

UNIVERSITÀ DEGLI STUDI DI MODENA
E REGGIO EMILIA

Dipartimento di Scienze e Metodi dell'Ingegneria

Corso di Laurea Magistrale in Ingegneria Meccatronica

**Implementazione e Sviluppo di
un'Architettura di Controllo per
una Linea Robotica Adattativa**

Relatore:
Prof. Cristian Secchi

Tesi di Laurea di:
Luca Cristuib Grizzi

Correlatore:
Dott. Ing. Davide Ferrari

Anno Accademico 2022/2023

"Alla mia famiglia"

Sommario

Nelle moderne linee di produzione, la richiesta e la necessità di un approccio adattativo sono sempre più cruciali. Questo non solo consente di incrementare la produttività della linea ma, al contempo, anche di ridurre i prodotti non conformi generati da essa come conseguenza di guasti che si verificano ai dispositivi della linea. Questo lavoro di tesi si prefigge di realizzare un modello in scala di una linea di produzione e di sviluppare un'architettura di controllo adattativa per mitigare alcuni dei sopraccitati problemi. Per raggiungere questa caratteristica, il controllo della linea deve intervenire in risposta a eventuali non conformità con azioni correttive che ripristinino il funzionamento nominale. Gli errori considerati per l'applicazione di algoritmi correttivi, ovvero per la ricerca di una possibile soluzione da applicare, sono tre dei più comuni in una linea di produzione: rallentamento di un dispositivo, errato assemblaggio di un prodotto e mancato arrivo del prodotto alla stazione successiva.

Indice

1 Linee di Produzione Robotiche nell'Industria Moderna	1
1.1 Linee di Produzione: Da Ford all'Industria 4.0	3
1.2 Adaptive Manufacturing	7
1.3 Obiettivo del Lavoro di Tesi	9
2 Hardware e Software	11
2.1 ROS	12
2.2 FlexBE	14
2.3 Niryo	17
3 Tavola Rotante	21
3.1 Cablaggio e Montaggio	24
3.2 Controllo	26
4 Implementazione Linea di Produzione e Algoritmo di Controllo	27
4.1 Layout della Linea di Produzione	30
4.2 Linea di Produzione a Singolo Prodotto	32
4.3 Linea di Produzione Adattiva Multi-Prodotto	35
4.3.1 Gestione Errore di Rallentamento	39
4.3.2 Gestione Errore di Componente Non Arrivato	39
4.3.3 Gestione Errore di Assemblaggio	42
5 Esperimenti	45
5.1 Linea di Produzione a Singolo Prodotto	47
5.2 Linea di Produzione Parallela Multi-Prodotto	48
5.2.1 Errore di Rallentamento	48

INDICE

5.2.2	Errore di Componente Non Arrivato	49
5.2.3	Errore Di Assemblaggio	50
6	Conclusione e Sviluppi Futuri	53
A	Guida Per Settaggio e Uso Della Linea di Produzione	57

Capitolo 1

Linee di Produzione Robotiche nell'Industria Moderna

Le linee di produzione hanno fatto la loro comparsa con la rivoluzione di *Henry Ford*, caratterizzando le industrie fino all'epoca moderna dell'*Industria 4.0*. In questo periodo hanno subito molte variazioni: dall'introduzione di macchinari sempre più avanzati, all'automatizzazione robotica, fino a sfociare nei nuovissimi sistemi produttivi, noti come *Cyber-Physical Systems (CPS)*. Tutte le innovazioni introdotte nel mondo della manifattura sono legate tra loro dallo scopo per cui queste nascono: il miglioramento costante della produttività dell'azienda. Il concetto di produttività di un'azienda non è rimasto inalterato nel tempo, ma si è evoluto di pari passo con il progresso della società, tenendo in considerazione diversi fattori, molti dei quali esterni all'azienda stessa, con particolare attenzione alla domanda di mercato e le sue caratteristiche.

Nella società moderna, la sempre maggior richiesta di personalizzazione da parte del mercato sta portando le industrie del settore manifatturiero verso l'integrazione di una sempre crescente automatizzazione che, però, garantisca elasticità e adattabilità alle linee di produzione. Da ciò, è emerso un nuovo paradigma produttivo, noto come *adaptive manufacturing*. Questo modello si sviluppa su due fronti: gestione aziendale e gestione della produzione. La gestione della produzione, in particolare, esige che le linee di produzione siano in grado di prendere decisioni in modo autonomo e rapido, rispondendo alle richieste del mercato e affrontando gli imprevisti del ciclo produttivo. Questa innovativa modalità di fabbricazione richiede l'ideazione di nuove architetture di controllo per affrontare al meglio le sfide proposte.

Un possibile approccio, introdotto con la moderna *Industria 4.0*, riguarda l'utilizzo dell'*intelligenza artificiale* (*AI, Artificial Intelligence*) per allenare la linea a trovare autonomamente una soluzione ai problemi dell'*adaptive manufacturing*. In alternativa, come proposto in questo lavoro di tesi, è possibile programmare algoritmi classici di controllo adattivo per gestire automaticamente alcuni errori comuni all'interno della produzione, consentendo di migliorare produttività ed efficienza e riducendo blocchi, prodotti difettosi e tempi morti. Anche se questa soluzione è basata su metodologie più tradizionali, si è dimostrata efficace nella gestione degli errori di rallentamento di un dispositivo della linea, di assemblaggio errato e di mancato arrivo del prodotto alla stazione successiva.

In questo primo capitolo verrà presentato un excursus sul mondo dell'industria del passato, del presente e del futuro, necessario alla comprensione delle motivazioni che hanno portato alla realizzazione di questo elaborato. Più precisamente, nella *Sezione 1.1* sono descritti i cambiamenti che sono avvenuti nell'industria manifatturiera a partire dall'inizio del XX secolo fino al giorno d'oggi, con un focus particolare sulle linee di produzione. Nella *Sezione 1.2* verrà descritto l'*adaptive manufacturing* che assume importante rilievo con l'introduzione all'*Industria 4.0*. Infine, nella *Sezione 1.3* vengono delineati gli obiettivi di questo elaborato.

1.1 Linee di Produzione: Da Ford all'Industria 4.0

L'industria, nel corso di tutta la sua esistenza, è sempre stata alla ricerca del miglioramento della propria produttività, seguendo il mercato (domanda e offerta) e il progresso tecnologico, portando così a continue modifiche delle modalità di produzione e, di conseguenza, dell'ambiente di lavoro. Nonostante i risultati di tali sforzi hanno spesso suscitato critiche severe, lo sviluppo della manifattura non si è mai interrotto; anzi, il confronto con le critiche ha permesso talvolta di comprenderle e integrarne parte, soprattutto nei tempi più recenti.

Una pietra miliare in questo ambito fu l'ideazione e l'utilizzo delle linee di produzione, quando *Henry Ford* (*Figura 1.1*), nel 1913, iniziò ad utilizzare la catena di montaggio nello stabilimento di *Highland Park* per la realizzazione dell'automobile *Model T* (*Figura 1.2*)^[1]. Il *fordismo* si poggiava fortemente sul *paradigma industriale tayloristico*, ideato da *Frederick Taylor* nel 1911 e adottato nella catena di montaggio di *Ford*, che si caratterizzava per la parcellizzazione del processo di lavoro per aumentare l'efficienza complessiva della produzione, seguendo quattro capisaldi per migliorare l'organizzazione del lavoro:^[2]

1. Studio scientifico dei migliori metodi di lavoro.
2. Selezione e addestramento scientifico della manodopera.
3. Sviluppo dei rapporti di stima e di collaborazione tra direzione e manodopera.
4. Uniforme distribuzione di lavoro e di responsabilità tra amministrazione e manodopera.

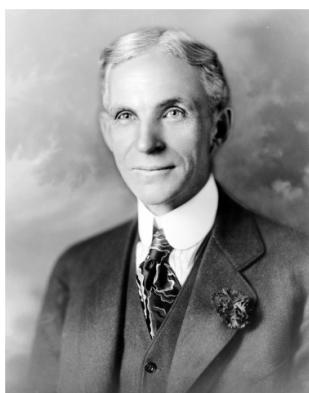


Figura 1.1: Henry Ford ^[3]



Figura 1.2: Ford Model T ^[4]

1.1. LINEE DI PRODUZIONE: DA FORD ALL'INDUSTRIA 4.0

Il metodo di produzione adottato dalla *Ford Motor Company* non fu esente da contestazioni. Sebbene i lavoratori avessero tratto vantaggio dall'orario di lavoro più breve, dei salari più alti con un salario giornaliero di \$5 e dal calo dei prezzi di consumo, erano preoccupati dalla perdita di libertà sul luogo di lavoro e dalle ripercussioni sulla salute dovute alla ripetizione infinita degli stessi movimenti per otto ore che, a detta degli operai, "intorpidivano" l'anima [5]. Nonostante le critiche, la produzione di massa attuata con queste modalità, divenne lo standard nell'industria per gran parte del Novecento.

La svolta si ebbe con la "Terza Rivoluzione Industriale", riferendosi all'evoluzione tecnologica avvenuta a cavallo tra gli anni '60 e '70 del XX secolo [6][7], spesso denominata *rivoluzione digitale*, perché è lo sviluppo di semiconduttori, computer e PLC che permette di introdurre le tecnologie IT e l'elettronica nelle linee di produzione, cambiandole radicalmente [8]. Nella società del dopoguerra, grazie a una crescita culturale ed economica, le persone sono diventate più consapevoli di ciò che desideravano acquistare, dando così origine a nicchie di mercato [9], che richiesero all'industria di adattarsi al cambiamento: non sono più le aziende a regolare il mercato, ma ora esse devono rispondere alle sue richieste, rendendo la flessibilità il *leitmotiv* della produzione.



Figura 1.3: Modicon 084, primo PLC (Programmable Logic Controller) [10]
inventato nel 1968 da Morley, Boishevain, Schwenk e Landau

1.1. LINEE DI PRODUZIONE: DA FORD ALL'INDUSTRIA 4.0

Questi cambiamenti tecnologici e culturali hanno portato alla realizzazione da parte di *KUKA*, ad oggi uno dei fornitori leader a livello mondiale di robot industriali e soluzioni per l'automazione industriale, di due passaggi chiave nella trasformazione delle linee di produzione: la messa a terra nel 1971, per conto di *Daimler-Benz*, della prima linea di produzione per la saldatura in Europa e la progettazione e la realizzazione nel 1973 di *FAMULUS* (*Figura 1.4*), il primo robot industriale a sei assi con motore elettrico [11].



Figura 1.4: Il Robot Industriale FAMULUS del 1973 [11]

L'introduzione dell'automazione ha avuto impatti significativi sul mondo del lavoro, registrando un aumento dell'occupazione di forza lavoro impiegatizia con funzioni di carattere intellettuale, a scapito dei lavoratori manuali e degli operai. Per alcune tipologie di lavoro, si è assistito anche a una dequalificazione delle mansioni, con una conseguente riduzione delle competenze richieste [12].

Ad oggi, il modello predominante nell'industria è ancora quello *postfordista*, anche se già da alcuni anni si è iniziato a parlare di una "Quarta Rivoluzione Industriale", meglio nota con il nome di "*Industria 4.0*", termine utilizzato per la prima volta in una conferenza stampa tenuta nel 2011 alla Fiera di Hannover [13]. *Klaus Schwab*, economista, fondatore e presidente del *World Economic Forum*, nel suo libro "*The Fourth Industrial Revolution*", identifica i tre "*megatrends*" che stanno portando alla "Quarta Rivoluzione Industriale" [8]:

- **Trend Fisico:** coinvolge lo sviluppo tecnologico e si riferisce a innovazioni come: veicoli autonomi, stampa 3D, robotica avanzata e nuovi materiali, noti anche come "*smart materials*".

1.1. LINEE DI PRODUZIONE: DA FORD ALL'INDUSTRIA 4.0

- **Trend Digitale:** connessione tra applicazioni fisiche e digitali grazie all'introduzione di sensori, software e altre tecnologie integrate che abilitano, tra l'altro, l'uso dei "*Digital Twins*", controparti digitali che permettono un'accurata simulazione del mondo reale. Un'altra parte molto rilevante di questo trend è l'intelligenza artificiale e il suo utilizzo sia nella quotidianità sia nella produzione di beni.
- **Trend Biologico:** si riferisce ai progressi significativi degli ultimi anni nel campo della genetica, in particolare nel sequenziamento genetico: si sono ridotti i costi, è aumentata la facilità con cui vengono eseguite le operazioni ed è anche cambiato il metodo di lavoro, che non è più "*trial and error*".

Un'analisi approfondita di quel che sarà l'industria del futuro è stata fatta in un lavoro realizzato da *acatech*, l'Accademia Nazionale Tedesca di Scienze e Ingegneria [14]. Nell'era dell'*Industria 4.0*, le fabbriche dovranno necessariamente adottare il modello di *smart factory*, nel quale la manifattura deve essere raggiunta attraverso l'utilizzo di sistemi di produzione intelligenti, ovvero sistemi basati sulla conoscenza e in grado di comunicare e cooperare tra loro. La richiesta di una personalizzazione massima del prodotto e l'idea di produzione quasi *realtime*, necessitano di flessibilità e riconfigurabilità estreme del processo di produzione. Questa nuova rivoluzione è attuabile solamente grazie all'uso sistematico delle tecnologie *ICT, Information and Communication Technology*, che permettono di trasformare i classici sistemi produttivi fisici nei nuovi *Cyber-Physical Systems (CPS)*. I *CPS* sono sistemi progettati comprendendo tutta la produzione, dai macchinari alle linee di produzione fino anche ai sistemi di immagazzinamento, e consentono di scambiare le informazioni precedentemente rilevate, interpretarle e, infine, prendere decisioni autonomamente. È facile comprendere come i *CPS* facilitino un miglioramento dei processi industriali che riguardano la produzione, l'ingegnerizzazione, l'utilizzo dei materiali e la gestione della *supply chain* e del *life cycle* del prodotto. La *smart factory* è composta da sistemi che sono capaci di interpretare il contesto in cui stanno operando e di definire e realizzare in modo autonomo le operazioni necessarie al raggiungimento degli obiettivi prefissati; inoltre, grazie all'analisi dei dati sono in grado di migliorare le loro prestazioni in funzione dell'esperienza pregressa. In conclusione, il modello di produzione dell'*Industria 4.0* è estremamente reattivo agli stimoli esterni, prende decisioni in modo autonomo e, sin da subito, auto-ottimizza il processo produttivo, mantenendo elevata efficienza e produttività.

1.2 Adaptive Manufacturing

La forte dinamicità del mercato all'interno della rivoluzione 4.0, determinata dalla rapida successione delle nuove tendenze e dalla richiesta estremamente personalizzata dei prodotti, obbligherà inevitabilmente l'industria ad essere ancora più flessibile di quanto lo sia oggi (*Sezione 1.1*). In aggiunta a questo, ci sono anche altri fattori del mercato attorno al mondo industriale che spingono in questa direzione, tra cui: la volatilità della *supply chain*, l'aumento dei prezzi e la carenza di manodopera. Di conseguenza, è necessario ricorrere a nuove strategie di produzione che si adattino al contesto attuale e futuro per mantenere elevata la capacità e l'efficienza produttiva, implementando caratteristiche di flessibilità e di adattamento del processo produttivo, che hanno portato a concepire un nuovo modello: l'*adaptive manufacturing*. Si tratta di un approccio alla produzione che mira a migliorare la flessibilità e la capacità di adattamento dei processi produttivi ai vari fattori esterni, sviluppandosi su due livelli diversi, ma interconnessi tra loro: organizzazione della gestione aziendale e organizzazione della produzione [15].

I sistemi di produzione adattativi sono in grado di ottimizzare autonomamente le loro performance in condizioni ambientali mutevoli, elaborando ed eseguendo comportamenti che siano intelligenti e adattativi. Questo è possibile solamente grazie all'aumento della capacità sensitive dei sistemi stessi grazie alla maggiore diffusione dell'utilizzo della sensoristica, e, soprattutto, all'impiego di complesse logiche di controllo che attuano, dopo un'attenta analisi dei dati, un nuovo comportamento del sistema che ottimizza le produttività in funzione dei nuovi fattori ambientali rilevati [16]. Una peculiarità che molto spesso passa in secondo piano, oltre alla possibilità di adattarsi ad una personalizzazione di massa mantenendo elevata la produttività, è la possibilità di gestire gli imprevisti. Ad esempio, per un prodotto generato in modo non conforme viene adottata un'azione correttiva che riporta il prodotto alla sua forma prevista.

Un processo adattivo può essere scomposto in cinque step: [17]

1. Il componente viene prodotto seguendo le normali procedure.
2. Un sistema visivo esegue delle misurazioni sulle caratteristiche critiche e di interesse del prodotto.
3. Viene elaborata e applicata una strategia per correggere le anomalie riscontrate.

1.2. ADAPTIVE MANUFACTURING

4. Il componente viene ulteriormente ispezionato per assicurarsi che questo sia stato corretto.
5. La parte finita prosegue con le altre operazioni nominali.



Figura 1.5: Step di un Processo Adattivo [17]

In conclusione, i vecchi sistemi di produzione che avevano un processo rigidamente sequenziale non sono più adeguati alle nuove condizioni lavorative. I nuovi sistemi sono altamente reattivi sia agli stimoli esterni che a quelli interni, ovvero agli errori di produzione, portando molteplici vantaggi. Innanzitutto, si assiste ad un miglioramento della qualità di produzione. La capacità di gestire gli imprevisti consente di ridurre gli scarti, inevitabili in una classica linea di produzione, riducendo il costo di produzione. Inoltre, l'azione correttiva viene applicata al volo, senza necessità di blocchi della linea quindi senza perdite di produttività [18]. Un ultimo vantaggio deriva dal fatto che la linea risponde ai cambiamenti in *realtime*, mantenendo le aziende competitive in un mercato estremamente sfidante e complesso, rendendole in grado di adeguarsi rapidamente a nuove condizioni.

1.3 Obiettivo del Lavoro di Tesi

Nel contesto dell'*Industria 4.0*, l'*adaptive manufacturing* emerge come uno degli argomenti di maggior rilievo. In ambito ingegneristico, uno dei più importanti ambiti di ricerca riguarda come rendere i dispositivi meccatronici compatibili con questo modello produttivo. La corretta integrazione di elettronica, informatica e meccanica è il fattore fondamentale per rendere possibile la progettazione e la realizzazione di sistemi produttivi adattativi.

Ad oggi, la quasi totalità delle linee di produzione sono robotizzate e non implementano i principi dell'*Industria 4.0*. Queste riescono ad eseguire compiti in modo ripetibile, ma sono carenti dal punto di vista dell'accuratezza. Per superare questa mancanza è necessario implementare processi adattativi, ovvero creare sistemi che sono in grado di adattare il processo in *realtime* in funzione di un cambiamento di condizioni che viene riscontrato, prendendo decisioni in modo autonomo con grande rapidità e consentendo così di mantenere elevata la produttività. Oltre alla diffusione di sensori per valutare lo stato del processo produttivo, è lecito chiedersi come implementare questa tipologia di controlli. Una soluzione potrebbe essere quella dell'utilizzo dell'intelligenza artificiale per valutare le condizioni e, successivamente, elaborare una strategia che sia da subito ottimizzata. Un'altra strada è quella di sfruttare ancora algoritmi implementati in modo tradizionale. Quest'ultima tipologia richiede maggiore attenzione nella fase iniziale di progettazione e nella comprensione di quali sono le specifiche del processo produttivo su cui concentrare i propri sforzi, portando comunque a ottimi risultati.

Questo lavoro di tesi si concentra sulla realizzazione di un modello di una linea di produzione robotica adattativa, in particolare su un algoritmo di controllo che implementi la gestione degli errori che possono sopraggiungere durante le fasi di assemblaggio dei prodotti. Come visto nella *Sezione 1.2*, questa caratteristica giocherà un ruolo fondamentale nelle linee di produzione delle fabbriche del futuro ed è importante fin da ora trovare le modalità più proficua con cui andare ad inserirla. Nel *Capitolo 4* verrà presentata l'architettura del controllo dell'intera linea e quella che realizza la gestione degli errori per cui è prevista un'azione correttiva.

Capitolo 2

Hardware e Software

La realizzazione in scala di una linea di produzione robotica adattativa, obiettivo di questo lavoro di tesi, è stata concepita all'interno di *ROS* (*Sezione 2.1*), *framework* per la comunicazione tra i vari dispositivi, utilizzando *FlexBE* (*Sezione 2.2*), un'applicazione per realizzare macchine a stati necessarie per l'implementazione dell'architettura di controllo adattiva e per la gestione degli errori. Sono stati impiegati i manipolatori robotici *Niryo* (*Sezione 2.3*), *Ned* e *Ned2* e la *Conveyer Belt*. In aggiunta, è stata costruita una tavola rotante, che verrà dettagliata nel *Capitolo 3*, ed è stata utilizzata la telecamera *Logitech C922 Pro Stream Webcam* [19] per l'acquisizione di immagini.

2.1 ROS

ROS, acronimo di *Robot Operating System*, è un *framework middleware open-source* progettato per lo sviluppo e il controllo di robot [20]. Fornisce strumenti per l’astrazione dell’hardware, per il controllo a basso livello dei dispositivi e per la comunicazione tra processi. Una caratteristica non tecnica, ma molto importante, di *ROS* è la presenza di una vasta e attiva *community* di ricercatori e appassionati nell’ambito della robotica; sebbene questa possa sembrare a primo avviso una qualità di scarsa rilevanza, in realtà si dimostra fondamentale durante lo sviluppo di nuove applicazioni. *ROS* gestisce il *software* suddividendolo in pacchetti. Ognuno di questi contiene diverse risorse tra cui: librerie, file di configurazione, eseguibili, etc. Questa organizzazione semplifica e promuove la condivisione dei componenti software sviluppati con il resto della *community*.



Figura 2.1: *ROS (Robot Operating System)* [20]

La comunicazione e condivisione di dati, messaggi e informazioni attraverso i vari blocchi del software rappresenta il principale punto di forza di *ROS*. Il meccanismo con cui vengono scambiati dati e informazioni si basa su alcuni elementi chiave:

- **Master (ROS Master)**: svolge il ruolo di coordinatore per la comunicazione tra i nodi.
- **Nodo**: è un processo autonomo in cui vengono elaborate delle funzioni per il controllo di un attuatore, di un sensore oppure di un algoritmo generico, ad esempio l’esecuzione di operazioni di *computer vision*.
- **Topic**: è un bus di comunicazione in cui i nodi possono pubblicare messaggi oppure a cui possono sottoscriversi per riceverli.
- **Messaggi (ROS Messages)**: sono i dati scambiati tra i nodi su un topic e vengono classificati in funzione della loro tipologia, che può essere una di quelle standard oppure una personalizzata.

La comunicazione in *ROS* utilizza il meccanismo *publisher/subscriber*. Un nodo che invia dati su un *topic* è detto *publisher*, mentre un nodo che riceve messaggi da un *topic* è detto *subscriber*. Un nodo quando inizia la sua esecuzione si registra al *Master*. Questo, nel momento in cui si accorge che due o più nodi fanno riferimento allo stesso *topic* con un'identica tipologia di messaggio, attiva la comunicazione tra essi (*Figura 2.2*) e permette lo scambio di messaggi.

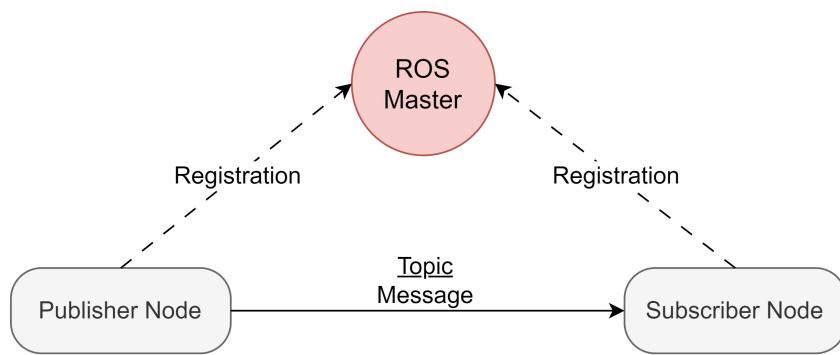


Figura 2.2: Schema di Funzionamento della Comunicazione ROS

2.2 FlexBE

FlexBE (the Flexible Behavior Engine) è un *behavior engine* di alto livello, potente e *user-friendly* e applicabile a numerosi sistemi e scenari, con un focus particolare sui sistemi robotici [21]. In altri termini, è un *framework* per la progettazione e l'esecuzione di comportamenti complessi e flessibili di robot, fornendo uno scheletro su cui costruire la propria applicazione senza la necessità di scrivere manualmente tutto il codice. I *behaviors* vengono gestiti con delle macchine a stati finiti (*Finite State Machine, FSM*), organizzati in stati che si susseguono in modo sequenziale in funzione della transizione attivata (*Figura 2.3*). L'utilizzo di un'interfaccia grafica intuitiva, fornita di un *editor* di tipo *drag&drop*, semplifica notevolmente la realizzazione delle *FSM*. Inoltre, la nativa implementazione di costruzione di *FlexBE* in *ROS*, rende i *behaviors* creati perfettamente compatibili con l'ecosistema di controllo dei robot. L'interfaccia grafica è anche utilizzata per iniziare l'esecuzione e per monitorare il funzionamento delle *FSM*. In aggiunta, *FlexBE* è stata sviluppata seguendo il concetto di autonomia collaborativa, quindi l'utente può interagire con il comportamento del robot durante la sua esecuzione, forzando una transizione tra stati oppure rispondendo a una richiesta del robot. Questa caratteristica risulta molto utile nella fase di *debugging* dei comportamenti. Un'altra peculiarità è la riusabilità dei *behaviors* in diverse situazioni, integrandosi perfettamente con la modularità con cui sono costruiti i sistemi robotici.

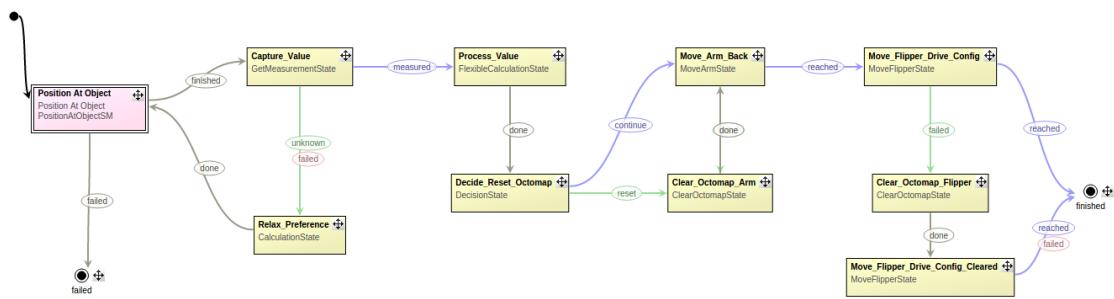


Figura 2.3: Esempio di una FSM per il Behavior di un Robot [22]

L'approccio utilizzato da *FlexBE* per la costruzione dei *behaviors* è basato sugli stati. Ogni stato contiene un algoritmo specifico che implementa un'azione e, in funzione dei risultati di questa, può attivare diverse transizioni, che definiscono come il sistema si sposta tra gli stati. Oltre ad alcuni stati predefiniti è possibile realizzarne

appositamente per la propria applicazione, costruendone il *lifecycle* (*Figura 2.4*) che è composto dalle seguenti funzioni:

- **`__init__`**: è il costruttore dello stato. Quando il *behavior* inizia l'esecuzione, vengono inizializzati tutti gli stati che lo compongono.
- **`execute`**: è la funzione che viene chiamata periodicamente quando lo stato è attivo. Vengono controllate le condizioni per innescare una delle possibili transizioni in uscita. Lo stato rimane attivo fino a quando non viene restituita un'uscita.
- **`on_start`**: è l'evento che viene chiamato contemporaneamente per tutti gli stati quando l'ultimo di questi ha finito l'esecuzione del suo costruttore.
- **`on_enter`**: è l'evento avviato quando lo stato diventa attivo. È l'evento più utilizzato nella programmazione degli stati di *FlexBE*.
- **`on_exit`**: al contrario di *on_enter*, viene eseguito quando uno stato ha finito la sua esecuzione, ovvero quando un'uscita viene restituita e un altro stato diventa attivo.
- **`on_stop`**: è l'evento opposto di *on_start*. Viene chiamato quando il *behavior* termina oppure viene fermato manualmente per eseguire le operazioni di *clean up*.
- **`on_pause`**: è l'evento che viene eseguito quando l'utente, tramite l'interfaccia, mette in pausa l'esecuzione del comportamento. Serve ad implementare le azioni da effettuare per indicare al sistema robotico controllato l'entrata in pausa.
- **`on_resume`**: è l'evento complementare di *on_pause*. Viene attivato quando l'utente fa ripartire l'esecuzione del comportamento dopo che era stata messa in pausa.

2.2. FLEXBE

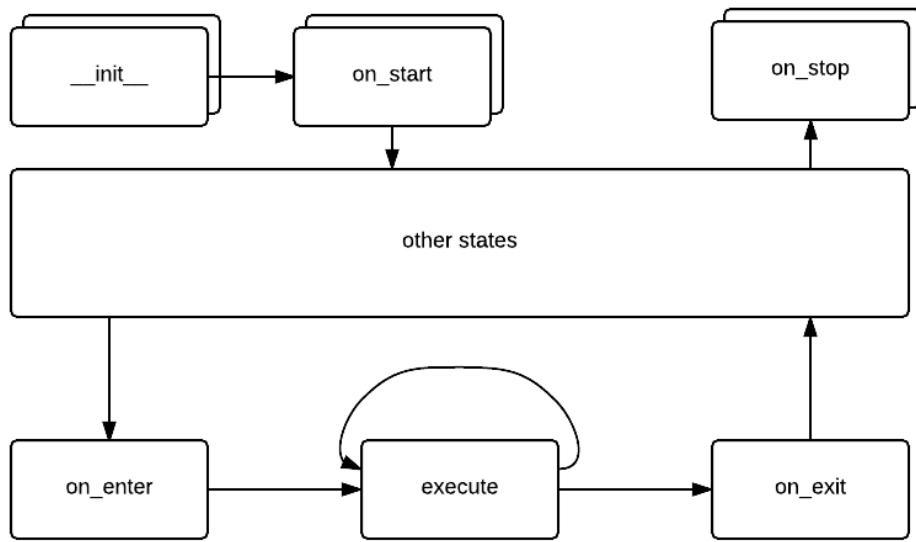


Figura 2.4: Lifecycle degli Stati di FlexBE [22]

2.3 Niryo

Niryo è un'azienda francese specializzata in robotica e automazione educativa con sede nei pressi di Lille [23]. I loro robot non sono di tipo industriale, ma sono pensati per essere utilizzati in ambito educativo, per universitari o ragazzi più giovani, e in ambito di ricerca. Le dimensioni ridotte dei dispositivi li rende ideali per un primo approccio alla robotica, ma anche per riprodurre in scala un sistema di produzione che adotta i principi dell'*Industria 4.0*.

Nella linea di produzione, a cui questo elaborato fa riferimento, sono utilizzati: un robot *Ned*, due robot *Ned2* e la prima versione del loro nastro trasportatore (*Conveyor Belt v1*).



Figura 2.5: *Niryo* [23]

Tutti e tre i dispositivi possono essere controllati con diverse modalità: *Niryo Studio*, direttamente da *ROS* oppure con *PyNiryo*, libreria proprietaria in linguaggio di programmazione *Python*. Per questo elaborato si è scelto di utilizzare la libreria *PyNiryo* per una veloce e flessibile integrazione del controllo dei robot con gli stati di *FlexBE*; questo ha inoltre permesso di parallelizzare l'esecuzione di comandi tra i tre robot.

Ned

Ned [24] (Figura 2.6) è un robot collaborativo a 6 gradi di libertà basato su *Ubuntu 18.04* e *ROS Melodic*. Il controllore è una *Raspberry Pi 4* nella quale sono presenti tutti i pacchetti *ROS* necessari al controllo dei movimenti del robot. *Ned* ha una struttura in alluminio che gli permette di essere robusto e, allo stesso tempo, di riuscire a eseguire con fluidità i movimenti richiesti grazie a motori *stepper*. Le piccole dimensioni (200x200x425mm) e il peso leggero di circa 6,5kg comportano un *payload* di soli 300g e una *reach* massima di 440mm. Il robot è compatibile con diversi *tool* facilmente intercambiabili ed è dotato di porta *Ethernet*, sei GPIO a 5V e due connettori per *NiryoStepper*, ovvero il motore del nastro trasportatore.

2.3. NIRYO



Figura 2.6: Niryo Ned [24]



Figura 2.7: I/O del Niryo Ned [24]

Parametri	Valore
Peso	6,5 kg
Payload	300 g
Reach	440 mm
Range giunti	$-170^\circ \leq \text{giunto 1} \leq 170^\circ$ $-120^\circ \leq \text{giunto 2} \leq 35^\circ$ $-77^\circ \leq \text{giunto 3} \leq 90^\circ$ $-120^\circ \leq \text{giunto 4} \leq 120^\circ$ $-100^\circ \leq \text{giunto 5} \leq 55^\circ$ $-145^\circ \leq \text{giunto 6} \leq 145^\circ$
Limiti di velocità giunti	giunto 1 $\leq 2.6 \text{ rad/s}$ giunto 2 $\leq 2.0 \text{ rad/s}$ giunto 3 $\leq 2.5 \text{ rad/s}$ giunto 4 $\leq 3.14 \text{ rad/s}$ giunto 5 $\leq 3.14 \text{ rad/s}$ giunto 6 $\leq 3.14 \text{ rad/s}$
Ripetibilità	$\pm 0.5 \text{ mm}$

Tabella 2.1: Specifiche Tecniche Niryo Ned

Ned2

Ned2 [25] rappresenta l’evoluzione del robot *Ned*. La struttura generale del robot, il *computer* di controllo (*Raspberry Pi 4* con *Ubuntu 18.04* e *ROS Melodic*) e gli ingombri complessivi sono rimasti sostanzialmente invariati. Oltre alle migliori rifiiture, alla modifica di alcuni connettori e l’aggiunta di GPIO analogici, sono stati aggiunti un anello LED alla base del robot e un altoparlante per segnalare lo stato del robot o eseguire *pattern* impostabili via codice. Infine, sono stati introdotti nuovi motori per il controllo dei giunti, più silenziosi e con migliori *performance*, e sono stati corretti numerosi *bug* nel pacchetto *Niryo ROS Stack* rispetto alla versione precedente.



Figura 2.8: Niryo Ned2 [25]



Figura 2.9: I/O del Niryo Ned2 [25]

2.3. NIRYO

Conveyor Belt v1

La *Conveyor Belt v1* [26] (*Figura 2.10*) è la prima versione del nastro trasportatore prodotto da Niryo, costituita da un singolo motore che muove il nastro. In dotazione con la *Conveyor Belt* è compreso un sensore a infrarossi, posizionabile in qualsiasi punto lungo il nastro grazie ad un apposito supporto; è in grado di rilevare un oggetto entro una distanza massima predeterminata, che può essere regolata in un intervallo compreso tra 6 e 80 cm. Solo il *Ned* è in grado di controllare la prima versione del nastro trasportatore. Il controllo della *Conveyor Belt* può avvenire anche tramite una *Control Box* (*Figura 2.11*) collegata al motore. Questa è dotata di un potenziometro rotativo che consente di variare velocità e direzione del nastro.



Figura 2.10: Conveyor Belt v1 [26]



Figura 2.11: Control Box [26]

Capitolo 3

Tavola Rotante

La tavola rotante è stata progettata *ad hoc* per la realizzazione della linea di produzione che verrà descritta successivamente in questo elaborato, utilizzando un mix di componenti *off-the-shelf* e stampati in 3D. Lo scopo della tavola all'interno della linea è quello di trasportare il prodotto in due punti diametralmente opposti, ovvero deve eseguire una rotazione di 180°. Gli ingombri della tavola, con un diametro di 200mm e un'altezza di 50mm, consentono di integrarsi in modo ottimale con il resto della linea di produzione.

I componenti commerciali sono quattro:

- **Cuscinetti SKF 608:** Cuscinetti a sfere di dimensioni ridotte (diametro esterno di 22mm, diametro interno di 8mm e larghezza di 7mm), le quali consentono di rispettare i vincoli di ingombro presenti nella progettazione della tavola. Inoltre, nel progetto non sono richieste particolari caratteristiche meccaniche per i cuscinetti, quindi la facile reperibilità e il basso prezzo di questo modello hanno inciso nella sua scelta [27].
- **Motore Stepper 28BYJ-48 a 5 V:** Motore passo-passo di piccole dimensioni ampiamente utilizzato in progetti elettronici e hobbistici. L'alimentazione di 5 V lo rende adatto all'uso con microcontrollori. Inoltre, il riduttore presente al suo interno ha un rapporto di 1/64, che gli conferisce buona precisione ed elevate coppie in rapporto alle sue dimensioni [28].
- **Driver ULN2003:** Driver di potenza utilizzato comunemente in combinazione con il motore passo-passo 28BYJ-48. Può essere alimentato sia a 5 V che a

12V in funzione dell'applicazione di cui deve fare parte. Il suo utilizzo è fondamentale in quanto il motore può richiedere una corrente più elevata rispetto a quella che i *pin* del microcontrollore possono fornire, rendendo impossibile un collegamento diretto [29].

- **Raspberry Pi 3 B+:** Computer *single-board* sviluppato da *Raspberry Pi Foundation*, dotato di un processore *quad-core ARM Cortex-A53* a 64 bit con una frequenza di *clock* di 1.4GHz. Supporta diversi tipi di connessione, tra cui: *Wi-Fi*, *Bluetooth Low Energy* versione 4.2, *Ethernet* e *USB*. Ha un *header GPIO* a 40 *pin* che consente di connettere e controllare vari dispositivi elettronici. Infine, usa una scheda *microSD* per l'archiviazione del sistema operativo, che può essere scelto tra i vari disponibili, e dei dati [30].



Figura 3.1: Cuscinetto SKF 608 [27]



Figura 3.2: Motore Stepper 28BYJ-48 [28]

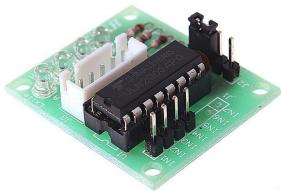


Figura 3.3: Driver ULN2003 [29]



Figura 3.4: Raspberry Pi 3 B+ [30]

Figura 3.5: Componenti Commerciali usati nel Progetto della Tavola Rotante

Il resto dei componenti è stato ottenuto attraverso la stampa 3D in *Onyx*, un materiale composito di nylon e microfibre di carbonio [31], con un riempimento *honeycomb* con *in-fill* del 40-50%:

-
- **Base:** Struttura portante del sistema. In questa si trova una sede dove alloggiare il motore, dei supporti funzionali al fissaggio dei cuscinetti e un perno centrale che, nell'accoppiamento con la tavola superiore, definisce il centro di rotazione. Infine, esiste un'apertura per consentire il passaggio dei fili di cablaggio del sistema.
 - **Tavola Superiore:** Piatto che prevede, al centro della parte sottostante, una sede per il cuscinetto che si andrà ad accoppiare con il perno centrale della base e, nella zona del bordo, una dentatura interna da 85 denti che in abbinamento con la ruota motrice realizza il movimento del sistema.
 - **Ruota Motrice:** Ruota dentata da 15 denti montata sul motore per trasmettere il moto alla tavola superiore.
 - **Spina:** Spinotti di diametro di 6.8mm che servono al fissaggio dei cuscinetti alla base della tavola.

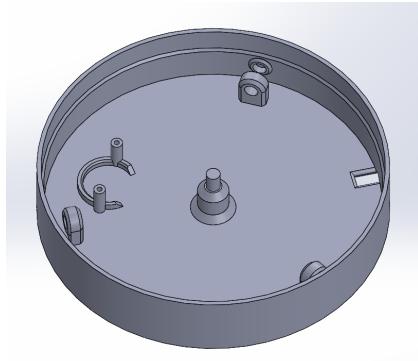


Figura 3.6: Base

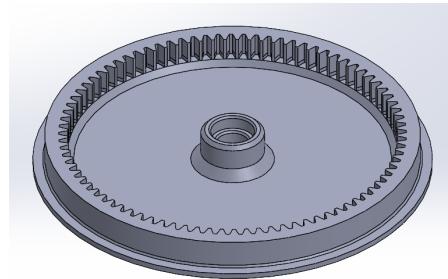


Figura 3.7: Tavola Superiore

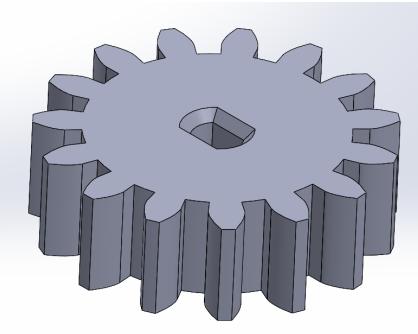


Figura 3.8: Ruota Motrice

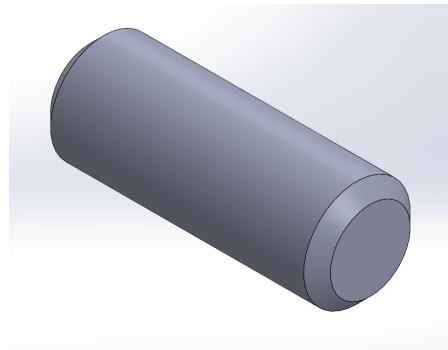


Figura 3.9: Spina

Figura 3.10: CAD 3D dei Componenti della Tavola Rotante

3.1 Cablaggio e Montaggio

In *Figura 3.11* è riportato lo schema elettrico adottato per il controllo della tavola rotante. È importante osservare che la scheda *Raspberry Pi* viene alimentata direttamente dai *pin*, senza passare per dispositivi di protezione da corti o sovratensione; questa scelta richiede particolare accortezza nel corretto collegamento dei *pin* e una tensione di alimentazione precisa, per evitare guasti alla scheda. Inoltre, è fondamentale collegare alla stessa massa sia scheda che *driver*. I quattro ingressi del *driver* (*IN1*, *IN2*, *IN3* e *IN4*), responsabili del controllo del movimento del motore, sono stati collegati arbitrariamente ai *pin* GPIO 17, 18, 27 e 22 della scheda.

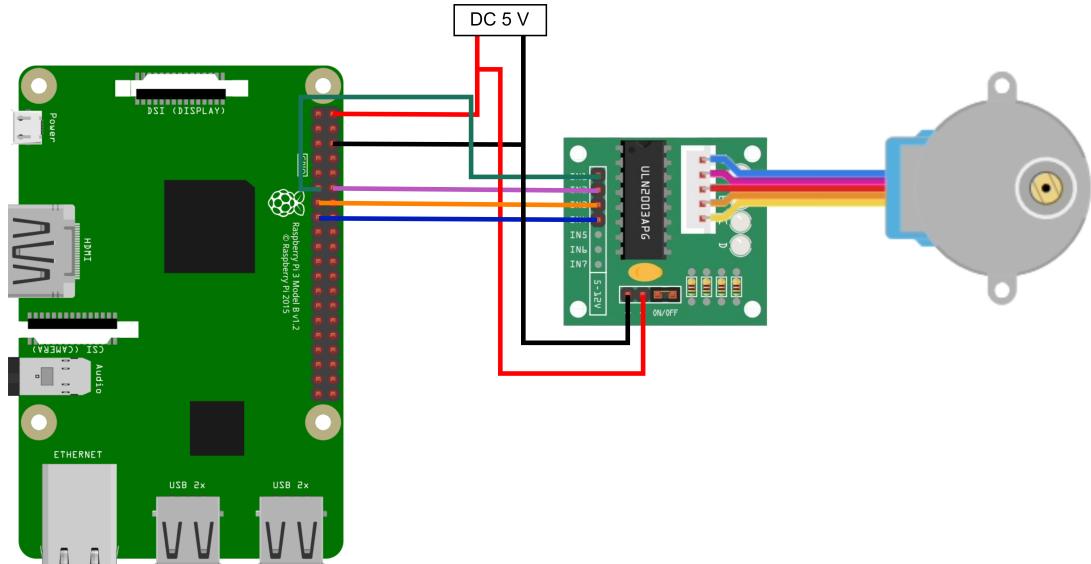


Figura 3.11: Schema Elettrico della Tavola Rotante

3.1. CABLAGGIO E MONTAGGIO

La *Figura 3.12(b)* mostra l'interno della tavola rotante con i componenti in posizione; da notare la presenza di tre cuscinetti nella zona periferica per creare un appoggio stabile e facilitare la rotazione della tavola superiore. *Figura 3.12(a)* mostra invece la tavola superiore con il cuscinetto centrale inserito nell'apposita sede, mentre in *Figura 3.12(c)* si trova il sistema completamente assemblato e cablato.

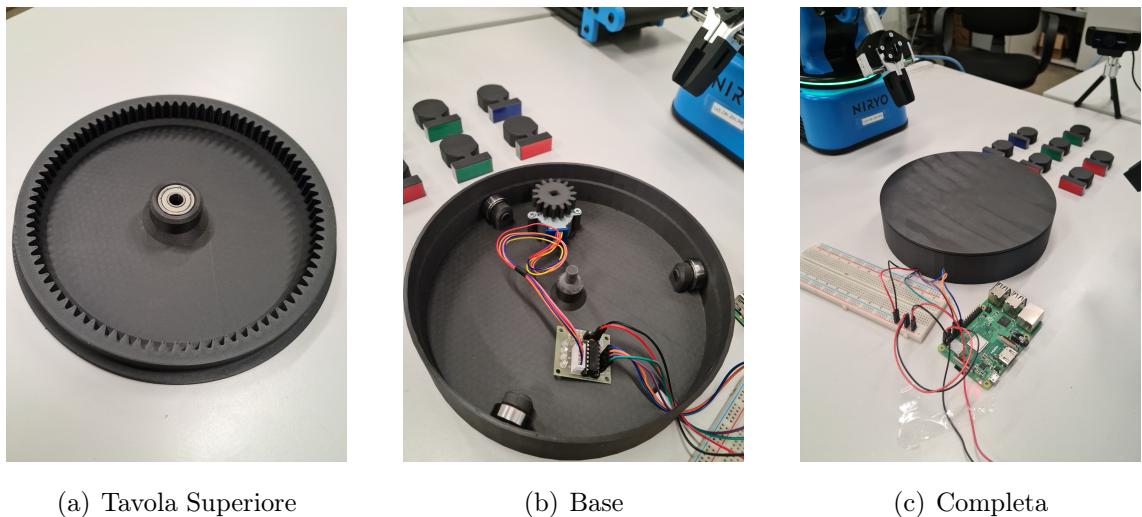


Figura 3.12: Fasi di Montaggio della Tavola Rotante

3.2 Controllo

Il controllo della tavola rotante, come per la linea di produzione, deve essere implementato nel *framework ROS* per potersi sincronizzarsi con gli altri dispositivi della linea ed eseguire il corretto funzionamento. Il sistema operativo (*OS*) da installare su la scheda *Raspberry Pi* deve pertanto essere compatibile e supportato da *ROS*. Questo vincolo introduce delle limitazioni sui possibili *OS* da installare. Inoltre, esiste un ulteriore vincolo, ovvero il modello della scheda *Raspberry Pi* influisce sulle versioni degli *OS* utilizzabili. La scelta è ricaduta su *Ubuntu Server 20.04 LTS* [32]. È una versione del sistema operativo *Ubuntu* specifica per la gestione dei *server*, ma è anche adatta all'uso con schede *IoT*. Essendo un ambiente *headless*, cioè privo di interfaccia grafica e gestito da una *shell* a riga di comando, è particolarmente conveniente in casi in cui la potenza di calcolo del computer e la memoria sono particolarmente ridotte, come per le schede *IoT*.

Il controllo della tavola rotante¹ è basato sul calcolo degli *step* necessari a far compiere al motore una rotazione di 180°. Il motore passo-passo 28BYJ-48, controllato in modalità *half-step*, impiega 64 *step* per completare una rivoluzione dell'albero in ingresso al riduttore; considerando ora il rapporto di riduzione, gli *step* necessari a completare una rivoluzione dell'albero in uscita sono 4096 (*Eq.3.1*). Infine, considerando il numero di denti della ruota motrice (Z_{ruota}), della tavola superiore (Z_{tavola}) e l'angolo desiderato, mediante l'equazione *Eq.3.2* si ottiene il numero di *step* del motore per ottenere una rotazione di 180° della tavola rotante: 11605.

$$\text{Step Albero Uscita} = \frac{\text{Step Albero Ingresso}}{\text{Rapporto Riduttore}} = \frac{64}{\frac{1}{64}} = 4096 \text{ step} \quad (3.1)$$

$$\begin{aligned} \text{Step Per Rotazione Tavola} &= \\ &= \text{Step Albero Uscita} \cdot \frac{Z_{tavola}}{Z_{ruota}} \cdot \frac{\text{AngoloDesiderato}}{360^\circ} \\ &= 4096 \cdot \frac{85}{15} \cdot \frac{180^\circ}{360^\circ} = 11605 \text{ step} \end{aligned} \quad (3.2)$$

¹Codice GitHub: https://github.com/LucaCristuibGrizzi/rotating_table

Capitolo 4

Implementazione Linea di Produzione e Algoritmo di Controllo

L'obiettivo della tesi è la realizzazione di un modello di una linea di produzione robotica adattativa (*Sezione 1.3*). Il primo passaggio è la definizione del *layout* della linea, includendo tutti i dispositivi indicati nella *Sezione 2.3* e nel *Capitolo 3*. Per progettare la configurazione e l'organizzazione della linea, è necessario individuare il prodotto da far assemblare ai robot, considerando alcuni vincoli introdotti dalle operazioni di montaggio e sono i seguenti:

- Le operazioni di rilascio degli oggetti da parte dei robot non garantiscono un'elevata precisione e l'errore generato è difficilmente predicibile, quindi il prodotto deve avere delle tolleranze sufficienti a raggiicare questo problema.
- Il robot *Ned* con il *gripper* disponibile non è in grado di afferrare oggetti al di sotto dei 25mm , inoltre l'apertura massima, sia per il *Ned*, sia per il *Ned2*, è di 49mm .
- Il prodotto deve consentire di eseguire delle operazioni di *computer vision* per la verifica del corretto assemblaggio.
- In caso di errore in fase di assemblaggio deve essere garantita la possibilità di smontare il prodotto e correggerlo.

Si è quindi proceduto alla realizzazione mediante stampa 3D di un prodotto che rispettasse questi vincoli. È composto da due tipologie di componenti:

1. **Case** (*Figura 4.1(a)*): È un parallelepipedo cavo a base quadrata di altezza 60mm e lato 40mm . Nella parte superiore è presente uno smusso, utile per facilitare l'ingresso degli elementi. Nella vista frontale c'è un'apertura dotata di invito che consente di inserire e impilare gli elementi nel *case* in maniera corretta, anche in presenza di un piccolo disallineamento angolare attorno all'asse verticale.
 2. **Elemento Interno** (*Figura 4.1(b)*): Elemento di forma particolare, studiata per agevolare le operazioni di verifica dell'assemblaggio e di smontaggio del prodotto. La parte posteriore è un disco con due tagli laterali, i quali conferiscono all'oggetto un migliore punto di presa per i robot. Questa parte è stata progettata con un gioco di circa 2mm in tutte le direzioni rispetto al *case*, permettendo di assorbire piccoli errori di posizionamento. La parte anteriore è un parallelepipedo sporgente dal *case*, con due funzioni principali: favorire il controllo dell'assemblaggio grazie al colore applicato alla faccia frontale (rosso, verde o blu) e consentire ai robot di afferrare l'oggetto in modo ottimale per rimuoverlo dal *case*.

Il prodotto finito (*Figura 4.1(c)*) è formato da un *case* che ospita al suo interno tre elementi di colore diverso, seguendo un codice colore predeterminato.

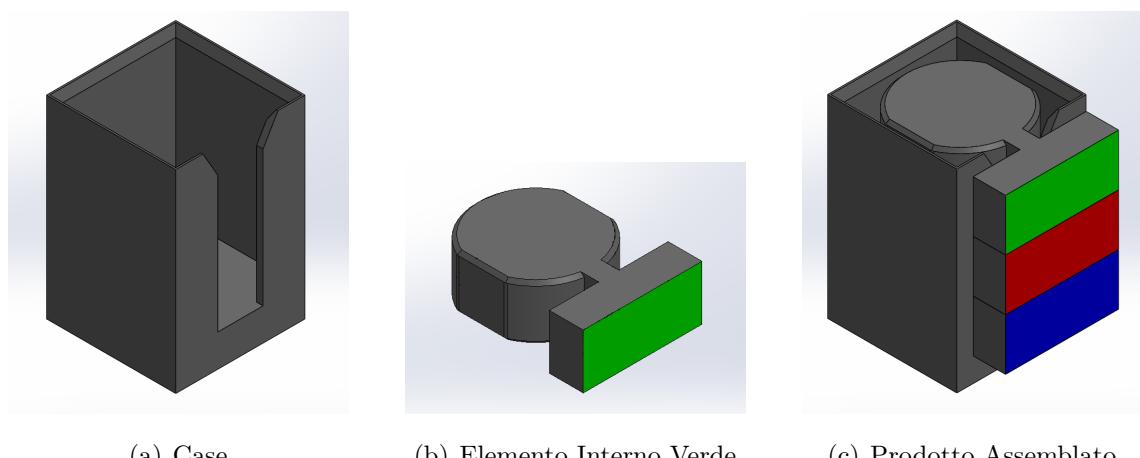


Figura 4.1: Disegni 3D del Prodotto

Un’importante considerazione da fare per la realizzazione di una linea di produzione adattativa riguarda la ridondanza del sistema; per gestire e risolvere possibili errori di produzione è necessario fornire più modalità con cui eseguire un’operazione. Nella linea in questione sono presenti sia zone con ridondanza sia zone senza, come verrà evidenziato nella *Sezione 5.2.2*.

Gli errori di produzione che si possono riscontrare in una linea sono molteplici. Di conseguenza, diventa complesso definire un’azione correttiva per tutti e assume grande rilevanza la scelta degli errori su cui si vuole agire. In questo elaborato, ci si è concentrati su tre degli errori più comuni:

- Rallentamento di un dispositivo (*Sezione 4.3.1*);
- Componente non arrivato alla stazione di lavoro successiva (*Sezione 4.3.2*);
- Assemblaggio del prodotto sbagliato (*Sezione 4.3.3*).

4.1. LAYOUT DELLA LINEA DI PRODUZIONE

4.1 Layout della Linea di Produzione

Nota la conformazione del prodotto e fatta la precisazione sulla ridondanza, è ora possibile illustrare il *layout* della linea di produzione (*Figura 4.2*). È a forma di L, con il processo produttivo che ha inizio sul lato sinistro e si sviluppa verso destra. Il *robot 1* (il *Ned*) deve prelevare dal *magazzino 1* il *case* e l'*elemento* del colore assegnato e assemblarli sulla *Conveyor Belt*. Questa porta il prodotto verso la seconda stazione dove il *robot 2* (un *Ned2*) inserisce il secondo *elemento* prelevandolo dal *magazzino 2*. L'arrivo alla seconda stazione di lavoro è rilevato dal *sensore ad infrarossi* del *nastro trasportatore*. Il *magazzino 2* si trova tra il *robot 2* e il *robot 3* (un altro *Ned2*) ed è condiviso tra i due robot. Il *robot 2* si occupa anche di spostare il prodotto alla terza stazione, posizionandolo sulla *tavola rotante*. In questa stazione, la *telecamera* posta dietro il *magazzino 2* esegue due operazioni: verifica che il prodotto sia arrivato alla postazione corretta e, in caso positivo, controlla che il codice colore del prodotto rispetti quello generato per verificare il corretto assemblaggio del prodotto. Se entrambe le ispezioni hanno esito positivo, la *tavola rotante* completa una rotazione di 180° che porta il prodotto alla quarta stazione di lavoro in cui le operazioni della *telecamera* e del *robot 3* sono intervallate tra loro: per prima cosa la *telecamera* esegue una nuova scansione per controllare che il prodotto sia sul lato corretto della tavola, in seguito il *robot 3* inserisce il terzo e ultimo *elemento*, prelevandolo dal *magazzino 2* e, infine, dopo una rotazione del prodotto di 180° per riallineare il lato colorato degli elementi alla *telecamera*, essa controlla nuovamente il corretto montaggio. Se non vi sono errori, il prodotto finito viene riposto dal *robot 3* nel *magazzino 3* di fine linea.

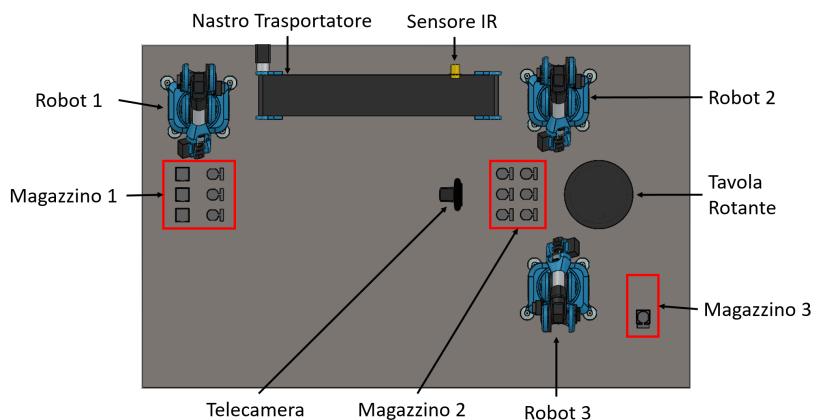


Figura 4.2: Layout della Linea di Produzione

4.1. LAYOUT DELLA LINEA DI PRODUZIONE

È importante evidenziare due particolarità della linea:

- Posizione del magazzino 3: Questo si trova nascosto dietro il *robot 3* rispetto alla vista della *telecamera*. Il posizionamento non è casuale. Infatti, se i prodotti finiti fossero sistemati posteriormente alla *tavola rotante*, quindi visibili alla *telecamera*, ostacolerebbero la corretta esecuzione delle operazioni di *computer vision*, potenzialmente portando a risultati errati.
- Posizione della tavola rotante: Questo dispositivo si trova tra il *robot 2* e il *robot 3* ma è disallineato rispetto ai due robot. Questa configurazione consente di avvicinare il *magazzino 2* ai robot, in modo tale che tutti gli elementi di questo cadano all'interno del *workspace* di entrambi. Inoltre, la disposizione non allineata semplifica le operazioni di smontaggio del prodotto che possono avvenire sulla *tavola*, poiché riduce la complessità delle pose che i robot dovrebbero assumere con l'allineamento dei tre dispositivi.

Il processo appena descritto illustra le operazioni coinvolte nell'assemblaggio di un singolo prodotto. Ulteriori dettagli, specifici dell'architettura di controllo, verranno introdotti nella *Sezione 4.2* e nella *Sezione 4.3*.

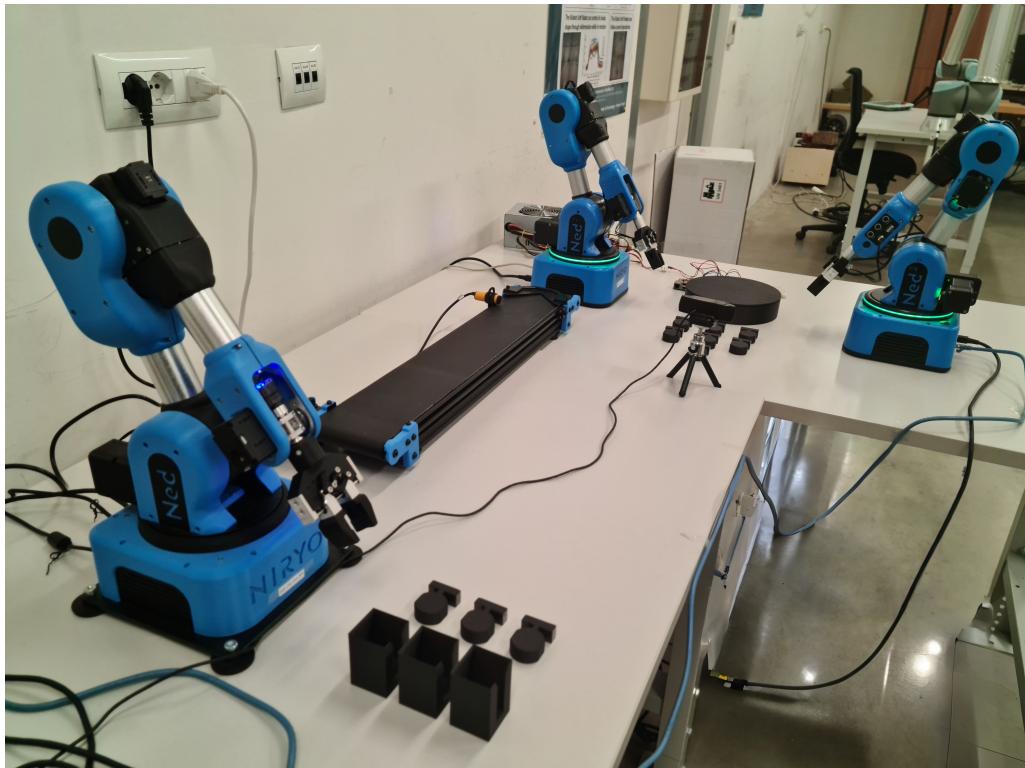


Figura 4.3: Linea di Produzione

4.2 Linea di Produzione a Singolo Prodotto

La prima versione della linea di produzione è stata pensata per realizzare un singolo prodotto in modo sequenziale; ogni dispositivo esegue la sua operazione solamente al termine della precedente. Sebbene questa versione base sia una versione semplificistica della linea finale (*Sezione 4.3*), è utile come termine di confronto tra le prestazioni che si ottengono dai due diversi modelli.

La caratteristica principale della linea a singolo prodotto è l'attuazione sequenziale dei passaggi di assemblaggio. Pertanto, l'utilizzo di una macchina a stati finiti è perfetta per implementare l'architettura di controllo. Come descritto nella *Sezione 2.2*, *FlexBE* è un'applicazione che permette di creare per via grafica delle *FSM* che realizzano complessi comportamenti di sistemi robotici. Prima di entrare nei dettagli della *FSM* che controlla la linea, va evidenziato il fatto che questo modello non è una linea adattativa, perciò non è prevista la gestione degli errori. Inoltre, un'ipotesi fatta e valida per il prosieguo dell'elaborato, è che le posizioni utili ai robot per eseguire il montaggio del prodotto siano note a priori. Questa potrebbe sembrare un'ipotesi restrittiva, tuttavia, grazie alla modularità con cui è stato implementato il controllo dei robot, è possibile implementare, in applicazioni future, l'utilizzo di sistemi di *object recognition* e *pose estimation* per ottenere le posizioni dei componenti. La *FSM* della linea per singolo prodotto (*Figura 4.4*) è suddivisa in tre macro-blocchi: "*Color Code State*" che genera il codice colore del prodotto che la linea dovrà realizzare, "*Main State*" che definisce le posizioni dei componenti necessari all'assemblaggio e "*Niryo Production Line*" che si occupa dell'esecuzione effettiva delle operazioni.

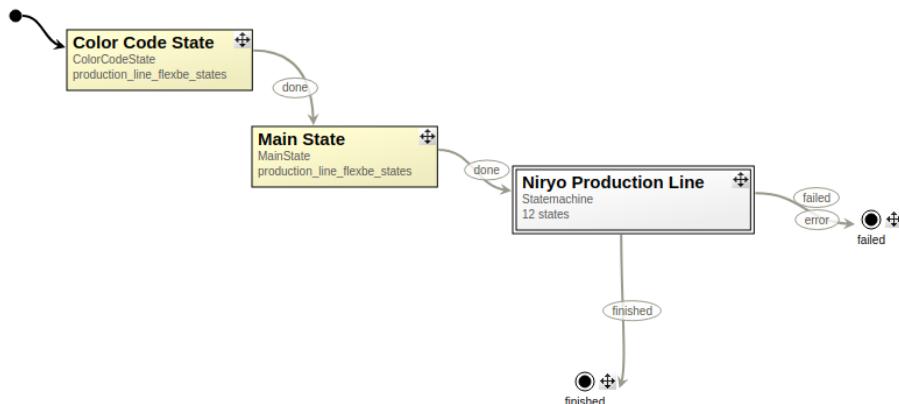


Figura 4.4: FSM per la Linea di Produzione a Singolo Prodotto

4.2. LINEA DI PRODUZIONE A SINGOLO PRODOTTO

Lo stato *Niryo Production Line* è a sua volta suddiviso in molteplici stati che controllano l'effettiva attuazione dei componenti della linea. Segue una descrizione degli stati, riportati in ordine di esecuzione:

- **R1 PickAndPlace State:** Richiede al *robot 1* di effettuare il *pick & place* del *case* e del primo *elemento*. Infine, aspetta il segnale di fine operazione per passare allo stato successivo.
- **Conveyor Belt Run State:** Avvia il *nastro trasportatore* per portare il prodotto alla seconda stazione di lavoro.
- **Check IR:** Ascolta la risposta del *sensore IR*; quando questo rileva il prodotto si passa allo stato successivo. In caso il prodotto non arrivi alla seconda stazione entro un tempo limite predeterminato, la *FSM* verrà reindirizzata in uno stato di errore.
- **Conveyor Belt Stop State:** Comunica al *nastro trasportatore* di fermarsi. Lo stato viene attivato dalla transizione dello stato *Check IR* innescata dalla risposta affermativa del *sensore IR*.
- **R2 PickAndPlace State:** Richiede al *robot 2* di eseguire il *pick & place* del secondo *elemento* e del prodotto sulla *tavola rotante*. Infine, aspetta il segnale di fine operazione per passare allo stato successivo.
- **Object Detection State 1:** Invia alla *telecamera* il comando per controllare che il prodotto sia effettivamente sulla *tavola rotante*. In caso affermativo, avviene la transizione allo stato successivo, altrimenti viene generato un errore.
- **Check Assembly State 1:** Invia alla *telecamera* il comando per controllare che il prodotto sia assemblato correttamente, verificando che i primi due *elementi* rispettino il codice colore generato. Le transizioni in uscita da questo stato seguono le stesse modalità dello stato *Object Detection State 1*.
- **Rotating Table State:** Comanda alla *tavola rotante* una rotazione di 180°, portando così il prodotto alla quarta stazione di lavoro. Il passaggio allo stato successivo avviene alla fine dell'operazione.
- **Object Detection State 2:** Esegue la stessa operazione del primo stato di *object detection*, però questa volta verifica che il prodotto sia sul lato opposto della tavola, controllando l'avvenuta rotazione della *tavola*.

4.2. LINEA DI PRODUZIONE A SINGOLO PRODOTTO

- **R3 PickAndPlace State 1:** Comunica al *robot 3* di inserire il terzo *elemento* e di ruotare il prodotto di 180° per eseguire la verifica del montaggio.
- **Check Assembly State 2:** Duplicato dello stato *Check Assembly State 1*, ma il controllo del codice colore avviene con tutti e tre gli *elementi* inseriti.
- **R3 PickAndPlace State 2:** Come gli altri stati di *pick & place*, viene indicato al *robot 3* di portare il prodotto nel magazzino di fine linea.

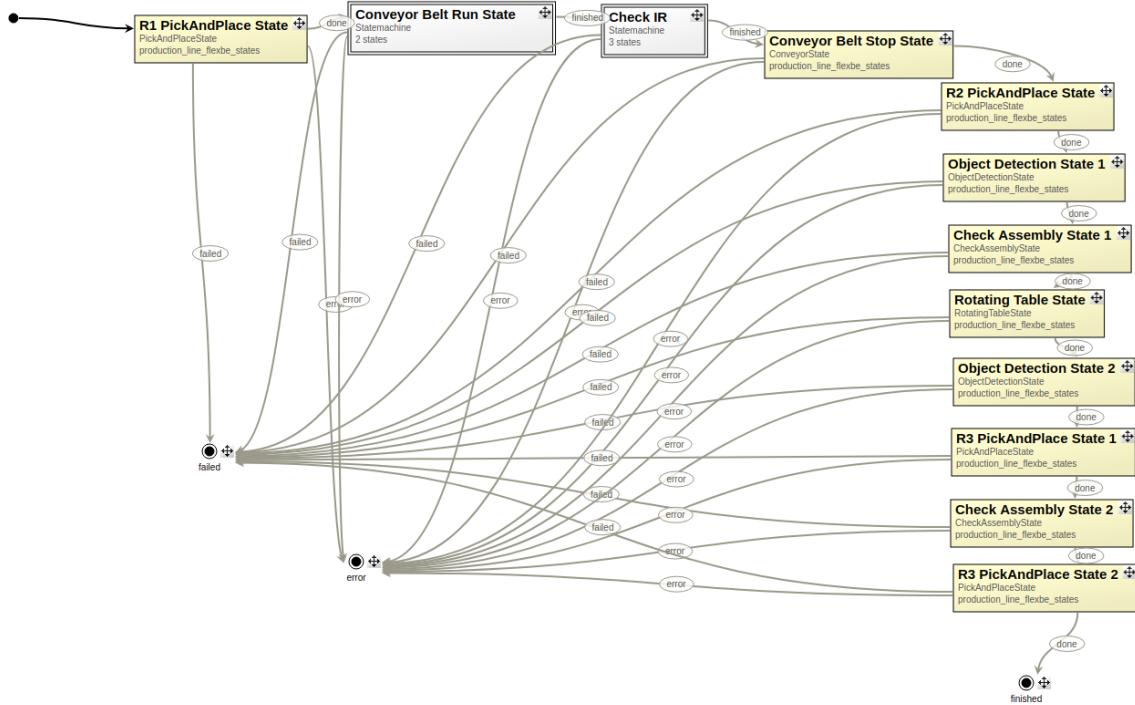


Figura 4.5: Niryo Production Line: Controllo delle Operazioni della Linea

Gli stati di *FlexBE* non eseguono in modo diretto il controllo dei vari dispositivi della linea, ma ciascun dispositivo è associato al proprio nodo *ROS*, all'interno del quale sono inseriti gli algoritmi di controllo. Ogni stato si occupa di generare i messaggi e di comunicare lo stato delle operazioni attraverso *publisher* e *subscriber* con il nodo di riferimento. Questa caratteristica permette di assicurare un'elevata modularità; gli stati non fanno riferimento al modello specifico del dispositivo con cui comunicano, permettendo una facile sostituzione dei dispositivi della linea con qualsiasi altro modello esistente, purché la comunicazione tra il nodo *ROS* e lo stato di *FlexBE* rimanga invariata. Inoltre, si potranno creare nuovi processi produttivi semplicemente aggiungendo e/o rimuovendo stati e collegandoli in modo diverso.

4.3 Linea di Produzione Adattiva Multi-Prodotto

Per rendere la linea di produzione più vicina ad un'applicazione reale, è importante implementare l'esecuzione delle operazioni dei dispositivi in parallelo, per permettere l'esecuzione simultanea di molteplici prodotti. Questo cambiamento aumenta la complessità della linea e errori o rallentamenti generati su un prodotto possono inficiare la realizzazione anche dei successivi; è perciò necessario rendere la linea adattiva, implementando una corretta comunicazione tra i dispositivi per renderli in grado di gestire gli errori. Il processo produttivo del singolo articolo rimane identico a quello illustrato precedentemente, ma ciascun dispositivo passerà al prossimo componente senza aspettare la fine di tutte le operazioni sul precedente. Per raggiungere il parallelismo delle operazioni, è stata implementata un'architettura di controllo gerarchica (*Figura 4.6*), suddivisa in tre livelli distinti: *Livello Alto* per il controllo della gestione degli errori, *Livello Intermedio* per il controllo della linea di produzione e *Livello Basso* per il controllo dei singoli dispositivi della linea. La comunicazione tra i livelli avviene seguendo regole precise: un livello più alto può sempre comunicare con uno più basso, ma il viceversa può avvenire solamente se il livello più basso è stato precedentemente attivato da una comunicazione di un livello più alto; in questo caso la comunicazione diventa biunivoca e i nodi del *Livello Basso* hanno la possibilità di raggiungere il *Livello Intermedio*.

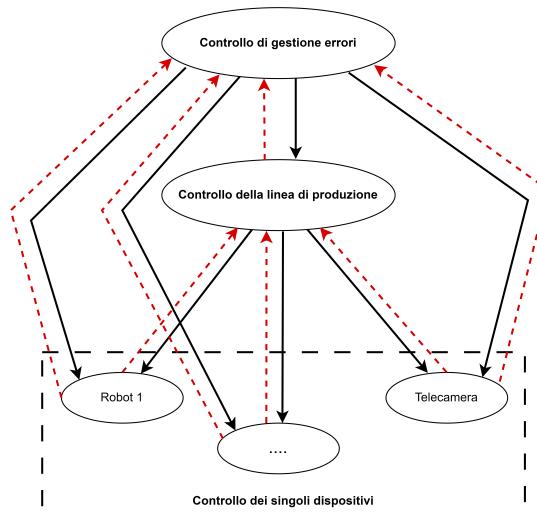


Figura 4.6: Gerarchia dell'Architettura di Controllo della Linea Multi-Prodotto.

Le frecce in nero rappresentano la comunicazione dall'alto verso il basso;
le frecce rosse e tratteggiate rappresentano la comunicazione opposta.

4.3. LINEA DI PRODUZIONE ADATTIVA MULTI-PRODOTTO

Grazie alla modularità e all'implementazione del codice di controllo dei dispositivi in nodi *ROS* distinti, è possibile fare eseguire in contemporanea operazioni ai diversi *device*. Il problema si sposta su come sincronizzare il controllo della linea di produzione per azionare i suoi elementi per ottenere un'esecuzione parallela. L'impiego di una *FSM* da sola non è più adatto per ottenere il parallelismo, in quanto una macchina a stati è, per definizione, sequenziale. La soluzione adottata è l'uso di un nodo *ROS* che, tramite il meccanismo *publisher/subscriber* consente di associare una *callback*, cioè una funzione, a un *subscriber*. Questa funzione viene eseguita in un *thread* isolato dal resto del processo ogni volta che il *subscriber* riceve un messaggio; un nodo *ROS* con molteplici *subscribers* è di default un ambiente *multithreading* in grado di eseguire più operazioni in parallelo. L'aggiunta di variabili globali consente di far interagire i vari *thread*, creando un contesto ideale dove sviluppare il controllo della linea. In caso le condizioni per attivare una data operazione siano state raggiunte, effettuando la verifica mediante un *thread* ciclico di controllo, è possibile utilizzare il *publisher* relativo al dispositivo per avviare le sue operazioni; le *callbacks* di fine operazione dei singoli *device* vengono eseguite, invece, in modo discontinuo e agiscono sulle variabili globali, modificando le condizioni del *thread* principale (*Figura 4.7*).

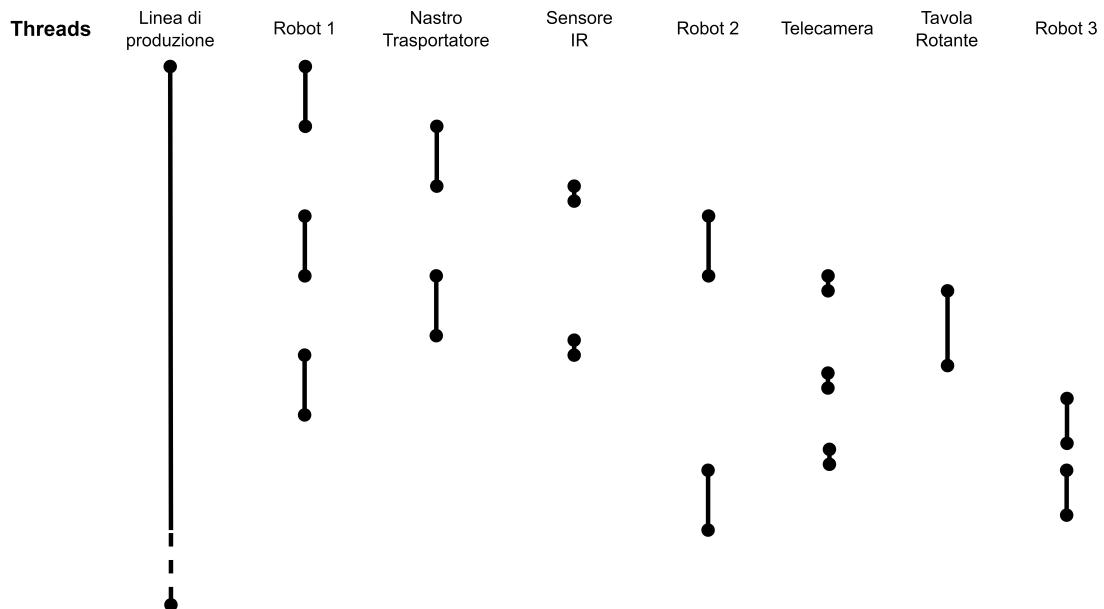


Figura 4.7: Esempio di Esecuzione dei Thread delle Callback dei Subscribers

4.3. LINEA DI PRODUZIONE ADATTIVA MULTI-PRODOTTO

Il *Livello Alto* è comunque possibile gestirlo con una *FSM* (*Figura 4.8*). Questa ha una costruzione semplice, infatti la maggiore complessità è inserita negli algoritmi di sincronizzazione delle operazioni nel nodo *ROS* della linea di produzione e in quelli di ricerca della soluzione a un errore.

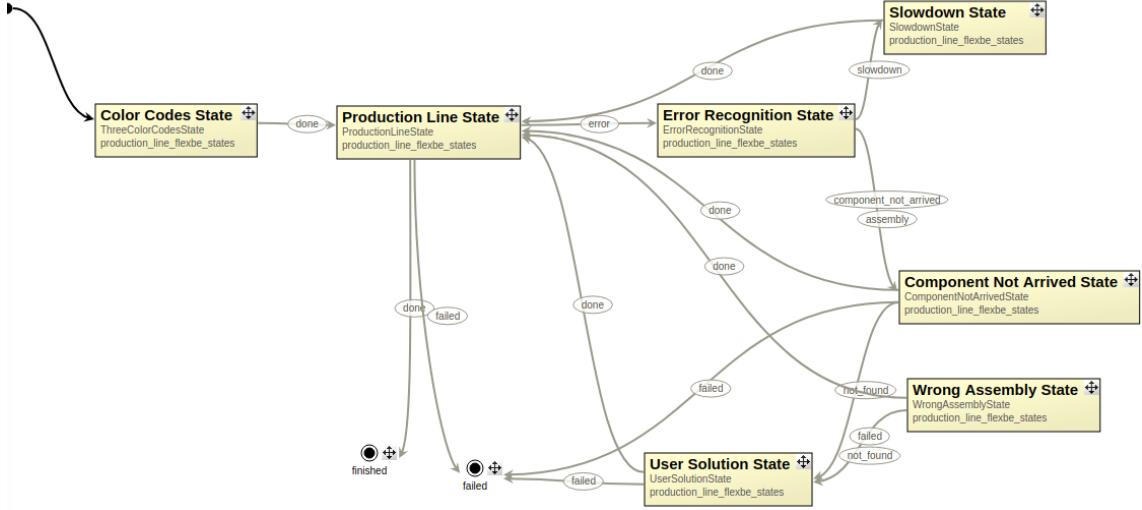


Figura 4.8: FSM di FlexBE per la Linea di Produzione Multi-Prodotto

Di seguito l'elenco degli stati implementati in questo *behavior* di *FlexBE*:

- **Color Codes State:** Genera in modo randomico i codici colore dei prodotti da assemblare, tenendo in considerazione la presenza nella linea di solamente nove *elementi* (uno per colore nel *magazzino 1* e due per colore nel *magazzino 2*) e l'esigenza che ogni prodotto contenga tutti e tre i colori.
- **Production Line State:** Definisce le posizioni nelle quali i robot troveranno l'*elemento* del colore corretto nei magazzini, in modo da assemblare i prodotti secondo il loro codice colore. Creato il messaggio *ROS*, lo invia al nodo della linea di produzione avviando il suo funzionamento. Inoltre, a *runtime*, gestisce la variabile che rappresenta lo stato dei magazzini, utilizzata in seguito. Le transizioni in uscita sono il completamento di tutte le operazioni oppure la captazione di un errore nella linea. In aggiunta, la struttura di questo stato rende l'architettura del controllo del *Livello Alto* indipendente dalla parte sottostante, ovvero essa può essere utilizzata con una linea di produzione diversa.

4.3. LINEA DI PRODUZIONE ADATTIVA MULTI-PRODOTTO

da quella di questo elaborato, offrendo anche la possibilità di applicarla, con piccoli accorgimenti, a una linea già esistente per renderla adattativa.

- **Error Recognition State:** Attiva il corrispondente stato di gestione dell'errore in funzione della tipologia di errore riscontrato. Una peculiarità di questo stato è di avere *output* dinamici, ovvero, a differenza degli altri stati, le sue uscite possono essere modificate dall'utente nell'interfaccia grafica di *FlexBE* in base all'applicazione che si sta sviluppando, fornendo la possibilità di aggiungere la gestione di nuovi errori.
- **Slowdown State:** Gestisce l'errore di rallentamento di un dispositivo della linea (*Sezione 4.3.1*). Una volta applicata la soluzione, torna ad attivare lo stato *Production Line State*.
- **Component Not Arrived State:** Gestisce l'errore di un componente che non è arrivato alla stazione di lavoro prevista (*Sezione 4.3.2*). Se la soluzione è stata trovata e applicata, la transizione porta allo stato *Production Line State*; se la soluzione non è stata trovata, viene attivato lo stato *User Solution State*.
- **Wrong Assembly State:** Gestisce l'errore di assemblaggio errato (*Sezione 4.3.3*). Le uscite ripercorrono quanto visto per lo stato *Component Not Arrived State*.
- **User Solution State:** Attivato quando la gestione degli errori non trova nessuna soluzione da applicare, viene aperta una finestra nel *desktop* in cui si informa l'utente del tipo di errore e del dispositivo che lo ha generato. Inoltre, viene chiesta la conferma di risoluzione del guasto: in caso affermativo la linea riprende il suo funzionamento, in caso contrario viene bloccata.

La comunicazione dell'errore e dei relativi dati avviene sempre dal *Livello Intermedio*, ossia dal nodo della linea, verso il *Livello Alto*, in particolare allo stato *Production Line State*. Dall'architettura della *FSM* è possibile intuire che, durante la fase di gestione degli errori, il *thread* della linea di produzione non è attivo, pertanto, per un certo periodo, non vengono eseguiti nuovi comandi. Il nodo della linea non viene chiuso., permettendo alle risposte di fine operazione dei comandi precedentemente inviati di essere comunque gestite normalmente e, alla ripresa del funzionamento nominale, continuare il processo come se la linea non fosse mai stata interrotta.

4.3.1 Gestione Errore di Rallentamento

L'*errore di rallentamento* si verifica quando uno dei dispositivi mobili della linea di produzione, in questo caso i *robot*, il *nastro trasportatore* o la *tavola rotante*, deve diminuire la sua velocità di esecuzione a causa di un guasto temporaneo o meno. In questa situazione, l'obiettivo è di sincronizzare la lavorazione della linea alla velocità del dispositivo guasto. Il meccanismo di gestione dell'errore utilizza il seguente schema: un nodo del *Livello Basso* lancia un errore e indica la sua nuova velocità, il nodo del *Livello Intermedio* capta l'errore e lo passa al *Livello Alto* da cui viene attivata la gestione dell'errore: lo stato *Slowdown State* comunica a tutti i nodi dei *device* della linea la nuova velocità di lavorazione. Rallentando alla velocità del dispositivo più lento vengono eliminati i tempi morti che si sarebbero venuti a creare senza una gestione opportuna dell'errore.

È importante sottolineare che ogni nodo che riceve la nuova velocità può elaborare questo dato in modo diverso; la strategia da adottare di fronte a questo errore può cambiare in funzione del dispositivo e della sua posizione nella linea, al fine di ottimizzare le prestazioni globali. Sarà compito del programmatore di scegliere e implementare le migliori strategie.

4.3.2 Gestione Errore di Componente Non Arrivato

L'*errore di componente non arrivato* si concretizza quando il prodotto, durante il passaggio tra due stazioni della linea di produzione, non raggiunge la successiva. Nella linea realizzata ci sono tre casi possibili in cui questo errore viene rilevato:

- Mancato arrivo alla fine del *nastro trasportatore*.
- Dopo l'operazione del *robot 2*: assenza sulla *tavola rotante*.
- Dopo la rotazione della *tavola rotante*: presenza sul lato sbagliato.

Il meccanismo di comunicazione, ancora una volta, va dal *Livello Basso* al *Livello Alto*. In questo caso, però, il *Livello Intermedio* aggiunge delle informazioni in quanto è l'unico a poter comprendere qual è il dispositivo guasto.

Nella progettazione dell'algoritmo risolutivo di questo errore sono state considerate due ipotesi:

1. Il *device* che ha generato l'errore viene escluso dalla possibile soluzione, in quanto presumibilmente guasto.

4.3. LINEA DI PRODUZIONE ADATTIVA MULTI-PRODOTTO

2. Solo i dispositivi più vicini possono risolvere l'errore, con una preferenza per quelli successivi.

La seconda ipotesi è legata alla ridondanza del sistema, ma soprattutto ad un fattore di onere computazionale. Infatti, considerando una linea di grandi dimensioni, la ricerca di una soluzione al dispositivo i -esimo+10, a seguito di un errore al dispositivo i -esimo, avrà con altissime probabilità un esito negativo dovuto alle grandi distanze tra essi. Pertanto, valutare tutte le possibili soluzioni con tutti i dispositivi, sapendo già che quelli più lontani saranno inutili, non aggiunge valore al risultato ottenuto ma aumenta solo l'onere computazionale.

L'algoritmo per la ricerca della azione correttiva da applicare implementa il diagramma di flusso in *Figura 4.9*: rilevato l'errore e noto il dispositivo guasto (dispositivo i -esimo), la prima operazione è la selezione dei due dispositivi più vicini a quello guasto. Si presentano tre situazioni:

- Il dispositivo i -esimo è l'ultimo della linea, quindi vengono scelti i due dispositivi precedenti (in ordine il k -esimo e il j -esimo).
- Il dispositivo i -esimo è il penultimo, quindi vengono scelti il primo successivo e il primo precedente (rispettivamente il k -esimo e il j -esimo).
- Altrimenti, vengono scelti i due dispositivi successivi (rispettivamente il k -esimo e il j -esimo).

Il passaggio successivo si occupa di determinare le posizioni di *start* e di *goal*, rispettivamente la stazione di lavoro dove si trova il prodotto e quella in cui dovrebbe arrivare. Per ciascuno dei dispositivi scelti si effettua la ricerca della capacità di raggiungere le posizioni necessarie alla risoluzione dell'errore, dando precedenza al k -esimo dispositivo in quanto potenzialmente più probabile che sia il più vicino a entrambe le stazioni, soprattutto a quella di *start*. In sequenza, si controlla se il k -esimo dispositivo può raggiungere la posizione di *start* e, in caso di risposta positiva, si controlla se lo stesso può raggiungere anche la posizione di *goal*. In caso affermativo si applica la soluzione, diversamente si verifica se il j -esimo può raggiungere quest'ultima posizione. In caso di risposta negativa non esiste una soluzione, altrimenti è possibile impiegare entrambi i dispositivi, ma è necessario accertare che esista una posizione in comune dove eseguire il passaggio del prodotto. Se viene trovata una soluzione questa viene applicata, in caso contrario la linea rimane ferma. In generale, la soluzione che utilizza un solo dispositivo è preferibile.

4.3. LINEA DI PRODUZIONE ADATTIVA MULTI-PRODOTTO

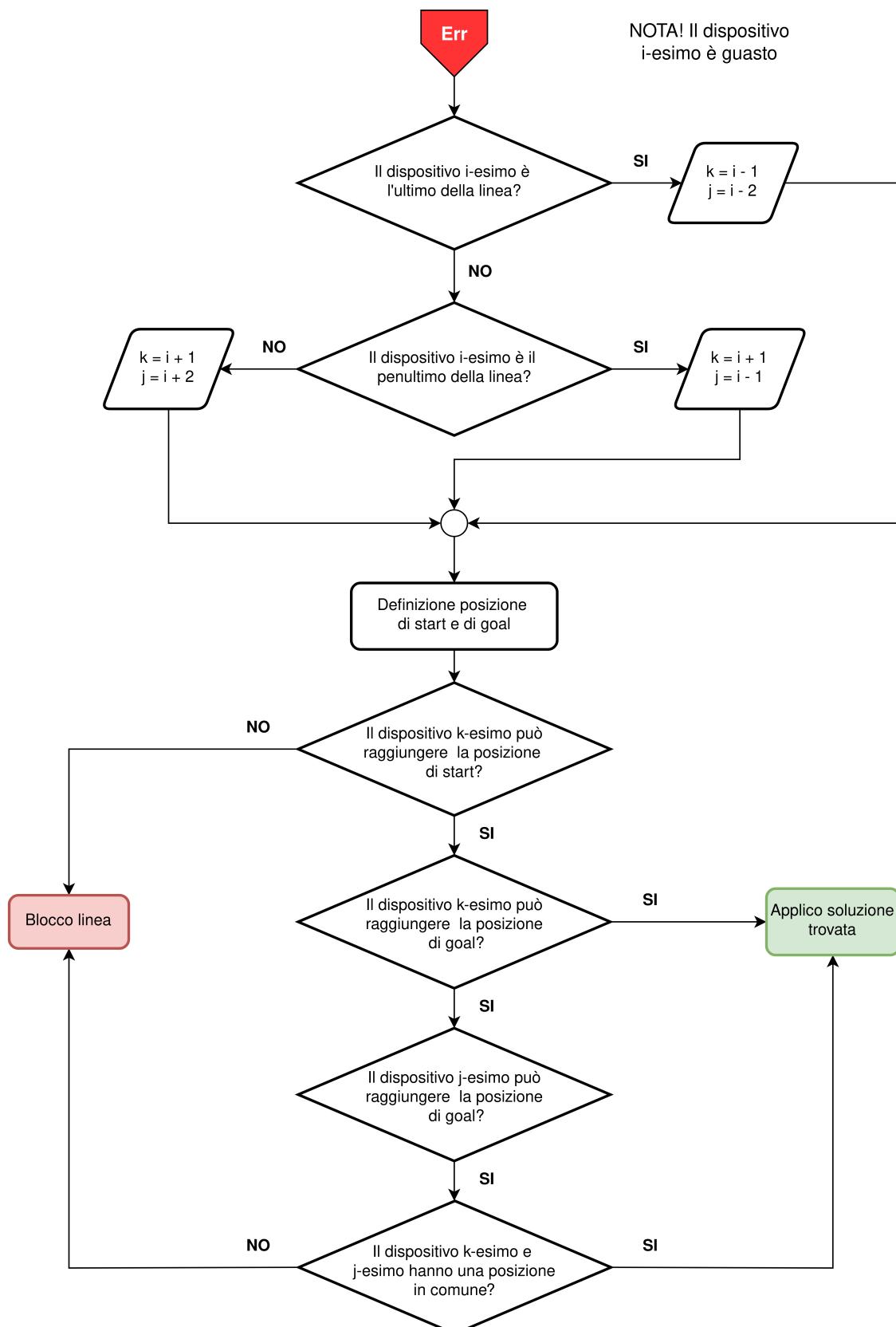


Figura 4.9: Diagramma di Flusso per la Ricerca della Soluzione all'Errore di Componente Non Arrivato

4.3.3 Gestione Errore di Assemblaggio

L'errore di assemblaggio si verifica quando una delle operazioni di montaggio del prodotto non produce il risultato aspettato:

- Un robot non inserisce un *elemento interno* nel *case*.
- Un robot inserisce un *elemento interno* del colore errato.

Il meccanismo di comunicazione dell'errore è identico a quanto già descritto nelle *Sezioni* precedenti. Si deve notare che il dispositivo potenzialmente guasto che genera questo errore sarà sempre un robot. Inoltre, alle ipotesi considerate per la ricerca della soluzione illustrate nella *Sezione 4.3.2*, ne vengono aggiunte altre due:

1. Solo i manipolatori possono risolvere un assemblaggio errato.
2. Se sono presenti *elementi interni* di colore diverso rispetto al codice colore generato, allora presumibilmente l'errore era nel magazzino e, di conseguenza, il robot potenzialmente responsabile dell'errore può essere impiegato nella ricerca della soluzione.

Il diagramma di flusso che rappresenta la gestione dell'errore (*Figura 4.10*), dopo la prima verifica, si scomponete in due rami differenti:

- Al prodotto mancano solamente uno o più elementi, quindi è necessario inserire gli *elementi interni* del colore corretto. Il primo controllo di questa parte riguarda, ancora una volta, la posizione del dispositivo guasto (i-esimo) nella linea. Se è l'ultimo, viene selezionato il primo robot precedente all'i-esimo, altrimenti verrà utilizzato il primo robot successivo. Viene definita la posizione nel magazzino dove è presente l'*elemento* del colore mancante e la posizione di rilascio per l'inserimento nel *case*. Infine, viene accertato se il robot prescelto può raggiungere tutte le posizioni e, in caso affermativo, viene applicata la soluzione; diversamente non è stata trovata alcuna soluzione.
- I colori degli *elementi* non rispettano le posizioni determinate dal codice colore generato. Le operazioni risolutive sono più complesse, in quanto è necessario utilizzare due robot: uno addetto alla rimozione e all'inserimento degli elementi, un altro che svolge la funzione di aiutante, ovvero mantiene in posizione il prodotto durante tutte le fasi del procedimento. Il robot aiutante serve principalmente nelle operazioni di rimozione per evitare che piccole imprecisioni

4.3. LINEA DI PRODUZIONE ADATTIVA MULTI-PRODOTTO

nel movimento dell'altro robot, producono uno spostamento del prodotto che comprometterebbe la riuscita dell'azione correttiva. In ingresso di questo ramo vengono definiti gli *elementi* da rimuovere. Per la scelta dei robot si presentano due situazioni: la prima si ha quando il dispositivo i-esimo è l'ultimo, quindi il robot operativo viene scelto tra i successivi all'i-esimo con questo compreso (ovviamente in questo caso la scelta ricadrà sempre sull'ultimo robot), mentre il robot aiutante viene cercato tra i robot precedenti senza considerare l'i-esimo; la seconda si ha quando il dispositivo i-esimo non è l'ultimo, quindi la ricerca dei due robot avviene allo stesso modo, ma l'inclusione o meno del robot è invertita. Tutta la parte successiva, si presenta con lo stesso schema di definizione delle posizioni e verifica che queste siano raggiungibili dal robot preposto. Se tutti i controlli vanno a buon fine viene applicata la soluzione, altrimenti non è stata trovata una soluzione per la correzione dell'errore.

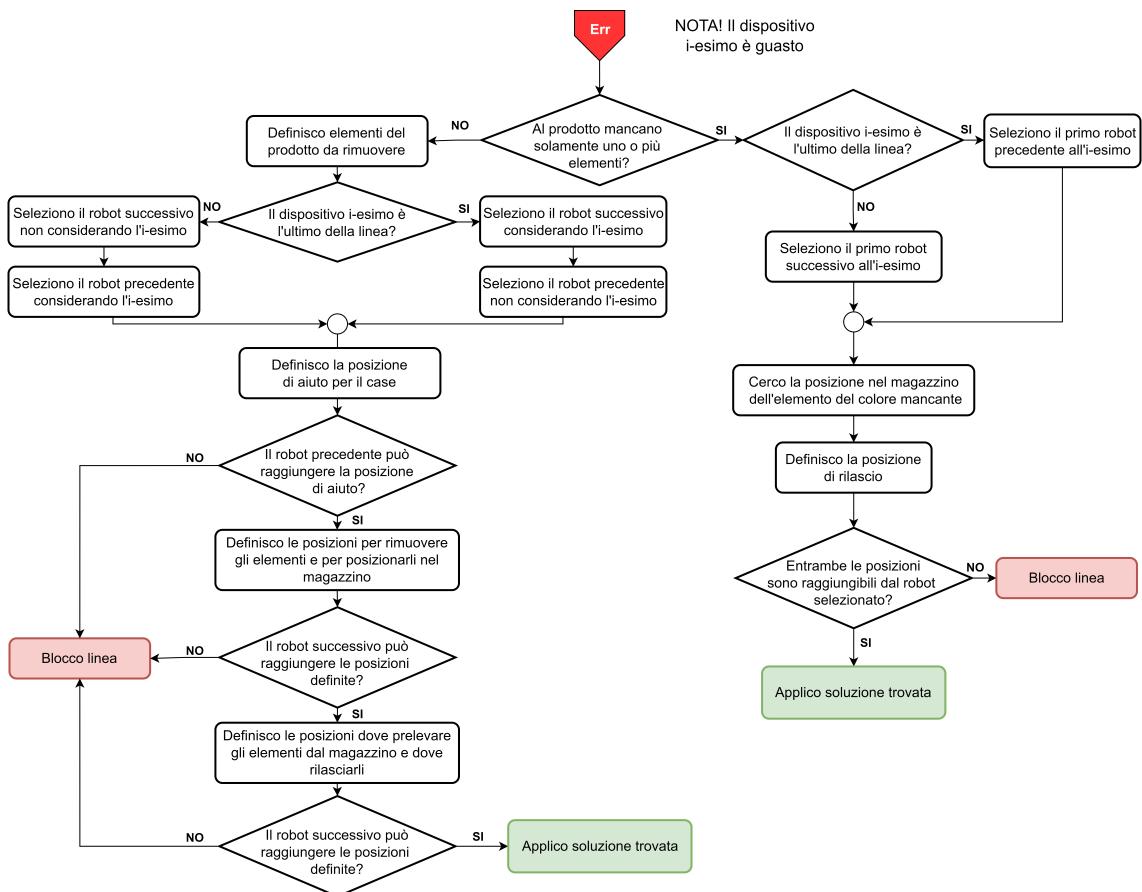


Figura 4.10: Diagramma di Flusso per la Ricerca della Soluzione all'Errore di Assemblaggio

Capitolo 5

Esperimenti

Dopo aver implementato l’architettura di controllo per rendere una linea di produzione robotica adattativa, è essenziale testarla per verificare il suo corretto funzionamento e le sue prestazioni. Da ciò, nasce l’esigenza di realizzare degli esperimenti, sia per la linea con un comportamento classico (*Sezione 5.1*), sia per quella con un comportamento adattivo (*Sezione 5.2*). Gli esperimenti¹ raccolti in questo *Capitolo* non saranno esaustivi rispetto alle molitudine di errori possibili, ma copriranno i principali eventi che si possono verificare nella gestione degli errori, sufficienti a dimostrare il comportamento di una linea adattiva all’insorgere di un problema. Gli errori sono stati volutamente attivati via codice (*errore di rallentamento* e *errore di componente non arrivato*) oppure agendo direttamente nel mondo fisico (*errore di assemblaggio*), in modo da simulare l’effetto di un possibile guasto di una linea reale.

¹Repo GitHub del codice degli esperimenti :

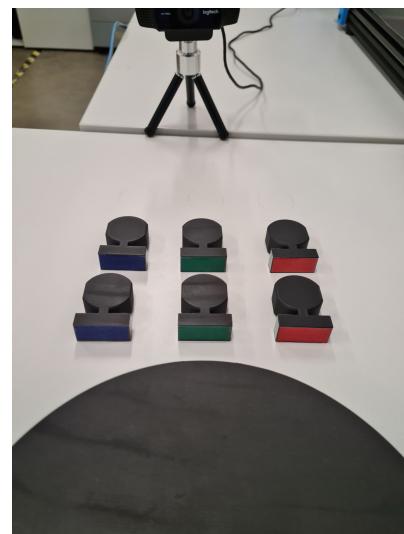
- FlexBE Behaviors e State: https://github.com/LucaCristuibGrizzi/production_line_behaviors;
- Nodi device: https://github.com/LucaCristuibGrizzi/production_line_device;
- Tavola Rotante: https://github.com/LucaCristuibGrizzi/rotating_table.

Video degli esperimenti disponibili a : <https://doi.org/10.5281/zenodo.10210928>

Una caratteristica comune alla maggior parte degli esperimenti è la disposizione dei magazzini della linea. Le eccezioni saranno mostrate nel momento opportuno. Il *magazzino 1* (*Figura 5.1(a)*) ha tre *case* e un *elemento interno* per colore, che seguono l'ordine rosso, verde e blu (da destra a sinistra). Il *magazzino 2* (*Figura 5.1(b)*) contiene solamente *elementi interni*, in particolare due per colore e segue lo stesso ordine del *magazzino 1*.



(a) Magazzino 1



(b) Magazzino 2

Figura 5.1: Disposizione degli Elementi Interni nei Magazzini in Condizioni Nominali

5.1 Linea di Produzione a Singolo Prodotto

Il primo esperimento ha riguardato il funzionamento in condizioni nominali della linea di produzione per il singolo prodotto. La sua architettura di controllo (*Sezione 4.2*) esegue i comandi in modo sequenziale: durante la fase di *test*, si è verificato questo comportamento, controllando che il processo di assemblaggio segua perfettamente le operazioni desiderate. In aggiunta, con un secondo esperimento, è stato riprodotto un errore di assemblaggio, agendo sulla posizione degli *elementi* nei magazzini. La linea a singolo prodotto non implementa una gestione degli errori, quindi gli effetti sulla linea di questo errore sono estendibili a qualsiasi altro tipo di errore. Questo esperimento è stato utilizzato come metro di confronto con la linea adattativa.

Simulando un errore nel *magazzino 1*, disponendo gli *elementi* in modo errato come mostrato in *Figura 5.2*, il sistema non è consapevole del differente assetto dei magazzini. L’assemblaggio funziona correttamente fino al primo controllo della disposizione colori, in cui il sistema si accorge che il *robot 1* ha inserito il primo *elemento interno* in modo non corretto, perché nel *magazzino 1* la posizione di quello blu e quello verde era invertita. La linea, non essendo in grado di gestire l’errore, si blocca in attesa di un intervento umano.



Figura 5.2: Differente Disposizione del Magazzino 1

5.2 Linea di Produzione Parallelia Multi-Prodotto

Per testare il parallelismo delle operazioni e la gestione degli errori previsti per la linea di produzione adattativa sono stati necessari quattro esperimenti. Il primo di questi prevede di eseguire il processo produttivo di tre prodotti in modo nominale, ovvero senza errori. La maggiore differenza che si può riscontrare con la linea a singolo prodotto è all'arrivo alla stazione di lavoro situata alla fine del *nastro trasportatore*. Infatti, in quell'istante inizia l'operazione del *robot 2*, come nel caso precedente, ma, questa volta, anche il *robot 1* esegue un nuovo *task*. Questo parallelismo permane per tutto il periodo di esecuzione della linea. Come ci si può aspettare, il processo di assemblaggio procede fino a conclusione di tutti e tre i prodotti seguendo le indicazioni definite in fase di progettazione.

5.2.1 Errore di Rallentamento

Il secondo esperimento verifica il comportamento della linea a fronte della necessità di un dispositivo di rallentare la sua velocità di lavorazione. Nessuno dei *device* della linea è dotato di strumenti per misurare la propria velocità; inoltre, anche nel caso in cui ci fosse stata la possibilità di valutarla, ripetere l'esperimento fino a quando uno dei dispositivi, avrebbe rallentato in seguito a un guasto, non era praticabile. Per questi motivi è stato definito via codice il dispositivo che lancerà l'errore, il momento in cui dovrà farlo e la nuova velocità da impostare, simulando la capacità di un sistema di fare "*auto-diagnostica*".

Nel caso in questione, il *robot 2*, alla prima chiamata, comunica il suo rallentamento a una velocità del 50% rispetto a quella nominale. Tutti i dispositivi ricevono e impostano la nuova velocità per sincronizzarsi. Un limite dei robot *Niryo* è l'impossibilità di eseguire contemporaneamente due istruzioni. La ripercussione sulla linea è che un manipolatore che riceve la notifica di un errore mentre sta eseguendo un'operazione, deve prima terminare quest'ultima e poi procedere con le attività di gestione dell'errore. Per il rallentamento può accadere, ed è successo in uno degli esperimenti, che un manipolatore a cui è stato detto di iniziare un'operazione un istante prima di ricevere la nuova velocità, la aggiorni solamente al *task* successivo. Si può concludere che l'assestamento alla velocità di errore avviene con una tolleranza di un'operazione. Questo è però dovuto all'*hardware* impiegato e non al sistema generale. La linea di produzione procede e completa le sue operazioni con la nuova velocità fino a nuova comunicazione.

5.2.2 Errore di Componente Non Arrivato

Il terzo esperimento emula l'*errore di componente non arrivato*: il prodotto, nel passaggio tra due stazioni di lavoro successive, non arriva a destinazione. Per generare questo errore viene imposto ad un dispositivo di non muoversi, in modo tale da generare l'errore.

Per la risoluzione di questo errore assume fondamentale importanza la ridondanza del *layout* della linea di produzione. Infatti, se non esistesse una seconda modalità con cui trasferire il prodotto tra due stazioni di lavoro, allora non sarebbe possibile portare a termine l'operazione. Nella linea sono presenti sia zone non ridondanti, sia altre che lo sono. Diventa significativo mostrare nell'esperimento la gestione di entrambe le situazioni.

Il primo contesto si ottiene simulando un guasto temporaneo al *nastro trasportatore*: il prodotto con il primo elemento inserito rimarrà bloccato all'inizio di questo dispositivo. Passato il tempo limite entro il quale il *sensore IR* deve rilevare l'arrivo del prodotto nella seconda stazione, viene lanciato un errore per il quale non esiste soluzione. A questo punto la *FSM* entra nello stato *User Solution State* e nel desktop appare la finestra in *Figura 5.3*. Se l'utente trova una soluzione all'errore indicato, risponde "Yes" e la linea riprende dal comando che non aveva funzionato; altrimenti, risponde "No" e la linea si blocca. Si deve notare che, durante questi passaggi di gestione dell'errore, la linea è stata inserita in uno stato di pausa. Nell'esperimento viene considerato che l'utente sia stato in grado di riparare il guasto al *nastro trasportatore*, quindi la linea può riprendere con quest'ultimo dispositivo che finalmente porta il prodotto alla seconda stazione.

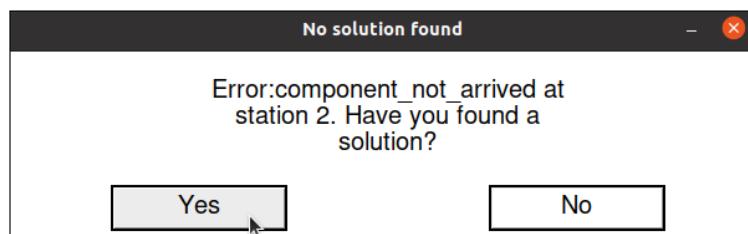


Figura 5.3: Finestra per la Risoluzione dell'Errore da Parte dell'Utente

Il secondo contesto si ricava riproducendo un guasto alla *tavola rotante*: il prodotto rimane fermo alla stazione del primo controllo dell'assemblaggio. In questo caso, la soluzione per portare il prodotto sul lato opposto della *tavola* esiste: il *robot 3* esegue un *pick & place* tra le due postazioni.

5.2.3 Errore Di Assemblaggio

L'ultimo esperimento riproduce la gestione dell'errore più complesso da risolvere, ovvero l'*errore di assemblaggio*. A differenza dei due errori precedenti, questa volta l'errore si ottiene andando a modificare la disposizione del *magazzino 2* (*Figura 5.4*), dove i due *elementi interni* verdi sono traslati indietro di una posizione per far spazio a un aggiuntivo *elemento* rosso. Nella trattazione successiva, non viene descritta la circostanza in cui l'algoritmo di gestione dell'errore non trova una soluzione, perché sopravviene la stessa finestra per l'interazione con l'utente vista nella *Sezione 5.2.2*. Prima di illustrare, il comportamento della linea, si deve ricordare che l'errore si divide in due tipologie: mancanza di un *elemento interno* oppure errata configurazione del codice colore.

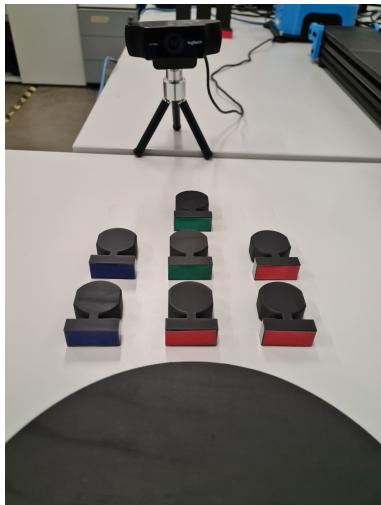


Figura 5.4: Differente Disposizione del Magazzino 2

Si considerino come codice colori generati, riportatati in ordine del prodotto da assemblare, (B,G,R), (R,B,G) e (G,R,B). Il primo prodotto arriva alla terza stazione di lavoro, dove viene controllato il rispetto del codice colore; non è rispettato ed è necessario rimuovere il secondo *elemento* in quanto è rosso e non verde. La linea entra in uno stato di pausa, il manipolatore assistente (il *robot 2*), porta il prodotto in una posizione comoda al *robot 3* per sfilare l'*elemento rosso* e inserire quello verde. Il *robot 3* ripone il componente rimosso in una posizione libera del *magazzino 2* tra quelle previste per gli *elementi interni* di colore rosso. L'operazione si conclude con il *robot 2* che riporta il prodotto corretto nella terza postazione di lavoro, da dove la linea riprende il suo funzionamento nominale. Per il secondo prodotto, il *robot 3* dovrebbe inserire l'*elemento interno* verde, ma nella posizione a lui comunicata

5.2. LINEA DI PRODUZIONE PARALLELA MULTI-PRODOTTO

precedentemente non è presente nessun articolo. Al successivo controllo, questo genera un errore in quanto manca il terzo *elemento*. Questo errore viene risolto sfruttando il *robot 2* per inserirlo.

In conclusione, si deve osservare che alcuni movimenti dei robot nella fase di smontaggio sono inconsueti. Questi sono dovuti alla struttura meccanica dei robot e al poco spazio a disposizione per cambiare posa. Inoltre, il secondo errore è generato dal fatto che al *robot 3* non viene comunicato che la posizione a lui assegnata per prelevare l'*elemento interno* verde è ora vuota e di aggiornarsi a una nuova. Questo è fatto volontariamente per testare in un unico esperimento entrambe le opzioni dell'errore di assemblaggio.

Capitolo 6

Conclusione e Sviluppi Futuri

La crescente necessità di inserire i principi di *Industria 4.0* nella manifattura del giorno d'oggi e di creare nuovi modelli produttivi per il futuro hanno ispirato la realizzazione di questo lavoro di tesi. Più precisamente, è stata realizzato un modello in scala di una linea di produzione ed è stata implementata un'architettura di controllo per renderla adattativa. Questa architettura concretizza parte del nuovo paradigma industriale, noto come *adaptive manufacturing*.

La linea di produzione robotica adattativa sviluppata ha mostrato fin da subito evidenti vantaggi rispetto a una linea classica. Molto interessante è l'architettura del controllo che realizza la gestione degli errori. Infatti, è stata progettata per essere generica e modulare, perciò ha la possibilità di adattarsi e/o di essere adattata al controllo di una qualsiasi linea di produzione. Tuttavia, è doveroso anche sottolineare alcuni limiti che sono stati riscontrati. Tra questi il più evidente è sicuramente l'*hardware* utilizzato; nonostante giochino a suo favore la facilità di controllo e le sue dimensioni ridotte, che hanno consentito di creare una linea in spazi minimi, di contro alcune sue caratteristiche hanno frenato alcune operazioni realizzabili dalla linea. Inoltre, la sua scadente precisione in alcune azioni, frequentemente ha portato durante lo sviluppo a distogliere l'attenzione dal reale obiettivo della tesi.

I benefici che si possono ottenere sono talmente significativi che è essenziale continuare a ricercare nuove soluzioni in questo campo, malgrado le difficoltà. Anzi da queste è possibile ricavare spunti di miglioramento per il futuro. Il miglioramento dell'*hardware* o, anche, lo studio di una nuova applicazione che nasconde i suoi difetti è uno dei primi passi che è possibile percorre. Un altro è la creazione di una linea di produzione più complessa che consenta di testare e sfruttare tutte le possibilità offerte dall'architettura implementata. Di notevole interesse sarebbe anche

l'applicazione in un caso reale, in quanto questo potrebbe permettere di eseguire un'analisi sulle *performance* con dei riscontri numerici. Anche l'aggiunta di nuovi errori e, di conseguenza, di nuovi algoritmi risolutivi è una sfida avvincente, che potrebbe rendere la linea adattativa ancor più conveniente. Infine, l'integrazione tra l'uso di algoritmi tradizionali e l'*AI* per la ricerca di soluzioni per errori che possono sopravvenire in una linea potrebbe aprire a nuovi orizzonti ancor più vantaggiosi.

Ringraziamenti

In primis, un ringraziamento speciale ai miei relatori, il Prof. Cristian Secchi e il Dott. Ing. Davide Ferrari, per i loro indispensabili consigli, per la loro disponibilità, per le loro conoscenze trasmesse durante tutto il percorso di realizzazione del progetto.

Un ringraziamento particolare va a tutte le persone che ho conosciuto durante il mio percorso universitario con cui abbiamo condiviso una risata con cui alleggerire i momenti più difficili.

Infine, ringrazio la mia famiglia per avermi supportato e sopportato in questi cinque anni.

Appendice A

Guida Per Settaggio e Uso Della Linea di Produzione

Quest'appendice è stata pensata per chi in futuro vorrà utilizzare e/o ricreare la linea di produzione di questo elaborato. Verranno riportati consigli per evitare alcuni problemi riscontrati durante questo lavoro e norme di buona pratica per il corretto utilizzo della linea. Inoltre, verranno indicati i *link* da dove è possibile scaricare il codice sviluppato per l'implementazione delle architetture di controllo viste in precedenza.

Setup Pacchetti ROS, Librerie Python e .bashrc

Per iniziare la parte più semplice, ossia l'installazione della libreria *PyNiryo*; attualmente, l'unica versione compatibile con entrambe le *release* della *Niryo ROS Stack* presente su *Ned* e *Ned2* è la versione 1.1.2. Essendo una libreria *Python*, è facilmente scaricabile utilizzando il comando standard del pacchetto *pip*:

```
pip install pyniryo==1.1.2
```

Per installare *FlexBE* è possibile seguire le indicazioni nel loro sito ufficiale¹ oppure quelle presenti nella *repository* di *GitHub*². In quest'ultimo caso è importante selezionare la *branch* corrispondente alla *release* di *ROS* impiegata (per questo progetto è *ROS Noetic*). Infine, nel *workspace* si devono inserire i due pacchetti *ROS*: uno che contiene gli stati e i comportamenti di *FlexBE* della linea³; l'altro in cui sono presenti i nodi per controllare i singoli dispositivi⁴. È probabile che *FlexBE* non rilevi immediatamente le informazioni dei nuovi pacchetti installati, una prima soluzione può essere di aprire la *tab "Configuration"* dell'interfaccia grafica dell'applicazione e cliccare sul pulsante "*Force Discover*".

La parte più complessa riguarda il *setup* della *Raspberry Pi*. La prima operazione è quella di installare e inizializzare l'*OS Ubuntu Server 20.04 LTS*⁵. La connessione via *Wi-Fi* non è garantita alla prima accensione, quindi l'utilizzo del collegamento via *Ethernet* è consigliato. Inoltre, per il collegamento *ssh* la seguente forma si è rilevata meno problematica: `ssh <username>@<Raspberry Pi's IP address>`. Va osservato che è possibile utilizzare anche altri *OS* supportati da *ROS*. Successivamente, si esegue l'installazione di *ROS* con le procedure standard⁶. Infine, si deve inserire nel *workspace* il pacchetto che contiene il controllo della tavola rotante⁷.

¹Sito FlexBe: <http://philserver.bplaced.net/fbe/download.php>

²GitHub FlexBe: https://github.com/FlexBE/flexbe_behavior_engine/tree/noetic

³Codice GitHub: https://github.com/LucaCristuibGrizzi/production_line_behaviors

⁴Codice GitHub: https://github.com/LucaCristuibGrizzi/production_line_device

⁵Tutorial: <https://ubuntu.com/tutorials/how-to-install-ubuntu-on-your-raspberry-pi#1-overview>

⁶Tutorial: <https://medium.com/geekculture/robot-operating-system-installation-configuration-and-auto-startup-on-a-raspberry-pi-with-ubuntu-6eb8e4e1038e>

⁷Codice GitHub: https://github.com/LucaCristuibGrizzi/rotating_table

Come ultima operazione, fondamentale per eseguire *ROS* attraverso macchine multiple, consiste nell'inserire nel *.bashrc* del proprio *PC* e della *Raspberry Pi* i seguenti comandi:

```
export ROS_MASTER_URI=http://<ROS Master device's IP address>:11311  
export ROS_IP=<device's IP address>
```

Setup del Layout della Linea

Per far funzionare correttamente il codice scaricato da *GitHub* è opportuno riprodurre il *layout* della linea di produzione visto in *Sezione 4.1*. Questo non ha la necessità di essere duplicato in modo millimetrico, in quanto tutte le pose dei robot saranno da ridefinire. Il procedimento per procurarsi le pose può essere fatto in due modalità: sfruttando *Niryo Studio* oppure con l'utilizzo di un semplice *script* che permetta di muovere il robot in *jog mode* e leggere le sue posizioni di giunto. È importante mantenere una nomenclatura comune delle pose in modo tale da rendere possibile la ricerca e selezione di queste. Quella adottata è la seguente *Pose_Type_Info_Where*, nella quale:

- **Pose:** è il prefisso di tutti i nomi.
- **Type:** serve ad indicare la tipologia dell'elemento che il robot dovrà afferrare. Può essere "C" per il *case*, "E" per l'*elemento interno* oppure "E1", "E2" ed "E3" per l'*elemento* durante le fasi di smontaggio del prodotto.
- **Info:** è usato per aggiungere informazioni aggiuntive. Ad esempio, "Top" per indicare la posizione di rilascio per l'inserimento degli *elementi interni* nel *case*, oppure "Help" per indicare che è una posizione di assistenza per il disassemblaggio.
- **Where:** fornisce informazioni sulla zona del *layout* della linea. Per gli articoli nei magazzini viene seguito questo schema "*Wi_Color_Number*", dove *i* è il numero del magazzino, *Color* può essere "R", "G" e "B", ed è necessario solo per gli elementi colorati, infine *Number* serve a indicare la posizione nel magazzino, soprattutto nel caso di articoli che si ripetono. Per le posizioni delle stazioni di lavoro, viene utilizzata la numerazione dei dispositivi della linea, che va da 1 per il *robot 1* e arriva a 5 per il *robot 3* (alla sensoristica non

è associato un numero), quindi si ha " i_j ", dove i è il numero del dispositivo precedente e j è il numero di quello successivo. Ad esempio, per la stazione tra il *robot 2* e la *tavola rotante* sarà " 3_4 ".

Prima di utilizzare la linea è bene eseguire anche una calibrazione dei colori rilevati dalla *telecamera* e del *range* in cui il sensore ad infrarossi rileva gli oggetti. Per la telecamera si deve far particolare attenzione anche allo sfondo dell'immagine catturata. Infatti, è bene che questo non abbia colori che si possano confondere con quelli degli elementi del prodotto e che non abbia variazioni, in quanto comprometterebbero il funzionamento dello stato di *object detection*.

Infine, è importante sottolineare che la connessione tra *PC* e robot avviene via *Ethernet*, quindi si deve utilizzare uno *switch Ethernet*. Inoltre, ogni robot ha un indirizzo *IP* distinto che può essere modificato da *Niryo Studio*. Per il progetto sono stati usati i seguenti:

- *robot 1*: 164.254.200.200;
- *robot 2*: 164.254.200.201;
- *robot 3*: 164.254.200.202.

Anche l'indirizzo *IP* del *PC* per il collegamento con cavo di rete deve essere modificato. In particolare, su *Ubuntu* si deve disattivare la modalità *DHCP* (*Dynamic Host Configuration Protocol*) e fissare un indirizzo *IP* statico composto in questo modo: 164.254.200.205, con la parte ".205" che può essere sostituita con i valori desiderati, ma che renda l'indirizzo *IP* diverso da quelli utilizzati dai robot⁸.

⁸Tutorial: https://docs.niryo.com/applications/ned/v1.0.4/en/source/tutorials/setup_connect_ned_etherne.html

Uso Della Linea

Dopo aver realizzato il *setup* della linea, è ora possibile utilizzarla. Vengono riportati i passaggi da effettuare:

1. Eseguire questo comando in un *terminal* per aprire *FlexBE*:

```
roslaunch flexbe_app flexbe_full.launch
```

2. Aprire il collegamento via *ssh* con la tavola rotante e lanciare il nodo *ROS*.

Dopo il primo comando verrà chiesta la password della *Raspberry Pi*, che nel progetto è *ARSCONTROL* oppure quella indicata nel processo di installazione di *Ubuntu Server*. I comandi sono i seguenti:

```
ssh <username>@<Raspberry Pi's IP address>
rosrun rotating_table rotating_table.py
```

3. Lanciare tutti i nodi della linea. Ognuno dei successivi comandi deve essere eseguito in un *terminal* separato:

```
rosrun production_line_device robot1.py
rosrun production_line_device robot2.py
rosrun production_line_device robot3.py
rosrun production_line_device visionCheck.py
rosrun production_line_device userWindow.py
```

4. L'ultima operazione è quella di caricare il *behavior* della linea di produzione, i.e "*Production Line*", e avviare la sua esecuzione da *FlexBE*

Per comprendere come realizzare un errore vedere la documentazione dei nodi dei robot e della tavola rotante. Un'ultima nota, tutti i passaggi potrebbero essere riuniti in un unico *launch* file, ma nella fase di *testing* la modalità appena illustrata fornisce maggiore controllo.

Bibliografia

- [1] **Ford Corporate.** *The Moving Assembly Line And The Five-Dollar Workday.*
- [2] **Frederick Winslow Taylor.** *The principles of scientific management.* Harper e Brothers, 1991.
- [3] **Bill of Rights Institute.** *Henry Ford and Alfred P. Sloan: Industrialization and Competition.*
- [4] **Ford Corporate.** *The Model T.*
- [5] **David E. Nye.** *America's Assembly Line.* 1^a ed. The MIT Press, 2013.
- [6] **ROY B. HELFGOTT.** «America's Third Industrial Revolution». In: *Challenge* 29.5 (1986), pp. 41–46.
- [7] **Mahmut Arslan.** «The third industrial revolution and post-industrial society». In: *Insight Turkey* 1.4 (1999), pp. 123–129.
- [8] **Klaus Schwab.** *The Fourth Industrial Revolution.* World Economic Forum, 2016.
- [9] **David Gartman.** «Postmodernism; or, the Cultural Logic of Post-Fordism?» In: *The Sociological Quarterly* 39.1 (1998), pp. 119–137.
- [10] **Barry C. Brusso.** «50 Years of Industrial Automation [History]». In: *IEEE Industry Applications Magazine* 24.4 (2018), pp. 8–11.
- [11] **KUKA.** *The history of KUKA.*
- [12] **ROY B. HELFGOTT.** «America's Third Industrial Revolution». In: *Challenge* 29.5 (1986), pp. 41–46.
- [13] **INGENIEUR.de.** *Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution.*

BIBLIOGRAFIA

- [14] **Henning Kagermann et al.** *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group*. Forschungsunion, 2013.
- [15] **Wouter Dessein e Tano Santos.** «Adaptive Organizations». In: *Journal of Political Economy* 114.5 (2006), pp. 956–995.
- [16] **Marcello Pellicciari et al.** «Engineering method for adaptive manufacturing systems design». In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 3 (mag. 2009), pp. 81–91.
- [17] **B&R Automotion.** *Produzione Adattiva*.
- [18] **AV&R.** *What is Adaptive Manufacturing Exactly?*
- [19] **Logitech.** *C922 PRO HD STREAM WEBCAM*.
- [20] **ROS.** *ROS - Robotic Operating System*.
- [21] **Philipp Schillinger, Stefan Kohlbrecher e Oskar von Stryk.** «Human-Robot Collaborative High-Level Control with an Application to Rescue Robotics». In: *IEEE International Conference on Robotics and Automation*. Stockholm, Sweden, mag. 2016.
- [22] **Philipp Schillinger, Stefan Kohlbrecher e Oskar von Stryk.** *flexbe/Tutorials*.
- [23] **Niryo.** *Niryo Homepage*.
- [24] **Niryo.** *Ned User Manual*.
- [25] **Niryo.** *Ned2 User Manual*.
- [26] **Niryo.** *Niryo Conveyor Belt - User Manual*.
- [27] **SKF.** *608*.
- [28] **OPENCIRCUIT ITALIA.** *28BYJ-48 Motore passo-passo 5V 4 fasi 5 fili*.
- [29] **OPENCIRCUIT ITALIA.** *Modulo driver motore passo-passo ULN2003*.
- [30] **Raspberry Pi Foundation.** *Raspberry Pi 3 Model B+*.
- [31] **Markforged.** *Onyx*.
- [32] **Ubuntu.** *Get Ubuntu Server*.