

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 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

1	Introduzione	1
1.1	Obiettivi	1
1.2	Organizzazione dei contenuti	1
2	Il software plc	2
2.1	Lo Structured Control Language	2
2.2	La configurazione hardware	3
2.3	Il protocollo Profinet	3
2.4	Lo scambio dati	3
2.5	La struttura del programma	6
2.5.1	Il blocco OB Main	6
2.5.2	Il blocco FC Inputs	6
2.5.3	Il blocco FC AlarmsHandler	7
2.5.4	Il blocco FC UnitsHandler	8
2.5.5	Il blocco FC StateHandler	14
2.5.6	Il blocco FC RecipeHandler	15
2.5.7	Il blocco FC Communication	16
2.5.8	Il blocco FC Outputs	19
3	Il software del braccio robotico	20
3.1	Introduzione al linguaggio rapid	20
3.1.1	Le routine	20
3.1.2	I tipi di dati	21
3.1.3	Le istruzioni	21
3.2	Il modulo CalibData	21
3.3	Il modulo WorldZoneModule	23
3.4	Il modulo ErrModule	24
3.5	Il modulo ToolModule	25
3.6	Il modulo VisionSysModule	26
3.7	Il modulo InterruptsModule	30
3.8	Il modulo PickAndPlaceModule	32
3.9	Il modulo PallettizeModule	35
3.10	Il modulo PallettizeInterlockingModule	37
4	Il sistema di visione	38
4.1	Il software per il formato standard	38
4.2	Il software per il formato bicolore	39
4.3	Luminosità e calibrazione	39

5	L'interfaccia uomo-macchina	40
5.1	La pagina Home	40
5.2	La pagina di manutenzione	43
5.3	La pagina delle segnalazioni	44
5.4	La pagina delle ricette	45
5.5	La pagina WorldZone	46
5.6	La pagina dei dati macchina	47
6	Conclusioni	48

Elenco delle tabelle

2.1	Dati in ingresso al robot	4
2.2	Dati in uscita dal robot	4
2.3	Dati in uscita dal sistema di visione	5
2.4	Dati in ingresso al sistema di visione	5

Elenco dei listati

2.1	OB1	6
2.2	Inputs()	6
2.3	AlarmsHandler()	7
2.4	La diagnostica della rete Profinet	8
2.5	HMI()	8
2.6	VisionSystem(): salvataggio programma corrente	10
2.7	VisionSystem(): lettura delle uscite del sistema di visione	11
2.8	Robot()	13
2.9	RecipesHandler(): ricetta attiva	15
2.10	RecipesHandler(): ricetta in editazione	16
2.11	RecipeHandler(): copiatura ricette	16
2.12	Communication()	16
2.13	Outputs()	19
3.1	CalibData: wobjTable	22
3.2	CalibData: tGripper	22
3.3	CalibData: pHome	23
3.4	CalibData: gli orientamenti	23
3.5	Esempio di utilizzo di OrientZYX() e EulerZYX()	23
3.6	WorldZoneModule: definizione del cubo della area di prelievo	24
3.7	WorldZoneModule: definizione della World Zone dell'area di prelievo	24
3.8	ErrModule	25
3.9	ToolModule: procedura di chiusura della pinza	26
3.10	ToolModule: procedura di apertura della pinza	26
3.11	VisionSysModule: VisionSysCalib	29
3.12	PickAndPlaceModule: Pick	32
3.13	PickAndPlaceModule: Place	33
3.14	PickAndPlaceModule: SearchProd	34
3.15	PickAndPlaceModule: PickVisionSysRobT	34
3.16	PickAndPlaceModule: PickAndPlace	35
3.17	PallettizeModule: Pallettize	35
3.18	PallettizaModule: PlaceProd	36
3.19	PallettizeInterlockingModule: deposito prodotto con incastro	37

Elenco delle figure

2.1	La configurazione hardware	3
2.2	Utilizzo del Program_Alarm per la visualizzazione di segnalazioni dinamiche	7
4.1	A sinistra Il software del sistema di visione A destra Il cv-x100	39
5.1	La Home page	41
5.2	La pagina slidin di rimozione prodotti	42
5.3	La pagina del sistema di visione	43
5.4	La pagina delle segnalazioni	44
5.5	La pagina delle ricette	45
5.6	La pagina WorldZone	46
5.7	La pagina dei dati macchina	47

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 a formare un incastro.

1.2 Organizzazione dei contenuti

Il presente documento si articola in sei capitoli. Terminata l'iniziale introduzione sul contesto e gli obiettivi che ci si era prefissati, si procede a riportare, nel secondo capitolo, lo sviluppo del software del braccio robotico, soffermandosi sulle peculiarità del linguaggio di programmazione RAPID. Il terzo capitolo è dedicato ad una breve presentazione sul 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. Per poi descrivere nel quinto capitolo il risultato dell'interfaccia grafica sviluppata. Per ultimo, il sesto capitolo esporrà le conclusioni ed analizzerà quali potrebbero essere gli eventuali 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. (*Programmable Logic Controller*).

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 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 Lo Structured Control Language

Tra i diversi linguaggi disponibili si scelto di utilizzare l'SCL *Structured Control Language*, scelta motivata dalla maggiore versatilità e velocità di sviluppo rispetto al classico linguaggio a contatti.

Le istruzioni SCL utilizzano operatori di programmazione standard, ad es. per l'assegnazione ($:=$) e le funzioni matematiche (+ per l'addizione, - per la sottrazione, * per la moltiplicazione e / per la divisione). SCL utilizza anche operazioni standard di controllo del programma PASCAL quali IF-THEN-ELSE, CASE, REPEAT-UNTIL, GOTO e RETURN. È possibile utilizzare qualsiasi riferimento PASCAL per gli elementi sintattici del linguaggio di programmazione SCL. [5]

2.2 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 (General Station Description Markup Language) che contengono le proprietà specifiche dell'apparecchiatura di campo.

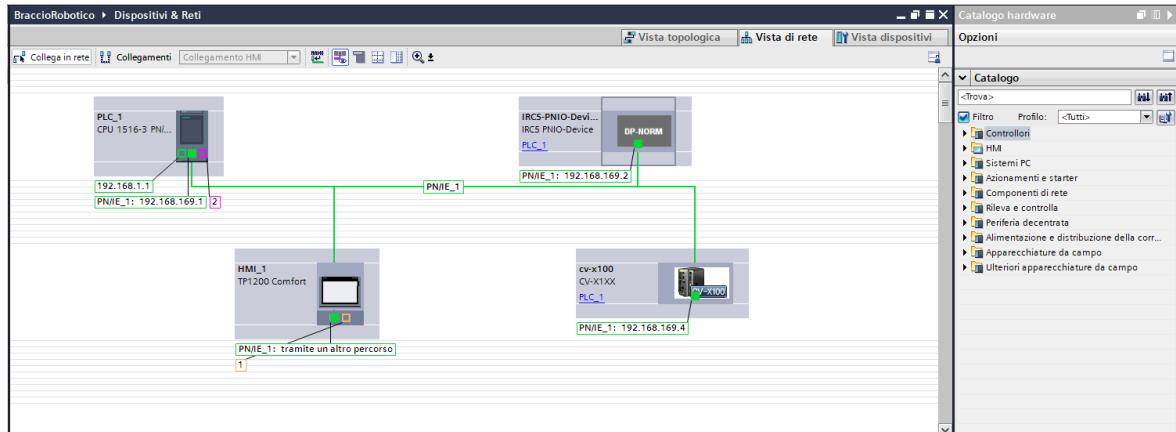


Figura 2.1: La configurazione hardware

2.3 Il protocollo Profinet

Il protocollo di comunicazione standard per i controllori Siemens è il Profinet, un protocollo di comunicazione 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 brevissimi (dell'ordine del millisecondo). Il Profinet utilizza il protocollo TCP/IP: la comunicazione con i dispositivi di campo avviene tramite l'utilizzo di indirizzi IP (Internet Protocol) e di Subnet Mask. Esso permette di collegare tra loro fino a 1024 dispositivi con una velocità di trasmissione di 100 Mbit/s e non ci sono limiti di distanza tra i dispositivi collegati. Queste caratteristiche rendono oggi il Profinet il protocollo più utilizzato nella gestione della comunicazione tra i dispositivi di un impianto e il PLC.[1]

2.4 Lo scambio dati

Il ruolo principale del controllore all'interno del sistema 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 ingresso, mappata a partire dall'indirizzo %I1000.0, è salvata nel dato RobotInput di tipo "typeRobotInput". Allo stesso modo i dati in uscita, dall'indirizzo %Q1000.0, sono scritti sulla variabile RobotInput di tipo "typeRobotOutput".

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 cv-x completed the command
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 coord
palletY	UInt	pallet y coord
palletZ	UInt	pallet z coord
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 the gripper is opened
trgControl	Bool	if TRUE robot request a vision system trig
resultAckFlag	Bool	if TRUE robot has received the vision system result
isReady	Bool	if TRUE robot and vision system are initialized
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 was placed
isStored	Bool	if TRUE a new product was 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 "" e "typeVisionSysOutput" ricalcano lo schema degli ingressi e delle uscite standard del produttore. Le rispettive variabili PLC sono VisionSysInput e VisionSysOutput.

VisionSystemOutput

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

2.5 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. Quest'ultima viene memorizzata nello stack dei dati locali e viene perduta dopo l'elaborazione dell'FC.
- **Blocchi Funzionali (FB):** blocchi programmabili simili alle FC, ma con la possibilità di definire al loro interno variabili con memoria.

È possibile scrivere l'intero programma utente nel OB1 (programmazione lineare) oppure, quando il sistema risulta complesso ed è necessaria una certa formalità e coerenza nell'organizzazione, si richiamano al suo interno diversi sottoprogrammi (programmazione procedurale). Quest'ultima tecnica è stata seguita per compilare il programma dell'applicazione descritta in questa relazione. Il blocco OB1 viene richiamato dal S.O. ad ogni ciclo (scan). È in questo blocco che andranno messe le chiamate agli altri blocchi del programma utente.

In questa sezione sono illustrati i diversi blocchi che costituiscono l'applicazione.

2.5.1 Il blocco OB Main

Ogni Programma per PLC Siemens deve contenere un blocco organizzativo di sottotipo Main chiamato nello stesso modo. Esso rappresenta la prima parte di codice ad essere eseguita all'avvio della CPU e viene richiamato all'inizio di ogni ciclo di scansione.

nel caso in esame, si è scelto di utilizzare il paradigma della programmazione procedurale, richiamando all'interno dell'OB1 i diversi sottoprogrammi che costituiscono l'applicazione.

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

Listato 2.1: OB1

2.5.2 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()

2.5.3 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. La scelta è stata motivata dalla possibilità di visualizzare testi dinamici all'interno della segnalazione, velocizzando la fase di sviluppo e la 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. La variabile visualizzata all'interno della segnalazione è il codice dell'errore contenuto nella variabile "RobotData".Outputs.ErrorNumber. Per gli errori più significativi è stata sviluppata una funzione per convertire l'informazione da numero a stringa per una maggiore chiarezza. Per gli errori standard dei robot ABB si è scelto di rimandare alla documentazione del produttore.

```
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: AlarmsHandler()

Una segnalazione aggiuntiva informa della mancanza di prodotti in aria di prelievo, la quale causa l'arresto del ciclo.

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 e 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(). ,la quale consente di richiamare una determinata informazione di stato concernente tutte le unità in un sistema IO, ad esempio tutti gli IO Device all'interno di un sistema PROFINET IO oppure tutti gli slave DP all'interno di un sistema master DP. Nel listato 2.4 è mostrata la diagnostica degli IO Device/slave DP disturbati della rete "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

2.5.4 Il blocco FC UnitsHandler

In questa FC sono richiamati i blocchi per la gestione delle diverse unità:

Il blocco FC HMI

In questo blocco è gestita la visualizzazione dei prodotti del pallet, dipendente dal formato del prodotto, e la rimozione dei dati dal pallet.

```
1
2 "HMIData".motorOnEnabled := ("StateData".state = "STATE_IDLE") AND (NOT "RobotData".
    outputs.motorOnState);
3
4 IF "StateData".state = "STATE_INIT" THEN
5
6     REGION palletVisualizationReset
7
8         "HMIData".prods := #initializedPallet;
9         "HMIData".pallet1 := #initializedPallet;
10        "HMIData".pallet2 := #initializedPallet;
11        "HMIData".actLevel := 0;
12        "HMIData".actProdsCtn := 0;
13
14    END_REGION
15
16 END_IF;
17
18 IF "StateData".state = "STATE_RUNNING" THEN
19
```

```

20  REGION Pallet Visualization
21
22  // if a product is placed, save his value in prods array
23  REGION countProductPlaced
24
25  // count products and find last product
26  "HMIData".actProdsCtn := 0;
27
28  FOR #i := 1 TO "MAX_PRODUCTS" DO
29
30      IF "HMIData".prods[#i] THEN
31
32          "HMIData".actProdsCtn += 1;
33
34          #lastprod := #i;
35
36      END_IF;
37
38  END_FOR;
39
40  END_REGION
41
42  // find current level
43  "HMIData".actLevel := "CalcLevel"(prod := #lastprod, dim := 2);
44
45  REGION Update Products Placed
46
47  //if index is in range, update the corresponding array element
48
49  IF "RobotData".outputs.index >= 1 AND "RobotData".outputs.index <= 20 THEN
50
51      IF "RobotData".outputs.isPlaced THEN
52
53          //if a product is placed, increase array
54
55          "HMIData".prods["RobotData".outputs.index] := TRUE;
56
57      ELSIF "RobotData".outputs.isStored THEN
58
59          //if a product is stored, reduce array
60
61          "HMIData".prods["RobotData".outputs.index] := FALSE;
62      END_IF;
63
64  END_IF;
65
66  END_REGION
67
68  REGION Remove Products
69
70  //If a product is removed by HMI, clear the corresponding array element
71  IF "HMIData".robotInputs.interrupts.removeProd THEN
72      //check if index is in range!
73      IF "HMIData".robotInputs.removedProd >= 1 AND "HMIData".robotInputs.
removedProd <= "MAX_PRODUCTS" THEN
74      // if removed product isn't in the last level, remove all superior
products
75      REGION Remove Superior Products

```



```

76          //the start level to remove is the first one over the removed
product level
77          #level := "CalcLevel"(prod := "HMIData".robotInputs.
removedProd, dim := "RobotData".inputs.word3);
78          //ex: level=2 dim=2 => startprod=5
79          #startprod := (#level * "RobotData".inputs.word3) + 1;
80          //check if index is in range!
81          IF #startprod >= 1 AND #startprod <= "MAX_PRODUCTS" THEN
82
83              FOR #i := #startprod TO "MAX_PRODUCTS" DO
84                  "HMIData".prods[#i] := FALSE;
85              END_FOR;
86
87          END_IF;
88
89      END_REGION
90
91      "HMIData".prods["HMIData".robotInputs.removedProd] := FALSE;
92
93      END_IF;
94
95      END_IF;
96
97      END_REGION
98
99      CASE "RobotData".inputs.productsType OF
100      0:
101          "HMIData".pallet1 := "HMIData".prods;
102
103      1:
104          "HMIData".pallet2 := "HMIData".prods;
105      END_CASE;
106
107      END_REGION

```

Listato 2.5: HMI()

Il blocco FC VisionSys

In questo blocco è gestita la comunicazione con il sistema di visione.

Ogni formato ha un rispettivo programma all'interno del sistema di visione. Al variare del programma viene salvato il suo valore nella variabile "VisionSystemData".actProg.

```

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 programma corrente

In base al valore di "VisionSystemData".actProg vengono letti diversi dati in uscita dal sistema di visione.

```

1
2 REGION Read Vision System Program 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 la comunicazione con il robot, dipendente dalla missione in esecuzione. Per il formato standard vengono sempre inviate le coordinate di prelievo del primo prodotto rilevato. Per la pallettizzazione con in castro il sistema di visione invia i dati di tutti i prodotti al plc, il quale seleziona le coordinate del primo prodotto appartenente al gruppo richiesto dal robot. In caso di ricerca fallita, viene inviato al robot un giudizio negativo. Infine per la procedura di calibrazione vengono inviati in sequenza i punti calibrati per definire il sistema di riferimento all'interno del robot.

```
1 IF "StateData".state = "STATE_RUNNING" THEN
2
3     CASE "RobotData".inputs.mission OF
4
5         "PROGRAM_PALLET_INTERLOCKING":
6
7             REGION InterlockingProgram
8
9                 //Build an interlocking pallet.
10                //The products are divided in two groups based on
11                //their angle (90-(-90))
12
13                //If Vision system detect at least one products,
14                //checks all products to find one
15
16                IF "VisionSystemData".outputs.toolJudgeValue[0] THEN
17
18                    REGION "SelectPickCoordinates"
19
20                    //initalize loop index and set judge value false.
21                    //If no product is found, judge value will remain false,
22                    //otherwise it will become true
23
24                    #i := 0;
25                    "VisionSystemData".outputs.toolJudgeValue[0] := false;
26
27                    //loop to find the right product to pick
28                    //loop ends when all products are checked or a right product
29                    is found
30
31                    WHILE #i < "VisionSystemData".prodsCtn AND NOT "
32                    VisionSystemData".outputs.toolJudgeValue[0] DO
33
34                        //Check if a product belongs to the right group based on
35                        its angle
36
37                        #isGroup1 := NOT ("VisionSystemData".arrayCoordinates[#i].
38                        angle / 1000 >= -90
39
40                        AND "VisionSystemData".
41                        arrayCoordinates[#i].angle / 1000 <= 90);
42
43                        "VisionSystemData".outputs.toolJudgeValue[0] := ("
44                        RobotData".outputs.isGroup1 = #isGroup1);
45
46                    #i := #i + 1;
47                END IF
48            END REGION
49        END CASE
50    END IF
```

```

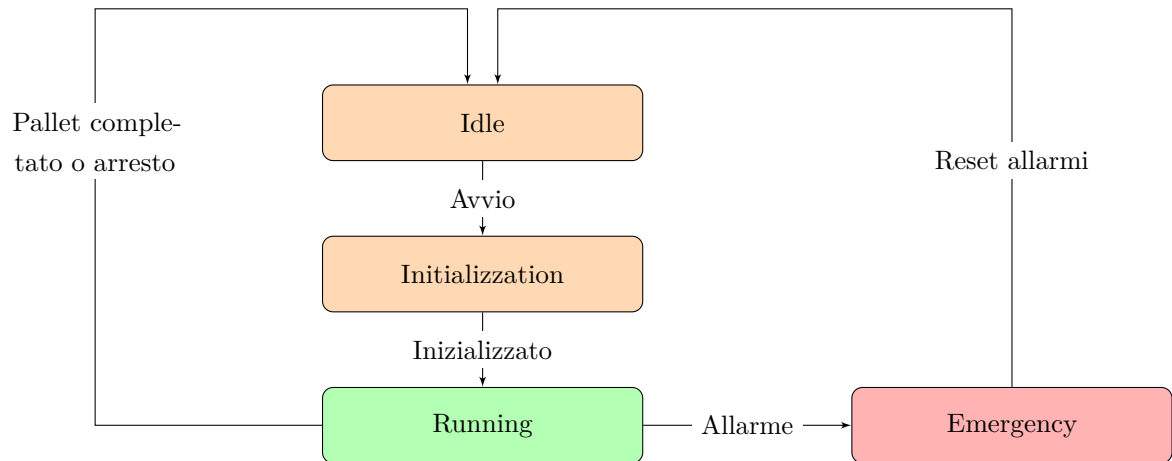
40         //if a right product is found, save coordinates for robot
41         IF "VisionSystemData".outputs.toolJudgeValue[0] THEN
42
43             "VisionSystemData".coordinates := "VisionSystemData".
arrayCoordinates[#i];
44
45         END_IF;
46
47         #i := #i + 1;
48
49         END_WHILE;
50
51         END_REGION
52
53         END_IF;
54
55         END_REGION
56
57         "PROGRAM_CALIBRATION":
58
59         REGION CalibrationProgram
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()

2.5.5 Il blocco FC StateHandler

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



2.5.6 Il blocco FC RecipeHandler

Per rendere il sistema personalizzabile dal cliente finale sono state predisposte 20 ricette. Per la loro gestione è stata definita l'UDT "typeRecipe" con i seguenti dati:

- 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 e editabili dall'operatore da HMI. Quando la macchina non si trova in modalità manuale è possibile modificare la ricetta attiva:

```

1 REGION Recipe Active
2
3 IF "Machine".Auto THEN
4
5     "Hmi".Recipe.ActiveNumberFromHmi := "Hmi".Recipe.ActiveNumber;
6
7 ELSE
8
9     "Hmi".Recipe.ActiveNumber := "Hmi".Recipe.ActiveNumberFromHmi;
10
11 END_IF;
12
13 "RecipeActive" := "Recipes".Recipe["Hmi".Recipe.ActiveNumber];
14
15
16 END_REGION
  
```

Listato 2.9: RecipesHandler(): ricetta attiva

È sempre possibile selezionare e editare le ricette. Quando una ricetta viene selezionata per l'editazione i dati vengono caricati dall'archivio nella struttura "RecipeEdit"; premendo il pulsante a pannello "Save" i dati della ricetta in editazione vengono salvati nell'archivio.

```

1 REGION Edit to Recipes
2
3 IF "Hmi".Recipe.Save THEN
4
5     "Recipes".Recipe["Hmi".Recipe.EditNumber] := "RecipeEdit";
6     "Hmi".Recipe.Save := false;
7
8 END_IF;
9
10 END_REGION
11
12 REGION Recipes to Edit
13
14 IF "Hmi".Recipe.EditNumber <> "Hmi".Recipe.EditNumberFromHmi THEN
15
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 ricetta sorgente e destinazione corrispondono la funzione è disabilitata.

```

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

```

Listato 2.11: RecipeHandler(): copiatura ricette

2.5.7 Il blocco FC Communication

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

```

Listato 2.12: Communication()

2.5.8 Il blocco FC Outputs

In questo blocco i dati globali vengono scritti sulla periferia.

```

1
2 "visionSystemInput" := "VisionSystemData".inputs;
3 "robotInput" := "RobotData".inputs;

```

Listato 2.13: Outputs()

Capitolo 3

Il software del braccio robotico

In questo capitolo sarà descritto il programma del braccio robotico. Per una più chiara comprensione si è deciso di accennare dove necessario alla teoria del linguaggio RAPID, un linguaggio di programmazione di alto livello utilizzato per controllare i robot industriali ABB. Si andranno poi a illustrati i diversi moduli e le routine contenute.

3.1 Introduzione al linguaggio rapid

Il robot a disposizione per l'applicazione oggetto della tesi è IRB1600.6.1,45 con un (Payload) di 6Kg ed estensione del braccio di 1,45m. [2]

Il linguaggio di programmazione standard per questa marca di robot è il Rapid, un linguaggio testuale che assomiglia al Python o a una versione ridotta di un linguaggio C. Sono disponibili tutte le istruzioni principali, come FOR, WHILE, CASE e IF. Sono disponibili tipi di dati standard, array e tipi di dati definiti dall'utente (RECORD). Di seguito una breve introduzione agli elementi di questo linguaggio. Un programma rapid è costituito tipicamente da uno o più moduli, i quali possono comprendere dati e routine diverse. È possibile copiare ciascun modulo, o l'intero programma, su disco, e così via e viceversa. Uno dei moduli contiene la procedura di ingresso, una procedura 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:

- Un tipo di dati atomico è tale, poiché che 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 composito con componenti nominati ed ordinati, ad esempio pos. Un componente può essere di tipo atomico o record. Un valore record può essere espresso utilizzando una rappresentazione aggregata, ad esempio: valore aggregato record pos [300, 500, depth]. Ad uno specifico componente di dati si può accedere utilizzando il nome di tale componente, ad esempio: pos1.x := 300; attribuzione del componente x di pos1.
- Un tipo di dati alias è, per definizione uguale ad un altro tipo. I tipi alias rendono possibile la classificazione di oggetti di dati.

3.1.3 La istruzioni

Come già accennato in precedenza, il linguaggio RAPID deriva dal linguaggio strutturato, da cui deriva le istruzioni standard quali IF, FOR, WHILE, TEST (corrispondente al CASE).

I linguaggi di programmazione robot si contraddistinguono inoltre 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 modulo CalibData

In questo modulo sono definiti i dati necessari alla definizione delle traiettorie. Per i definire i punti di una traiettoria è infatti necessario definire il sistema di riferimento a cui riferiscono i punti e dati del TCO (*Tool Center Point*), 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 corrispondono 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 sufficiente aggiustare i dati del 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° 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° su X e -90° su Z mentre le coordinate della relativa origine sono $x = 0$, $y = 200$ e $z = 30$ mm nel sistema di coordinate utente.

```

1
2  PERS wobjdata wobjTable := [FALSE, TRUE, "", [[690, -506, 88], [0.7071, 0, 0, -0.7071]],
3                                [[0, 0, 0], [0, -0.7071, 0.7071, 0]]];

```

Listato 3.1: 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 per la mia applicazione era una pinza pneumatica con i seguenti valori:

- 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.
- Le direzioni dell'utensile sono ruotate in angoli di Eulero ZYX di -180° su Z e 90° 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 2,8 mm sull'asse y del sistema di coordinate del polso,
- Il momento di inerzia è trascurabile.

```

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 3.2: CalibData: tGripper

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 3.3 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 3.3: 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 3.4: 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 forse 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 OrientZYX() 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 3.5: Esempio di utilizzo di OrientZYX() e EulerZYX()

Per una maggiore leggibilità sono state definite due costanti per rappresentare l'orientamento, vedi il Listato 3.4, utilizzato per tutto il codice.

3.3 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 durante l'esecuzione del programma sia durante il movimento manuale.

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 viene utilizzata per definire la zona e attivare una World Zone per la supervisione dei movimenti del robot. Dopo avere eseguito questa istruzione, quando il TCP del robot o il robot e gli assi esterni (zona nei giunti) si trovano all'interno della World Zone definita o la stanno per raggiungere, un segnale di output viene impostato sul valore specificato.

```

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 3.6: 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 mondo, è stata utilizzata la funzione posTableToWCS() mostrata nel Listato 3.7, 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 mondo.

```

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 3.7: WorldZoneModule: definizione della World Zone dell'area di prelievo

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

3.4 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 di 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 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. Inoltre, è possibile utilizzare l'istruzione `RAISE` per creare un errore nel programma e, successivamente, per chiamare un gestore errori della routine. Con questa istruzione è inoltre possibile propagare un errore da un gestore errori, a partire da una catena di chiamate nidificate fino a un livello superiore, in un'unica operazione.

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 3.8: 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`.

3.5 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, la quale controlla le 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 3.9: 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 3.10: ToolModule: procedura di apertura della pinza

3.6 Il modulo VisionSysMoule

Per gestire la comunicazione con il sistema di visione è stata sviluppata una libreria di routine. Di seguito una veloce presentazione delle stesse.

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      PROC VisionSysTrig(\num nDelay)
2
3          IF diResultReady=1 VisionSysResetTrig;
4
5          IF Present(nDelay) WaitTime nDelay;
6
7          PulseD0\high,doTrgControl;
8
9      ERROR
10         RAISE ;
11     ENDPROC

```

La procedura VisionSysSendCommand

Invia un comando al sistema di visione e ritorna 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;
19     WaitDI diCommandCompleteFlag,high\MaxTime:=30;
20     Reset doCommandRequestFlag;
21
22     ERROR
23         RAISE ;
24
25     ENDPROC

```

La funzione VisionSysCProgram

Restituisce il programma attualmente presente nel sistema di visione.

```

1      FUNC dnum VisionSysCProgram()
2
3      VAR dnum CProgram;

```

```

4
5     VisionSysSendCommand READ_PROG_COMMAND;
6     Cprogram:=GInputDnum(giCommandData2);
7
8     RETURN CProgram;
9
10    ERROR
11        RAISE ;
12
13    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. Se il programma richiesto è attualmente presente, ne resetta le statistiche.

```

1     PROC VisionSysSetProgram(dnum nProgram)
2
3     Reset doTrgControl;
4     Reset doResultACK;
5
6     IF diRunMode=0 THEN
7         VisionSysSendCommand RUN_MODE_COMMAND;
8     ENDIF
9
10    IF VisionSysCProgram() <> nProgram THEN
11        Set doChangingVisionSysProg;
12        VisionSysSendCommand CHANGE_PROG_COMMAND,
13            \nParameter1:=1,\nParameter2:=nProgram;
14        Reset doChangingVisionSysProg;
15    ELSE
16        VisionSysSendCommand RESET_COMMAND;
17    ENDIF
18
19    VisionSysTrig;
20
21    ERROR
22        RAISE ;
23
24    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 si divisione a quello del tavolo. La funzione poseMult restituisce il risultato della trasformazione.

```

1     FUNC robtarget VisionSysRobT()
2
3     peProd:=PoseMult(peVisionSysToTable,[[ (aiPosx/1000)
4     *nresolution,(aiPosY/1000)*nresolution,0],
5     OrientZYZ(aiRotz/1000,0,0)]);
6
7     RETURN [peProd.trans,peProd.rot,[1,0,0,0],
8     [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
9
10    ERROR

```

```

11      RAISE ;
12  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 dopo aver acquisito una serie di destinazioni in due posizioni diverse. Per definire un sistema di riferimento sono sufficienti tre destinazioni, ma per una maggiore accuratezza è necessario utilizzare un array di 5 punti. Dopo aver impostate il corrispettivo programma nel sistema di visione, viene scatta una foto all'area di prelievo e inviati i dati dei punti fissi di calibrazione precedentemente salvati nell'array pCam. Le coordinate dei punti sono quindi convertite nel sistema di riferimento mondo e utilizzate per definire il sistema di riferimento del sistema di visione.

```

1
2  CONST robtarget pCam{5}:=[pCam1,pCam2,pCam3,pCam4,pCam5];
3
4  PERS robtarget pCamVS{5}:=[[[66.7898,168.901,0],[1,0,0,0],[1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],
5                                [[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]],
6                                [[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]],
7                                [[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]],
8                                [[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]]];
9
10 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]],
11                                [[1079,84,93],[1,0,0,0],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],
12                                [[1033,-8,93],[1,0,0,0],[-1,-1,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],
13                                [[1005,-104,93],[1,0,0,0],[-1,-1,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],
14                                [[972,65,93],[1,0,0,0],[0,0,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]];
15
16
17 PERS pos psVisionSysToTable:=[-346.134,419.617,0];
18
19
20 PROC VisionSysCalib()
21
22     CONST num nCalibPoints:=5;
23
24     VisionSysSetProgram N_CALIBRATION_PROG;
25     SearchProd;
26
27     FOR i FROM 1 TO nCalibPoints DO
28         SetGO goIndex,i-1;
29
30         ! get point in Word coordinate system
31         MoveL pCam{i},vfast,fine,tGripper,\WObj:=wobjTable;
32         pCamWCS{i}:={CRobT(\Tool:=tgripper\WObj:=wobj0);

```

```

33         pCamWCS{i}.rot:=[1,0,0,0];
34
35         ! Get point in vision system Coordinate system
36         waituntil GOutput(goIndex)=giIndex;
37         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]];
38     ENDFOR
39
40     peVisionSys:=DefAccFrame(pCamVS,pCamWCS,nCalibPoints,nmaxErr,nmeanErr);
41
42     mvHome;
43
44     ERROR
45     RAISE ;
46
47
48     ENDPROC

```

Listato 3.11: VisionSysModule: VisionSysCalib

3.7 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 il robot si trova all'interno del sistema di visione durante la fase di prelievo, 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
9
10

```

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
9
10

```

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
25

```

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         PulseDO\High,doTrgControl;
6     ENDTRAP
7
8

```

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         setAlarm 204;
6         ExitCycle;
7     ENDTRAP
8
9

```

3.8 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 3.12: 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 3.13: 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 una elaborazione. Dopo 3 elaborazioni ritorna un 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 3.14: 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

```

1
2  PROC PickVisionSysRobT(num nZOffset)
3
4  VAR robtarget pProd;
5  VAR num nRotZ;
6
7  SearchProd;
8  pProd:=VisionSysRobT();
9
10 nRotZ:=EulerZXY(\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:=OrientZXY(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 3.15: 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 3.16: PickAndPlaceModule: PickAndPlace

3.9 Il modulo PallettizeModule

In questo modulo è sviluppata la pallettizzazione vero e propria, utilizzando le routine e dati presentate 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 Pallettize(robtarget p0)
3
4         VisionSysSetProgram N_PALLETITIZE_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 3.17: PallettizeModule: Pallettize

La funzione PlaceProd

Di seguito la funzione PlaceProd, la quale ritorna TRUE se è stato depositato un nuovo prodotto, FALSE se il ogni elemento dell'array del pallet risulta completo.

```

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 3.18: PallettizaModule: PlaceProd

3.10 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, che si devono alternare tra loro. Per questo motivo si è mantenuta invariata la struttura principale, 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 ininterruzione del ciclo e successiva rimozione di un prodotto, viene controllato lo stato della pinza: se contiene un prodotto non appartenente al gruppo richiesto, viene scartato e il ciclo riprende da un nuovo 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 3.19: 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.

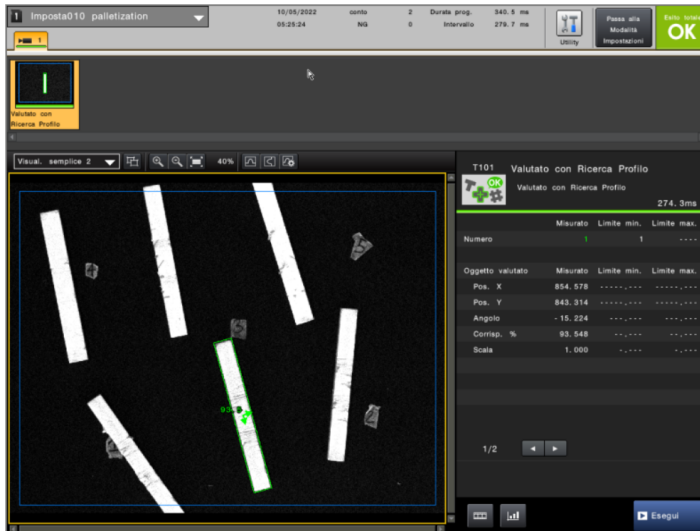
Il sistema di visione in dotazione appartiene alla famiglia cv-x della Keyence. Il suo editor di sviluppo è costituito da un'unica interfaccia grafica semplice ed intuitiva. Ogni programma è costituito da un set di strumenti disponibili dal catalogo. Ogni strumento può essere personalizzato con diversi 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. Nelle impostazioni, sotto la voce *Profinet* è quindi possibile definire quali dati scambiare in ingresso e in uscita, sotto forma di bit o word.

4.1 Il software per il formato standard

Nello sviluppo del software per il prodotto monocolori 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 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. In caso di applicazioni in cui non è possibile alterarne le caratteristiche si sarebbe potuto aggirare il problema con l'utilizzo di un sistema di visione più performante o con l'applicazione di codici a barre rimovibili in una fase successiva.

4.3 Luminosità e calibrazione

Nell'utilizzo dei sistemi di visioni ci si scontra con diverse problematiche. La prima è quella della luminosità: essi infatti sono estremamente soggetti alle variazioni luminose. per questa ragione per l'utilizzo in un magazzino con una luminosità non controllata è necessario sempre regolare la sensibilità prima di ogni utilizzo.

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 all'uomo 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 barra di stato nella parte alta della pagina e una barra bassa di navigazione. Dopo l'avvio dell'impianto, il display visualizza il menu principale, denominato HOME. Questo menu è sempre richiamabile da qualsiasi altra pagina mediante la pressione dell'apposito 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

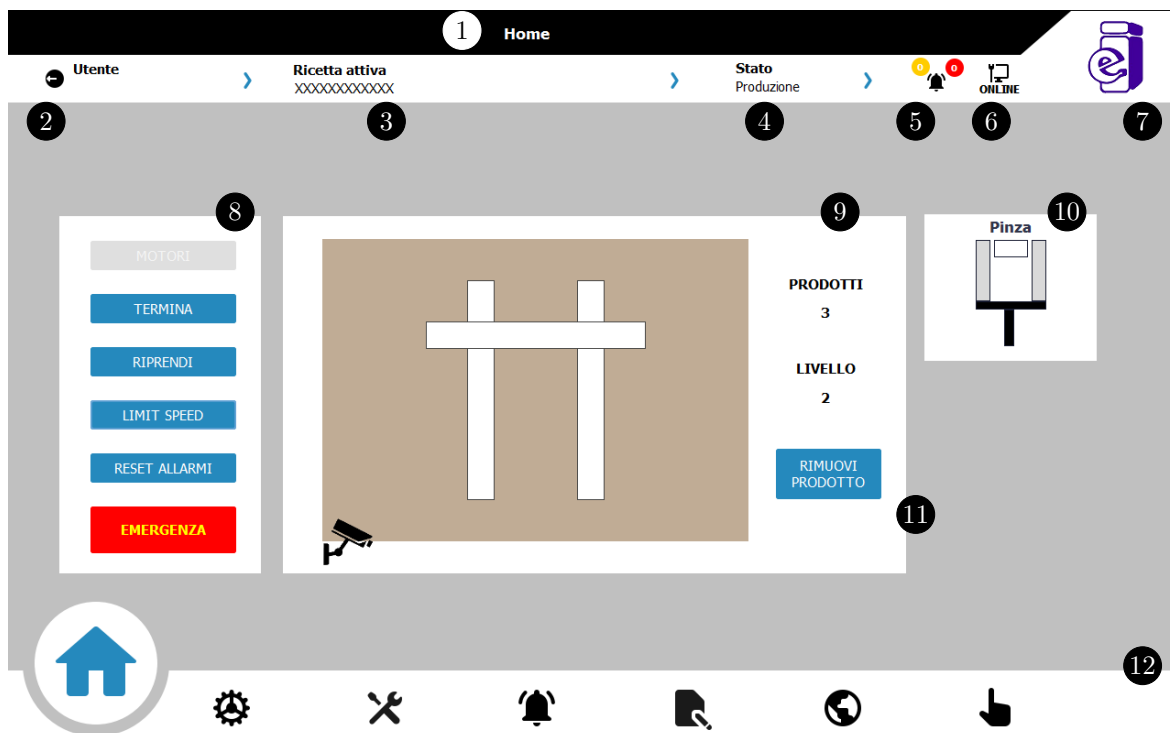


Figura 5.1: La Home page

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 o l'eventuale presenza di un prodotto. La parte centrale della pagina è riservata alla visualizzazione dello stato del pallet in costruzione(9): 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 slidin 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.

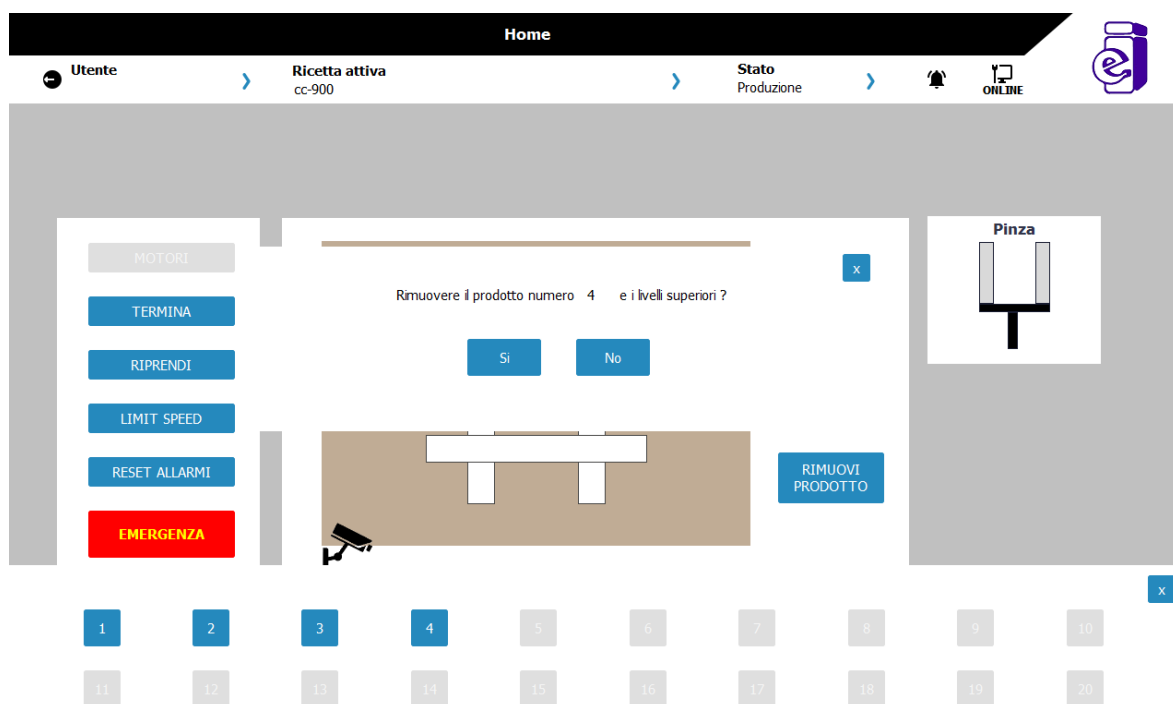


Figura 5.2: La pagina slidin di rimozione prodotti

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 l'apposito 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 dell'allarme viene risolta, mentre le segnalazioni con ripristino manuale vengono memorizzate per essere necessariamente visionate e riconosciute dall'operatore: tutti i movimenti legati all'allarme in corso rimangono pertanto congelati in modo da consentire un eventuale sopralluogo; in questo caso ciclo riprenderà solo dopo il reset dell'allarme.

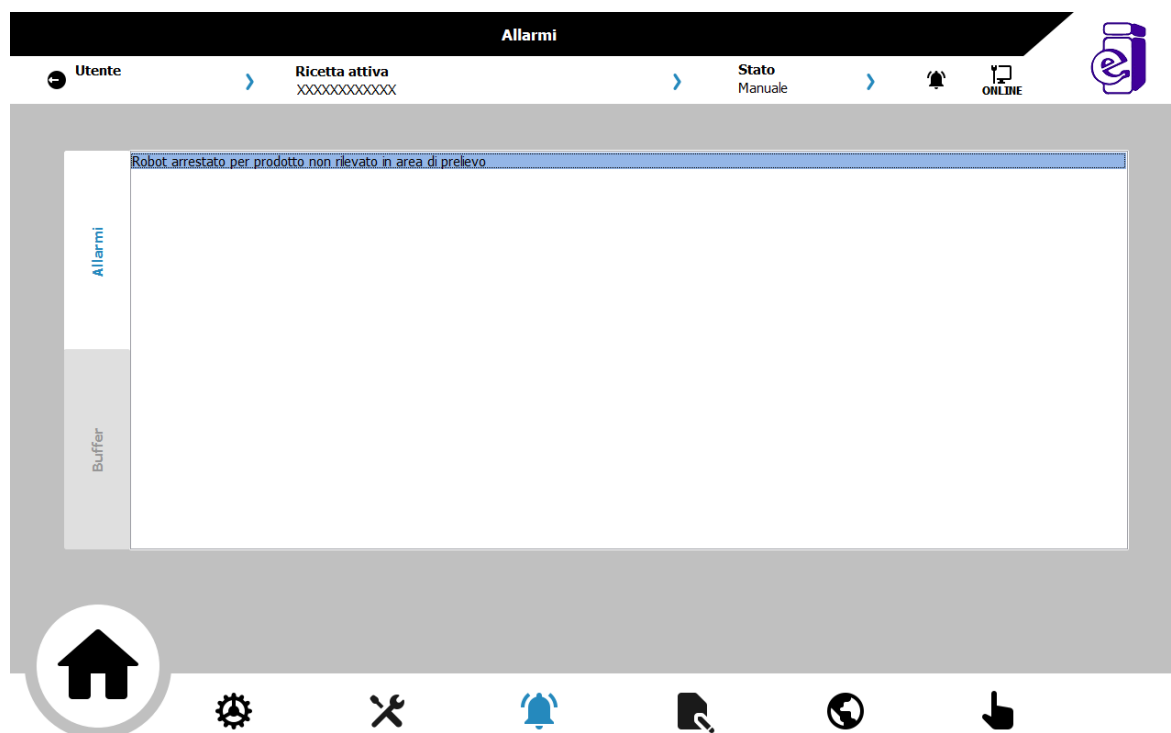


Figura 5.4: La pagina delle segnalazioni

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) ed 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.

The screenshot displays the 'Menu ricette' (Recipe Menu) interface. At the top, a navigation bar includes 'Utente', 'Ricetta attiva Speciale', 'Stato Manuale', and an 'ONLINE' status indicator. A search bar labeled 'Speciale' is marked with a circled '1'. To its right are buttons for 'MODIFICA DESCRIZIONE' and 'SALVA', with the latter marked with a circled '2'. The main content area is divided into three sections: 'PRODOTTI' (Products), 'PALLET', and 'OFFSET'. The 'PRODOTTI' section includes a 'Formato' dropdown set to 'STANDARD', a 'Quantità' input field set to '6', and a diagram of a pallet layout with dimensions (100, 500, 800, 1400) and arrows indicating movement. The 'PALLET' section has input fields for 'X' (300), 'Y' (800), 'Z' (0), and 'Rot' (0). The 'OFFSET' section has input fields for 'Prelievo' (50) and 'Deposito' (50), along with a diagram of a robotic arm. At the bottom, a navigation bar features a home icon, a settings gear, a wrench, a bell, a document, a globe, and a hand icon. On the right side of this bar are buttons for 'ATTIVA' (marked with a circled '3') and 'COPIA' (marked with a circled '4').

Figura 5.5: La pagina delle ricette

5.5 La pagina WorldZone

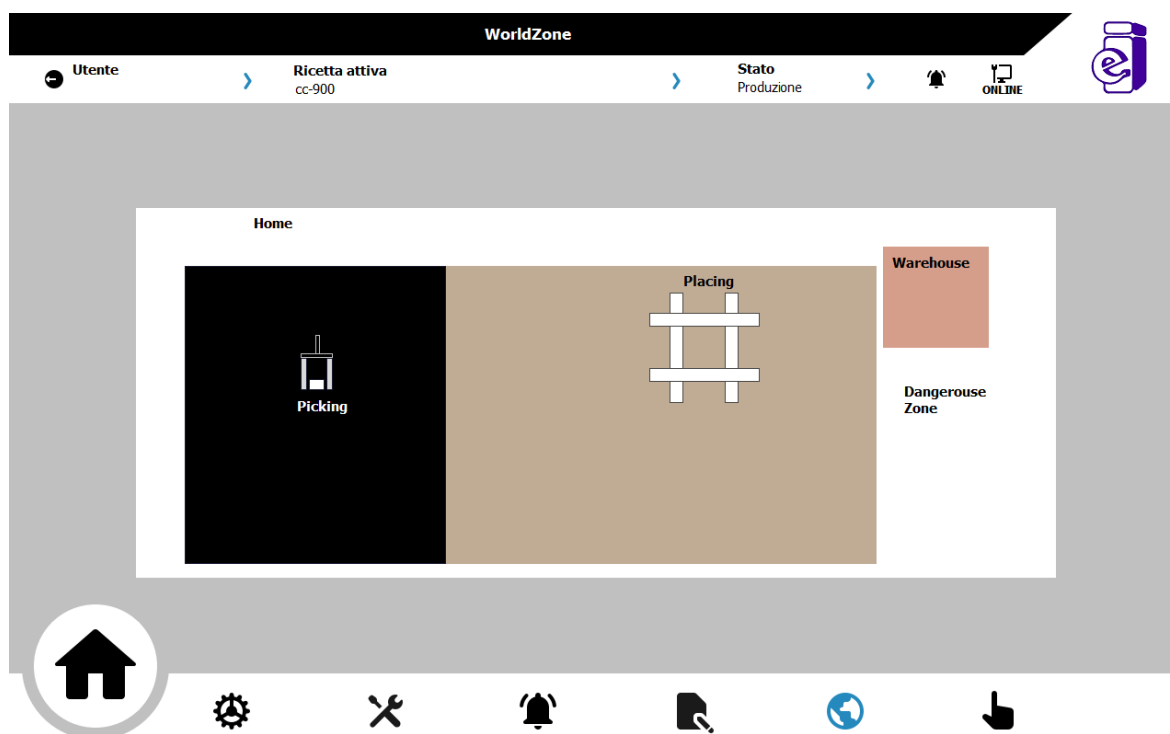


Figura 5.6: La pagina WorldZone

Dalla vista WorldZones è possibile visualizzare la posizione attuale del TCP del robot all'interno del layout. L'area è divisa nell'area di prelievo, deposito, home, magazzino e area al di fuori dell'area di lavoro.

5.6 La pagina dei dati macchina

L'ambiente di setup è accessibile premendo l'apposito tasto presente sulla home page del pannello. Tramite questa pagina è possibile modificare i dati fissi della macchina. nel caso specifico di questa applicazione l'unico dato issa consiste nella velocità del robot, selezionabile dalla dallo slider dai tasti alle sue estremità

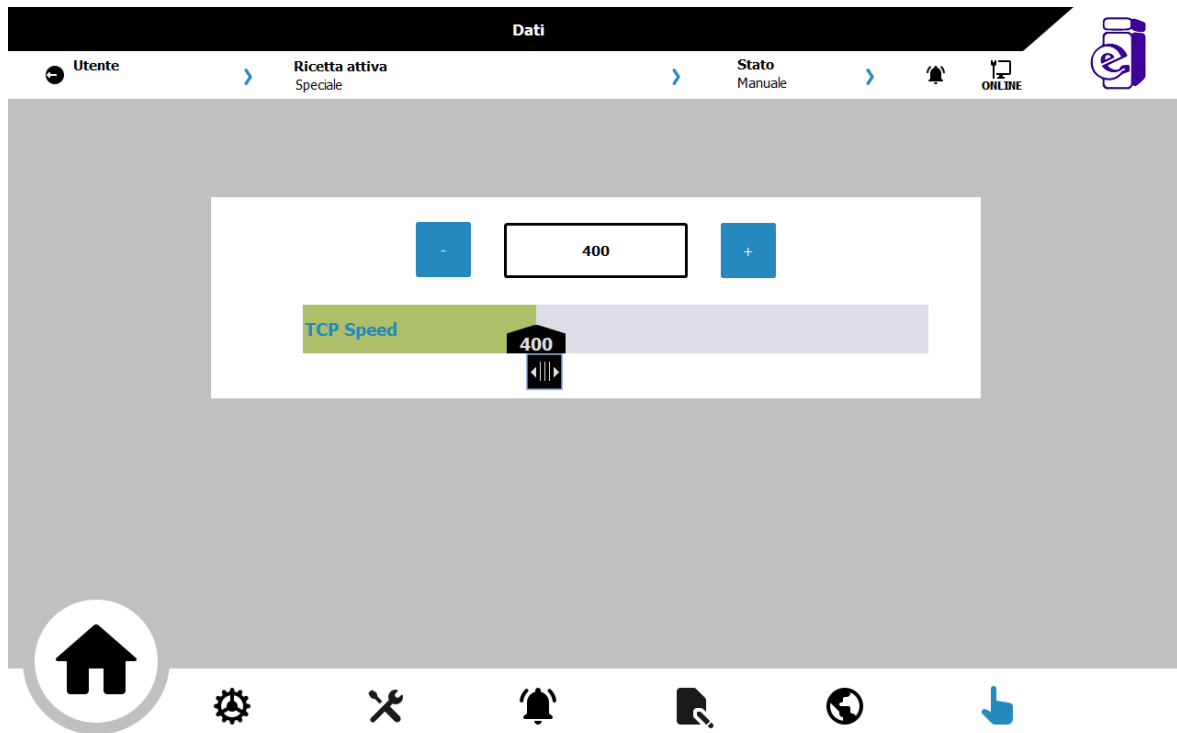


Figura 5.7: La pagina dei dati macchina

Capitolo 6

Conclusioni

In questa trattazione è stato analizzato lo sviluppo di una tipica applicazione di automazione industriale. Si è quindi discusso il linguaggio di programmazione RAPID per i robot, evidenziandone gli aspetti di maggiore interesse e le particolarità rispetto ad altre marche di robot. Dopo un breve accenno al sistema di visione e alle difficoltà affrontate durante il suo utilizzo si è descritto il software lato plc e la comunicazione tra le diverse unità.

È stato infine discusso il funzionamento del pannello per permettere 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.

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.

PLC Programmable Logic Controller.

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