Alma Mater Studiorum - Università di Bologna

SCUOLA DI INGEGNERIA

DIPARTIMENTO di
INGEGNERIA DELL' ENERGIA ELETTRICA E DELL' INFORMAZIONE
Guglielmo Marconi
DEI

CORSO DI LAUREA IN
INGEGNERIA MECCATRONICA

TESI DI LAUREA
in
Informatica Industriale

SVILUPPO DI UN'APPLICAZIONE DI ROBOTICA INDUSTRIALE

CANDIDATA

Sofia Chiarini

RELATORE

Chiar.mo Prof. Eugenio Faldella

CORRELATORE

Ing. Alan Sassi

Anno Accademico
2021/2022

Sessione II

A mio nonno

Indice

Elenco delle tabelle

Elenco dei listati

Elenco delle figure

Capitolo 1

Introduzione

1.1 Obiettivi

L'obiettivo del presente lavoro di tesi è quello di realizzare un' applicazione di pallettizzazione per diversi formati di prodotto. I prodotti dovranno essere rilevati dall'area di prelievo da un sistema di visione e successivamente depositati sul pallet. Dal pannello, l'operatore deve avere la possibilità di arrestare il robot in qualsiasi momento, rimuovere uno o più prodotti e riprendere il ciclo dalla prima posizione libera sul pallet. Deve essere inoltre possibile personalizzare il pallet e il tipo di formato del prodotto. I formati in questione sono due: un primo formato standard, completamente bianco, e un secondo formato diviso in due parti, una chiara e una scura. Quest'ultimo dovrà essere posizionato alternando le due parti una sull'altra.

1.2 Organizzazione dei contenuti

Il presente documento si articola in sei capitoli. Terminata l'iniziale introduzione sul contesto e gli obiettivi prefissati, si procede a riportare, nel secondo capitolo, lo sviluppo del software del braccio robotico. Il terzo capitolo è dedicato ad una breve presentazione del lavoro svolto con il sistema di visione. Nel quarto, si tratterà nel dettaglio lo sviluppo del software plc, focalizzandosi sullo scambio dati tra le varie unità. Nel quinto capitolo viene descritto il risultato dell'interfaccia grafica sviluppata. Nel sesto capitolo verranno esposti i risultati ottenuti e i possibili sviluppi futuri.

Capitolo 2

Il software plc

I principi dell' automazione industriale suggeriscono l'utilizzo di sistemi flessibili, capaci di gestire con facilità i vari componenti di un impianto e di permetterne eventuali modifiche durante il funzionamento, riducendo al minimo l'impatto economico e il tempo d'intervento. Questa è la filosofia cardine sulla base della quale è stato ideato e sviluppato il PLC. In questo capitolo, dopo una breve introduzione ai PLC di marca Siemens, verranno illustrati i vari blocchi di programma e le strutture dati che costituiscono il software del controllore.

Il PLC utilizzato nell'applicazione oggetto della trattazione è prodotto dalla Siemens AG con il codice S7-1500. La programmazione è stata realizzata mediante il software TIA Portal v.16, fornito dalla casa produttrice, necessario per interfacciarsi con il PLC ed eseguire il caricamento dei programmi in memoria. Numerose altre funzionalità sono disponibili, quali ad esempio la registrazione dei valori assunti da ingressi e uscite, nonché da ogni variabile presente nella memoria del controllore.

2.1 La configurazione hardware

La configurazione hardware in TIA Portal comprende la configurazione dei dispositivi, composta da hardware dei sistemi di automazione, apparecchiature da campo intelligenti e hardware per la visualizzazione. Nella *Vista di rete* è possibile stabilire la comunicazione tra i diversi componenti, i quali vengono prelevati dai cataloghi e inseriti nella configurazione hardware. Per i dispositivi non appartenenti alla famiglia Siemens è prima necessario importare i GSDML, file forniti nel linguaggio GSDML che contengono le proprietà specifiche dell'apparecchiatura di campo.

2.2 Il protocollo Profinet

Il protocollo di comunicazione standard per i controllori Siemens è il Profinet. Basato sulla tecnologia Ethernet, nasce come evoluzione del Profibus e permette di collegare i dispositivi di campo ai sistemi gestionali dell'azienda, consentendo una comunicazione in tempi dell'ordine del millisecondo. Il Profinet

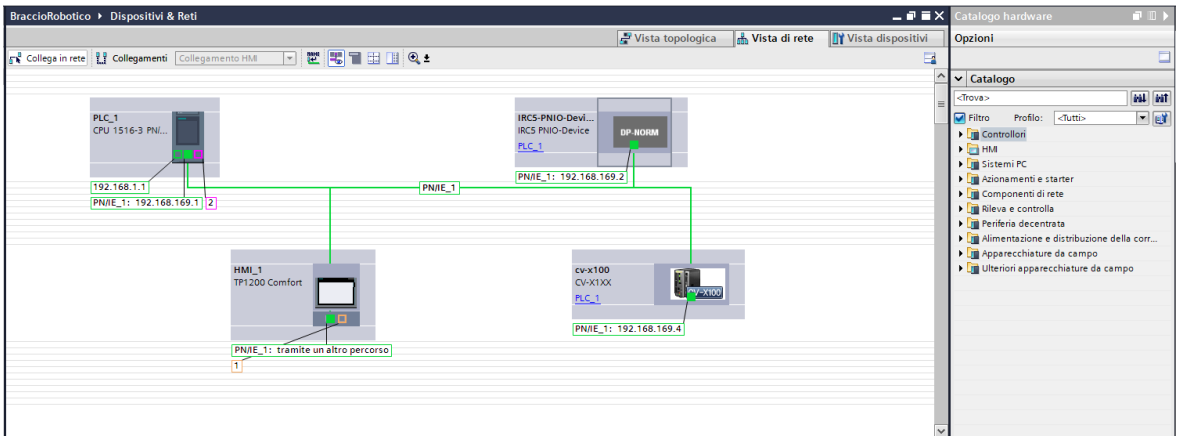


Figura 2.1: La configurazione hardware

utilizza il protocollo TCP/IP: la comunicazione con i dispositivi di campo avviene tramite lutilizzo di indirizzi IP e di Subnet Mask. Esso permette di collegare tra loro fino a 1024 dispositivi con una velocità di trasmissione di 100 Mbit/s, senza limiti di distanza. Queste caratteristiche rendono oggi il Profinet il protocollo più utilizzato nella gestione della comunicazione tra i dispositivi di un impianto e il PLC.[1]

2.3 Lo scambio dati

Il ruolo principale del controllore all'interno del sistema in oggetto è la gestione della comunicazione tra le diverse unità.

Il plc scambia con il robot 124 bytes in ingresso e 124 bytes in uscita. La memoria in ingesso, mappata a partire dall'indirizzo %I1000.0, è salvata nella variabile RobotInput di tipo typeRobotInput. Allo stesso modo, i dati in uscita sono scritti sulla variabile RobotInput di tipo typeRobotOutput a partire dall'indirizzo %Q1000.0.

RobotInput

Varibile	Tipo	Descrizione
commands	typeRobotCommands	robot System Input
interrupts	typeRobotInterrupts	robot interrupts
toolJudgeValue	Array[0..15] of Bool	vision System Tool Judge
resultReadyFlag	Bool	if TRUE, result can be read
commandReadyFlag	Bool	if TRUE, cv-x can execute a new command
runMode	Bool	if TRUE, cv-x is in run mode
commandCompleteFlag	Bool	if TRUE, command is completed
speedRef	UInt	robot speed set by HMI
removedProd	UInt	removed product index
index	UInt	index
mission	UInt	reserved
productsNum	UInt	products number
productsType	UInt	products type
pickOffs	UInt	pick z offset
placeOffs	UInt	place z offset
palletX	UInt	pallet x coordinate
palletY	UInt	pallet y coordinate
palletZ	UInt	pallet z coordinate
palletR	UInt	pallet rotation (0-45)
coordinates	typeCoordinates	product's coordinates in robot frame
commandResult	DInt	command's result
commandData	Array[1..2] of DInt	command's result data

Tabella 2.1: Dati in ingresso al robot

RobotOutput

Varibile	Tipo	Descrizione
worldzone	typeRobotWorldZone	robot worldzones
motorOnState	Bool	if TRUE, motors are in motor On state
emergencyStop	Bool	if TRUE, robot is in emergency stop state
executionError	Bool	if TRUE, an execution error occurred
isGripperOpen	Bool	if TRUE, gripper is opened
trgControl	Bool	if TRUE, robot requests a vision system trig
resultAckFlag	Bool	if TRUE, robot has received the vision system result
isReady	Bool	if TRUE, robot and vision system are inizialized
wProdNotFound	Bool	if TRUE, there is no product to pick
autoOn	Bool	if TRUE, robot is in auto mode
cycleOn	Bool	if TRUE, robot program is cycling
notMoving	Bool	if TRUE, robot is not moving
limitSpeed	Bool	if TRUE, robot speed is limited
commandRequestFlag	Bool	if TRUE, robot is sending a vision system command
alarm	Bool	if TRUE, a robot error occurred
isGroup1	Bool	if TRUE, robot requests a products in range -90 +90
isPlaced	Bool	if TRUE, a new product is placed
isStored	Bool	if TRUE, a new product is stored
isGripperFull	Bool	if TRUE, gripper is full
actProcedure	UInt	current robot procedure
index	UInt	index
errorNumber	DInt	errorNumber
commandNumber	DInt	vision system command number)
commandParameter	Array[1..2] of DInt	vision system command parameters

Tabella 2.2: Dati in uscita dal robot

Al sistema di visione sono dedicati gli indirizzi a partire da %I0.0 e %Q0.0. Le strutture dati typeVisionSysInput e typeVisionSysOutput riprendono lo schema degli ingressi e delle uscite standard del produttore[5]. Le rispettive variabili PLC sono VisionSysInput e VisionSysOutput.

VisyionSystemOutput

Varibile	Tipo	Descrizione
commandCompleteFlag	Bool	if TRUE, cv-x command is completed
commandErrorFlag	Bool	if TRUE, cv-x error occurred
commandReadyFlag	Bool	if TRUE, cv-x can execute a new command
resultReadyFlag	Bool	if TRUE, cv-x result can be read
trgReadyStatus	Bool	if TRUE, cv-x is ready for a new trigCommand
trgAckStatus	Bool	if TRUE, cv-x has received the trigCommand
busy	Bool	if TRUE, cv-x is busy
error	Bool	if TRUE, cv-x is in error state
run	Bool	if TRUE, cv-x is in run state
toolJudgeValue	Array[0..63] of Bool	0=NG 1=OK
errorCode	Int	error code
totalCount	DInt	total trig count
commandResult	DInt	command result code
commandData	Array[1..6] of DInt	command data
resultData	Array[1..200] of DInt	tools result data

Tabella 2.3: Dati in uscita dal sistema di visione

VisionSystemInput

Varibile	Tipo	Descrizione
commandRequestFlag	Bool	if TRUE, a command is requested
resultAckFlag	Bool	if TRUE, result has been received
errorResetRequestFlag	Bool	if TRUE, error reset is requested
trgControl	Bool	if TRUE, a new trig is requested
reset	Bool	if TRUE, reset is requested
commandNumber	DInt	command number
commandParameter	Array[1..510] of DInt	command parameters

Tabella 2.4: Dati in ingresso al sistema di visione

2.4 Il linguaggio di programmazione

Tra i diversi linguaggi disponibili si è scelto di utilizzare l'SCL, un linguaggio di programmazione che segue lo standard DIN 61131-3 per i linguaggi di programmazione destinati ai controllori programmabili. Seguendo uno standard internazionale, risulta molto simile ai linguaggi strutturati di altri costruttori che seguono la stessa norma, il che permette di scrivere del codice riutilizzabile su tipologie di controllori diversi. È inoltre un linguaggio molto comodo quando si hanno algoritmi complessi, caratterizzati da una grande quantità di dati da gestire e con molte operazioni cicliche da effettuare.
[7]

2.5 La struttura del programma

Ogni progetto TIA Portal è costituito da blocchi che vengono richiamati all'interno di un blocco principale e/o di altri blocchi. I blocchi si possono classificare in tre categorie:

- **Blocchi Organizzativi (OB)**: costituiscono l'interfaccia tra il sistema operativo e il programma utente. Alcuni esempi sono l' OB1 , che esegue l' elaborazione ciclica del programma, e l' OB100, che descrive il comportamento del sistema di automazione all'avvio.
- **Funzioni (FC)**: blocchi programmabili adatti a realizzare logiche di funzionamento di impianto. Una FC contiene un programma che viene eseguito ogni qualvolta essa viene richiamata da un altro blocco di codice. Per ogni funzione è possibile dichiarare dei parametri di ingresso, di uscita, di ingresso/uscita, nonché delle variabili temporanee. Queste ultime vengono memorizzate nello stack dei dati locali e vanno perdute dopo lelaborazione dellFC.
- **Blocchi Funzionali (FB)**: blocchi programmabili simili alle FC, ma con la possibilità di definire al loro interno variabili con memoria.[6]

L'applicazione sviluppata è costituita dal solo blocco organizzativo OB1, richiamata dal s.o. ad ogni ciclo (scan). L'intervallo di tempo tra uno scan e il successivo non è predeterminato ma dipende dal tempo di esecuzione dell'OB1, al netto di eventuali interruzioni.

```
1
2     "Inputs"();
3     "AlarmsHandler"();
4     "UnitsHandler"();
5     "StateHandler"();
6     "RecipesHandler"();
7     "Comunication"();
8     "Outputs"();
```

Listato 2.1: OB1

In questa sezione sono illustrati i diversi blocchi, FC e FB, richiamati all'interno dell' OB1.

2.5.1 Il blocco FC Inputs

In questo blocco i dati provenienti dalla periferia sono salvati nelle rispettive variabili globali.

```
1
2 "RobotData".outputs        := "robotOutput";
3 "VisionSystemData".outputs := "visionSystemOutput";
```

Listato 2.2: Inputs()

2.5.2 Il blocco FC AlarmsHandler

Il blocco AlarmsHandler è adibito alla gestione degli allarmi e delle rispettive segnalazioni.
In presenza di un allarme proveniente dal robot o dal sistema di visione, il sistema si deve arrestare e deve essere visualizzata una segnalazione sul pannello operatore.
Tra le diverse modalità di gestione delle segnalazioni possibili in TIA Portal si è scelto di utilizzare l' FB Program_Alarm. Questa scelta è motivata dalla possibilità di visualizzare testi dinamici all'interno della segnalazione, velocizzando la fase di sviluppo e migliorando flessibilità del codice.

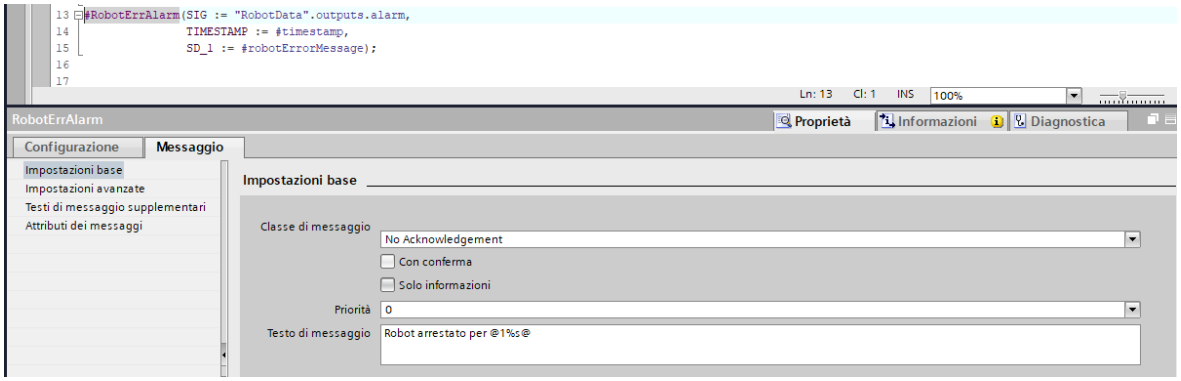


Figura 2.2: Utilizzo del Program_Alarm per la visualizzazione di segnalazioni dinamiche.

Gli allarmi da ABB

Gli allarmi provenienti da ABB sono segnalati dal fronte di salita del segnale "RobotData".Outputs.Alarm. Per aiutare la comprensione della segnalazione da parte dell'operatore, è stata sviluppata la funzione ErrNoToString(vedi Listato 2.3) per convertire il codice dell'errore in una stringa di testo per la segnalazione a pannello.

```
1
2  CASE #errNo OF
3      "ERR_OPEN_GRIPPER_TIMEOUT":
4          #errStrig := 'pinza non aperta entro il timeout';
5      "ERR_PROD_NOT_FOUND":
6          #errStrig := 'prodotto non rilevato in area di prelievo';
7      "ERR_RESULT_TIMEOUT":
8          #errStrig := 'risultato del sistema di visione non ricevuto entro il timeout';
9      "ERR_MOTION_INTERRUPTED":
10         #errStrig := 'moto interrotto';
11     "ERR_PICK_INTERRUPTED":
12         #errStrig := 'moto interrotto durante la fase di prelievo';
13     "ERR_PLACE_INTERRUPTED":
14         #errStrig := 'moto interrotto durante la fase di posizionamento';
15     "ERR_PRODS_REMOVED":
16         #errStrig := 'prodotti rimossi';
17     "ERR_STORE_INTERRUPTED":
18         #errStrig := 'moto interrotto durante la fase di deposito';
19
20     "ERR_COLLISION":
21         #errStrig := 'collisione';
22     ELSE
23         #errStrig := CONCAT(IN1 := DINT_TO_STRING((#errNo)), IN2 := ', leggere la
               documentazione ABB per ulteriori dettagli');
24 END_CASE;
```

Listato 2.3: La funzione ErrNoTostring()

Gli allarmi da CV-X

Per quanto riguarda gli allarmi del sistema di visione, viene attivata una segnalazione al fronte di salita del bit di errore. Nel testo viene riportato il codice di errore, per il quale si rimanda direttamente al manuale Keyence.

La diagnostica della rete Profinet

La diagnostica della rete Profinet è gestita con l'utilizzo del FB Siemens DeviceStates(), il quale consente di richiamare una determinata informazione di stato concernente tutte le unità in un sistema IO. Nel listato 2.4 è mostrata la diagnostica degli IO Device/slave DP disturbati della rete, identificata dalla costante "Local PROFINETIO-System". Il risultato è salvato all'interno dell'array di booleani profinetState.

```
1
2 #ret := DeviceStates(LADDR := "Local~PROFINET_IO-System", MODE := 2, STATE := #
      profinetState);
```

Listato 2.4: La diagnostica della rete Profinet

2.5.3 Il blocco FC UnitsHandler

In questa FC sono richiamati i blocchi per la gestione delle diverse unità: HMI, robot e sistema di visione.

Il blocco FC HMI

In questo blocco è gestita la visualizzazione dei prodotti sul pannello. A questo scopo è stato definito l'array di booleani "HMIData".prods, in cui ogni elemento corrisponde ad un prodotto attualmente depositato sul pallet. Ad ogni ciclo sono quindi calcolati la somma dei prodotti depositati e il livello attuale del pallet. In base al formato della ricetta attuale, i dati contenuti in "HMIData".prods vengono in seguito passati all'array "HMIData".pallet1, per la visualizzazione del primo formato, o "HMIData".pallet2, per la visualizzazione del secondo.

```
1
2  IF "StateData".state = "STATE_INIT" THEN
3
4      REGION palletVisualizationReset
5
6          "HMIData".prods := #inizializedPallet;
7          "HMIData".pallet1 := #inizializedPallet;
8          "HMIData".pallet2 := #inizializedPallet;
9          "HMIData".actLevel := 0;
10         "HMIData".actProdsCtn := 0;
11
12     END_REGION
13
14 END_IF;
15
16 IF "StateData".state = "STATE_RUNNING" THEN
17
18     REGION Pallet Visualization
19
20         // if a product is placed, save his value in prods array
21         REGION countProductPlaced
22
```

```
23              // count products and find last product
24              "HMIData".actProdsCtn := 0;
25
26              FOR #i := 1 TO "MAX_PRODUCTS" DO
27
28                  IF "HMIData".prods[#i] THEN
29
30                      "HMIData".actProdsCtn += 1;
31
32                      #lastprod := #i;
33
34                  END_IF;
35
36              END_FOR;
37
38          END_REGION
39
40          // find current level
41          "HMIData".actLevel := "CalcLevel"(prod := #lastprod, dim := 2);
42
43          REGION Update Products Placed
44
45              //if index is in range, update the corresponding array element
46
47              IF "RobotData".outputs.index >= 1 AND "RobotData".outputs.index <= 20 THEN
48
49                  IF "RobotData".outputs.isPlaced THEN
50
51                      //if a product is placed, increase array
52
53                      "HMIData".prods["RobotData".outputs.index] := TRUE;
54
55                  ELSIF "RobotData".outputs.isStored THEN
56
57                      //if a product is stored, reduce array
58
59                      "HMIData".prods["RobotData".outputs.index] := FALSE;
60                  END_IF;
61
62              END_IF;
63
64          END_REGION
65
66          REGION Remove Products
67
68          //If a product is removed by HMI, clear the corresponding array element
69              IF "HMIData".robotInputs.interrupts.removeProd THEN
70                  //check if index is in range!
71                  IF "HMIData".robotInputs.removedProd >= 1 AND "HMIData".robotInputs.
                        removedProd <= "MAX_PRODUCTS" THEN
72                  // if removed product isn't in the last level, remove all superior
                        products
73                      REGION Remove Superior Products
74                      //the start level to remove is the first one over the removed
                            product level
75                          #level := "CalcLevel"(prod := "HMIData".robotInputs.
                                removedProd, dim := "RobotData".inputs.word3);
76                          //ex: level=2 dim=2 => startprod=5
77                          #startprod := (#level * "RobotData".inputs.word3) + 1;
```

```
78                          //check if index is in range!
79                          IF #startprod >= 1 AND #startprod <= "MAX_PRODUCTS" THEN
80
81                              FOR #i := #startprod TO "MAX_PRODUCTS" DO
82                                  "HMIData".prods[#i] := FALSE;
83                              END_FOR;
84
85                          END_IF;
86
87                      END_REGION
88
89                      "HMIData".prods["HMIData".robotInputs.removedProd] := FALSE;
90
91                  END_IF;
92
93              END_IF;
94
95          END_REGION
96
97          CASE "RobotData".inputs.productsType OF
98              0:
99                  "HMIData".pallet1 := "HMIData".prods;
100
101             1:
102                 "HMIData".pallet2 := "HMIData".prods;
103         END_CASE;
104
105     END_REGION
```

Listato 2.5: HMI()

Il blocco FC VisionSys

In questo blocco è gestita la comunicazione con il sistema di visione.
Per ogni formato è presente un diverso programma all'interno del cv-x., il cui valore è salvato nella variabile "VisionSystemData".actProg. Quest'ultimo viene aggiornato ad ogni cambio o lettura del programma (vedi Listato 2.6).

```
1  REGION Save Current Vision System Program
2
3      //When vision system program is read or changed, save its number.
4
5      IF "VisionSystemData".outputs.commandCompleteFlag THEN
6
7          CASE "VisionSystemData".inputs.commandNumber OF
8
9              #CHANGE_PROGRAM:
10
11                 "VisionSystemData".actProg := "VisionSystemData".inputs.
                       commandParameter[2];
12
13             #READ_PROGRAM:
14
15                 "VisionSystemData".actProg := "VisionSystemData".outputs.commandData
                       [2];
16
17         END_CASE;
18
```

```
19      END_IF;
20
21  END_REGION
```

Listato 2.6: VisionSystem(): salvataggio del programma attuale

In base al valore di "VisionSystemData".actProg viene letta una diversa configurazione dei dati in ingresso.

```
1
2   REGION Read Vision System Outputs
3   //For each program number, read different data based
4   //on cv-x profinet outputs configuration
5       CASE "VisionSystemData".actProg OF
6
7           2: //interlocking
8               REGION VisionSystemProgramInterlocking
9
10                  "VisionSystemData".statistics.total := "VisionSystemData".outputs.
                        resultData[1];
11                  "VisionSystemData".statistics.OK := "VisionSystemData".outputs.
                        resultData[2];
12                  "VisionSystemData".statistics.NG := "VisionSystemData".outputs.
                        resultData[3];
13
14                  "VisionSystemData".executionTime := "VisionSystemData".outputs.
                        resultData[4];
15
16                  "VisionSystemData".prodsCtn := "VisionSystemData".outputs.resultData
                        [5];
17
18                  #resultDataIndex := 6;
19                  #arraySize := 8; // max number of detected products
20
21                  //Save coordinates(X,Y,Rz) from vision system data results outputs
22                  FOR #i := 0 TO #arraySize - 1 DO
23                      "VisionSystemData".arrayCoordinates[#i].x := "VisionSystemData".
                            outputs.resultData[#resultDataIndex];
24                      #resultDataIndex := #resultDataIndex + 1;
25                  END_FOR;
26
27                  FOR #i := 0 TO #arraySize - 1 DO
28                      "VisionSystemData".arrayCoordinates[#i].y := "VisionSystemData".
                            outputs.resultData[#resultDataIndex];
29                      #resultDataIndex := #resultDataIndex + 1;
30                  END_FOR;
31
32                  FOR #i := 0 TO #arraySize - 1 DO
33                      "VisionSystemData".arrayCoordinates[#i].angle := "VisionSystemData
                            ".outputs.resultData[#resultDataIndex];
34                      #resultDataIndex := #resultDataIndex + 1;
35                  END_FOR;
36
37              END_REGION
38
39          10: //palletization
40
41              REGION VisionSystemProgramPallettization
42
```

```
43                  "VisionSystemData".coordinates.x := "VisionSystemData".outputs.
                        resultData[1];
44                  "VisionSystemData".coordinates.y := "VisionSystemData".outputs.
                        resultData[2];
45                  "VisionSystemData".coordinates.angle := "VisionSystemData".outputs.
                        resultData[3];
46
47                  "VisionSystemData".statistics.total := "VisionSystemData".outputs.
                        resultData[4];
48                  "VisionSystemData".statistics.OK := "VisionSystemData".outputs.
                        resultData[5];
49                  "VisionSystemData".statistics.NG := "VisionSystemData".outputs.
                        resultData[6];
50
51                  "VisionSystemData".executionTime := "VisionSystemData".outputs.
                        resultData[7];
52
53              END_REGION
54          11:
55              REGION VisionSystemProgramCalibration
56
57                  "VisionSystemData".statistics.total := "VisionSystemData".outputs.
                        resultData[1];
58                  "VisionSystemData".statistics.OK := "VisionSystemData".outputs.
                        resultData[2];
59                  "VisionSystemData".statistics.NG := "VisionSystemData".outputs.
                        resultData[3];
60
61                  "VisionSystemData".executionTime := "VisionSystemData".outputs.
                        resultData[4];
62
63                  #resultDataIndex := 5;
64                  #arraySize := 5; // max number of detected products
65
66                  //Save coordinates(X,Y,Rz) from vision system data results outputs
67                  FOR #i := 0 TO #arraySize - 1 DO
68
69                      "VisionSystemData".arrayCoordinates[#i].x := "VisionSystemData".
                            outputs.resultData[#resultDataIndex];
70                      #resultDataIndex := #resultDataIndex + 1;
71
72                      "VisionSystemData".arrayCoordinates[#i].y := "VisionSystemData".
                            outputs.resultData[#resultDataIndex];
73                      #resultDataIndex := #resultDataIndex + 1;
74
75                  END_FOR;
76
77                  "VisionSystemData".prodsCtn := 5;
78
79              END_REGION
80
81      END_CASE;
82
83  END_REGION
```

Listato 2.7: VisionSystem(): lettura delle uscite del sistema di visione

Il blocco FC Robot

In questo blocco è gestita l'elaborazione dei dati dal sistema di visione al robot. Quest'ultimo può eseguire tre diverse procedure, definite "missioni". La prima missione corrisponde al primo formato di prodotto, per il quale sono sufficienti i dati in arrivo direttamente dal cv-x. Per il formato con incastro, invece, il sistema di visione invia i dati di tutti i prodotti riconosciuti. Il plc seleziona le coordinate del primo prodotto appartenente al gruppo richiesto oppure, in caso di ricerca fallita, invia al robot un giudizio negativo. Infine, per la procedura di calibrazione, descritta più nel dettaglio nel capitolo dedicato al software del robot, vengono inviati in sequenza i dati corrispondenti ai punti di calibrazione.

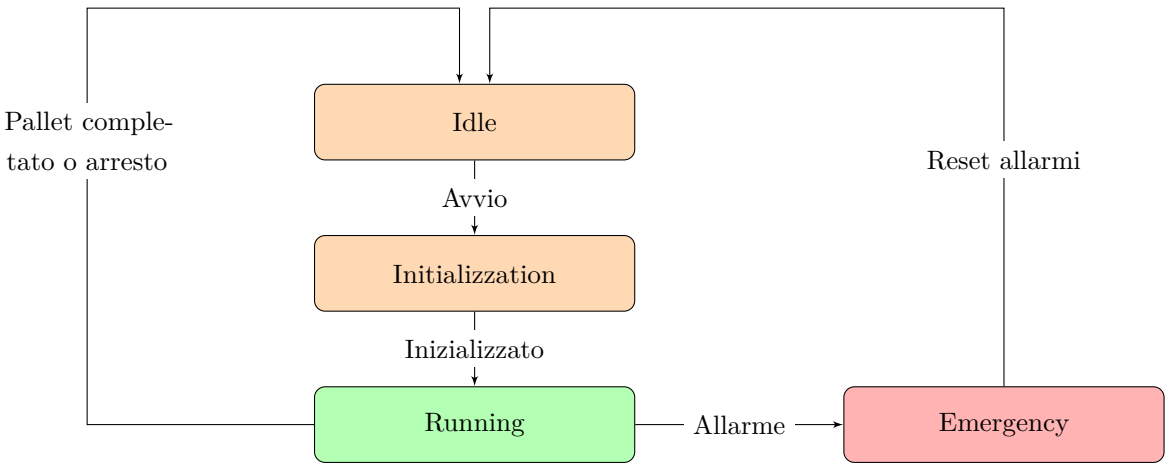
```
IF "StateData".state = "STATE_RUNNING" THEN

    CASE "RobotData".inputs.mission OF

        "PALLET_INTERLOCKING":

            REGION Interlocking

                //Build an interlocking pallet.
                //The products are divided in two groups based on
                //their angle (90-(-90))

                //If Vision system detect at least one products,
                //checks all products to find one

                IF "VisionSystemData".outputs.toolJudgeValue[0] THEN

                    REGION Select Pick Coordinates

                        //inizialize loop index and set judge value false.
                        //If no product is found, judge value will remain false,
                        //otherwise it will become true

                        #i := 0;
                        "VisionSystemData".outputs.toolJudgeValue[0] := false;

                        //loop to find the right product to pick
                        //loop ends when all products are checked or
                        //a right product is found

                        WHILE (#i < "VisionSystemData".prodsCtn AND NOT "
                            VisionSystemData".outputs.toolJudgeValue[0]) DO

                            //Check if a product belongs to the right group based on
                                its angle

                            #isGroup1 := NOT ("VisionSystemData".arrayCoordinates[#i].
                                angle / 1000 >= -90 AND "VisionSystemData".
                                arrayCoordinates[#i].angle / 1000 <= 90);

                            "VisionSystemData".outputs.toolJudgeValue[0] := ("
                                RobotData".outputs.isGroup1 = #isGroup1);


                            //if a right product is found, save coordinates for robot
                            IF "VisionSystemData".outputs.toolJudgeValue[0] THEN

```

```
43                                  "VisionSystemData".coordinates := "VisionSystemData".
                                        arrayCoordinates[#i];
44
45                              END_IF;
46
47                              #i := #i + 1;
48
49                          END_WHILE;
50
51                      END_REGION
52
53                  END_IF;
54
55              END_REGION
56
57          "CALIBRATION":
58
59              REGION Calibration
60
61                  IF "RobotData".outputs.index < "VisionSystemData".prodsCtn AND "
                        RobotData".outputs.index >= 0 THEN
62
63                      "VisionSystemData".coordinates := "VisionSystemData".
                            arrayCoordinates["RobotData".outputs.index];
64                      "RobotData".inputs.index := "RobotData".outputs.index;
65
66                  END_IF;
67
68              END_REGION
69
70      END_CASE;
71
72  END_IF;
```

Listato 2.8: Robot()

2.5.4 Il blocco FC StateHandler

In questo blocco è gestita la macchina a stati schematizzata nel grafico sottostante.



2.5.5 Il blocco FC RecipeHandler

Per rendere il sistema personalizzabile dal cliente finale sono state predisposte 20 ricette. Per la loro gestione è stato definito il tipo di dato typeRecipe con i seguenti valori:

- description: Wstring, descrizione della ricetta;
- palletX: Real, coordinata X del centro del pallet;
- palletY: Real, coordinata Y del centro del pallet;
- palletZ: Real, coordinata Z del centro del pallet;
- palletR: Real, rotazione del pallet;
- pickOffs: Uint, offset di approccio in prelievo;
- placeOffs: Uint, offset di approccio in deposito;
- productsNum: Uint, numero dei prodotti da pallettizzare;
- productsType: Uint, formato del prodotto;

Le ricette sono salvate nel controllore all'interno di un blocco dati con memoria a ritenzione. Quando il sistema non si trova in modalità automatica, l'operatore può selezionare la ricetta per il ciclo successivo.

```
REGION Recipe Active

    IF "Machine".Auto THEN

        "Hmi".Recipe.ActiveNumberFromHmi := "Hmi".Recipe.ActiveNumber;

    ELSE

        "Hmi".Recipe.ActiveNumber := "Hmi".Recipe.ActiveNumberFromHmi;

    END_IF;

    "RecipeActive" := "Recipes".Recipe["Hmi".Recipe.ActiveNumber];


END_REGION
```

Listato 2.9: RecipesHandler(): ricetta attiva

È sempre possibile selezionare ed editare le ricette. Quando una ricetta viene selezionata per l'editazione, i dati vengono caricati dall'archivio nella DB "RecipeEdit". Premendo il pulsante a pannello "Save", i dati della ricetta in editazione sono salvati nell'archivio.

```
REGION Edit to Recipes

    IF "Hmi".Recipe.Save THEN

        "Recipes".Recipe["Hmi".Recipe.EditNumber] := "RecipeEdit";
        "Hmi".Recipe.Save := false;

    END_IF;

END_REGION

REGION Recipes to Edit

    IF "Hmi".Recipe.EditNumber <> "Hmi".Recipe.EditNumberFromHmi THEN

```

```
16          "Hmi".Recipe.EditNumber := "Hmi".Recipe.EditNumberFromHmi;
17
18          "RecipeEdit" := "Recipes".Recipe["Hmi".Recipe.EditNumber];
19
20      END_IF;
21
22  END_REGION
```

Listato 2.10: RecipesHandler(): ricetta in editazione

L'operatore ha inoltre la possibilità di copiare una ricetta su un'altra. Se la ricette sorgente e destinazione corrispondo la funzione è disabilitata.

```
1
2
3   REGION Recipe Copy
4
5       "Hmi".Recipe.DestinationDescription := "Recipes".Recipe["Hmi".Recipe.
            DestinationNumber].description;
6       "Hmi".Recipe.SourceDescription := "Recipes".Recipe["Hmi".Recipe.SourceNumber].
            description;
7
8       "Hmi".Recipe.EnableCopy := "Hmi".Recipe.DestinationNumber <> "Hmi".Recipe.
            SourceNumber;
9
10      IF "Hmi".Recipe.EnableCopy AND "Hmi".Recipe.Copy THEN
11          "Recipes".Recipe["Hmi".Recipe.DestinationNumber] := "Recipes".Recipe["Hmi".
                Recipe.SourceNumber];
12          "Hmi".Recipe.Copy := false;
13      END_IF;
14  END_REGION
```

Listato 2.11: RecipeHandler(): copiatura ricette

2.5.6 Il blocco FC Comunication

In questo blocco vengono cambiati i dati tra le diverse unità.

```
1   REGION Write Robot Data
2
3       REGION From Vision System (@\textcolor{black}{to}@) Robot
4
5           //write robot values from Vision System
6
7           "RobotData".inputs.commandCompleteFlag := "VisionSystemData".outputs.
                commandCompleteFlag;
8           "RobotData".inputs.commandReadyFlag := "VisionSystemData".outputs.
                commandReadyFlag;
9
10          "RobotData".inputs.runMode := "VisionSystemData".outputs.run;
11
12          "RobotData".inputs.resultReadyFlag := "VisionSystemData".outputs.
                resultReadyFlag;
13
14          "RobotData".inputs.coordinates := "VisionSystemData".coordinates;
15
16          REGION CommandData
17              "RobotData".inputs.commandData[1] := "VisionSystemData".outputs.
                    commandData[1];
```

```
18              "RobotData".inputs.commandData[2] := "VisionSystemData".outputs.
                    commandData[2];
19          END_REGION
20
21          REGION ToolJudgeValues
22              "RobotData".inputs.toolJudgeValue[0] := "VisionSystemData".outputs.
                    toolJudgeValue[0];
23              "RobotData".inputs.toolJudgeValue[1] := "VisionSystemData".outputs.
                    toolJudgeValue[1];
24          END_REGION
25
26
27
28      END_REGION
29
30
31          //write robot values from HMI
32          "RobotData".inputs.commands := "HMIData".robotInputs.commands;
33
34          //value of robot program to run (check robot code for details)
35          "RobotData".inputs.mission := "HMIData".robotInputs.mission;
36
37          REGION interrupts
38              // write Interrupts
39              "RobotData".inputs.interrupts.startMove := "HMIData".robotInputs.
                    interrupts.startMove;
40              "RobotData".inputs.interrupts.stopMove := "HMIData".robotInputs.interrupts
                    .stopMove;
41              "RobotData".inputs.interrupts.exitCycle := "HMIData".robotInputs.
                    interrupts.exitCycle OR "StateData".stopRobot;
42              "RobotData".inputs.interrupts.removeProd := "HMIData".robotInputs.
                    interrupts.removeProd;
43              "RobotData".inputs.interrupts.changeSpeed := ("RobotData".inputs.speedRef
                    <> "HMIData".robotInputs.speedRef);
44          END_REGION
45
46          "RobotData".inputs.speedRef := "HMIData".robotInputs.speedRef;
47
48          REGION recipeData
49              //write number of products and dimension for palletization
50              "RobotData".inputs.productsNum := "RecipeActive".productsNum;
51              "RobotData".inputs.productsType := "RecipeActive".productsType;
52
53              "RobotData".inputs.pickOffs := "RecipeActive".pickOffs;
54              "RobotData".inputs.placeOffs := "RecipeActive".placeOffs;
55
56              "RobotData".inputs.palletX := "RecipeActive".palletX;
57              "RobotData".inputs.palletY := "RecipeActive".palletY;
58              "RobotData".inputs.palletZ := "RecipeActive".palletZ;
59              "RobotData".inputs.palletR := "RecipeActive".palletR;
60
61          END_REGION
62
63          //get number of products removed by HMI
64          "RobotData".inputs.removedProd := "HMIData".robotInputs.removedProd;
65
66      END_REGION
67
68
```

```
69
70 REGION Write Vision System Data
71
72     "VisionSystemData".inputs.trgControl := "RobotData".outputs.trgControl OR "HMIData
       ".visionSystemInput.trgControl;
73     "VisionSystemData".inputs.resultAckFlag := "RobotData".outputs.resultAckFlag OR "
       HMIData".visionSystemInput.resultAckFlag;
74     "VisionSystemData".inputs.commandRequestFlag := "RobotData".outputs.
       commandRequestFlag OR "HMIData".visionSystemInput.commandRequestFlag;
75
76     IF  "RobotData".outputs.commandRequestFlag THEN
77
78         "VisionSystemData".inputs.commandNumber := "RobotData".outputs.commandNumber;
79         "VisionSystemData".inputs.commandParameter[1] := "RobotData".outputs.
               commandParameter[1];
80         "VisionSystemData".inputs.commandParameter[2] := "RobotData".outputs.
               commandParameter[2];
81
82     ELSIF "HMIData".visionSystemInput.commandRequestFlag THEN
83
84         "VisionSystemData".inputs.commandNumber := "HMIData".visionSystemInput.
               commandNumber;
85         "VisionSystemData".inputs.commandParameter[1] := "HMIData".visionSystemInput.
               commandParameter[1];
86         "VisionSystemData".inputs.commandParameter[2] := "HMIData".visionSystemInput.
               commandParameter[2];
87
88     END_IF;
89
90
91 END_REGION
92
93 REGION Write HMI Data
94
95         //convert value to ms
96         "HMIData".executionTime := "VisionSystemData".executionTime / 1000;
97         "HMIData".statistics := "VisionSystemData".statistics;
98
99         REGION robotCoordinates
100
101             //convert value to mm
102             "HMIData".coordinates.x := "VisionSystemData".coordinates.x / 1000;
103             "HMIData".coordinates.y := "VisionSystemData".coordinates.y / 1000;
104             //convert value to grades
105             "HMIData".coordinates.angle := "VisionSystemData".coordinates.angle /
                    1000;
106
107     END_REGION
108
109
110 END_REGION
```

Listato 2.12: Comunication()

2.5.7 Il blocco FC Outputs

In questo blocco i dati globali vengono scritti sulla periferia.

```
1
```

```
"visionSystemInput" := "VisionSystemData".inputs;
"robotInput" := "RobotData".inputs;
```

Listato 2.13: Outputs()

Capitolo 3

Il software del braccio robotico

In questo capitolo saranno descritti i diversi moduli che costituiscono il programma del braccio robotico, un IRB1600.6.1,45 con un payload di 6Kg ed estensione del braccio di 1,45m[2]. Per una più chiara comprensione, si è deciso di introdurre la trattazione con alcuni accenni teorici al linguaggio RAPID, un linguaggio di programmazione sviluppato per i robot industriali di marca ABB.

3.1 Introduzione al linguaggio RAPID

Il linguaggio di programmazione standard per questa marca di robot è il RAPID, un linguaggio strutturato simile al Python o a una versione ridotta di un linguaggio C. Un programma RAPID è costituito tipicamente da uno o più moduli, i quali possono comprendere dati e diverse routine. È possibile copiare ciascun modulo, o l'intero programma, su disco, e così via e viceversa. Uno dei moduli contiene la procedura di ingresso, globale, denominata Main. L'esecuzione del programma indica, in effetti, l'esecuzione di questa procedura.[3]

3.1.1 Le routine

Esistono tre tipologie di routine: procedure, funzioni e trap.

- Una procedura viene utilizzata come un sottoprogramma.
- Una funzione restituisce un valore di un tipo specifico e viene utilizzata come un argomento di un'istruzione.
- Le trap consentono di rispondere agli interrupt. È possibile associare una routine trap a uno specifico interrupt; ad esempio la routine trap viene eseguita automaticamente se si verifica un determinato interrupt impostato precedentemente.

Sono disponibili varie routine standard. Di seguito un breve elenco con alcuni esempi.

- Routine di gestione dei segnali: Set, Reset, SetGO, SetAO, PulseDO.
- Routine di attesa: WaitTime, WaitUntil, WaitAI, WaitAO, WaitDI, WaitDO.
- Routine sui sistemi di riferimento: Offs, RelTool.

3.1.2 I tipi di dati

Per quanto riguarda i tipi di dati, anch'essi si possono dividere in tre categorie: atomici, record e alias.

- Un tipo di dati atomico è tale, poiché non viene definito in base ad alcun altro tipo e non può essere suddiviso in parti o componenti. Esempi di tipi atomici sono i num e i bool.
- Un tipo di dati record è un tipo composto da componenti nominati ed ordinati, ad esempio pos = [300,100,0]. Ad uno specifico componente di dati si può accedere utilizzando il nome di tale componente, ad esempio pos1.x := 300 per attribuire all'elemento x il valore 300.
- Un tipo di dati alias è, per definizione uguale ad un altro tipo.

3.1.3 La istruzioni

Il linguaggio RAPID, oltre alle classiche istruzioni di un linguaggio strutturato quali IF, FOR, WHILE, CASE, si contraddistingue per le istruzioni di movimento. Una classica istruzione di movimento in RAPID si presenta come:

```
1           MoveL p1,v500,z10,tool1\wobj:=wobj1;
```

dove:

- MoveL: istruzione di movimento lineare
- p1: robtarget, punto di destinazione
- v500: speeddata, velocità
- z10: zonedata, zona di raccordo
- tool1: tooldata, sistema di riferimento dell' utensile
- wobj1: wobjdata, sistema di riferimento dell'oggetto di lavoro

3.2 Il software

3.2.1 Il Main

La procedura Main() è costituita da un costrutto TEST, corrisponde allo switch del linguaggio C, che lancia un diversa routine in base al codice della missione inviato da PLC. Dopo aver disattivato il controllo sulla configurazione degli assi con le istruzioni ConfJ e ConfL, il robot viene quindi portato in posizione di Home con la procedura mvHome. Se il ciclo si conclude con successo il robot torna nuovamente in posizione di Home, in caso contrario gli eventuali errori sono gestiti dalla procedura errHandler.

```
1
2     PROC Main()
3
4         ConfL\Off;
5         ConfJ\Off;
6
7         mvHome;
8
9         TEST giMission
10        CASE 1:
```

```
            Palletize pFirstPlace;
        CASE 2:
            PalletizeInterlocking pFirstPlace;
        CASE 3:
            VisionSysCalib;
        ENDTEST

        mvHome;

    ERROR
        errHandler;

    ENDPROC
```

Listato 3.1: La procedura Main()

3.2.2 Il modulo CalibData

In questo modulo sono definiti i dati necessari per calcolare le traiettorie. Prima di prendere i punti di una traiettoria è infatti necessario definire il sistema di riferimento e i dati dello strumento montato sulla flangia, chiamati rispettivamente workobject e tool.

Workobject

Il workobject è un sistema di coordinate utilizzato per descrivere la posizione di un oggetto di lavoro. L'oggetto di lavoro si compone di due sistemi di riferimento: un sistema di riferimento utente e un sistema di riferimento oggetto. Tutte le posizioni programmate saranno relative al sistema di riferimento oggetto, che è relativo al sistema di riferimento utente, a sua volta relativo al sistema di coordinate universali. Queste ultime corrispondo di default al sistema di riferimento della base così definito:

- L'origine si trova sull'intersezione dell'asse 1 con la superficie di montaggio della base.
- Il piano xy è lo stesso della superficie di montaggio della base.
- L'asse delle x punta in avanti.
- L'asse y punta verso sinistra (dalla prospettiva del robot).
- L'asse z punta verso l'alto.

Nell' applicazione di pallettizzazione sviluppata si è scelto di definire le traiettorie all'interno del sistema di riferimento del tavolo. In questo modo, a variazioni dello spostamento del tavolo, è possibile modificare il solo sistema di riferimento mantenendo invariati i punti di lavoro.
I valori del sistema di riferimento sono i seguenti:

- Il robot non sta sostenendo l'oggetto di lavoro.
- Viene utilizzato il sistema di coordinate utente fisso.
- Il sistema di coordinate utente viene ruotato in angoli di Eulero ZYX di -90 gradi su Z mentre le coordinate della relativa origine sono x= 690, y = -506 e z = 88 mm nel sistema di coordinate universali.
- Il sistema di coordinate oggetto non viene ruotato in angoli di Eulero ZYX di 180 gradi su X e -90 gradi su Z mentre le coordinate della relativa origine sono x = 0, y = 200 e z = 30 mm nel sistema di coordinate utente.

```

    PERS wobjdata wobjTable :=  [FALSE,TRUE,"",[[690,-506,88],[0.7071,0,0,-0.7071]],
                                [[0,0,0],[0,-0.7071,0.7071,0]]];
```

Listato 3.2: CalibData: wobjTable

Tool

Il tooldata viene utilizzato per descrivere le caratteristiche di un utensile, ad esempio una testa di saldatura o una pinza. Queste caratteristiche sono la posizione e l'orientamento del punto centrale dell'utensile (TCP) e le caratteristiche fisiche del carico dell'utensile.
L'utensile utilizzato è una pinza pneumatica con i seguenti valori:

```
1
2     PERS tooldata tGripper:=[TRUE,[[-125,0,72.4],[0,0.7071,0,-0.7071]],
3                             [1.4,[-40.5,2.8,26.4],[1,0,0,0],0.081,0.076,0.019]];
```

Listato 3.3: CalibData: tGripper

- Il robot sta sostenendo l'utensile.
- Il TCP si trova su un punto a 72,4 mm sulla normale alla flangia di montaggio e a -125 mm lungo l'asse X del sistema di coordinate polso.
- La direzioni dell'utensile sono ruotate in angoli di Eulero ZYX di -180ř su Z e 90ř su Y.
- La massa dell'utensile è di 1,4 kg.
- Il baricentro si trova su un punto a 26,4 mm sulla normale alla flangia di montaggio -40,5 mm lungo l'asse X e 2,8mm sull'asse y del sistema di coordinate del polso,
- Il momento di inerzia è trascurabile.

Robtarget

Una volta definito il workobject e il tool, è possibile definire i punti della traiettoria. Questi possono essere salvati come posizioni angolari dei singoli assi, jointtarget, oppure come posizione del TCP rispetto ad determinato workobject. In questo caso si parla di robtarget.
Il robtarget pHome nel Listato 3.4 è definito dai seguenti valori:

```
1
2     CONST robtarget pHome:=[[-50,400,-200],[1,0,0,0],[-1,1,-1,0],
3                             [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Listato 3.4: CalibData: pHome

- La posizione del robot: x = -50, y = 400, e z = -200 mm nel sistema di coordinate oggetto.
- L'orientamento dell'utensile nella stessa direzione del sistema di coordinate oggetto.
- Il terzo valore indica i dati di configurazione.
- Non sono presenti assi esterni.

L'orientamento

A differenza della maggior parte dei robot industriali, che utilizzano la rappresentazione in angoli di Eulero, l'orientamento nei robot ABB è descritto nella forma di un quaternione costituito da quattro componenti: q1, q2, q3 e q4.

```
1     CONST orient orVertical:=[1,0,0,0];
2     CONST orient orHorizontal:=[0.707,0,0,-0.707];
```

Listato 3.5: CalibData: gli orientamenti

In confronto agli angoli di Eulero, i quaternioni presentano funzioni più semplici da comporre ed evitano il problema del blocco cardanico. Inoltre, confrontati con le matrici di rotazione, essi sono più stabili numericamente e più efficienti. Lo svantaggio di questa notazione è la perdita di leggibilità e la difficoltà nel calcolarli a mente. Per ovviare a queste problematiche il linguaggio RAPID fornisce due

funzioni per la conversione da angoli di Eulero a quaternioni e viceversa, rispettivamente OrienrZYX() e EulerZYX().

```
1         nRotZ:=EulerZYX(\z,pProd.rot);
2
3         IF nRotZ>0 THEN
4             nRotZ:=nRotZ-90;
5         ELSE
6             nRotZ:=nRotZ+90;
7         ENDIF
8
9         pProd.rot:=OrientZYX(nRotZ,0,0);
```

Listato 3.6: Esempio di utilizzo di OrientZYX() e EulerZYX()

Per una maggiore leggibilità sono state definite due costanti per rappresentare l'orientamento, vedi il Listato 3.5, utilizzato per tutto il codice.

3.2.3 Il modulo WorldZoneModule

All'interno di un' applicazione RAPID è possibile definire delle World Zone, cioè aree di lavoro supervisionate durante i movimenti del robot, sia in modalità manuale che automatica. Nel modulo WorldZoneModule sono definite le seguenti aree cubiche:

- wzHome: area di Home.
- wzPicking: area di prelievo sottostante la camera del sistema di visione.
- wzPlacing: area di deposito.
- wzWarehouse: area del magazzino.
- wzDangerouse: area al di fuori della zona di lavoro.

Il primo passo per creare una World Zone cubica è definire il volume del cubo con l'istruzione WZBoxDef. L'istruzione successiva WZDOSet imposta l'attivazione di un output digitale quando il TCP si trova al di fuori(Outside) o all'interno(Inside) dell'area supervisionata.

```
1
2     CONST pos psPickingCorner1:=[-150,200,0];
3     CONST pos psPickingCorner2:=[-700,800,-800];
4
5     VAR wztemporary wzPicking;
6     VAR shapedata pickingVolume;
7
8     WZBoxDef\Inside,pickingVolume,posTableToWCS(psPickingCorner1),
9             posTableToWCS(pspickingCorner2);
10
11    WZDOSet\Temp,wzPicking\Inside,pickingVolume,dowzPicking,1;
```

Listato 3.7: WorldZoneModule: definizione del cubo della area di prelievo

I dati di posizione psPickingCorner1 e psPickingCorner2 descrivono gli angoli opposti del cubo all'interno del sistema di riferimento del tavolo. Dato che l'istruzione WZBoxDef necessita di dati di posizione all'interno del sistema di riferimento universale, è stato utilizzata la funzione posTableToWCS() mostrata nel Listato 3.8, che utilizza le funzioni standard poseVect() e poseMult() per convertire una posizione nel sistema di riferimento del tavolo in una posizione nel sistema di riferimento universale.

```
1
2     FUNC pos posTableToWCS(pos ps)
3
4         RETURN poseVect(poseMult(wobjtable.uframe,wobjtable.oframe),ps);
```

```
5
6       ERROR
7           RAISE ;
8
9       ENDFUNC
```

Listato 3.8: WorldZoneModule: definizione della World Zone dell'area di prelievo

L"utilizzo delle aree di lavoro verrà approfondite in seguito lungo la trattazione.

3.2.4 Il modulo ErrModule

Molte condizioni di errore che si verificano durante l'esecuzione di un programma possono essere gestite al suo interno senza interromperne l'esecuzione. Questi tipi errori sono rilevati dal sistema, ad esempio la divisione per zero, o dal programma, ad esempio gli errori generati quando viene letto un valore non corretto da un lettore di codice a barre. Se nella routine corrente non è disponibile un gestore di errori, viene attivato direttamente il gestore interno del robot, che genera un messaggio di errore e termina l'esecuzione del programma.[4]
Ogni errore è identificato da un numero univoco, l' ERRNO, che può essere utilizzato all'interno del programma per determinare il tipo di errore verificatosi. Con l'istruzione RAISE è possibile definire ulteriori errori di tipo errnum, un dato alias del num. RAISE permette inoltre di propagare un errore da un gestore errori, a partire da una catena di chiamate nidificate, fino a un livello superiore.
Nel modulo errModule sono state definite le costanti per gli errori richiamati all'interno delle routine del programma.

```
1
2  MODULE ErrModule
3
4      CONST errnum errOpenGripperTimeout:=1;
5      CONST errnum errProdNotFound:=2;
6      CONST errnum errResulTimeOut:=3;
7      CONST errnum errMotionInterrupted:=4;
8      CONST errnum errPickInterrupted:=5;
9      CONST errnum errPlaceInterrupted:=6;
10     CONST errnum errProdsRemoved:=7;
11     CONST errnum errStoreInterrupted:=9;
12
13
14     PROC errHandler()
15
16         ResetOutputs;
17         setAlarm ERRNO;
18
19         ExitCycle;
20
21     ENDPROC
22
23
24     PROC setAlarm(num nErr)
25
26         SetGO goErrorNumber,nErr;
27         PulseDO\High\PLength:=2,doAlarm;
28
29     ENDPROC
30
31 ENDMODULE
```

Listato 3.9: ErrModule

Ogni procedura presenta sempre al suo interno un istruzione RAISE per propagare l'errore al livello successivo. Se l'errore raggiunge il Main(), senza essere stato precedentemente gestito, viene richiamata la procedura errHandler():

1. vengono resettati gli outputs correnti;
2. viene inviato il codice di errore contenuto della variabile ERRNO;
3. viene inviato il segnale di allarme;
4. l' esecuzione del ciclo termina con l'istruzione ExitCycle.

3.2.5 Il modulo ToolModule

In questo modulo sono gestite le procedure di apertura e chiusura della pinza. Questa è collegata alla scheda di comunicazione interna al robot per la gestione delle sue elettrovalvole. Con i segnali i_GripperExitClose e o_RetractorEntry è quindi possibile comandare e leggere lo stato della pinza.

```
1   ! i_GripperExitClose : 1=closed 0=opened !!o_RetractorEntry : 1=closing 0=opening
2   PROC CloseGripper()
3
4       IDisable;
5
6       SetDO\Sync,o_RetractorEntry,high;
7       WaitTime 0.2;
8       Reset doGripperOpen;
9
10      IEnable;
11
12  ERROR
13      RAISE;
14
15  ENDPROC
```

Listato 3.10: ToolModule: procedura di chiusura della pinza

```
1
2   PROC OpenGripper()
3
4       VAR bool bTimeOut:=FALSE;
5
6       IDisable;
7
8       SetDO\Sync,o_RetractorEntry,low;
9       WaitDI i_GripExitClose,low\MaxTime:=2\TimeFlag:=bTimeOut;
10
11      IF bTimeOut THEN
12          RAISE errOpenGripperTimeout;
13      ELSE
14          Set doGripperOpen;
15      ENDIF
16
17      IEnable;
18
19  ERROR
20      RAISE;
21
22  ENDPROC
```

Listato 3.11: ToolModule: procedura di apertura della pinza

3.2.6 Il modulo VisionSysMoule

Per gestire la comunicazione con il sistema di visione è stata sviluppata un' apposita libreria. Di seguito una breve descrizione delle procedure che la costituiscono.

La procedura VisionSysResetTrig

Resetta il segnale di trigger per una successiva elaborazione.

```
1     PROC VisionSysResetTrig()
2
3         Set doResultACK;
4         WaitDI diResultReady,low\MaxTime:=5;
5         Reset doResultACK;
6
7     ERROR
8         RAISE ;
9
10    ENDPROC
```

La procedura VisionSysTrig

Invia il segnale di trigger per una nuova elaborazione.

```
1
2     PROC VisionSysTrig(\num nDelay)
3
4         IF diResultReady=1 VisionSysResetTrig;
5
6         IF Present(nDelay) WaitTime nDelay;
7
8         PulseDO\high,doTrgControl;
9
10    ERROR
11        RAISE ;
12    ENDPROC
```

La procedura VisionSysSendCommand

Invia un comando al sistema di visione e restituisce i dati del risultato.

```
1     PROC VisionSysSendCommand(num nCommand,\dnum nParameter1,\dnum nParameter2)
2
3         VAR dnum nCommandParameter1:=0;
4         VAR dnum nCommandParameter2:=0;
5
6         IF Present(nParameter1) nCommandParameter1:=nParameter1;
7         IF Present(nParameter2) nCommandParameter2:=nParameter2;
8
9         IF diCommandReadyFlag=0 THEN
10            Reset doCommandRequestFlag;
11            WaitDI diCommandReadyFlag,high;
12        ENDIF
13
14        SetGO goCommandNumber,nCommand;
15        SetGO goCommandParameter1,nCommandParameter1;
16        SetGO goCommandParameter2,nCommandParameter2;
17
18        Set doCommandRequestFlag;
```

```
        WaitDI diCommandCompleteFlag,high\MaxTime:=30;
        Reset doCommandRequestFlag;

    ERROR
        RAISE ;

    ENDPROC
```

La funzione VisionSysCProgram

Restituisce il programma attualmente presente nel sistema di visione.

```
    FUNC dnum VisionSysCProgram()

        VAR dnum CProgram;

        VisionSysSendCommand READ_PROG_COMMAND;
        Cprogram:=GInputDnum(giCommandData2);

        RETURN CProgram;

    ERROR
        RAISE ;

    ENDFUNC
```

LA procedura VisionSysSetProgram

Se non lo è già, imposta il sistema di visione in modalità *RUN*. Imposta un nuovo programma all'interno del sistema di visione opppure, se il programma richiesto è attualmente presente, ne resetta le statistiche.

```
    PROC VisionSysSetProgram(dnum nProgram)

        Reset doTrgControl;
        Reset doResultACK;

        IF diRunMode=0 THEN
            VisionSysSendCommand RUN_MODE_COMMAND;
        ENDIF

        IF VisionSysCProgram()<>nProgram THEN
            Set doChangingVisionSysProg;
            VisionSysSendCommand CHANGE_PROG_COMMAND,
            \nParameter1:=1,\nParameter2:=nProgram;
            Reset doChangingVisionSysProg;
        ELSE
            VisionSysSendCommand RESET_COMMAND;
        ENDIF

        VisionSysTrig;

    ERROR
        RAISE ;

    ENDPROC
```

La funzione VisionSysRobT

Elabora le coordinate provenienti dal sistema di visione(aiPosX, aiPosY, aiRotZ) in un punto del sistema di riferimento del tavolo. PeVisionSysToTable contiene i dati della trasformata per passare dal sistema di riferimento del sistema di visione a quello del tavolo. La funzione poseMult restituisce il risultato della trasformazione.

```
FUNC robtarget VisionSysRobT()

    peProd:=PoseMult(peVisionSysToTable,[[(aiPosx/1000)
    *nresolution,(aiPosy/1000)*nresolution,0],
    OrientZYX(aiRotz/1000,0,0)]);

    RETURN [peProd.trans,peProd.rot,[1,0,0,0],
    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

ERROR
    RAISE ;
ENDFUNC
```

La funzione VisionSysCalib

Per convertire i punti dal sistema di coordinate della camera al sistema di coordinate del tavolo è necessario prima definire le trasformazioni tra i due sistemi. Per questa ragione è stata sviluppata una procedura di calibrazione che utilizza l'istruzione RAPID DefAccFrame, con la quale è possibile definire un sistema di riferimento a partire da due array di punti. Gli elementi dell'array con lo stesso indice corrispondono allo stesso punto calcolato in due sistemi di riferimento differenti: il primo (punti pCamVS) è il sistema di riferimento del sistema di visione, il secondo (punti pCamWCS) è il sistema di riferimento universale 3.2.2. Quando la procedura di calibrazione viene avviata, il sistema di visione scatta una foto all'area di prelievo e rileva i punti di calibrazione, le cui coordinate sono state precedentemente salvate nel sistema di riferimento tavolo. Una volta ricevuti i dati dal sistema di visione, il robot si muove sui punti di calibrazione e salva le coordinate nel sistema di riferimento universale. Conclusa la procedura, peVisionSys contiene i dati del sistema di riferimento del sistema di visione.

```

CONST robtarget pCam{5}:=[pCam1,pCam2,pCam3,pCam4,pCam5];

PERS robtarget pCamVS{5}:=[[[66.7898,168.901,0],[1,0,0,0],[1,0,0,0],[9E+09,9E+09,9
    E+09,9E+09,9E+09,9E+09]],[[233.956,149.256,0],[1,0,0,0],[1,0,0,0],[9E+09,9E
    +09,9E+09,9E+09,9E+09,9E+09]],[[147.652,91.9167,0],[1,0,0,0],[1,0,0,0],[9E
    +09,9E+09,9E+09,9E+09,9E+09,9E
    +09]],[[53.3243,55.8337,0],[1,0,0,0],[1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E
    +09,9E+09]],[[226.856,39.3594,0],[1,0,0,0],[1,0,0,0],[9E+09,9E+09,9E+09,9E
    +09,9E+09,9E+09]]];

PERS robtarget pCamWCS{5}:=[[[1114,-79,93],[1,0,0,0],[-1,-1,2,1],[9E+09,9E+09,9E
    +09,9E+09,9E+09,9E+09]],[[1079,84,93],[1,0,0,0],[0,0,1,1],[9E+09,9E+09,9E+09,9
    E+09,9E+09,9E+09]],[[1033,-8,93],[1,0,0,0],[-1,-1,2,1],[9E+09,9E+09,9E+09,9E
    +09,9E+09,9E+09]],[[1005,-104,93],[1,0,0,0],[-1,-1,2,1],[9E+09,9E+09,9E+09,9E
    +09,9E+09,9E+09]],[[972,65,93],[1,0,0,0],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E
    +09,9E+09]]];


PERS pos psVisionSysToTable:=[-346.134,419.617,0];

```

```
11
12      PROC VisionSysCalib()
13
14          CONST num nCalibPoints:=5;
15
16          VisionSysSetProgram N_CALIBRATION_PROG;
17          SearchProd;
18
19          FOR i FROM 1 TO nCalibPoints DO
20              SetGO goIndex,i-1;
21
22              ! get point in Word coordinate system
23              MoveL pCam{i},vfast,fine,tGripper,\WObj:=wobjTable;
24              pCamWCS{i}:=CRobT(\Tool:=tgripper\WObj:=wobj0);
25              pCamWCS{i}.rot:=[1,0,0,0];
26
27              ! Get point in vision system Coordinate system
28              waituntil GOutput(goIndex)=giIndex;
29              pCamVS{i}:=[[(aiPosx/1000)*nresolution,(aiPosy/1000)*nresolution
                    ,0],[1,0,0,0],[1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
30          ENDFOR
31
32          peVisionSys:=DefAccFrame(pCamVS,pCamWCS,nCalibPoints,nmaxErr,nmeanErr);
33
34          mvHome;
35
36      ERROR
37          RAISE ;
38
39
40      ENDPROC
```

Listato 3.12: VisionSysModule: VisionSysCalib

3.2.7 Il modulo InterruptsModule

Per comunicare in maniera asincrona rispetto all'esecuzione del programma del robot, è presente un modulo contenente interrupts e trap.

L'interrupt intStopMove

Interrompe il movimento del moto.

```
1
2       ISignalDI diIntStopMove,high,intStopMove;
3
4       TRAP trapstopmove
5           StopMove;
6           Set doNotMoving;
7           nRemovedProds:=0;
8       ENDTRAP
```

L'interrupt intStartMove

Riprende il movimento interrotto. Se durante la fase di prelievo il robot si trova all'interno del sistema di visione, si sposta al di fuori ed esegue un nuovo PickVisionSysRobT.

```
1
2     ISignalDI diIntStartMove,high,intStartMove;
3
4     TRAP trapStartmove
5
6         VAR bool bPickRestarted:=FALSE;
7
8         Reset doNotMoving;
9
10        IF bEnablePickRestart bPickRestarted:=CheckRestartPick();
11
12        IF CheckRemovedProds() OR bPickRestarted THEN
13            RAISE errMotionInterrupted;
14        ELSE
15            StartMove;
16        ENDIF
17
18    ERROR
19        IF ERRNO=errMotionInterrupted THEN
20            ClearPath;
21            StartMove;
22        ENDIF
23
24        RAISE ;
25
26    ENDTRAP
```

L'interrupt intExitCycle

Conclude l'esecuzione del programma.

```
1
2     ISignalDI diIntExitCycle,high,intExitCycle;
3
4     TRAP trapExitCycle
5         ResetOutputs;
6         suctionDisable;
7         ExitCycle;
8     ENDTRAP
```

L'interrupt intChangeSpeed

Cambia la velocità del robot.

```
1
2     ISignalDI diIntChangeSpeed,high,intChangeSpeed;
3
4     TRAP trapChangeSpeed
5
6         vfast.v_tcp:=giSpeedRef;
7
8     ENDTRAP
```

L'interrupt intRemoveProd

Segnala al robot che un prodotto è stato rimosso.

```
1
2     ISignalDI diIntRemoveProd,high,intRemoveProd;
3
4
5     TRAP trapRemoveProds
6
7         !if removed Prod isn't in the last level,
8         !remove all superior products.
9         !ex: level=2 dim=2 => index=5
10
11        pallet.index:=Calclevel(giRemovedProd)*pallet.dim+1;
12
13        WHILE pallet.index<=pallet.nProds DO
14            IF prods{pallet.index}.placed THEN
15                prods{pallet.index}.placed:=FALSE;
16                incr nRemovedProds;
17            ENDIF
18            incr pallet.index;
19        ENDWHILE
20
21        prods{giRemovedProd}.placed:=FALSE;
22        Incr nRemovedProds;
23
24    ENDTRAP
```

L'interrupt intTrig

Esegue un VisionSysTrig per il prelievo successivo non appena il robot è al di fuori dell'area visualizzata dal sistema di visione.

```
1
2     ISignalDO dowzPicking,low,intTrig;
3
4     TRAP trapTrig
5
6         PulseDO\High,doTrgControl;
7
8     ENDTRAP
```

L'interrupt intColl

Riconosce le collisioni ed invia una segnalazione.

```
1
2     IError MOTION_ERR\ErrorId:=204,TYPE_ALL,intErrColl;
3
4     TRAP trapErrColl
5
6         setAlarm 204;
7         ExitCycle;
8
9     ENDTRAP
```

3.2.8 Il modulo PickAndPlaceModule

Questo modulo contiene le diverse procedure sviluppate per eseguire un ciclo di "pick and place".

La procedura Pick

Procedura di prelievo.

```
1   PROC Pick(robtarget p,num nZOffset)
2
3       OpenGripper;
4       mvTopDown p,nZOffset;
5       CloseGripper;
6       Set doGripperFull;
7       MoveL Offs(p,0,0,nZOffset),vFast,z50,tGripper\WObj:=wobjTable;
8
9
10  ERROR
11
12      IF ERRNO=errMotionInterrupted THEN
13          IF nRemovedProds>0 THEN
14              RAISE errProdsRemoved;
15          ELSE
16              RAISE errPickInterrupted;
17          ENDIF
18      ELSE
19          RAISE ;
20      ENDIF
21
22  ENDPROC
```

Listato 3.13: PickAndPlaceModule: Pick

La procedura Place

Procedura di deposito.

```
1   PROC Place(robtarget p,num nZOffset)
2
3
4
5       mvTopDown p,nZOffset;
6       OpenGripper;
7       Reset doGripperFull;
8       MoveL Offs(p,0,0,nZOffset),vFast,z50,tGripper\WObj:=wobjTable;
9
10
11
12  ERROR
13
14      IF ERRNO=errMotionInterrupted THEN
15
16          IF NOT gripperFull() Pick p,0;
17          IF nRemovedProds>0 THEN
18              RAISE errProdsRemoved;
19          ELSE
20              RAISE errPlaceInterrupted;
21          ENDIF
22
23      ELSE
24          RAISE ;
25      ENDIF
26  ENDPROC
```

Listato 3.14: PickAndPlaceModule: Place

La procedura SearchProd

Procedura di ricerca di un prodotto nella zona di prelievo. Invia una richiesta di elaborazione e attende il risultato. Se l'elaborazione fallisce o non arriva entro 10s, ritenta una nuova elaborazione. Dopo 3 elaborazioni restituisce un segnale di errore.

```
1
2     PROC SearchProd()
3
4         VAR bool bProdFound:=false;
5         VAR bool bTimeOut:=false;
6         VAR num nFailsCounter:=0;
7
8         WHILE ((NOT bProdFound) AND (nFailsCounter<3)) DO
9
10            WaitDI diResultReady,high\MaxTime:=10\TimeFlag:=bTimeOut;
11
12            IF bTimeOut THEN
13                RAISE errResulTimeOut;
14            ELSE
15                IF diToolJudge0=1 THEN
16                    VisionSysResetTrig;
17                    bProdFound:=true;
18                ELSE
19                    Incr nFailsCounter;
20                    VisionSysTrig;
21                ENDIF
22            ENDIF
23
24        ENDWHILE
25
26        IF NOT bProdFound RAISE errProdNotFound;
27
28    ERROR
29        RAISE ;
30    ENDPROC
```

Listato 3.15: PickAndPlaceModule: SearchProd

La procedura PickVisionSysRobT

Procedura di prelievo con l'utilizzo del sistema di visione. Se la procedura SearchProd si conclude con successo, viene prelevato il prodotto alle coordinate ricevute dal sistema di visione. La rotazione viene mantenuta all'interno di un range di 180 gradi.

```
1
2     PROC PickVisionSysRobT(num nZOffset)
3
4         VAR robtarget pProd;
5         VAR num nRotZ;
6
7         SearchProd;
8         pProd:=VisionSysRobT();
9
10        nRotZ:=EulerZYX(\z,pProd.rot);
11
12        IF nRotZ>0 THEN
13            nRotZ:=nRotZ-90;
14        ELSE
```

```
15              nRotZ:=nRotZ+90;
16          ENDIF
17
18          pProd.rot:=OrientZYX(nRotZ,0,0);
19
20          Pick pProd,nZOffset;
21
22      ERROR
23          IF ERRNO=errProdNotFound THEN
24              ProdNotFoundMessage;
25              RETRY;
26          ELSE
27              RAISE ;
28          ENDIF
29
30
31      ENDPROC
```

Listato 3.16: PickAndPlaceModule: PickVisionSysRobT

La procedura PickAndPlace

Procedura completa di prelievo e deposito con i dati provenienti dal sistema di visione.

```
1
2       PROC PickAndPlace(robtarget pPlace,num nZOffset)
3
4           bEnablePickRestart:=TRUE;
5
6           PickVisionSysRobT nZOffset;
7
8           IWatch intTrig;
9
10          Place pPlace,nZOffset;
11
12          bEnablePickRestart:=FALSE;
13          ISleep intTrig;
14
15      ERROR
16
17          IF ERRNO=errPickInterrupted THEN
18              RETRY;
19          ELSE
20              RAISE ;
21          ENDIF
22
23      ENDPROC
```

Listato 3.17: PickAndPlaceModule: PickAndPlace

3.2.9 Il modulo PallettizeModule

In questo modulo è sviluppata la procedura di pallettizzazione, utilizzando le routine e dati illustrati precedentemente.
Per prima cosa, viene impostato il programma all'interno del sistema di visione corrispondente al formato. Successivamente, vengono calcolati i dati del pallet con i dati a ricetta selezionati dall'operatore. Il ciclo di prelievo e deposito continua fino al completamento del pallet, dopodiché il ciclo termina con il robot in posizione di Home.

```
1
2     PROC Palletize(robtarget p0)
3
4         VisionSysSetProgram N_PALLETTIZE_PROG;
5
6         CalcPalletPoints p0;
7
8         WHILE PlaceProd() DO
9         ENDWHILE
10
11        mvSafeZ tGripper;
12        mvHome;
13
14    ERROR
15        RAISE ;
16
17    ENDPROC
```

Listato 3.18: PallettizeeModule: Pallettize

La funzione PlaceProd

Di seguito viene descritta la funzione PlaceProd. Ad ogni richiamo, viene controllato l'intero array contenente le informazioni sui prodotti depositati. Se il pallet non è ancora completo, viene prelevato un nuovo prodotto dall'area di prelievo e successivamente depositato sulla prima posizione libera del pallet. La funzione restituisce il valore TRUE se un nuovo prodotto è stato depositato, FALSE in caso contrario.

```
1
2     FUNC bool PlaceProd()
3
4         VAR bool newprodplaced:=FALSE;
5
6         pallet.index:=1;
7
8         WHILE pallet.index<=pallet.nProds AND (NOT newprodplaced) DO
9
10            IF NOT prods{pallet.index}.placed THEN
11                IF gripperFull() THEN
12                    Place prods{pallet.index}.point,nZOffset;
13                ELSE
14                    PickAndPlace prods{pallet.index}.point,nZOffset;
15                ENDIF
16                IncrProds;
17                newprodplaced:=TRUE;
18            ELSE
19                Incr pallet.index;
20            ENDIF
21
22        ENDWHILE
23
24        RETURN newprodplaced;
25
26    ERROR
27
28        IF ERRNO=errProdsRemoved THEN
29            RETURN TRUE;
30        ELSE
```

```
31              RAISE ;
32          ENDIF
33
34      ENDFUNC
```

Listato 3.19: PallettizationModule: PlaceProd

3.2.10 Il modulo PallettizeInterlockingModule

La presente funzione è del tutto analoga a quella precedente, con la differenza di effettuare una pallettizzazione "con incastro": i prodotti di questo formato, divisi in una parte bianca e una nera, devono essere depositati alternando le due parti. Si è quindi proceduto partendo dall'idea di base della pallettizzazione, sostituendo la procedura PlaceProduct con la procedura PlaceInterlocking. In base all'indice del prodotto viene richiesto un prodotto appartenente ad un diverso gruppo, identificato dalla rotazione. In caso di un' interruzione del ciclo e successiva rimozione di un prodotto, viene controllato lo stato della pinza: se contiene un prodotto non appartenente al gruppo richiesto, quest'ultimo viene scartato e il ciclo riprende dal successivo prelievo.

La funzione PlaceInterlocking

```
1
2       FUNC bool PlaceInterlocking()
3
4           VAR bool newprodplaced:=FALSE;
5
6           pallet.index:=1;
7
8           WHILE pallet.index<=pallet.nProds AND (NOT newprodplaced) DO
9
10              IF NOT Prods{pallet.index}.placed THEN
11                  IF gripperFull() THEN
12                      IF selectGroup()=DOutput(doGroup1) THEN
13                          Place Prods{pallet.index}.point,nZOffset;
14                          IncrProds;
15                      ELSE
16                          Store;
17                      ENDIF
18                  ELSE
19                      AlternateProds;
20                      PickAndPlace Prods{pallet.index}.point,nZOffset;
21                      IncrProds;
22                  ENDIF
23                  newprodplaced:=TRUE;
24              ELSE
25                  Incr pallet.index;
26              ENDIF
27
28          ENDWHILE
29
30          RETURN newprodplaced;
31
32      ERROR
33
34          IF ERRNO=errProdsRemoved THEN
35              RETURN TRUE;
36          ELSE
```

```
37              RAISE ;
38          ENDIF
39
40      ENDFUNC
```

Listato 3.20: PallettizeInterlockingModule: deposito prodotto con incastro

Capitolo 4

Il sistema di visione

Per visione artificiale si intende l'insieme dei processi che mirano a creare un modello approssimato del mondo reale partendo da immagini bidimensionali. I sistemi di visione attualmente in commercio per applicazioni industriali sono strumenti di utilizzo estremamente semplice ed intuitivo. In questo capitolo si andranno ad illustrare i passaggi per lo sviluppo del software per l'applicazione oggetto della trattazione.

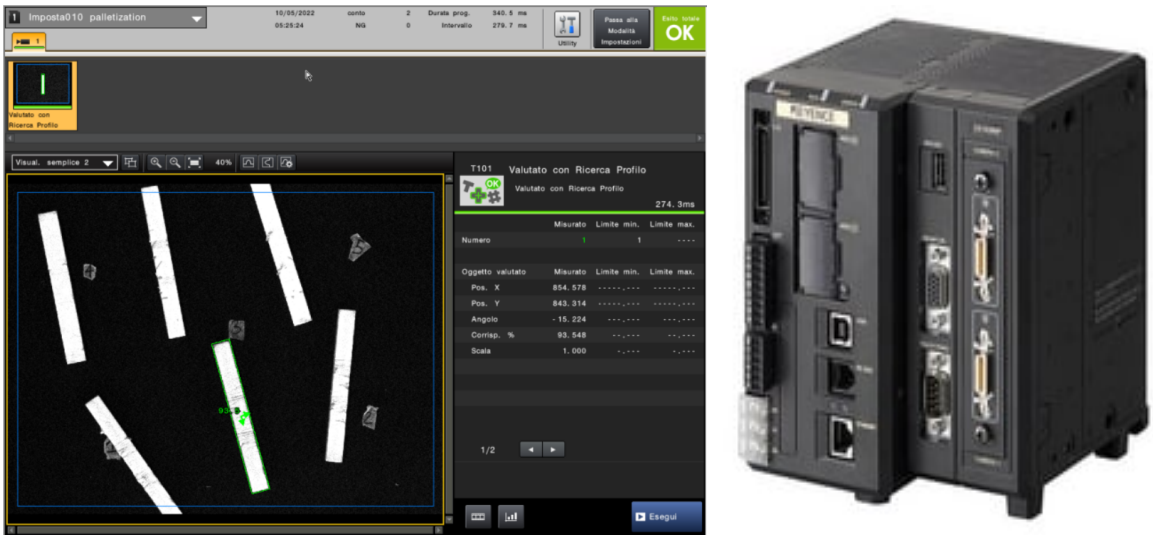
Il sistema di visione in dotazione appartiene alla famiglia cv-x della Keyence. Il suo editor di sviluppo è costituito da un' interfaccia grafica semplice ed intuitiva. Ogni programma è costituito da un set di strumenti disponibili dal catalogo. Ogni strumento può essere personalizzato con diversi filtri, per migliorare la rilevazione, e parametri quali la condizione di giudizio e la percentuale di somiglianza con il modello. Questi strumenti possono essere eseguiti in sequenza, in base a determinate condizioni o in parallelo, in funzione dell'applicazione. Le informazioni così ottenute possono poi essere inviate direttamente al robot o, come nel mio caso, al PLC. Nella impostazioni, sotto la voce *Profinet* è quindi possibile definire quali dati scambiare in ingresso e in uscita, sotto forma di bit o word[5].

4.1 Il software per il formato standard

Nello sviluppo del software per il prodotto monocolore si è proceduto nelle seguenti fasi:

1. Selezionare lo strumento *Valuta con Ricerca Profilo*
2. Scattare una foto dei prodotti
3. Definire il contorno di uno di essi
4. Definire la condizione di giudizio: OK se almeno un prodotto visualizzato, NG in caso contrario
5. Selezionare la percentuale di somiglianza con il modello: 40% è risultato essere il giusto trade-off

Infine è stata definita la configurazione delle uscite da inviare via Profinet al PLC.



(a) (b)

Figura 4.1: **A sinistra** Il software del sistema di visione**A destra** Il cv-x100

4.2 Il software per il formato bicolore

Nella pallettizzazione con incastro era necessario riconoscere la rotazione del prodotto in area di prelievo. Per raggiungere questo obiettivo è stato necessario aggirare la limitazione del sistema di visione in dotazione, non in grado di riconoscere i colori. La soluzione trovata è stata differenziare le due parti del prodotto con una piccola irregolarità di forma nel profilo della parte scura. In questo modo, il sistema di visione è stato in grado di riconoscere la differenza di rotazione dei prodotti rispetto al prodotto utilizzato come modello e inviarle al controllore. Quest'ultimo seleziona tra i dati ricevuti le coordinate del prodotto appartenente al gruppo richiesto dal robot. Nel caso di un'applicazione in cui non fosse stato possibile alterare le caratteristiche del prodotto, si sarebbe dovuto valutare l'acquisto di un sistema di visione più performante o l'applicazione di codici a barre rimovibili in una fase successiva della lavorazione.

4.3 Luminosità e calibrazione

Nell'utilizzo dei sistemi di visione ci si scontra con diverse problematiche. La prima è quella della luminosità in quanto estremamente soggetti alle variazioni luminose. Per questa ragione per l'utilizzo in ambienti con luminosità non controllata, come il magazzino in cui è stato sviluppato il progetto, è necessario regolare frequentemente la sensibilità della camera oppure progettare una regolazione automatica mediate l'uso di una fotoresistenza. La seconda difficoltà è quella della calibrazione: i sistemi di visione necessitano di essere perfettamente calibrati per restituire coordinate corrette. Per questa ragione è stata sviluppata una procedura di calibrazione automatica da lanciare dopo ogni modifica della posizione della camera o dopo lunghi periodi di non utilizzo.

Capitolo 5

L'interfaccia uomo-macchina

I sistemi di controllo industriali richiedono molto spesso la possibilità di variare in opera alcune funzioni del sistema controllato, il monitoraggio delle operazioni da parte degli operatori o ancora la possibilità di inviare comandi al sistema in modo rapido e affidabile. A tale scopo vengono utilizzate le interfacce uomo-macchina HMI (Human Machine Interface), che permettono alluomo di operare e interagire direttamente con le macchine mediante una dashboard opportunamente progettata e programmata. In questo capitolo sarà illustrata l'interfaccia uomo-macchina sviluppata per questa applicazione.

5.1 La pagina Home

L'interfaccia è basata su un template comune a tutte le pagine, composto dalla dalla barra di stato nella parte alta della pagina e una barra bassa di navigazione. Dopo lavvio dellimpianto, il display visualizza il menu principale, denominato HOME. Questo menu è sempre richiamabile da qualsiasi altra pagina mediante la pressione dellapposito tasto disponibile in basso a sinistra.
Nella barra di stato è mostrato il nome della pagina(1), l'utente attualmente connesso(2), la ricetta attiva(3), lo stato del sistema(4), la presenza di allarmi eventuali allarmi o segnalazioni(5), il feedback di comunicazione con il plc(6) e il logo dell'azienda(7). La barra di navigazione(12) permette di raggiungere le varie pagine dell'interfaccia, da sinistra verso destra.

1. La pagina Home
2. La pagina di impostazioni
3. La pagina di manutenzione
4. La pagina delle segnalazioni

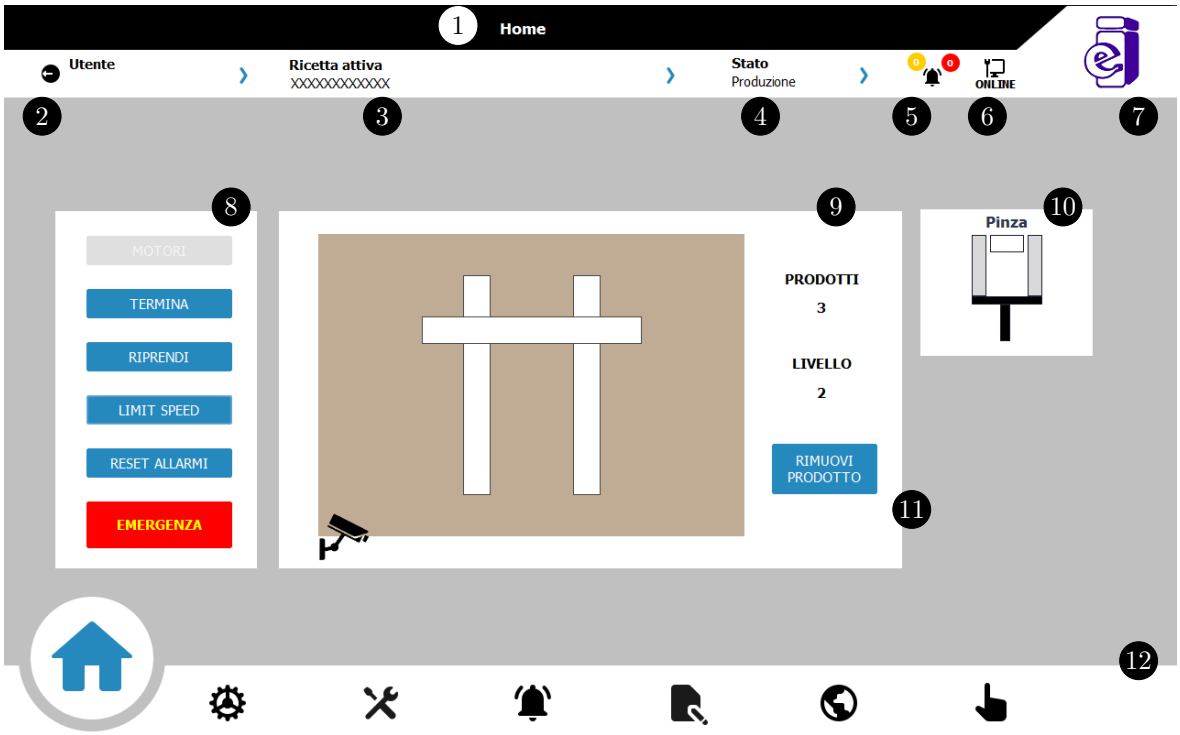


Figura 5.1: La Home page

5. La pagina delle ricette
6. La pagina WorldZone
7. La pagina dei dati macchina

Nella zona centrale sulla sinistra(8) sono disposti i comandi per il robot. Dall'alto verso il basso:

1. Accensione motori
2. Avvia/termina ciclo
3. Arresta/riprendi ciclo
4. Limitazione velocità
5. Arresto di emergenza

Sulla destra(10) è visibile lo stato della pinza e l'eventuale presenza di un prodotto. La parte centrale della pagina è riservata alla visualizzazione dello stato del pallet in costruzione(9): il numero di prodotti attualmente posizionati e il livello raggiunto. Quando il robot non è in movimento è possibile rimuovere un prodotto. L'operatore può selezionare direttamente il prodotto sul pallet oppure, premendo il pulsante(11) alla destra del pallet, aprire la pagina slid-in di selezione in Figura 5.2. Una volta selezionato un prodotto si apre un popup di conferma. Una volta ripreso il ciclo il robot deposita sulla prima posizione libera.

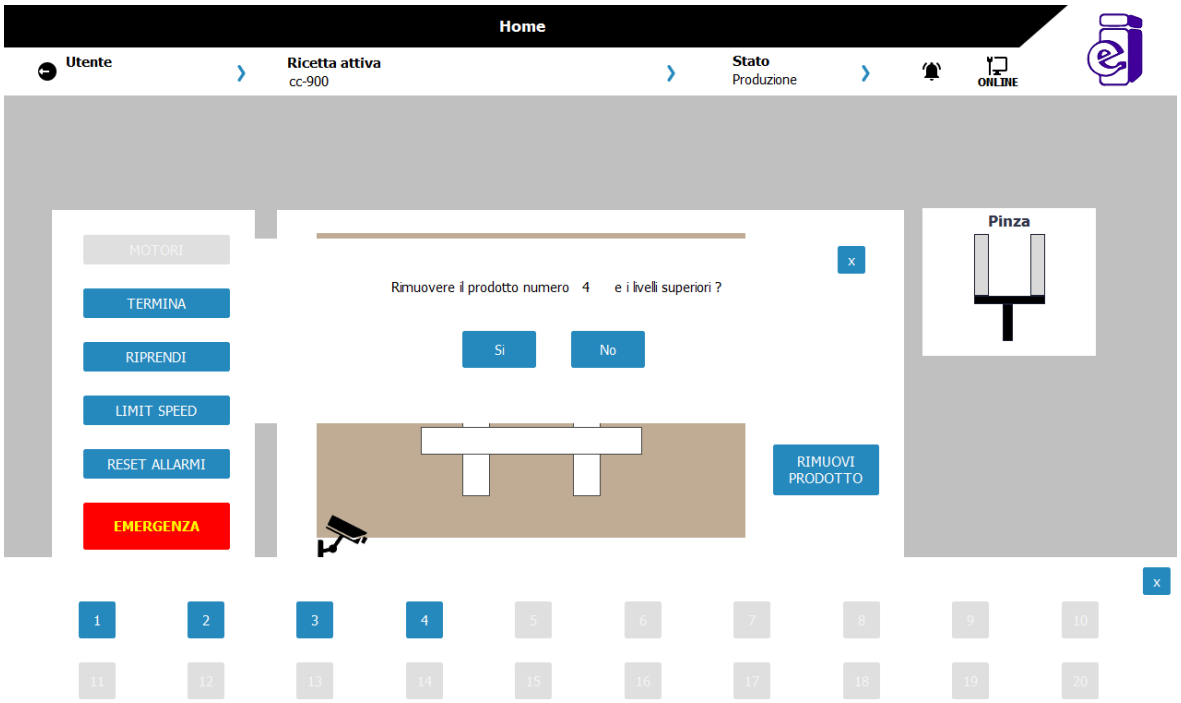


Figura 5.2: La pagina slid-in di rimozione prodotti

5.2 La pagina di manutenzione

Dalla pagina di manutenzione si accede alla pagina di diagnostica e a quella di gestione del sistema di visione. In Figura 5.3 sono visibili, sulla sinistra, i dati riguardanti le statistiche delle elaborazione, il tempo di elaborazione e il codice del programma attualmente presente all'interno del sistema(1). Sulla destra sono presenti i comandi per forzare i segnali in ingresso al cv-x(2)

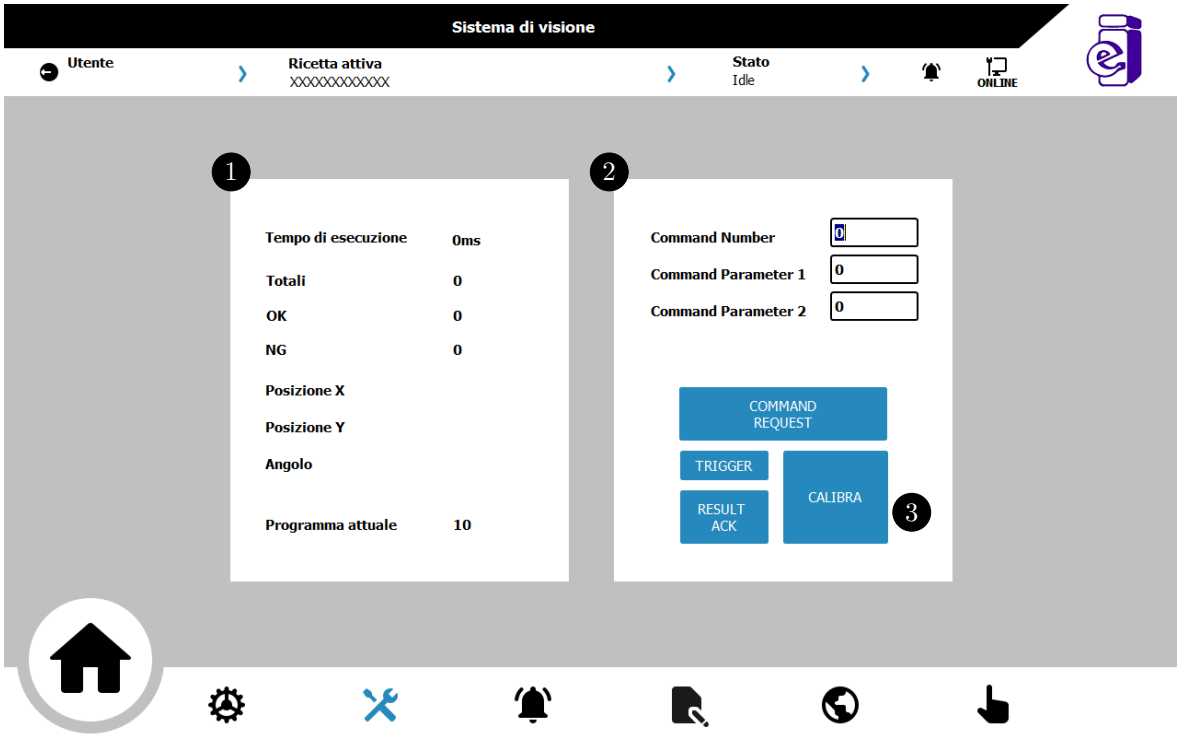


Figura 5.3: La pagina del sistema di visione

Infine con il pulsante CALIBRA(3) è possibile eseguire la procedura di calibrazione per l'aggiornamento dei dati del sistema di riferimento della camera.

5.3 La pagina delle segnalazioni

Accedendo alla pagina delle segnalazioni, premendo lapposito tasto precedentemente descritto, è possibile visualizzare tutte le segnalazioni dovute ad anomalie delle unità del sistema.
Le segnalazioni di allarme possono essere di due tipologie, con possibilità di ripristino automatico e con ripristino manuale. Tutte le segnalazioni abbinate al ripristino automatico scompaiono non appena la causa scatenante dellallarme viene risolta, mentre le segnalazioni con ripristino manuale vengono memorizzate per essere necessariamente visionate e riconosciute dalloperatore: tutti i movimenti legati allallarme in corso rimangono pertanto congelati in modo da consentire un eventuale sopralluogo; in questo caso ciclo riprenderà solo dopo il reset dell'allarme.

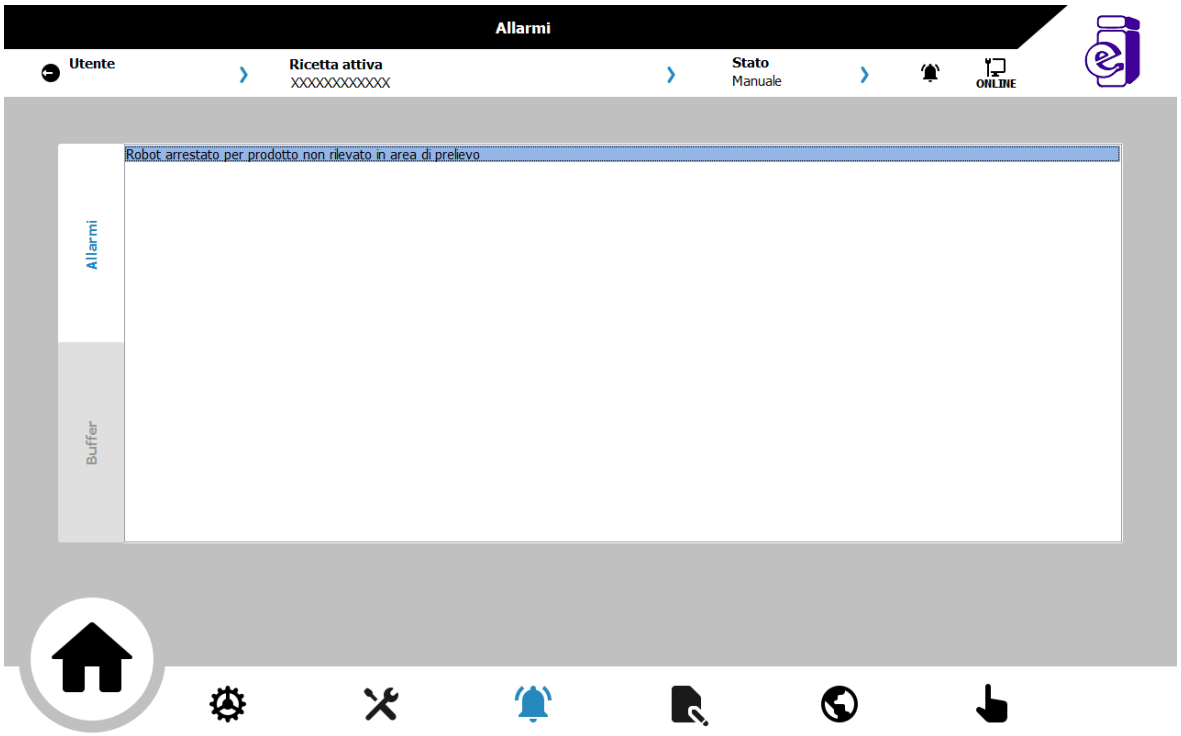


Figura 5.4: La pagina delle segnalazioni

In Figura 5.4 è mostrata una segnalazione di arresto del robot causata dalla mancanza di prodotti nell'area di prelievo.
Per effettuare il reset della segnalazione, è necessario premere il tasto di "Reset Allarmi" nella Home page.

5.4 La pagina delle ricette

Tramite la pagina di gestione delle ricette è possibile selezionare la ricetta in editazione(1) e modificarne i dati. Premendo il pulsante SALVA(2) è possibile salvare nell'archivio i dati della ricetta in editazione. Il pulsante ATTIVA(3) apre il popup di attivazione di una nuova ricetta, mentre il pulsante COPIA(4) alla sua destra apre il popup di copiatura ricette.

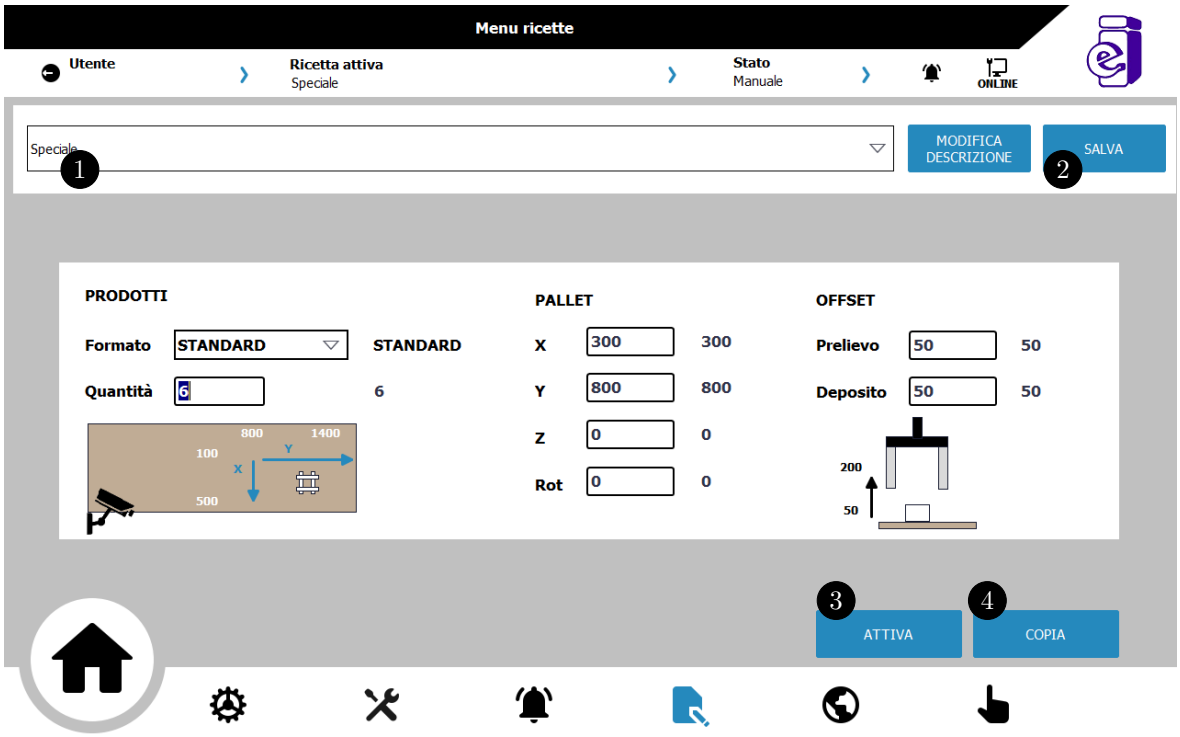


Figura 5.5: La pagina delle ricette

5.5 La pagina WorldZone

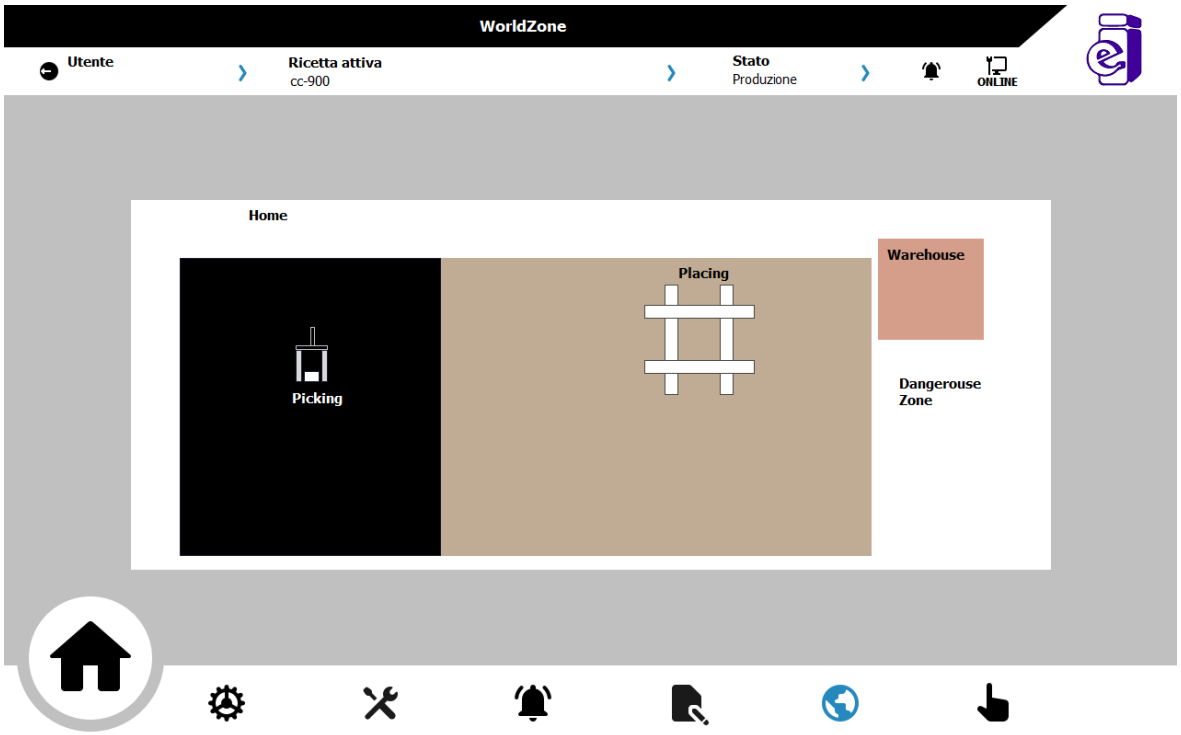


Figura 5.6: La pagina WorldZone

Dalla pagina WorlZone è possibile visualizzare la posizione attuale del TCP del robot all'interno dell' area di lavoro, suddivisa in area di prelievo, deposito, home, magazzino.

5.6 La pagina dei dati macchina

L'ambiente di setup è accessibile premendo l'apposito tasto della barra di navigazione. L'unico dato fisso è la velocità di lavoro del robot che può essere modificata tramite l'apposito slider o i pulsanti alle estremità.

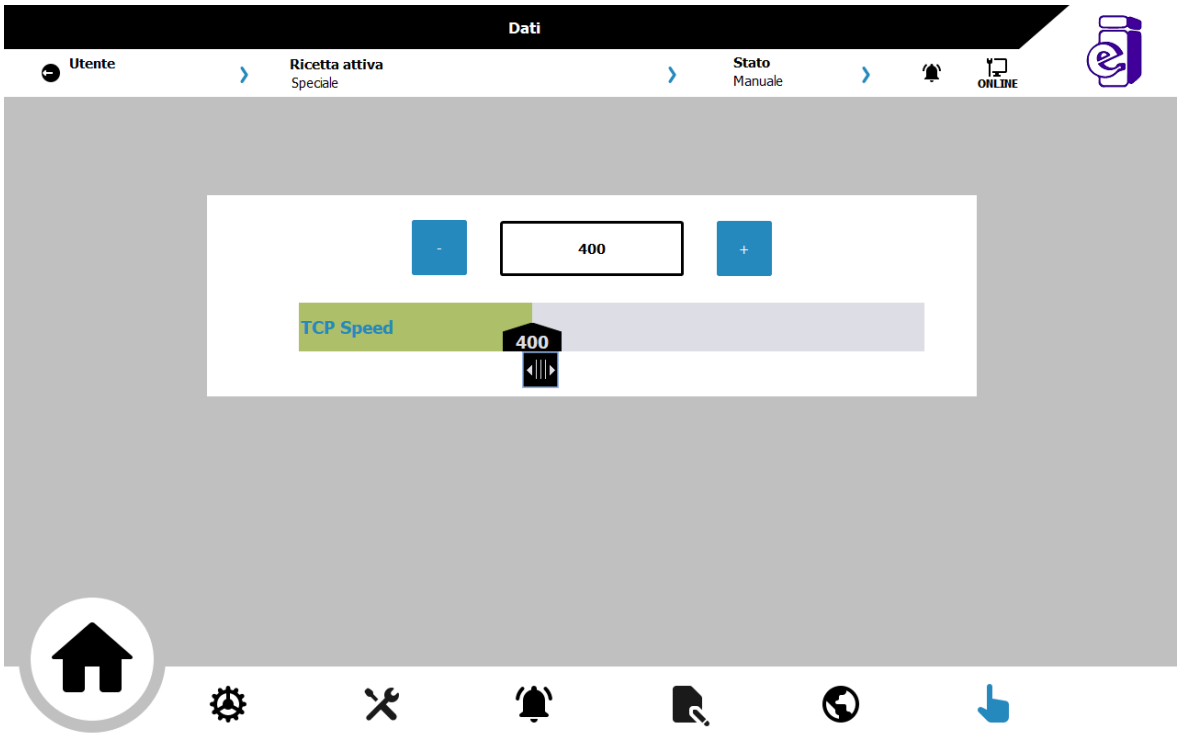


Figura 5.7: La pagina dei dati macchina

Capitolo 6

Conclusioni

In questa trattazione è stato analizzato lo sviluppo di una tipica applicazione di automazione industriale. Partendo dalla programmazione del controllore, è stata descritta la configurazione hardware, lo scambio dati e diversi blocchi di codice che costituiscono il programma. Si è poi passati alla descrizione del software del braccio robotico, riportando dove necessario alcuni accenni teorici alla robotica industriale e al particolare linguaggio di programmazione per i robot ABB. Successivamente sono state illustrate le fasi dello sviluppo del software del sistema di visione, soffermandosi sulle difficoltà incontrate. È stato infine discusso il funzionamento dell'interfacci grafica per la comunicazione tra l'operatore e il sistema. Nello sviluppo si è cercato di proporre una soluzione che fosse il più possibile semplice ed intuitiva per l'operatore.

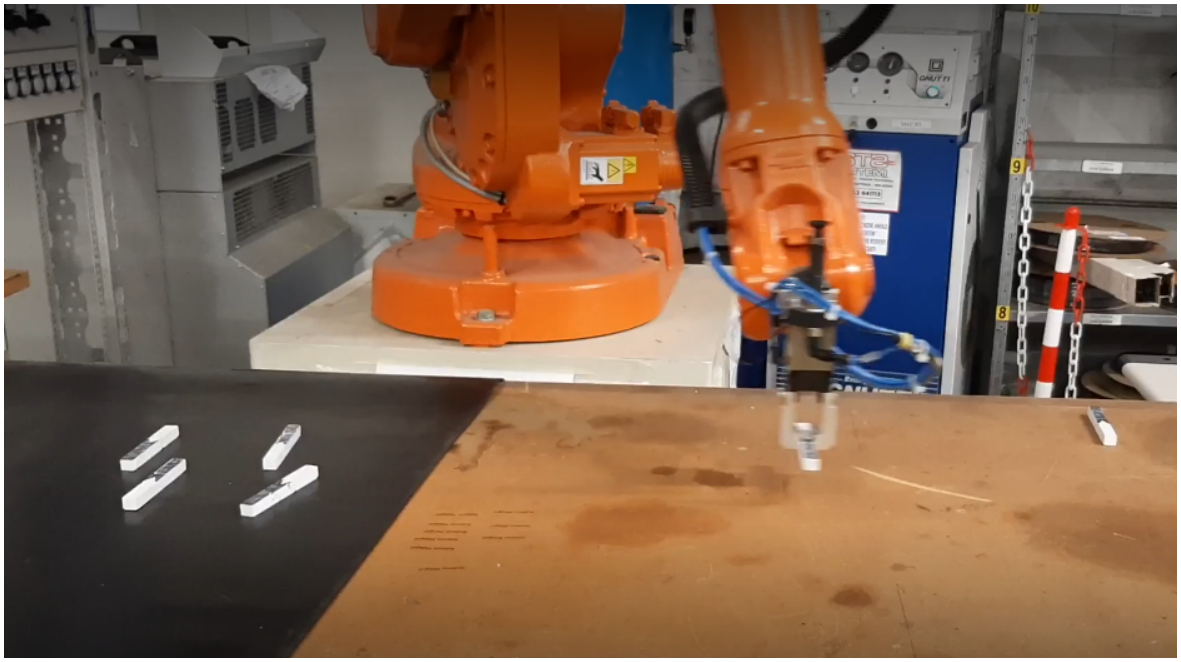


Figura 6.1: Il robot a lavoro

L'applicazione sviluppata si apre a diversi sviluppi futuri come la definizione di un numero superiore di formati e un' ottimizzazione del tempo ciclo.

Acronyms

GSDML General Station Description Markup Language.

HMI Human Machine Interface.

IP Internet Protocol.

PLC Programmable Logic Controller.

SCL Structured Control Language.

TCP Tool Center Point.

Bibliografia

[1] Profinet: lo standard di comunicazione sicuro nelle reti industriali.

[2] ABB. *IRC5 Industrial Robot Controller data-sheet.*

[3] ABB. *Manuale tecnico di riferimento: Istruzioni RAPID, Funzioni e Tipi di dati.*

[4] ABB. *Technical reference manual RAPID kernel.*

[5] Keyence. *CV-X Series, User's Manual.*

[6] Siemens. *Programming Guideline for S7-1200/1500 Programming.*

[7] Siemens. *Structured Control Language (SCL) for S7-300/S7-400 Programming.*