EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Sistemes. Pla 2001

Títol: Integració d'un sistema de visió per computador a una cèl·lula flexible de transport

Document: Memòria

Alumne: Oscar Rojas Romero

Director/Tutor: Robert Martí Marly i David Raba Sánchez
Departament: Arquitectura i Tecnologia de Computadors
Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): juny / 09

ÍNDEX DE CONTINGUTS

1. DESCRIPCIÓ DEL PROJECTE 8

1.1. ORIGEN 8

1.2. OBJECTIU 9

1.3. PLANIFICACIÓ 9

2. ENTORN DE TREBALL 12

2.1. LA CEL·LA DE FABRICACIÓ FLEXIBLE 12

2.1.1. Movemaster EX RV-M1 12

2.1.1.1. Programació 14

2.1.2. Autòmat programable 15

2.1.2.1. Programació 16

2.1.3. El procés de transport i classificació 17

2.2. AMPLIACIÓ DE LA CEL·LA 18

2.2.1. Sistema de visió integrat NetSight II 19

2.2.2. Càmera 20

3. COMUNICACIÓ ENTRE SISTEMES 22

3.1. ANÀLISI 22

3.1.1. Comunicacions en la cel·la original 22

3.1.1.1. Connexió amb els PCs 23

3.1.2. Comunicacions en la cel·la nova 23

3.1.2.1. Control del RV-M1 24

3.1.2.2. Control del PLC 24

3.2. DISSENY 25

3.2.1. Connexió entre el NetSight II i el RV-M1 25

3.2.3. Control del PLC 26

3.2.4. Aplicació 27

3.3. IMPLEMENTACIÓ 28
3.3.1. Configuració del port sèrie 28
3.3.1.1. Preparació de la unitat de control del RV-M1 29
3.3.1.2. Creació del port sèrie en C++ 31
3.3.1.3. Enviament de comandes 32
3.3.2. Coordinació de les tasques de l'autòmat 33

4. VISIÓ ARTIFICIAL 36
4.1. ANÀLISI 36
4.1.1 Disposició de la càmera 36
4.1.2. Calibració de la càmera 37
4.1.3. Tractament de la imatge 37
4.2. DISSENY 38
4.2.1. Aplicació 39
4.3. IMPLEMENTACIÓ 40
4.3.1. Llibreries 40
4.3.1.1. Llibreria IFC 40
4.3.1.2. Llibreria OpenCV 41
4.3.1.3. Llibreria cvBlobsLib 41
4.3.2. Inicialització de la càmera 41
4.3.3. Cal·libració de la càmera 42
4.3.4. Correcció de la imatge 45
4.3.5. Binarització 46
4.3.5.1. Algoritme Isodata 46
4.3.5.2. Algoritme Otsu 46
4.3.6. Detecció de blobs 47

5. RECOLLIDA DE LA PEÇA 48
5.1. ANÀLISI 48
5.1.1. Moviment de l'element terminal del RV-M1 48
5.1.2. Programació de l'autòmat 50

5.2. DISSENY 50
5.2.1. L'àrea de recollida 51
5.2.3. Posicions del manipulador 52
5.2.4. Programació de l'autòmat: grafcet de 1er nivell 53
5.3. IMPLEMENTACIÓ 54
5.3.1. Delimitar l'àrea de recollida 54
5.3.2. Conversió de coordenades 55
5.3.3. La instrucció de recollida 56
5.3.4. Programació de l'autòmat: grafcet de 2on nivell 57

6. EXECUCIÓ I RESULTATS 60
6.1. PROVES 60
6.1.1. L'àrea de recollida 60
6.1.2. Binarització 60
6.2. APLICACIÓ DE PREPARACIÓ 62
6.3. EXECUCIÓ DEL PROCÉS 62
6.4. ANÀLISI DELS RESULTATS 74

7. CONCLUSIONS I TREBALL FUTUR 77
7.1. CONCLUSIONS 77
7.2. TREBALL FUTUR 77

8. BIBLIOGRAFIA 79

9. ANNEX 80
9.1. LLIBRERIA DE CONTROL 80
9.1.1. Classe Coordenades 81
9.1.2. Classe Robot 81
9.1.3. Classe SistemaVisió 81
9.2. PROGRAMACIÓ DE L'APLICACIÓ 82

9.2.1. Utilització del port sèrie 82
9.2.1.1. Creació del port de comunicació 83
9.2.1.2. Enviament i lectura d'informació 84
9.2.2. Utilització de la càmera 84
9.2.2.1. Preparació de l'objecte de captures 84
9.2.2.2. Adquisició d'imatges 85
9.2.2.3. Calibració 86
9.2.3. Tractament de la imatge 87
9.2.3.1. Correcció 88
9.2.3.2. Binarització 88
9.2.3.3. Detecció de blobs 88
9.3. INCLUSIÓ DE LES LLIBRERIES OPENCV I CVBLOBSLIB 89
9.3.1. Llibreries OpenCV 89
9.3.2. Llibreria cvBlobsLib 90
9.4. PROGRAMACIÓ DEL PLC 91
9.4.1. Grafcet 91
9.4.2. Variables 92
9.4.3. Transicions d'estats 93
9.4.4. Contactes 94
9.5. APLICACIÓ DE PREPARACIÓ 96
9.5.1. Execució 96

ÍNDEX DE FIGURES

Figura 1.1: La cel·la de fabricació flexible..... 8
Figura 1.2: Planificació 10
Figura 1.3: Blog sobre el desenvolupament del projecte..... 11
Figura 2.1: Mitsubishi Movemaster EX RV-M1 (braç articulat i unitat de control)..... 13
Figura 2.2: Articulacions del braç del RV-M1 13
Figura 2.3: Espai de treball del RV-M1 14
Figura 2.4: Sensors i posicions de l'autòmat programable 15
Figura 2.5: Exemple de grafcet de 3 estats..... 16
Figura 2.6: Detall de les peces (metall a l'esquerra, plàstic a la dreta)..... 17
Figura 2.7: Esquema del procés de transport i classificació segons el tipus de peça..... 18
Figura 2.8: Sistema NetSight II..... 19
Figura 2.9: Ports i connexions del NetSight II 20
Figura 2.10: Càmera JAI CV-A11..... 21
Figura 3.1: Fragment d'execució format per la unió de tasques dels dos sistemes..... 22
Figura 3.2: Diagrama de connexions sense determinar..... 23
Figura 3.3: Diagrama de connexions determinades 26
Figura 3.4: Diagrama del procés de canvi d'estat del PLC 27
Figura 3.5: Diagrama del procés de lectura de l'estat del PLC 27
Figura 3.6: Taula de configuració del registre SW2..... 30
Figura 3.7: Taula de configuració del baud rate..... 31
Figura 3.8: Panell de la unitat de control ja configurada..... 31
Figura 3.9: Pseudocodi de la funció Disponible..... 33
Figura 3.10: Pseudocodi de la funció Enviar Instrucció..... 33
Figura 3.11: Senyals de comunicació entre el RV-M1 i el PLC 34
Figura 3.12: Taula d'estats de les senyals de comunicació 35
Figura 4.1: Disposició de la càmera en la cel·la..... 36
Figura 4.2: Distorsió radial generada per la lent 37
Figura 4.3: Exemple de binarització..... 38
Figura 4.4: Esquema del procediment de l'anàlisi de la imatge..... 39

Figura 4.5: Espai de memòria necessari segons el tipus d'imatge 42
Figura 4.6: Patró utilitzat en la calibració 43
Figura 4.7: Col·locació del patró de calibració sobre la safata 43
Figura 4.8: Captura del patró feta per la càmera 44
Figura 4.9: Cantonades del patró localitzades 44
Figura 4.10: Comparació entre la imatge original i la corregida 45
Figura 5.1: Eixos de coordenades del robot 48
Figura 5.2: Desplaçament de l'element terminal sobre la safata 49
Figura 5.3: Eixos de coordenades del robot en el pla de la safata 49
Figura 5.4: Esquema del procés conversió de coordenades 51
Figura 5.5: Àrea de recollida 52
Figura 5.6: Grafcet de primer nivell del procés 53
Figura 5.7: Delimitació de l'àrea de treball 54
Figura 5.8: Relació dels eixos de coordenades del manipulador i de la imatge 55
Figura 5.9: Grafcet de segon nivell del procés 57
Figura 6.1: Binarització amb un valor de llindar de 112 61
Figura 6.2: Binarització amb un valor de llindar de 108 61
Figura 6.3: Binarització amb un valor de llindar de 45 61
Figura 6.4: Autòmat en l'estat 0 63
Figura 6.5: Captura de la consola (1) 63
Figura 6.6: Captura de la consola (2) 64
Figura 6.7: Captura de la consola (3) 64
Figura 6.8: Autòmat en l'estat 1 65
Figura 6.9: Captura de la consola (4) 65
Figura 6.10: Autòmat en l'estat 2 66
Figura 6.11: Autòmat en l'estat 3 66
Figura 6.12: Captura de la consola (5) 67
Figura 6.13: Disposició del patró de calibració sobre la safata 67
Figura 6.14: Captura de la consola (6) 68
Figura 6.15: Col·locació de les peces de delimitació sobre la safata 68
Figura 6.16: Captura de la consola (7) 69
Figura 6.17: Captura de la consola (8) 69

Figura 6.18: Captura de la consola (9) 70
Figura 6.19: Autòmat en l'estat 1 un cop feta la calibració..... 70
Figura 6.20: Captura de la consola (10) 71
Figura 6.21: Recollida de la peça 72
Figura 6.22: Captura de la consola (11) 72
Figura 6.23: Captura de la consola (12) 73
Figura 6.24: Captura de la consola (13) 73
Figura 6.25: Error de precisió..... 74
Figura 6.26: Comparació del blob segons la situación de la peça en la safata..... 75
Figura 6.27: Desviació en el moviment sobre la safata..... 76
Figura 9.1: Diagrama de classes..... 80
Figura 9.2: Grafcet de segon nivell del procés 92
Figura 9.3: Transició entre estats 0 i 1..... 93
Figura 9.4: Transició entre estats 1 i 0..... 93
Figura 9.5: Transició entre estats 1 i 2..... 93
Figura 9.6: Transició entre estats 2 i 3..... 94
Figura 9.7: Transició entre estats 3 i 0..... 94
Figura 9.8: Contactes (1) 94
Figura 9.9: Contactes (2) 95
Figura 9.10: Contactes (3) 95
Figura 9.11: Aplicació de preparació 96

1. DESCRIPCIÓ DEL PROJECTE

Aquest primer capítol és una introducció al projecte final de carrera, on s'explica el seu origen, els objectius que es pretenien assolir i la planificació que s'ha utilitzat per a la seva realització.

1.1. ORIGEN

El departament d'Arquitectura i Tecnologia de Computadors de la Universitat de Girona disposa d'un sistema integrat de visió artificial anomenat NetSight II, amb el qual es poden desenvolupar aplicacions de visió per computador en entorns industrials. Al tractar-se d'un ordinador destinat principalment a la inspecció de processos es va decidir aplicar-lo a la cel·la de fabricació flexible del mateix departament. Aquesta cel·la (veure figura 1.1), formada per un autòmat programable i un manipulador robòtic, realitza una tasca de transport i classificació de peces, l'escenari ideal per aquest sistema.

Així doncs, vam proposar-nos millorar la funcionalitat de la cel·la alhora que apreníem a utilitzar les capacitats del NetSight II.

Figura 1.1: La cel·la de fabricació flexible

1.2. OBJECTIU

L'objectiu d'aquest projecte final de carrera és millorar les prestacions de la cel·la de fabricació actual duent a terme les següents fites:

- **Integració d'un sistema de visió artificial a la plataforma.** Aquest sistema ha de permetre la detecció i anàlisi dels objectes transportats, augmentant les seves possibilitats de manipulació i classificació. Es pretén eliminar la restricció actual que obliga a col·locar les peces en una posició fixa de la safata de transport.

- **Control centralitzat del procés a través d'un ordinador.** El procés serà operat a través d'un computador industrial, el qual controlarà els diferents sistemes que componen la cel·la per tal que duguin a terme les seves funcions de forma coordinada.

- **Disseny i implementació d'una API del sistema.** Es desenvoluparà la llibreria de control del sistema per tal de facilitar la programació del procés.

1.3. PLANIFICACIÓ

A l'inici del projecte es va realitzar una planificació inicial de les diferents tasques que calia realitzar, dividint el volum de treball en quatre apartats:

1. **Control del manipulador robòtic per mitjà d'un ordinador.**
    a. Establir comunicació a través de la interfície RS-232 utilitzant l'entorn de desenvolupament Microsoft Visual C++ 6.0.
    b. Definir l'enviament i lectura de comandes al manipulador per executar operacions bàsiques.
2. **Integració del sistema de visió artificial a la cel·la.**
    a. Configurar la càmera.
    b. Realitzar un procés de calibració de la càmera.
    c. Tractar la imatge per facilitar l'extracció d'informació.
    d. Detectar les peces dins la imatge i obtenir-ne la seva posició.

3. **Implementació del procediment de recollida de la peça.**

    a. Moure el robot a una posició determinada en funció de la informació obtinguda amb el sistema de visió artificial.

    b. Establir comunicació entre l'ordinador i l'autòmat programable a través del manipulador.

    c. Programar l'autòmat per realitzar una tasca de transport senzilla.

    d. Fer funcionar el conjunt de forma coordinada.

4. **Anàlisi de resultats i generació de la documentació**

    a. Realitzar proves de funcionament de la cel·la

    b. Desenvolupar una API de sistema per facilitar la programació del procés.

    c. Escriure la documentació.

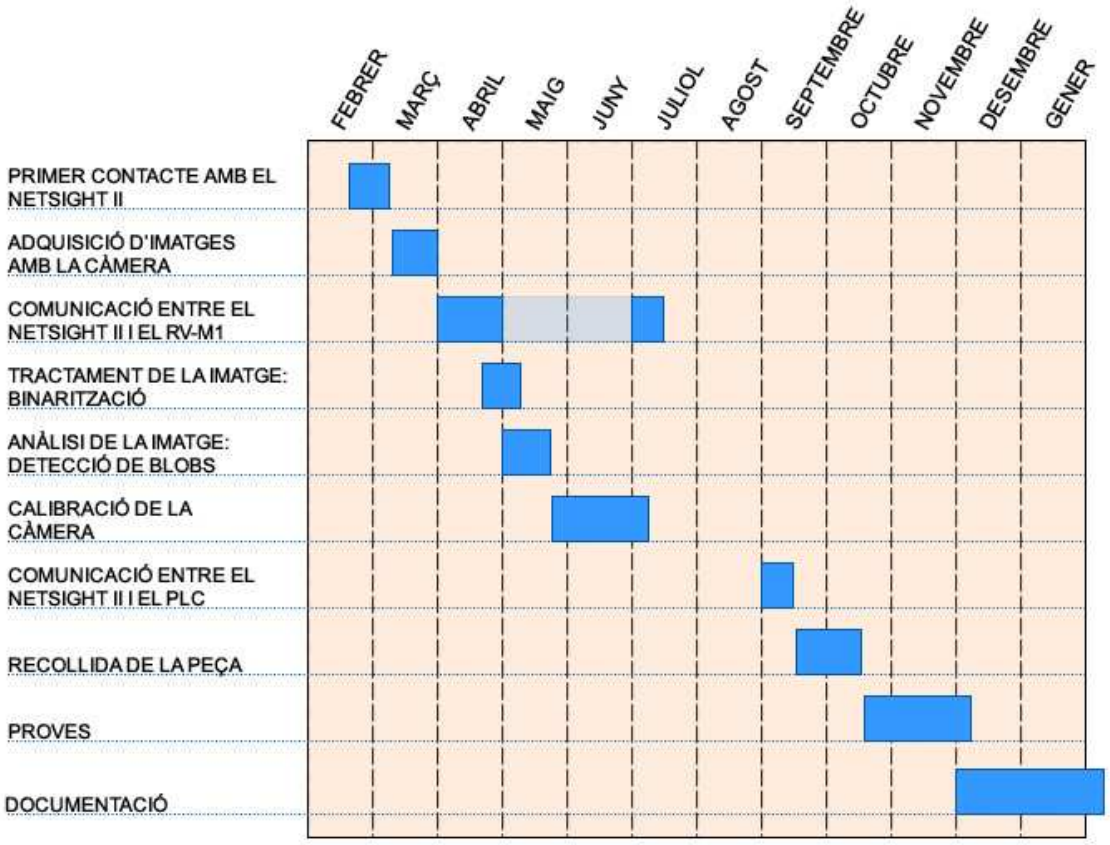


Figura 1.2: Planificació

Per altra banda es va optar per realitzar un seguiment de tot el desenvolupament del projecte creant un blog en el que periòdicament s'anirien descrivint els avenços en el treball i els problemes que anaven sorgint: **platflex.webcindario.com**

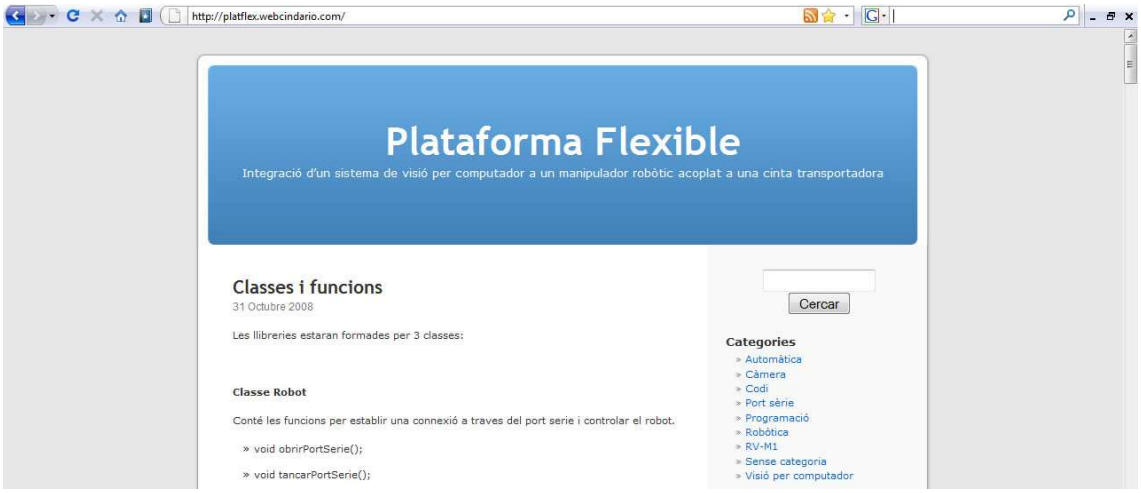


Figura 1.3: Blog sobre el desenvolupament del projecte

2. ENTORN DE TREBALL

En aquest capítol es fa una breu descripció de la cel·la de fabricació flexible, dels dos sistemes que en formen part i del procés que es desenvolupa. A continuació s'exposa l'ampliació que es pretén aplicar a la plataforma i els nous elements que s'integraran al conjunt.

2.1. LA CEL·LA DE FABRICACIÓ FLEXIBLE

La cel·la de fabricació flexible de la que disposa el departament d'Arquitectura i Tecnologia de Computadors està destinada a la realització d'un procés industrial senzill de transport i classificació de peces. Fruit del projecte final de carrera desenvolupat per l'estudiant d'ETIEI Eduard Albacar Morgó i dirigit per en Marc Carreras i Pérez, és actualment una eina utilitzada en la docència de l'assignatura Tècniques Avançades de Producció, la qual està enfocada a l'aprenentatge del funcionament i control d'autòmats programables.

La cel·la la componen dos sistemes: un autòmat programable i un manipulador robòtic. L'autòmat s'encarrega principalment de controlar tot el procés de transport gestionant dues cintes transportadores i un cilindre pneumàtic. El robot, per la seva banda, porta a terme les tasques de manipulació i classificació dels objectes transportats.

2.1.1. Movemaster EX RV-M1

El manipulador Movemaster EX RV-M1 (a partir d'ara RV-M1), desenvolupat per Mitsubishi, és un braç robòtic amb 5 graus de llibertat (GDL) i una capacitat de càrrega de 1.2 Kg. El sistema està format per:

- Braç articulat
- Element final per a manipulació (*gripper*)
- *Teaching box*
- Unitat de control
- Cables de connexió

Figura 2.1: Mitsubishi Movemaster EX RV-M1 (braç articulat i unitat de control)

El robot disposa de 5 articulacions de tipus rotatiu (veure figura 2.2), que li permeten posicionar-se en qualsevol punt d'un volum limitat anomenat espai de treball (veure figura 2.3). També té la possibilitat de realitzar moviments en coordenades cartesianes, característica que resultarà especialment útil en la realització d'aquest projecte. L'element final, de tipus *gripper* o pinça, li permet agafar els objectes.

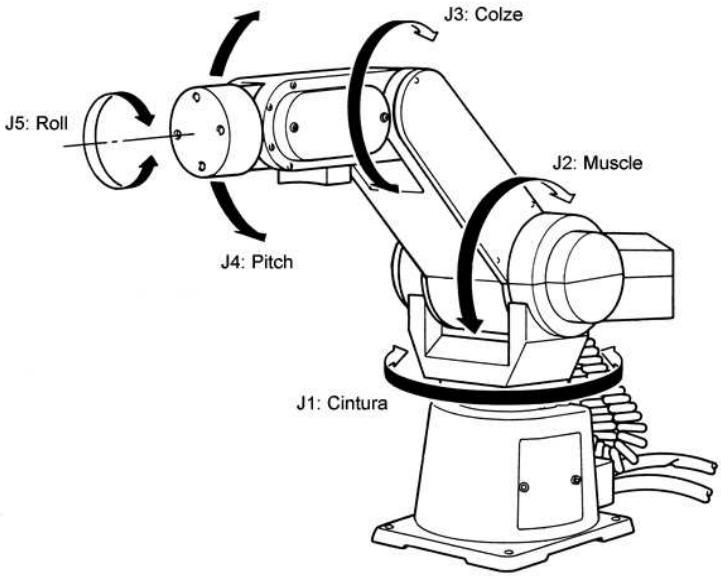


Figura 2.2: Articulacions del braç del RV-M1

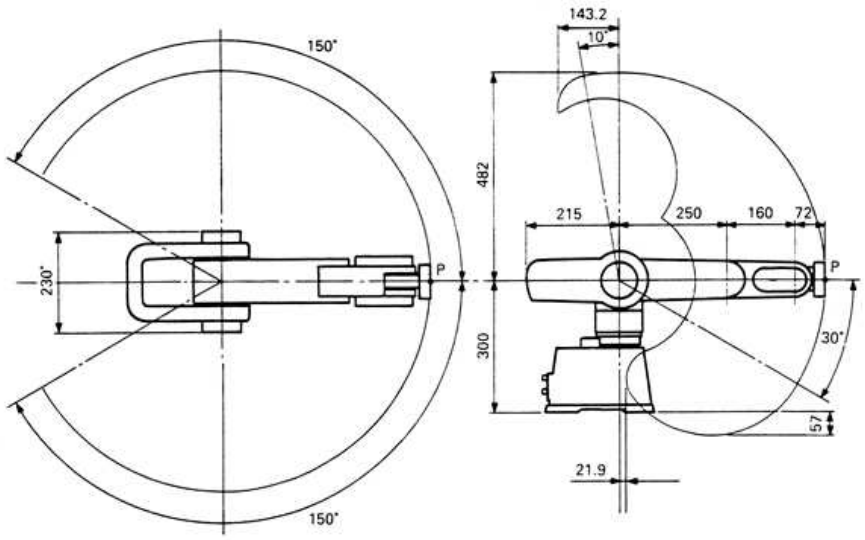


Figura 2.3: Espai de treball del RV-M1

El sistema va acompanyat d'un dispositiu en forma de teclat anomenat *Teaching Box* amb el qual es poden realitzar accions bàsiques relacionades amb el moviment del manipulador. Amb la seva ajuda es poden delimitar i emmagatzemar les posicions amb les que treballarà el robot per poder-les fer servir posteriorment en el programa.

2.1.1.1. Programació

La programació del RV-M1 es fa amb l'ajuda de comandes intel·ligents. Hi ha sis tipus d'instruccions segons la seva funció:

- Control de posició i moviment
- Control del programa
- Control de l'element terminal
- Control d'entrada/sortida
- Lectura a través de RS-232
- Altres

El programa es crea en un PC. Es poden executar comandes individuals enviant-les directament a la unitat de control o es pot escriure un programa format per línies numerades de comandes i emmagatzemar-lo a la memòria del sistema per a la seva posterior execució. A continuació es mostra un exemple de programa:

```
10 SP  4          ; Canviar la velocitat de desplaçament a 4
20 MO 6, O        ; Desplaçar-se a la posició 6 amb la pinça oberta
30 GC             ; Tancar la pinça
40 ED             ; Fi d'execució
```

2.1.2. Autòmat programable

Durant tota la documentació es fa referència al sistema de transport de peces com a autòmat programable o PLC. En realitat l'autòmat és el cervell que gestiona el procés, estant tot el conjunt format a més per sensors, motors trifàsics, cintes, variadors de potència i un sistema pneumàtic.

L'autòmat programable utilitzat pel transport de les peces és un model TSX 3710. Aquest mòdul principal és el que conté el programa d'execució i controla el sistema. El complementen un mòdul de comunicació i un d'entrada/sortida.

El mòdul de comunicació TSX SAZ 10 utilitza un bus AS-i (Actuator Sensor Interface). És el que avalua tot els sensors distribuïts per la cinta (veure figura 2.4) i controla els actuadors. En total hi ha 4 tipus de sensors: inductius, capacitius, òptics i d'efecte Hall.

El mòdul d'entrada/sortida TSX DMZ 28DR permet treballar amb elements d'E/S de lògica positiva i de lògica negativa.

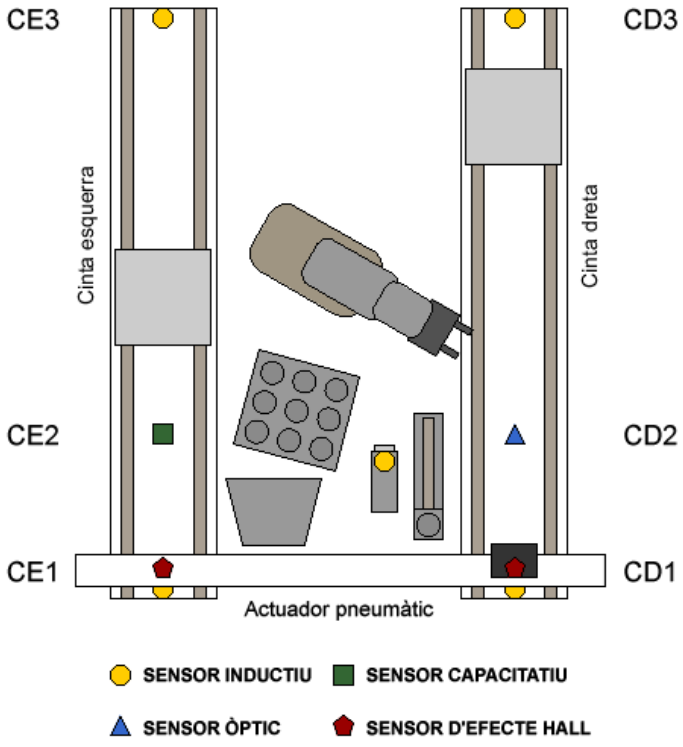


Figura 2.4: Sensors i posicions de l'autòmat programable

El desplaçament de les peces es fa per mitjà de dues safates situades a cada costat del braç articulat. Aquestes es mouen amb l'ajuda de cintes i el seu corresponent motor trifàsic. També hi ha la possibilitat de desplaçar una peça de la safata dreta a l'esquerra gràcies a un sistema pneumàtic que succiona l'objecte i el trasllada d'una banda a l'altra.

La funció dels sensors és detectar la posició de les safates i delimitar tres punts d'aturada en el seu recorregut (veure figura 2.4), que són:

- Zona de seguretat (CD3, CE3)
- Zona de manipulació (CD2, CE2)
- Zona de canvi de safata (CD1, CE1)

Hi ha, a més, un sensor inductiu encarregat de distingir el material de les peces transportades. Es troba situat en front del braç articulat per tal que aquest hi pugui apropar el objectes en el moment de diferenciar-los.

2.1.2.1. Programació

La programació de l'autòmat es fa des d'un PC. En ell es crea el grafcet del procés i s'envia al mòdul principal, el qual l'executarà tan bon punt rebi l'ordre d'inici.

Un grafcet és un conjunt d'estats amb unes accions definides que es van activant a mesura que es compleixen determinades condicions, com per exemple la resposta d'un dels sensors o una senyal provinent del RV-M1. A continuació podem veure un exemple de grafcet:

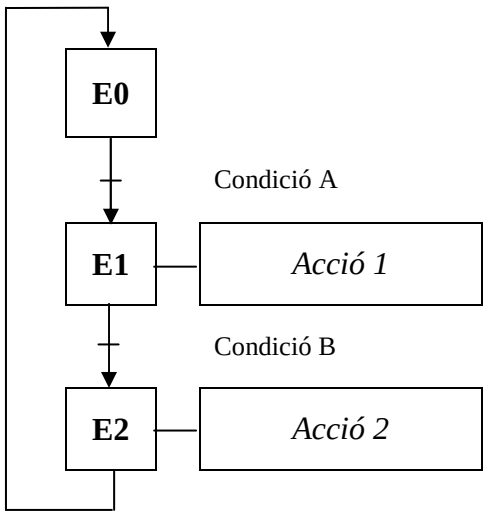


Figura 2.5: Exemple de grafcet de 3 estats

2.1.3. El procés de transport i classificació

El procés que executa la cel·la consisteix en transportar i classificar unes peces cilíndriques segons el tipus de material del que estan compostes: metall o plàstic (veure figura 2.6). L'objectiu és que les peces, barrejades i manipulades de forma individual, quedin finalment separades en dos grups segons el seu tipus (veure figura 2.7).

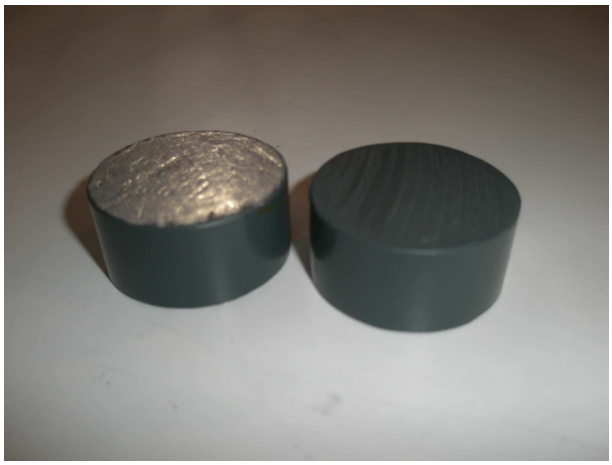


Figura 2.6: Detall de les peces (metall a l'esquerra, plàstic a la dreta)

Inicialment es posicionen les dues safates als seus respectius punts de sortida o zones de seguretat. Tot seguit les peces es col·loquen, una a una i de forma manual, sobre la safata de transport dreta. És important destacar que la posició de la peça és fixa i es correspon a una posició prèviament emmagatzemada pel manipulador (aquesta és la restricció que es vol eliminar).

A continuació la peça és transportada fins a l'interior de l'espai de treball del RV-M1: la zona de manipulació. El braç articulat recull l'objecte i el col·loca davant del sensor inductiu per diferenciar el seu material. Si és metàl·lic el col·locarà al palet que té davant seu, altrament el retornarà a la safata on un conjunt de desplaçaments (entre ells l'efectuat per l'actuador pneumàtic) el portaran a la safata esquerra on el robot el recol·locarà per tal que no interfereixi amb les peces que vindran després.

Un cop s'ha tractat la totalitat de les peces el manipulador farà la última tasca de traslladar les peces paletitzades a la safata dreta. Finalment l'autòmat desplaça les dues safates (cadascuna amb un grapat de peces d'un sol tipus) a la zona de seguretat.

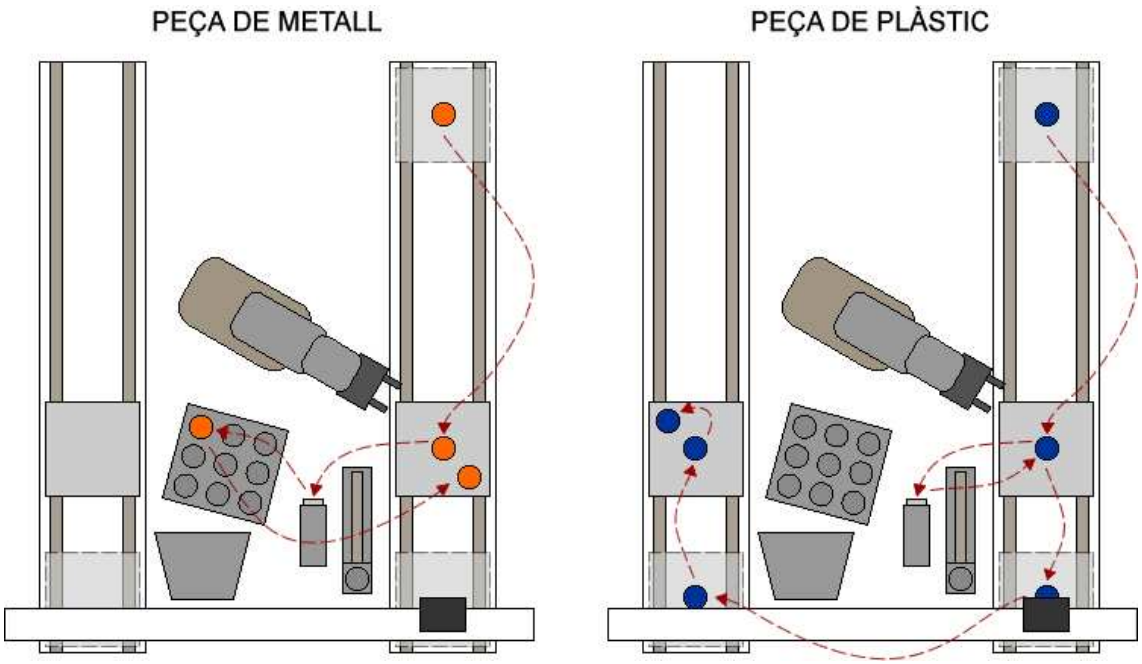


Figura 2.7: Esquema del procés de transport i classificació segons el tipus de peça

2.2. AMPLIACIÓ DE LA CEL·LA

Per assolir l'objectiu de millorar les prestacions de la cel·la flexible s'inclourà al conjunt un sistema de visió artificial, format per un ordinador integrat i una càmera analògica monocromàtica. Aquest sistema tindrà tres funcions, que són les següents:

- **Controlar les operacions del manipulador.** L'ordinador ha de controlar el Movmaster RV-M1, en lloc de la seva pròpia unitat de control. Es substituirà l'ús d'un programa prèviament escrit i sense possibilitat de variació per l'enviament directe de comandes a través d'un programa escrit en llenguatge C++.

- **Coordinar els moviments de la cinta de l'autòmat programable.** L'ordinador ha d'indicar al PLC quan ha de desplaçar la safata de transport per tal de portar les peces a l'àrea de treball del manipulador.

- **Detectar la peça dins la safata de transport.** La funció del sistema de visió artificial serà localitzar la peça dins la safata de transport, amb l'objectiu de poder determinar la posició a la que s'haurà de traslladar l'element terminal del manipulador per poder agafar l'objecte. D'aquesta manera s'evita haver de definir una posició sobre la safata.

2.2.1. Sistema de visió integrat NetSight II

El NetSight II (veure figura 2.8) és un sistema de visió artificial multicàmera desenvolupat per l'empresa IPD. Es tracta d'una plataforma compacte destinada al disseny i execució d'aplicacions d'inspecció en processos industrials. Inclou eines de captura, processament i visualització d'imatges, comunicació E/S, xarxa, etc. Les seves característiques principals són:

- Suport per fins a tres càmeres progressives monocromes, el que li permet inspeccionar una peça des de diversos angles de forma simultània o avaluar diferents objectes al mateix moment.

- Llibreries IFC (Imaging Classes Foundation) integrades, que permeten la gestió del hardware i dels elements d'entrada i sortida, així com la captura i processat bàsic d'imatges.

- Programari per facilitar el disseny d'aplicacions d'inspecció industrial (iNspect i Sherlock).

- Sistema operatiu Windows XP, idoni per instal·lar i utilitzar aplicacions de tercers.

- Connexions RS-232, Centronics i Ethernet per a la comunicació i integració dins la xarxa (veure figura 2.9).

Figura 2.8: Sistema NetSight II

Especificacions

- Processador Intel Celeron 1,2 GHz
- 240 MB de memòria RAM
- 40 GB de capacitat de disc dur
- Tres ports per a càmeres analog.
- LEDS programables
- Sortida de vídeo VGA
- Entrada i sortida d'àudio
- Port ethernet 10/100
- Port sèrie RS-232
- Port paral·lel
- Dos ports USB 1.1
- Port IEEE 1394
- Port de control Input/Output
- Port PS/2 per teclat i per ratolí

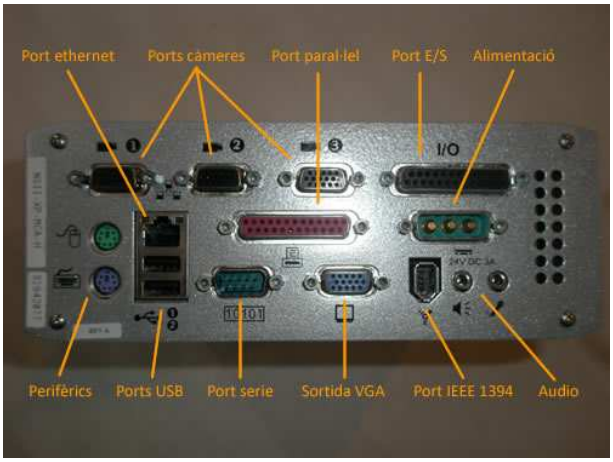


Figura 2.9: Ports i connexions del NetSight II

2.2.2. Càmera

La càmera que s'utilitzarà és una JAI CV-A11 (veure figura 2.10), una càmera monocroma progressiva amb un sensor de 1/3" i resolució 648 x 490 píxels. És compacta, robusta i de baix cost, apte per aplicacions industrials i amb possibilitat d'escombrat variable.

Les seves característiques principals són:

- Sensor d'alta resolució 1/3" IT CCD.
- Extrema sensitivitat : - 0.05 Lux en el sensor.
- Rang dinàmic millorat i relació senyal/soroll superior a 56db.
- Funcionament en format entrellaçat o progressiu.
- Possibilitat de treballar amb un llarg temps d'exposició, per aplicacions amb baixa il·luminació.

Figura 2.10: Càmera JAI CV-A11

3. COMUNICACIÓ ENTRE SISTEMES

En aquest capítol s'explica com es comuniquen els tres sistemes que componen la cel·la de fabricació flexible i les connexions que s'han establert per fer-ho possible.

3.1. ANÀLISI

La cel·la està composta de diversos elements interconnectats que es comuniquen per poder dur a terme un procés comú. La inclusió del sistema de visió implica crear noves connexions i nous sistemes de comunicació.

Començarem per explicar les comunicacions de la configuració original i posteriorment analitzarem la forma en que el NetSight II controlarà els altres sistemes per determinar els nous enllaços que caldrà establir.

3.1.1. Comunicacions en la cel·la original

Tenim dos sistemes que treballen sobre un programa d'execució propi. L'autòmat i el manipulador realitzen un conjunt de tasques que, unides, formen el procés de transport i classificació. Aquestes accions no són simultànies, sinó que es van intercalant tasques de desplaçament i de manipulació de les peces (veure figura 3.1).

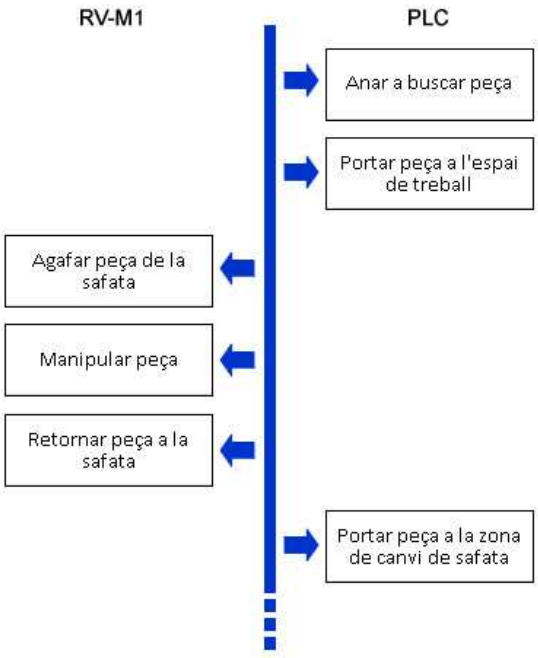


Figura 3.1: Fragment d'execució format per la unió de tasques dels dos sistemes

Per tal de que els dos sistemes coordinin les seves accions ha d'existir una comunicació entre ells. Per fer-ho hi ha implementada una connexió d'entrada/sortida entre els dos sistemes a través de la qual s'envien missatges. Quan un sistema finalitza una operació li comunica a l'altre que ja pot començar a treballar.

3.1.1.1. Connexió amb els PCs

A part de la connexió abans esmentada, tant l'autòmat com el manipulador disposen d'una connexió a un PC. Aquesta connexió sols s'utilitza per transferir el programa d'execució a la unitat de control (per tant és unidireccional). Un cop emmagatzemat el programa els sistemes esdevenen independents i no requereixen més aquest enllaç.

3.1.2. Comunicacions en la cel·la nova

El sistema NetSight II significa un canvi important en el funcionament de la cel·la de fabricació flexible. No sols s'encarregarà de l'apartat de visió artificial sinó que pretenem que es converteixi en el cervell de tot el procés de transport i classificació.

Per esbrinar quines connexions cal establir i quines han de ser modificades analitzarem primer la forma en que passaran a ser controlats el manipulador i l'autòmat (veure figura 3.2).

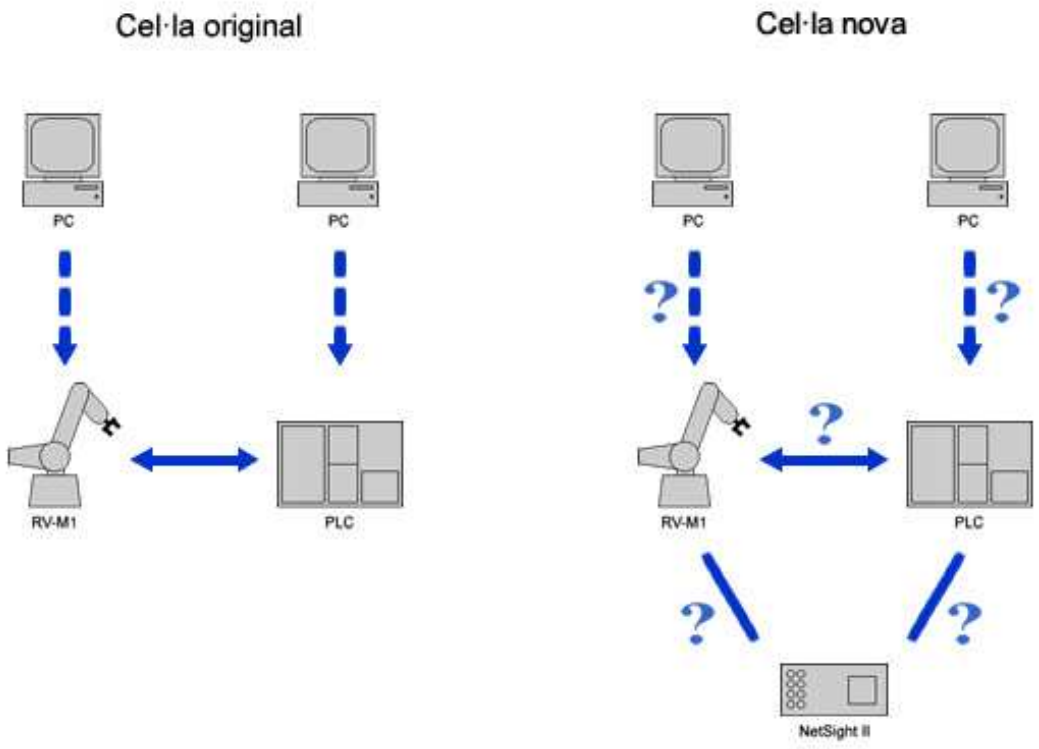


Figura 3.2: Diagrama de connexions sense determinar

3.1.2.1. Control del RV-M1

El manipulador RV-M1 disposa de dues configuracions de control, segons si aquest es realitza a través de la unitat de control o a través d'un ordinador:

- **Execució a través de la unitat de control**. Aquesta configuració fa servir la unitat de control per executar les instruccions. L'ordinador s'utilitza tan sols per escriure un programa complert basat en comandes i per transferir-lo a la RAM de la unitat (o a la EPROM si es vol una programació permanent). És una configuració especialment útil en entorns industrials, ja que d'aquesta manera es pot executar la tasca de forma autònoma sense necessitat d'instal·lar un PC en la planta de procés.

- **Execució a través d'un ordinador.** Amb aquesta configuració és l'ordinador qui controla l'execució del manipulador enviant-li directament les comandes, substituint el programa definit dins la unitat de control. D'aquesta manera s'incrementa considerablement la flexibilitat del sistema degut a que totes les accions del robot poden ser determinades per un programa escrit en un llenguatge de programació de propòsit general, el qual pot disposar d'altres dispositius o sistemes realitzant, per exemple, tasques d'anàlisi del procés.

Actualment la cel·la fa servir la primera configuració. Amb ella les comandes queden emmagatzemades a memòria i són, per tant, fixes. Això no és un problema si totes les posicions són invariables, però en el nostre cas no és així.

L'objectiu que ens hem proposat és poder determinar la posició de recollida segons l'anàlisi del sistema de visió. Així doncs, donat que la peça pot estar en diferents llocs de la safata, la comanda de desplaçament del robot a la posició de recollida ha de ser modificable. Per aquest motiu utilitzarem la segona configuració, creant una aplicació que vagi enviant les comandes una a una.

Aquesta implementació implica que ja no és necessari disposar d'un PC per transferir el programa d'execució. Per tant es pot simplement substituir l'ordinador pel sistema NetSight II, canviant el tipus de connexió si és necessari.

3.1.2.2. Control del PLC

A diferència del manipulador, l'autòmat només pot treballar amb un programa pre-carregat a memòria. Per aquest motiu es conservarà l'enllaç amb el PC des del que es programa el grafcet. Però un altre aspecte que cal analitzar és com ho farà el NetSight II per controlar les operacions de l'autòmat.

La solució a aquest problema consisteix en aprofitar la connexió existent entre l'autòmat i el manipulador i utilitzar-lo com a canal de comunicació entre el NetSight II i el PLC. Això és possible gràcies a que el RV-M1 disposa d'instruccions per enviar informació de la seva entrada/sortida a l'ordinador al que està connectat i viceversa.

3.2. DISSENY

La cel·la original té les connexions implementades de la següent manera:

- **RV-M1 amb el PC:** Interfície RS-232 (cable sèrie)
- **PLC amb el PC:** Interfície Ethernet
- **RV-M1 amb el PLC:** Interfície I/O

La única modificació que aplicarem a la cel·la serà substituir el PC connectat al RV-M1 pel NetSight II. Hem de veure si podem aprofitar la connexió RS-232 o hem d'utilitzar-ne una de diferent.

3.2.1. Connexió entre el NetSight II i el RV-M1

El RV-M1 disposa de dues interfícies per establir una connexió amb un ordinador:

- **Interfície Centronics.** Dissenyada per Centronics Corporation, és l'estàndard utilitzat per connectar dispositius paral·lels, tals com les impressores. La transmissió paral·lela assegura una ràpida velocitat sense requerir una configuració especial. Tot i així la comunicació entre els dos elements és simplex, és a dir en un sol sentit. Això implica que no hi hagi disponible comandes per sol·licitar informació al robot.

- **Interfície RS-232.** És l'estàndard utilitzat en la comunicació pel port sèrie. La transmissió a traves d'un sol canal és considerablement més lenta que la paral·lela si el baud rate és baix. A més cal tenir molt en compte que les configuracions de comunicació dels dos dispositius han de coincidir. L'avantatge que presenta aquesta interfície sobre la Centronics és que és duplex (doble sentit), el que ens permet llegir informació interna del robot.

Donat que el NetSight II pren el paper d'unitat de control és imprescindible que pugui obtenir informació interna del manipulador. Això ens fa descartar la interfície Centronics per la seva falta de bidireccionalitat. És per aquest motiu que només podem fer servir la interfície RS-232, que és la que ja es feia servir en la connexió de l'autòmat amb el PC i que podrem aprofitar.

A la figura 3.3 es pot veure la modificació realitzada en les connexions de la cel·la.

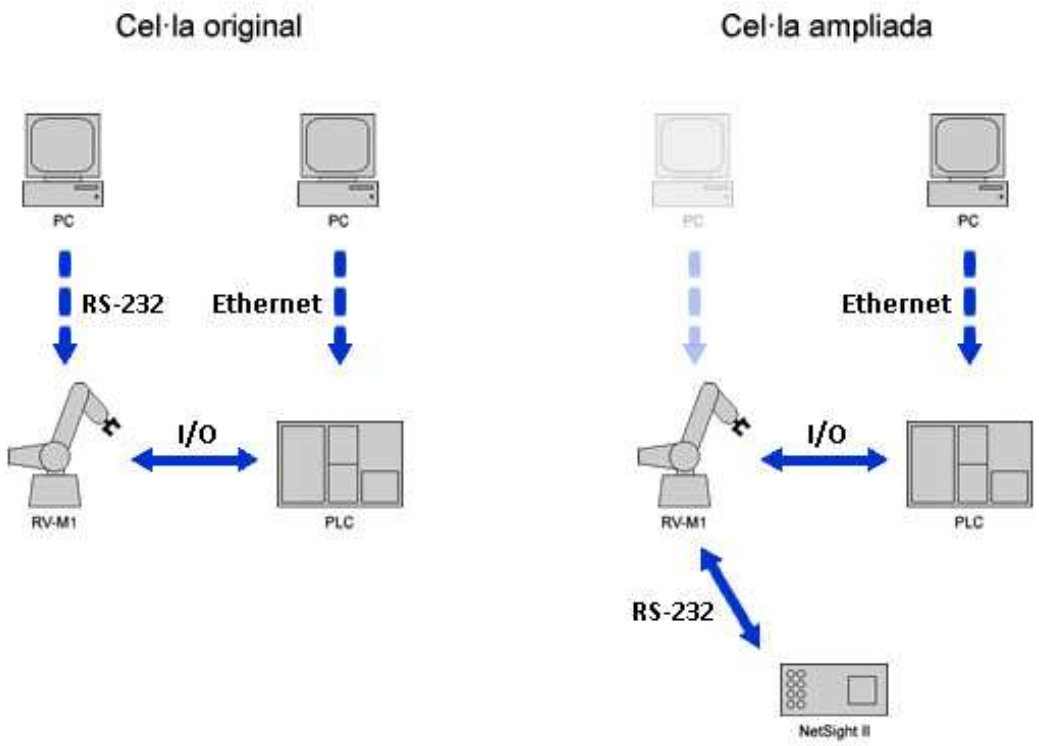


Figura 3.3: Diagrama de connexions determinades

3.2.3. Control del PLC

Donat que el NetSight II no està comunicat directament amb l'autòmat la coordinació d'aquest (canvis d'estat en el grafcet) es farà a través del manipulador. Es conservarà el sistema de comunicació existent entre el RV-M1 i el PLC basat en la modificació de dues senyals binàries, amb la diferència que ara el manipulador actuarà segons ordres del NetSight II i l'hi haurà de transmetre a aquest les respostes de l'autòmat.

A la figura 3.4 es mostra el procediment que el NetSight II ha de seguir per canviar l'estat del PLC, i a la figura 3.5 els passos per esbrinar en quin estat es troba l'autòmat.

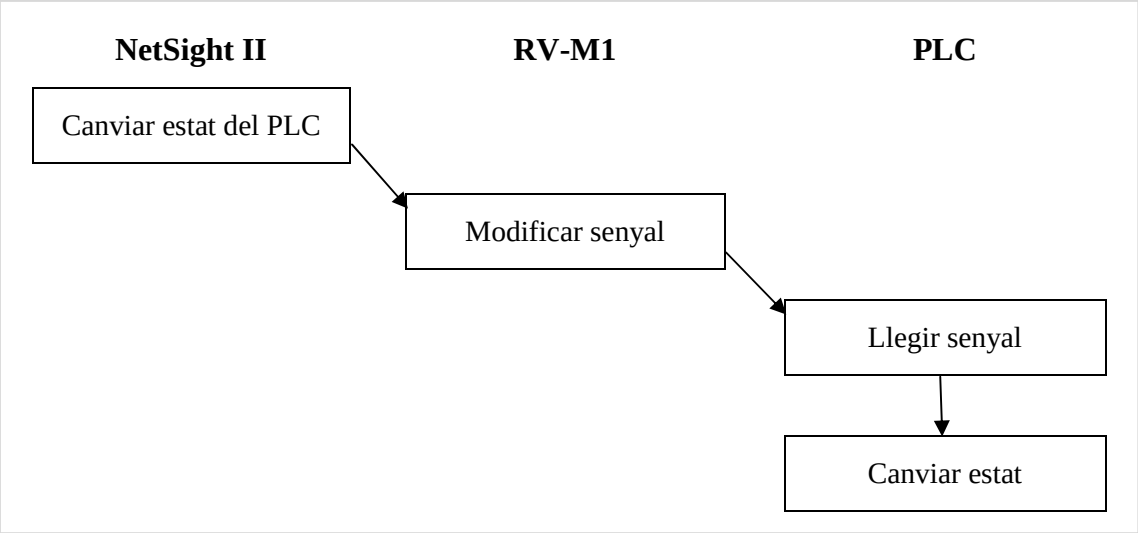


Figura 3.4: Diagrama del procés de canvi d'estat del PLC

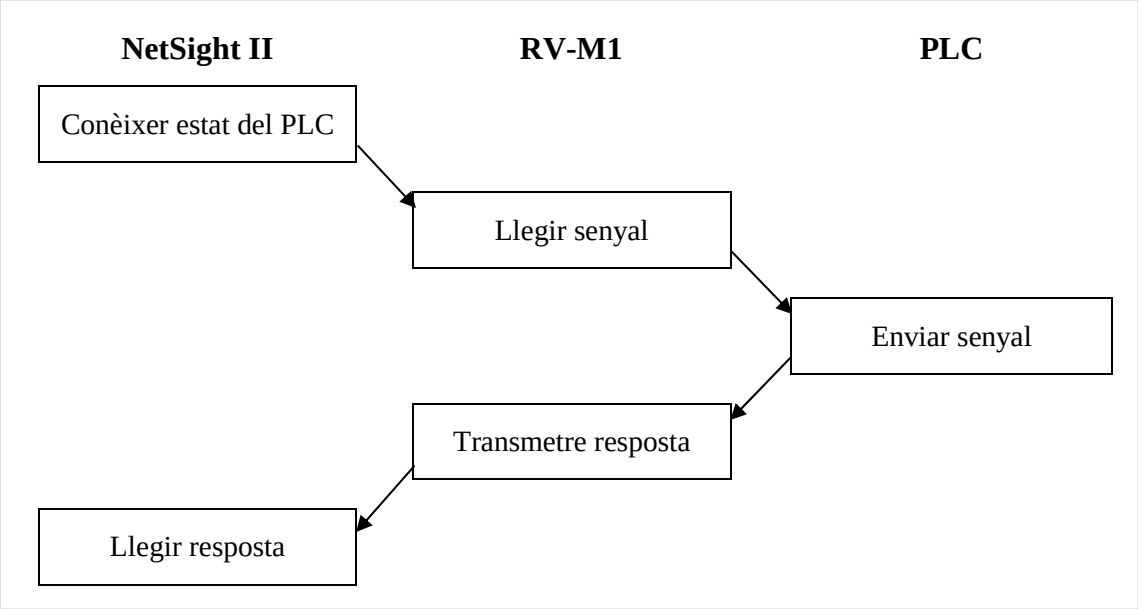


Figura 3.5: Diagrama del procés de lectura de l'estat del PLC

3.2.4. Aplicació

L'aplicació que es desenvoluparà per portar a terme el procediment de transport i classificació disposarà de totes les funcions de control del robot en la classe anomenada *Robot*. Aquesta classe permetrà les següents accions:

- **Obrir un canal de comunicació amb el manipulador.** Abans de poder comandar el robot cal preparar el port a traves del qual s'enviaran les instruccions.

- **Enviar instruccions al manipulador.** A través del port se l'hi enviaran les comandes que ha d'executar.

- **Coordinar les accions de l'autòmat.** Enviant determinades comandes al manipulador i llegint les respostes d'aquest es podrà coordinar el desplaçament de la safata.

3.3. IMPLEMENTACIÓ

L'única modificació a realitzar en la cel·la de fabricació flexible és substituir el PC connectat al RV-M1 pel NetSight II. Per fer-ho s'utilitzarà el cable sèrie que es feia servir en la connexió anterior.

L'aplicació de comunicació i control del RV-M1 i el PLC s'ha programat sota la plataforma Windows utilitzant l'entorn de desenvolupament Visual C++ 6. Per més detall es poden consultar les implementacions del codi a l'annex.

3.3.1. Configuració del port sèrie

L'estàndard RS-232 defineix un tipus de comunicació sèrie i asíncrona. La paraula sèrie significa que només s'envia un bit d'informació per cada unitat de temps. Per altra banda el terme asíncron vol dir que no hi ha una relació temporal entre l'emissor i el receptor i que són necessaris uns bits especials per delimitar el començament i el final d'un missatge.

Per utilitzar la interfície RS-232 cal configurar prèviament diversos paràmetres als dos elements que es comuniquen. Si hi ha alguna discrepància la comunicació no es podrà dur a terme correctament. Aquests paràmetres són:

- **Baud rate.** El baud rate és una mesura de la velocitat en que es transmet informació a través del canal de comunicació. La interfície RS-232 utilitza únicament dos estats de voltatge anomenats MARK (voltatge negatiu) i SPACE (voltatge positiu). Amb aquesta codificació, el baud rate és el nombre de bits d'informació, incloent bits de control, que es transmeten per segon.

- **Longitud de caràcter.** Determina el nombre de bits que té cada caràcter enviat. El seu valor és entre 5 i 9.

- **Paritat.** El bit de paritat ve a continuació de les dades i s'utilitza per a la detecció d'errors. El seu valor por ser senar o parell: si la paritat és senar, l'últim bit d'informació que s'enviarà serà un 1 si el nombre de 0s al camp de dades és imparell; si és parell l'últim bit serà 1 quan el nombre de 0s sigui parell.

- **Bits de stop.** Delimita el final del caràcter. El seu valor pot ser 1 o 2.

3.3.1.1. Preparació de la unitat de control del RV-M1

La unitat de control del manipulador disposa d'un conjunt de commutadors o *switchs* presents a la placa de control (veure figura 3.8) per tal de configurar diversos paràmetres, entre ells els de comunicació:

- **ST1 (Mode de control).** Permet seleccionar entre operar el manipulador a través de la unitat de control (switch pujat) o a través d'un ordinador (switch baixat).

  ✓ Valor: baixat

- **ST2 (Transferència de dades de la EPROM a la RAM).** Permet el traspàs de les dades que conté la EPROM a la memòria RAM de la unitat de control a l'encendre el manipulador (switch pujat per transferir, baixat altrament). S'utilitza quan el manipulador ha d'executar el programa emmagatzemat a la EPROM.

  ✓ Valor: baixat

- **SW1 (Selecció de funcions).** Permet escollir diverses opcions com per exemple el tipus de targeta d'E/S utilitzada o la desactivació de l'alarma sonora en cas d'error. No és una configuració que ens afecti, pel que mantindrem el valor original.

  ✓ Valor: 11100001

- **SW2 (Format de comunicació a través de la interfície RS-232).** Permet seleccionar els valors dels atributs de comunicació (factor multiplicador del baud rate, longitud dels caràcters, utilització o no del bit de paritat i el seu tipus, i nombre de bits de stop). A la figura 3.6 es pot veure la taula de configuració.

Configuració desitjada:

- Baud rate: 9600
- Bit de paritat: senar
- Longitud de caràcter: 7 bits
- Bits de stop: 2 bits

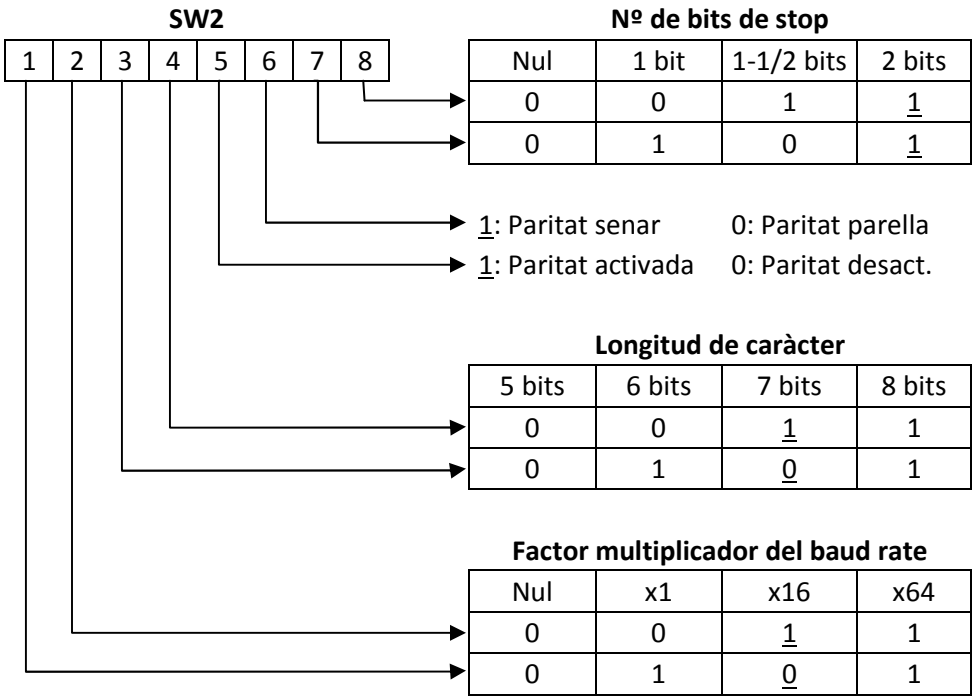


Figura 3.6: Taula de configuració del registre SW2

- ✓ Valor: 01011111

- **SW3 (Selecció del baud rate).** Permet triar la velocitat de transferència entre 1200 i 9600 bit/s (en combinació amb el factor multiplicador seleccionat al registre de commutadors SW2). A la figura 3.7 es pot veure la taula de configuració.

Bit SW3	Factor del baud rate (SW2)		
	x1	x16	x64
1	1200	75	-
2	2400	150	-
3	4800	300	75
4	9600	600	150
5	-	1200	300
6	-	2400	600
7	-	4800	1200
8	-	9600	2400

Figura 3.7: Taula de configuració del baud rate

- ✓ Valor: 00000001

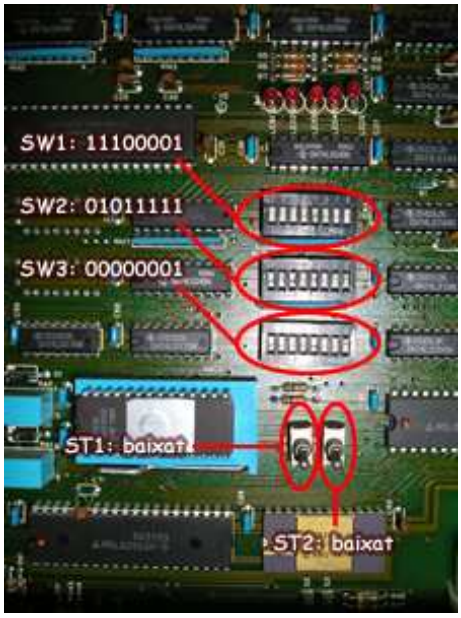


Figura 3.8: Panell de la unitat de control ja configurada

3.3.1.2. Creació del port sèrie en C++

Microsoft posa a la nostra disposició l'API del sistema Win32 amb la qual es pot obrir un canal de comunicació amb un dispositiu a través del port sèrie. A continuació hi ha l'explicació del procediment a seguir.

1. **Obrir el port.** El primer pas és crear l'objecte a través del qual accedirem al port sèrie. L'únic aspecte a considerar és que ha de permetre l'escriptura i la lectura d'informació.

2. **Preparar els atributs de comunicació.** L'objecte que acabem de crear disposa d'uns atributs de comunicació predeterminats. Hem d'actualitzar-los per tal que es corresponguin amb els que hem establert a la unitat de control del manipulador (altrament provocarem un estat d'error al transmetre-li dades).

3. **Establir el timeout**. El manipulador no disposa d'una pila on emmagatzemar les instruccions d'execució directa que li envia l'ordinador. Per aquest motiu tota comanda que rebi mentre està fent una operació es perd. Si a més aquesta instrucció requereix una resposta del manipulador l'aplicació queda aturada indefinidament. El timeout és un valor que indica el temps màxim d'espera per rebre dades. Si passat aquest temps la funció de lectura del port sèrie no ha obtingut cap missatge tanca la petició i continua amb l'execució del programa.

Un cop preparat el port de comunicacions ja es pot procedir a enviar i llegir dades. Al finalitzar no s'ha d'oblidar tancar el port.

3.3.1.3. Enviament de comandes

Per enviar dades tan sols cal preparar una taula amb la informació i transmetre-la a través del port sèrie.

En el cas del RV-M1 el que enviarem són cadenes de caràcters amb la comanda a executar i acabades amb el caràcter de retorn **\r**. Per exemple, per enviar la comanda MO 10 es prepararia la següent cadena:

M	O		1	0	\r

Com ja s'ha comentat les instruccions enviades es perden si el manipulador no està en repòs. Per tant cal saber si el robot està disponible abans d'enviar-li una nova ordre. Malauradament no hi ha cap operació que retorni l'estat actual del manipulador. Per resoldre aquest problema s'ha creat un mètode per comprovar la disponibilitat del RV-M1 (veure figura 3.9). El que fa aquesta funció és demanar-li al manipulador una informació interna. Si finalitzat el timeout no ha rebut resposta significa que encara està treballant, pel que tornarà a sol·licitar-la. Quan finalment el robot hagi acabat l'execució podrà respondre a la petició (trencant el bucle), el que significa que ja pot rebre una nova instrucció.

```
funció Disponible( )
 {
        repetir fins resposta > 0
         {
                sol·licitar informació al robot;
                resposta = nombre de bytes rebuts;
        }
}
```

Figura 3.9: Pseudocodi de la funció Disponible

La funció per enviar una instrucció consistirà en comprovar primerament que el robot estigui disponible i procedir a continuació a enviar la cadena de caràcters de la instrucció pel port sèrie (veure figura 3.10)

```
funció EnviarInstruccio(char[] instruccio )
 {
        Disponible( );
        EnviaPerPortSerie (instrucció);
}
```

Figura 3.10: Pseudocodi de la funció Enviar Instrucció

3.3.2. Coordinació de les tasques de l'autòmat

La coordinació entre l'autòmat i el manipulador es realitza a través del pas de missatges mitjançant una connexió I/O. Cada sistema disposa de dues senyals binàries per comunicar el seu estat i/o donar instruccions (veure figura 3.11). Per una banda les senyals del manipulador són condicions de canvi d'estat en el grafcet de l'autòmat. Per l'altra, les senyals de l'autòmat són variables dins el codi del manipulador que activaran un procediment o altre.

		Manipulador	Autòmat
Senyal 1	Entrada	IN 9	%I1.11
	Sortida	OUT 9	%Q2.1
Senyal 2	Entrada	IN 10	%I1.10
	Sortida	OUT 10	%Q2.2

Figura 3.11: Senyals de comunicació entre el RV-M1 i el PLC

Amb l'actualització de la cel·la les senyals que fins ara enviava o rebia la unitat de control del manipulador passen a ser controlades pel NetSight II. L'activació o desactivació d'un bit del RV-M1 no varia i es continua utilitzant la comanda OB (Output Bit):

Activar senyal 1	OB +9
Desactivar senyal 1	OB -9

A l'hora de llegir l'estat d'una senyal, el primer pas és emmagatzemar al registre intern del manipulador la informació del port d'entrada extern. Per fer-ho s'executa la comanda ID (Input Direct). Tot seguit ja es pot veure si l'entrada està o no activada. Fins ara la unitat de control del manipulador ho comprovava amb la comanda TB (Test Bit):

Obtenir senyals externes	ID
Si senyal 1 activat, saltar a la instrucció 200	TB +9, 200

Ara, però, cal que la informació sobre l'estat de les senyals arribi al NetSight II. La comanda DR (Data Read) és una instrucció exclusiva per a la comunicació via RS-232, la funció de la qual és transmetre el contingut del registre intern pel port de sortida:

Obtenir senyals externes	ID
Enviar contingut del registre intern pel port de sortida	DR

Si després d'executar aquesta comanda l'aplicació fa una lectura del port sèrie s'obté una cadena de nombres hexadecimals (iniciada amb "&H" i acabada amb el caràcter de retorn) que inclou informació sobre l'estat de les senyals externes del RV-M1. En concret és el quart valor de la cadena el que porta tal informació (veure figura 3.12).

Lectura[3]	Estat senyal 1	Estat senyal 2
0	Desactivat	Desactivat
1	Activat	Desactivat
2	Desactivat	Activat
3	Activat	Activat

Figura 3.12: Taula d'estats de les senyals de comunicació

4. VISIÓ ARTIFICIAL

Aquest capítol descriu com s'aplica la visió artificial en la cel·la de transport, detallant el procés de calibració de la càmera, de captura d'imatges i de tractament i localització de blobs.

4.1. ANÀLISI

L'objectiu del sistema de visió és localitzar la posició de la peça dins la safata de transport. Concretament es pretenen obtenir les coordenades del seu centre per tal de poder generar la instrucció de recollida de la peça en un procés posterior.

4.1.1 Disposició de la càmera

La càmera connectada al NetSight II es col·locarà en la zona de manipulació, just a sobre de l'espai de treball del robot (veure figura 4.1). Quan la safata s'aturi en aquest punt la càmera realitzarà una captura de la seva superfície, en la que s'hi haurà de determinar la presència de peces i la seva posició.

Per facilitar l'anàlisi les imatges obtingudes mostraran una representació bidimensional de les peces (només es veurà la part superior).

Figura 4.1: Disposició de la càmera en la cel·la

4.1.2. Calibració de la càmera

La calibració de la càmera és un procés important en les aplicacions de visió per computador. Quan realitzem una captura la imatge resultant no és totalment exacta. Això és degut a que les lents de les càmeres no tenen un comportament lineal i provoquen una distorsió (veure figura 4.2). Amb la calibració aconseguim reduir aquest defecte per tal que les imatges adquirides es corresponguin al màxim amb el món real, obtenint d'aquesta manera valoracions més fidedignes. Donat que hem d'extreure informació de posició de les imatges és imprescindible aplicar aquest procés.

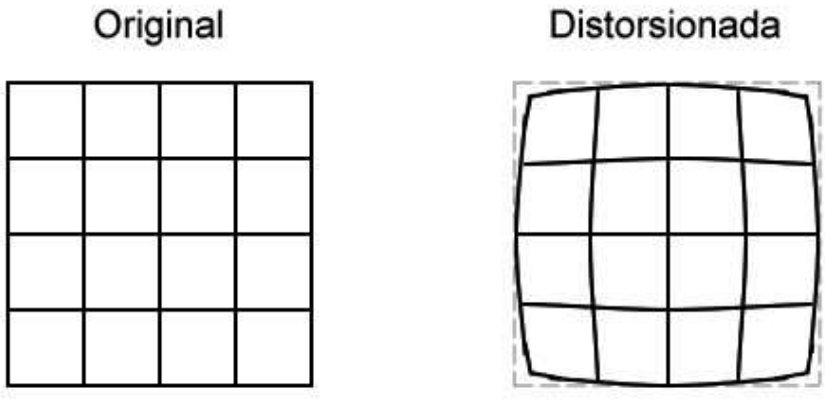


Figura 4.2: Distorsió radial generada per la lent

4.1.3. Tractament de la imatge

A l'hora d'analitzar una imatge és essencial poder distingir els objectes d'interès de "la resta". Per tant el primer pas que hem de fer abans de determinar la posició de la peça és ressaltar la forma de l'objecte.

La tècnica de segmentació que aplicarem és la més senzilla de totes: la binarització. Aquest procediment consisteix en reduir la informació de cada píxel d'una imatge en escala de grisos a dos valors possibles: mínim (0) o màxim (255). El resultat final és una imatge on els objectes importants es visualitzen de color negre sobre un fons blanc o a l'inrevés (veure figura 4.3).

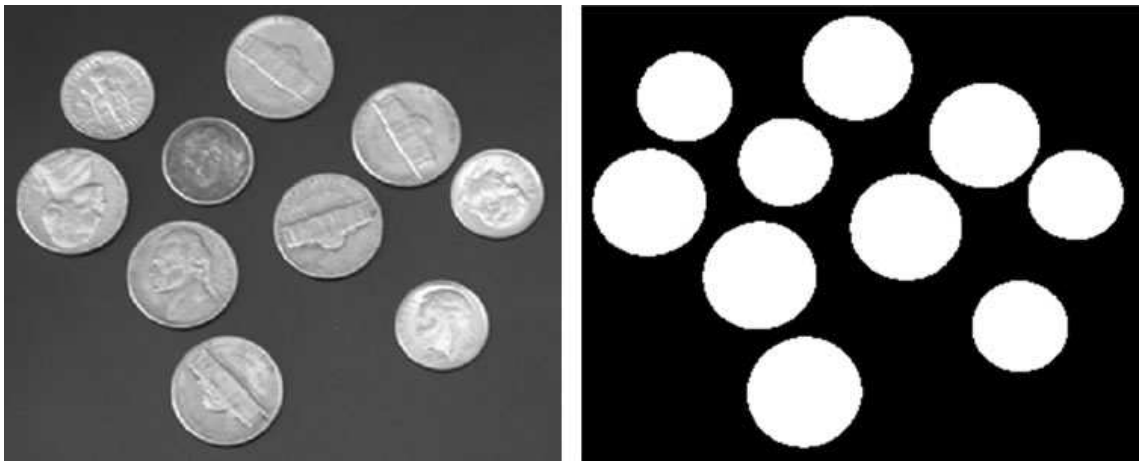


Figura 4.3: Exemple de binarització

4.2. DISSENY

El funcionament del sistema de visió integrat a la cel·la es divideix en dues parts:

- **El procés de calibració.** La calibració de la càmera es farà abans de posar en funcionament el procés de transport i classificació. Un cop completada es disposarà de la informació necessària per corregir les captures que es realitzin en la següent part.

- **L'anàlisi de la imatge.** L'anàlisi s'iniciarà quan la safata s'aturi a la zona de recollida. Consta dels següents passos (veure figura 4.4):

    1. Capturar una imatge de la safata.

    2. Eliminar la distorsió que provoca la lent de la càmera.

    3. Binaritzar la imatge per ressaltar les peces

    4. Localitzar els blobs i extreure'n les coordenades X i Y del seu centre.

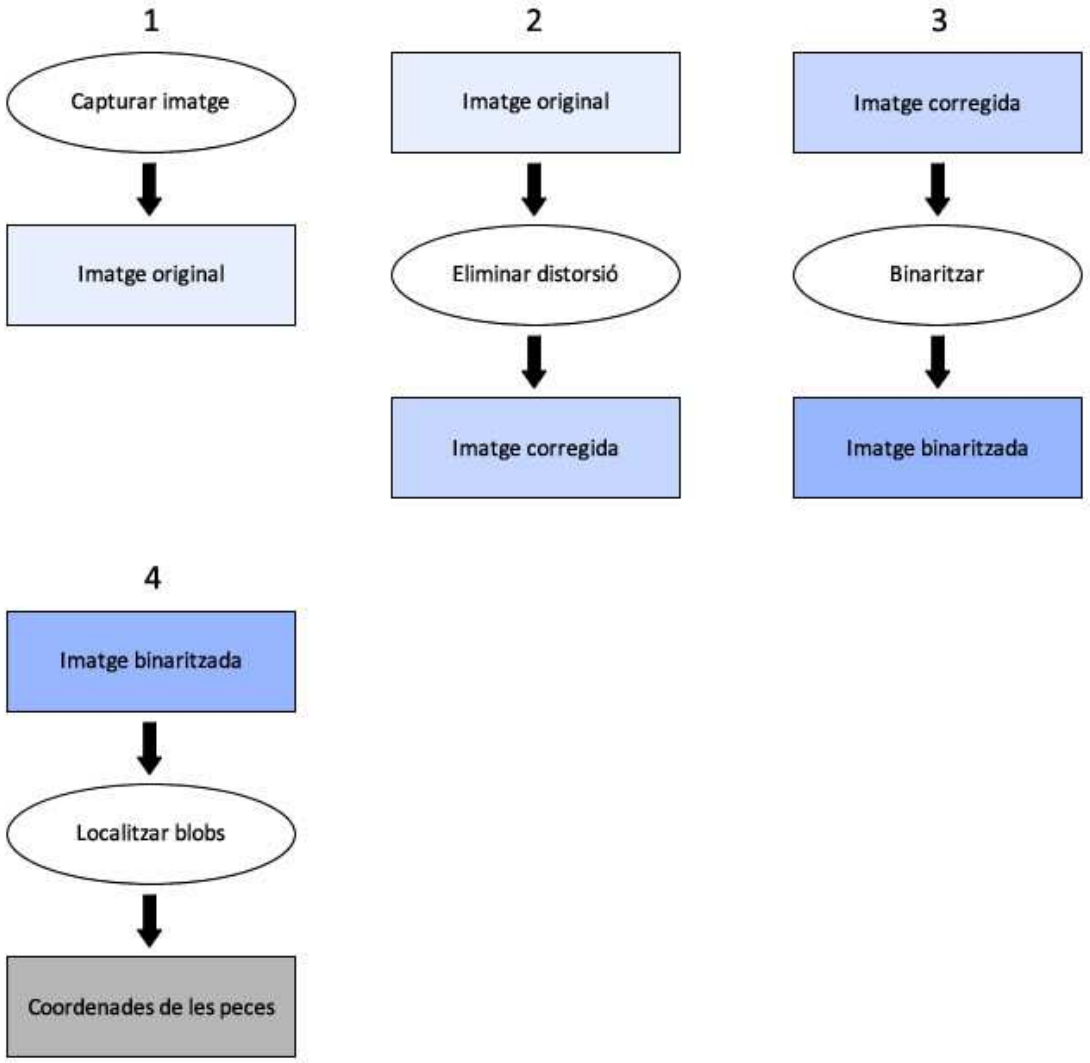


Figura 4.4: Esquema del procediment de l'anàlisi de la imatge

4.2.1. Aplicació

Totes les funcions de l'aplicació de visió artificial es trobaran a la classe *SistemaVisió*. Aquesta classe permetrà les següents accions:

- **Inicialitzar i calibrar la càmera.** El procés de calibració permetrà corregir la distorsió de la lent.
- **Capturar imatges.** Permetrà l'adquisició d'imatges monocromes en espais de memòria modificables.

- **Fer el tractament de les captures.** Eliminar la distorsió, binaritzar i detectar el nombre de peces i la seva situació dins la imatge.

Es crearà a més una nova classe anomenada *Coordenades* que actuarà de contenidor. En ella s'hi emmagatzemaran les coordenades trobades en l'anàlisi de la imatge. D'aquesta manera estaran disponibles pel procés posterior, el qual utilitzarà aquesta informació a l'hora de preparar la funció de recollida de la peça per part del manipulador.

Totes les imatges que es generen durant el procediment queden guardades al directori C:\:

- cap.bmp: captura original
- cap_cor.bmp: captura corregida
- cap_cor_bin.bmp: captura binaritzada

4.3. IMPLEMENTACIÓ

Les eines de visió artificial s'han programat sota la plataforma Windows utilitzant l'entorn de desenvolupament Visual C++ 6 i diverses llibreries que es mostren a continuació. Per més detall es poden consultar a l'annex el codi de les operacions bàsiques.

4.3.1. Llibreries

Per programar tot el codi referent a l'apartat de visió artificial s'utilitzaran llibreries especialitzades per aquesta tasca. El NetSight II inclou la seva pròpia llibreria per l'ús de càmeres, però les funcions de tractament d'imatge que incorpora són molt bàsiques. Per aquest motiu ha calgut cercar altres llibreries que permetessin realitzar els processos d'anàlisi que necessitàvem.

4.3.1.1. Llibreria IFC

La llibreria IFC (Imaging Foundation Classes) és una API per a C++ inclosa en el NetSight II. La seva funció principal és permetre la gestió de les càmeres que s'acoblen al sistema i l'adquisició d'imatges. Inclou a més algunes funcions per al tractament de les captures i per l'ús d'elements de hardware com les connexions i l'entrada/sortida.

S'utilitzarà per:

- Inicialització de la càmera.
- Captura d'imatges.
- Escriptura de les imatges a disc.

4.3.1.2. Llibreria OpenCV

La llibreria OpenCV (Open source Computer Vision library) és una API de codi obert desenvolupada per Intel. S'utilitza per a la programació d'aplicacions de visió per computador en C/C++. Inclou nombroses eines per al tractament de la imatge, des de l'aplicació d'operacions bàsiques fins a reconstrucció 3D.

S'utilitzarà per:

- Calibració de la càmera
- Correcció de les imatges capturades

4.3.1.3. Llibreria cvBlobsLib

La llibreria cvBlobsLib està basada en OpenCV i fou concebuda per detectar blobs en imatges binaritzades en imitació al mòdul de la Matrox Imaging Library (MIL). Incorpora també eines per filtrar els resultats segons diversos criteris i per extreure'n la informació resultant.

S'utilitzarà per:

- Detecció de blobs a la imatge
- Obtenció de les coordenades dels blobs

4.3.2. Inicialització de la càmera

Abans de fer ús de la càmera cal inicialitzar-la creant l'objecte de captura i reservant els espais de memòria on s'emmagatzemaran les imatges adquirides. Per aquesta tasca s'utilitza la llibreria IFC.

L'espai de memòria (o buffer) és una taula de tipus BYTE que conté els valors de cada píxel de la imatge (en un rang de 0 a 255). La longitud d'aquesta taula ve determinada pel nombre de píxels que componen la imatge multiplicat pel nombre de bytes d'informació que conté cada píxel.

En el cas d'una imatge a color cada píxel té 3 bytes d'informació (un pel vermell, un pel verd i un pel blau). En canvi una imatge en blanc i negre només disposa d'un byte d'informació per cada píxel (veure figura 4.5).

Donat que la càmera que hem utilitzat disposa d'una resolució de 648 x 490 píxels i és monocroma, la mida dels buffers que caldrà crear seran de 317520 bytes. En total se n'utilitzaran tres:

- **m_imgBuf**: imatge original, tal i com ha sigut captada per la càmera.
- **m_imgBufCor**: imatge corregida, el resultat de reduir la distorsió de la lent sobre la imatge.
- **m_imgBufCorBin**: imatge corregida i binaritzada, el resultat de convertir la imatge en blanc i negre per ressaltar-ne les peces.

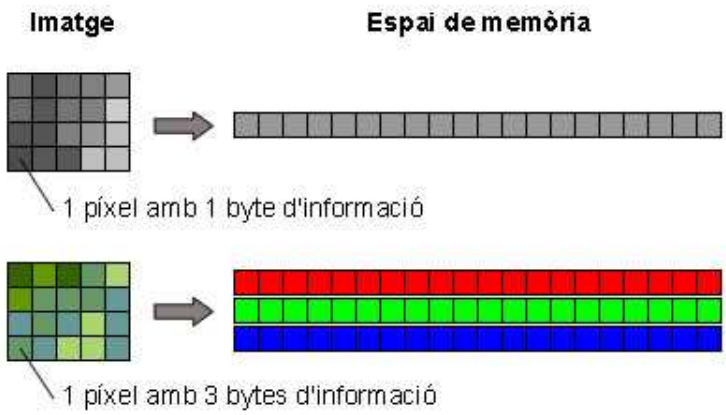


Figura 4.5: Espai de memòria necessari segons el tipus d'imatge

Un cop definits els espais de memòria podrem fer-los servir com a contenidor de les captures. L'avantatge que presenten aquests buffers és que ens permeten accedir a cada píxel de la imatge i manipular-lo directament.

4.3.3. Cal·libració de la càmera

La cal·libració es farà utilitzant els algoritmes de que disposa la llibreria OpenCV. A partir d'una captura d'un patró senzill (un tauler d'escacs) s'estimaran els valors dels paràmetres intrínsecs de la càmera i dels coeficients de distorsió. Aquestes matrius ens permetran realitzar la correcció de qualsevol imatge.

El primer pas és preparar el patró de calibració. Mitjançant un programa de dibuix s'ha de crear i imprimir un tauler d'escacs que ocupi tant d'espai de captura com sigui possible. El nombre de caselles ha de ser prou alt per disposar de prous punts de referència, però tampoc ha de ser excessiu per evitar que el temps de càlcul de l'algoritme sigui massa llarg.

Nosaltres utilitzarem un patró de 14x10 caselles de 40x40 píxels cadascuna (veure figura 4.6).

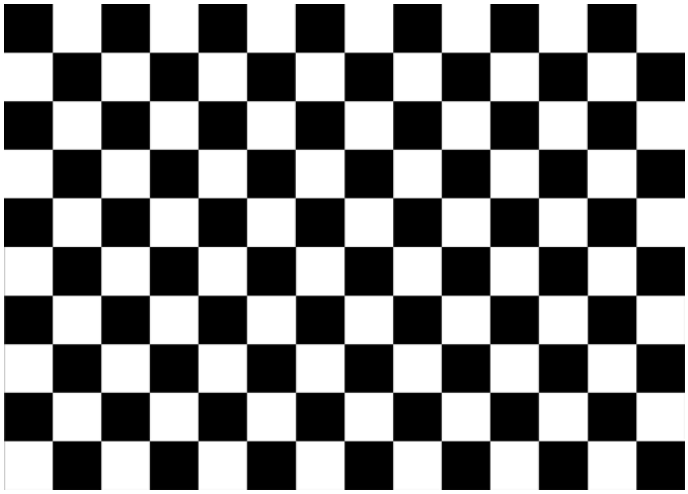


Figura 4.6: Patró utilitzat en la calibració

Situem la plantilla sobre la safata de transport (veure figura 4.7) i realitzem una captura de l'escena (veure figura 4.8).

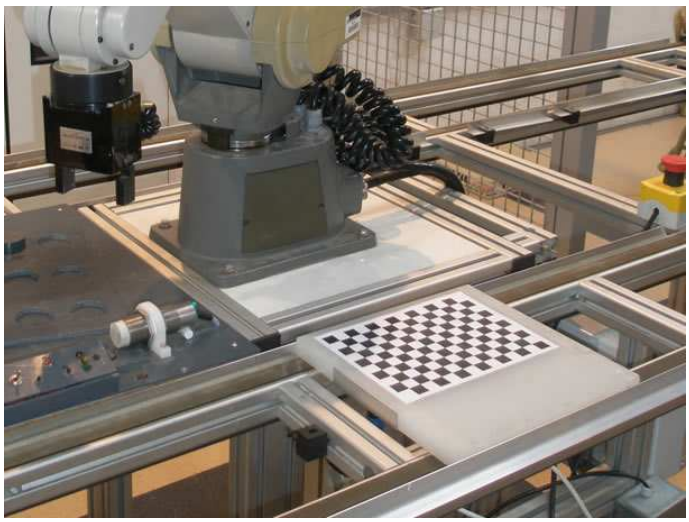


Figura 4.7: Col·locació del patró de calibració sobre la safata

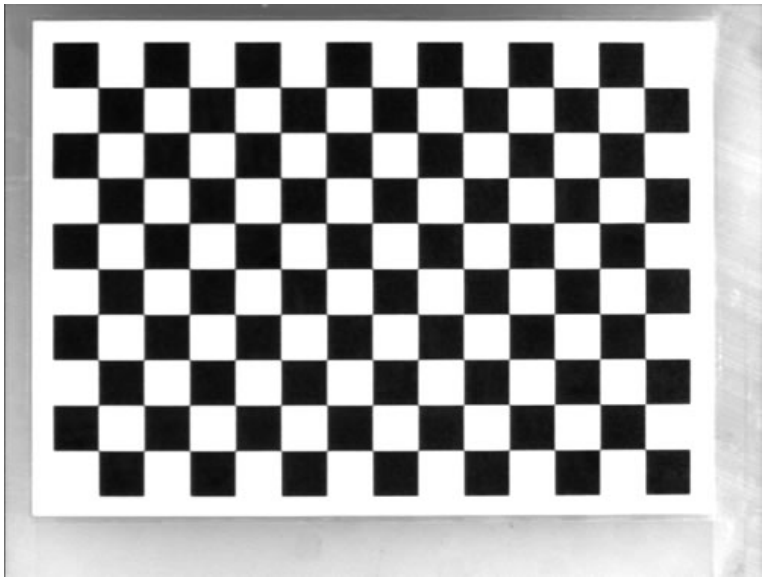


Figura 4.8: Captura del patró feta per la càmera

La primera funció que s'aplica (*cvFindChessboardCorners*) detecta totes les cantonades que formen les caselles del tauler (veure figura 4.9). És necessari que es detectin totes les cantonades (117 en aquest cas) o el procés no continuarà.

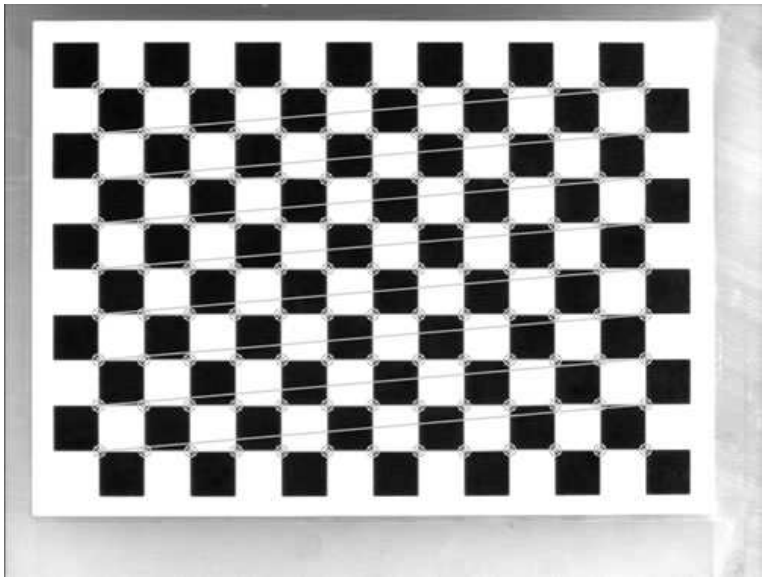


Figura 4.9: Cantonades del patró localitzades

La següent funció (*cvCalibrateCamera2*) utilitza la informació recopilada de la imatge així com la dada del nombre de píxels que té cada quadrat per calcular dues matrius: la de paràmetres intrínsecs i la de coeficients de distorsió. Aquestes dades són suficients per a la posterior correcció de la imatge.

4.3.4. Correcció de la imatge

Novament utilitzarem la llibreria OpenCV per realitzar la correcció de les imatges. La funció aplicada (*cvUndistort2*) fa servir les matrius de paràmetres intrínsecs i de coeficients de distorsió obtinguts amb la calibració per realitzar la correcció d'una imatge donada. Concretament el que fa és deformar la imatge de manera que les línies d'aquesta siguin paral·leles a l'escena.

El resultat és poc apreciable, però a la figura 4.10 podem observar com la imatge corregida presenta un lleuger estirament de les puntes, deixant unes petites franges negres a dalt i a baix

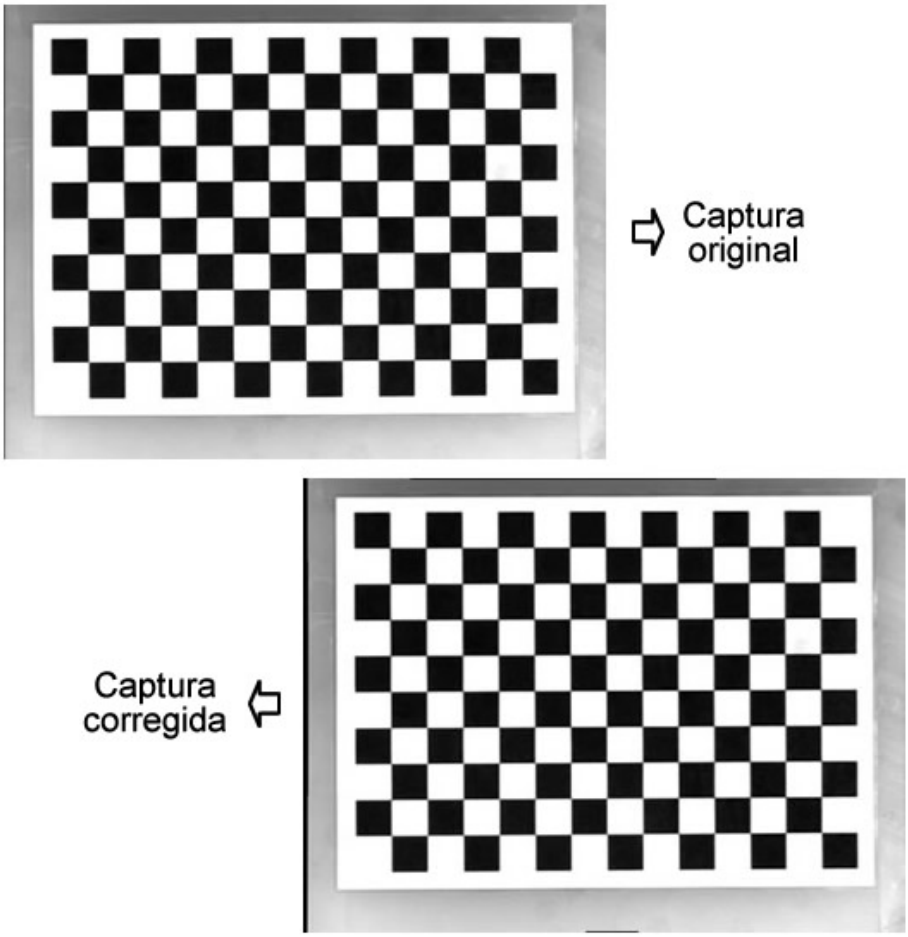


Figura 4.10: Comparació entre la imatge original i la corregida

4.3.5. Binarització

El procés de binarització consistirà en modificar cada valor del buffer de la imatge corregida segons el llindar (θ) escollit, utilitzant la següent fórmula:

si m_imgBufCor[i] < θ	**llavors**	m_imgBufCorBin[i] = 0
altrament		m_imgBufCorBin[i] = 255

El resultat d'aplicar aquesta funció a cada valor del buffer serà una imatge amb les peces de color negre (valor del píxel igual a 0) sobre un fons totalment blanc (valor del píxel igual a 255).

La dificultat d'aquesta tècnica de segmentació resideix en trobar el valor òptim del llindar. Existeixen algoritmes per fer-ho basant-se en l'anàlisi de l'histograma de la imatge. En aquest projecte s'han testejat dos: l'Isodata i l'Otsu.

4.3.5.1. Algoritme Isodata

Aquesta tècnica iterativa va ser desenvolupada per Ridler i Calvard. L'histograma és inicialment segmentat en dos parts començant per un valor determinat de llindar (normalment la meitat del rang màxim). A continuació es calcula la mitjana de l'escala de grisos del segment de l'objecte i la del segment del fons. Un nou valor per al llindar és computat a partir del promig dels dos càlculs anteriors. El procés es repeteix fins que el valor del llindar no canvia.

El codi que s'ha implementat és una adaptació d'un l'algoritme en llenguatge Java creat pel programador sota el pseudònim yannart.

4.3.5.2. Algoritme Otsu

El mètode Otsu es basa en l'existència de dues classes definides en la imatge, buscant el punt que minimitzi la variància entre aquestes dues classes. La implementació d'aquest és molt més costosa que l'Isodata, tot i que es considera que els seus resultats són millors.

El codi que s'ha implementat és un algoritme en llenguatge C++ creat per Joerg Schulenburg i posteriorment modificat per Ryan Dibble.

4.3.6. Detecció de blobs

Per localitzar la peça dins la imatge binaritzada s'ha utilitzat la llibreria cvBlobsLib. Amb ella es realitza la localització i etiquetatge de regions, numerant cada blob que es troba. Per a cadascun d'ells es recull informació, de la qual se n'ha fet ús de la següent:

- Àrea
- Circularitat
- Posició en l'eix X
- Posició en l'eix Y

Una altra utilitat que té la llibreria és el filtratge dels resultats. D'aquesta manera podem delimitar els blobs útils segons les seves característiques (per exemple que l'àrea sigui més gran que 1000 i més petita que 1500) i descartar artefactes que hagin aparegut a la imatge.

Analitzant els resultats obtinguts en funció de la distància a la que hem situat la càmera de la safata, nosaltres hem aplicat el següent filtratge:

- Àrea del blob més gran que 10000
- Àrea del blob més petita que 12000
- Circularitat del blob més gran que 1
- Circularitat del blob més petita que 1.3

Un cop detectat el blob es guarden les coordenades del seu centre. El següent pas és relacionar aquest punt de la imatge amb una posició de l'espai de treball del robot, procés que s'explica en el següent capítol.

5. RECOLLIDA DE LA PEÇA

En aquest capítol s'hi exposa el procés de recollida de la peça, des de la generació del vector de posició del manipulador fins a la programació de l'autòmat.

5.1. ANÀLISI

El sistema de visió ja està programat per poder extreure les coordenades de les peces que es troben dins la safata de transport. Ara cal generar la instrucció de recollida del manipulador en funció d'aquesta informació. Aquest procediment fa ús de dades generades pel sistema de visió com de dades generades pel manipulador.

5.1.1. Moviment de l'element terminal del RV-M1

El RV-M1 té la capacitat de realitzar moviments del seu element terminal en coordenades cartesianes (veure figura 5.1). Amb aquest sistema un vector de posició té el següent format:

[coord. eix X],[coord. eix Y],[coord. eix Z],[angle de pitch],[angle de roll]

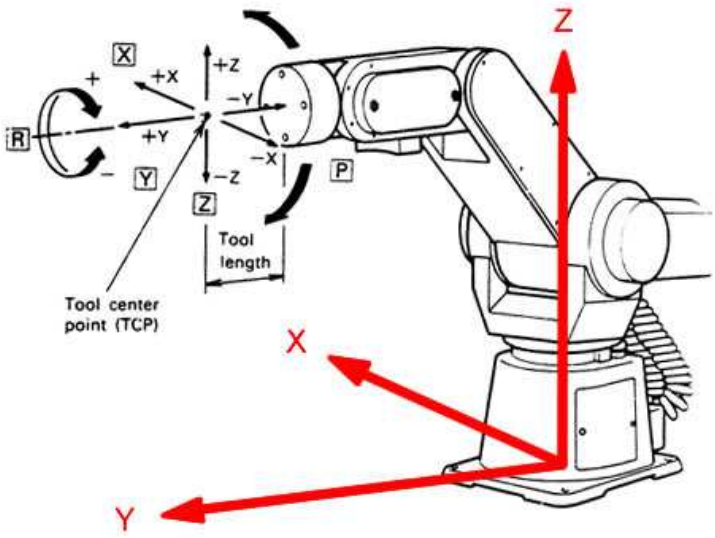


Figura 5.1: Eixos de coordenades del robot

Modificant els valors de X i Y podem desplaçar l'element terminal pel sistema de coordenades sense variar la seva orientació. D'aquesta manera si determinem una posició de referència en que l'element terminal es situï just a sobre de la safata podrem desplaçar-lo per tota la superfície de recollida (veure figura 5.2).

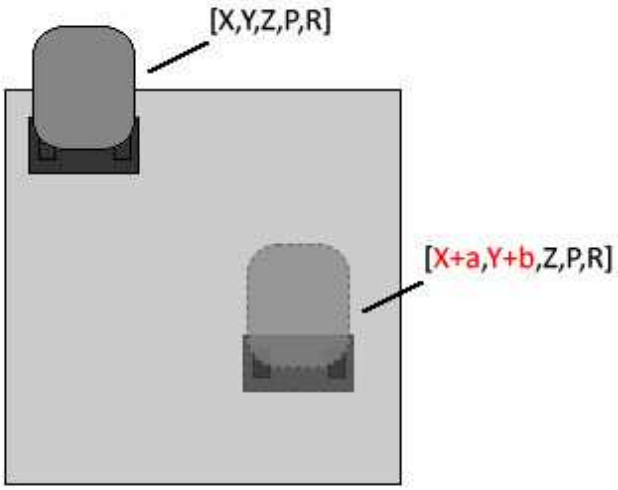


Figura 5.2: Desplaçament de l'element terminal sobre la safata

Amb aquest mètode es redueix el sistema a només dos eixos situats en el mateix pla que la safata i, per tant, de la imatge capturada pel sistema de visió (veure figura 5.3). D'aquesta manera la situació de la peça dins la imatge permetrà determinar la variació que s'ha d'aplicar a les coordenades X i Y del robot. Per fer-ho però caldrà buscar una expressió que relacioni el sistema de coordenades de la imatge amb el del RV-M1.

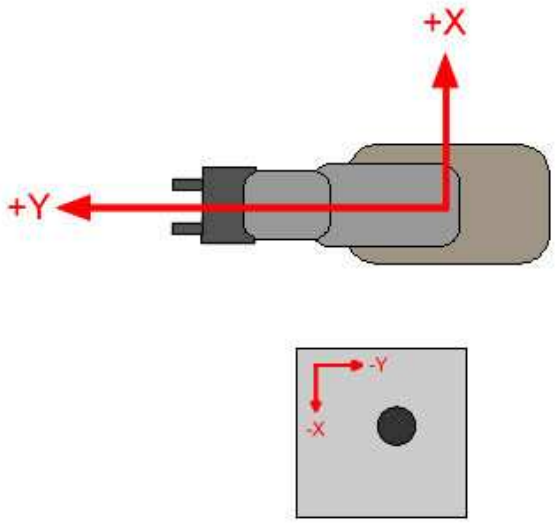


Figura 5.3: Eixos de coordenades del robot en el pla de la safata

5.1.2. Programació de l'autòmat

Un cop implementat el mètode de recollida de les peces faltarà transportar les peces a l'espai de treball del robot. Per posar en pràctica l'aplicació que s'ha creat en aquest projecte s'ha optat per programar una tasca molt més senzilla en la que només s'utilitzarà una part de la plataforma.

El procés de transport consistirà en portar les peces una a una des del punt d'inici (zona de seguretat) fins al punt de recollida. Només s'utilitzarà la cinta esquerra, la qual farà sempre el mateix recorregut.

A més del procés normal de transport es realitza una tasca prèvia que només es farà un sol cop: desplaçar la safata al punt de recollida per poder realitzar la calibració de la càmera i la delimitació de l'àrea de recollida. Quan s'hagin completat aquestes accions la safata iniciarà el cicle de transport descrit anteriorment.

5.2. DISSENY

Per trobar el vector de posició que situa l'element terminal sobre d'una peça a recollir es seguirà el següent procediment:

- **Delimitar una posició de referència.** Aquesta posició situarà l'element terminal orientat perpendicularment a la safata, preparat per agafar una peça.
- **Trobar les coordenades de la peça en els eixos X i Y de la imatge.** El sistema de visió analitzarà la posició de la peça en la safata i guardarà les coordenades del seu centre.
- **Traslladar les coordenades de la imatge als eixos del manipulador.** Una funció de relació convertirà les coordenades X i Y de la peça en la imatge en coordenades X i Y en l'espai de treball del manipulador.
- **Modificar els valors de X i Y de la posició de referència.** Canviant els valors de X i Y pels trobats en el punt anterior s'obtindrà el vector de posició que situa el gripper a sobre de la peça (veure figura 5.4).

Un cop obtingut el vector de posició s'enviarà al manipulador amb la comanda MP (Move Position).

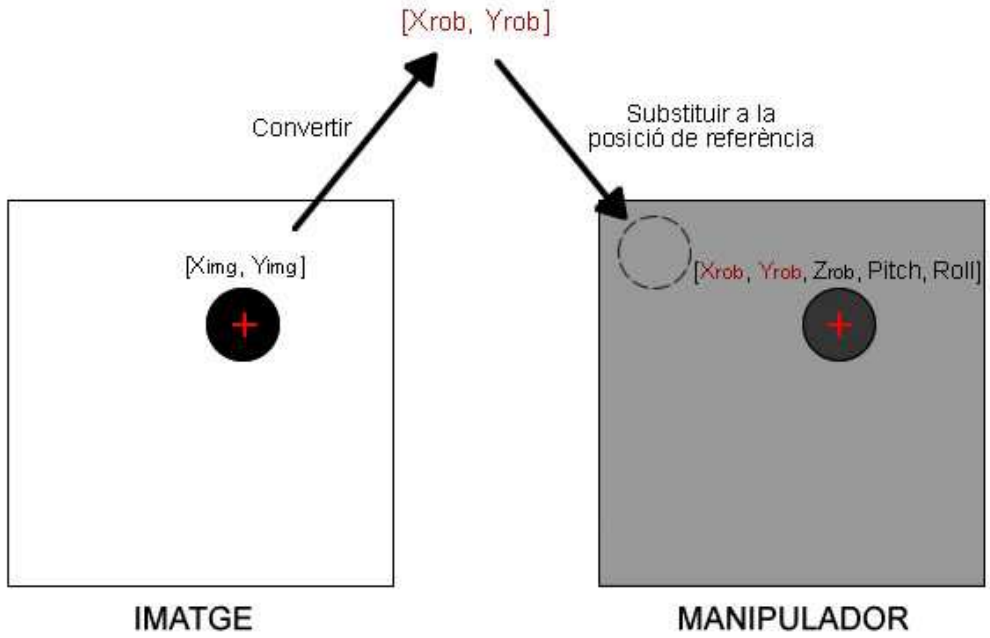


Figura 5.4: Esquema del procés conversió de coordenades

5.2.1. L'àrea de recollida

Per poder convertir un punt dels eixos de la imatge als eixos del robot cal cercar una funció que relacioni els dos sistemes.

El mètode empleat ha sigut definir una àrea de recollida que quedi delimitada en tots dos sistemes de coordenades. D'aquesta manera qualsevol punt dins d'aquesta zona en el pla de la imatge es pot traslladar al pla del robot aplicant la conversió adequada.

Per fer-ho es marcaran dos punts corresponents a les cantonades oposades de l'àrea (veure figura 5.5). Aquests punts seran en realitat dues de les peces utilitzades en la cel·la, que seran col·locades pel manipulador. Cada cop que el robot dipositi una peça emmagatzemarà el vector de posició actual. Tot seguit el sistema de visió analitzarà la safata i guardarà les coordenades de les dues peces de delimitació.

Un cop delimitada, qualsevol peça detectada pel sistema de visió que tingui el seu centre a l'interior de l'àrea podrà ser recollida pel manipulador.

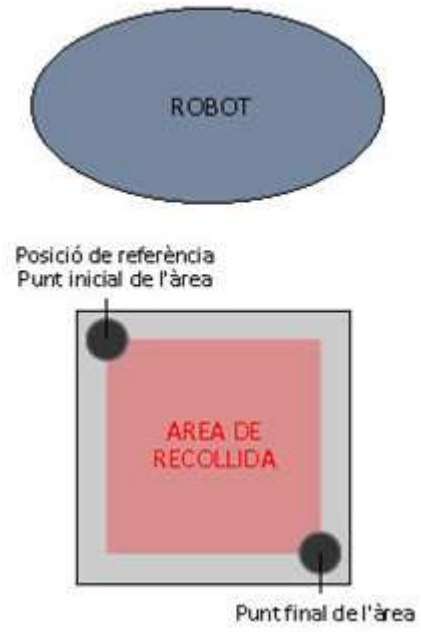


Figura 5.5: Àrea de recollida

5.2.3. Posicions del manipulador

Les posicions definides pel manipulador són les següents:

- 230: Posició de recollida de la primera peça de delimitació, situada en el palet.
- 232: Posició de recollida de la segona peça de delimitació, situada en el palet.
- 234: Posició d'espera de l'arribada de la safata.
- 235: Posició de col·locació de la primera peça de delimitació en el punt inicial de l'àrea de recollida (o posició de referència).
- 237: Posició de col·locació de la segona peça de delimitació en el punt final de l'àrea de recollida.
- 239: Posició sobre la caixa on es dipositen les peces.

A part d'aquestes posicions fixes, existeix la posició variable de recollida de la peça. Un cop el robot s'ha situat sobre l'objecte (al enviar -li el vector de posició que es calcula a partir de l'anàlisi de la imatge) es guarda la posició en la número 240 per tal de simplificar les següents instruccions de moviment.

5.2.4. Programació de l'autòmat: grafcet de 1er nivell

El grafcet de primer nivell és una descripció global i poc detallada del procés desenvolupat per l'autòmat. En ell s'hi representen les accions que volem que realitzi el PLC sense especificar la seva implementació real en termes mecànics i de programació.

La figura 5.6 mostra el grafcet que s'utilitzarà en el procés de la cel·la nova.

Com s'observa no hi ha diferents estats pel procediment de transport i pel de calibració i delimitació de l'àrea de recollida. Donat que el moviment és el mateix (zona de seguretat ⟶ zona de manipulació ⟶ zona de seguretat) per a totes dues accions es poden aprofitar els mateixos estats.

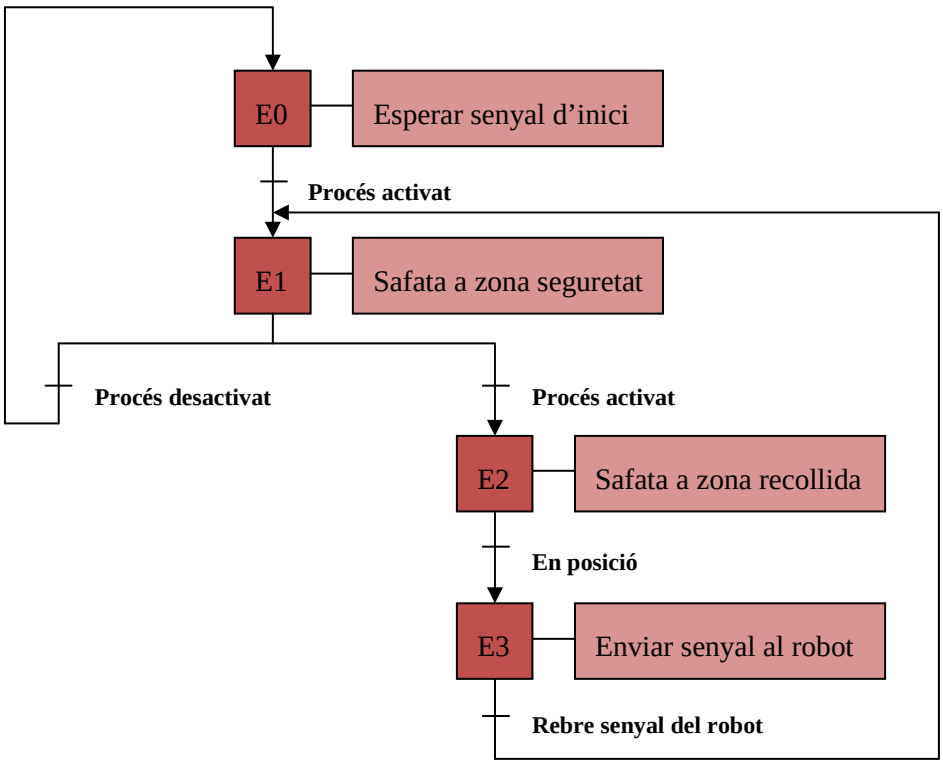


Figura 5.6: Grafcet de primer nivell del procés

5.3. IMPLEMENTACIÓ

S'afegiran noves funcions a les classes *SistemaVisió* i *Robot*. Aquestes classes tindran com a nexe la classe *Coordenades*, amb la qual compartiran la informació que genera cada sistema.

La programació de l'autòmat s'ha realitzat amb l'aplicació PL7 Pro v4.4.

5.3.1. Delimitar l'àrea de recollida

El procediment per delimitar l'àrea consta dels següents passos (veure figura 5.7):

1- El manipulador agafa una peça del palet i la diposita a la posició de referència. Quan la col·loca l'aplicació li sol·licita el seu vector de posició actual i l'emmagatzema per poder generar posteriorment les noves posicions a partir de la seva modificació. D'aquest vector se n'extreu el punt inicial de l'àrea de recollida: **iniXrobot** i **iniYrobot**.

2- El manipulador agafa una segona peça del palet i la diposita a la posició més allunyada de la primer peça, una posició que també ha estat definida prèviament. De nou se n'obté el vector de posició actual (sense emmagatzemar-lo) i se n'extreu el punt final de l'àrea de recollida: **fiXrobot** i **iniXrobot**.

3- Un cop col·locades les peces de delimitació el sistema de visió realitza una captura de la safata. El seu anàlisi retorna les coordenades dels dos punts en la imatge: **iniXimatge**, **iniYimatge**, **fiXimatge** i **fiYimatge**.

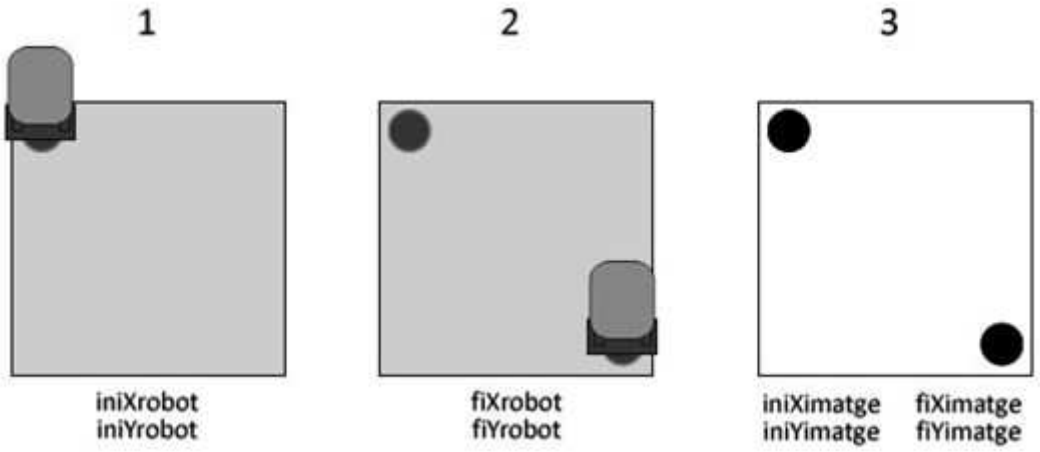


Figura 5.7: Delimitació de l'àrea de treball

Un cop es disposa dels punts de delimitació ja es pot calcular l'àrea de recollida tant en el pla de la imatge com en el del manipulador:

- **ampladaARrobot** = fiXrobot - iniXrobot
- **alçadaARrobot** = fiYrobot - iniYrobot
- **ampladaARimatge** = fiXimatge - iniXimatge
- **alçadaARimatge** = fiYimatge - iniYimatge

Totes les dades obtingudes en aquest procediment queden emmagatzemades en l'objecte de la classe *Coordenades*.

5.3.2. Conversió de coordenades

Ara que ja tenim la representació de l'àrea de recollida en els dos plans podem preparar una funció que traslladi les coordenades d'una peça en els eixos de la imatge (**pecaXimatge**, **pecaYimatge**) a un punt en els eixos del robot (**pecaXrobot**, **pecaYrobot**).

Abans de tot, però, cal puntualitzar algunes coses:

- Els eixos X i Y de la imatge estan invertits respecte als eixos X i Y del robot. Per tant a l'hora de fer el càlcul caldrà tenir en compte que la X d'un sistema de coordenades va en funció de la Y de l'altre sistema (veure figura 5.8).

- El moviment del robot sobre la safata es fa en els eixos negatius, pel que caldrà vigilar amb els signes i utilitzar el valor absolut quan sigui necessari (veure figura 5.8).

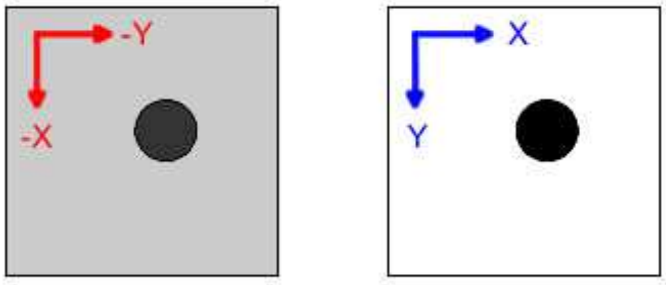


Figura 5.8: Relació dels eixos de coordenades del manipulador i de la imatge

La conversió d'un punt es fa amb una senzill càlcul:

Xrobot = iniXrobot - ((Yimatge - iniYimatge) * ampladaARrobot / alcadaARimatge);
Yrobot = iniYrobot - ((Ximatge - iniXimatge) * alcadaARrobot / ampladaARimatge);

5.3.3. La instrucció de recollida

Per anar a buscar la peça utilitzarem la comanda MP (Move Position), amb la qual podem desplaçar l'element terminal del robot a les coordenades que vulguem dintre de l'espai de treball. La seva forma és la següent:

MP [coord. eix X],[coord. eix Y],[coord. eix Z],[angle de pitch],[angle de roll]

Per preparar la instrucció disposem del vector de coordenades de la posició de referència. Un exemple seria:

-283.0,+110.6,+91.7,-90.0,+87.1

Aquesta posició ha estat emmagatzemada en forma de cadena de caràcters:

-	2	8	3	.	0	,	+	1	1	0	.	6	,	+	9	0	...	.	1

Declarem una nova cadena de caràcters on els primers valors seran la comanda MP i tot seguit el vector de la posició de referència:

M	P		-	2	8	3	.	0	,	+	1	1	0	.	6	,	...	.	1

Finalment substituïm els caràcters corresponents als eixos X i Y amb els valors de la posició on es troba la peça. Per exemple si la peça es troba a [-310.2 , -5.1] seria:

M	P		-	3	1	0	.	2	,	-	0	0	5	.	1	,	...	.	1

Ja podem enviar la instrucció al manipulador. Si les coordenades són vàlides l'element terminal quedarà just a sobre de la peça que volem agafar. Amb la comanda HE 240 (Here) li direm al robot que guardi la posició actual en la posició 240, facilitant les instruccions d'aproximació i allunyament que conformen la recollida de la peça.

5.3.4. Programació de l'autòmat: grafcet de 2on nivell

El grafcet de segon nivell ja és una representació amb les senyals reals utilitzades per l'autòmat a l'hora d'executar el procés. L'aplicació PL7 Pro v4.4. ens permet generar fàcilment aquest grafcet i enviar-lo a l'autòmat per a la seva posterior execució (veure figura 5.8).

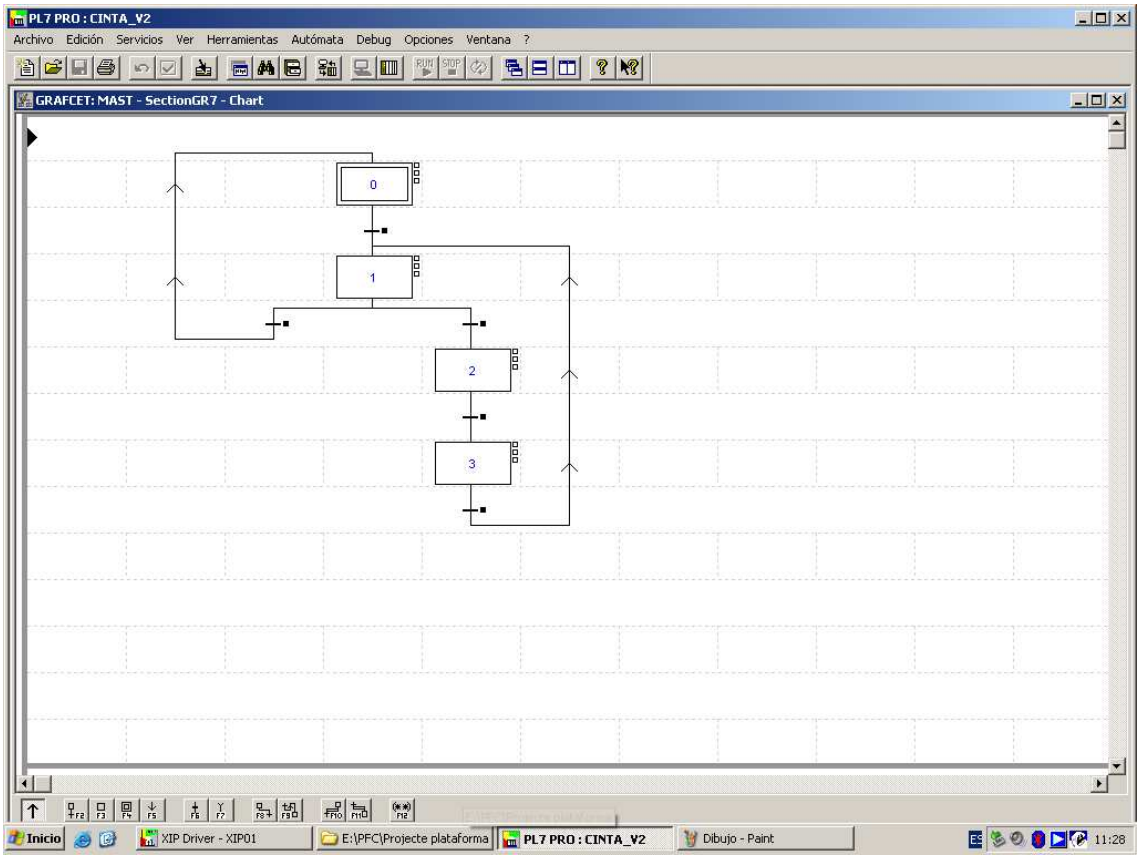
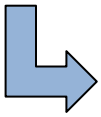


Figura 5.9: Grafcet de segon nivell del procés

A continuació es detallen les accions que es duen a terme a cada estat així com les condicions que han de complir-se per saltar al següent estat.

Estat 0:

- No es realitza cap acció. L'autòmat roman en repòs a l'espera que el robot li comuniqui el inici del procés.

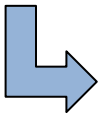


Condicions d'activació de l'estat 1

- Avís del robot indicant el inici del procés (senyal d'entrada %I1.10 activat).

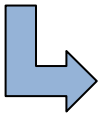
Estat 1:

- Activar motor de la cinta dreta per que la safata es desplaci cap a la zona de seguretat.
- Activar balisa lluminosa de color vermell.



Condicions d'activació de l'estat 0

- Avís del robot indicant la finalització del procés (senyal d'entrada %I1.10 desactivat).

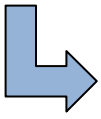


Condicions d'activació de l'estat 2

- Safata situada a la zona de seguretat (senyal del sensor situat a CD3).

Estat 2:

- Activar inversió de gir de la cinta dreta per que la safata es desplaci cap a la zona de recollida.
- Activar balisa lluminosa de color vermell.

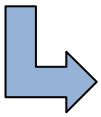


Condicions d'activació de l'estat 3

- Safata situada a la zona de recollida (senyal del sensor situat a CD2).

Estat 3:

- Avisar al robot per que comenci la seva activitat (ja sigui realitzar la delimitació de l'àrea de recollida com recollir la peça que ha transportat la safata, amb la senyal de sortida %Q2.1).



Condicions d'activació de l'estat 1

- Avís del robot sol·licitant la recerca d'una peça (senyal d'entrada %I1.11 activat).

A l'annex s'hi pot consultar amb més detall la programació de l'autòmat.

6. EXECUCIÓ I RESULTATS

En aquest capítol es presenta l'execució de tot el procés, els canvis que ha calgut realitzar i els resultats obtinguts.

6.1. PROVES

Les proves que s'han fet han mostrat algunes limitacions i la necessitat d'aplicar alguns canvis en els processos implementats.

6.1.1. L'àrea de recollida

Hem determinat que l'àrea de recollida és l'espai dins la safata de transport en el qual una peça pot ser agafada pel manipulador. Idealment aquesta àrea ocuparia tota la safata, però la disposició actual del braç robòtic sobre la cel·la de fabricació impedeix que l'element terminal pugui arribar als punts més allunyats de la seva base. Això ens obliga a definir una àrea de recollida més petita.

6.1.2. Binarització

S'ha plantejat l'ús de dos algoritmes diferents per localitzar un valor de llindar òptim en el procés de binarització: l'algoritme Isodata i l'algoritmes Otsu. Els resultats no han estat els esperats, doncs en diverses proves el llindar generat no permetia una correcta segmentació de les peces. A continuació es mostren alguns resultats:

- Isodata → llindar = 112 (veure figura 6.1)
- Otsu → lindar = 108 (veure figura 6.2)
- Valor manual → llindar = 45 (veure figura 6.3)

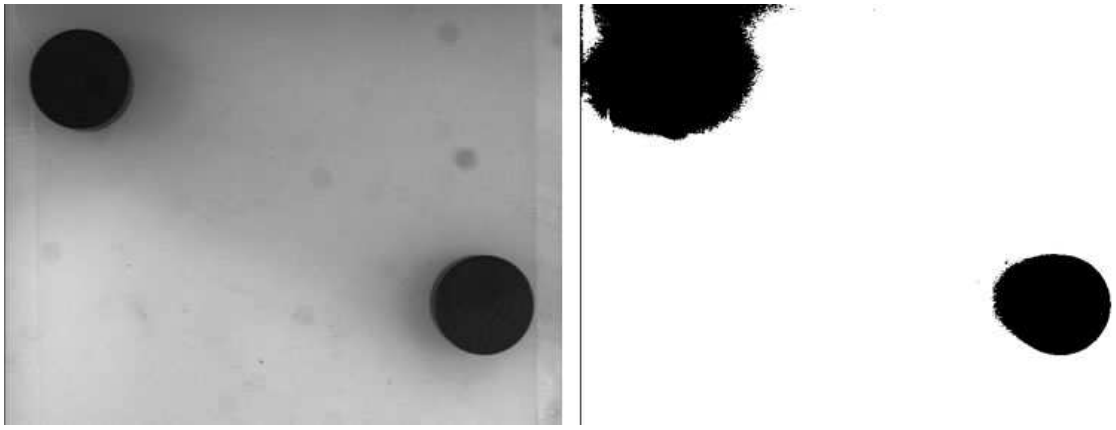


Figura 6.1: Binarització amb un valor de llindar de 112

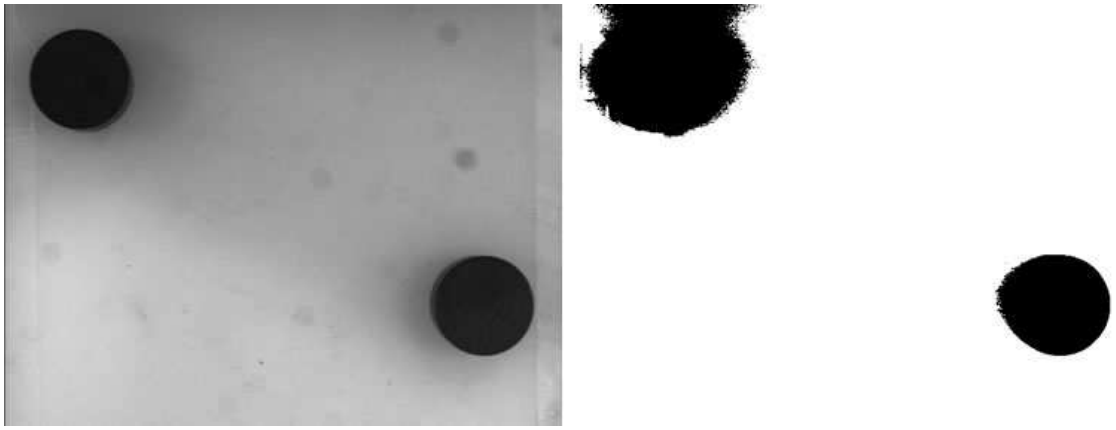


Figura 6.2: Binarització amb un valor de llindar de 108

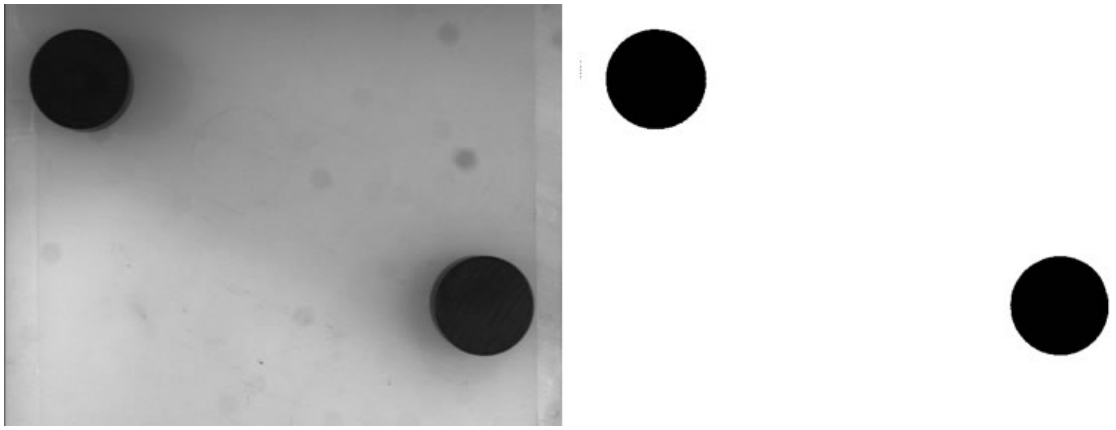


Figura 6.3: Binarització amb un valor de llindar de 45

El problema ve donat per la presència d'ombres que provoquen una estimació dolenta del llindar. Això és degut a que no es manté un control sobre la il·luminació. Una segmentació incorrecta provoca un error d'execució durant la detecció dels blobs.

Per evitar aquest problema cal modificar l'obertura de la lent per tal de reduir les ombres en la imatge. Però com la il·luminació de la sala varia constantment cal fer la modificació abans de cada execució. Amb una obertura correcte l'algoritme Otsu sembla tenir més adaptabilitat per realitzar una segmentació correcte de les peces, pel que serà el que finalment quedi implementat en la llibreria de visió artificial.

La solució ideal consistiria en col·locar un sistema d'il·luminació adequat en la cel·la de fabricació flexible, el qual permetria la correcte aplicació de l'algoritme i mantindria el llindar pràcticament constant. Malauradament no disposem de tal sistema.

6.2. APLICACIÓ DE PREPARACIÓ

Per tal de garantir que la càmera està ben situada (només captura l'interior de la safata) i que l'obertura de l'objectiu és adequada (la segmentació del blobs es realitza correctament) s'ha creat un senzill programa que a l'executar-lo mostra per pantalla una captura que tot seguit es binaritza en funció del llindar determinat per l'algoritme Otsu. A l'annex s'hi pot trobar més informació al respecte.

L'execució d'aquest programa és imprescindible abans de posar en marxa el procés de transport de classificació per tal d'aconseguir una disposició correcte de la càmera. D'aquesta manera evitem que a la meitat del procés aparegui un error degut a un reconeixement incorrecte de les peces.

6.3. EXECUCIÓ DEL PROCÉS

Tot seguit s'exposen els diferents passos d'execució de l'aplicació.

Abans de posar en marxa el programa cal activar l'autòmat. Aquest romandrà a l'estat 0 esperant la senyal que indiqui el inici del procés de transport i classificació (veure figura 6.4).

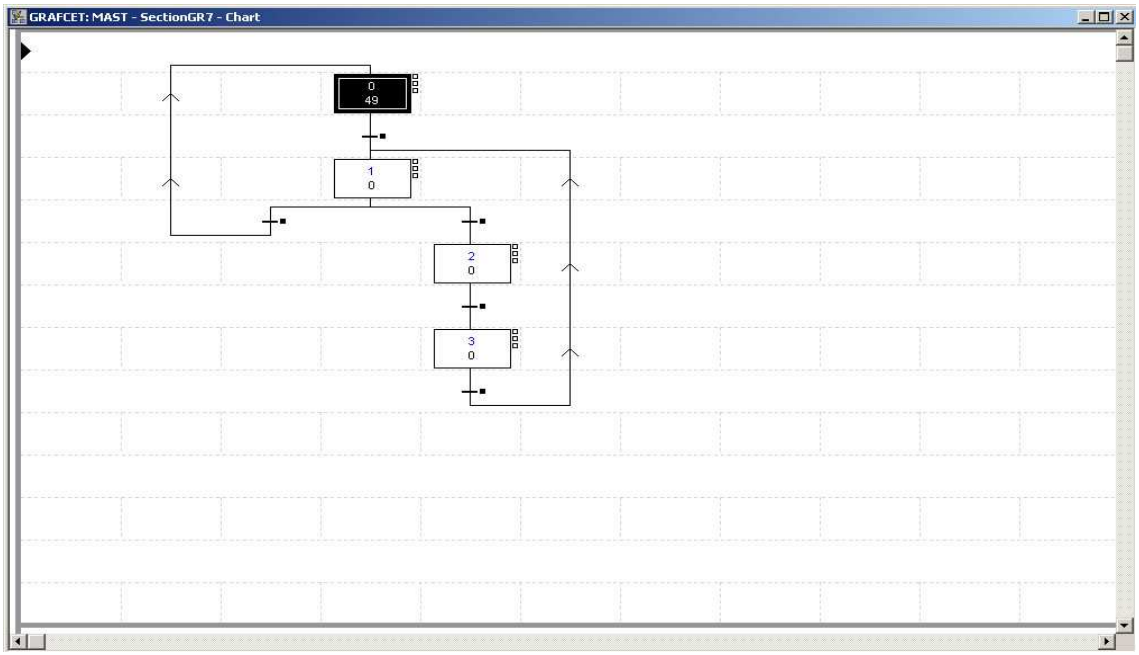


Figura 6.4: Autòmat en l'estat 0

Iniciem l'aplicació de transport i classificació. La seva execució consta de diverses fases tal i com veurem a continuació:

1. **Opertura del port sèrie.** El NetSight II crea el port de comunicació amb el manipulador i configura els atributs de comunicació i timeout (veure figura 6.5).

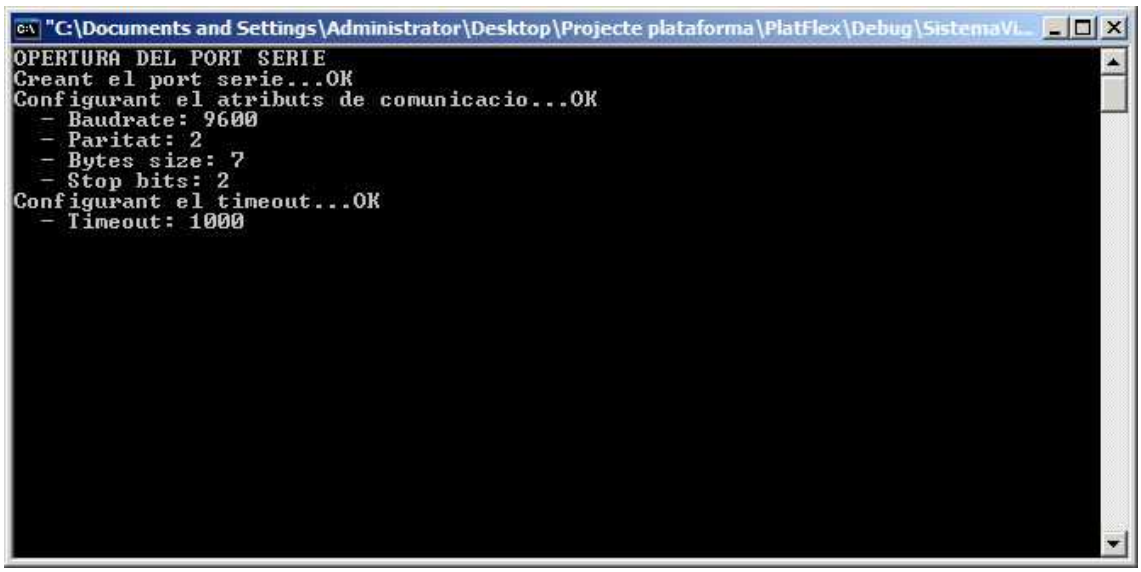


Figura 6.5: Captura de la consola (1)

2. **Inicialització de la càmera.** Es crea l'objecte de captura i es reserven els espais de memòria on es guardaran les imatges adquirides i les tractades (veure figura 6.6).

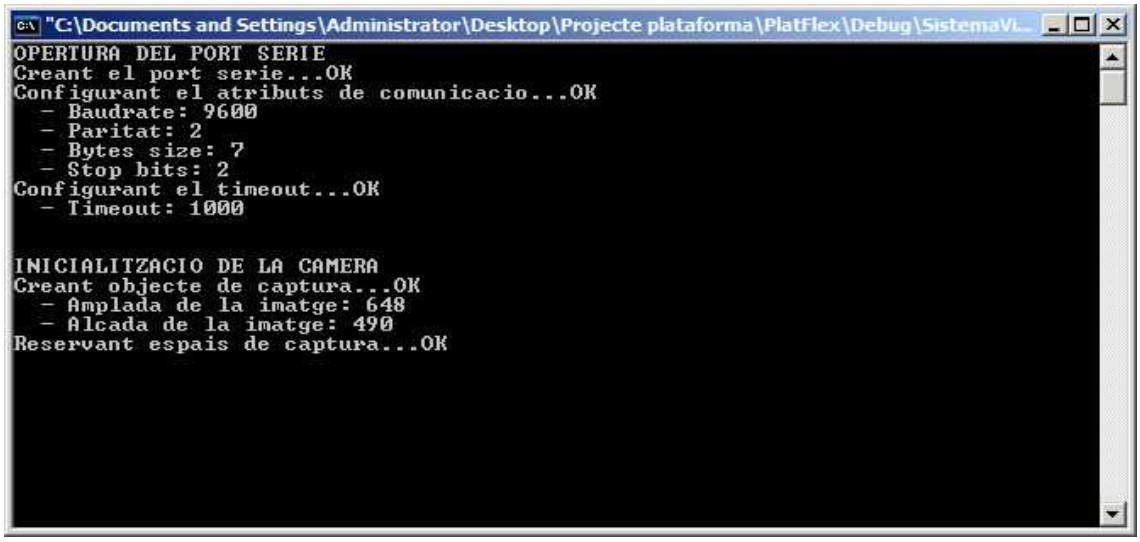


Figura 6.6: Captura de la consola (2)

3. **Activació de la cinta.** S'indica a l'autòmat que inicii el procés de transport de peces activant el bit 10, que es mantindrà actiu mentre s'estigui executant el procés (veure figura 6.7). Amb el canvi a l'estat 1 del grafcet la safata iniciarà el seu moviment cap a la zona de seguretat (veure figura 6.8).

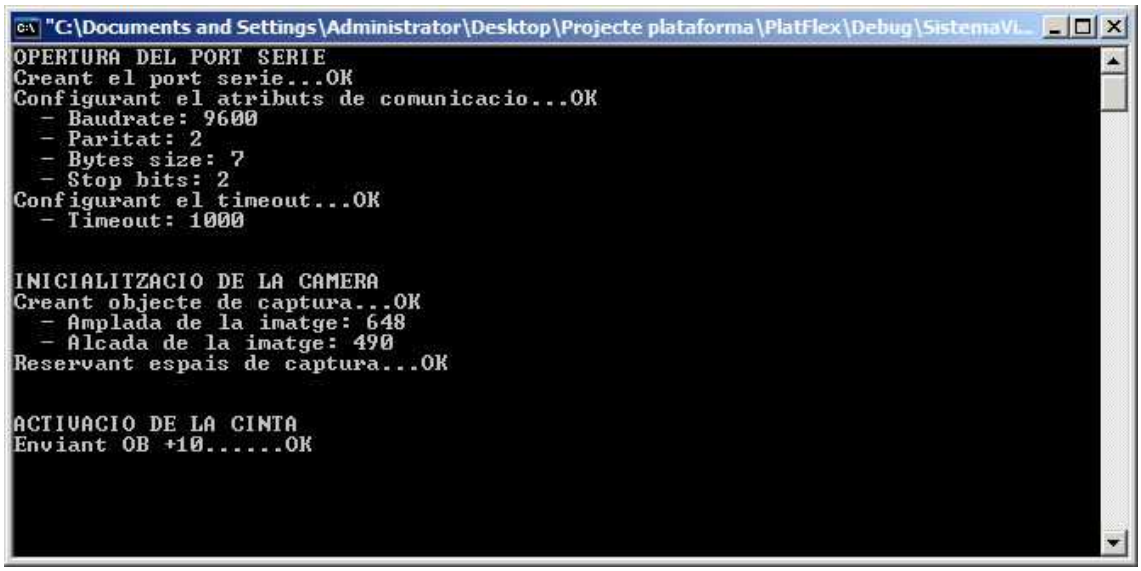


Figura 6.7: Captura de la consola (3)

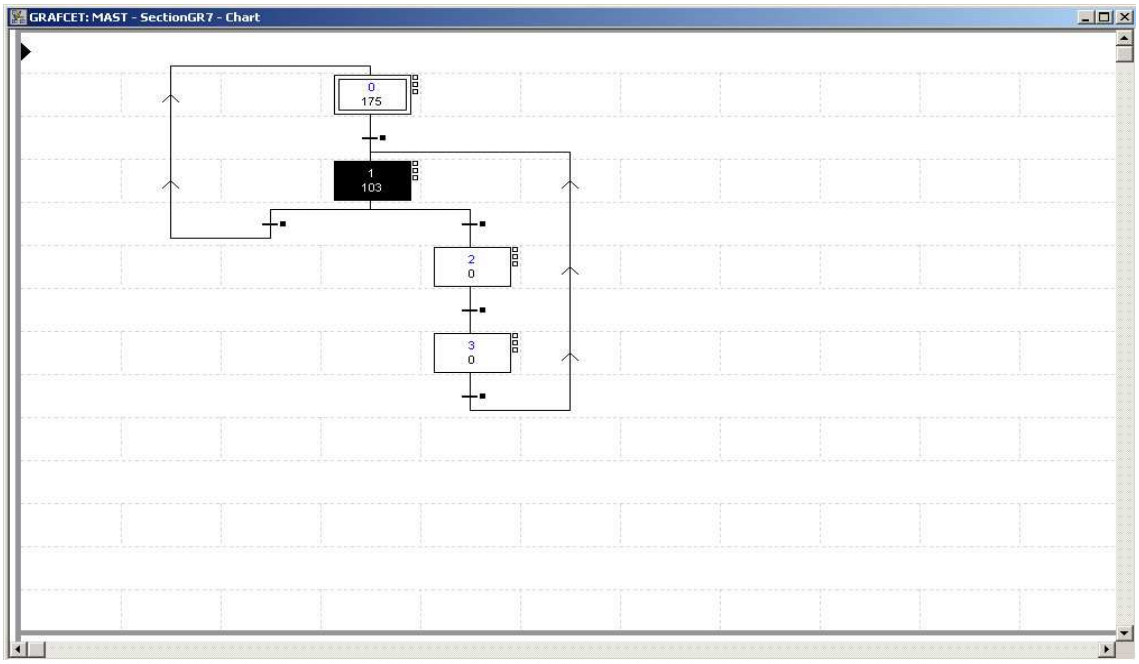


Figura 6.8: Autòmat en l'estat 1

4. **Inicialització del robot.** S'envia la comanda NT (Nest) al robot perquè es desplaci a la seva posició d'origen, s'augmenta la velocitat de desplaçament amb la instrucció SP (Speed) i es col·loca a la posició d'espera (veure figura 6.9).

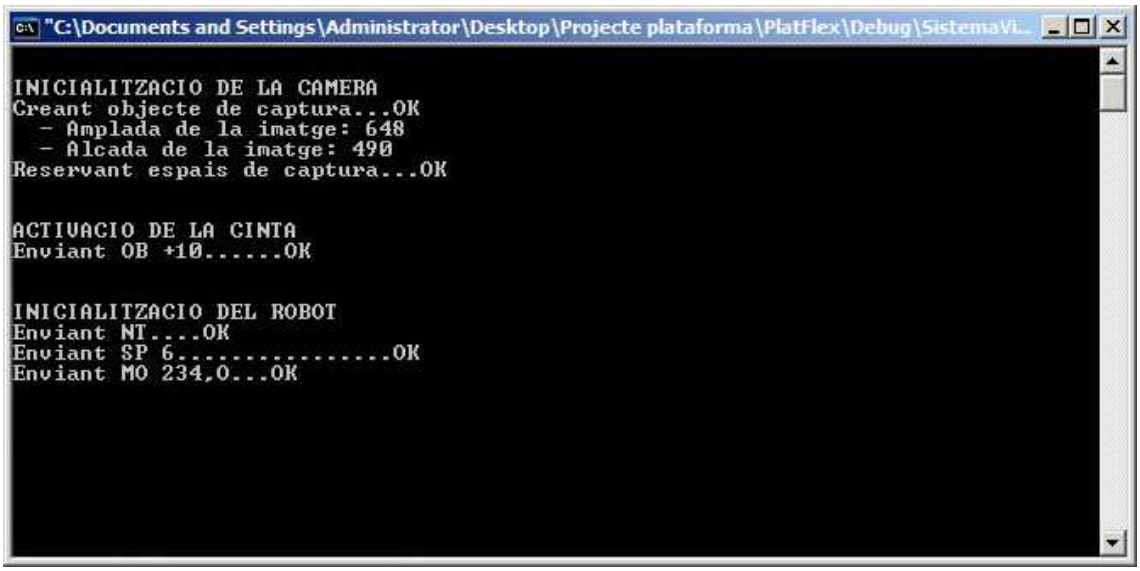


Figura 6.9: Captura de la consola (4)

5. **Esperant la safata.** Cal que la safata es situï a la zona de recollida abans de continuar. Quan hagi arribat a la zona de seguretat l'autòmat canviarà a l'estat 2 i invertirà el gir de la cinta (veure figura 6.10).

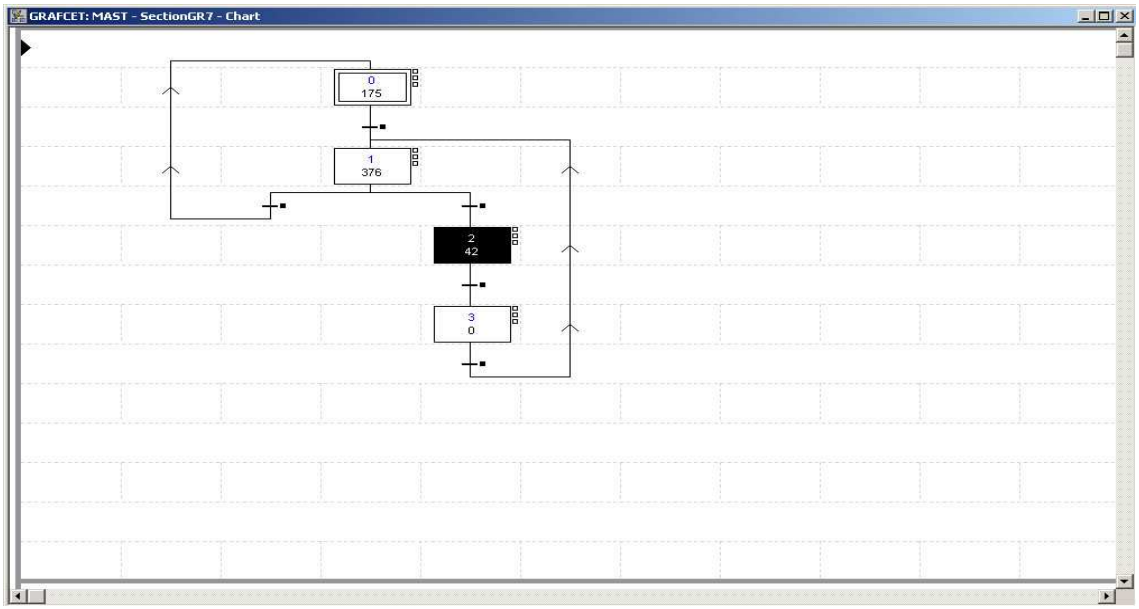


Figura 6.10: Autòmat en l'estat 2

En el moment que el sensor situat a la zona de recollida detecti la safata la cinta s'aturarà i es passarà a l'estat 3 de l'autòmat, el qual activarà el senyal de sortida %Q2.1 (veure figura 6.11).

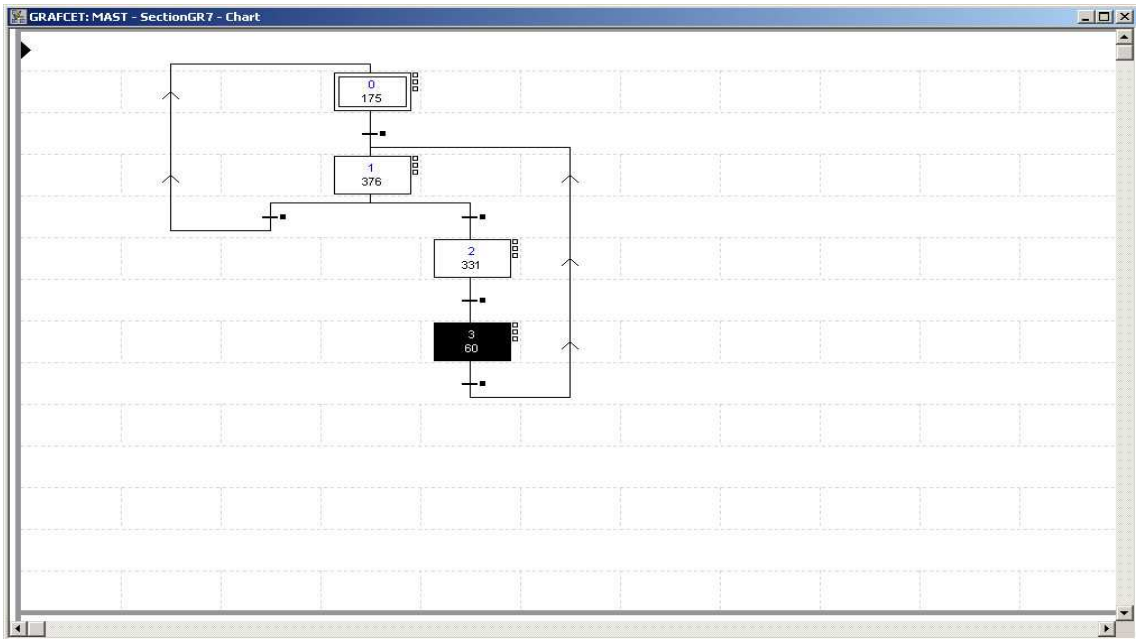


Figura 6.11: Autòmat en l'estat 3

Durant l'estona en que la safata està en moviment es va comprovant la primera senyal per saber si la safata ha arribat a la zona de recollida (veure figura 6.12). Quan està activada continua l'execució del programa.

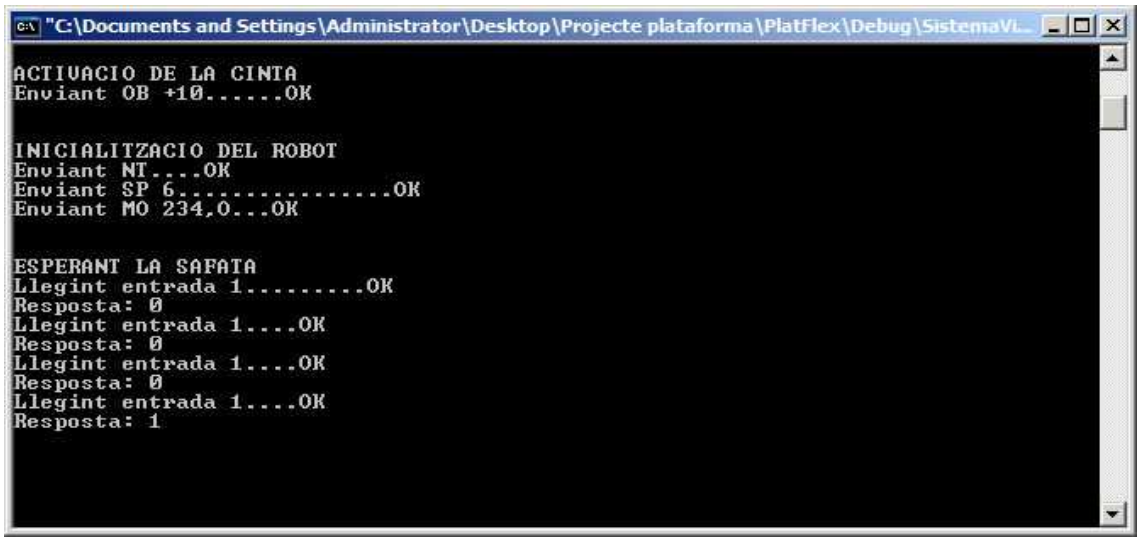


Figura 6.12: Captura de la consola (5)

6. **Calibració de la càmera.** Es situa el tauler d'escacs sobre la safata i es prem la tecla ENTER (veure figura 6.13). Finalitzada l'operació es torna a prémer la tecla (veure figura 6.14).

Figura 6.13: Disposició del patró de calibració sobre la safata

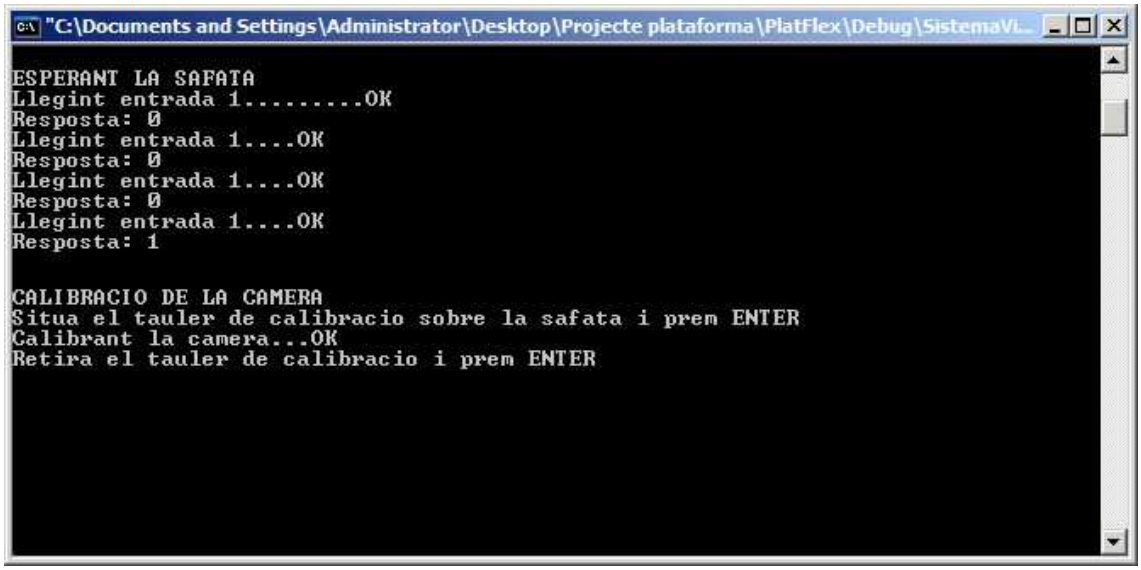


Figura 6.14: Captura de la consola (6)

7. **Col·locació de les peces de delimitació de l'àrea d recollida.** El braç robòtic situa les dues peces sobre la safata (veure figura 6.15). Cada cop que en posa en una retorna el seu vector de posició (veure figura 6.16).

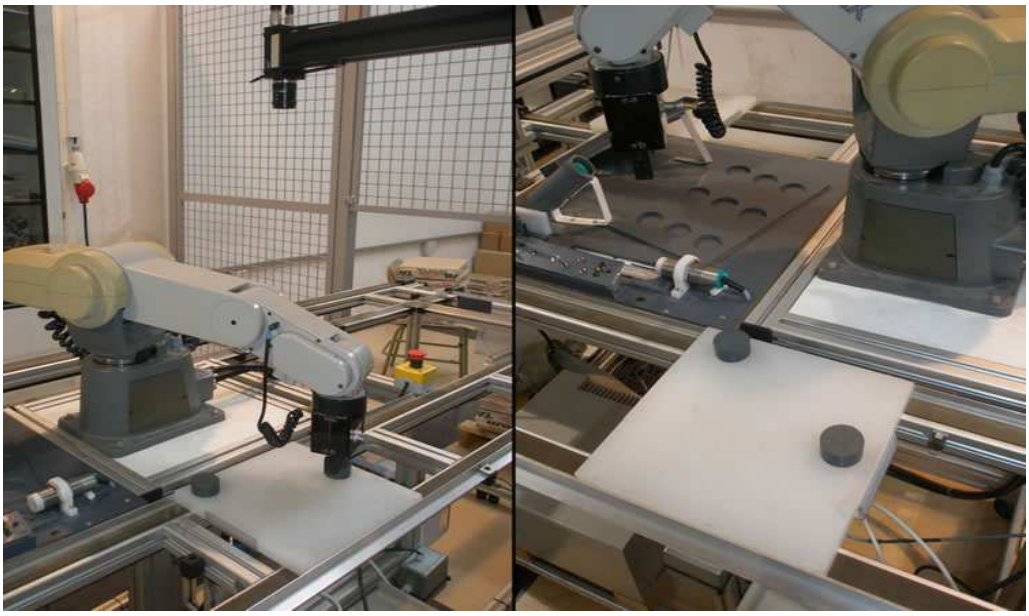


Figura 6.15: Col·locació de les peces de delimitació sobre la safata

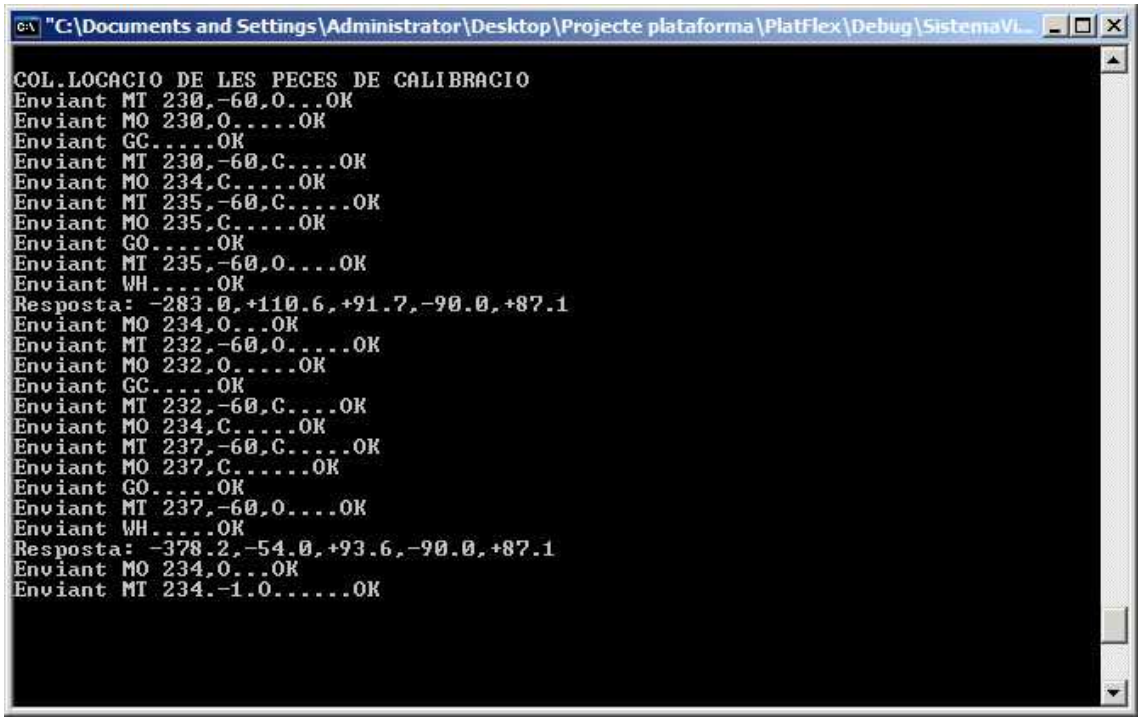


Figura 6.16: Captura de la consola (7)

8. **Delimitació de l'àrea de recollida.** Es captura una imatge i es comprova que hi apareguin els dos blobs de delimitació (veure figura 6.17).

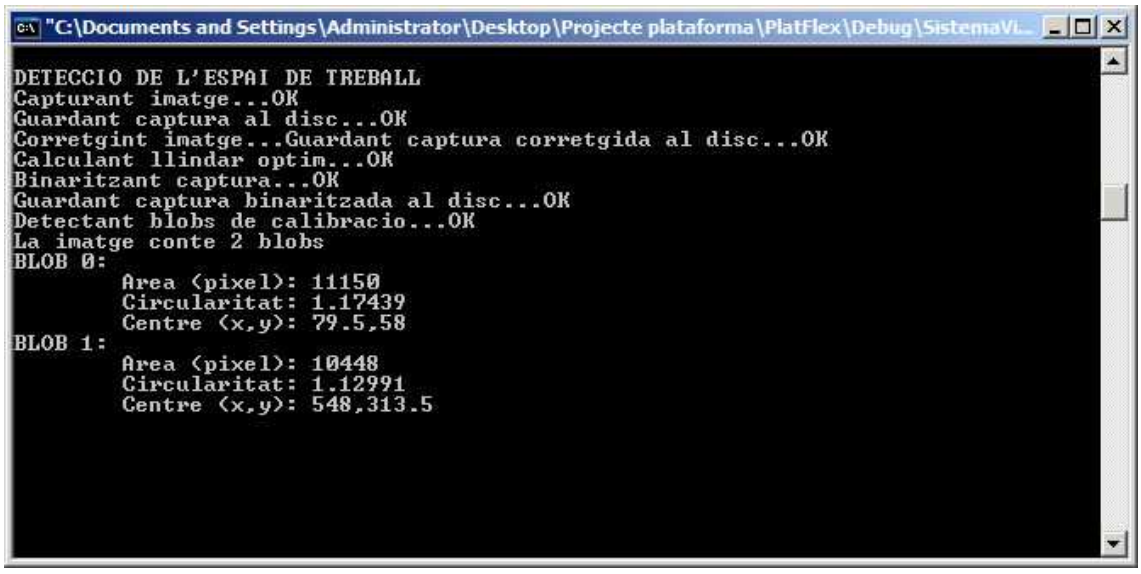


Figura 6.17: Captura de la consola (8)

9. **Retirada de les peces de delimitació.** El robot retira les dues peces i avisa al PLC perquè vagi a buscar una peça a la zona de seguretat (veure figura 6.18).

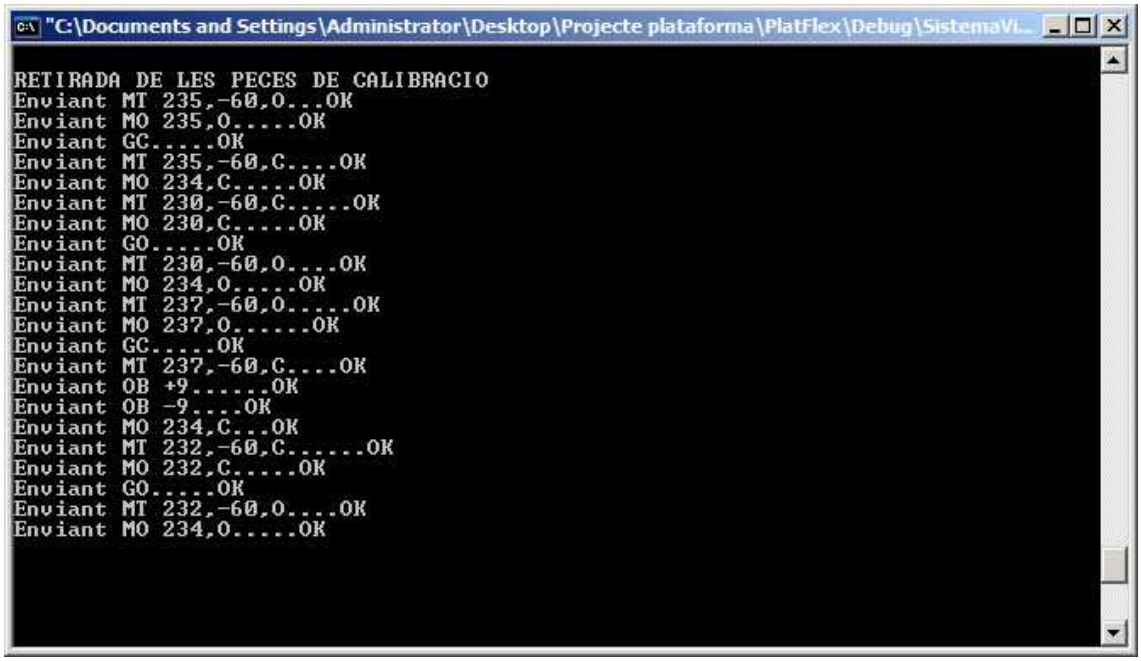


Figura 6.18: Captura de la consola (9)

L'autòmat torna a l'estat 1, iniciant una nova iteració (veure figura 6.19).

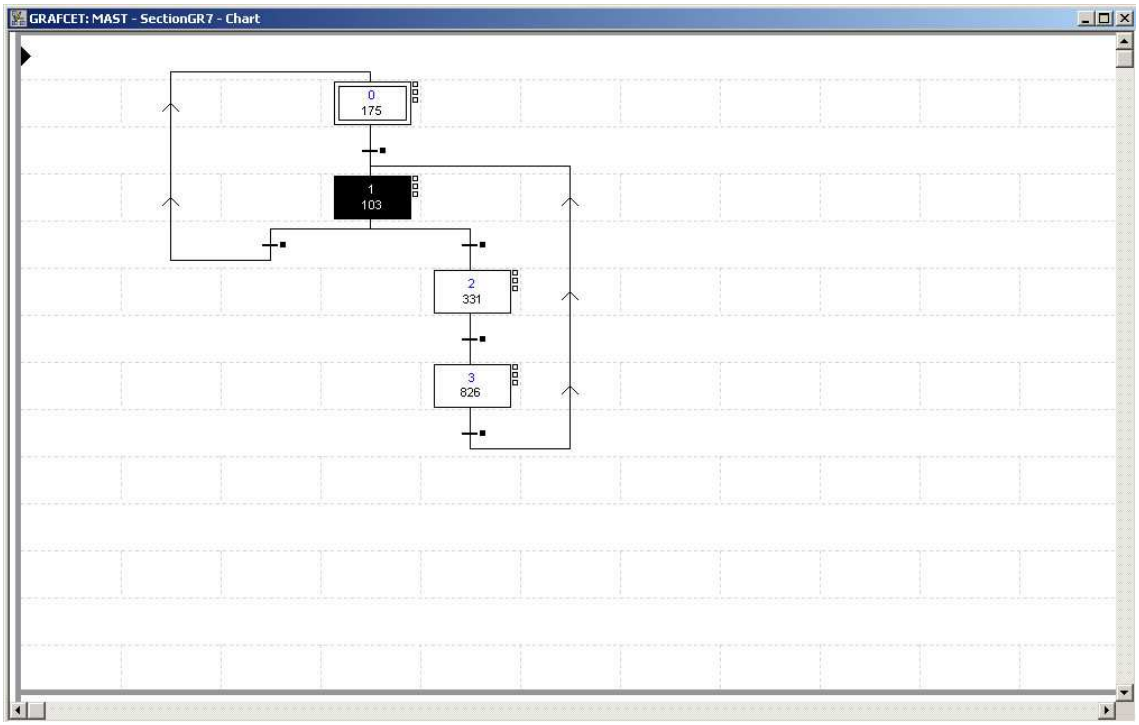


Figura 6.19: Autòmat en l'estat 1 un cop feta la calibració

10. **Recollida de les peces.** Aquest procés es repetirà tantes vegades com peces es vulguin manipular. Quan es detecta que la safata està en la zona de recollida la càmera captura una imatge que es processa en recerca de la peça (veure figura 6.20). Si es detecta un sol blob i aquest està dins de l'àrea de recollida es farà la conversió de coordenades per traslladar el punt dels eixos de la imatge als eixos del robot.

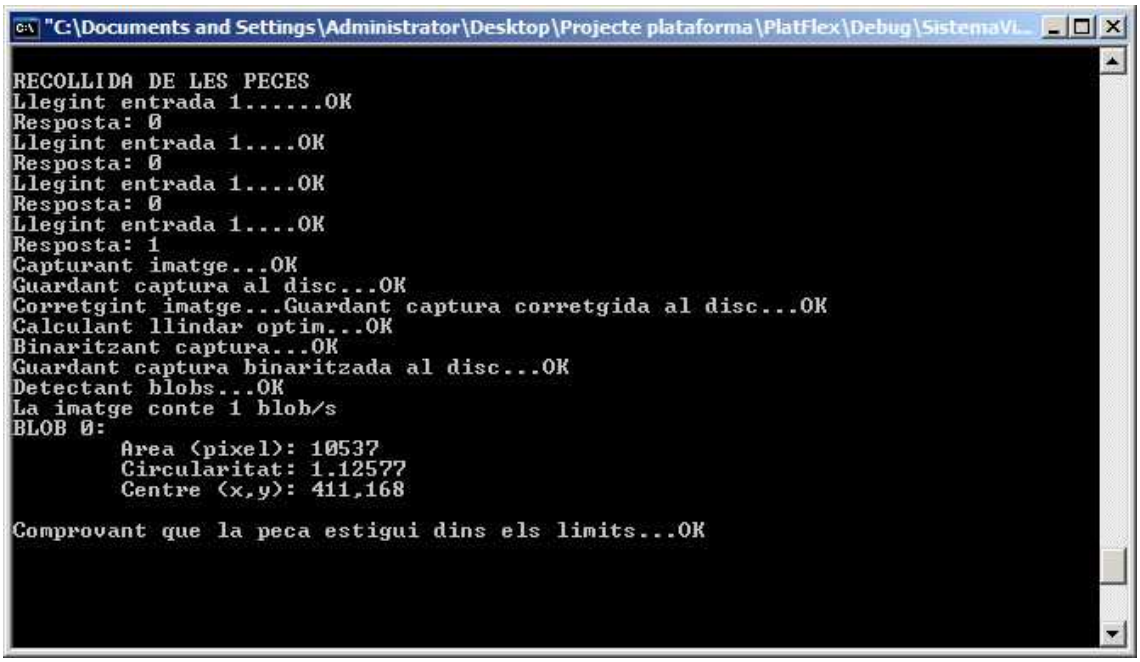


Figura 6.20: Captura de la consola (10)

Un cop creada la instrucció que situa l'element terminal sobre l'objecte s'envia al robot juntament amb la resta d'instruccions de recollida (veure figura 6.21). Alhora se li indica a la cinta que torni cap a la zona de seguretat (veure figura 6.22).

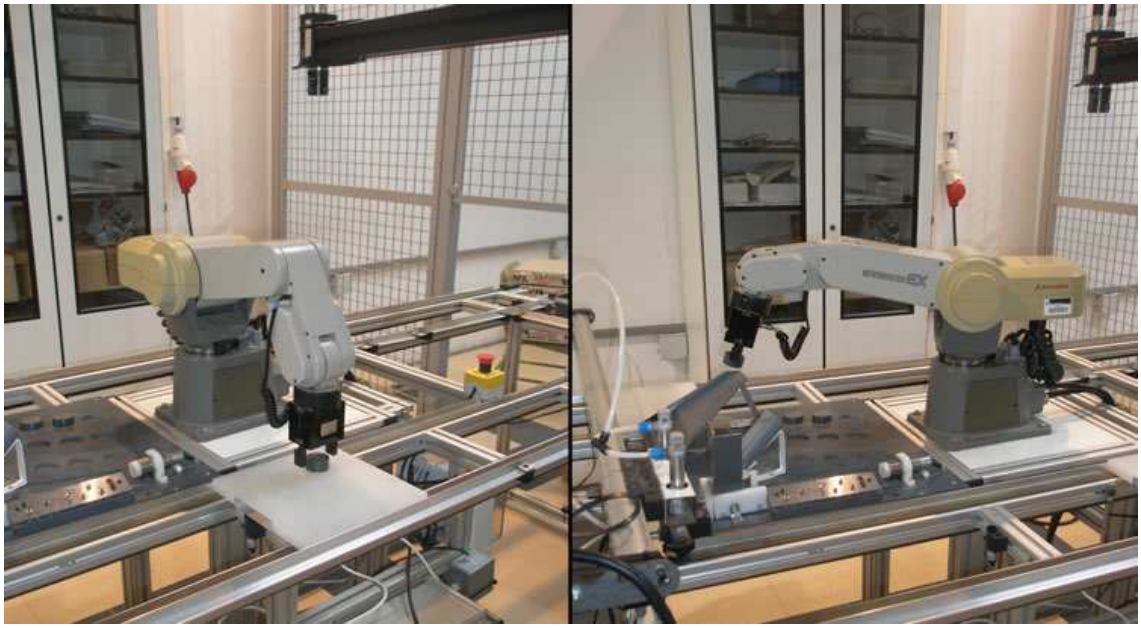


Figura 6.21: Recollida de la peça

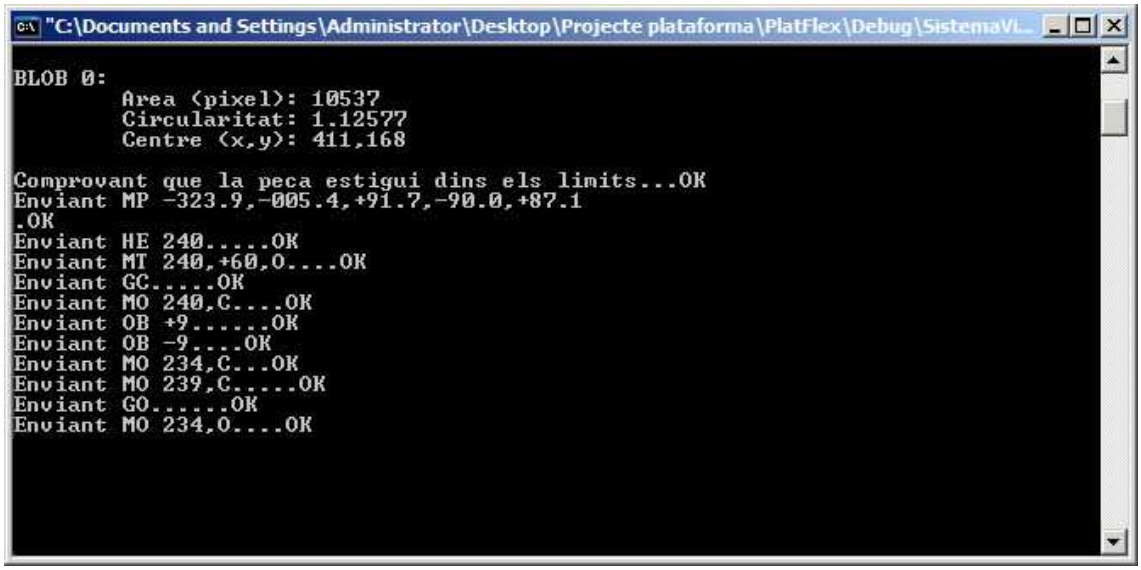


Figura 6.22: Captura de la consola (11)

11. **Desactivació de la cinta.** Si ja s'han processat totes les peces es desactiva el bit de procés actiu (veure figura 6.23). D'aquesta manera l'autòmat retorna a l'estat 0 on roman en espera d'un nou procés de transport.

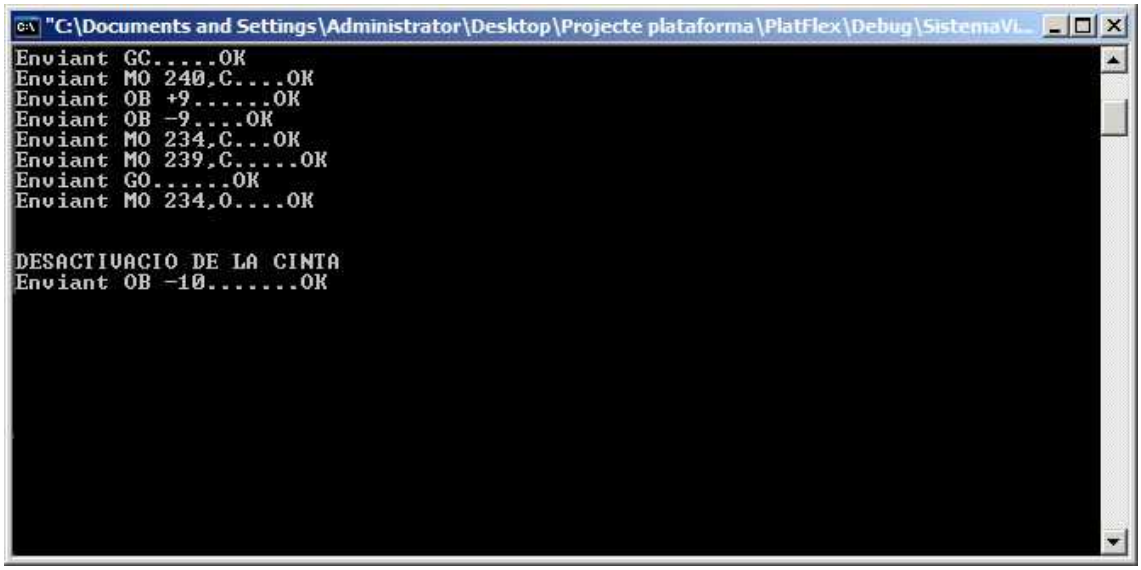


Figura 6.23: Captura de la consola (12)

12. **Tancament del port sèrie.** Després de que s'hagi completat tot el procés es tanca el port de comunicacions (veure figura 6.24).

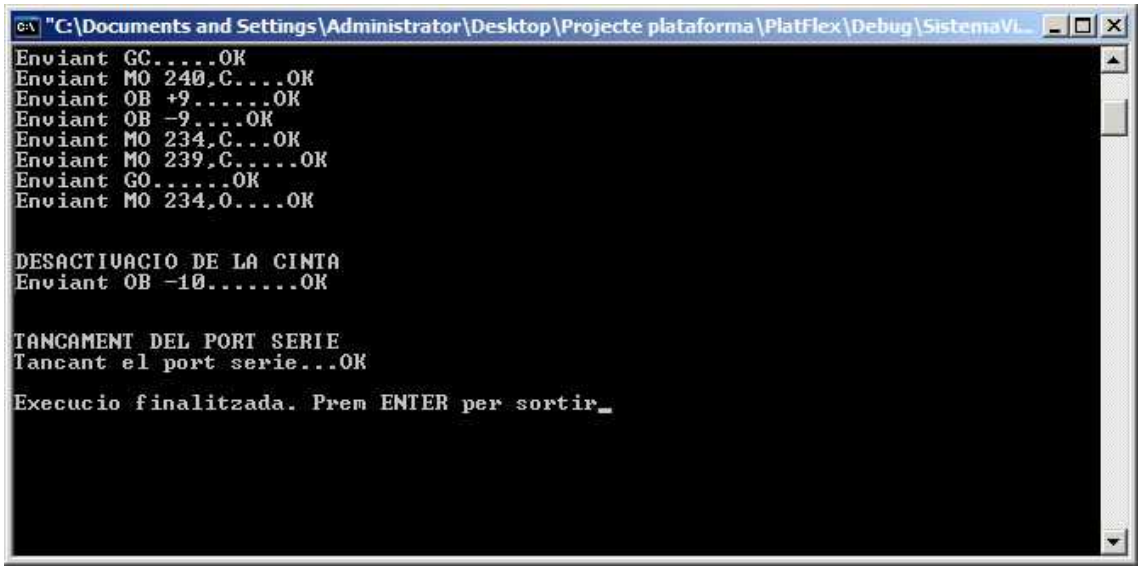


Figura 6.24: Captura de la consola (13)

6.4. ANÀLISI DELS RESULTATS

El procés de recollida de la peça utilitzant el sistema de visió funciona correctament. El sistema és capaç de detectar qualsevol peça que tingui el seu centre dins l'àrea delimitada i d'ordenar al manipulador que la reculli.

Existeix, però, un problema de precisió a l'hora de recollir la peça en certs punts de la safata. Mentre que en la zona central la precisió és acurada, les zones laterals mostren una petita desviació a l'hora de situar l'element terminal sobre l'objecte (veure figura 6.25), provocant que al tancar les pinces la peça es desplaci una mica del seu lloc.

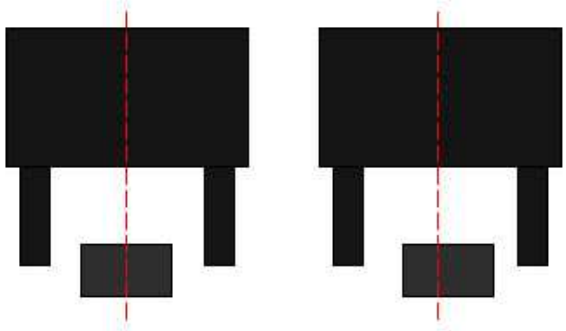


Figura 6.25: Error de precisió

Aquest marge d'error no manté uns valors constants, doncs són diversos motius els que intervenen:

- **La representació de la peça en la imatge.** Aquest és potser el motiu principal. Donada la proximitat de la càmera a la safata les captures no mostren les peces de forma totalment perpendicular. És a dir, mentre que els objectes situats al centre de la imatge sols es veu la seva part superior, als situats cap a les bores se'ls aprecia el seu lateral, provocant que el blob resultant en la binarització no sigui totalment circular (veure figura 6.26).

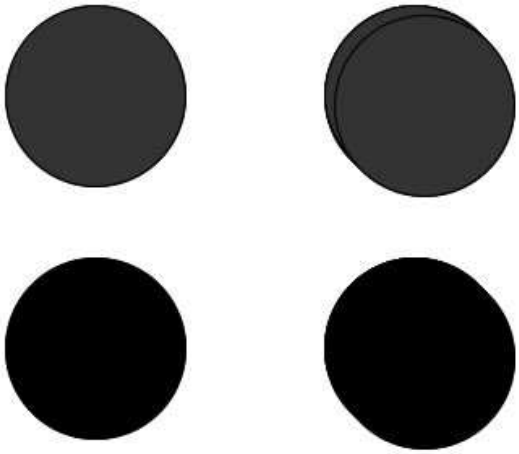


Figura 6.26: Comparació del blob segons la situación de la peça en la safata

La deformació del blob implica que les coordenades obtingudes amb la seva detecció no es corresponguin exactament amb el seu centre. Per tant es perd precisió en la delimitació de l'àrea de recollida (on els dos blobs estan deformats) i en el moment de recollir la peça.

La solució a aquest problema vindria amb l'aplicació d'un sistema d'il·luminació adequat, el qual ressaltés la part superior de la peça per tal de facilitar l'eliminació dels laterals durant la segmentació. Un focus de llum direccional coaxial milloraria els resultats substancialment.

Donat que no disposem de tal sistema, una possible solució més rudimentària seria pintar d'un color més fosc la part superior de les peces per tal de fer-la més visible que la part lateral. Aquest procediment no s'ha pogut posar en pràctica per la falta de peces de recanvi.

- **La desviació dels eixos del robot respecte la safata.** Aquest és un problema que afecta en menor mesura però és igualment important. La posició de la safata no és perfectament paral·lela als eixos del robot, el que provoca certa desviació a mesura que desplacem l'element terminal pels eixos X i Y.

  Com que l'espai de la safata és reduït l'efecte no és gaire considerable, però podem observar que si hi desplacem l'element terminal en línia recta es produeix una petita desviació (veure figura 6.27).

Figura 6.27: Desviació en el moviment sobre la safata

Es podria realitzar una nova calibració dels eixos del robot per corregir el problema, però aquest tornaria tan bon punt es produís una col·lisió amb la plataforma degut a una mala programació del manipulador.

- **La variació de la posició de l'àrea de recollida.** Un altre motiu, ja menys significatiu, és la petita variació de posició que pot patir la safata degut al seu desplaçament. El sensor que detecta la col·locació de la safata en l'espai i la posterior aturada del moviment del motor poden provocar petites diferències en la posició de l'àrea de recollida.

- **La col·locació de les peces de delimitació de l'àrea de treball.** Pot succeir que al dipositar les peces de delimitació sobre la safata aquestes es moguin lleugerament del lloc al deixar-les anar, el que fa que es perdi precisió en el vector de posició del robot. Aquesta petita variació pot afectar en la funció de conversió de coordenades. De totes maneres si es fa una col·locació acurada de les peces això no ha de succeir.

7. CONCLUSIONS I TREBALL FUTUR

Aquest capítol és una conclusió del desenvolupament del projecte i una valoració a possibles treballs futurs que podrien realitzar-se.

7.1. CONCLUSIONS

L'objectiu d'aquest projecte era millorar la funcionalitat de la cel·la de transport flexible. S'ha aconseguit el propòsit de dues maneres:

- Determinant les accions el manipulador Movemaster RV-M1 utilitzant un PC i un llenguatge de programació de propòsit general. Concretament hem aconseguit establir una comunicació via port sèrie entre una aplicació creada en C++ i el RV-M1, obtenint el control del braç robòtic de forma directa i del PLC de forma indirecta.

- Incorporant un sistema de visió artificial per tal d'afegir noves característiques a la cel·la. Aquest nou sistema ha permès implementar un procediment de detecció de la peça que millora la seva recollida. S'ha fet ús de les llibreries OpenCV, les quals tenen grans possibilitats en el camp de l'anàlisi de la imatge que poden ser aplicades en la cel·la.

7.2. TREBALL FUTUR

Després d'assolir satisfactòriament l'objectiu es pot dir que aquest projecte és una porta oberta a nombroses ampliacions associades al procés de transport i classificació que desenvolupa la cel·la de fabricació flexible.

Amb aquest treball ens hem limitat a aplicar un procés de detecció de posició, però la integració del sistema de visió artificial aporta moltes altres possibilitats que podrien ser considerades en treballs futurs. La més evident és la seva aplicació en un procediment d'inspecció del material. Amb les llibreries OpenCV disposem de les eines necessàries per fer processos de classificació segons diversos criteris, com per exemple la forma de l'objecte o la presència de defectes entre d'altres.

Recordem també que ha quedat clara la importància de tenir un control sobre la il·luminació de l'escena. Caldria, doncs, intentar afegir un sistema d'il·luminació adequat que permetés, per una banda, aplicar correctament l'algoritme que calcula el

llindar òptim de binarització i, per l'altra, eliminar la deformació dels blobs que poden reduir la precisió en el procediment de recollida.

Finalment, creades les eines bàsiques per a establir una comunicació amb el Movemaster RV-M1, seria interessant implementar una interfície que facilités les operacions amb el robot així com els canvis en el procés (actualment qualsevol modificació representa una reescriptura del codi).

8. BIBLIOGRAFIA

- Gonzalez, Rafael C.; Woods, Richard E. *Digital Image Processing*. 2ª ed. Prentince Hall, 2002.
- MSDN: Microsoft Developer Network [en línia]. Accessible a http://msdn.microsoft.com/en-us/default.aspx [Consulta: 27 febrer 2008]
- RS232 Tutorial on Data Interface and cables [en línia]. Accessible a http://www.arcelect.com/rs232.htm [Consulta: 26 febrer 2008]
- InformIT: Visual C++ 6 Unleashed > Serial Communications [en línia]. Accessible a http://www.informit.com/library/content.aspx?b=Visual_C_PlusPlus&seqNum=86 [Consulta: 11 març 2008]
- Introduction To RS232 Serial Communication [en línia]. Accessible a http://www.wcscnet.com/Tutorials/SerialComm/Page1.htm [Consulta: 8 març 2008]
- OpenCV: Image Processing and Computer Vision Reference Manual [en línia]. Accessible a http://vision.cis.udel.edu/opencv/ref/OpenCVRef_cv.htm [Consulta: 30 maig 2008]
- cvBlobsLib - OpenCV Wiki [en línia]. Accessible a http://opencv.willowgarage.com/wiki/cvBlobsLib [Consulta: 12 juny 2008]
- Manual d'instruccions del Mitsubishi Movemaster EX RV-M1
- Fitxers d'ajuda de les llibreries IFC

9. ANNEX

Les properes pàgines contenen informació relacionada amb la programació del projecte.

9.1. LLIBRERIA DE CONTROL

La llibreria que s'ha desenvolupat per a la creació d'aquest projecte consta de 3 classes:

- Robot
- SistemaVisio
- Coordenades

A la figura 9.1 s'hi pot observar el diagrama de classes.

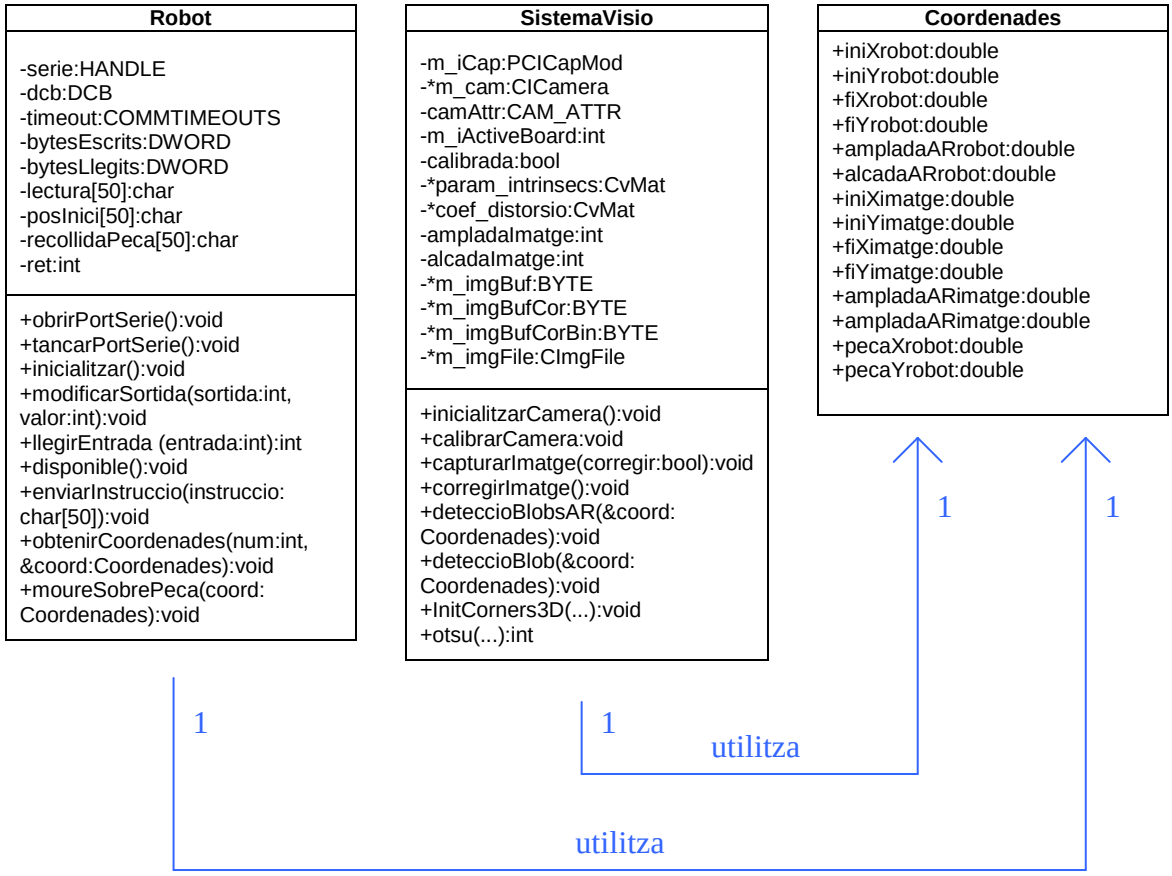


Figura 9.1: Diagrama de classes

9.1.1. Classe Coordenades

Classe contenidor, sense mètodes. S'hi emmagatzema la informació de les coordenades (tant en el pla de la imatge com en l'espai real del robot) que generen les classes Robot i SistemaVisió.

9.1.2. Classe Robot

Aquesta classe s'utilitza per establir la comunicació amb el robot. Permet l'enviament de comandes i la lectura d'informació interna de l'autòmat. Inclou els següents mètodes:

- **obrirPortSerie( ) -** Crea el port de comunicacions.
- **tancarPortSerie( ) -** Elimina el port de comunicacions.
- **enviarInstruccio(char instruccio[50]) -** Envia al robot la cadena de caràcters que forma la comanda a executar.
- **inicialitzar( ) -** Prepara el robot abans de treballar amb ell: envia la instrucció d'inicialització, modifica la velocitat de desplaçament i el situa a la posició d'espera.
- **modificarSortida(int sortida, int valor) -** Modifica un dels bits de sortida per a la comunicació amb l'autòmat.
- **llegirEntrada(int entrada) -** Llegeix els dos bits d'entrada de comunicació amb l'autòmat.
- **obtenirCoordenades(int num, Coordenades &coord).** Sol·licita al manipulador el seu vector de posició actual, que conté les coordenades en que es troba el seu element terminal.
- **moureSobrePeca(Coordenades coord).** Prepara la instrucció de posicionament de l'element terminal sobre la peça a recollir i l'envia.
- **disponible( ).** Comprova que el robot estigui llest per rebre noves instruccions, evitant que aquestes es puguin perdre.

9.1.3. Classe SistemaVisió

Aquesta classe inclou els mètodes per a la utilització del sistema de visió, tant la captura d'imatges com el seu tractament i posterior obtenció d'informació.

- **inicialitzarCamera( ) -** Prepara la càmera per ser utilitzada i reserva els espais de memòria on s'emmagatzemaran les captures.
- **calibrarCamera( ) -** Realitza el procediment de calibració de la càmera a partir d'un patró amb forma de tauler d'escacs, trobant els paràmetres intrínsecs i els coeficients de distorsió de la càmera.
- **capturarImatge(bool corregir) -** Emmagatzema una imatge al buffer i crida al mètode de correcció si es demana per paràmetre.
- **corregirImatge( ) -** Corregeix la distorsió que origina la lent a partir dels paràmetres obtinguts amb la calibració.
- **deteccioBlobsAR(Coordenades &coord) -** Busca a la imatge els dos blobs de delimitació de l'àrea de recollida, guarda la informació de les seves coordenades i fa el càlcul de la mida de l'àrea.
- **deteccioBlob(Coordenades &coord) -** Busca a la imatge el blob de la peça a recollir, comprovant que es trobi dins els límits de l'àrea de recollida.
- **InitCorners3D(CvMat *Corners3D, CvSize ChessBoardSize, int NImages, float SquareSize) -** Prepara una matriu de punts del tauler d'escacs per fer-la servir en els càlculs del mètode de calibració.
- **otsu (unsigned char *image, int rows, int cols, int x0, int y0, int dx, int dy, int vvv) -** Calcula el llindar de binarització òptim d'una imatge per mitjà de l'algoritme Otsu.

9.2. PROGRAMACIÓ DE L'APLICACIÓ

Aquest apartat conté el codi i la corresponent explicació de les funcions bàsiques que s'han utilitzat en l'aplicació principal: utilització del port sèrie, utilització de la càmera i tractament de la imatge.

9.2.1. Utilització del port sèrie

La programació del port sèrie es realitza mitjançant l'API Win32 de Microsoft. Tot seguit es mostra com obrir un port de comunicacions COM1 i com utilitzar-lo.

9.2.1.1. Creació del port de comunicació

La creació del port es fa amb la funció *CreateFile*, el qual retorna un objecte de tipus HANDLE definit per la configuració que indiquem en els paràmetres. És aquest objecte retornat el que ens permet accedir al dispositiu d'entrada/sortida:

```
HANDLE serie;
serie = CreateFile("\\\\.\\COM1",        //nom de l'objecte, en aquest cas el port sèrie
       GENERIC_READ|GENERIC_WRITE,       //tipus d'accés lectura i escriptura
       0,                                //l'objecte no es pot compartir ni obrir de nou
                                         //sense tancar-lo abans
       NULL,                             //sense atributs de seguretat
       OPEN_EXISTING,                    //obrir objecte existent, si no existeix retorna
                                         //error
       FILE_ATTRIBUTE_NORMAL,            //l'objecte no té altres atributs
       NULL);                            //sense punter a una plantilla d'atributs
```

Un cop creat el port cal configurar els paràmetres de comunicació (baud rate, bits de paritat, etc.). Aquests s'emmagatzemen en una estructura DCB associada al port, que haurà de ser recuperada amb la funció *GetCommState*. Per aplicar els canvis que s'hagin realitzat s'ha d'assignar el nou DCB al port utilitzant la funció *SetCommState*:

```
GetCommState (serie,&dcb);

dcb.BaudRate = BAUDRATE;     //baud rate de 9600 bps
dcb.Parity = PARITAT;        //bit de paritat senar
dcb.ByteSize = BYTESIZE;     //longitut dels caràcters de 7 bits
dcb.StopBits = STOPBITS;     //2 bits de stop

SetCommState(serie,&dcb)
```

La configuració del timeout és similar a la dels paràmetres de comunicació. Es recupera la informació amb la funció *GetCommTimeouts* i, un cop modificada, s'actualitza amb la funció *SetCommTimeouts*:

```
GetCommTimeouts(serie, &timeout);
timeout.ReadTotalTimeoutConstant = 1000; //1 segon
SetCommTimeouts(serie,&timeout);
```

Amb això el port està llest per ser utilitzat. Cal recordar que al finalitzar l'aplicació cal tancar-lo amb la funció *CloseHandle*:

```
CloseHandle(serie);
```

9.2.1.2. Enviament i lectura d'informació

Un cop s'ha creat el port ja es pot enviar i rebre informació a través d'ell. L'escriptura es fa amb la funció *WriteFile*, passant per paràmetre la cadena de caràcters a enviar:

```
char instruccio[50];
DWORD bytesEscrits;

WriteFile(serie,        //port
      instruccio,       //cadena de caràcters a enviar
      50,               //nombre de bytes a enviar
      &bytesEscrits,    //variable amb el total de bytes escrits
      NULL);            //sense estructura OVERLAPPED
```

I la lectura amb un *ReadFile*, quedant la informació rebuda en una altra cadena de caràcters:

```
char lectura[50];
DWORD bytesLlegits;

ReadFile(serie,         //port
      lectura,          //cadena de caràcters on s'emmagatzemarà la resposta
      50,               //nombre de bytes a llegir
      &bytesLlegits,    //variable amb el total de bytes llegits
      NULL);            //sense estructura OVERLAPPED
```

9.2.2. Utilització de la càmera

En aquest apartat s'explica com utilitzar la càmera per poder adquirir imatges i emmagatzemar-les pel seu posterior processament.

9.2.2.1. Preparació de l'objecte de captures

La utilització de la càmera es fa a partir d'un objecte de captura, el qual s'ha de crear i assignar-li el port en el que està connectada la càmera:

```
PCICapMod m_iCap;        //objecte de captura d'imatge
CICamera *m_cam;         //objecte càmera
int m_iActiveBoard = 0; //nº de placa activa

//Creació d'un objecte de captura d'imatges
if (!(m_iCap = IfxCreateCaptureModule("NSP", m_iActiveBoard,
"nsptest.txt"))) {
      m_iCap = IfxCreateCaptureModule("NSP", 0);
      m_iCap->ProcessCameraFilesInDir("camdb", TRUE);
}

//Creació de la càmera situada al port 0
m_cam = m_iCap->GetCam(0);
```

És important obtenir els atributs de la càmera (alçada, amplada i bytes per píxel de la imatge) per a posteriors operacions i càlculs:

```
CAM_ATTR camAttr;        //atributs de la camera
DOWRD ampladaImatge;     //píxels d'amplada de la imatge
DWORD alcadaImatge;      //píxels d'alçada de la imatge
DWORD bytesPerPixel;     //nombre de bytes d'informació per cada píxel

m_cam->GetAttr(&camAttr);
ampladaImatge = camAttr.dwWidth;
alcadaImatge = camAttr.dwHeight;
bytesPerPixel = camAttr.dwBytesPerPixel;
```

9.2.2.2. Adquisició d'imatges

La informació de cada captura s'emmagatzema en un buffer de tipus BYTE, la mida del qual ve definit pel nombre de píxels de la imatge multiplicat pel nombre de bytes per píxel:

```
BYTE *m_imgBuf;    //espai de memoria destinat a la captura

m_imgBuf = (BYTE*)GlobalAllocPtr(GMEM_FIXED, ampladaImatge *
           alcadaImatge * bytesPerPixel);
```

La instrucció *Snap* realitza la captura:

```
m_cam->Snap(m_imgBuf);
```

Existeix la possibilitat de guardar les imatges adquirides al disc (en format BMP o TIFF):

```
m_imgFile = IfxCreateImgFile("C:\\captura.bmp");
if (m_imgFile){
      m_imgFile->WriteFile(m_imgBuf, ampladaImatge, alcadaImatge,
      bytesPerPixel);
}
```

La manipulació de les imatges es realitza amb la llibreria OpenCV. Aquesta llibreria però treballa amb estructures IplImage, pel que caldrà traspassar el contingut del buffer de la imatge a una estructura d'aquest tipus

```
IplImage *imatgeIpl;    //imatge IPL

imatgeIpl = cvCreateImage(cvSize(ampladaImatge,alcadaImatge),
IPL_DEPTH_8U, 1);
for(i=0; i < img_tauler->imageSize; i++){
      img_tauler->imageData[i] = m_imgBuf[i];
}
```

9.2.2.3. Calibració

La calibració de la càmera es porta a terme amb les llibreries OpenCV i amb l'ús d'un tauler d'escacs. Primerament es defineix la mida del tauler i el nombre de cantonades total i, a continuació, es crida la funció *cvFindChessboardCorners*. Si l'algoritme no localitza totes les cantonades la calibració no pot continuar:

```
IplImage *img_tauler;            //imatge del tauler d'escacs
IplImage *img_tauler_cantonades;//imatge del tauler amb les cantonades senyalades
CvSize tamany_tauler;            //tamany del tauler segons nombre de cantonades
CvPoint2D32f* cantonades;        //punts on es troben les cantonades
int cantonades_totals;
int cantonades_trobades;
int resultat;

tamany_tauler = cvSize(CANTONADES_TAULER_X,CANTONADES_TAULER_Y);
cantonades_totals = CANTONADES_TAULER_X*CANTONADES_TAULER_Y;
cantonades = (CvPoint2D32f*)cvAlloc(cantonades_totals *
             sizeof(CvPoint2D32f));
cantonades_trobades = 0;

resultat = cvFindChessboardCorners(img_tauler, tamany_tauler,
           cantonades, &cantonades_trobades,
           CV_CALIB_CB_ADAPTIVE_THRESH);
```

Trobades les cantonades del tauler d'escacs es pot millorar la precisió de la localització amb la instrucció *cvFindCornerSubPix*:

```
cvFindCornerSubPix(img_tauler, cantonades, cantonades_trobades,
                   tamany_tauler, cvSize(-1,-1),
                   cvTermCriteria (CV_TERMCRIT_ITER | CV_TERMCRIT_EPS,
                   20, 0.01));
```

Fins i tot podem dibuixar les cantonades identificades en la imatge amb la funció *cvDrawChessboardCorners*:

```
cvDrawChessboardCorners(img_tauler_cantonades, tamany_tauler,
                        cantonades, cantonades_trobades, resultat);
```

El següent pas és crear i inicialitzar les matrius de punts objecte (realitat) i punts imatge (captura) amb la informació adquirida. És a partir d'aquestes matrius que la funció de calibració obté els paràmetres intrínsecs i els coeficients de distorsió:

```
CvMat *num_punts;
CvMat *punts_objecte;
CvMat *punts_imatge;

num_punts = cvCreateMat(1,1,CV_32SC1);
punts_objecte = cvCreateMat(cantonades_totals,3,CV_32FC1);
punts_imatge = cvCreateMat(cantonades_totals,2,CV_32FC1);
cvSetData(punts_imatge, cantonades, sizeof(CvPoint2D32f));
cvSetData(num_punts, &cantonades_trobades, sizeof(int));

// omplir la matriu de punts objecte
InitCorners3D(punts_objecte, tamany_tauler, 1, TAMANY_QUADRAT);
```

Sols queda cridar la funció *cvCalibrateCamera2*, a la qual se li passen les matrius on deixarà els resultats (paràmetres intrínsecs i coeficients distorsió):

```
cvCalibrateCamera2(punts_objecte, punts_imatge, num_punts,
cvSize(ampladaImatge,alcadaImatge), param_intrinsecs, coef_distorsio,
NULL, NULL, 0);
```

9.2.3. Tractament de la imatge

Tot seguit es s'expliquen els procediments de tractament de la imatge que s'han aplicat en el desenvolupament del projecte.

9.2.3.1. Correcció

La correcció de les imatges un cop calibrada la càmera es realitza amb la funció cv*Undistort2*, a la qual hi passarem la imatge a corregir, l'objecte IplImage on deixarem el resultat i les dues matrius de calibració:

```
IplImage *img_original;
IplImage *img_corretgida;

img_corretgida = cvCloneImage(img_original);
cvUndistort2(img_original, img_corretgida, param_intrinsecs,
coef_distorsio);
```

9.2.3.2. Binarització

Per binaritzar una imatge modificarem directament el valor de cada píxel emmagatzemat al buffer segons el valor de llindar escollit:

```
cout << "Binaritzant captura...";
for(i=0; i<ampladaImatge * alcadaImatge; i++){
      if((int)m_imgBufCor[i] < llindar)
            m_imgBufCorBin[i]=0;
      else
            m_imgBufCorBin[i]=255;
}
```

9.2.3.3. Detecció de blobs

La detecció de blobs es fa amb la llibreria especialitzada cvBlobsLib, basada en OpenCV. Aquesta llibreria parteix d'una imatge IPL i, mitjançant la funció *CBlobsResult*, s'obté informació de tots els blobs que s'hi localitzin. Aquesta informació es guarda en un objecte de tipus CblobsResult:

```
IplImage *img_blobs;    //Punter a la imatge IPL que conté els blobs
CBlobResult blobs;      //Blobs identificats a la imatge}
Int numblobs

//localitzar blobs
blobs = CBlobResult(img_blobs, NULL, 50, true);
```

Per saber el nombre de blobs que conté l'objecte CBlobResult executarem la funció *GetNumBlobs:*

```
int num_blobs = blobs.GetNumBlobs();
```

Per obtenir uns resultats més acurats podem filtrar els resultats. A continuació es mostra com filtrar per tamany de l'àrea del blob i per la seva circularitat:

```
//Filtratge d'area: 10000 < area del blob < 14000
blobs.Filter(blobs, B_INCLUDE, CBlobGetArea(), B_GREATER, 10000 );
blobs.Filter(blobs, B_INCLUDE, CBlobGetArea(), B_LESS, 14000 );

//Filtratge de circularitat: 10000 < circularitat del blob < 14000
blobs.Filter(blobs, B_INCLUDE, CBlobGetCompactness(), B_GREATER, 1);
blobs.Filter(blobs, B_INCLUDE, CBlobGetCompactness(), B_LESS, 1.3 );
```

Finalment veiem com obtenir la informació dels blobs que s'ha generat amb la funció *GetSTLResult*:

```
double_stl_vector area = blobs.GetSTLResult(CBlobGetArea());
double_stl_vector circ = blobs.GetSTLResult(CBlobGetCompactness());
double_stl_vector cenX = blobs.GetSTLResult(CBlobGetXCenter());
double_stl_vector cenY = blobs.GetSTLResult(CBlobGetYCenter());

for(i=0; i<num_blobs; i++){
      cout << "BLOB " << i << ":" << endl;
      cout << "  Area (pixel): " << area[i] << endl;
      cout << "  Circularitat: " << circ[i] << endl;
      cout << "  Centre (x,y): "<< cenX[i] << "," << cenY[i] << endl;
}
```

9.3. INCLUSIÓ DE LES LLIBRERIES OPENCV I CVBLOBSLIB

En aquest apartat s'explica com incloure les llibreries OpenCV i cvBlobsLib a un projecte de Visual C++ 6.0 per tal de poder utilitzar la API desenvolupada.

9.3.1. Llibreries OpenCV

- Descarregar la llibreria (*http://sourceforge.net/projects/opencvlibrary)* i instal·lar-la. El directori per defecte és *C:/Archivos de programa/OpenCV*.

- Afegir els directoris següents al projecte des de la finestra **Tools** > **Options** > **Directories**

  - Library files:

    - "C:/Archivos de programa/OpenCV/lib"
  - Include files:
    - "C:/Archivos de programa/OpenCV/cv/include"
    - "C:/Archivos de programa/OpenCV/cxcore/include"
    - "C:/Archivos de programa/OpenCV/cvaux/include"
    - "C:/Archivos de programa/OpenCV/otherlibs/highgui"
    - "C:/Archivos de programa/OpenCV/otherlibs/cvcam/include"
  - Source files:
    - "C:/Archivos de programa/OpenCV/cv/src"
    - "C:/Archivos de programa/OpenCV/cxcore/src"
    - "C:/Archivos de programa/OpenCV/cvaux/src"
    - "C:/Archivos de programa/OpenCV/otherlibs/highgui"
    - "C:/Archivos de programa/OpenCV/otherlibs/cvcam/src/windows"

- Copiar els següents DLLs del directori *C:\Archivos de programa\OpenCV\lib* al directori destí de l'executable:
  - cv100.dll
  - cvaux100.dll
  - cvcam100.dll
  - cxcore100.dll
  - cxts001.dll
  - highgui100.dll
  - libguide40.dll
  - ml100.dll

9.3.2. Llibreria cvBlobsLib

- Descarregar la llibreria (*http://opencv.willowgarage.com/wiki/cvBlobsLib#Download*).
- Obrir el projecte que ve a l'arxiu descarregat, incloure-hi els directoris de OpenCV (tal i com s'ha vist a l'apartat 9.3.1) i compilar-lo.
- Copiar la llibreria resultant (cvblobslib.lib) al directori desitjat.
- Afegir al nostre projecte els següents directoris:
  - Library files:
    - La ruta del directori on es troba la llibreria compilada cvbloblib.lib

- Include files:
    - La ruta del directori cvBlobLib

- Anar a la finestra **Project > Settings > C++ > Preprocessor** i escriure la ruta del directori que conté el fitxer cvblobslib.lib a "Additional include directories".
- Anar a la finestra **Project > Settings > Link > Input**, escriure la ruta del directori que conté el fitxer cvblobslib.lib a "Additional library path" i afegir cvblobslib.lib a "Object/library modules".
- Verificar que, a **Project > Settings > C/C++ > Code generation**, la opció de "Use run-time library" està en Debug Multithreaded DLL o Multithreaded DLL.
- Verificar que, a **Project > Settings > General**, està seleccionat Use MFC in a Shared DLL.

Si apareixen problemes de linkatge, crear una carpeta al FileView del nostre projecte que contingui tots els arxius de *C:/Archivos de programa/OpenCV/lib*.

9.4. PROGRAMACIÓ DEL PLC

En aquest apartat s'exposa la programació del PLC utilitzant el programa PL7 Pro v4.4.

9.4.1. Grafcet

A la figura 9.2 s'hi observa el conjunt d'estats que formen el grafcet de segon nivell de l'aplicació.

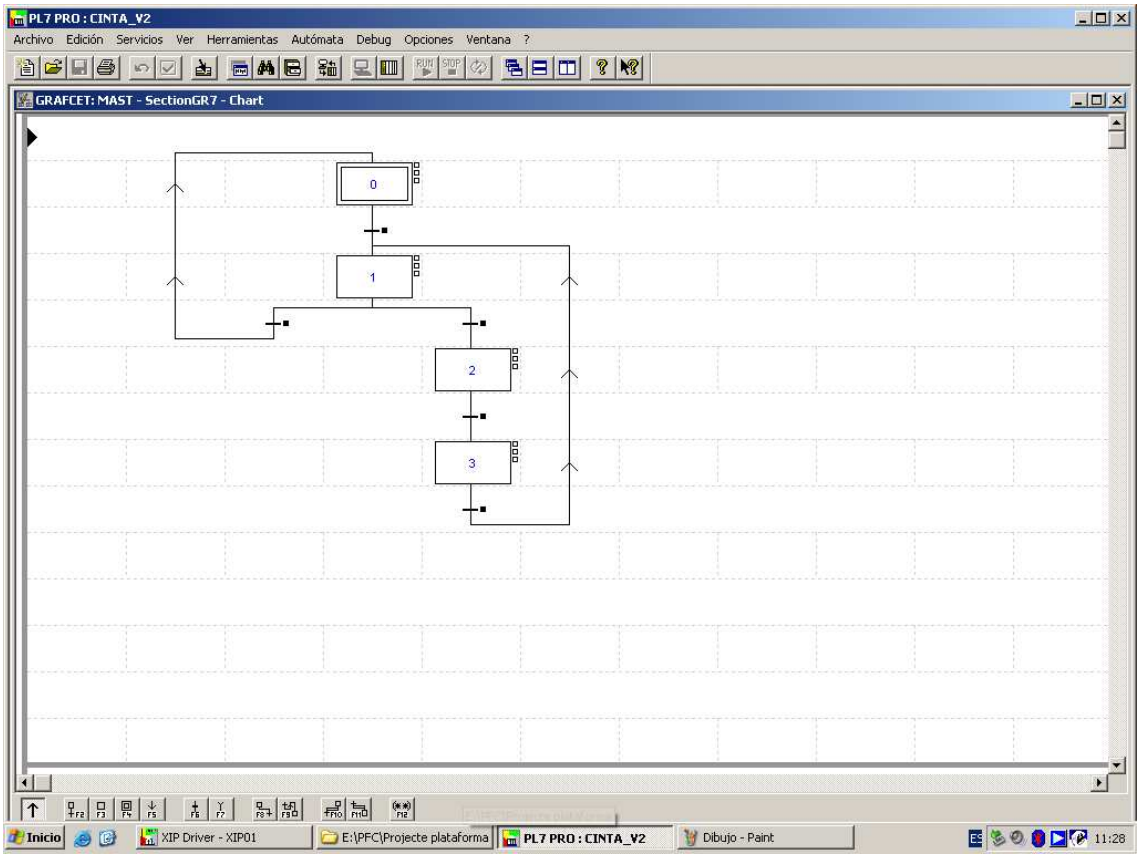


Figura 9.2: Grafcet de segon nivell del procés

9.4.2. Variables

El funcionament del PLC es basa en l'activació i desactivació de variables. Les que s'han utilitzat en la programació del grafcet són:

- %I1.10 – Senyal d'entrada 1 de l'autòmat
- %I1.11 – Senyal d'entrada 2 de l'autòmat
- %I\4.0\2.0 – Sensor de la posició cinta dreta a la zona de seguretat
- %I\4.0\1.1 – Sensor de la posició cinta dreta a la zona de manipulació
- %I\4.0\1.0 – Polsador d'emergència
- %Q\4.0\1.2 – Activació de la cinta dreta cap a la zona de seguretat
- %Q\4.0\1.3 – Activació de l'inversió de gir de la cinta dreta
- %Q\4.0\10.0 – Balisa lluminosa de color vermell

- %Q2.1 – Senyal de sortida 1 de l'autòmat
- %X1 – Estat 1
- %X2 – Estat 2
- %S6 – Intermitència d'1s per la balisa lluminosa
- Les senyals del tipus %Mx corresponen a variables internes

9.4.3. Transicions d'estats

A les figures que es mostren a continuació (9.3 a 9.7) s'hi observen les transicions d'estats, és a dir, les condicions que marquen el pas d'un estat al següent.

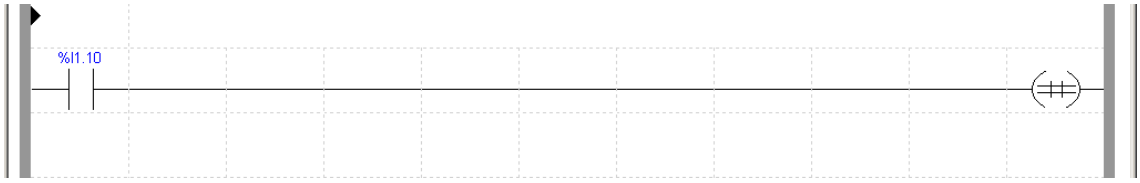


Figura 9.3: Transició entre estats 0 i 1

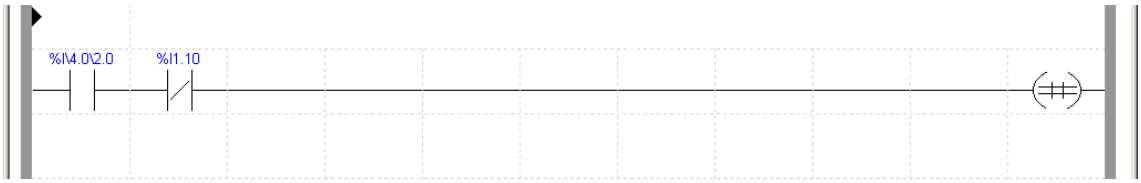


Figura 9.4: Transició entre estats 1 i 0

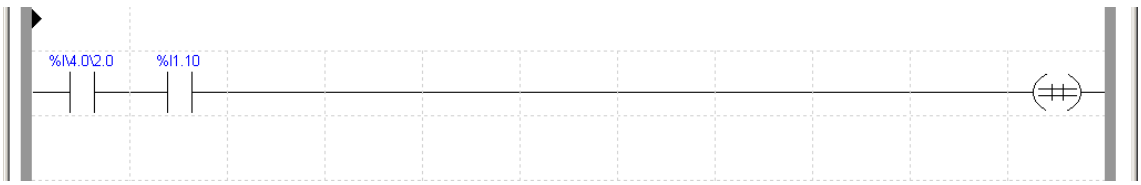


Figura 9.5: Transició entre estats 1 i 2

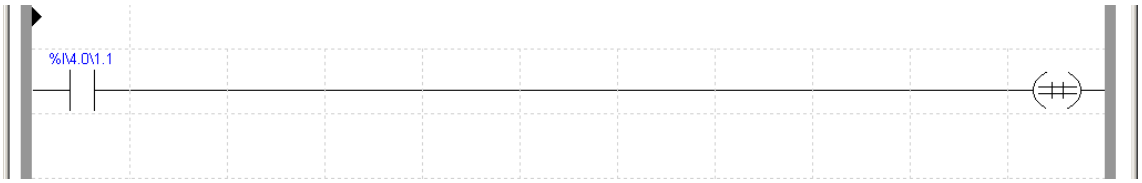


Figura 9.6: Transició entre estats 2 i 3

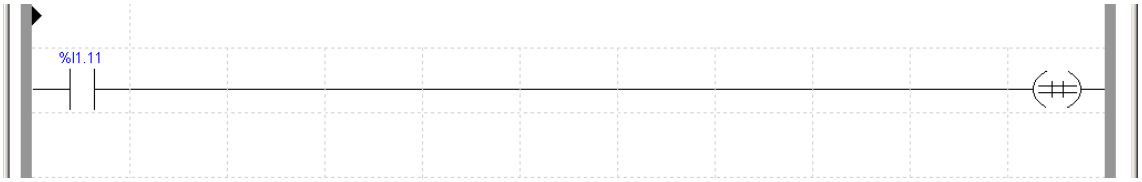


Figura 9.7: Transició entre estats 3 i 0

9.4.4. Contactes

A les figures que apareixen a continuació (9.8 a 9.10) es mostren els contactes, és a dir, les condicions d'activació i desactivació de les senyals del PLC.

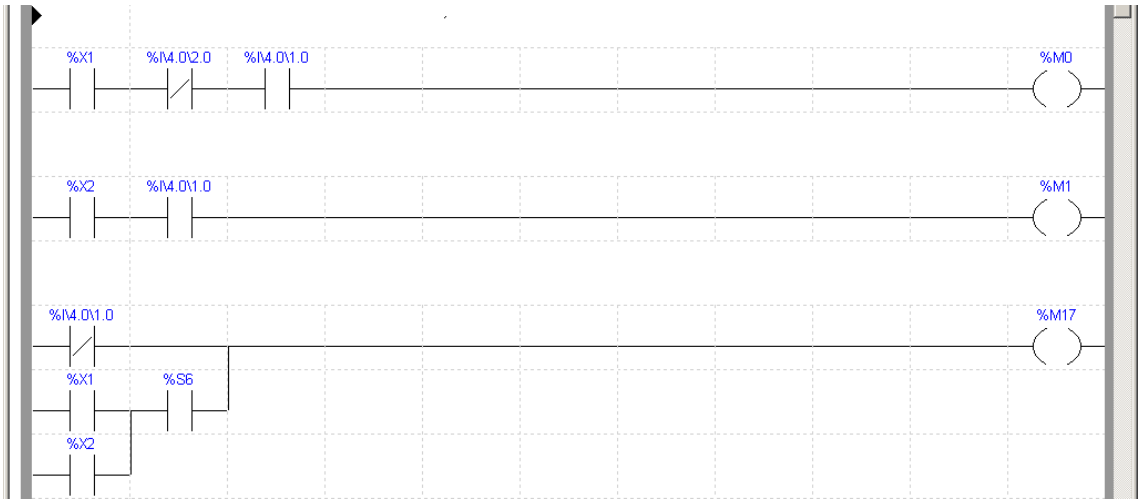


Figura 9.8: Contactes (1)

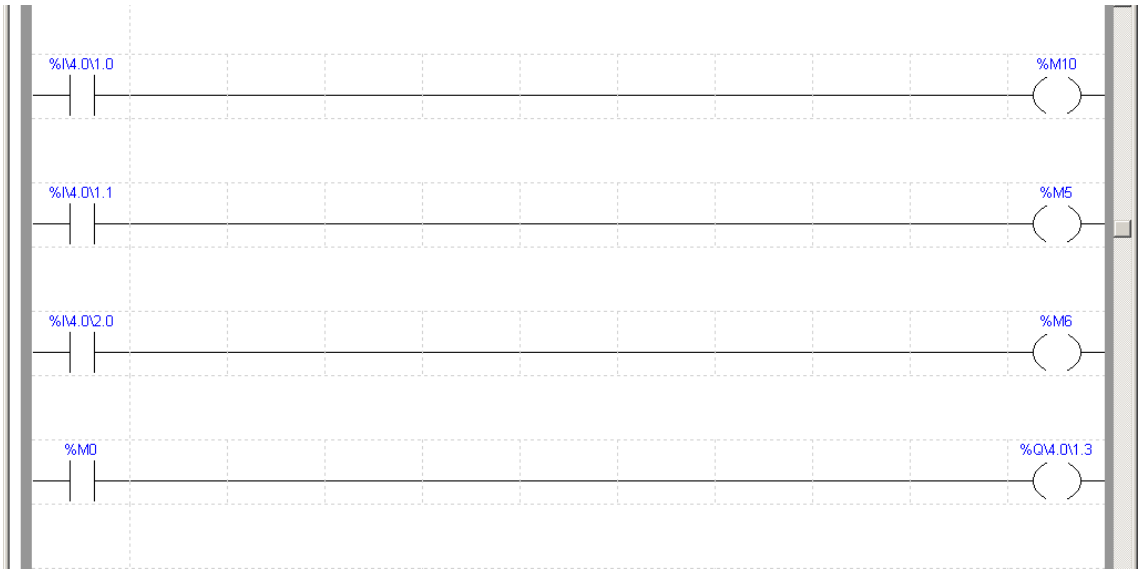


Figura 9.9: Contactes (2)

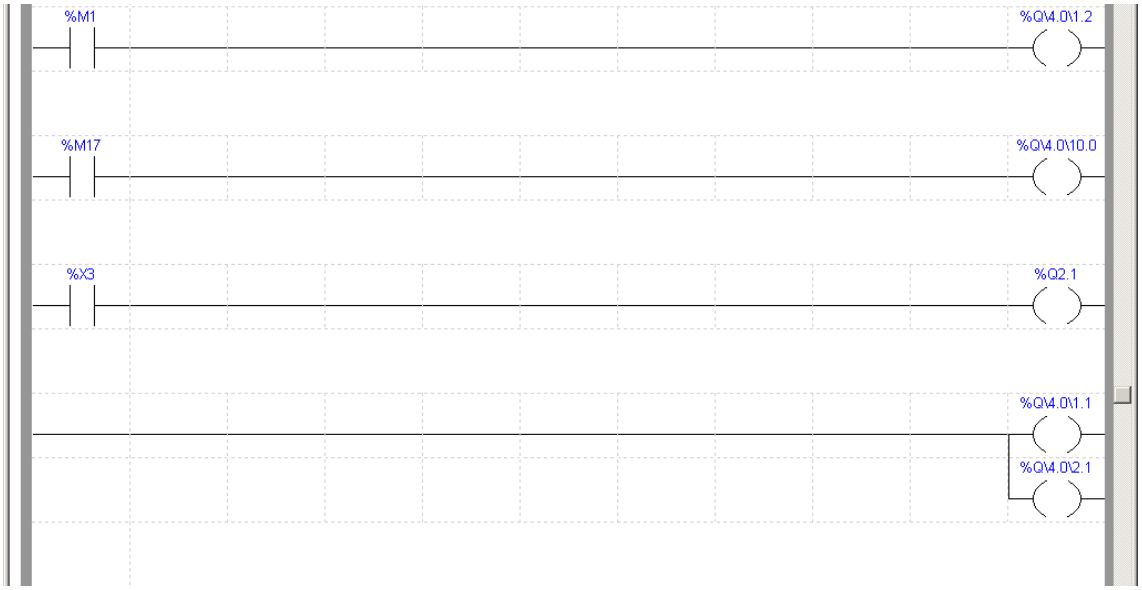


Figura 9.10: Contactes (3)

9.5. APLICACIÓ DE PREPARACIÓ

L'aplicació anomenada "Preparació" permet preparar la càmera per tal que es desenvolupi correctament l'apartat de visió artificial del procés de transport i classificació. Els seus dos objectius són:

- **Comprovar la correcta disposició de la càmera sobre la safata de transport.** És necessari que les captures mostrin únicament l'interior de la safata, doncs els objectes exteriors actuen d'artefactes, els quals afecten negativament a la detecció de les peces. Desplaçarem el trípode fins trobar una posició que compleixi aquesta condició.

- **Comprovar la correcte segmentació de les peces.** Les ombres que apareixen a la safata poden provocar una segmentació errònia dels blobs. Modificarem l'obertura de la lent fins que la segmentació mitjançant Otsu sigui correcta.

9.5.1. Execució

A l'iniciar l'aplicació apareixeran es mostraran les instruccions de funcionament (veure figura 9.11). Al mateix temps una finestra mostrarà la captura que ha realitzat la càmera.

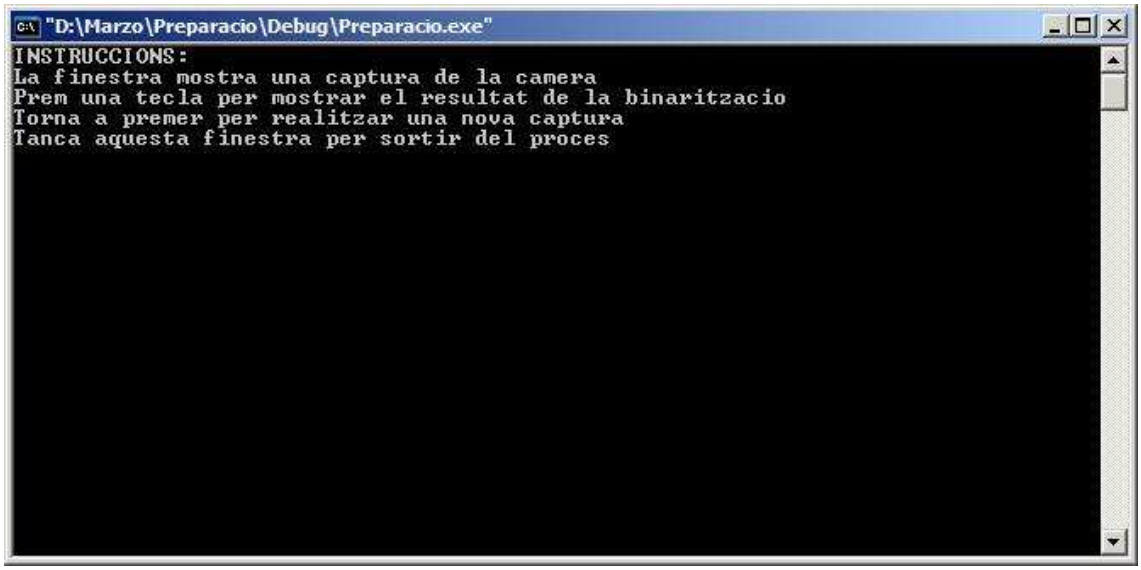


Figura 9.11: Aplicació de preparació

Al prémer qualsevol tecla la finestra passa a mostrar el resultat de la binarització que s'ha aplicat a la imatge prèviament mostrada, la qual ha estat tractada utilitzant el llindar determinat per l'algoritme Otsu. Si es prem de nou una tecla es realitzarà una nova captura, començant de nou el procés.