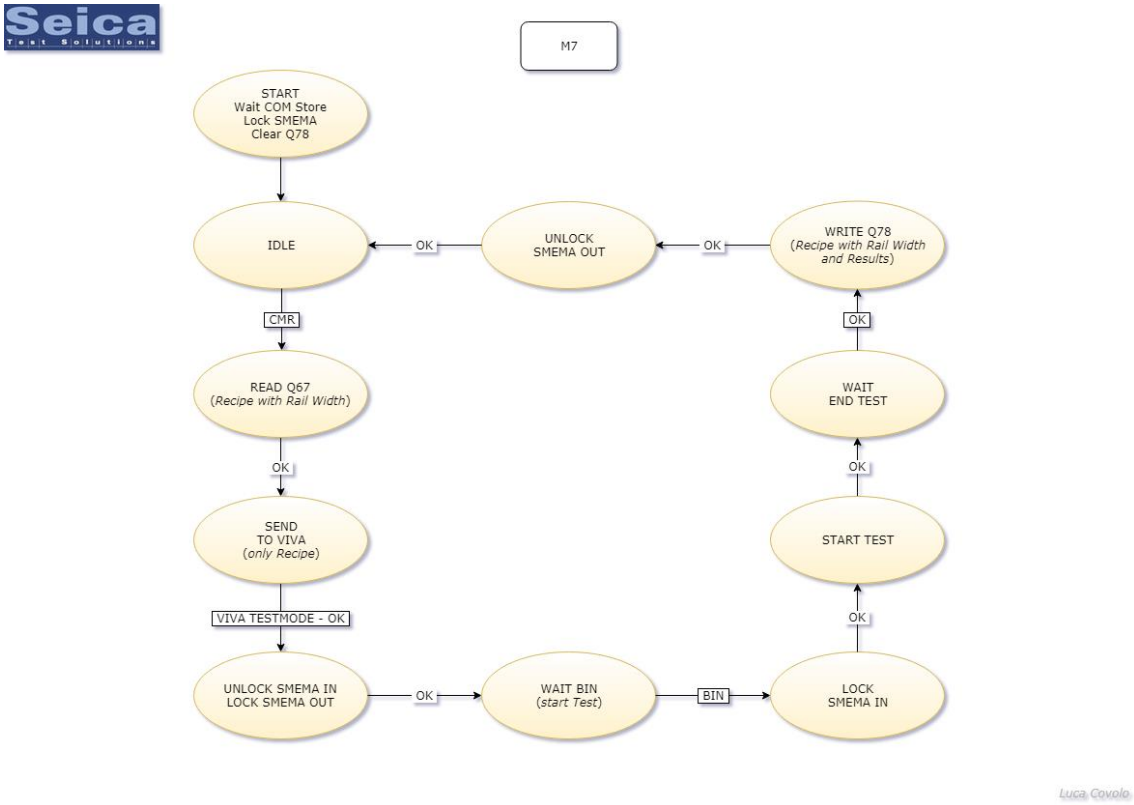

Immagine 3.9 Schema a blocchi M7

Come è possibile dedurre dallo schema 3.9, il ciclo di produzione di questa stazione è del tutto identico a quello della stazione *Seica V8 ICT* (paragrafo 3.3.2), ragion per cui non ne verrà documentata in dettaglio la macchina a stati.

3.8 M8 – Seica Plc Unloader

L'ottava e ultima applicazione della linea è *Seica Plc Unloader*. Probabilmente una delle stazioni più complesse, questa deve reperire,

per ogni pannello in transito, tutti i risultati delle analisi su di esso, elaborarli, comunicarli al MES, e infine decidere in quale dei due Rack di scarico depositare il pannello.

3.8.1 L'interfaccia utente

L'interfaccia utente di *Seica Plc Unloader* è del tutto identica quella presentata nel primo capitolo per *Seica Plc Loader* (vedi immagine 3.1).

3.8.2 La macchina a stati

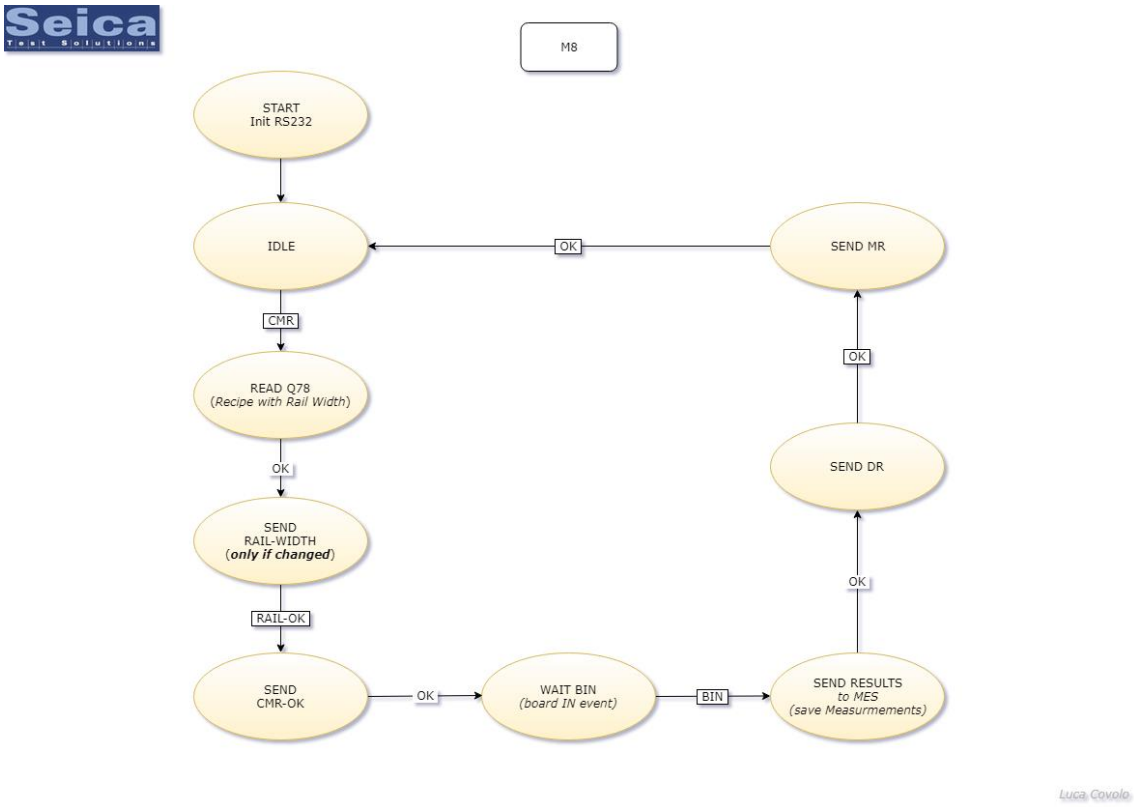

Immagine 3.10 Schema a blocchi M8

Al momento dell'accensione il processo si trova in uno stato 'fittizio' (non riportato negli schemi a blocchi) di '*Off*' a indicare che il processo è pronto ma non è ancora stato attivato.

- *Start*: alla pressione del pulsante di 'Start' sull'interfaccia l'applicazione esegue l'inizializzazione dei suoi componenti: in questo caso si tratta solamente della connessione tramite seriale RS232 con il PLC, dal momento che non è necessaria alcuna coda di informazioni in uscita in quanto si trova al termine della linea;

- *Idle*: terminata l'inizializzazione si entra in stato di attesa di una richiesta di ingresso da parte di un pannello in uscita dalla stazione precedente. Quando il PLC riceve la richiesta tramite segnale elettrico, la rimbalza all'applicazione tramite il segnale logico 'CMR' (*Consenso al Machine Ready*);

- *Read Q78*: a questo punto è possibile andare a leggere dalla coda MSMQ in ingresso il pacchetto di informazioni, ormai in forma finale, relativo al pannello in attesa di ingresso;

- *Send Rail Width*: una volta letta dalla ricetta la larghezza del pannello, questa viene inviata, con un apposito colloquio, al PLC, il quale si occupa dell'allargamento fisico dei binari. L'applicazione attende fino al compimento di questa operazione, segnalato dal PLC con un segnale 'RAIL-OK';

- *Send CMR-OK*: a tal punto il processo segnala al PLC che è possibile eseguire il trasferimento. Il PLC rimbalza il segnale (tramite segnale elettrico) alla stazione precedente e inizia lo spostamento fisico del pannello;

- *Wait BIN*: l'applicazione entra in attesa dell'arrivo del pannello nella posizione prestabilita all'interno della stazione. Una volta arrivato, un sensore avverte il PLC che invia un segnale 'BIN' (*Board IN*) al processo, facendolo avanzare allo stato successivo;

- *Send to MES*: a questo punto si entra nello stato dove avvengono la maggior parte delle operazioni. Innanzitutto, l'applicazione deve comunicare al MES i risultati cumulativi delle analisi sui pannelli e sulle singole schede: per farlo ha a disposizione la ricetta, sulla quale – macchina dopo macchina – viene aggiornato lo stato delle schede componenti i pannelli; queste informazioni vengono rielaborate in una stringa e inviate al MES. Viene poi eseguita un'altra operazione: ogni stazione di analisi, salva i risultati (dettagliati) dei test su un determinato pannello in un file detto *Datalog* reperibile in una cartella denominata come il codice a barre del pannello in questione; è compito dell'ultima applicazione andare a reperire i tre *Datalog* (uno per stazione di analisi) relativi al pannello in esame, leggerli e convertirli in un formato richiesto su misura dal cliente. Questa operazione è fondamentale al fine di rendere completamente tracciabile qualsiasi errore nelle analisi o nel funzionamento delle schede. Una volta eseguite queste operazioni l'applicazione sarà in grado di decidere se il pannello in questione dovrà essere rielaborato

(eseguire un nuovo ciclo all'interno della linea) oppure potrà essere scaricato definitivamente;

- *Send DR*: a questo punto è necessario riferire al PLC di quale dovrà essere il Rack di destinazione del pannello in esame. I Rack sono due: uno di 'rielaborazione' in cui vengono inseriti tutti i pannelli che hanno fallito alcune analisi (che ripeteranno l'intero ciclo) e uno di 'uscita' in cui vengono scaricati sia i pannelli che hanno passato tutte le prove che quelli che hanno riscontrato errori fatali, tali da non consentirne la rielaborazione. Per comunicare al PLC il Rack di destinazione, il processo invia un segnale 'DR' (*Destination Rack*) seguito da un codice che indica su quale dei due debbano essere indirizzati i binari;

- *Send MR*: una volta che il PLC ha spostato i binari verso il Rack corretto, è possibile comunicare a esso di eseguire lo scarico vero e proprio del pannello. Questo viene fatto tramite l'invio di un segnale 'MR' (*Machine Ready*). Una volta eseguito lo scarico, il ciclo è completo e l'applicazione torna in stato di '*Idle*'.

3.9 Seica API

Tutte le applicazioni illustrate finora utilizzano un *API (Application Programming Interface)* che ho realizzato e via via migliorato, per creare dei metodi che riguardassero la comunicazione con il MES tramite la suite iTAC, la lettura e la scrittura sulle code di messaggi di Microsoft e infine la lettura, l'elaborazione e la memorizzazione dei risultati dei singoli test di ogni componente di ogni scheda secondo il formato richiesto dal cliente, il tutto seguendo un ***facade pattern***, che definisce un'interfaccia di livello più alto rendendo il sottosistema più facile da usare.

3.9.1 Il singleton pattern

Tutti i metodi sfruttano uno dei pattern fondamentali descritti dalla *"Gang of Four"* (Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides) nel celebre libro *Design Patterns*, ovvero sia il ***singleton pattern*** (con inizializzazione *"lazy"*):

```
static readonly Lazy<Seica> lazy = new Lazy<Seica>(() => new Seica());
public static Seica Singleton { get { return lazy.Value; } }
private Seica() { }
```

In questo modo è possibile garantire che di una determinata classe (nell'esempio: `Seica`) venga creata **una e una sola** istanza, fornendo un punto di accesso globale a tale istanza. L'inizializzazione *"lazy"* fa sì che la creazione dell'istanza della classe venga rimandata nel tempo e messa in atto solo quando ciò diventa strettamente necessario (cioè al primo tentativo di uso).

3.9.2 Precondizioni ed eccezioni

Ogni metodo pubblico facente parte dell'API ha le seguenti caratteristiche:

- È di tipo `bool`, così da permettere facilmente al programma chiamante di sapere se l'operazione ha avuto successo o meno, ossia se il metodo ha fatto ritornare valore `true` o `false`
- Il corpo del metodo è racchiuso in un blocco `try and catch` in modo da riuscire a "catturare" qualunque tipo di "exception" eventualmente "lanciata" dal metodo
- Nel corpo del metodo vengono verificate alcune precondizioni riguardanti gli eventuali parametri che vengono passati o variabili memorizzate
    - Nel caso in cui alcune di queste precondizioni non si verifichino, il metodo ritornerà il valore `false` e per riconoscere la precondizione che non è stata rispettata, viene registrato un errore che si potrà ottenere con un'opportuna funzione dell'API la quale permette di ottenere l'ultimo errore riscontrato all'interno di tutto l'API

Di seguito un esempio con codice e commento di un generico metodo dell'API.

```
public bool method ( string param )
{
	try
	{
		// verifica delle precondizioni
		// ed eventuale registrazione
		// dell'errore
	}
	catch (Exception E)
	{
		Trace.Assert(false, $"{ E.Source }\n { E.Message }");
	}
	return answer;
}
```

3.9.3 Funzioni per iTAC

Un primo corpo di metodi è definito da quelli che servono a comunicare con il MES tramite opportuni comandi della suite iTAC. Su richiesta del cliente, sono necessari solo tre comandi iTAC da inviare al MES e la connessione deve avvenire dal solo supervisor. Vengono utilizzati i seguenti esempi di metodi:

- **Un metodo di inizializzazione**: vengono controllati e memorizzati i dati necessari a stabilire una connessione col MES (indirizzo IP del server e numero della porta)
- **Tre metodi, ognuno dei quali invia un comando diverso:** permette di connettersi al MES, inviare il primo comando con dei parametri validi, ricevere una risposta dal MES, memorizzare tale risposta, restituirla al programma chiamante e chiudere la connessione con il MES
- **Un metodo di terminazione:** chiude ogni tipo di connessione rimasta aperta con il MES

La comunicazione col server avviene tra un `TcpClient` e relativo `NetworkStream` (utilizzati dai metodi che inviano i comandi iTAC)

```
TcpClient _client = new TcpClient();
_client.Connect(new IPEndPoint(IPAddress.Parse(_tcpIP), _tcpPort));
NetworkStream _clientStream = _client.GetStream();
```

e un `TcpListener` (utilizzato dall'emulatore del server da me creato per verificare il corretto funzionamento delle funzioni dell'API). Quest'ultimo è di utilizzo meno immediato in quanto occorre che venga gestito su più thread, tramite opportuni "*event handler*", per permettere connessioni multiple da parte di uno o più client.

```
// main
TcpListener _server = new TcpListener(new IPAddress.Parse(_ip), _port);
new Thread(new ThreadStart(Listening)).Start();

// thread 1
void Listening()
{
	_server.Start();
	for(;;)
	{
		TcpClient client = _server.AcceptTcpClient();
		new Thread(new ParameterizedThreadStart(HandleClientComm)).Start(client);
	}
}

// thread 2
void HandleClientComm(object client)
{
	TcpClient tcpClient = (TcpClient)client;
	NetworkStream clientStream = tcpClient.GetStream();
	for(;;)
	{
		// comunicazione col client
	}
}
```

3.9.4 Funzioni per MSMQ

Queste funzioni permettono di aprire una certa applicazione (e quindi una certa macchina) alla scrittura e/o alla lettura di messaggi dal sistema di code di Microsoft. Affinché queste operazioni vadano a buon fine è necessario verificare preventivamente che le code siano state create manualmente tramite gli strumenti offerti da Microsoft sulle macchine interessate. L'API comprende le seguenti funzioni:

- **Un metodo di inizializzazione**: vengono controllati e memorizzati i dati necessari a identificare una coda di Microsoft (indirizzo IP del computer su cui si trova e nome della coda)
- **Un metodo che permette di scrivere un messaggio su una coda esistente**
- **Un metodo che permette di leggere un messaggio da una coda esistente**
- **Un metodo di controllo:** verifica che una certa coda sia stata precedentemente inizializzata o meno

- **Un metodo di pulizia:** elimina qualunque messaggio che sia rimasto su una determinata coda, rendendola vuota
- **Un metodo di terminazione:** chiude qualunque coda sia stata precedentemente inizializzata

3.9.5 Funzioni per il cliente

Quest'ultimo corpo di funzioni è più particolare dei precedenti in quanto implementa un API creato dal cliente e, conseguentemente, necessita di un file di configurazione in cui inserire alcuni parametri necessari per i metodi dell'API del cliente.

Obiettivo principale di questi metodi è quello di leggere i file in formato XML di riepilogo dei test generati da VIVA, elaborare queste informazioni e memorizzarle, tramite l'API del cliente, in un nuovo file TXT secondo un formato richiesto dal cliente. Nel file di configurazione (formato INI) vanno introdotti dei parametri esterni che vanno aggiunti a quelli dei metodi per memorizzare i risultati dei test.

Vengono utilizzati i seguenti metodi:

- **Un metodo di inizializzazione:** si occupa di preparare la macchina corrente alla memorizzazione dei nuovi file. Crea la cartella in cui andare a memorizzarli, legge dal file di configurazione i parametri aggiuntivi e crea un nuovo oggetto di una classe "*client*" definita dall'API del cliente
- **Un metodo per la lettura, l'elaborazione e la memorizzazione dei dati:** tramite il file di configurazione è noto il percorso dei file XML generati da VIVA, i quali vengono letti ed elaborati e, dopo opportuni controlli, memorizzati con i metodi dell'API del cliente
- **Un metodo di terminazione:** libera le risorse trattenute dall'oggetto "*client*" e chiude il file TXT in cui memorizza i dati

3.10 Supervisor

Ultimata la programmazione degli applicativi e verificatone il corretto funzionamento tramite opportuni emulatori, le applicazioni sono pronte per essere installate sui rispettivi computer, garantendone la gestione attraverso un unico computer supervisor.

Le applicazioni M1, M2, M8 vengono installate direttamente sul supervisor, gestendo l'inizio e la fine di ogni singolo ciclo di test; l'applicazione M3 viene installata sulla macchina che effettua il primo test; le applicazioni M4 e M5 sulla seconda macchina; le applicazioni M6 e M7 sull'ultima macchina.

La gestione di queste applicazioni e delle rispettive macchine su un unico computer supervisor è garantita tramite la connessione in desktop remoto fornita da Windows, gestita "dividendo" lo schermo del supervisor in 4 parti, una per ogni macchina che partecipa al processo di test (immagine 3.11).

GUI (Graphic User Interface) del supervisor:

1) Il riquadro in alto a sinistra è ottenuto connettendo in desktop remoto la prima macchina che effettua il test ICT
2) Il riquadro in alto a destra è proprio del supervisor ed è diviso in tre tabelle, una per ogni applicazione che lavora sul supervisor (M1, M2 e M8)
3) Il riquadro in basso a sinistra è ottenuto connettendo in desktop remoto la seconda macchina che effettua il test Flash
4) Il riquadro in basso a destra è ottenuto connettendo in desktop remoto la terza macchina che effettua il test LED
5) La prima tabella del supervisor è riferita alla prima applicazione (M1) *loader*
6) La seconda tabella del supervisor è riferita alla seconda applicazione (M2) *barcode reader*
7) La terza tabella del supervisor è riferita all'ottava applicazione (M8) *unloader*
8) L'interfaccia del software VIVA, simile per le tre macchine che effettuano i test
9) Interfaccia della terza applicazione (M3) prima di essere avviata
10) Interfaccia della quarta applicazione (M4) subito dopo essere avviata
11) Interfaccia della quinta applicazione (M5) subito dopo essere avviata
12) Interfaccia della sesta applicazione (M6) prima di essere avviata
13) Interfaccia della settima applicazione (M7) prima di essere avviata

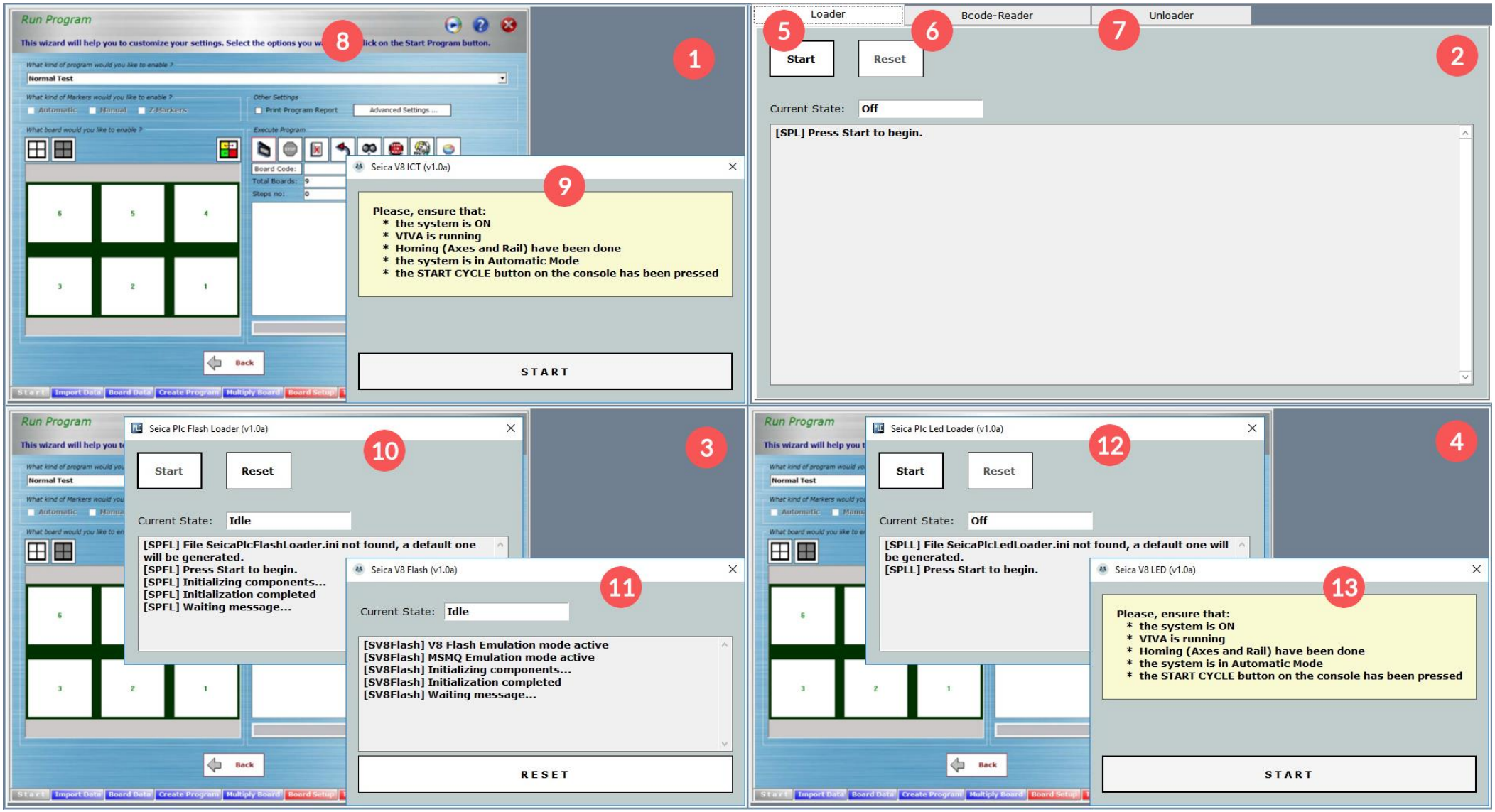

Immagine 3.11 GUI del supervisor

Capitolo 4

Conclusioni

L'esperienza di tirocinio è stata indubbiamente molto formativa soprattutto dal punto di vista professionale. Oltre ad ampliare le mie conoscenze informatiche pregresse, quest'esperienza è stata l'occasione per imparare a confrontarmi oltre che con il mio gruppo di lavoro anche con altri settori dell'azienda o persone esterne ad essa.

Il tutor aziendale, e in generale tutti i dipendenti appartenenti al mio gruppo di lavoro, si sono sempre dimostrati estremamente disponibili per chiarire dubbi e perplessità, nonché in grado di farmi sentire a mio agio all'interno dell'ambiente lavorativo nonostante la mia inesperienza e insicurezza trattandosi della mia prima esperienza in campo lavorativo.

Questa esperienza è stata l'occasione per dimostrare le mie conoscenze e soprattutto per ampliarle con le nozioni che il tutor e gli altri dipendenti mi hanno fornito. Facendo tesoro di questo, ho imparato a lavorare in un "team" di persone, dove ognuno svolge un compito ben specifico per poi confrontare e mettere in relazione i propri lavori per il raggiungimento del risultato finale.

In conclusione, posso dire che il tirocinio è un'esperienza sicuramente utile per chiunque intraprenda questa strada in ambito lavorativo ma non solo, perché aiuta a relazionarsi con altre persone, ad avere un confronto, ad aiutare e ad essere aiutati. È un'esperienza che sicuramente consiglierei a chiunque per tutti questi motivi e perché è la prima che permette di avvicinarsi così tanto al mondo del lavoro, permettendo di capire quali sono i meccanismi principali all'interno di un'azienda e come il lavoro viene suddiviso e gestito tra i vari membri di questa.