POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Elettronica

Tesi di Laurea Magistrale

Realizzazione di un robot cartesiano a tre assi per test automatico di sensori IoT

Relatore:
Prof. Maurizio Zamboni

Candidato:
Matteo Loro

Marzo 2019

Ringraziamenti

Desidero ringraziare tutte le persone con cui ho lavorato in Sysdev per avermi permesso di realizzare il progetto di tesi nella loro azienda, seguendomi sempre con la massima disponibilità e gentilezza e permettendomi di realizzare questo progetto.
Desidero infine ringraziare la mia famiglia e tutte le persone che mi hanno accompagnato in questo percorso per il loro sostegno nei momenti difficili e per aver condiviso con me le esperienze di crescita.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Il progetto, svolto presso Sysdev srl, ha il compito di realizzare un dispositivo di test e configurazione automatico di sensori IoT prodotti dall'azienda. I sensori da testare vengono utilizzati per il monitoraggio di strutture in cemento armato; l'obiettivo è realizzare una struttura di monitoraggio a basso costo e modulare, per ridurre i costi si realizzano grosse produzioni di sensori.
Da questa necessità nasce l'idea di realizzare un dispositivo di test e validazione automatico dei sensori, chiamato LR-TEST, da poter fornire ai produttori, che permetta, dopo il montaggio, il test e la configurazione automatica del sensore appena prodotto.

Capitolo 2

Richieste di progetto

LR-TEST deve essere utilizzata da aziende esterne che producono e assemblano i sensori, quindi viene realizzato un dispositivo completo da fornire al produttore; è necessario che la macchina sia di facile da utilizzare per il collaudatore, significa che bisogna realizzare una struttura grafica intuitiva, che minimizzi gli errori durante l'uso, che sia di rapido apprendimento e che verifichi in autonomia il corretto funzionamento della struttura. Infine, siccome LR-TEST può essere utilizzata sia in Italia sia all'estero bisogna sviluppare una struttura multilingua (italiano e inglese); inoltre, per verificare il funzionamento e controllare il flusso di produzione bisogna raccogliere, attraverso cloud, tutte le informazioni necessarie sul funzionamento sia di LR-TEST sia dei sensori (DUT), questa operazione permette di individuare malfunzionamenti sistematici in LR-TEST o nel test del DUT, e fornisce statistiche sulla produzione dei sensori. LR-TEST è strutturata come in Figura 2.1.

Nei paragrafi seguenti vengono riportati tutti i dettagli sui componenti utilizzati e sulla realizzazione del progetto.

NB: le componenti hardware, le parti meccaniche e i software di sviluppo sono stati scelti da Sysdev, quindi è stato necessario imparare ad utilizzare sia i software sia l'hardware utilizzato.

2.1 Componentistica hardware

2.1.1 Motori Igus Drylin

Per effettuare il movimento della testa del robot vengono utilizzati quattro motori passo-passo abbinati a delle guide lineari, sugli assi X e Z viene utilizzato un singolo motore; mentre sull'asse Y/U vengono utilizzati due motori che si muovono parallelamente, questo asse è quello che deve trasportare il peso maggiore.

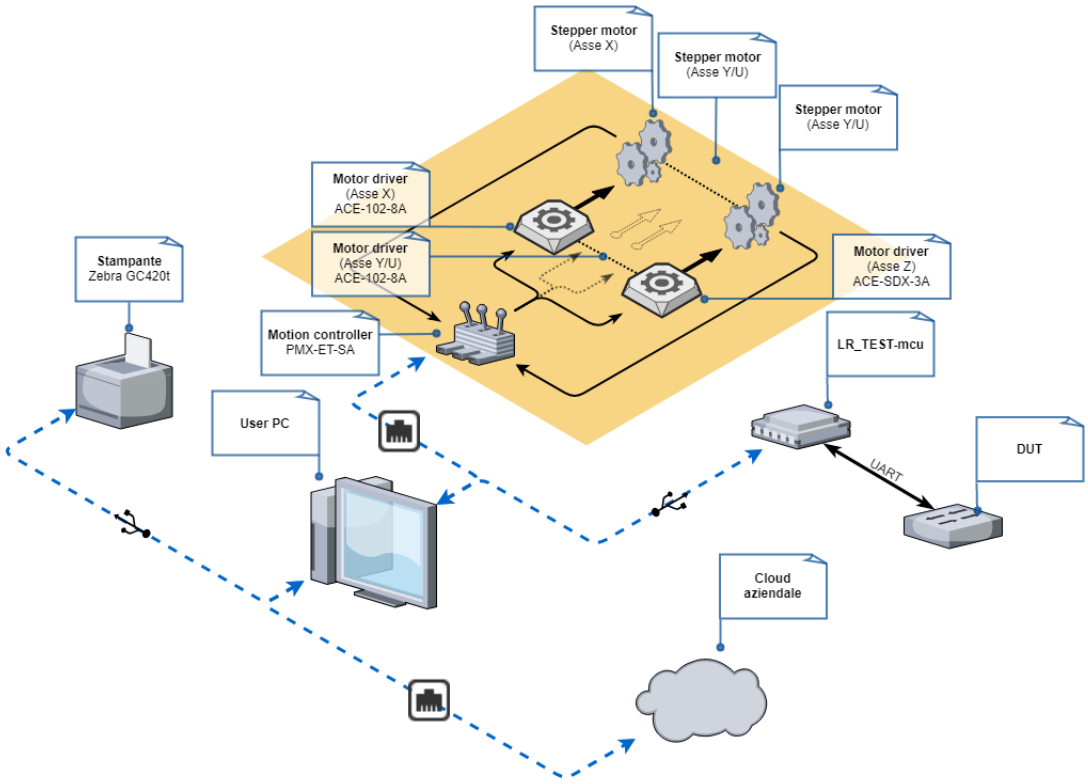

Figura 2.1: Struttura LR-TEST

Ogni motore è connesso ad un driver di potenza che gestisce la corrente nelle fasi, inoltre ogni motore è dotato di un decoder incrementale, quest'ultimo è connesso al driver di controllo del movimento dei motori e serve per verificare la corretta posizione del motore.
All'inizio di ogni guida lineare è presente uno switch di homing, serve per effettuare la calibrazione dei motori all'accensione ed assegnare la posizione iniziale di riferimento per ogni asse; ogni switch è connesso al driver di controllo di movimento.

Di seguito vengono riportate nel dettaglio le caratteristiche di ogni asse:

- **Asse X:** viene utilizzato un singolo motore (potenza nominale 4,2A) montato su una guida lineare a cinghia con escursione massima di 400 mm.
- **Asse Y/U:** vengono utilizzati due motori (potenza nominale 4,2A ciascuno) posti parallelamente e montati su una guida lineare a vite senza fine con escursione massima di 400 mm.
- **Asse Z:** viene utilizzato un singolo motore (potenza nominale 1,8A) montato su una guida lineare a vite senza fine con escursione massima di 100 mm.

In questa fase sono stati scelti motori e guide sovradimensionati per avere maggiore flessibilità in fase di sviluppo, durante il progetto bisogna anche verificare se è possibile diminuire le dimensioni e la potenza delle varie componenti.

2.1.2 Driver motori ACE

Per il controllo di potenza dei motori vengono utilizzati quattro driver, abbinati nel modo seguente:

- **Assi X Y/U:** vengono utilizzati 3 driver (uno per motore) ACE-102-8A, in grado di fornire massimo 8A e settabili attraverso switch per erogare la potenza corretta.
- **Asse Z:** viene utilizzato un ACE-SDX-3A, in grado di fornire massimo 3A e configurabile collegandolo al PC attraverso specifico cavo USB.

Ogni driver deve essere settato per erogare una corrente non superiore alla massima corrente supportata dai motori; infine verificare se è possibile far lavorare i

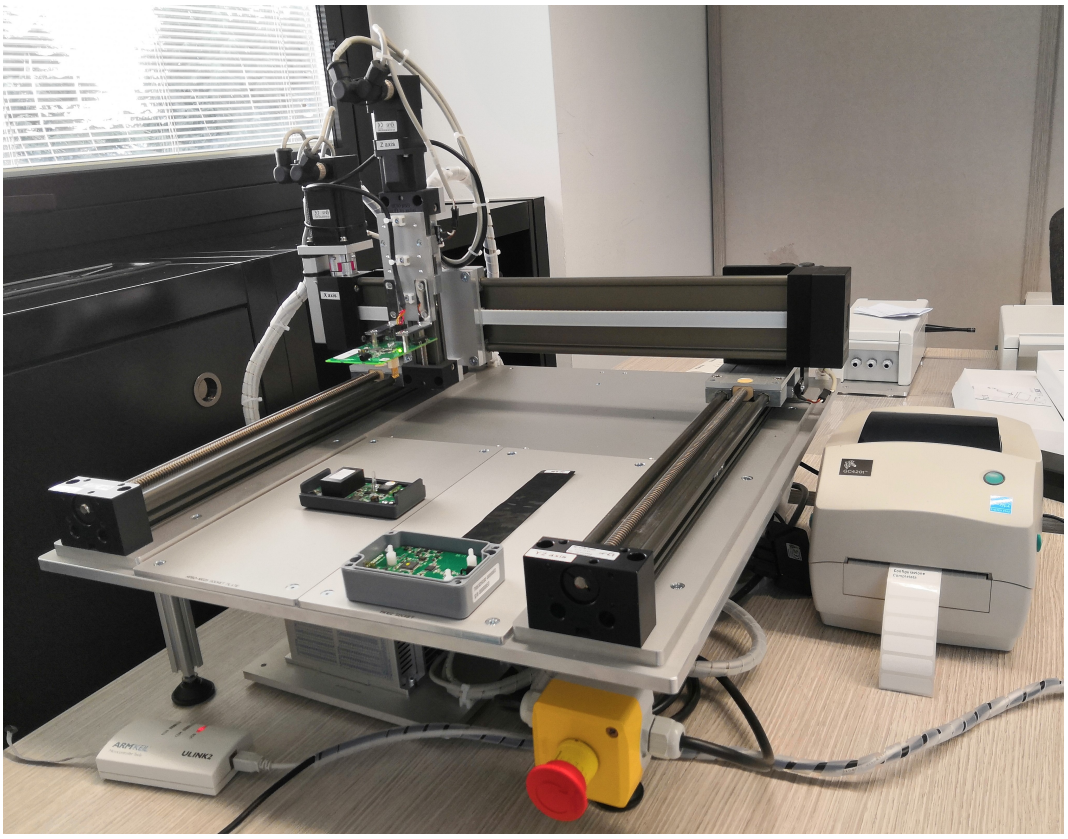

Figura 2.2: LR-TEST

motori con una corrente inferiore alla massima supportata (per ridurre la coppia ed evitare danni alla macchina in caso di movimenti errati).

2.1.3 Driver controllo di movimento PMX-4ET-SA

Per il controllo del movimento dei motori viene utilizzato il controller PMX-4ET-SA; la connessione al PC avviene tramite ethernet e il driver viene comandato attraverso LabView. Per il controllo viene utilizzata la libreria LabView del driver per l'invio e la ricezione dei comandi, mentre le operazioni da eseguire vengono inviate attraverso stringhe proprietarie (sul datasheet è presente il linguaggio di programmazione per la comunicazione con il componente).

2.1.4 Stampante Zebra GC420t

Su ogni sensore è necessario applicare le etichette contenenti le informazioni di base sul sensore e le marchiature di conformità, per la stampa di queste etichette viene utilizzata una stampante Zebra GC420t. Con questa stampante è possibile stampare etichette personalizzate resistenti agli agenti atmosferici, la grafica delle etichette viene realizzata un linguaggio di programmazione proprietario (EPL).

Per ogni sensore devono essere stampate le seguenti etichette:

- N 1 etichetta contente logo aziendale, tipo di sensore e versione del firmware, marchiatura CE e RAEE. Questa etichetta va applicata sul sensore.

- N 2 etichette contenti DevEUI e codice a barre dello stesso; utile per la lettura automatica dei codici nella fase di gestione del magazzino. Queste etichette vanno applicate sul sensore e sulla confezione dello stesso.

Sviluppare inoltre un applicativo che permetta di stampare manualmente le etichette; nel programma deve essere presente la possibilità di effettuare più copie di un'etichetta e la possibilità di stampare etichette con DevEUI e Serial Number in sequenza partendo da uno scelto dall'utente.

Figura 2.3: Stampante Zebra GC420t

2.1.5 LR-TEST mcu

LR-TEST-mcu è posizionata sulla testa del robot; permette la connessione tra DUT e PC (facendo da ponte) ed effettua alcune operazioni di test e memorizzazione. Nello specifico viene collegata al PC tramite connessione seriale con Baud Rate di 37400; può ricevere due tipologie di comandi; i comandi che iniziano con il carattere "$" sono usati per effettuare operazioni su LR-TEST-mcu, mentre gli altri comandi devono essere inviati al DUT tramite una seconda UART che funziona con lo stesso Baud Rate (in questa fase LR-TEST-mcu viene usata come ponte). Per ogni comando viene inviata una risposta al PC dopo averlo eseguito.

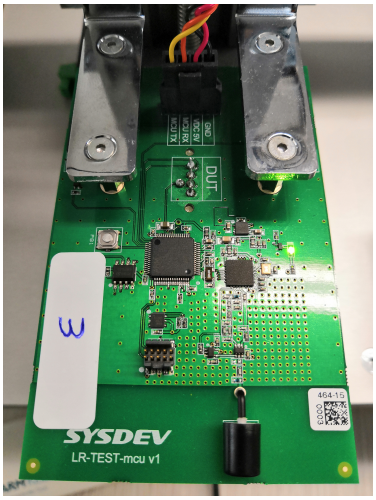

Figura 2.4: LR-TEST-mcu

LR-TEST-mcu deve essere in grado di compiere le seguenti operazioni:

- **Controllo posizionamento del DUT:** attraverso un circuito di riconoscimento è possibile verificare se il DUT è connesso correttamente con LR-TEST-mcu oppure no, questa verifica deve essere effettuata prima di accendere il DUT per evitare errori di posizionamento (disallineamento o inserimento del DUT ruotato di 180°; il riconoscimento è reso possibile tramite un pin di connessione aggiuntivo che effettua il riconoscimento anche con il DUT spento.
- **Accensione e spegnimento del DUT:** attraverso due relè montati sulla scheda è possibile accendere e spegnere il DUT attraverso due appositi comandi, la sequenza di accensione deve essere: attivazione del relè Vcc, attesa

di 3ms, attivazione del relè di GND. Analogamente per lo spegnimento invece: disattivazione del relè Vcc, attesa di 3ms, disattivazione del relè di GND.

- **Reset LR-TEST-mcu:** il reset di LR-TEST-mcu può avvenire tramite apposito comando inviato tramite UART, oppure tenendo premuto il bottone presente sulla scheda per 5sec; prima di effettuare il reset LR-TEST-mcu manda un comando di risposta al PC e il led effettua cinque lampeggi a 1Hz (0,5 sec on, 0,5 sec off).

- **Misura temperatura:** attraverso apposito comando la scheda misura la temperatura ambiente grazie ad un sensore montato su di essa e la invia al PC; non è necessario effettuare letture ripetute perché il dato del sensore è già stabilizzato.

- **Lettura segnale LoRa:** LR-TEST-mcu deve leggere la potenza (espressa in dB) che riceve l'antenna LoRa integrata su di essa, questa operazione serve in fase di test per verificare che il DUT trasmetta correttamente dati attraverso LoRa.

- **Trasmissione continua LoRa:** attraverso due comandi LR-TEST-mcu deve abilitare e disabilitare la trasmissione continua di un segnale LoRa, questo permette di verificare il corretto funzionamento della scheda stessa.

- **Self test:** attraverso questo comando LR-TEST-mcu deve verificare il corretto funzionamento di tutte le sue componenti e inviare il risultato del test al PC; nella verifica deve controllare, il sensore di temperatura, il tranceiver LoRa e la EEPROM integrata nella scheda.

- **Lettura/scrittura EEPROM:** nella EEPROM devono essere memorizzati i parametri di offset, rispetto alla calibrazione di default, quando viene effettuata una ricalibrazione dei motori; questi valori sono espressi in millimetri (float) e devono essere salvati nella EEPROM per evitare la perdita della ricalibrazione allo spegnimento della macchina. All'accensione i valori vengono letti dal PC e vengono aggiornati i valori di calibrazione, in questo modo anche se il PC viene sostituito i valori di calibrazione rimangono associati alla macchina.

- **Invio/ricezione comandi sul DUT:** Ogni comando che non inizia con il "$" viene ritrasmesso tramite la seconda UART al DUT, dopo la trasmissione del comando LR-TEST-mcu attende una risposta dal DUT, nel caso la risposta non arrivi entro il timeout impostato la scheda risponde con un errore di timeout sulla seconda UART e può nuovamente ricevere comandi dal PC. Durante la fase di attesa di risposta del DUT (cioè quando la seconda UART è in ricezione)il led integrato nella scheda si spegne, così da poter riconoscere la fase di attesa di risposta del DUT.

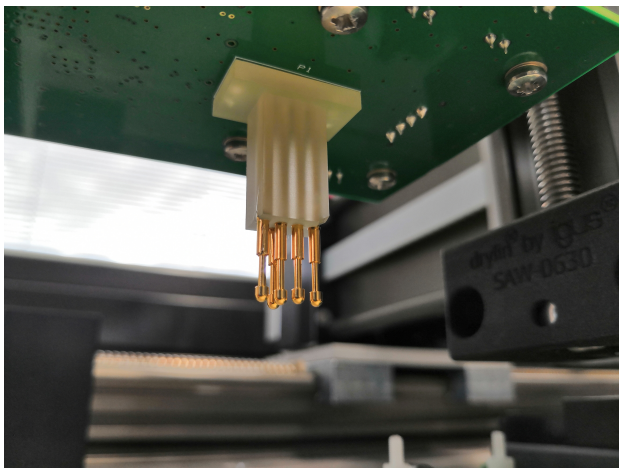

Figura 2.5: pin connessione LR-TEST-mcu

2.1.6 PC

Per minimizzare i problemi di compatibilità e di configurazione alla macchina di test viene abbinato un PC sul quale, in precedenza, vengono installati tutti i software e i driver necessari al corretto funzionamento della macchina. Il PC deve essere configurato in modo da poter essere utilizzato solo per il funzionamento della macchina di test, l'utente non può effettuare altre operazioni (per esempio installare altri programmi). Il PC da utilizzare è un Advantech ARK-2150F; al quale viene abbinato mouse, tastiera e schermo.

2.1.7 DUT

I DUT da testare sono sensori IoT per il monitoraggio di strutture in cemento armato, esistono quattro tipologie di DUT:

- **LR-SS:** sensore a striscia in cui è presente un elastometro per il monitoraggio di deformazioni, ha una risoluzione elevata ma un range ristretto (Figura 7.5).
- **LR-SS-F:**sensore a pistone resistivo per il monitoraggio di deformazioni, esistono quattro lunghezze di pistone, 25mm, 50mm, 100mm, 200mm; rispetto a LR-SS ha una risoluzione minore ma maggiore portata (Figura 7.8).
- **LR-SS-T:** sensore con accelerometro ad alta risoluzione per il monitoraggio di inclinazioni su tre assi.
- **LR-SS-C:** sensore galvanico per il monitoraggio della corrosione del metallo.

Figura 2.6: LR-SS

Su ogni tipologia di sensore devono essere testate le proprie componenti e successivamente deve essere configurato e validato sul cloud; la sequenza di test è la seguente:

Figura 2.7: LR-SS-F

- **Apertura della connessione:** verifica che il sensore sia connesso e apre la comunicazione tramite UART attraverso specifica PassKey

- **Test sensore temperatura:** su ogni DUT è presente un sensore di temperatura, per il test si chiede la temperatura del sensore e la si confronta con la temperatura di riferimento misurata da LR-TEST-mcu

- **Test accelerometro:** su ogni DUT è presente un accelerometro, per verificare il componente, il DUT esegue un self test e invia la risposta tramite UART

- **LR-SS - Test MUX - Test potenziometro - Test A/D:** solo su LR-SS sono presenti un potenziometro, un MUX e un convertitore A/D, per verificare i componenti il DUT esegue un self test e invia la risposta tramite UART. Il test di questi componenti verifica anche il corretto montaggio della striscia (elastometro) sul DUT.

- **LR-SS-F - Test A/D:** solo su LR-SS-F è presente un convertitore A/D, per verificare il componente il DUT esegue un self test e invia la risposta tramite UART.

- **LR-SS-T - Test accelerometro HR:** solo su LR-SS-T è presente un accelerometro ad alta risoluzione, per verificare il componente il DUT esegue un self test e invia la risposta tramite UART.

- **Test LoRa:** il DUT effettua una prova di trasmissione con la LR-TEST-mcu per verificare il corretto funzionamento della trasmissione tramite LoRa

- **Configurazione del DUT:** se tutti i test sono stati superati il DUT deve essere configurato scrivendo tutti i parametri ricevuti dal Cloud al suo interno (per la lista dei parametri riferirsi alle specifiche del Cloud).

Dopo aver completato il test e la configurazione vengono stampate le relative etichette e viene effettuata la validazione sul Cloud attraverso apposita service.

2.2 Software

Il software da installare sul PC dovrà essere sviluppato utilizzando LabView 2017-32 bit, si utilizza suddetta versione per avere maggiore compatibilità in caso di PC con sistema operativo 32 bit. Il software deve essere sviluppato utilizzando la struttura Queued message handler, presente il LabView, questo permette di creare un programma che lavora su più thread paralleli che interagiscono tra loro. La parte grafica deve essere implementata utilizzando il pacchetto silver di icone presente nell'applicativo. L'utilizzo di un software di programmazione grafico, come LabView, permette di sviluppare velocemente strutture complesse di codice focalizzandosi sugli aspetti esecutivi del programma e non sulle strutture sottostanti (gestione dei thread e interfaccia grafica). Di seguito viene riportato un semplice esempio del software utilizzato e viene analizzata una problematica di temporizzazione (elemento fondamentale in questo progetto): Il codice scritto deve eseguire le seguenti operazioni:

$$Number1 + Number2 = Result \quad \rightarrow \quad 2 + 3 = 5$$

$$Result + Number3 = Result \quad \rightarrow \quad 5 + 1 = 6$$

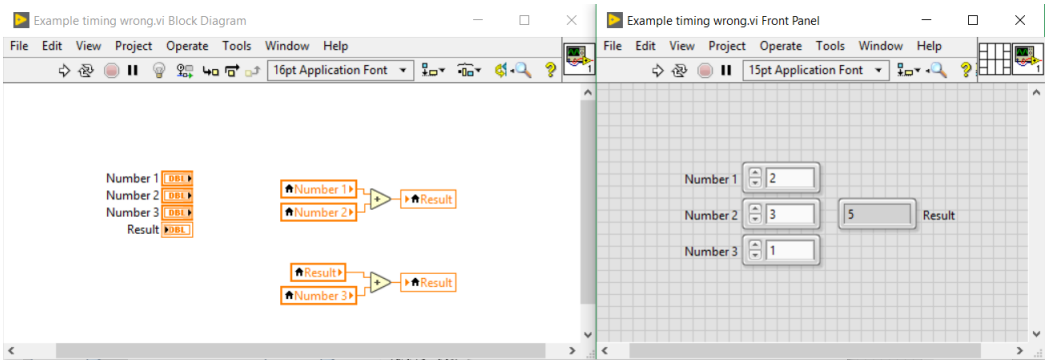

Figura 2.8: Esempio di timing errato

In Figura 2.8 non vengono assegnate delle tempistiche, quindi il compilatore decide autonomamente quali operazioni eseguire prima, in questo caso:

$$Result + Number3 = Result \quad \rightarrow \quad ultimo\ risultato\ in\ memoria + 1 = indefinito$$

$$Number1 + Number2 = Result \quad \rightarrow \quad 2 + 3 = 5$$

In Figura 2.9 viene assegnato un ordine, quindi le operazioni seguono il corretto flusso di esecuzione.

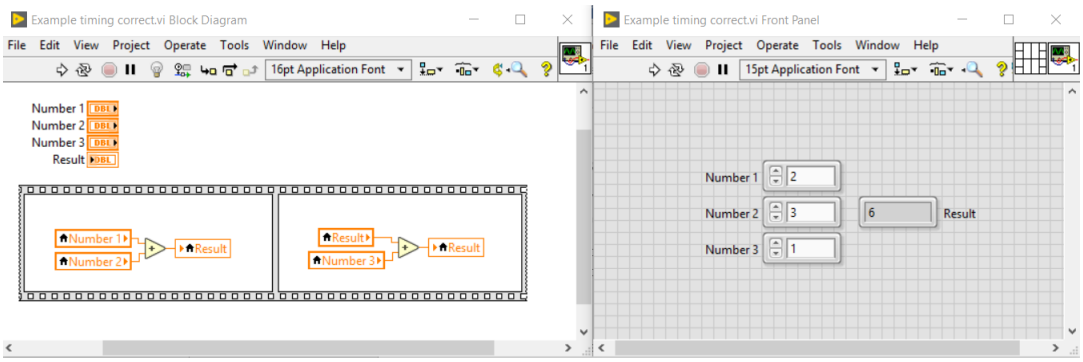

Figura 2.9: Esempio di timing corretto

2.2.1 Interfaccia utente

L'interfaccia utente è suddivisa in tre parti, come in figura Figura 2.10:

- **Header:** contiene i led che indicano lo stato di connessione delle componenti hardware, i bottoni di connessione, la posizione dei motori in millimetri, il logo aziendale e il bottone di arresto dei motori in caso di emergenza.
- **Body:** è suddiviso in tab; all'apertura apparirà il tab home, in cui è possibile effettuare la connessione al cloud da parte dell'utente; inoltre dovranno essere presenti un tab setting, in cui sono presenti le impostazioni del software; un tab calibration, per il movimento manuale del robot e sua calibrazione; infine un tab test sensor, in cui si esegue la procedura di test del sensore.
- **Footer:** è presente una casella che indica le informazioni sullo stato del sistema e il tasto di uscita dal programma.

L'interfaccia utente deve essere sviluppata in modo da minimizzare i possibili errori da parte dell'utilizzatore e ridurre i tempi di apprendimento nell'utilizzo del software. Per realizzare questa struttura viene implementata un'interfaccia sensibile al contesto, cioè i vari elementi visibili all'utente sono da lui utilizzabili solo in

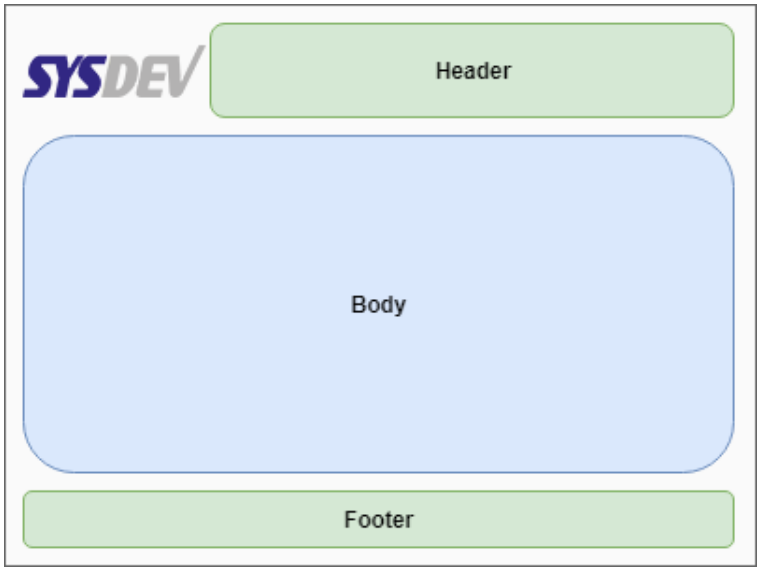

Figura 2.10: Struttura interfaccia grafica

specifiche fasi; per esempio non è possibile accedere alla fase di test se prima non si sono verificate le connessioni a tutte le componenti hardware.
Per semplificare la risoluzione di eventuali errori o evidenziare operazioni importanti devono essere visualizzati messaggi pop-up esplicativi, con cui l'utente può, nei casi più comuni, comprendere cosa sta accadendo e risolvere l'eventuale problema; per esempio in caso di malfunzionamento di un componente hardware deve apparire un messaggio che dichiara il tipo di malfunzionamento e mostra i problemi comuni a quel componente, in modo che l'operatore possa risolvere in autonomia il problema. La messaggistica pop-up è anche utilizzata per confermare tutte le operazioni rilevanti che l'operatore può effettuare. Il sistema deve essere sviluppato interamente sia in inglese sia in italiano, l'utente può scegliere la lingua di sistema nella pagina impostazioni del software. È necessario l'utilizzo della doppia lingua, perché la macchina di collaudo potrebbe essere utilizzata anche da produttori esteri.

2.2.2 Funzioni principali del programma per l'utente

All'avvio del programma deve visualizzarsi una schermata di home in cui l'utente può effettuare la connessione al cloud inserendo la sua chiave (Username), la chiave è una stringa alfanumerica composta da 36 caratteri; vista la complessità di memorizzazione della chiave, l'utente ha la possibilità di memorizzare la chiave per i successivi avvii del programma attraverso una spunta "mantieni in memoria".
Solo dopo aver effettuato correttamente la connessione al cloud sarà possibile connettere le componenti della macchina attraverso appositi bottoni. Al momento della connessione LR-TEST-mcu effettuerà un self test, la stampante una stampa di prova e il robot l'homing dei motori; se tutte le componenti funzionano correttamente verranno abilitate la fase di test e di ricalibrazione dei motori, mentre la pagina delle impostazioni rimane sempre accessibile all'utente.

Nella fase di test sono presenti tutti i valori che vengono acquisiti dal sensore, più una barra di stato che indica le varie fasi del test. La procedura di test è automatica, l'utente dovrà solo inserire i parametri di calibrazione iniziali del sensore e successivamente avviare il test. Dopo aver avviato il test il robot si posizionerà sul sensore, effettuerà il test, la configurazione e la validazione sul cloud, infine riporterà i motori in posizione di riposo ed effettuerà la stampa delle etichette necessarie in

caso di esito positivo del test.
In questa fase l'utente può monitorare le varie fasi da display, inoltre l'utente può configurare nuovamente un sensore già configurato correttamente semplicemente inserendolo nella macchina di test, senza dover eseguire ulteriori operazioni. Nella fase di ricalibrazione dei motori l'utente può memorizzare la nuova posizione del sensore in caso di disallineamento rispetto alla calibrazione iniziale. Questa operazione viene effettuata muovendo liberamente i motori, attraverso appositi comandi, per posizionarsi sul sensore; una volta raggiunta la posizione desiderata si salva la nuova posizione e questa rimarrà memorizzata nella EEPROM di LR-TEST-mcu anche per le accensioni successive della macchina, è possibile ricalibrare i motori ogni volta che si ha necessità.
Nella pagina delle impostazioni sono presenti tutte le impostazioni modificabili dall'utente (per esempio lingua dell'applicazione) e devono essere visualizzate informazioni specifiche sullo stato della macchina (per esempio posizione dei motori, valore della calibrazione).

2.2.3 Struttra del codice

Il codice deve essere sviluppato usando una struttura multithread, in cui ogni thread viene gestito come una coda di stati eseguiti in successione, è possibile chiamare gli stati dallo stesso thread oppure da altri.
La struttura da utilizzare è *Queued message handler*, template disponibile in LabView 2017. In questa struttura ogni thread viene gestito come una macchina a stati indipendente, in cui la successione degli stati viene gestita da una coda con priorità, è possibile inserire nuovi stati (message) nella coda sia dallo stesso thread sia da thread concorrenti. In Figura 2.11 viene riportato un esempio sulla gestione della queue nel codice.

Il codice è strutturato in cinque thread separati, suddivisi nel seguente modo:

- **GUI (Graphic User Interface):** thread principale, gestisce il flusso del programma, rende l'interfaccia utente sensibile al contesto, implementa la doppia lingua di visualizzazione, gestisce eventuali errori del software e della macchina.
- **MCL (Motion Control Loop):** thread che effettua il movimento e il controllo del corretto movimento dei motori, si occupa di comunicare con il driver

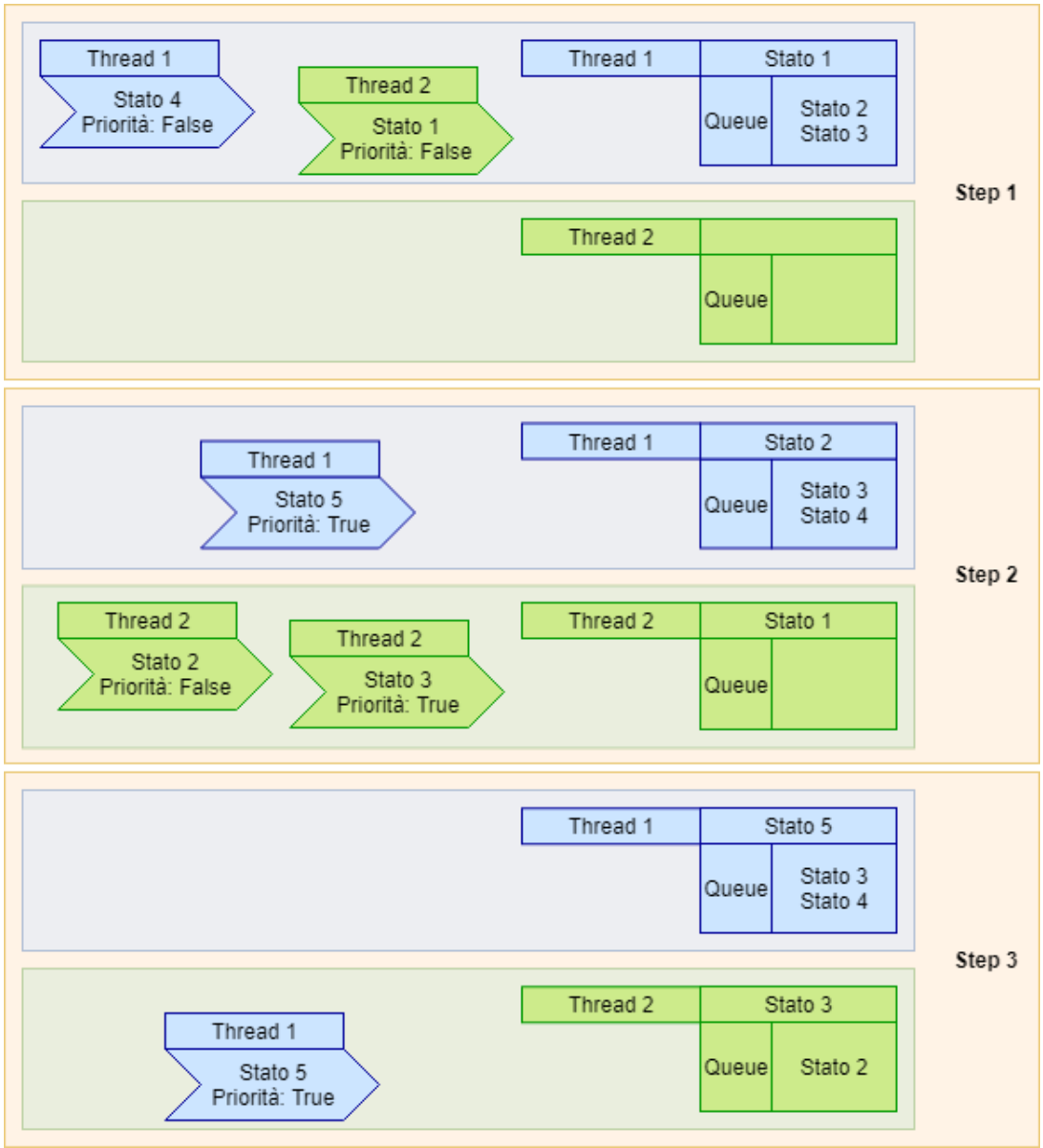

Figura 2.11: Struttura *queue message handler* di LabView

dei motori (PMX-ET-SA) e di controllarne la corretta posizione, in caso di errori o di necessità da parte dell'utente arresta i motori.

- **BCL (Board Control Loop):** thread che gestisce la comunicazione con LR-TEST-mcu e con il sensore, effettua la connessione con la scheda, invia e riceve comandi specifici da essa, invia e riceve comandi al sensore utilizzando LR-TEST-mcu come ponte.

- **CCL (Cloud Communication Loop):** thread che gestisce la comunicazione con il cloud, invia e riceve i dati di configurazione del sensore, invia log di sistema in caso di necessità.

- **PCL (Printer Control Loop):** thread che gestisce la stampa delle etichette associate al sensore, inoltre deve verificare il corretto funzionamento della stampante ed effettuare l'allineamento delle testine di stampa all'accensione.

NB: per maggiori dettagli sulle funzioni associate ad ogni thread riferirsi alle funzioni del componente interessato.
Ogni thread avrà il compito di controllare il funzionamento del componente associato e in caso di errore comunicarlo al thread principale.

2.3 Cloud aziendale

La comunicazione con il cloud viene utilizzata per richiedere i dati di configurazione di un sensore, validare il test di un sensore e comunicare vari log di sistema per comprendere eventuali malfunzionamenti della macchina o errori continui durante il test del sensore (per esempio componenti spesso non funzionanti che indicano problemi di produzione).
Ogni chiamata al cloud viene effettuata utilizzando una specifica service, protetta tramite certificato (le chiamate sono effettuate tramite HTTPS), tutte le service vengono chiamate attraverso una post e i dati scambiati devono essere in formato JSON.

2.3.1 Service richiesta dati configurazione sensore

La seguente service deve essere chiamata, attraverso una post al cloud aziendale, ogni volta che si devono ricevere dal cloud i parametri di configurazione di un sensore testato che si deve ancora configurare. Alla chiamata devono essere inviati i seguenti parametri in formato JSON:

- **Username:** nome utente dell'operatore che ha effettuato l'accesso alla macchina.
- **AppEUI:** AppEUI del sensore, sono contenute le informazioni sul tipo di sensore e la versione del software installata.
- **DevEUI:** DevEUI del sensore, identificativo univoco del sensore che si sta configurando.
- **OTAA/ABP:** prima di iniziare il test l'utente sceglie se configurare il DUT in modalità OTAA/ABP; in modalità OTAA non vengono inseriti i parametri del gateway al momento della configurazione, verranno acquisiti alla prima connessione (modalità di default di test). In modalità ABP il cloud fornisce i dati di connessione al gateway al momento della configurazione del sensore, questa modalità viene utilizzata per sensori distanti dal gateway che non riescono a ricevere correttamente la configurazione al momento dell'accensione.

Quando i parametri sono stati inviati, il cloud controlla (attraverso il DevEUI) che il sensore non sia già stato precedentemente configurato e registrato, in tal caso rimuove la precedente configurazione; una volta effettuato tale controllo risponde. La risposta avviene attraverso una stringa in formato JSON in cui sono contenuti i seguenti parametri:

- **AppEUI:** AppEUI del sensore, sono contenute le informazioni sul tipo di sensore e la versione del software installata. Viene restituito tale valore per successiva verifica prima di eseguire l'installazione dei parametri sul sensore.
- **DevEUI:** nuovo DevEUI del sensore, identificativo univoco del sensore che si sta configurando. Deve essere memorizzato nel sensore e il precedente valore di DevEUI viene eliminato.

- **AppKey:** AppKey del sensore, chiave univoca che serve per connettere il sensore alla rete LoRa durante il funzionamento, con questo parametro si identifica il sensore che sta effettuando una comunicazione.
- **DevAddr:** viene utilizzato per identificare il sensore all'interno della rete.
- **NwkSKey:** viene utilizzato per criptare i pacchetti LoRa dal sensore al Network server e viceversa, la coppia DevAddr e NwkSKey deve essere univoca.
- **AppSKey:** viene utilizzata per criptare il contenuto dei messaggi (payload).
- **Parameter:** campo che attualmente viene lasciato libero, utile in caso di ulteriori parametri che il cloud dovrà comunicare alla macchina di test durante la configurazione di un sensore.

2.3.2 Service validazione dati configurazione sensore

La seguente service deve essere chiamata, attraverso una post al cloud aziendale, ogni volta che si deve validare un sensore correttamente testato e configurato. Attraverso questa chiamata il cloud memorizza il sensore e lo dichiara funzionante. Alla chiamata devono essere inviati i seguenti parametri in formato JSON:

- **Username:** nome utente dell'operatore che ha effettuato l'accesso alla macchina.
- **AppEUI:** AppEUI del sensore, sono contenute le informazioni sul tipo di sensore e la versione del software installata.
- **DevEUI:** DevEUI del sensore, identificativo univoco del sensore che è stato correttamente configurato.
- **AppKey:** AppKey del sensore, chiave univoca che serve per connettere il sensore alla rete LoRa durante il funzionamento, con questo parametro si identifica il sensore che sta effettuando una comunicazione.
- **TimeStamp:** data e ora del PC al momento della chiamata, deve essere inviata nel seguente formato YYYY-MM-DDThh:mm:ss.xxxZ (es: 2018-01-31T12:01:59.000Z).

- **OTAA/ABP:** prima di iniziare il test l'utente sceglie se configurare il DUT in modalità OTAA/ABP; in modalità OTAA non vengono inseriti i parametri del gateway al momento della configurazione, verranno acquisiti alla prima connessione (modalità di default di test). In modalità ABP il cloud fornisce i dati di connessione al gateway al momento della configurazione del sensore, questa modalità viene utilizzata per sensori distanti dal gateway che non riescono a ricevere correttamente la configurazione al momento dell'accensione.

- **DevAddr:** viene utilizzato per identificare il sensore all'interno della rete, questo valore viene configurato solo in modalità ABP.

- **NwkSKey:** viene utilizzato per criptare i pacchetti LoRa dal sensore al Network server e viceversa, la coppia DevAddr e NwkSKey deve essere univoca, questo valore viene configurato solo in modalità ABP.

- **AppSKey:** viene utilizzata per criptare il contenuto dei messaggi (payload), questo valore viene configurato solo in modalità ABP.

La service risponderà con una stringa in formato JSON contente i seguenti parametri:

- **Stato:** se la validazione è stata eseguita correttamente risponde "OK", altrimenti risponde "FAIL"

- **Message:** messaggio esplicativo del problema riscontrato, attualmente non in uso da parte della macchina di test.

2.3.3 Service log della macchina

La seguente service deve essere chiamata, attraverso una post al cloud aziendale, ogni volta che si deve inviare un log riguardante la macchina di test; nello specifico ci sono le seguenti tipologie di log:

- **Connessione dell'utente:** ogni volta che viene avviato il software l'utente deve connettersi al cloud utilizzando una specifica chiave (Username), al momento della connessione viene chiamata questa service per verificare che lo Username sia corretto e distinguere i diversi utenti.

- **Ricalibrazione motori:** ogni volta che l'utente effettua una nuova calibrazione dei motori, attraverso la specifica procedura, i nuovi valori vengono inviati al cloud e memorizzati.

- **Test sensore fallito:** ogni volta che fallisce il test di un sensore a causa di un malfunzionamento dello stesso viene inviato un log con il report dei test effettuati e di quelli falliti; in questo modo è possibile analizzare eventuali malfunzionamenti ripetuti e avere statistiche sul numero di sensori non funzionanti.

- **Errori della macchina:** ogni volta che si verifica un errore della macchina viene inviato un log che riporta lo stato attuale di tutti i componenti della macchina e lo stato precedente all'errore. Gli errori possono essere software (eventuali bug) o hardware (uno specifico componente non si riesce a connettere o non funziona correttamente), avendo a disposizione la situazione di errore e quella precedente è possibile individuare il problema e capire in che momento è stato generato anche da remoto.

I parametri che devono essere inviati alla service cambiano in base al tipo di log, sono presenti i seguenti parametri comuni a tutti i log:

- **Username:** nome utente dell'operatore che ha effettuato l'accesso alla macchina.

- **TimeStamp:** data e ora del PC al momento della chiamata, deve essere inviata nel seguente formato YYYY-MM-DDThh:mm:ss.xxxZ (es: 2018-01-31T12:01:59.000Z).

- **Action:** tipologia di log che si sta inviando:

  - CONNECTION_TEST: quando si esegue la connessione al cloud
  - RECALIBRATION: quando si effettua la ricalibrazione dei motori
  - SENSOR_TEST: quando fallisce il test di un sensore
  - MACHINE_ERROR: quando si verifica un errore nella macchina

Ai parametri comuni si aggiungono dei valori specifici in base al tipo di log:

- **Ricalibrazione motori:** devono essere inviati i valori X Y Z di offset, espressi in millimetri, della nuova calibrazione.
- **Test DUT fallito:** devono essere inviate tutte le fasi del test del sensore, indicando con una variabile intera; non ancora effettuato (‘0’), effettuato con successo (‘1’), fallito (‘2’), non necessario (‘3’). Il valore non necessario, indica, che per uno specifico tipo di DUT il test su un componente non deve essere effettuato in quanto mancante da progetto; per esempio, testando il sensore a pistone potenziometrico non deve essere effettuato il test di corretto montaggio della striscia micrometrica (elastometro) in quanto presente su una tipologia diversa di dispositivo.
- **Errori della macchina:** deve essere inviato lo stato corrente e lo stato precedente di ogni componente della macchina (per esempio stato_stampante ”non connessa”, ”in connessione”).

Capitolo 3

Software per il collaudatore

3.1 Interfaccia utente

L'interfaccia utente, sviluppata tramite LabView, è strutturata come in Figura 3.2; è suddivisa in header, body e footer. La grafica utilizzata per le icone è la libreria 'silver' presente il LabView; le dimensioni della finestra sono state progettate per un monitor 4:3 da 15', in modo che siano compatibili per la maggior parte degli schermi.
L'interfaccia utente è stata realizzata sensibile al contesto, cioè si possono effettuare solo alcune operazioni in base agli stati della macchina; per esempio, in Figura 3.1 gli unici tasti abilitati sono quelli per effettuare il login, dopo averlo effettuato (Figura 3.4) è possibile connettere le componenti hardware ed infine (Figura 3.2) è possibile accedere alle fasi di test.

3.1.1 Header

Nell'header sono presenti i seguenti elementi:

- **Logo aziendale:** posizionato in alto a sinistra dell'interfaccia grafica.
- **Tasto "Connetti":** effettua la connessione dei motori e di LR-TEST-mcu in automatico; quando il tasto viene premuto, si invia il comando di "SELF_TEST" a LR-TEST-mcu e si effettua l'homing dei motori.
- **Tasto "Connetti stampante":** serve per verificare la connessione della stampante, premendo il tasto viene inviato alla stampante il comando di allineamento della testina e si stampa un etichetta di prova (vedi Capitolo7).
- **Led "Connessione motori":** Indica lo stato di connessione dei motori:

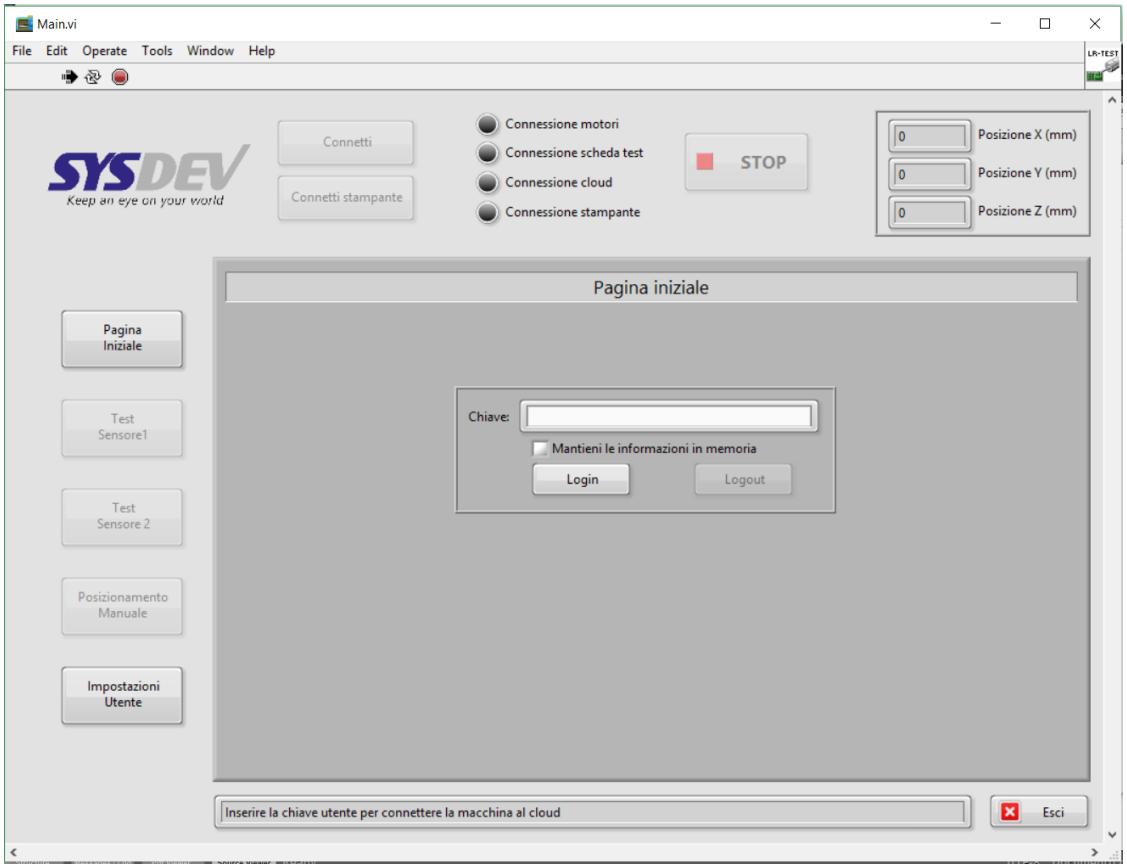

Figura 3.1: Interfaccia utente all'apertura del software

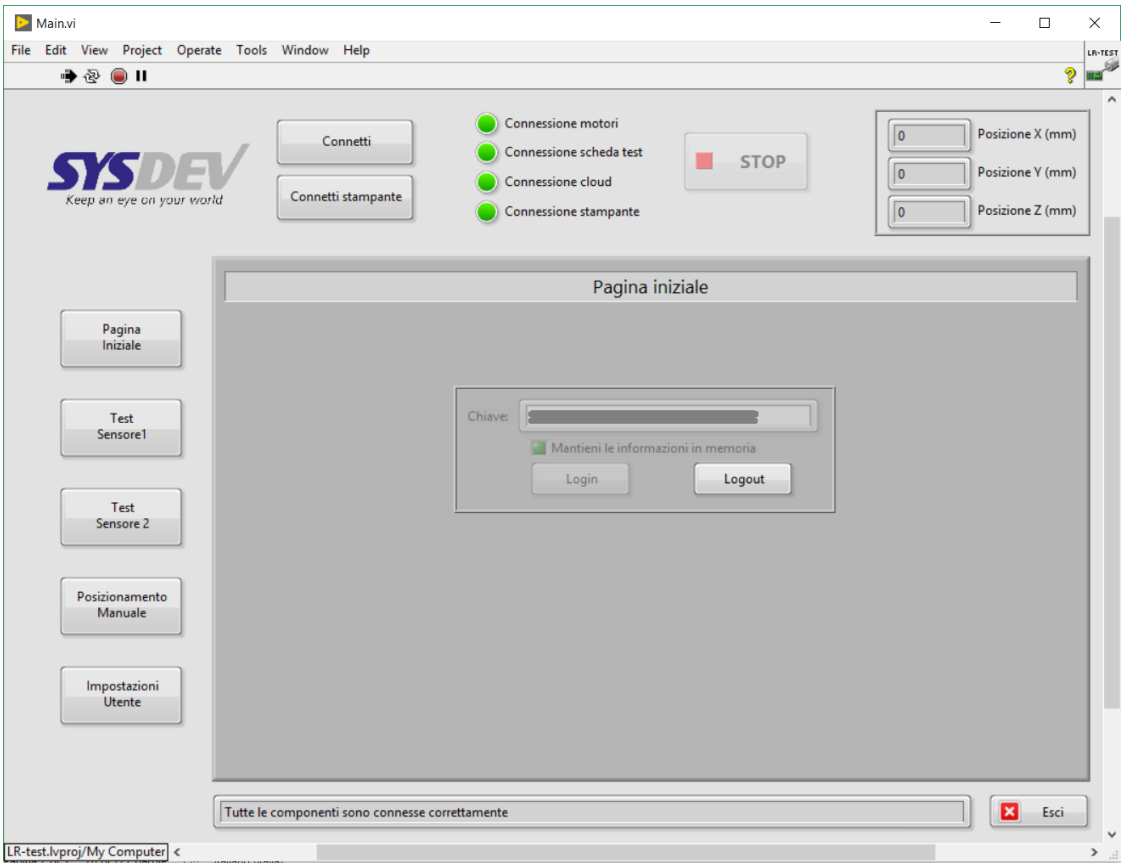

Figura 3.2: Interfaccia utente dopo il login

  - **Grigio:** motori non connessi o mancata connessione.
  - **Verde:** motori connessi e homing correttamente effettuato.
  - **Rosso:** homing non correttamente terminato o interruzione del movimento tramite tasto "Stop". In questo caso la connessione è stabile ma si è verificato un errore/interruzione durante il movimento.
  - **Giallo lampeggiante:** Motori in fase di homing.

- **Led "Connessione scheda test":** Indica lo stato di connessione di LR-TEST-mcu:
  - **Grigio:** LR-TEST-mcu non connessa o mancata connessione.
  - **Verde:** LR-TEST-mcu connessa e "SELF_TEST" correttamente effettuato.
  - **Giallo lampeggiante:** LR-TEST-mcu in fase di "SELF_TEST".

- **Led "Connessione cloud":** Indica lo stato di connessione del cloud:
  - **Grigio:** Cloud non connesso o mancata connessione ad internet da parte del PC.
  - **Verde:** Cloud connesso e accesso correttamente effettuato.
  - **Giallo:** Errore di autenticazione al cloud, è presebte la connessione ad internet ma è stata inserita una chiave sbagliata da parte dell'utente.
  - **Giallo lampeggiante:** Cloud in fase di connessione.

- **Led "Connessione stampante":** Indica lo stato di connessione della stampante:
  - **Grigio:** Stampante non connessa o mancata connessione.
  - **Verde:** Stampante connessa correttamente e stampa di prova effettuata con successo.

- **Tasto "STOP":** permette di interrompere il movimento dei motori in caso di necessità, questo interruttore viene abilitato solo quando i motori sono in movimento (al fine di realizzare un'interfaccia sensibile all'utente). Dopo aver

premuto il taso di STOP i motori andranno in fase di errore (Led rosso), sarà quindi necessario effettuare nuovamente l'homing.

- **Posizione motori:** viene visualizzata la posizione corrente dei motori, il valore è dato in millimetri e la precisione è al decimo di millimetro.

3.1.2 Body

Nel body, nella parte sinistra sono presenti i tasti di selezione dei tab accessibili all'utente; i tab sono i seguenti:

- **Pagina iniziale:** sezione iniziale in cui è possibile effettuare la connessione al cloud (Figura 3.1).
- **Test sensore 1:** sezione in cui si possono eseguire le fasi di test del sensore inserito nella posizione 1 di LR-TEST (Figura 3.6).
- **Test sensore 2:** sezione in cui si possono eseguire le fasi di test del sensore inserito nella posizione 1 di LR-TEST. Attualmente è stato predisposto il posizionamento e l'interfaccia grafica iniziale, ma non è stata sviluppata la fase di test del sensore in posizione 2.
- **Posizionamento manuale:** sezione in cui si può muovere manualmente i motori ed effettuare la ricalibrazione (Figura 3.8).
- **Impostazioni utente:** sezione in cui l'operatore può settare le preferenze sulla lingua del sistema ed eseguire alcuni settaggi e controlli sulla macchina (Figura 3.5).

3.1.3 Footer

Nel footer sono presenti i seguenti elementi:

- **Log:** vengono visualizzate le informazioni principali sullo stato della macchina e sulle successive operazioni che l'operatore deve eseguire.
- **Tasto "Exit":** permette di chiudere il programma.

3.2 Struttura del codice

Il codice, realizzato tramite la struttura queue message handler di LabView, è suddiviso in cinque thread; in ogni thread è presente una variabile di stato, che viene utilizzata per comprendere gli stati generali del software:

- **GUI (Graphic User Interface):** gestisce gli eventi generati dai tasti, la lingua, la sensibilità al contesto dell'interfaccia e lo stato generale del codice. La variabile "general status" può assumere i seguenti valori:
  - **Initialize:** indica lo stato iniziale della macchina, prima di effettuare il login e di connettere le componenti hardware.
  - **Login success:** si entra in questo stato dopo aver effettuato correttamente il login, qui viene abilitata la possibilità di connettere l'hardware; in caso di disconnessione di qualche componente si torna in questo stato.
  - **Wait:** indica tutte le fasi in cui l'utente deve attendere delle operazioni della macchina, per esempio, durante il movimento dei motori o durante la connessione degli altri componenti.
  - **All connected:** indica che tutte le componenti sono connesse correttamente e si può accedere alla fase di test.
  - **Test s1:** indica che si sta eseguendo il test del DUT in posizione 1.
  - **Stato non considerato:** indica tutti gli stati non attualmente considerati, nel caso in cui il software finisca in questo stato significa che è presente un bug, quindi, per evitare malfunzionamenti ulteriori, ci si riconduce allo stato "Initialize" e l'utente deve effettuare tutte le riconnessioni.
- **MCL (Motion Control Loop):** gestisce il movimento e il controllo dei movimenti sul robot cartesiano; controlla inoltre le funzionalità del led che indica lo stato dei motori. La variabile "motor status" può assumere i seguenti valori:
  - **Initialize:** stato iniziale del thread, indica che i motori non sono ancora connessi, oppure che la connessione con i motori è fallita.

  - **Connecting:** tentativo di apertura di una connessione con il driver dei motori.
  - **Homing:** motori in fase di homing.
  - **Connected (pause):** motori connessi correttamente e non in movimento; in questo stato l'homing è stato completato correttamente.
  - **Moving:** motori in movimento verso una posizione specifica.
  - **Failed homing:** homing non correttamente terminato, si è verificato un problema di movimento oppure un interruzione tramite il tasto "stop"; in questo stato la connessione con il driver PMX-4ET-SA è ancora presente.
  - **Failed position:** posizionamento non correttamente terminato, si è verificato un problema di movimento oppure un interruzione tramite il tasto "stop"; in questo stato la connessione con il driver PMX-4ET-SA è ancora presente.

- **BCL (Board Control Loop):** gestisce la comunicazione con LR-TEST-mcu, sia la comunicazione con la stessa, sia la comunicazione con il DUT, utilizzandola come ponte. La variabile "board status" può assumere i seguenti valori:
  - **Initialize:** stato iniziale del thread, indica che LR-TEST-mcu non è ancora connessa, oppure che la connessione è fallita.
  - **Connecting:** indica che LR-TEST-mcu è in fase di connessione e self test.
  - **Connected (pause):** indica che LR-TEST-mcu è connessa e in attesa di comando dal PC.
  - **Write EEPROM:** indica che è in corso la scrittura di nuovi parametri di calibrazione sulla EEPROM della scheda.
  - **Test s1:** indica che LR-TEST-mcu sta eseguendo il test del DUT.

- **CCL (Cloud Communication Loop):**gestisce la comunicazione con il cloud aziendale, chiamando le service necessarie ed elaborando le risposte. La variabile "cloud status" può assumere i seguenti valori:

  - **Initialize:** stato iniziale del thread, indica che il cloud non è ancora connesso, oppure che la connessione ad internet è assente.
  - **Connecting:** indica che il cloud è in fase di connessione e autenticazione dell'operatore.
  - **Connected (pause):** indica che il cloud è connesso e in attesa.
  - **Wait login:** indica che è presente la connessione ad internet, ma che l'operatore non ha effettuato l'accesso al cloud, oppure si è disconnesso.
  - **Failed login:** indica che è presente connessione ad internet, ma che l'operatore ha inserito una chiave sbagliata.

- **PCL (Printer Control Loop):** gestisce la comunicazione con la stampante, inviando la configurazione e le etichette da stampare. La variabile "printer status" può assumere i seguenti valori:
  - **Initialize:** stato iniziale del thread, indica che la stampante non è ancora connessa.
  - **Connecting:** indica che la stampante è in fase di connessione e stampa dell'etichetta di prova.
  - **Connected (pause):** indica che la stamapnte è connessa e in attesa.
  - **Printing:** indica che sono state inviate alla stampante delle etichette.

3.3 Aggiornamenti del codice

Al primo utilizzo su un nuovo PC è necessario installare il programma e tutte le librerie LabView utili, questo avviene tramite apposito installer; per le successive versioni del software invece non è necessario eseguire ulteriori installazioni, basta utilizzare il nuovo file .exe creato in LabView. In Figura3.3 si vede il progetto LabView, nella parte inferiore, sotto la voce build specification, sono presenti le due sezioni per creare il file .exe e l'installer, in queste sezioni è possibile scegliere l'icona da associare al programma, le librerie da aggiungere e eventuali files aggiuntivi.
Gli aggiornamenti del codice possono essere di due tipologie: per implementare nuove funzioni/correzione di bug, oppure per inserire nuove versioni del firmware

compatibili. Non è infatti possibile eseguire test su versioni del firmware non compatibili (vedi Capitolo3.5), viene rilasciata una nuova versione del software per ogni nuova versione del firmware del DUT.

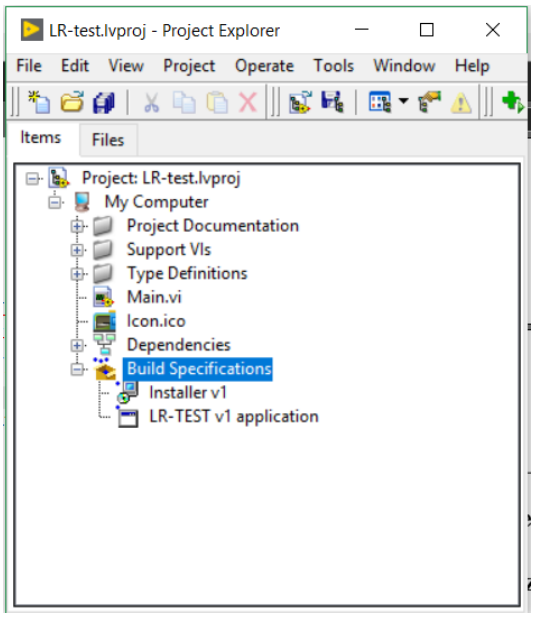

Figura 3.3: File progetto di LabView

3.4 Fase di start-up e configurazione di LR-TEST

Ogni volta che viene avviato il programma, prima di poter effettuare il test, è necessario verificare la connessione di tutte le parti di LR-TEST e di effettuare la connessiona al cloud.
Nello specifico, l'utente, dovrà eseguire le seguenti fasi:

- Connessione al cloud inserendo apposita passkey.
- Connessione di LR-TEST-mcu e dei motori, effettuando l'opportuna calibrazione. Questa operazione viene eseguita premento il tasto "Connetti" presente nell'header dell'interfaccia.
- Connessione della stampante e controllo di funzionamento tramite stampa di prova.

Se non si eseguono queste operazioni, per evitare errori d'uso, non è possibile accedere alle sezioni in cui viene effettuato il test del DUT; come si nota in Figura3.4, i tasti di accesso alle sezioni di test non sono disponibili perchè non sono stati collegati correttamente tutti i componenti. Inoltre, in caso di disconnessione o errori di un componente durante l'utilizzo, il software torna in automatico in questa schermata e blocca tutte le operazioni che stava eseguendo.

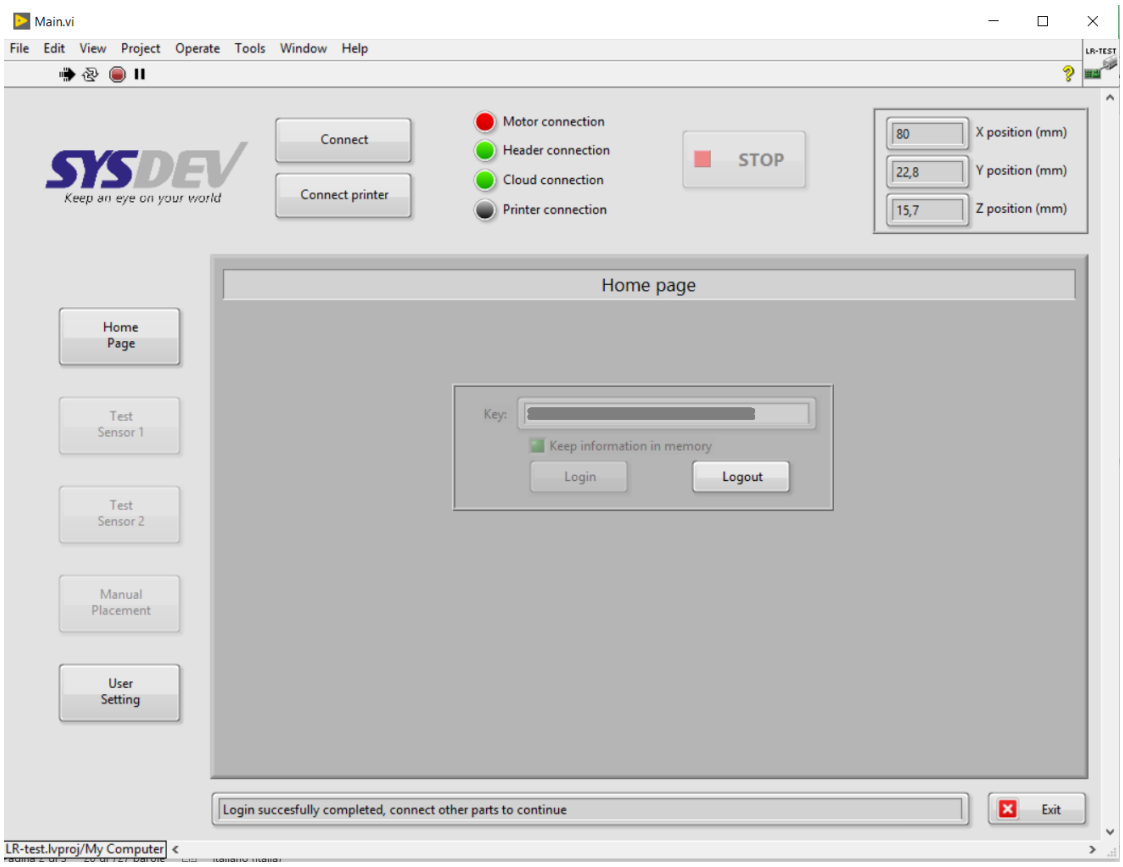

Figura 3.4: Interfaccia utente dopo il login in lingua inglese

In Figura3.5 vengono mostrate tutte le impostazioni a cui ha accesso l'operatore durante tutte le fasi di funzionamento di LR-TEST, nel dettaglio sono presenti:

- **Message log di LR-TEST-mcu:** sono presenti tutti i comandi ricevuti da LR-TEST-mcu, inclusi i comandi ricevuti dal DUT e ritrasmessi da LR-TEST-mcu; non vengono visualizzati i comandi inviati dal PC per non permettere all'utilizzatore di essere a conoscenza del protocollo di comunicazione

utilizzato.

- **Connessione manuale di LR-TEST-mcu:** permette di connettere manualmente, scegliendo la COM, LR-TEST-mcu; viene utilizzato nel caso in cui il riconoscimento automatico della scheda non funzioni correttamente.
- **Coordinate di posizione dei motori:** vengono visualizzate le posizioni in millimetri dei motori durante le fasi di utilizzo, dopo la connessione di LR-TEST-mcu vengono aggiornate con i valori di offset memorizzati nella scheda.
- **Offset di posizione memorizzato in LR-TEST-mcu:** viene visualizzato l'offset memorizzato in LR-TEST-mcu durante la fase di ricalibrazione.
- **Lingua applicazione:** l'utente può scegliere la lingua dell'applicazione, modificandola tra italiano e inglese

3.5 Fase di test del DUT

Nella sezione del programma mostrata in Figura3.6 è possibile eseguire il test e la configurazione automatica dei DUT. Per poter accedere a questa sezione bisogna prima aver connesso correttamente tutte le componenti hardware di LR-TEST.
Nell'interfaccia grafica di questa sezione è possibile visualizzare tutte le informazioni di configurazione durante il test e tutte le fasi di esecuzione del test; è inoltre possibile configurare i parametri da inserire nel sensore prima di iniziare il test, dopo l'avvio non è più possibile eseguire questa operazione. I parametri che si possono impostare sono:

- Fattore GFh
- Fattore KTh
- Modalità ABP

NB: per maggiori dettagli riferirsi al Capitolo2.3

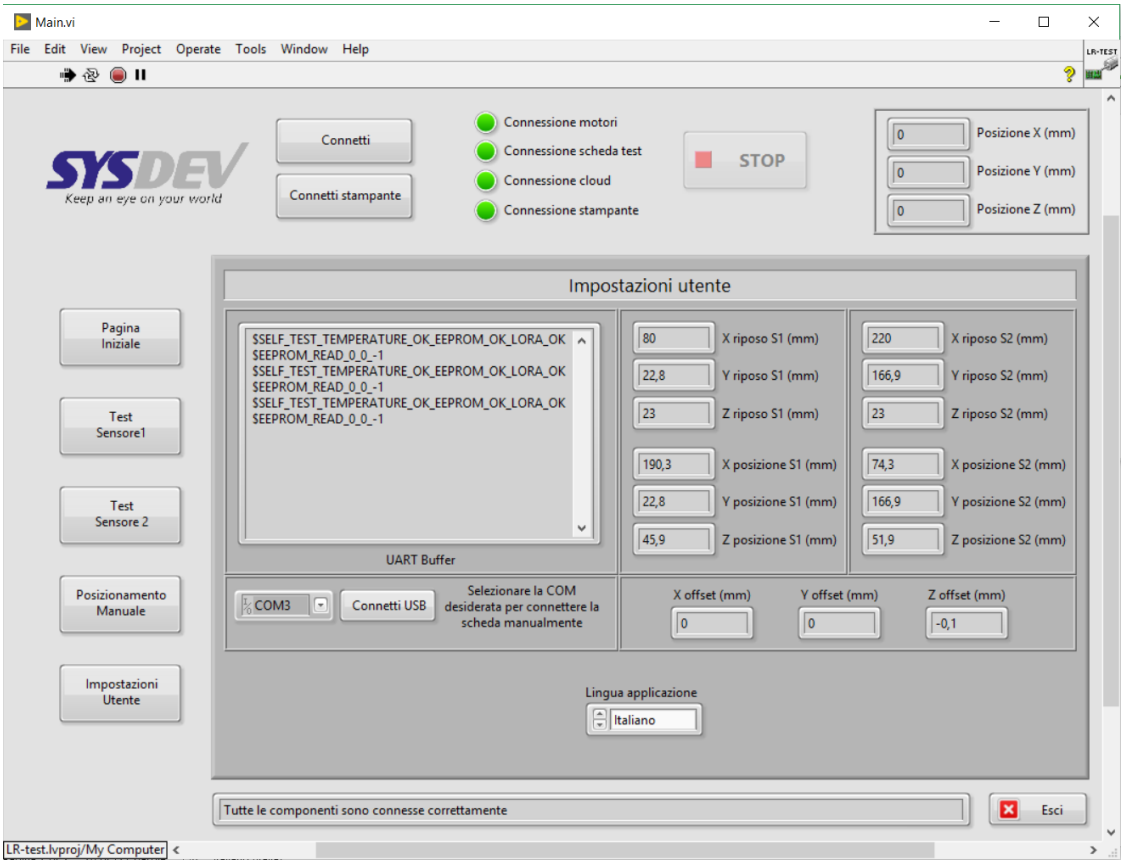

Figura 3.5: Impostazioni accessibili all'utente

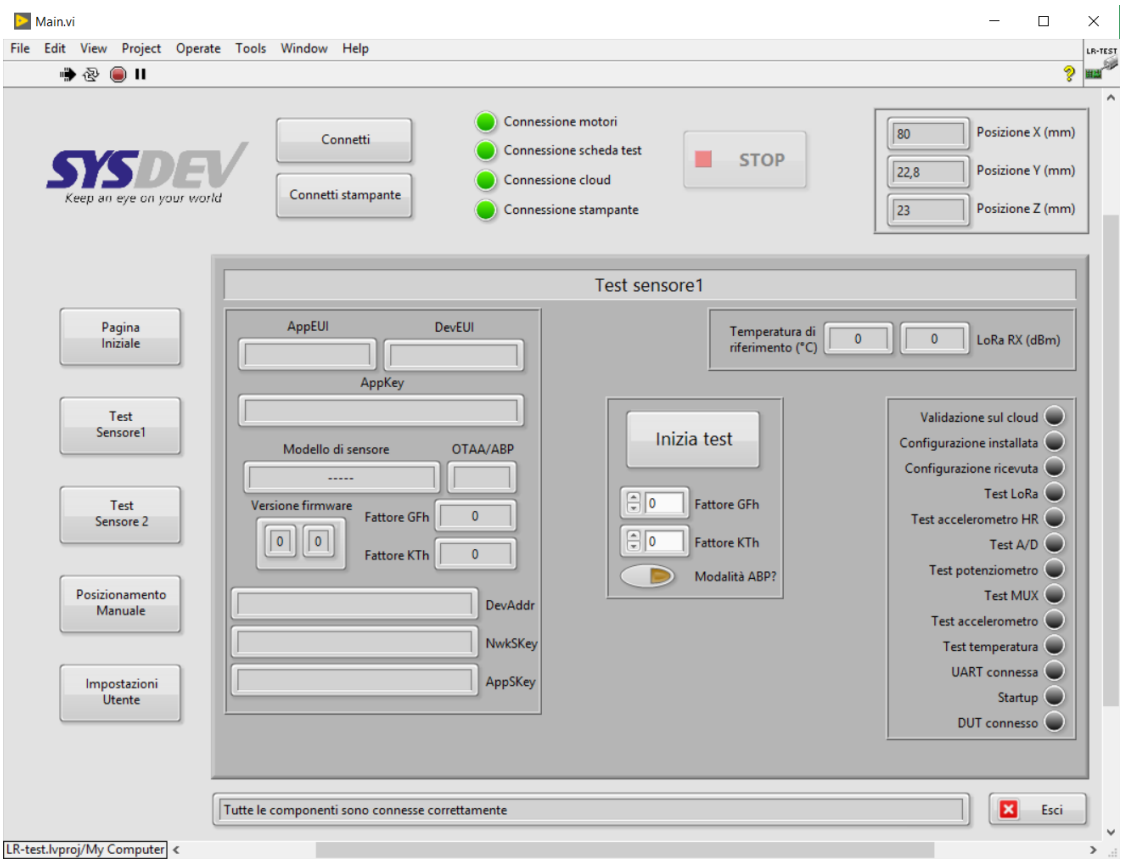

Figura 3.6: Interfaccia utente per effettuare il test sul DUT

Nella parte destra dell'interfaccia vengono mostrate tutte le fasi di esecuzione del test, che si aggiornano in dinamicamente durante l'esecuzione; ogni fase, come mostrato in Figura3.7, viene indicata con un led di diverso colore in base allo stato:

- **Led grigio:** fase in attesa di essere eseguita.
- **Led giallo lampeggiante:** fase in esecuzione.
- **Led verde:** fase eseguita correttamente
- **Led rosso:** fase in cui si è verificato un errore.
- **Led e scritta grigio:** fase che non viene eseguita perchè non necessaria per il DUT sottoposto a test (per esempio, in 3.7, non viene eseguito il test del convertitore AD perchè non è presente in quella tipologia di DUT).

Il test completo viene eseguito elaborando in ordine i seguenti step:

- **DUT connesso:** verifica il corretto allineamento del DUT rispetto ai pin di LR-TEST-mcu; in questa fase il DUT non è alimentato.
- **Startup:** verifica la corrretta accensione del DUT.
- **UART connessa:** invia al DUT la passkey per accedere alla modalità di configurazione, infine verifica la corretta risposta. Durante questa fase vengono anche letti tutti i dati con cui è configurato il sensore, dopo la lettura vengono mostrati all'utente sul display.
- **Test temperatura:** Effettua il test del sensore di temperatura integrato nel DUT; per eseguire il test, LR-TEST-mcu legge la temperatura dal sensore presente sulla scheda e confronta il dato con il valore fornito dal DUT, il test è positivo se il valore si discosta per un massimo di $\pm 5°C$ dal valore misurato da LR-TEST-mcu.
- **Test accelerometro:** Invia al DUT il comando per effettuare il self test dell'accelerometro integrato in esso; in base alla risposta ricevuta verifica la correttezza del test.

- **Test MUX:** Invia al DUT il comando per effettuare il self test del MUX integrato in esso; in base alla risposta ricevuta verifica la correttezza del test. Questo test viene eseguito solo su LR-SS.
- **Test potenziometro:** Invia al DUT il comando per effettuare il self test del potenziometro integrato in esso; in base alla risposta ricevuta verifica la correttezza del test. Questo test viene eseguito solo su LR-SS.
- **Test A/D:** Invia al DUT il comando per effettuare il self test del covertitore A/D integrato in esso; in base alla risposta ricevuta verifica la correttezza del test. Questo test viene eseguito solo su LR-SS-F.
- **Test accelerometro HR:** Invia al DUT il comando per effettuare il self test dell'accelerometro HR integrato in esso; in base alla risposta ricevuta verifica la correttezza del test. Questo test viene eseguito solo su LR-SS-T.
- **Test LoRa:** Invia al DUT il comando per abilitare la trasmissione LoRa continua e verifica la ricezione del segnale tramite LR-TEST-mcu, misurando i dBm ricevuti.
- **Configurazione ricevuta:** Richiede i nuovi dati di configurazione del DUT al cloud. Appena i nuoiv dati sono ricevuti vengono visualizzati sulla schermata i test.
- **Configurazione installata:** Installa i parametri ricevuti dal cloud sul DUT. Al termine dell'installazione invia alla stampante il comando per la stampa delle etichette necessarie.
- **Validazione sul cloud:** Invia al cloud la conferma di avvenuta installazione dei nuovi parametri di configurazione.

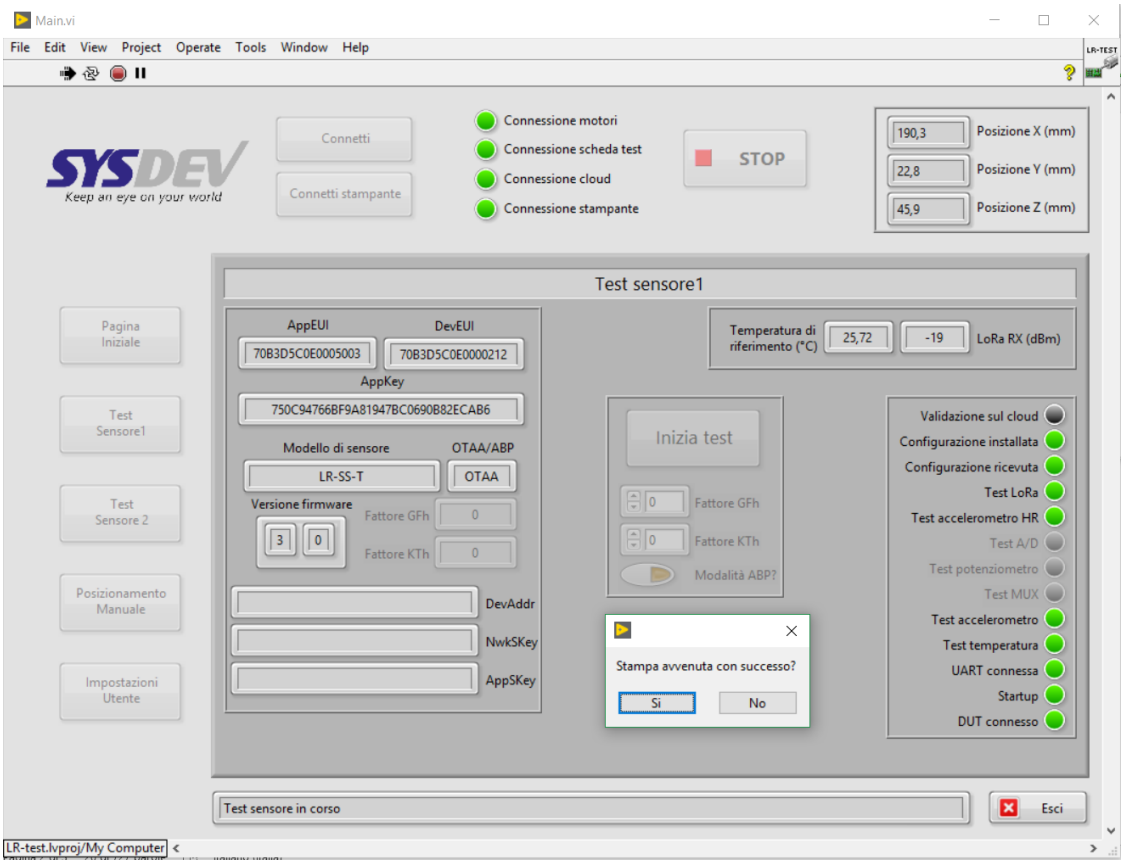

Figura 3.7: Interfaccia utente durante l'esecuzione di un test sul DUT

3.6 Ricalibrazione degli assi

In caso di mancato allineamento tra DUT e LR-TEST-mcu, l'operatore può ricalibrare la posizione del DUT attraverso il movimento manuale dei motori e il salvataggio della nuova posizione in LR-TEST-mcu. Nella EEPROM viene salvato l'offset rispetto alla calibrazione iniziale del DUT in posizione uno; così in caso di aggiunta di sensori in posizioni diverse sul piano della macchina la ricalibrazione deve essere effettuata una sola volta per tutte le posizioni. Il valore di offset è ottenuto secondo la formula:

$$X_{off} = X_{new} - X_0 \qquad Y_{off} = Y_{new} - Y_0 \qquad Z_{off} = Z_{new} - Z_0$$

Alla connessione di LR-TEST-mcu viene richiesta la lettura dei valori di offset salvati nella EEPROM e viene aggiornati tutti i valori di posizione sul PC, secondo la formula:

$$X_{new} = X_0 + X_{off} \qquad Y_{new} = Y_0 + Y_{off} \qquad Z_{new} = Z_0 + Z_{off}$$

Per lo spostamento manuale dei motori, l'utente ha l'interfaccia in Figura 3.8, con cui è possibile muovere manualmente ogni asse aumentando o diminuendo i valori di posizione di $0, 1mm, 1mm, 10mm$; adottando questa scelta si minimizzano gli errori dell'utente durante lo spostamento manuale rispetto a un interfaccia in cui è possibile inserire direttamente le coordinate ed effettuare il movimento su più assi in contemporanea. Il problema che può verificarsi in questo caso è l'abbassamento dell'asse Z prima dello spostamento di X e Y, in questo caso LR-TEST-mcu può urtare contro il DUT e danneggiarsi.

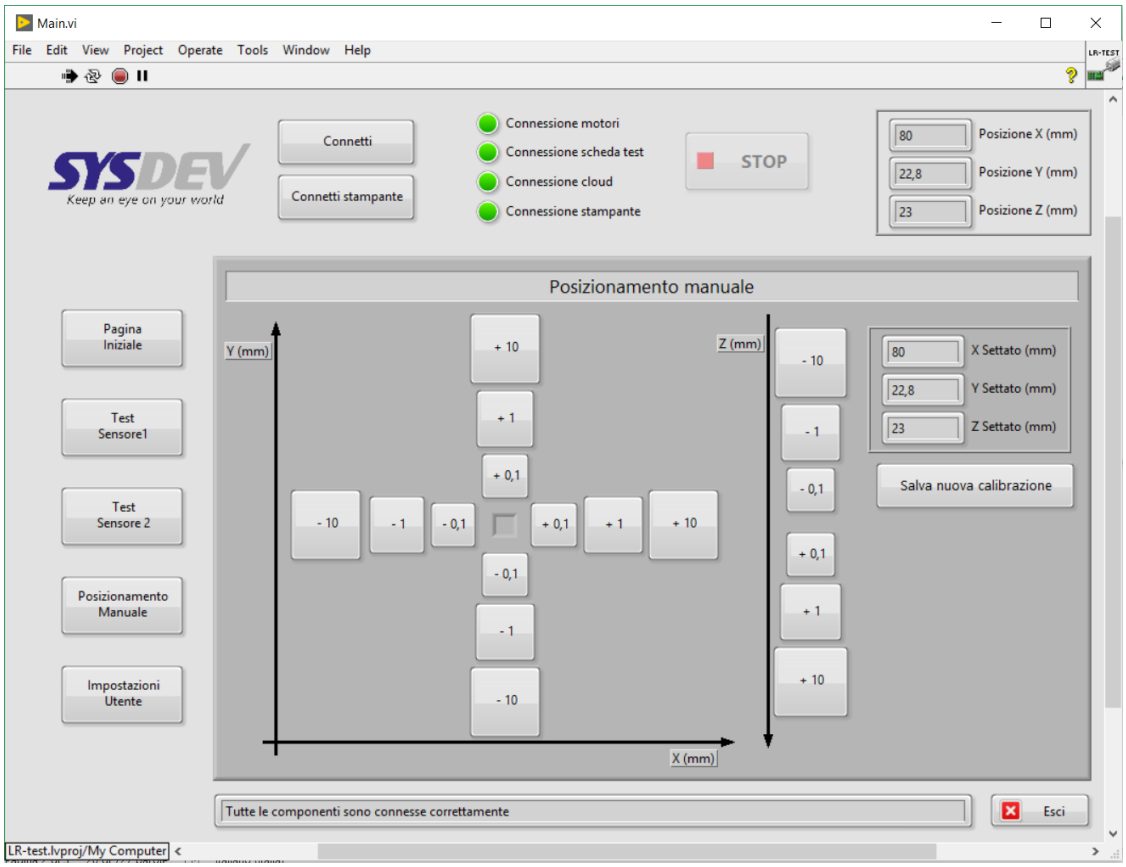

Figura 3.8: Interfaccia utente per il posizionamento manuale dei motori e per la nuova calibrazione

Capitolo 4

Robot cartesiano a tre assi

Il movimento dei motori viene gestito dal driver PMX-4ET-SA, attraverso specifici comandi inviati tramite Ethernet. Il driver viene controllato in LabView attraverso apposita libreria, previa configurazione della rete come in Figura 4.1.
Dopo aver configurato la rete possono essere iniviati i comandi tramite LabView; la corretta procedura di avviamento dei motori viene effettuata in due fasi, prima viene inviata la configurazione iniziale del driver, successivamente viene effettuato l'homing dei motori.

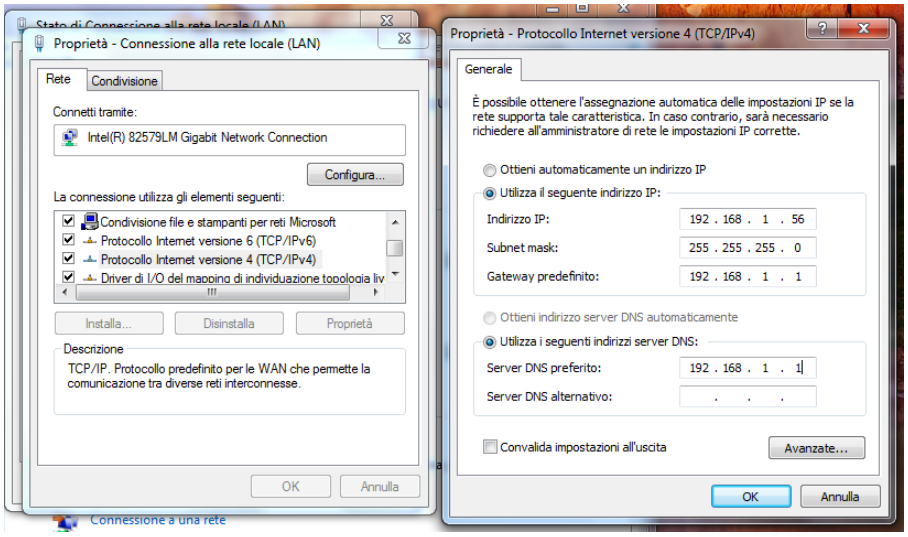

Figura 4.1: Configurazione della connessione Ethernet per l'utilizzo di PMX-4ET-SA

4.1 Inizializzazione del driver

All'accensione della macchina vengono inviati i seguenti comandi per inizializzare il driver, solo se tutti i comandi rispondono con esito positivo si procedere alle fasi

successive di utilizzo. I comandi inviati sono:

- **EO1=1 EO2=1 EO3=1 EO4=1:** viene abilitato l'output per i 4 motori, senza abilitazione ($EO_X = 0$) tutti i successivi comandi di movimento vengono ignorati dal driver.

- **ACC=200 DEC=200:** vengono impostati i i valori di accelerazione e decelerazione globali (validi su ogni asse), i valori sono espressi in ""numero di impulsi".

- **HS=2000 LS=70:** vengono impostati i valori di velocità massima e minima globali, i valori sono espressi in "numero di impulsi al secondo"

- **ABS:** con questo comando viene impostata la gestione del movimento in modalità assoluta, cioè per il movimento di un motore viene inviata una posizione riferita allo zero effettuato durante l'homing; è altrimenti possibile utilizzare la modalità incrementale, in cui la posizione inviata viene sommata alla posizione in cui si trova attualmente il motore.

- **POZ=15:** Viene settato l'asse Z per avere i riferimenti di posizione inversi, l'asse Z ha riferimento positivo verso il basso per essere più intuitivo

In Figura 4.2 viene mostrato il codice di inizializzazione del driver, con l'apertura della connessione e l'invio del settaggio iniziale.

4.2 Gestione del movimento

Per gestire il movimento dei tre assi, è scelto di muovere un solo asse alla volta (vedi Capitolo 4.3.1). Questa scelta è stata effettuata per due ragioni, l'asse Z deve muoversi quando gli assi X e Y sono fermi per evitare la rottura dei puntali di LR-TEST-mcu; siccome Z alza e abbassa i puntali, in caso di movimento di X e Y prima che essi siano stati completamente alzati/abbassati, LR-TEST-mcu può urtare contro il DUT e danneggiarsi (i motori hanno coppia elevata e rompono le saldature dei pin di LR-TEST-mcu, Figura 2.5).

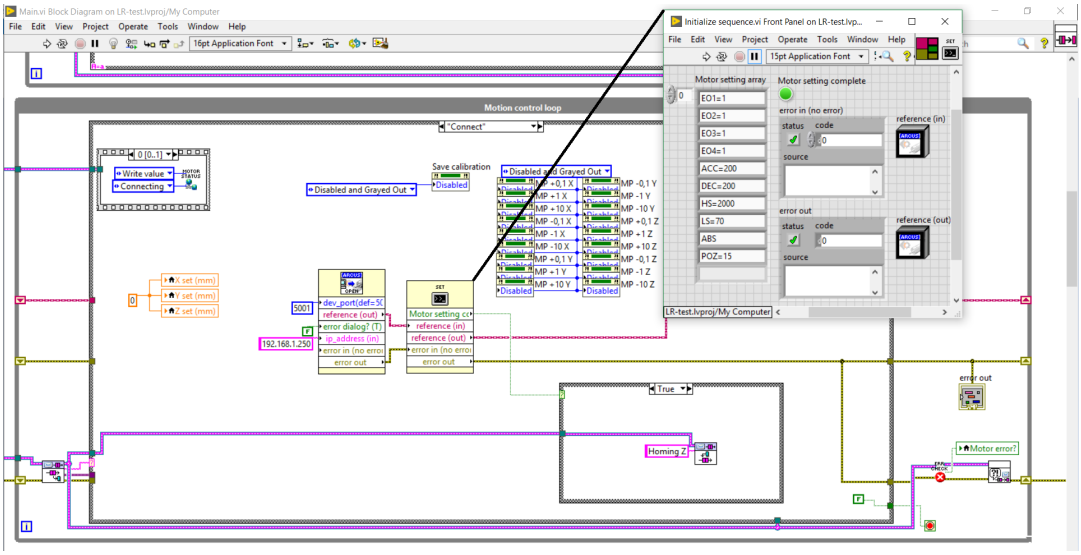

Figura 4.2: Codice con apertura della connessione e settaggio iniziale di PMX-4ET-SA

Gli assi X e Y, teoricamente, potrebbero muoversi contemporaneamente per velocizzare il posizionamento della testa, ma a causa di un bug nel driver di PMX-4ET-SA non è possibile effettuare questa operazione.
Nello specifico, ogni asse, a causa del diverso passo delle guide, deve muoversi a velocità differente; settando la velocità globale (attraverso il comando HS=xxx LS=yyy) tutti gli assi si muovono alla stessa velocità, per settare la velocità su un singolo asse bisogna utilizzare il comando HSX=xxx LSX=yyy, HSY=zzz LSY=www, etc.. ma questo non viene correttamente gestito dal driver, cioè il comando è accettato ma i motori si muovono con velocità globale. Per ovviare a questo problema, prima del movimento di ogni asse, viene reimpostata la velocità globale corrispondente all'asse che deve muoversi; il bug è stato segnalato al produttore che ha evidenziato il problema e si attende l'uscita di un nuovo firmware; a quel punto sarà possibile muovere contemporaneamente assi X e Y.

4.2.1 Homing dei motori

Per effettuare l'homing dei motori, per ogni asse, vengono settate le velocità relative all'homing e successivamente viene inviato il comando di homing per il relativo asse; come per ogni movimento bisogna attendere lo stop di un asse prima di iniziare l'homing del successivo.
Di seguito viene riportata la sequenza di comandi inviati per ogni asse con le relative velocità:

- **Asse Z:**
  HS=9000
  LS=1500
  ACC=400
  DEC=200
  HZ-4

- **AsseYU:**
  HS=9000
  LS=1500
  ACC=400

DEC=200
HY-4
HU-4

- **Asse X:**
  HS=3000
  LS=100
  ACC=100
  DEC=100
  HX-4

I valori di velocità massima/minima e accelerazione/decelerazione sono stati ricavati effettuando delle prove e trovando il giusto compromesso tra velocità e sicurezza per l'operatore; in caso di velocità dei motori troppo elevata, l'operatore non ha tempo di reagire per bloccare la macchina al bisogno (per esempio oggetti che ostacolano il movimento di un asse).
Il comando H$_{axis}$-4 indica l'avvio dell'homing in direzione negativa in modalità 4, cioè come in Figura 4.3, ricavata dal manuale del componente. Delle modalità disponibili questa è quella che minimizza gli errori dovuti alla decelerazione dei motori consentendo un movimento iniziale a velocità non troppo basse.

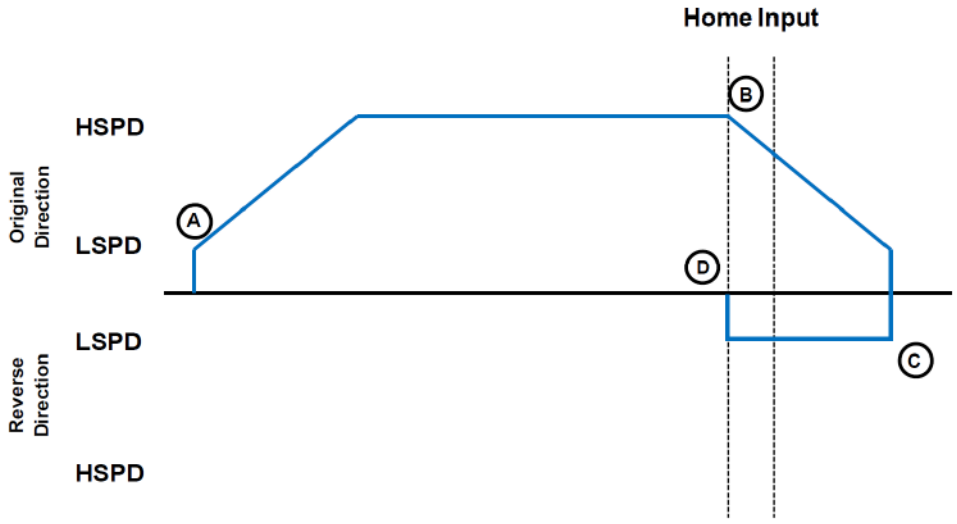

Figura 4.3: Grafico sulla gestione dell'homing ricavato dal manuale di PMX-4ET-SA

4.2.2 Movimento in posizione specifica

Per effettuare il movimento dei motori ad una certa coordinata, per ogni asse, vengono settate le velocità relative all'asse e successivamente viene inviato il comando di posizione; come per ogni movimento bisogna attendere lo stop di un asse prima di iniziare quello del successivo.
Di seguito viene riportata la sequenza di comandi inviati per ogni asse con le relative velocità:

- **Asse Z:**
  HS=18500
  LS=1500
  ACC=400
  DEC=200
  Z*zzz*
- **AsseYU:**
  HS=14600
  LS=1500
  ACC=400
  DEC=200
  Y*yyy*U*yyy*
- **Asse X:**
  HS=4500
  LS=100
  ACC=100
  DEC=100
  X*xxx*

I valori di velocità massima/minima e accelerazione/decelerazione sono stati ricavati effettuando delle prove e trovando il giusto compromesso tra velocità e sicurezza per l'operatore; in caso di velocità dei motori troppo elevata, l'operatore non ha tempo di reagire per bloccare la macchina al bisogno (per esempio oggetti che ostacolano il movimento di un asse).
I comandi X*xxx*, Y*yyy*U*yyy*, Z*zzz* servono per far muovere in una posizione specifica

il relativo asse, nel caso di YU i due assi si muovono in contemporanea. I valori di posizione devono essere numeri interi e indicano il numero di impulsi da inviare al motore, per semplificare la visualizzazione da parte dell'utente le posizioni vengono mostrate in millimetri, quindi viene effettuata una conversione secondo le seguenti formule:

$$X_{pulse} = 57,1 * X_{mm} \quad | \quad Y_{pulse} = 1333,4 * Y_{mm} \quad | \quad Z_{pulse} = 2668,5 * Z_{mm}$$

Nel Capitolo 4.4 viene spiegeto nel dettaglio come sono stati ottenuti i valori di conversione.

4.3 Strutture software di controllo

In questa sezione vengono analizzati tutti i controlli software sviluppati per garantire il corretto funzionamento della macchina. Il driver PMX-4ET-SA gestisce in autonomia il movimento dei motori ma senza effettuare alcuni controlli, per questo vengono aggiunti i seguenti check software:

- Identificazione del movimento e della posizione di ogni asse, è necessario riconoscere se ogni asse è in movimento oppure fermo, inoltre bisogna avere la posizione di ogni asse anche mentre si stanno muovendo (per permettere all'utente di avere sempre la posizione attuale del robot cartesiano).

- Controllo di corrispondenza tra numero di impulsi e posizione dell'encoder; serve perchè in caso di motore scollegato o non funzionante viene evidenziato il problema e interrotto il movimento.

- Controllo di allineamento tra asse Y e asse U; in caso di disallineamento vengono bloccati i motori per evitare la rottura delle guide.

- Controllo di movimento di un singolo asse alla volta; come dettagliato nel Capitolo 4.2 non è possibile effettuare il movimento in contemporanea di più assi

4.3.1 Macchina a stati per il controllo del movimento

Al fine di garantire il corretto posizionamento, ogni volta che viene avviato un movimento (anche quando viene avviato l'homing), il software avvia una struttura di controllo come mostrato in Figura 4.4. La sezione "control position" viene analizzata nel Capitolo 4.3.2.
Di seguito vengono analizzati gli stati di uscita del diagramma di flusso:

- **Motor error:** in caso di errore di connessione o da parte dei motori si entra in questo stato interrompendo il flusso delle operazioni; qui viene inviato il comando "STOP" che interrompe tutti i movimenti in caso ci siano, dopo chiude la connessione e porta la macchina nello stato di inizializzazione (quindi è necessario effettuare nuovamente l'homing prima di riutilizzare LR-TEST).

- **Connected (pause):** se tutti i movimenti sono stati eseguiti correttamente, il software chiude la connessione con il driver, dopo questa operazione controlla ciclicamente (ogni 5 secondi) che il driver sia connesso correttamente (aprendo e chiudendo la connessione). Il controllo è necessario per evitare che lo spegnimento e la riaccensione di LR-TEST porti a movimenti in posizioni errate; se, per esempio, i motori sono in posizione 10-20-30 e la macchina viene spenta, al momento della riaccensione il driver vedrà i motori in posizione 0-0-0 anche se non è stato effettuato l'homing; quindi inviando il comando per posizionarli in 10-20-30 i motori andranno in realtà in 20-40-60.

Infine, l'utente, può interrompere il movimento dei motori via software tramite apposito tasto di stop presente sull'interfaccia grafica; in questo caso vengono interrotti tutti i movimenti e la macchina torna nello stato di inizializzazione (è necessario rieffettuare la connessione e l'homing).

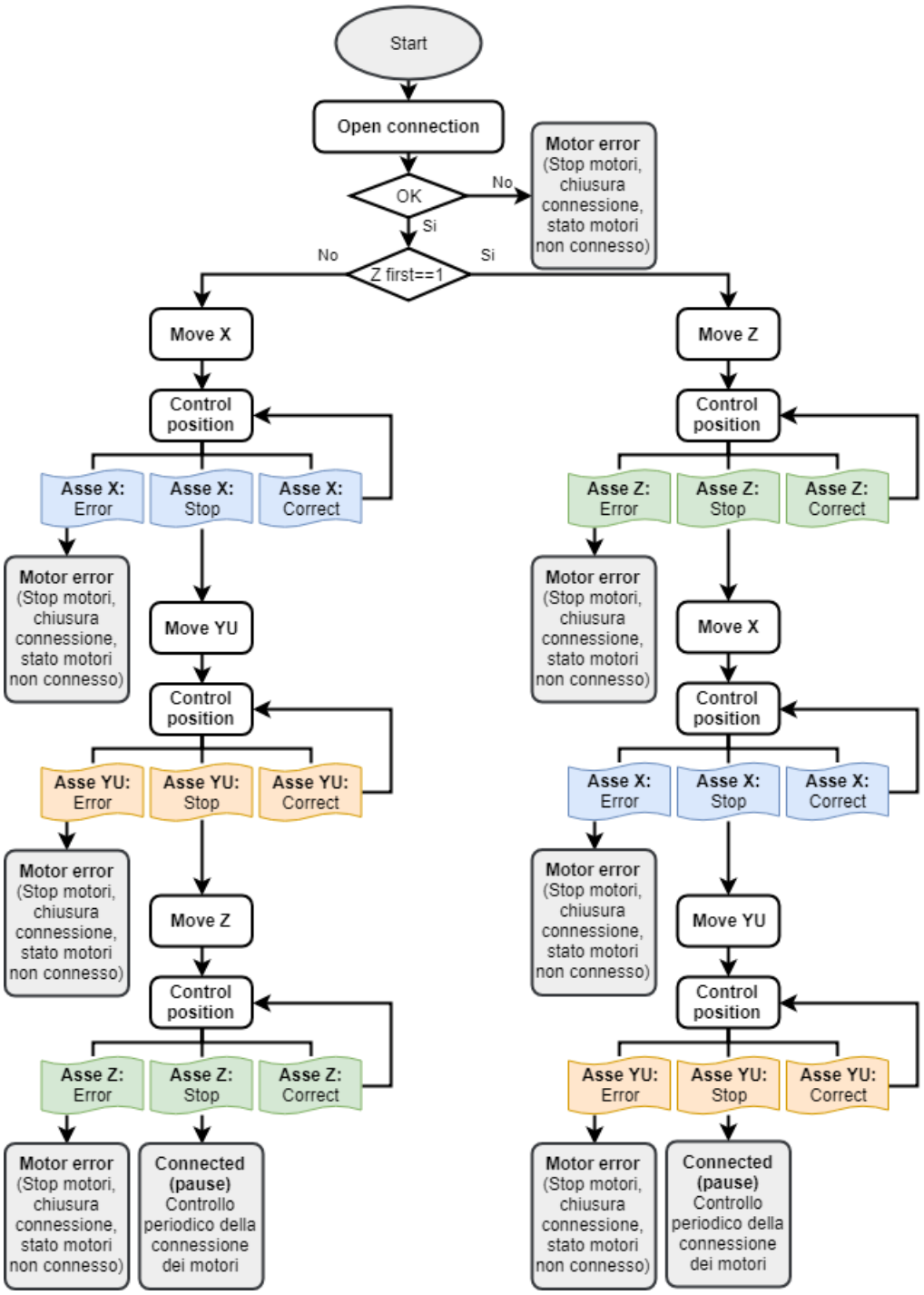

Figura 4.4: Diagramma di flusso per la gestione del movimento dei motori

4.3.2 Monitoraggio della posizione durante il movimento

Per verificare lo stato e la posizione di ogni motore in un singolo istante vengono inviati al driver i seguenti comandi:

- MST: restituisce lo stato dei motori su 12 bit, nello specifico vengono utilizzati i bit 0 (ACC), 1 (DEC), 2 (MOV), 6 (HOME).
- PP: restituisce la "pulse position" dei quattro assi.
- PE: restituisce la "encoder position" dei quattro assi.

In Figura 4.5 viene mostrato come si stabilisce lo stato di ogni asse, che può essere:

- **Stop:** l'asse ha raggiunto la posizione correttamente ed è fermo.
- **Error:** è presente un errore di allineamento tra pulse ed encoder position, oppure un errore di allineamento tra asse Y e U.
- **Correct:** l'asse si sta muovendo correttamente.

Oltre allo stato dei motori viene restituita la posizione in millimetri di ogni asse, vedi conversioni in Capitolo 4.4. Dopo aver eseguito tutte le operazioni viene inserito un ritardo di 100ms per evitare di riempire il buffer della porta ethernet in caso di richieste ripetute sullo stato dei motori (operazione che si verifica durante il movimento).

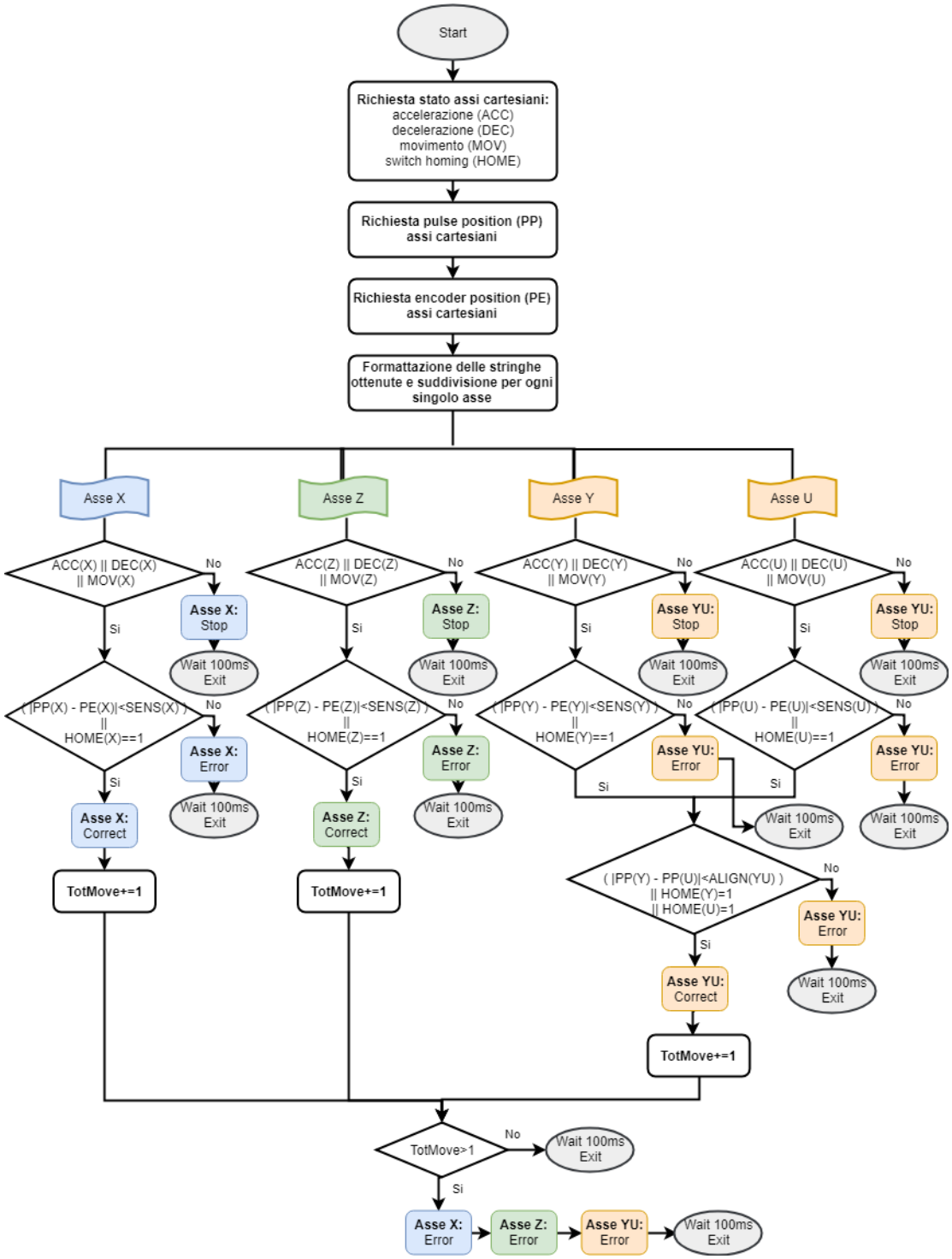

Figura 4.5: Diagramma di flusso per il monitoraggio del corretto movimento dei motori durante il movimento

4.4 Calibrazione iniziale degli assi

I comandi di movimento di ogni asse vengono inviati al PMX-4ET-SA (vedi Capitolo 4.2) come numero di step, quindi è necessario effettuare una conversione per permettere all'utente di visualizzare la posizione di ogni asse in millimetri.
Siccome le guide sono lineari, il fattore di conversione è stato ricavato effettuando misure in posizioni differenti per ogni asse e interpolando i valori, al fine di trovare il coefficiente di conversione di ogni asse. Le misure sono state effettuate con un calibro analogico ($Portata = 280mm,\ Sensibilita' = 0,05mm$).
In Tabella 4.1 sono riportate le misure effettuate, in Figura 4.6, Figura 4.7 e Figura 4.8, sono riportati i grafici interpolanti le misure effettuate per ogni asse; per ottenere la conversione bisogna quindi:

$$X_{mm} = \frac{X_{pulse}}{57,1} \quad | \quad Y_{mm} = \frac{1333,4}{Y_{pulse}} \quad | \quad Z_{mm} = \frac{2668,5}{Z_{pulse}}$$

Asse X		Asse Y/U		Asse Z	
Position(mm)	Step	Position(mm)	Step	Position(mm)	Step
64,80	0	19,22	0	20,92	0
82,46	1000	37,94	25000	24,66	10000
99,80	2000	56,80	50000	28,32	20000
117,46	3000	75,60	75000	32,48	30000
135,10	4000	94,34	100000	36,10	40000
152,50	5000	112,92	125000	39,68	50000
169,90	6000	131,90	150000	43,66	60000
187,54	7000	150,52	175000	47,16	70000
205,00	8000	169,26	200000	50,94	80000
222,50	9000	188,06	225000	54,68	90000
239,98	10000	206,78	250000	58,38	100000
257,56	11000	225,40	275000	62,04	110000
275,00	12000	244,28	300000	66,10	120000

Tabella 4.1: Misure per la calibrazione degli assi

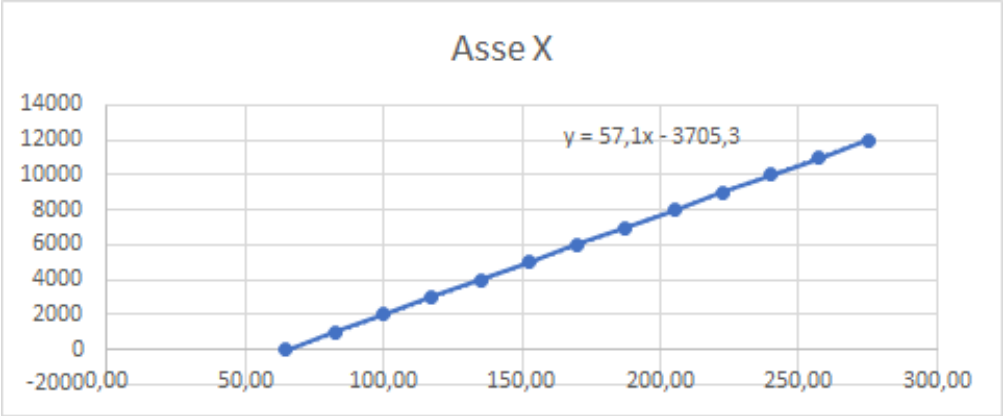

Figura 4.6: Grafico di calibrazione dell'asse X

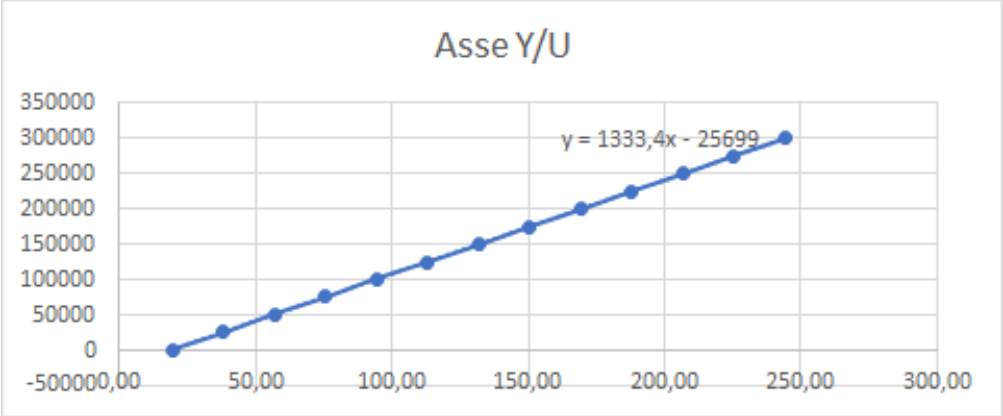

Figura 4.7: Grafico di calibrazione dell'asse Y/U

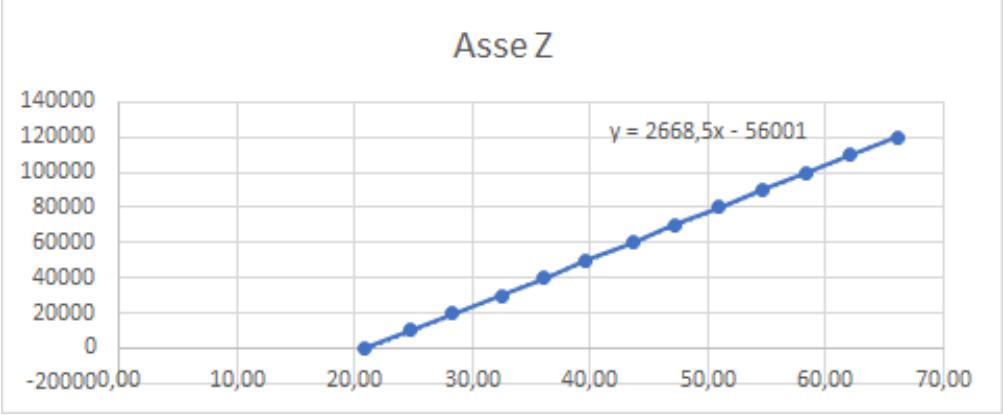

Figura 4.8: Grafico di calibrazione dell'asse Z

Capitolo 5

LR-TEST-mcu

LR-TEST-mcu viene utilizzata per gestire la comunicazione tra DUT e PC, per effettuare alcuni test e per memorizzare i dati di configurazione di LR-SS. La struttura di funzionamento di LR-SS-mcu è mostrata in Figura 5.1.
Ogni comunicazione con LR-TEST-mcu viene eseguita con un comando e si attende una risposta, sia con il DUT sia con il PC. L'UART comunica a 37400 BaudRate, sia il DUT, sia il PC devono trasmettere/ricevere allo stesso BaudRate.

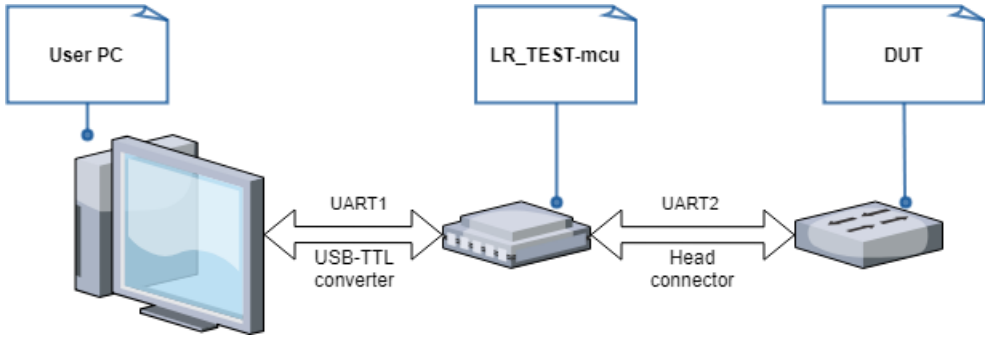

Figura 5.1: Struttura di collegamento tra PC, mcu e DUT

5.1 Comunicazione con il PC

La comunicazione con il PC avviene in varie fasi di utilizzo della macchina, LR-TEST-mcu all'accensione si predispone per ricevere comandi dal PC; in tutte le fasi in cui non vengono richieste azioni attende un comando da PC (non può ricevere/inviare comandi al DUT e non può inviare comandi al PC).
il PC invia a LR-TEST-mcu due tipologie di comandi, quelli che iniziano con il carattere "$" sono comandi che elabora LR-TEST-mcu, mentre gli altri vengono ritrasmessi al DUT.

Vengono inseriti timeout di comunicazione in alcune fasi per evitare blocchi di comunicazione; nello specifico:

- All'avvio inizializzazione con:
  - UART1: ricezione (senza timeout)
  - UART2: spenta
  Si rimane in questo stato finchè LR-TEST-mcu non riceve un comando da PC.

- Ricezione di un comando con "$":
  - UART1: trasmissione
  - UART2: spenta
  LR-TEST-mcu risponde anche in caso di comando non compreso o non correttamente eseguito.

- Ricezione di un comando senza "$":
  - UART1: ricezione
  - UART2: trasmissione
  LR-TEST-mcu dopo aver inviato il comando al DUT: - UART1: trasmissione
  - UART2: ricezione (con timeout)
  Nel caso in cui il DUT non risponda, LR-TEST-mcu invia al PC il comando "- TIMEOUT UART2" (si distingue una mancata connesione del DUT da una mancata connessione di LR-TEST-mcu). In questa situazione il timeout di LR-TEST-mcu deve essere minore del timeout impostato sul PC per la UART1.

5.1.1 Comandi di elaborazione di LR-TEST-mcu

I comandi che LR-TEST-mcu elabora sono i seguenti:

- **$SELF_TEST:** Verifica il corretto funzionamento della scheda; testando il funzionamento del sensore di temperatura, della EEPROM e del transceiver LoRa. Per ogni test risponde "_OK" se il componente funziona correttamente, oppure "_FAIL" in caso di errore.
  Dopo aver effettuato il test, LR-TEST-mcu risponde, per esempio:
  "$SELF_TEST_TEMPERATURE_OK_EEPROM_OK_LORA_FAIL"

- **$RF_RECEIVE:** Restituisce il valore in dBm dell'antenna LORA in ricezione.
  Legge il valore del transceiver LORA; usando la funzione: SX1272ReadRssi (modem), con modem=MODEM_FSK. LR-TEST-mcu risponde con il valore in dBm e lo stato transceiver LORA (in caso di ricezione vale 1), per esempio: "$RF_RECEIVE_-30_dBm_1"

- **$TEMPERATURE:** Restituisce il valore di temperatura fornito dal sensore integrato nella scheda. LR-TEST-mcu risponde con il valore i temperatura, oppure _FAIL in caso di errore, per esempio:
  "$TEMPERATURE_23,45_∘C"

- **$EEPROM_READ:** Legge i valori di offset di posizione scritti all'interno della EEPROM (Capitolo 3.6).
  I valori sono inviati al PC come numeri interi con segno al decimo di millimetro (es: 34 → 3,4mm); LR-TEST-mcu risponde con i valori X Y Z, in caso di errore risponde con FFFF, per esempio:
  $EEPROM_READ_2_-10_5 (corrispondenti a X=0,2 mm; Y=-1 mm; Z=0,5 mm)

- **$EEPROM_WRITE**_*X*_*Y*_*Z***:** Scrive i valori di offset di posizione all'interno della EEPROM (Capitolo 3.6).
  I valori sono inviati a LR-TEST-mcu come numeri interi con segno al decimo di millimetro (es: 34 → 3,4mm); LR-TEST-mcu risponde con i valori X Y Z, in caso di errore risponde con FFFF, per esempio:
  $EEPROM_WRITE_2_-10_5 (corrispondenti a X=0,2 mm; Y=-1 mm; Z=0,5 mm)

- **$UART_OPEN:** Apre la comunicazione di UART2 in ricezione, successivamente accende il DUT; LR-TEST-mcu rimane in ricezione su UART2 finchè non riceve il carattere "$"; alla ricezione del comando $UART_OPEN viene settato un timeout di 5 secondi, superato questo tempo viene interrotta la ricezione su UART2 e inviato su UART1 il comando "$UART_TIMEOUT". Dopo aver chiuso la comunicazione con UART2 (sia in caso di timeout sia in caso di carattere trovato) tutti i caratteri letti da UART2 vengono ritrasmessi

su UART1, così da poter sempre verificare i messaggi inviati dal DUT.
In questa fase il led integrato nella scheda rimane spento.

- **$POSITION:** Verifica se il DUT è correttamente posizionato rispetto alla head di LR-TEST-mcu prima di effettuare l'accensione. Questa verifica serve per evitare disallineamenti oppure posizionamento ruotato di 180∘ del DUT. In caso di corretto posizionamento risponde "$POSITION_CORRECT" altriemnti "$POSITION_WRONG".

- **$DUT_ON:** Fornisce l'alimentazione al DUT tramite chiusura dei relè predisposti; successivamente risponde con la stringa "$DUT_ON".

- **$DUT_OFF:** Spegne l'alimentazione del DUT tramite apertura dei relè predisposti; successivamente risponde con la stringa "$DUT_OFF".

- **$REBOOT:** Effettua il riavvio di LR-TEST-mcu; prima di riavviarsi risponde con la stringa "$REBOOT_CONFIRMED".

In caso di comando $ non riconosciuto, LR-TEST-mcu risponde con la stringa "$- COMMAND NOT FOUND".

5.2 Comunicazione con il DUT

La comunicazione con il DUT viene effettuata da UART2, come mostrato in Figura 5.1, la connessione fisica avviene tramite puntali a molla (Figura 2.5). Sono presenti cinque puntali, di cui quattro di connessione (Vcc, GND, TX, RX) e uno per la verifica di corretto posizionamento; questo puntale serve per verificare la posizione quando il DUT non è ancora alimentato, quindi evita di fornire alimentazione qunado il DUT non è posizionato correttamente; questo controllo può essere effettuato da LR-TEST-mcu tramite il comando "$POSITION".
LR-TEST-mcu ,se riceve comandi non preceduti dal carattere "$", funziona come ponte e ritrasmette la stringa ricevuta al DUT, dopo la trasmissione della stringa abilita UART2 in ricezione e fa partire un timer per il timeout; se riceve una risposta dal DUT questa viene ritrasmessa su UART1, se invece scatta il timeout, interrompe la ricezione su UART2 e invia su UART1 la stringa "$UART_TIMEOUT"; con

questa procedura tutte le stringhe inviate dal DUT arrivano fino al PC e vengono rese disponibili all'utente.

5.3 Struttura del firmware

Il firmware è sviluppato tramite Keil μVision in linguaggio C, utilizza la struttura creata attraverso CubeMX per la configurazione delle porte di comunicazione del microcontroller ARM integrato in LR-TEST-mcu.
Nel firmware vengono integrate tutte le librerie utili a gestire le componenti hardware integrate, nello specifico, la trasmissione seriale, il sensore di temperatura, il transceiver LoRa, le porte IO (bottone e led) e i timer.
La programmazione è stata effettuata tramite una struttura event based, quindi il sistema rimane in attesa degli eventi che arrivano dalle periferiche o dagli interrupt interni.
Questa struttura è stata implementata nella comunicazione seriale acquisendo un carattere per volta in modalità interrupt (usando la funzione HAL_UART_Receive_IT); cioè la funzione attende la lettura del buffer di ricezione finchè non viene scatenato l'interrupt corrispondente, con questo metodo si riesce a garantire la corretta velocità di comunicazione. L'interruzione della ricezione su UART1 avviene quando sono stati letti un numero prefissato di caratteri (è importande che il PC invii sempre il corretto numero di caratteri); mentre UART2 termina la ricezione quando viene inviato dal DUT il comando di fine linea (in questo caso è possibile inviare stringhe di lunghezza variabile). La ritrasmissione su UART1 viene effettuata dopo aver completato la ricezione, perchè il microcontrollore non riesce a gestire ricezione e trasmissione su due UART differenti in contemporanea. Quando viene avviata la ricezione su UART2 si abilita un timer (che viene usato come timeout) che genera un interrupt in caso di mancata ricezione e interrompe la lettura per evitare che LR-TEST-mcu si blocchi in uno stato in cui il PC non può comunicare con essa.
In Figura 5.2 viene mostrato nel dettaglio come viene gestita la comunicazione sulle due UART in LR-TEST-mcu.

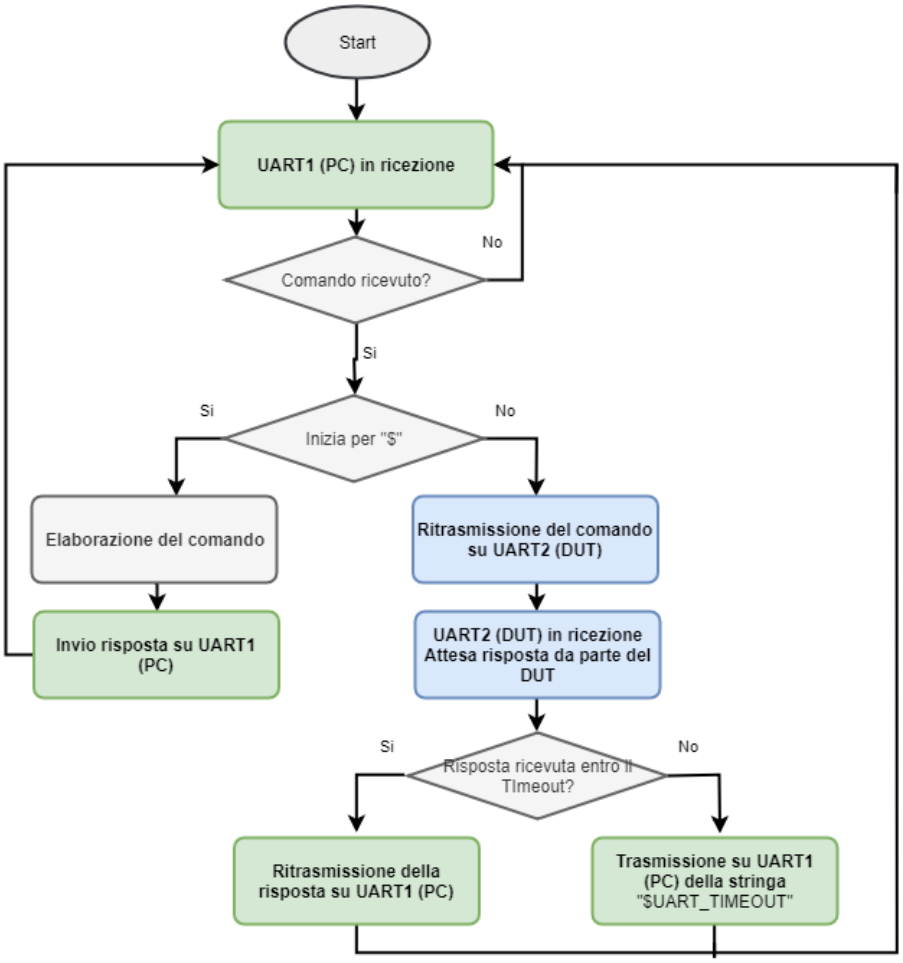

Figura 5.2: Diagramma di flusso del firmware di LR-TEST-mcu

Capitolo 6

Cloud aziendale

La comunicazione con il cloud avviene tramite service, che vengono chiamate al bisogno; lo scambio dei dati viene inserito nella chiamata e avviene in formato JSON come dettagliato nei capitoli seguenti.

6.1 Connessione al Cloud

Ogni volta che viene chiamata una service, oltre alla chiamata all'indirizzo corretto è necessario autenticarsi tramite chiave, la comunicazione avviene tramite protocollo HTTPS. La SubVi in cui vengono chiamate le service, restituisce lo stato della comunicazione come segue:

- **OK:** la comunicazione è avvenuta correttamente.
- **Fail connection:** è fallita la comunicazione con il server, per mancanza di connessione ad internet o per indisponibilità del server.
- **Fail autentication:** è fallita la comunicazione con il server perchè non è stata riconosciuta correttamente la chiave fornita dall'utente, la connessione a internet e la comunicazione con il server sono correttamente funzionanti.

6.2 Service di richiesta dati

La service viene utilizzata per ricevere la configurazione da installare sul DUT, tutti i parametri vengono forniti dal cloud che effettua la gestione e la memorizzazione dei parametri di tutti i DUT. Per chiamare correttamente la service bisogna inviare, in formato JSON, i seguenti parametri:

- Username

- AppEUI
- DevEUI
- OTAA/ABP

Se tutti i parametri sono corretti, la service risponde, con i dati in formato JSON:

- AppEUI
- DevEUI
- AppKey
- DevAddr
- NwkSKey
- AppSKey

NB: per maggiori dettagli sui parametri consultare il Capitolo 2.3

6.3 Service di validazione del test

La service viene utilizzata per confermare la configurazione installata sul DUT, tutti i parametri vengono forniti al cloud che effettua la gestione e la memorizzazione dei parametri di tutti i DUT. Per chiamare correttamente la service bisogna inviare, in formato JSON, i seguenti parametri:

- Username
- AppEUI
- DevEUI
- AppKey
- TimeStamp
- OTAA/ABP

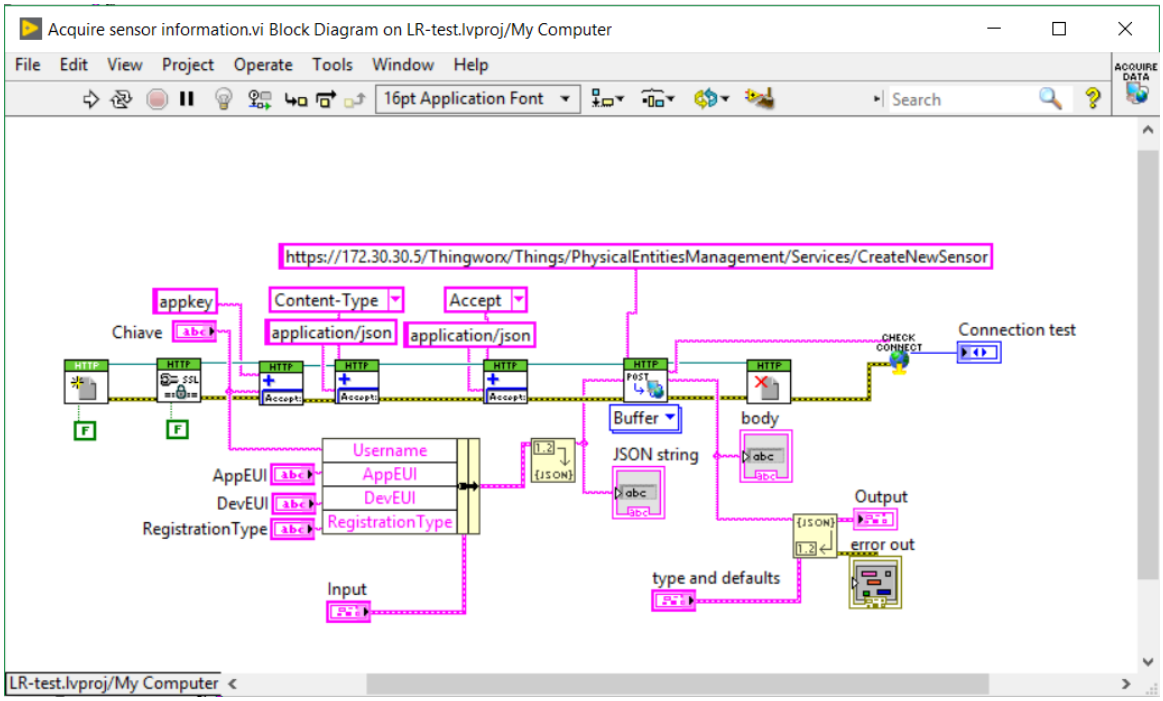

Figura 6.1: SubVi che gestisce la chiamata della service di richiesta dati

- DevAddr
- NwkSKey
- AppSKey

Se tutti i parametri sono corretti, la service risponde, con i dati in formato JSON:

- OK/FAIL
- Message

NB: per maggiori dettagli sui parametri consultare il Capitolo 2.3

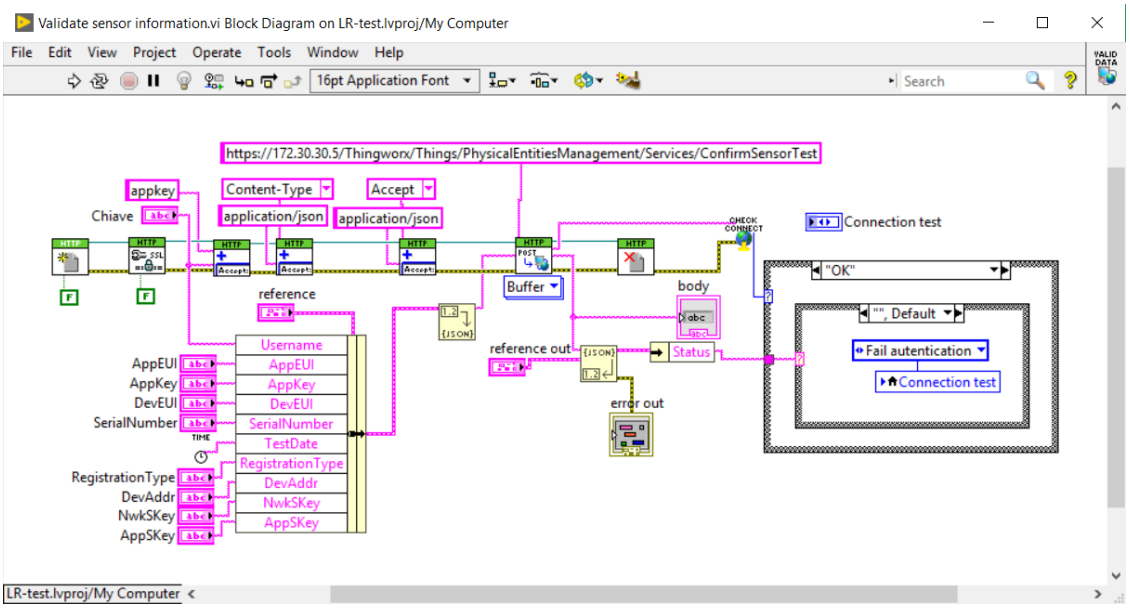

Figura 6.2: SubVi che gestisce la chiamata della service di validazione dati

6.4 Service di log

La service viene utilizzata per effettuare quattro operazioni differenti, tutti i parametri vengono forniti al cloud che effettua la gestione e la memorizzazione dei log. Per chiamare correttamente la service bisogna inviare, in formato JSON, i seguenti parametri in base alla tipologia di log richiesta:

- **Connessione al cloud:**
  - Timestamp
  - Username
  - CONNECTION_TEST
- **Ricalibrazione motori:**
  - Timestamp
  - Username
  - RECALIBRATION
  - Xoffset (mm)
  - Yoffset (mm)
  - Zoffset (mm)
- **Fallimento test DUT:**
  - Timestamp
  - Username
  - SENSOR_TEST
  - DUT_connesso (0-1-2-3-4)
  - Startup (0-1-2-3-4)
  - UART_connessa (0-1-2-3-4)
  - Test_temperatura (0-1-2-3-4)
  - Test_accelerometro (0-1-2-3-4)
  - Test_MUX (0-1-2-3-4)
  - Test_potenziometro (0-1-2-3-4)
  - Test_AD (0-1-2-3-4)
  - Test_accelerometro_HR (0-1-2-3-4)
  - Test_LoRa (0-1-2-3-4)

  - Configurazione_ricevuta (0-1-2-3-4)
  - Configurazione_installata (0-1-2-3-4)
  - Validazoine_sul_cloud (0-1-2-3-4)

- **Errori di LR-TEST:**
  - Timestamp
  - Username
  - MACHINE_ERROR
  - GUI_present
  - GUI_previous
  - MCL_present
  - MCL_previous
  - BCL_present
  - BCL_previous
  - CCL_present
  - CCL_previous
  - PCL_present
  - PCL_previous

Se tutti i parametri sono corretti, la service risponde, con i dati in formato JSON:

- OK/FAIL
- Message

NB: per maggiori dettagli sui parametri consultare il Capitolo 2.3

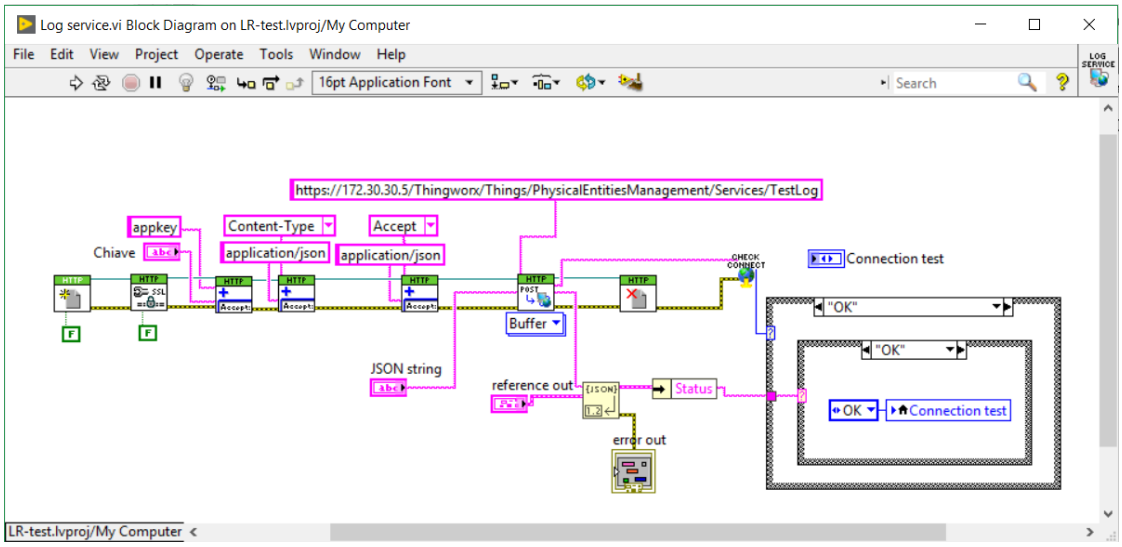

Figura 6.3: SubVi che gestisce la chiamata della service di log

Capitolo 7

Stampante Zebra

Per effettuare la stampa delle etichette viene utilizzata una stampante Zebra GC420t, per la stampa viene usato il linguaggio di programmazione proprietario EPL, il codice viene inviato direttamente tramite LabView.

7.1 Installazione della stampante

Per il corretto funzionamento della stampante tramite LabView bisogna effettuare la configurazione in Windows attraverso i seguenti step:

- Impostare la stamante come predefinita.
- Installare Zebra setup utilities sul PC per ottenere i corretti driver di funzionamento.
- Verificare che il driver Zebra GC420t (EPL) sia correttamente impostato.
- Abilitare la modalità passthrough con inizio sequenza "${" e fine sequenza "}$". Per abilitare questa modalità:
  - Proprietà stampante
  - Generale
  - Preferenze
  - Impostazioni avanzate stampante
  - Altro
  - Attiva modalità passthrough

  Per maggiori dettagli riferirsi a Figura 7.1

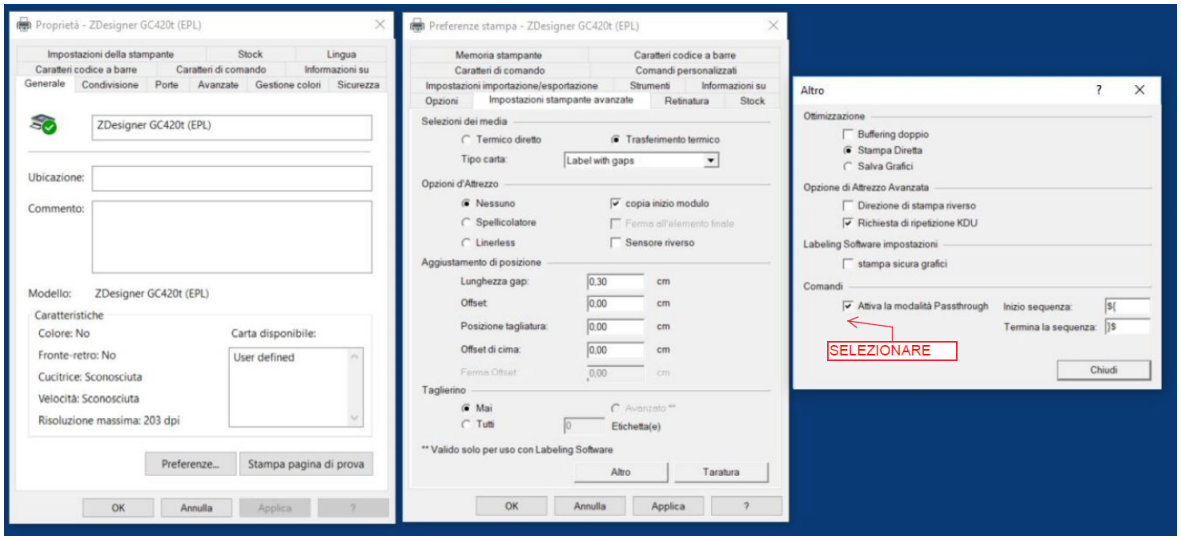

Figura 7.1: Abilitazione della modalità passthrough

7.2 Formato delle etichette

Per realizzare la grafica delle etichette, è stato utilizzato il linguaggio di programmazione EPL (proprietario della Zebra). Con questo linguaggio è possibile realizzare scritte e codici a barre tramite comandi specifici, mentre per la realizzazione delle immagini è stato utilizzato un convertitore online, che permette di inserire un immagine .png e la converte in codice EPL. In Figura 7.2 viene mostrata la SubVi di LabView in cui è presente il codice EPL e la parte di codice che invia il listato alla stampante. Sono state realizzate varie tipologie di etichette da applicare in zone

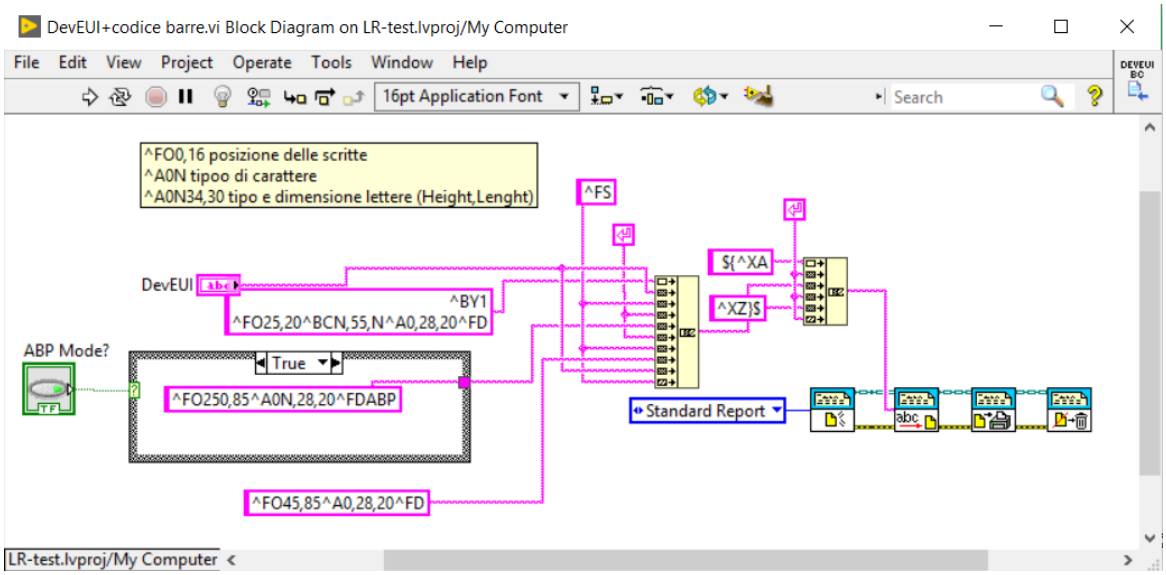

Figura 7.2: SubVi che permette la stampa dell'etichetta DevEUI e codice a barre

diverse del sensore:

- In Figura 7.3 viene mostrata l'etichetta su cui è presente il DevEUI del sensore, il codice a barre corrisponde al DevEUI per facilitare l'organizzazione del magazzino. LR-TEST stampa 2 etichette di questo tipo per ogni DUT, una viene applicata su di esso e una sulla scatola. Viene inoltre visualizzato se il sensore è configurato in modalità APB oppure no.

- In Figura 7.4 viene mostrata l'etichetta su cui sono presenti le informazioni riguardanti la batteria del DUT, in particolare la versione del firmware batteria, la marchiatura CE e RAEE.

Figura 7.3: Etichetta con numero seriale e codice a barre

Figura 7.4: Etichetta con l'identificativo e la versione della batteria

- In Figura 7.5, Figura 7.7, Figura 7.6 e Figura 7.8 viene mostrata l'etichetta su cui sono presenti le informazioni sulla tipologia di sensore, sulla versione del firmware installata, sulla marchiatura CE e RAEE.

Figura 7.5: Etichetta con l'identificativo del sensore e la versione del firmware

Figura 7.6: Etichetta con l'identificativo del sensore e la versione del firmware

Figura 7.7: Etichetta con l'identificativo del sensore e la versione del firmware

Figura 7.8: Etichetta con l'identificativo del sensore ela versione del firmware

7.3 Software per la stampa manuale delle etichette

Sempre utilizzando LabView, è stato realizzato un programma specifico per la stampa manuale delle etichette, chimato Zebra labeler. Questo software viene utilizzato in tutti i casi in cui i DUT vengono testati e configurati manualmente (per esempio in fase di sviluppo o nelle preserie), oppure è necessaria una ristampa delle etichette. L'interfaccia grafica, visibile in Figura 7.9 permette all utente di effettuare le seguenti operazioni:

- Effettuare la configurazione della stampante, in questa fase viene effettuato l'allineamento della testina di stampa, configurata la stampante per utilizzare il linguaggio EPL, infine viene effettuata una stampa di prova.
- Selezionare il tipo di etichetta che si vuole stampare, in base al tipo di etichetta vengono visualizzate le impostazioni aggiuntive corrispondenti (per maggiori dettagli sui parametri aggiuntivi riferirsi al Capitolo 7.2).
- Selezionare il numero di copie da stampare, in caso di stampa di DevEUI+BarCode, è anche possibile stampare più copie con DevEUI crescenti in sequenza partendo da uno indicato dall'utente.
- Interruzione della stampa in caso di errore da parte dell'utente; utile quando si mandano grosse quantità di etichette in stampa e, per esempio, ci si accorge di aver inviato un parametro sbagliato.

Tutto il prorgamma è stato realizzato sensibile al contesto per l'utente, cioè le funzionalità che non possono essere utilizzate in uno specifico momento sono disabilitate; per esempio, in Figura 7.9, il bottone di "Stop" è disabilitato perchè la stampa non è in corso; oppure la casella in cui inserire la versione del firmware non è abilitata perchè in quella specifica etichetta non viene utilizzata.

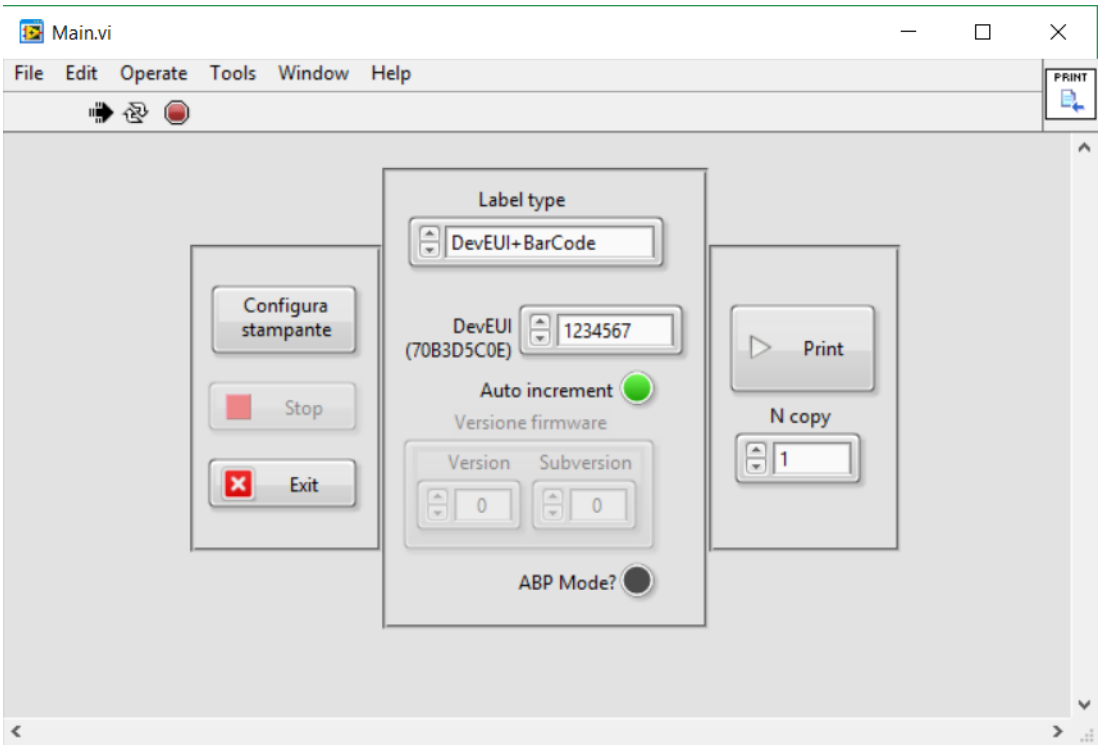

Figura 7.9: Interfaccia grafica di Zebra labeler