Universitat de Vic
Escola Politècnica Superior

Treball Final de Carrera

Control d'un sistema de marcatge làser de caixes

Josep Codina i Bau

Enginyeria Tècnica de Telecomunicació,
Especialitat de Sistemes de Telecomunicació

Director: Jordi Rifà i Burniol

Vic, Juny de 2009

Vull agrair el suport de totes aquelles persones que m'han ajudat en la realització d'aquest projecte.

De manera general a tot el personal que conforma l'oficina tècnica de JCM Technologies, i en especial a en Jordi Rifà, que tant en la seva faceta de director del projecte, com a la de responsable d'industrialització, s'ha portat amb mi de manera immillorable.

També a totes aquelles persones que em fan costat dia a dia i que aquests últims dies han estat una mica complicats.

Moltes gràcies.

Índex

Índex de figures

Índex de taules

Índex de diagrames

Resum Treball Final de Carrera

Enginyeria Tècnica de Telecomunicació,

Especialitat de Sistemes de Telecomunicació

Títol: Control d'un sistema de marcatge làser de caixes

Paraules clau: Marcatge, caixes, làser, manipuladora de caixes, interfície gràfica.

Autor: Josep Codina Bau

Direcció: Jordi Rifà Burniol

Data: Juny de 2009

Resum

Avui en dia, l'alta competitivitat que existeix al mercat, fa que les empreses hagin d'esprémer al màxim les seves possibilitats per no quedar-se enrere. Un dels processos en que aquest fet hi és més present és el productiu. L'empresa JCM Technologies també engloba aquest camp i és en un dels seus processos productius on aquest projecte pren part.

L'objectiu d'aquest projecte final de carrera ha estat desenvolupar un sistema per poder marcar caixes mitjançant un làser de CO_2 i un automatisme manipulador de caixes. D'aquesta manera aquest procés productiu té una durada molt inferior a l'antic procés, que consistia en enganxar una etiqueta al lloc on ara és marcat pel làser.

Per satisfer els objectius, s'ha creat una aplicació de Windows que per mitjà d'una interfície gràfica, permet a l'usuari realitzar els passos necessaris per fer el marcatge. Primerament es recullen les dades procedents de la comanda; seguidament es seleccionen les que s'han de marcar a les caixes i s'envien al làser mitjançant comunicació sèrie; una vegada aquesta inicialització ha finalitzat correctament, s'engega la seqüència de marcatge de les caixes, que en marcarà la quantitat indicada. Aquest procés de marcatge consisteix en supervisar l'estat en que es troben certes senyals, procedents de l'automatisme i del làser, i depenent d'aquestes generar-ne unes altres. Aconseguint així realitzar el procés de marcatge de cada caixa.

Com a conclusions cal dir, que els objectius s'han complert, ja que s'ha aconseguit un procés de marcatge ràpid i robust. També s'ha aconseguit que les parts de configuració de l'aplicació, i de les caixes siguin de fàcil manipulació. Per tant, amb l'acompliment dels objectius d'aquest projecte, aconseguim completar el sistema de marcatge, i permetre que les caixes siguin marcades de forma correcta, ràpida i eficient.

Degree Final Project Summary
Technical Telecommunications Engineering
in Telecommunications Systems

Title: Boxes laser marking control system

Key words: Marking, boxes, laser, boxes handler, graphical interface.

Author: Josep Codina Bau

Directed by: Jordi Rifà Burniol

Date: June 2009

SUMMARY

Nowadays, competitiveness make companies squeeze their possibilities to the maximum to keep up with the market. One of the processes most affected by competitiveness is the production process. JCM Technologies is constantly improving its production process, and part of it is handled by this project.

This project target is to speed-up the production processa by developing a system that marks boxes in the shortest time possible. The marking is done by means of a CO_2 laser and a box handling automatism. This way the production process will be much shorter than before, when boxes where identified with labels on top.

In order to meet this project's targets, a Windows application has been developed, which allows the user to do what is necessary for the marking. First of all, the order's information/data is picked. Then, the relevant part of this information is sent through serial communication to the laser. Once this initialization has finished correctly, the boxes marking sequence is turned on until the indicated quantity is marked. As the application knows the quantity, it finishes on its own. The marking process deals with knowing some signal levels coming from the automatism and the laser. Depending on that information, some out signals are generated or not. This way, each box is marked.

As a conclusion, all targets have been successfully achieved. The marking process is made fast and robust. Also, the configuration of boxes and application can be easily modified. So, with all project goals achieved, the marking system is completed and the boxes will be marked quikly, efficiently and correctly.

1. Introducció

Actualment, durant els processos productius de qualsevol producte és on les empreses intenten reduir al màxim el seu preu de cost. Per fer-ho, s'està constantment buscant alternatives i quan se'n troba alguna que aparentment és viable, s'hi aprofundeix. L'empresa JCM Technologies també treballa en aquest punt i aquest projecte pren part en un dels seus processos productius.

El procés que s'optimitza és el de personalització de les caixes amb les que es venen certs productes. Fins avui en dia, es feia enganxant-hi etiquetes, però el fet d'enganxar-les manualment tenia, entre altres inconvenients, un cost de ma d'obra massa elevat. Aquest fet va fer buscar alternatives i es va arribar a la conclusió que podien desaparèixer si el procés es modificava completament i la particularització es marcava mitjançant un làser.

La persona encarregada de la producció va estudiar a fons l'alternativa, i tant bon punt va tenir clar que era factible, va començar a moure fils per aconseguir els objectius. Es necessitava un làser per marcar les caixes. Un mecanisme capaç de mantenir-les davant el làser mentre es marcaven i les pogués enretirar una vegada marcades. I finalment una aplicació de Windows que permetés a l'operari tenir el control de tot el procés.

El fet que el mecanisme fos un aparell fet a mida, i que el làser hagués de controlar-se durant el procés, va fer que s'hagués de fer una aplicació de control totalment específica. La complexitat d'aquest fet va obrir les portes a la possibilitat de ser un projecte de final de carrera, i així ha sigut.

1.1. Objectius

L'objectiu principal d'aquest projecte és dissenyar un sistema de control i configuració dels elements que conformen el nou procés productiu. Aquest objectiu es pot desglossar en els següents:

- Crear una interfície gràfica capaç de recollir la informació necessària, tractar-la i enviar-la al làser.

- Aconseguir que aquesta interfície gràfica tingui una simplicitat molt elevada per l'operari que hi hagi de treballar.

- Crear, a la interfície gràfica un apartat on es pugui fer la configuració de cada tipus de caixa.

- Fer el control de les senyals del làser i de la màquina, per generar la seqüència de marcatge de cada caixa.

1.2. Estructura de la memòria

Aquesta memòria està dividida en cinc capítols molt diferenciats.

El primer és la introducció i amb ell s'intenta introduir el lector al projecte, explicant, de forma molt general en què consisteix. També és en aquest apartat on s'exposen els objectius del projecte.

El segon capítol és on s'explica amb més profunditat el projecte, així com també els motius que té aquest d'existir i la metodologia que s'ha utilitzat per poder-lo portar a terme.

El tercer capítol és en el que s'explica tot el que fa referència a la part de software del projecte. Es descriu la interfície gràfica creada, els fitxers que utilitza l'aplicació, les classes i mètodes utilitzats i el control d'errors que es segueix pel bon funcionament de l'aplicació.

Al quart capítol, per la seva part, és on es pot trobar tota la informació del hardware. S'explica, entre altres coses, la màquina manipuladora de caixes, el làser i fins i tot el cablejat que es necessita perquè tots els aparells estiguin correctament connectats.

Finalment, a l'últim capítol, és on s'ha fet una valoració econòmica del projecte, comparant el nou procés amb l'antic. També és en aquest capítol on es localitzen els resltats i conclusions d'aquest projecte.

2. Bases del projecte

En aquest apartat s'expliquen de forma general els aspectes bàsics d'aquest projecte. Primerament es parla una mica de l'empresa per la que s'ha portat a terme i les necessitats que aquesta tenia. També es parla de com es gestionen les comandes i els canvis que hi ha pel fet de posar aquest projecte en funcionament. Finalment es parla del projecte en general.

2.1. Empresa

Aquest projecte ha estat creat amb l'objectiu de satisfer dues necessitats. La primera és la que tenia l'autor, ja que per acabar la carrera universitària que està cursant havia de realitzar el Treball de Final de Carrera. La segona, en canvi és una que tenia l'empresa JCM Technologies en un dels seus processos productius.

JCM Technologies és una empresa que consta d'una oficina tècnica amb 27 anys d'experiència en disseny de circuits electrònics, especialitzada en circuits de radiofreqüència.

Actualment dedica la major part del seus recursos a l'estudi i creació de sistemes innovadors d'obertura i seguretat per a portes de garatge, (emissors, receptors, etc.) entre d'altres coses.

Dia a dia l'empresa ha anat guanyant mercat, fins al punt de vendre una quantitat d'emissors molt important a un gran ventall de clients. Aquests clients però, no són els usuaris finals d'aquests aparells, sinó que els clients de l'empresa són els instal·ladors i distribuïdors d'aquest producte. Aquest fet fa que cadascun d'ells vulgui al producte que vendrà, una o varies marques identificatives que facin el seu producte únic i reconeixedor amb un cop d'ull. Per això, per cada client es personalitza d'alguna manera el seu emissor.

Aquests emissors es comercialitzen dins d'una caixa de cartró que, normalment conté el logotip del client entre altres particularitzacions. És a la zona de la caixa de la figura 2.1. marcada amb negre, on es treballa en aquest projecte.

Apart de les peticions que fan els clients, les caixes també es particularitzen per interessos propis de l'empresa. Algunes d'aquestes particularitzacions poden ser els diferents articles que es comercialitzen amb un mateix tipus de caixa.

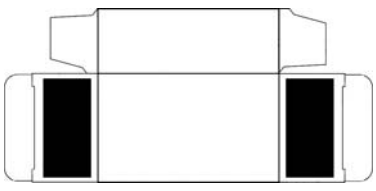


Figura 2.1.1 - Dibuix d'una caixa amb la zona que es marca amb làser de color negre.

El que es pretén fer és acabar de personalitzar la caixa amb la que es vendrà el producte, marcant-hi la informació necessària mitjançant un làser de CO_2.

El fet que la zona de la caixa on es marcarà sigui de color negre, està fet amb la finalitat que quan el làser cremi aquesta pintura, apareix el fons blanc del cartró de la caixa, permetent llegir el text.

2.1.1. Gestió de les comandes

JCM Technologies és una empresa amb una estructura molt ben dissenyada i uns departaments molt ben definits. Aquest fet ajuda molt a l'hora de treballar ja que el fet que cadascú tingui tant clar el seu rol fa les coses molt més simples.

En aquest projecte ens cal conèixer com es tracten les comandes i sobretot, la part final d'aquest procés en que passen de ser una petició a una realitat.

Processament d'una comanda:

- El departament comercial, després d'haver fet les gestions pertinents per aconseguir que el major nombre de clients comprin la màxima quantitat d'un cert producte, son els encarregats de rebre les comandes. Una vegada tenen clar el que desitja el client, entren les dades rebudes al sistema de planificació de recursos[1] de l'empresa. Així doncs, una vegada han entrat aquestes dades, per aquella comanda la seva feina ja ha acabat.

- En aquest punt la comanda passa a ser gestionada pel departament de logística, encarregant-se de generar ordres de treball a partir de les dades entrades pel departament comercial. Aquestes ordres de treball son on hi ha tota la informació del que s'ha de fer per satisfer la comanda.

- Una vegada generades les ordres de treball, via Internet, el taller d'acabats dels productes accedeix al sistema de planificació de recursos de l'empresa. Des del seu lloc es poden visualitzar totes les ordres de treball que tenen pendents i escollir la que més els convingui en cada moment.

- Un cop l'operari s'ha decidit per una, importa la informació relativa a l'etiquetatge i producció dels equips continguda en l'ordre de treball amb un programa informàtic. Aquest genera els fitxers d'informació necessaris perquè cadascuna de les parts del taller que ha d'intervenir en la comanda, tingui les dades que necessita. Aquests fitxers els genera amb format xml i depenent de l'ordre de treball sobre la que s'estigui treballant és possible que generi una quantitat de fitxers bastant important.

[1] Sistema de gestió i informació que integra i automatitza moltes de les tasques associades amb els aspectes operatius i productius d'una empresa.

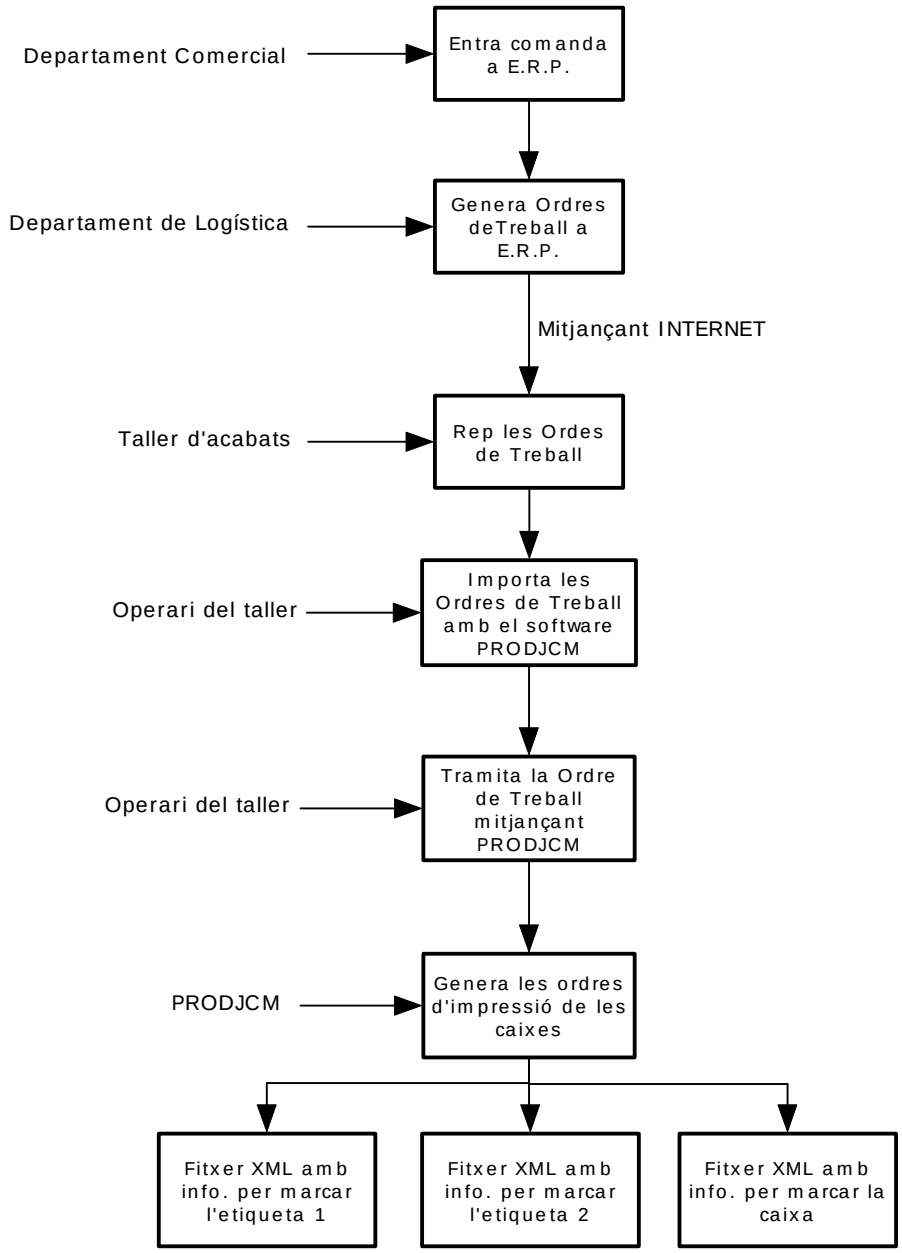


Diagrama 2.1.1 - Diagrama de blocs de la gestió de les comandes.

En aquest punt, quan l'ordre de treball s'ha dividit en petites tasques independents en forma de fitxers, aquests són recollits pels aparells encarregats de plasmar a la realitat la informació que contenen. És en aquest moment en el que aquest projecte comença a prendre part.

2.1.2. Millora del procés

Les diferents tasques a fer, derivades d'una comanda eren decidides, partint de la informació de producció continguda en unes instruccions. Llavors es procedia a la personalització de les caixes, fet que consistia en enganxar-los etiquetes amb la

informació pertinent. Aquest procés però, té un nombre bastant elevat de desavantatges que són els següents:

- Una persona ha d'enganxar totes les etiquetes gastant molt temps.
- Les etiquetes es poden desenganxar, quedar tortes o mal impreses.
- Cada etiqueta té un cost d'adquisició i un cost afegit de marcatge.
- La tinta amb la que s'imprimeixen les caixes pot acabar marxant.
- La introducció de dades manuals al sistema per generar les etiquetes pot induir a errors.

Aquests motius fan que es busqui un mètode més econòmic per portar a terme aquest procés productiu. Després d'estudiar-ho molt profundament, la persona encarregada de la producció arriba a la conclusió que el més adient és que les caixes siguin marcades amb un làser. D'aquesta manera, tots els motius que feien de les etiquetes, un mètode poc rendible per la personalització de les caixes, desapareixen.

2.2. Descripció del projecte

En aquest nou procés de producció es parteix d'una base bastant diferent de la del procés anterior. Com s'ha explicat a l'apartat 2.1.1., en aquest les dades ja no les entra l'operari sinó que es disposa d'un programa informàtic encarregat de generar-les i guardar-les en un fitxer xml.

Un altre dels elements importants dels que es disposa és d'una màquina manipuladora de caixes. Aquesta, manté les caixes davant el làser abans que aquestes siguin marcades, mentre que una vegada marcades, disposa dels elements necessaris per retirar-les.

Amb tots aquests elements, per poder portar a terme el marcatge de les caixes, el primer que s'ha fet ha estat crear una interfície gràfica, des d'on la persona que hi treballa pugui controlar-ho tot. Aquesta interfície està dividida en tres pestanyes. Dues de configuració, en les que hi treballa la persona encarregada del manteniment del sistema, i una per fer el marcatge de les caixes, que és en la que treballa l'operari.

La pestanya en la que treballa l'operari consisteix bàsicament en enviar les dades al làser i engegar la seqüència de marcatge de les caixes. Per poder fer aquests passos, l'operari visualitza els fitxers que te procedents de les ordres de treball que encara no ha tractat i escull el que desitja fer en aquell moment. Una vegada escollit, l'aplicació tria les dades que es necessiten i l'operari les envia al làser. Quan aquest està inicialitzat correctament i l'operari ha col·locat les caixes a la màquina manipuladora, aquest engega la seqüència de marcatge. En aquest moment l'operari pot abandonar aquest lloc de treball per realitzar-ne un altre, ja que si no hi ha cap imprevist, quan s'hagin acabat de marcar totes les caixes, l'aplicació l'avisarà.

La segona pestanya de la interfície gràfica s'anomena control de l'aplicació i el que el seu nom indica és el que fa. D'es d'ella es pot visualitzar la totalitat del contingut del fitxer de dades i també, en el cas que la seqüència estigui engegada, es pot saber

en tot moment el punt en el que es troba. D'aquesta manera, si hi ha algun problema, és immediat localitzar-lo.

A més a més d'aquesta part informativa, aquesta pestanya disposa d'una última zona en la que es poden configurar tres paràmetres de l'aplicació. El primer és el port sèrie de l'ordinador pel que s'envien les dades al làser. El segon és la ruta del directori on es localitzen els fitxers de dades i el tercer és la ruta del directori on aquests fitxers s'han de guardar una vegada han estat tractats.

La tercera pestanya és on es configura el que es vol que es marqui en cada tipus de caixa determinat. Cada tipus de caixa o cada client tenen una demanda diferent en la informació que s'ha de plasmar a la caixa, per aquest motiu, aquesta informació és totalment configurable. En aquesta tercera pestanya doncs, és on la persona encarregada del manteniment del sistema pot assignar la informació que s'ha de marcar a cada caixa.

2.3. Metodologia del projecte

2.3.1. Requeriments de l'empresa

Com ja s'ha explicat, l'empresa tenia la idea principal molt ben definida, cosa que va fer que una sèrie d'aspectes del nou procés productiu estiguessin ja establerts.

- El làser SmartLase 110i, que és amb el que es fa el marcatge de cada caixa.
- L'automatisme de manipulació de caixes.
- Creació d'una interfície gràfica.
- Per coincidir amb altres aplicacions de Windows que s'han realitzat a l'empresa, l'entorn de programació i el llenguatge també van ser un requeriment. L'entorn de programació calia que fos el Microsoft Visual Studio 2005 i el llenguatge, C-Sharp (C#).

2.3.2. Detecció de les necessitats

La primera cosa que es va fer una vegada escollit aquest projecte com a treball de final de carrera, va ser la identificació de les necessitats que hi havia partint dels requeriments que posava l'empresa.

1) Adaptació als elements ja escollits.
2) Elecció del sistema de control de l'automatisme de manipulació de caixes.
3) Localització d'un mètode ràpid i fàcil de control.
4) Creació del mètode de control.

2.3.3. Estudi possibilitats i elecció de la millor opció

Quan les necessitats estan ja ben clares el que cal fer és un estudi de les possibilitats que existeixen, per poder solucionar el problema de la manera més adient possible.

L'apartat que més temps va necessitar, va ser el de l'estudi de com controlar la màquina manipuladora de caixes. En un primer moment hi havia una quantitat bastant elevada de possibilitats, de les quals es va acabar amb tres de viables, que eren: una targeta industrial d'entrades i sortides, un controlador lògic programable (PLC) o un sistema electrònic propi.

D'aquestes tres opcions n'hi van haver dues que de mica en mica van deixar de ser-ho. Tant el controlador lògic programable com el sistema electrònic propi afavorien molt el tractament de les senyals, però el pas més important, el d'aconseguir que des de l'aplicació es poguessin llegir i generar senyals, continuava sent complicat. La opció d'un sistema electrònic propi va ser estudiat amb profunditat, però la feina que demanava de disseny, de programació i de proves, no es podia comparar amb les altres opcions. Pel motiu que en aquesta part només es precisava de funcionalitat i no de gastar-hi molt temps, es va escollir la opció de la targeta.

Tant bon punt es va decidir que es feia mitjançant una targeta industrial d'entrades i sortides, la búsqueda es va centrar en dos factors. El primer és que havia de disposar de detecció de nivell i de flanc en el cas de les senyals d'entrada. Mentre que el segon factor, era que els paràmetres de tensió i corrent havien de poder suportar les característiques dels elements dels que ja es disposava. L'últim factor que es va tenir en compte i que va ser el que va fer escollir-ne una i no les altres va ser el cost econòmic de les diferents targetes.

Una vegada es disposava de totes les senyals, independentment del mètode escollit per tractar-les, calia estudiar la manera amb la que l'usuari pogués fer tot el que calia. De dissenys d'interfícies gràfiques en van aparèixer una infinitat i la veritat és que anant tot sovint a fer proves de funcionament al taller d'acabats, entre els propis operaris, l'encarregat de producció i el programador, s'ha creat la definitiva.

2.3.4. Procés d'elaboració del projecte

Tant bon punt s'havia escollit tot el que calia, el que es va fer va ser l'estudi de cadascuna de les parts per separat.

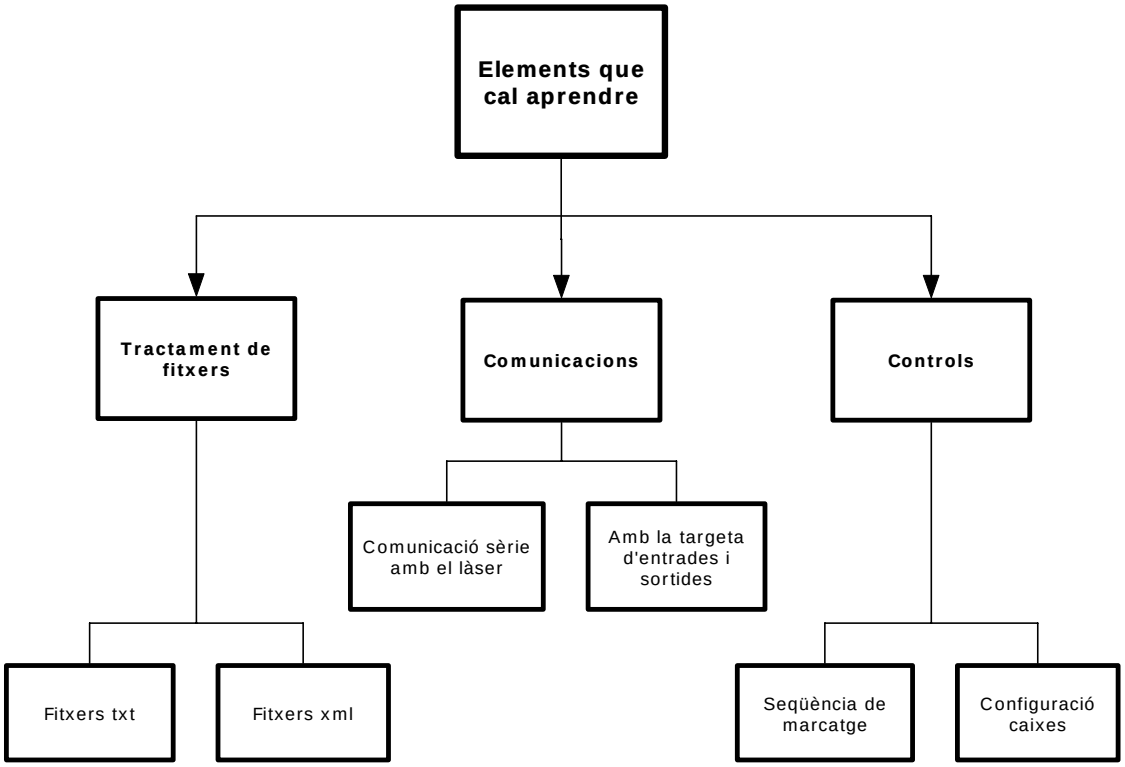


Diagrama 2.3.1 - Esquema de les coses que es va haver d'aprendre a fer.

Com es pot veure a l'esquema 2.3.1., hi havia molts blocs que no se sabia com portar a terme. Aquest fet va fer que per cadascun d'ells s'hi hagués d'invertir el temps necessari per aprendre'n el funcionament o la manera de programar-ho.

Aquest aprenentatge va ser igual per a tots els casos i és el següent:

1) Crear d'un projecte nou amb les mateixes característiques que el principal.
2) Afegir-hi tots els elements necessaris.
3) Aprendre el funcionament dels elements, la manera de programar-los i aconseguir fer el que es desitja.
4) Afegir, amb les modificacions pertinents, els nous elements de control al projecte principal.

3. Software

3.1. Fitxers que utilitza l'aplicació

Per poder treballar correctament, aquesta aplicació precisa d'una sèrie de fitxers que li aporten informació necessària, i això és el que s'explica en aquest apartat.

De fitxers d'aquests n'hi ha tres, dos es troben en format "xml" i el tercer està en format "txt".

3.1.1. Fitxer de configuració de les caixes

Aquest fitxer es troba en format xml i és on es guarda la informació de cada caixa, així com la dels diferents camps que es poden assignar a cadascuna d'elles, per poder generar totes les combinacions de caixes necessàries. És una base de dades dinàmica, és a dir, que es modifica habitualment.

El fet que els paràmetres estiguessin tant clars i fossin tant diferenciats entre ells no va donar lloc a dubte per com havia de ser l'esquema basic del fitxer.

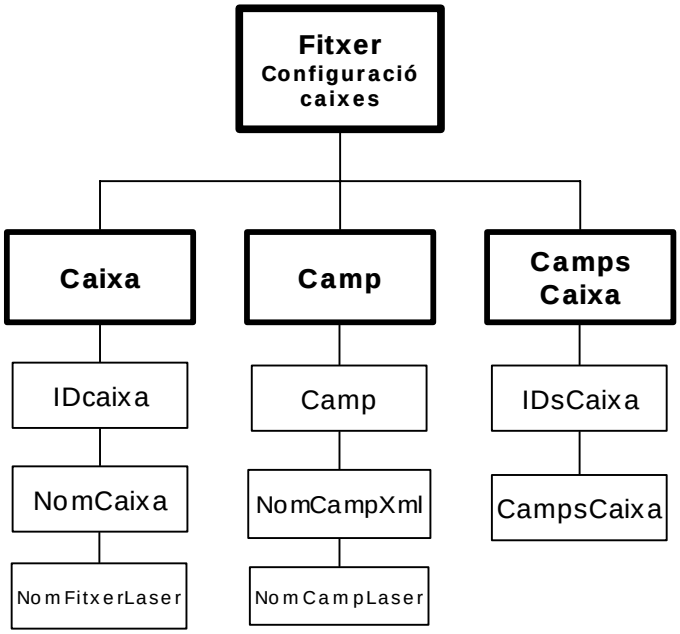


Diagrama 3.1.1 - Esquema de l'organització del fitxer de configuració de les dades.

Al diagrama 3.1.1. es pot veure com el fitxer de configuració està dividit en tres parts molt diferenciables: camps, caixes i assignació de camps a caixes.

Cadascun d'ells consta de la informació que es necessita per poder portar a terme totes les combinacions de camps i caixes.

Informació de caixa

Nom	Descripció
IDcaixa	Identificador numèric únic de la caixa.
NomCaixa	Variable que dóna nom a cada caixa. Depenent de la situació es prefereix treballar amb el nom o amb el valor del paràmetre anterior.
NomFitxerLaser	Variable que indica el nom amb el que la caixa està guardada al làser.

Taula 3.1.1 - Taula descriptiva dels elements que formen una caixa.

Informació de camp

Nom	Descripció
Camp	Nom del camp.
NomCampXml	Nom amb el que aquest camp es troba dins el fitxer de dades de la comanda.
NomCampLaser	Variable que indica el nom amb el que el camp està guardat a la memòria interna del làser.

Taula 3.1.2 - Taula descriptiva dels elements que formen un camp.

Configuració de caixa

Nom	Descripció
IDsCaixa	Identificador numèric únic de la caixa.
CampsCaixa	Nom del camp associat a un IDcaixa.

Taula 3.1.3 - Taula descriptiva de la configuració d'una caixa.

3.1.2. Fitxer de dades

Aquest fitxer també es troba en format xml i és on es troba la informació de la comanda o ordre de treball.

Cada vegada que l'empresa rep una comanda, tal com s'explica a l'apartat 2.1.1., el sistema informàtic recopila les dades i genera els fitxers necessaris perquè cada part del procés productiu es pugui portar a terme correctament. Els fitxers destinats a la impressió de caixes, és on es troba tota la informació del producte (nom del client, del producte, la quantitat, etc.). Apart d'aquesta informació esmentada se n'hi guarda molta d'altra.

Com l'altra base de dades de l'apartat 3.1.1., aquesta també es troba en format xml, però és estàtica, ja que la informació només és llegida i en cap moment es modifica.

Un altre aspecte a tenir en compte és el fet que no sempre hi ha els mateixos camps, ja que cada client i cada producte poden requerir d'informació diferent. Això fa que sigui una base de dades amb un esquelet semblant per tots, però depenent de la comanda, amb uns paràmetres probablement bastant diferents.

3.1.3. Fitxer de configuració de l'aplicació

Aquesta base de dades és un petit fitxer de text que emmagatzema la informació necessària perquè cada vegada que s'engegui l'aplicació estigui configurada tal com estava l'última vegada que es va utilitzar.

És una base de dades dinàmica, ja que es pot modificar sempre que sigui necessari i es llegeix cada vegada que s'obre la interfície gràfica.

Els paràmetres que es guarden a aquesta base de dades es resumeixen en la següent taula.

Paràmetre	Descripció
Valor del port sèrie	A aquesta variable s'hi guarda el número de port sèrie de comunicació amb el làser.
Directori fitxer de dades	És on hi ha guardada la ruta del directori on es troben els fitxers de dades de les comandes, generades pel software explicat a l'apartat 2.1.1..
Directori de Back Up del fitxer de dades	És on es guarda la ruta del directori on l'aplicació ha de deixar els fitxers de dades una vegada aquestes han estat extretes i la totalitat de les caixes ha estat marcada correctament. Per poder fer un seguiment del moment en que es fan els marcatges i els fitxers es processen, en el trasllat del fitxer se li afegeix la data i l'hora.

Taula 3.1.4 - Taula amb el contingut del fitxer de config. de l'aplicació.

3.2. Interfície gràfica

La interfície gràfica és la part del projecte que veu l'usuari de l'aplicació a la pantalla de l'ordinador i amb el que aquest ha d'interactuar.

Amb la intensió de buscar la màxima productivitat, s'ha realitzat una interfície gràfica simple i sense complicacions. Aquest fet té la finalitat de no crear problemes a l'operari a l'hora de recordar la funció de cada botó o control. Aquesta simplicitat però, suposa un increment de la complexitat interna en el seu desenvolupament.

La interfície gràfica es divideix en tres blocs diferenciats: el marcatge del làser, el control de l'aplicació i la configuració de les caixes.

3.2.1. Pestanya de Marcatge

Aquesta pestanya és la que més s'utilitza de totes ja que és amb la que treballa l'operari de producció.

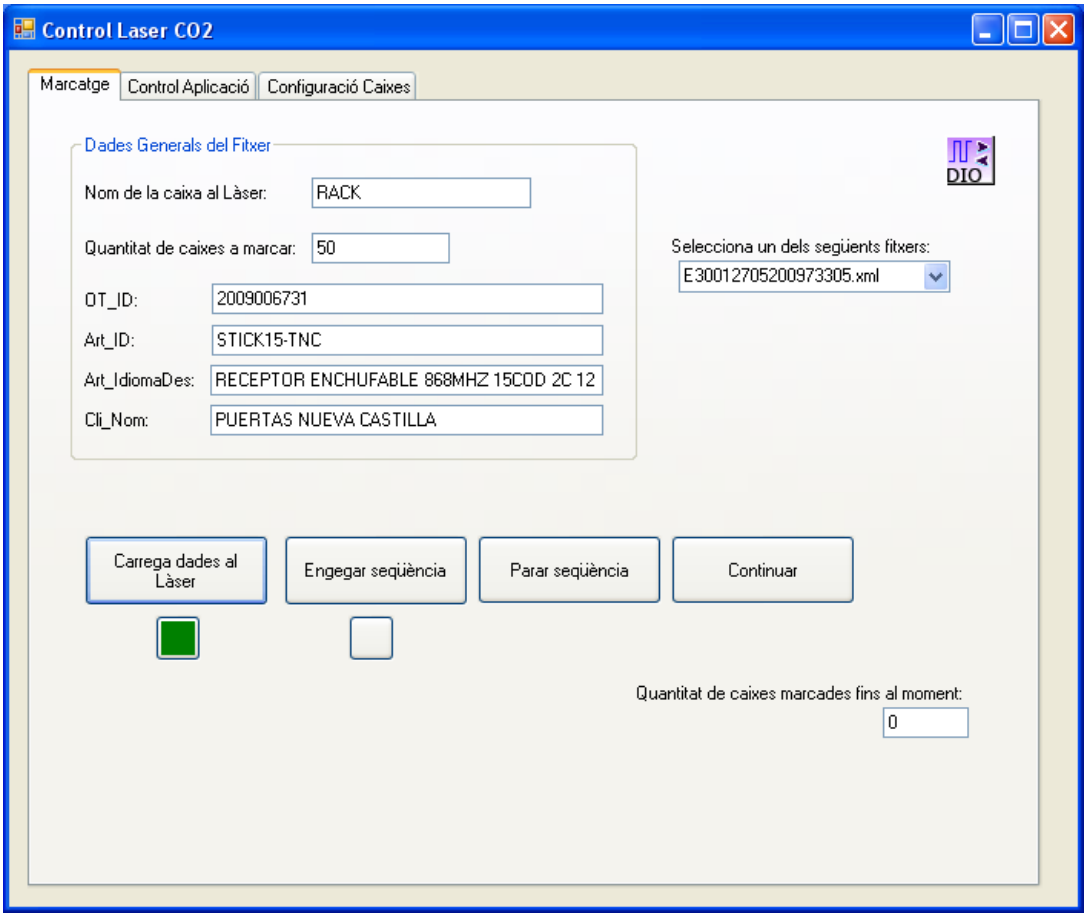


Figura 3.2.1 - Pestanya Marcatge.

3.2.1.1. Dades generals del fitxer de dades

A la part superior esquerra d'aquesta pestanya s'hi localitza un control "groupBox" amb el text "Dades Generals del Fitxer". Aquest control està format per diversos controls de text, la meitat dels quals el text varia i a l'altra meitat no.

Per mitjà del control "comboBox" que hi ha a la part superior dreta de la pestanya, l'operari pot veure tots els fitxers de dades que en aquell moment te pendents de tractar. En el moment que selecciona un dels fitxers de la llista, tots els valors dels camps del control "groupBox" que hi ha a la seva esquerra queden modificats mostrant la informació més important del fitxer. Això però, no passa només amb el primer que es selecciona, sinó que es pot fer tantes vegades com es vulgui sobre els diferents fitxers que hi ha a la llista.

D'aquesta manera abans de seleccionar un fitxer i enviar les dades cap al làser, l'operari pot conèixer el fitxer i confirmar que escull el correcte.

3.2.1.2. Botons de Control

Càrrega de dades al làser

Amb el fitxer de dades seleccionat, per recollir les dades necessàries i enviar-les al làser l'operari només prémer un control "button". En aquest moment l'aplicació recull les dades d'origen i les transmet al làser, cosa que fa que tardi unes dècimes de segon a reaccionar. En el moment en que acaba però, un altre control "button" que hi ha just a sota seu es posa verd indicant que la recollida d'informació i la transferència d'aquesta cap al làser ha estat correcte.

Si pel motiu que fos algun dels processos no es pot portar a terme correctament, l'aplicació informaria a l'operari amb els missatges d'error pertinents.

Perquè l'operari no pugui passar al següent pas si aquest no s'ha complert satisfactòriament es deshabilita el control d'engegar la seqüència durant el procés de transferència de dades i s'habilita tant bon punt ha acabat.

Botó d'engegar seqüència

En el moment en el que el làser té la informació carregada i es troba en el mode de marcatge, l'operari posa en marxa l'automatisme.

Per fer-ho només ha de prémer un control "button", i una vegada premut el botó, just a sota d'aquest, es posa en verd un indicador informant que la seqüència està engegada.

Una vegada engegada, si tot va bé, en el moment en que s'hagin marcat la quantitat de caixes que el fitxer de dades marcava, l'aplicació pararà la seqüència i n'informarà a l'operari. D'aquesta manera no ha d'estar pendent de quan acabarà, ja que sap que pot anar a buscar les caixes marcades quan li vagi bé.

Botó de parar seqüència

Aquest botó serveix per parar la seqüència.

Si per exemple, l'operari una vegada la seqüència està engegada s'adona que no ha posat prous caixes al dispensador i que si no en posa s'acabaran, i apareixerà el missatge d'error. Pot parar la seqüència amb aquest botó, posar les caixes que li faltaven i tornar a engegar la seqüència perquè marqui totes les que tocaven.

Botó de continuar

Com s'explica a l'apartat anterior és possible que la seqüència de marcatge s'hagi de parar a mig fer. Quan es dóna aquest cas, de la quantitat inicial de caixes que s'havien de marcar, n'hi ha unes quantes que ja s'han fet. Per aquest motiu, tornar a prémer el botó d'engegar la seqüència seria un error perquè tornaria a començar des del principi.

Perquè això no passi, s'ha posat aquest botó que el que fa és continuar la seqüència a partir del punt on s'havia deixat i d'aquesta manera es marquen només la quantitat de caixes que restaven del total.

3.2.2. Pestanya de configuració de l'aplicació

La pestanya de configuració de l'aplicació és on ha de recórrer l'encarregat del manteniment del sistema, quan alguna cosa no funciona

En aquesta pestanya, es pot veure la seqüència de marcatge de les caixes i es pot modificar alguna configuració en cas de detectar algun problema.

Aquesta pestanya es reserva a la persona encarregada del manteniment del sistema, ja que l'operari de producció no l'ha de modificar en cap moment.

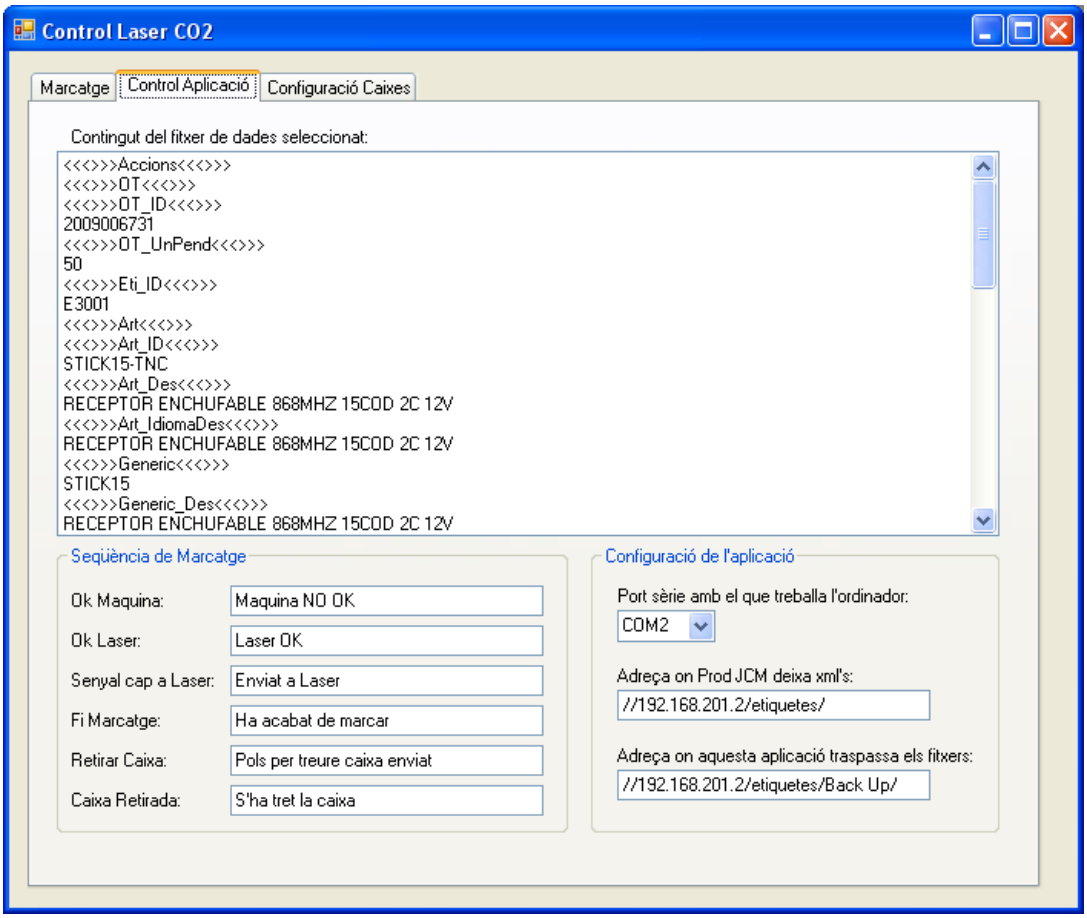


Figura 3.2.2 - Pestanya de Configuració de l'aplicació.

Aquesta pestanya, igual que la de configuració de les caixes, es troba dividida en tres parts força diferenciades.

3.2.2.1. Contingut del fitxer de dades

A la part superior hi ha un control "listBox" que en el moment que l'operari dóna l'ordre d'enviar les dades al làser, aquest visualitza tota la informació que hi havia al fitxer de dades. Això permet comprovar el valor de les dades del fitxer d'origen en el cas d'adonar-se que alguna dada no s'està marcant correctament.

3.2.2.2. Seqüència de marcatge

A la part inferior esquerra s'hi localitza un control "groupBox", que engloba una quantitat bastant elevada de controls de text. Aquests controls són els encarregats de donar informació de l'estat de la seqüència de marcatge i només són informatius.

Per resumir aquesta seqüència s'ha creat la taula 3.2.1. on es poden veure els senyals que es tracten a la primera columna. A la segona s'hi localitza el sentit que aquestes tenen, és a dir, qui les genera i qui les rep. A la tercera es descriu la seva

funcionalitat i finalment, a la quarta, hi ha les possibilitats de text amb les que s'informa a l'usuari de l'estat de cada senyal.

Senyal	Sentit de la comunicació	Funció	Variants
Estat manipuladora de caixes	Manipuladora → Aplicació	Comprovar si el sistema d'extracció de caixes està apunt.	Màquina OK, Màquina NO OK.
Estat del làser	Làser → Aplicació	Comprovar si el làser està apunt per marcar.	Làser OK, Làser NO OK.
Marcatge	Aplicació → Làser	Informar al làser que faci el marcatge.	Pols cap al làser enviat
Estat del marcatge	Làser → Aplicació	Comprovar si el làser ha acabat de marcar o encara no.	El làser encara no ha acabat de marcar, El làser ha acabat de marcar
Retirar caixa	Aplicació → Manipuladora	Informar a la màquina manipuladora que retiri la caixa marcada.	Pols per retirar caixa enviat.
Caixa retirada	Manipuladora → Aplicació	Comprovar si la caixa marcada ha estat retirada correctament.	La caixa no s'ha retirat, Caixa retirada.

Taula 3.2.1 – Taula descriptiva seqüència de marcatge.

3.2.2.3. Configuració de l'aplicació

A la part inferior dreta de la pestanya que s'està descrivint s'hi localitza un segon control "groupBox" amb el títol "Configuració de l'aplicació". Com el seu nom indica, és aquí on es poden fixar paràmetres operatius.

Configurar número de port sèrie

Aquest pas només s'ha de fer la primera vegada que s'envien dades cap al làser o la primera vegada que es fa amb un ordinador nou. El control "comboBox" amb el que s'ha fet aquest pas, al ser desplegat, permet visualitzar tots els ports sèrie dels que disposa l'ordinador i seleccionar l'adequat.

Si el port que es prova no és el correcte i l'aplicació no rep cap tipus d'informació del làser una vegada ha començat a enviar-li dades, després d'un temps d'espera, avisa amb un missatge d'error informant que el làser no contesta.

Figura 3.2.3 - Missatge d''error que informa que el làser no respon.

Un cop configurat, aquest paràmetre queda guardat en el fitxer de configuració de l'aplicació i cada vegada que aquesta s'executa, en recupera el valor.

Ruta del fitxer de dades

Semblant al pas anterior, aquest tampoc s'ha de fer cada vegada ja que normalment es trobaran al mateix lloc. Només s'ha de fer la primera vegada que s'executa l'aplicació o en el cas que pel motiu que fos aquests fitxers canviessin de localització.

Ruta dels fitxers de dades processats

Aquest pas és igual que l'anterior, només s'ha de configurar la primera vegada que s'executa l'aplicació o si es desitja que l'aplicació deixi els fitxers a un altre directori.

3.2.3. Pestanya de configuració caixes

A aquesta pestanya s'identifiquen les caixes i s'assignen els camps que s'han de marcar en cadascuna d'elles.

Igual que la pestanya de configuració de l'aplicació, aquesta també està reservada a la persona encarregada del manteniment del sistema.

3.2.3.1. Caixa

Els models de caixes són diversos segons els productes o els clients. Per cada model de caixa, depenent de les dades que s'hi vulguin marcar, es crearà una plantilla al làser amb un *IDcaixa* únic.

3.2.3.2. Camps

Són diferents informacions que es poden marcar a una caixa.

Exemples de camps poden ser: Nom de l'equip, data de fabricació, nom del client, etc.

3.2.3.3. Configuració dels dispositius

És on s'assigna a cada caixa els camps que se li han de marcar.

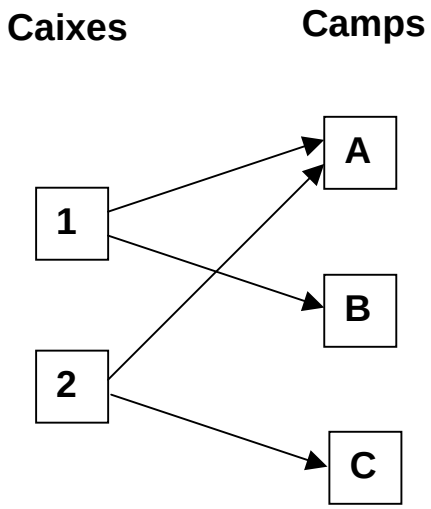


Diagrama 3.2.1 - Exemple de dues caixes amb configuracions diferents.

Degut a que cada client desitja que a la zona de marcatge de la caixa hi hagi escrit paràmetres diferents, i vulgui canviar-les en funció de les seves necessitats, s'ha creat una manera pràctica i senzilla de modificar-les. Aquest fer permet a la persona encarregada del manteniment del sistema interactuar amb el fitxer de configuració de les caixes, on hi ha aquesta informació.

S'utilitzen controls "dataGridView", amb els que l'operari pot modificar en tot moment les variables que vulgui, així com afegir-n'hi o borrar-ne.

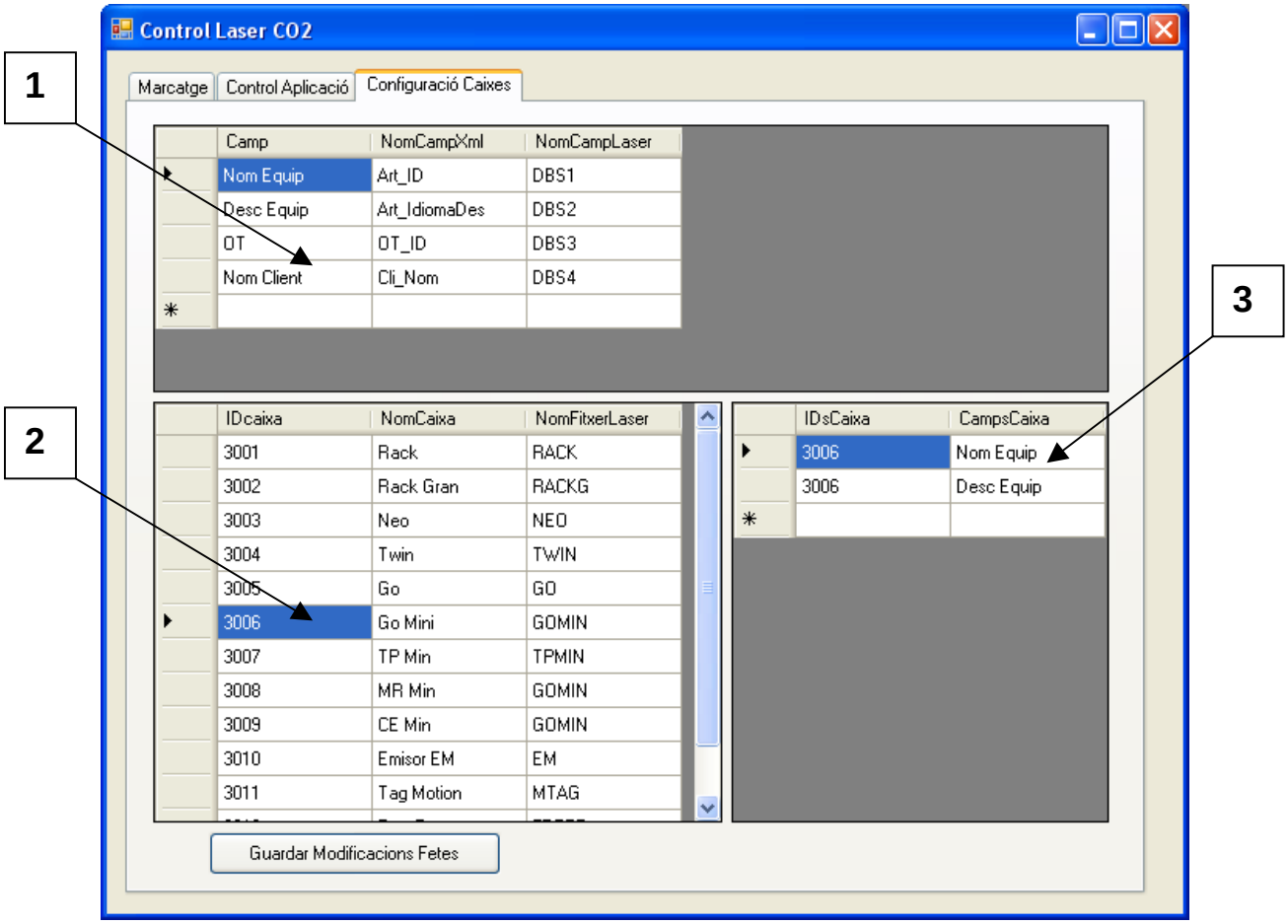


Figura 3.2.4 - Pestanya de Configuració de caixes.

Com es pot veure a la figura 3.2.4., és molt simple i ràpid el fet de visualitzar el que conté el fitxer de configuració. D'aquesta manera el fet de modificar-lo no genera cap mena de complicació.

1

A la part superior d'aquesta pestanya es pot observar com al primer control "dataGridView" s'hi troben localitzats els camps. S'han col·locat a la part superior perquè són el nivell més baix del marcatge i abans de crear una caixa nova o modificar-la, sempre s'haurà de mirar si el camp que es desitja existeix.

A la primera columna es pot veure el nom descriptiu que té el camp. La segona columna és la que mostra el nom que aquest camp té al fitxer de dades i finalment, la tercera relaciona el camp de dades, amb el camp del làser on cal deixar-les.

2

A la part inferior esquerra s'hi troba la taula on es poden visualitzar les diferents caixes que hi ha configurades. A la primera columna s'hi pot veure el número identificador de cada caixa. A la segona és on hi ha el nom descriptiu de la caixa, mentre que a la tercera hi ha el nom de la plantilla corresponent a la caixa que està guardada a la memòria interna del làser.

3

A la part inferior dreta de la pestanya, s'hi pot veure la taula que permet visualitzar ràpidament els camps que cada caixa te configurats. A la primera columna hi ha l'identificador de la caixa i a la segona el camp que aquella caixa té configurat. Com més files apareguin a aquesta taula, més camps té configurats la caixa.

Les dues primeres taules descrites estan visibles en el moment en que s'executa l'aplicació. La tercera en canvi, degut a que no se sap de quina caixa vol informació l'usuari, no es visualitza fins que aquest fa un clic sobre alguna fila de la taula de les caixes.

3.3. Introducció al codi font

En aquest apartat es fa una petita introducció del codi i de l'estructura que s'ha seguit a l'hora de programar.

Per realitzar el codi d'aquest projecte es va crear una Aplicació de Windows nova amb l'entorn de programació Microsoft Visual Studio versió 2005.

Amb això el que s'aconsegueix és que el propi entorn crea un formulari[2] base amb el que es pot treballar directament sense la necessitat d'haver de crear-lo un mateix. Gràcies a les propietats de les que disposa, a aquest formulari se li poden canviar les dimensions, el color, el títol, etc.

El programa però, no només és aquest formulari, sinó que s'han hagut de crear altres classes, que són les que conformen els apartats 3.6., 3.7. i 3.8..

El que s'obté amb això és que amb la classe *Form1* es conforma la interfície gràfica amb tots els seus controls i pestanyes. Les altres classes, per la seva part, donen funcionalitat a cadascun d'aquests components.

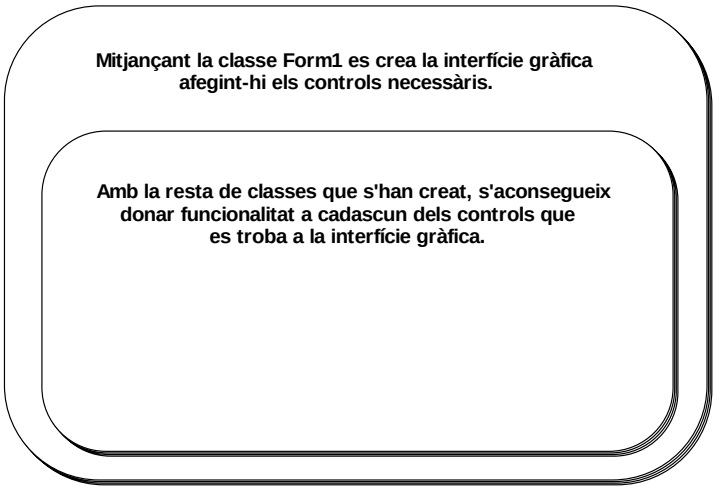


Diagrama 3.3.1 - Diagrama utilització de les diferents classes.

[2] El formulari és la interfície gràfica.

Degut al fet que s'ha considerat que posar el codi font que s'ha creat per portar a terme aquest projecte, a la memòria no era del tot apropiat, s'han creat dos apartats on aquest és descrit.

El primer dels quals és l'apartat 3.4., on s'explica pestanya a pestanya les classes i els mètodes que s'han utilitzat per poder realitzar cada part que les conforma.

L'apartat 3.5. per la seva part descriu les diferents classes que s'han citat a l'apartat 3.4., que son les que s'han creat per poder portar a terme tot el que el projecte requeria.

3.4. Ús de les classes a la interfície gràfica

En aquest apartat s'explica com i on s'utilitza cada classe a la interfície gràfica. Degut al fet que aquesta està dividida en pestanyes, s'explica independentment per cadascuna d'elles.

Pestanya de Marcatge

En el diagrama 3.4.1. es pot veure l'esquelet de la pestanya, on dins de cada control es localitzen les classes i els mètodes que s'han utilitzat per poder-los donar la funcionalitat desitjada.

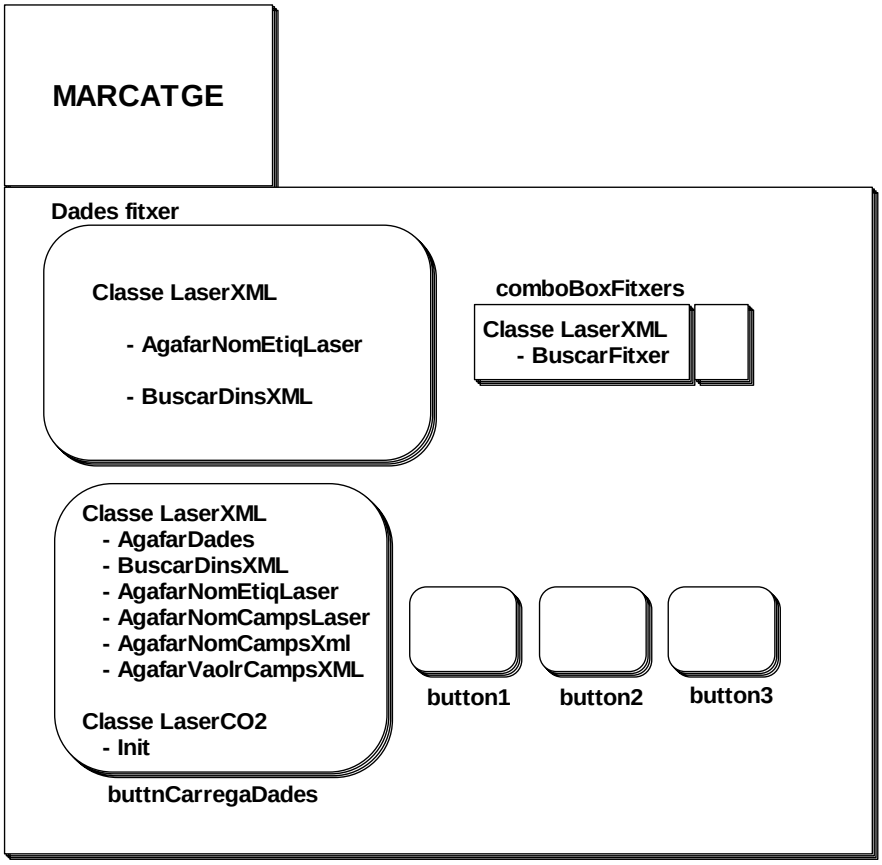


Diagrama 3.4.1 - Diagrama de les classes que s'utilitzen a la pestanya Marcatge.

En el moment que l'operari es disposa a fer el marcatge corresponent a una ordre de treball, el primer que fa és seleccionar el fitxer que correspon a aquella ordre de treball. Aquest pas es fa mitjançant el mètode *BuscarFitxer* de la classe *LaserXML*. En el moment en que es selecciona un fitxer, amb els mètodes *AgafarNomEtiqLaser* i *BuscarDinsXML* de la classe *LaserXML* es mostren les dades més importants del fitxer als seus respectius controls. D'aquesta manera, quan l'operari té clar el fitxer que vol tractar, s'envien les dades al làser.

Per fer-ho utilitza el botó "buttnCarregaDades" que en el moment de ser activat s'utilitzen tots els mètodes de la classe *LaserXML* que es poden veure al diagrama 3.4.1. per recollir les dades necessàries. En el moment en que es disposa de totes elles, amb el mètode *Init* de la classe *LaserCO2* s'envien al làser.

Els tres controls "button" que queden, no utilitzen cap de les classes creades, sinó que només posen en marxa o paren el temporitzador.

Pestanya Control de l'aplicació

Per aquesta pestanya també s'ha creat un diagrama amb el seu esquelet per detectar amb més facilitat les classes i mètodes que s'utilitzen.

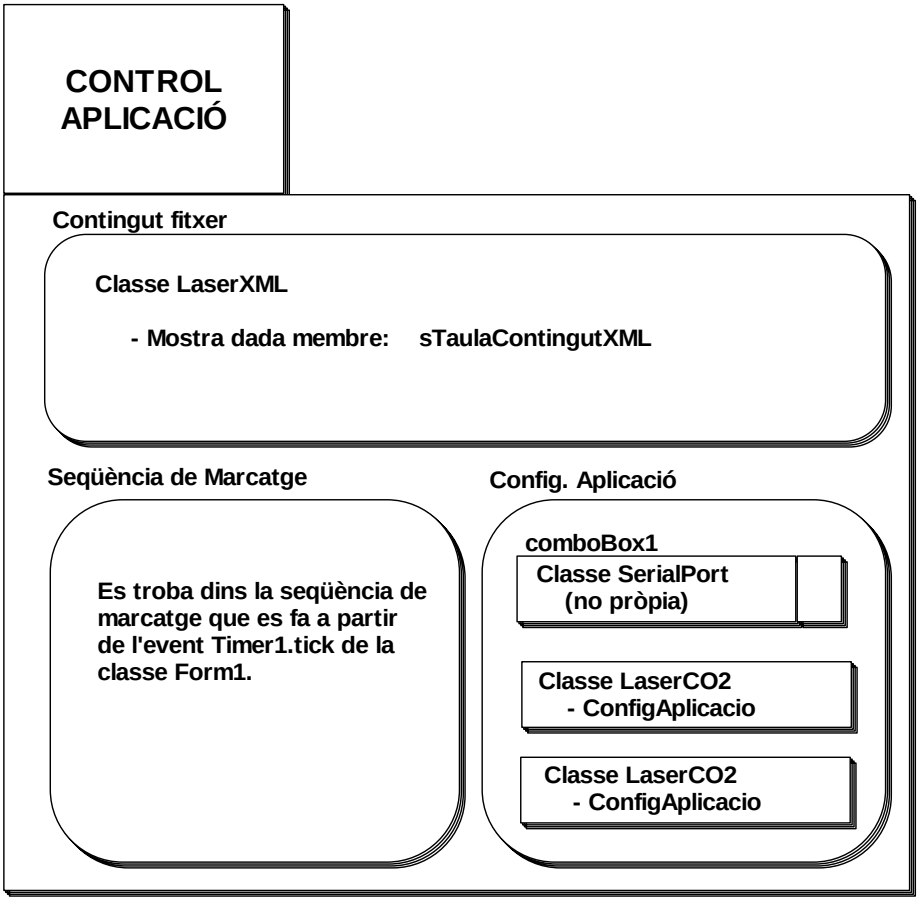


Diagrama 3.4.2 - Diagrama de les classes que s'utilitzen a la pestanya Control Aplicació.

Com ja s'ha descrit a l'apartat 3.2.2. aquesta pestanya també es troba dividida en diverses parts. A la superior el que fa el codi és mostrar la dada membre *sTaulaContingutXML* de la classe *LaserXML*, la qual ha estat omplerta pel mètode *AgafarDades* d'aquesta mateixa classe.

La part inferior esquerra és el lloc on s'informa de l'estat de la seqüència. Com és lògic però, només es fa mentre aquesta està engegada, cosa que passa dins l'event *Timer1.tick* de la classe *Form1*. La part inferior dreta consta de tres elements, els quals s'omplen en l'event *Form1_Load* de la classe *Form1*. Perquè en aquest event es puguin mostrar les dades correctes, prèviament, amb el mètode *ConfigAplicació* de la classe *LaserCO2* s'han recollit aquestes dades.

A aquesta mateixa part inferior dreta, si la persona que s'encarrega de la configuració obre el desplegable, per mostrar-ne el contingut s'utilitza el mètode *GetPortNames* de la classe *SerialPort*. Aquesta classe però, no ha estat creada en aquest projecte.

Pestanya Configuració de Caixes

Finalment, per l'última pestanya de l'aplicació també s'ha creat un diagrama per detectar més fàcilment la manera amb la que s'ha creat.

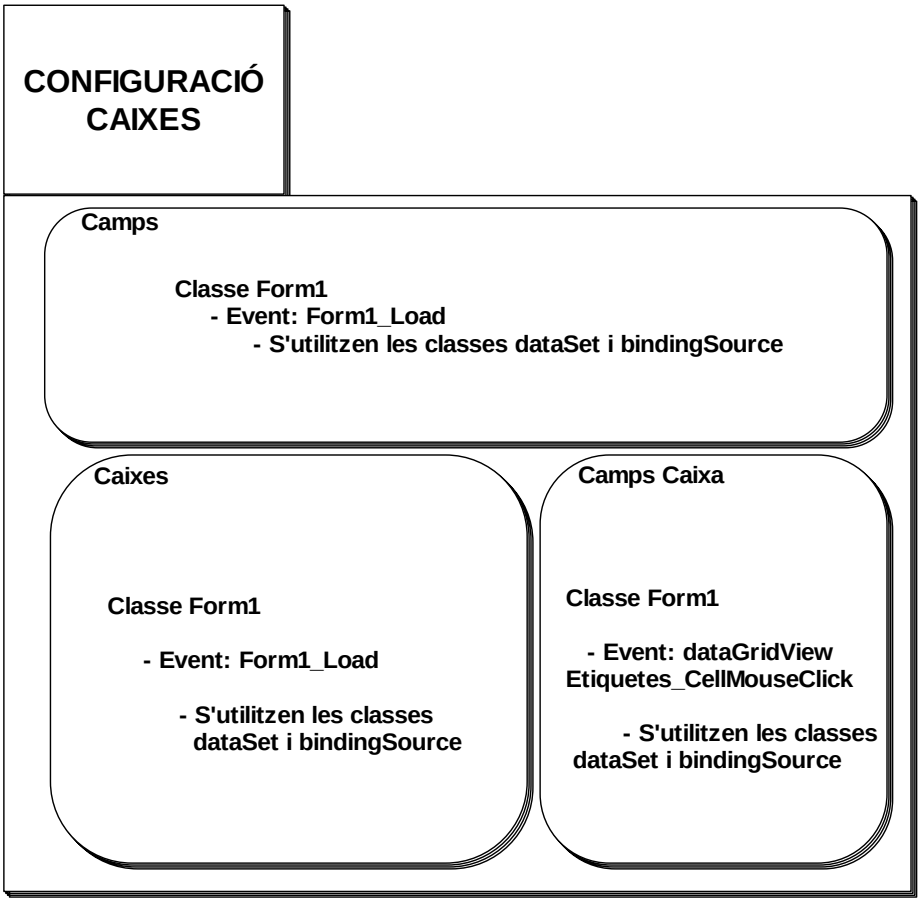


Diagrama 3.4.3 - Diagrama de les classes que s'utilitzen a la pestanya Configuració Caixes.

La part superior i la part inferior esquerra d'aquesta pestanya s'utilitzen els mateixos mètodes de les mateixes classes, amb la diferència però, que a cadascuna

s'hi mostra una taula diferent. Per fer-ho s'utilitzen mètodes de les classes dataSet i bindingSource, que es localitzen a l'event *Form1_Load* de la classe *Form1*.

A la part inferior dreta s'utilitzen els mateixos mètodes que a les altres dues parts, les classes dels quals no han estat creades per aquest projecte. La diferència però, és que l'event que fa mostrar aquesta taula és un altre, el dataGridViewEtiquetes_CellMouseClick.

3.5. Classe Form1

Aquesta classe és la classe principal i és on totes les altres classes acaben treballant, ja que és la que conforma el formulari de l'aplicació. Aquest fet fa que sigui diferent a les altres.

És una classe parcial de la classe *Form* cosa que significa que no tot el seu contingut es troba en el fitxer principal d'aquest projecte. Tota la informació que fa referència als controls que s'hi van afegint es troba en altres fitxers, així com la classe *Program* que conté la funció *Main.*

Tots aquests altres fitxers els genera de manera automàtica el propi entorn de programació i per aquest fet no són explicats a la memòria d'aquest projecte.

3.5.1. Dades membre de la classe Form1

- Definició d'un objecte de la classe *advDIO.*

- Variable de tipus enter que és la discriminatòria de la insrucció *switch* que hi ha a l'apartat 3.5.2.14..

- Tres variables de tipus enter que són els comptadors dels casos 1, 2 i 6 de l'apartat 3.5.2.14..

- Variable de tipus enter necessària per les interrupcions.

3.5.2. Events de la classe Form1

3.5.2.1. Form1

Secció de codi generat automàticament per l'entorn que s'executa en el moment en que s'obra el formulari. Serveix per inicialitzar tots els elements i controls que el programador hi ha afegit i d'aquesta manera una vegada carregat es poden visualitzar i utilitzar.

3.5.2.2. Form1_Load

Event que també s'executa en el moment que s'executa el formulari i també serveix per fer inicialitzacions. En aquest cas però, aquest event ha estat creat per la persona que ha creat el projecte per fer les inicialitzacions que necessitava.

Primer de tot es posa el control de text que indica la quantitat de caixes que s'han marcat a zero. D'aquesta manera aquesta variable ja queda inicialitzada i en el moment que es vulgui començar a marcar caixes no caldria tocar-la, tot i que es fa per seguretat.

Seguidament dins d'una instrucció try/catch es fan les següents inicialitzacions:

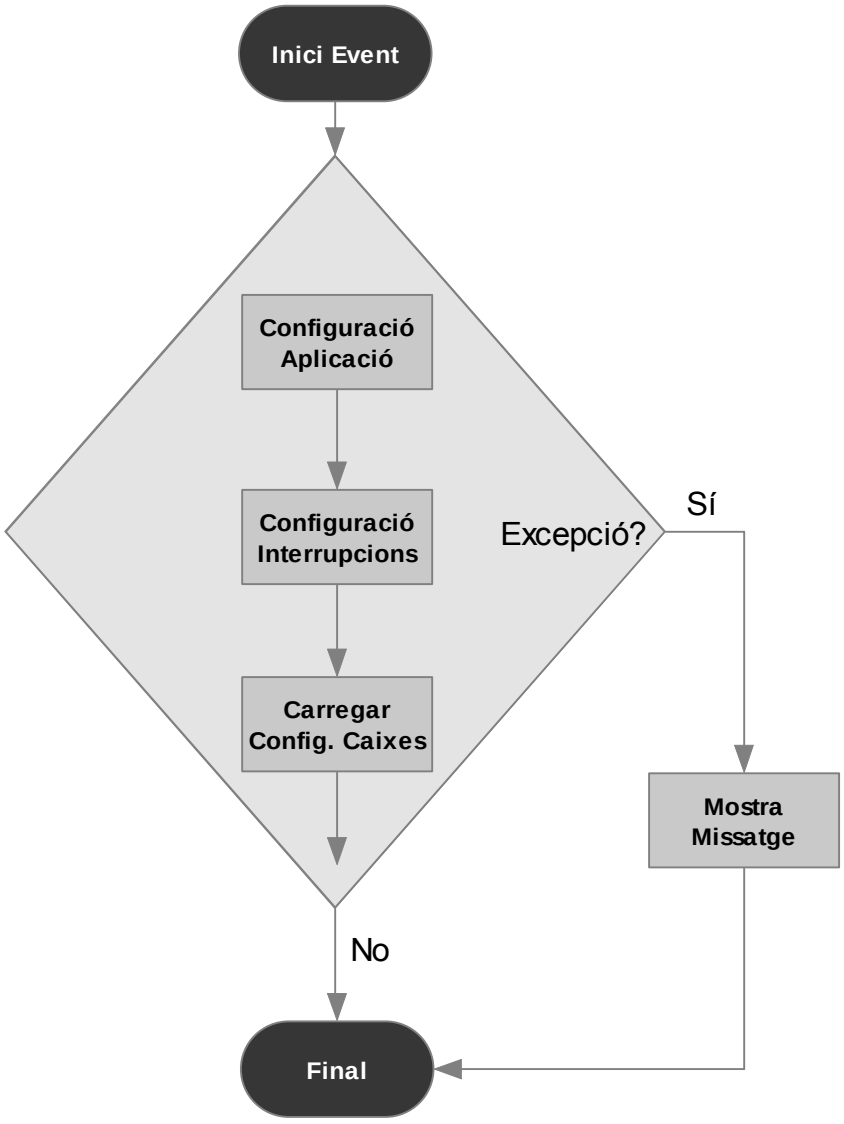


Diagrama 3.5.1 - Diagrama de l'event Form1.Load.

Configuració de l'aplicació

Recollir les dades del fitxer de configuració de l'aplicació perquè l'operari pugui treballar-hi correctament.

Per fer-ho el que es fa és crear un objecte de la classe LaserCO2. Mitjançant el mètode *ConfigAplicacio* d'aquesta mateixa classe, s'omple una taula de cadenes de caràcters que s'ha declarat en aquell mateix moment. Una vegada aquesta taula està inicialitzada, se'n recullen les dades i es col·loquen on han d'anar.

A la primera posició de la taula, que és on hi ha guardat el número de port sèrie amb el que l'aplicació enviarà i rebrà les dades del làser es col·loca al control que permet seleccionar el port.

A la segona posició de la taula s'hi troba el directori on l'aplicació ha d'anar a buscar els fitxers de dades. Aquesta cadena de caràcters es posa al control de text on ha d'anar aquesta informació.

La tercera posició de la taula conté el directori on l'aplicació ha de deixar els fitxers de dades una vegada la informació ha estat tractada i marcada a les caixes. Com l'anterior, també es posa al control de text pertinent.

Tant el control on hi ha la informació del número del port sèrie, com les dues rutes de directori, es localitzen a la segona pestanya de la interfície gràfica dins d'un control "*groupBox*" amb el text "Configuració de l'aplicació".

Configuració de les interrupcions

La següent inicialització que es fa és la de les interrupcions. Mitjançant l'entorn de programació es crea un objecte de la classe *AdvDIO.* Aquest objecte permet crear una interrupció cada vegada que hi ha un canvi de flanc en una entrada determinada. Aquesta inicialització es fa a partir de quatre mètodes d'aquesta classe, i són els següents:

1) Assignar com a interrupció un canal del port, fet que es fa amb el mètode *AssignDiInterruptDiScanRange.* Se li passen tres paràmetres, el primer és el canal o valor de l'entrada de la targeta d'entrades sortides. El segon és el port d'aquesta targeta, que com que només n'hi ha un aquest és el zero, i el tercer és el rang de comptatge i se li ha de passar un 1.

2) Habilitar la interrupció d'una entrada digital amb el mètode *EnableDiInterrupt.* A aquest mètode cal passar-li el canal que es desitja habilitar i un boleà que depenent d'ell s'habilita o es deshabilita la interrupció. En aquest cas, com que es vol habilitar s'hi passa *true*.

3) Assignar si la interrupció serà per flanc de pujada, de baixada o en ambdos casos. Aquesta informació s'assigna mitjançant el mètode *SetDiInterrupt,* al que se li ha de passar el valor del canal i un valor que indica el flanc que s'ha de detectar. En aquest moment, les necessitats de l'aplicació precisen de flanc de baixada.

4) Com a pas final el que s'ha de fer és habilitar l'event igualant la propietat *EventEnabled* a *true.* D'aquesta manera, a partir d'aquest moment, quan detecti que al canal seleccionat hi ha un flanc de baixada s'executarà l'event *OnDiInterrupt.*

Mostrar informació de configuració de les caixes

Per últim, la inicialització que es fa és la de carregar la informació que hi ha al fitxer de configuració de caixes a l'última pestanya de la interfície gràfica.

Mitjançant l'entorn de programació es crea un objecte de la classe *dataSet.* Aquest objecte permet llegir fitxers en format xml, i guardar en diferents taules la informació que conté. Aquest pas es simplifica al fet de cridar el mètode *ReadXml* de la classe *dataSet* passant-li el nom del fitxer com a paràmetre. Una vegada fet aquest pas, l'interior de l'objecte *dataSet* es divideix en diferents taules, cadascuna de les quals amb la informació pertinent.

En aquest cas, la quantitat de taules és tres, una per les caixes, una pels camps i una que conté la informació dels camps que conté cada caixa. Per fer la unió entre aquestes taules i els tres controls *dataGridView* que serveixen per visualitzar-les es necessiten tres controls *bindingSource*.

Al primer dels tres, se li assigna com a origen de dades la taula dels camps de l'objecte *dataSet.* Aquest pas es fa amb la propietat *DataMember* del control *bindingSource.*

Una vegada feta aquesta unió, només falta unir el control *dataGridView* amb el que es volen veure els camps, amb el control *bindingSource.* Per fer-ho s'utilitza la propietat *DataSource* del control *dataGridView* que s'iguala al control *bindingSource.*

Per visualitzar les caixes es fa exactament igual, amb la diferència que al segon control *bindingSource,* com a dades d'origen se li ha de posar la taula de les caixes de l'objecte *dataSet*. Al control *dataGridView* on s'han de veure les caixes, la propietat *DataSource* s'iguala al segon control *dataSource.*

Les unions entre caixes i camps, al ser la visualització una mica diferent, es configuren a l'event *dataGridViewEtiquetes_CellMouseClick.*

3.5.2.3. Form1_FormClosing

Event que s'executa en el moment que es tanca l'aplicació. Ha estat creat amb l'única finalitat de guardar els canvis que hi pugui haver a la part de configuració de l'aplicació realitzats per la persona encarregada del manteniment del sistema. D'aquesta manera, si el directori al que es troben els fitxers de dades o el port sèrie amb el que es treballa varien, la següent vegada que s'executi l'aplicació l'operari no haurà de tornar-ho a configurar, ja que es carregarà tal com estava quan es va tancar.

Per poder fer això el que es fa és crear un objecte de la classe *StreamWriter* passant-li per paràmetre el nom del fitxer que es vol escriure. Amb el mètode *WriteLine* d'aquesta classe s'escriu a la primera línia del fitxer el contingut del control *comboBox* dels ports sèrie. Executant una altra vegada aquest mateix mètode s'escriu, a la

segona línia del fitxer, el text que hi ha al control que conté la ruta dels fitxers de dades. Finalment, també amb el mateix mètode a la tercera línia s'hi escriu el text que hi ha dins el control de text que conté la ruta on s'han de traspassar els fitxers una vegada tractats.

3.5.2.4. button1_Click

És l'event que s'executa quan es fa un clic amb el ratolí sobre aquest botó. Pel codi que s'hi ha posat, aquest event és el que posa la seqüència de marcatge en marxa.

Com que aquesta seqüència funciona a partir d'un objecte de la classe *timer*, el primer que es fa és engegar-lo per mitjà del mètode *Start*. Després d'això es posa el text del control que controla la quantitat de caixes marcades a zero perquè amb aquest botó s'engega la seqüència des de l'inici.

Un altre pas que es fa és posar de color verd un indicador per informar a l'operari que la seqüència es troba en marxa.

3.5.2.5. button2_Click

Igual que l'últim event, aquest també s'executa en el moment que es fa un clic sobre un botó.

Aquest event és l'encarregat de parar la seqüència de marcatge de les caixes. Per parar-la, l'única cosa que s'ha de fer és parar l'objecte de la classe *timer* mitjançant el mètode *Stop.* Una vegada fet això, es posa l'indicador que a l'engegar la seqüència s'ha posat a verd, a vermell, indicant així que la seqüència de marcatge està parada.

3.5.2.6. button3_Click

Com el nom indica, aquest event també s'executa en el moment que es fa un clic sobre aquest botó número tres.

Aquest botó serveix per si l'operari evita que alguna cosa falli parant la seqüència de marcatge, que quan vulgui tornar-la a engegar continuï al punt on l'ha parat. L'altre botó d'engegar posa el comptador de caixes marcades a zero, aquest en canvi no ho fa. Per això, el que fa aquest event només és engegar l'objecte *timer* i posar l'indicador a verd indicant que la seqüència torna estar engegada. Com que no es toca cap variable, totes continuen tenint el valor que tenien al moment de parar-lo, i d'aquesta manera tot continua al punt on estava.

3.5.2.7. buttnGuardarConfig_Click

Aquest event també s'executa en el moment que es fa un clic sobre un botó i serveix per guardar les modificacions que es facin a l'última pestanya de la interfície gràfica. Allà és on hi ha tota la informació relacionada amb la configuració de les caixes i si alguna cosa es modifica, s'ha de poder guardar.

Per fer-ho, degut a que quan es modifica algun dels tres controls *dataGridView*, s'està modificant l'objecte de la classe *dataSet*, l'únic que cal fer per guardar els canvis és guardar aquest objecte. Amb el mètode *WriteXml* de la classe *dataSet* s'escriu el

nou i modificat objecte al mateix fitxer de configuració d'on s'havia extret la informació en el moment d'executar l'aplicació. D'aquesta manera, al fitxer de configuració de caixes hi queda l'última informació que s'hi ha guardat, que és el que interessa.

3.5.2.8. comboBox1_MouseClick

Un control *comboBox* és un control desplegable que permet visualitzar una sèrie d'elements i si es desitja, seleccionar-ne un. En aquest cas, quan es fa un clic damunt seu amb el ratolí, es desplega mostrant tots els ports sèrie que té disponibles l'ordinador amb el que està sent executada l'aplicació.

Per poder-ho fer primer de tot es declara una taula de cadenes de caràcters que amb el mètode *GetPortNames* de la classe *SerialPort* s'omple amb els noms dels ports sèrie que aquell ordinador té. Una vegada està la taula plena, amb el mètode *Clear*, de la propietat *Items* del control *comboBox* es neteja la llista d'elements que té aquest control. A priori, aquest fet d'esborrar els elements no té gaire sentit, però si no s'hi posa i l'operari desplega més d'una vegada el control, a aquest hi apareixen els elements repetits tantes vegades com aquest ha estat desplegat. Per aquest motiu, cal posar-hi aquesta línia d'instrucció.

Finalment, mitjançant el mètode *AddRange* de la propietat *Items* de la classe *comboBox* s'afegeix la taula de cadenes de caràcters a la llista i d'aquesta manera es poden visualitzar els diferents ports.

3.5.2.9. comboBoxFitxers_MouseDown

Event encarregat de mostrar els fitxers de dades pendents de ser tractats, al control "comboBoxFitxers".

El primer que es fa és crear un objecte de la classe *LaserXML*. Després, igual que es fa en l'altre control desplegable de la interfície gràfica, amb el mètode *Clear*, de la propietat *Items* del control *comboBox* es neteja la llista d'elements que podria tenir el control. Finalment s'omple aquesta llista amb el mètode *BuscarFitxer* de la classe *LaserXML* i amb el mètode *AddRange* de la propietat *Items* de la classe *comboBox* s'afegeix la llista al control. Al mètode *BuscarFitxer* se li passa el text que hi ha al control de text de la part de configuració de l'aplicació, que és on es troba el directori on hi ha els fitxers de dades.

3.5.2.10.comboBoxFitxers_SelectedValueChanged

Aquest event és el que s'executa cada vegada que es selecciona un fitxer al control *comboBox* que mostra els fitxers de dades. Per fer que l'operari tingui més informació a l'hora d'escollir entre un dels fitxers de dades que té disponibles, el que es fa en aquest event és mostrar les dades més importants de cada fitxer que es seleccioni.

Per fer-ho, degut a que els mètodes que tracten amb les dades dels fitxers de dades són de la classe *LaserXML*, es crea un objecte d'aquesta classe. Seguidament, amb el mètode *AgafarDades* es carrega tot el contingut del fitxer dins la taula corresponent. Una vegada carregada, mitjançant el mètode *BuscarDinsXML* s'omplen

els controls "*textBox*" amb la informació corresponent al camp que s'ha passat al mètode.

3.5.2.11.dataGridViewEtiquetes_CellMouseClick

Aquest event és el que s'executa en el moment en que es fa un clic amb el ratolí sobre una cel·la del control "*dataGridView*" que mostra cadascuna de les caixes. El que s'ha d'aconseguir fent això és que al tercer control "*dataGridView*" es vegi la relació dels camps que cada caixa té configurats.

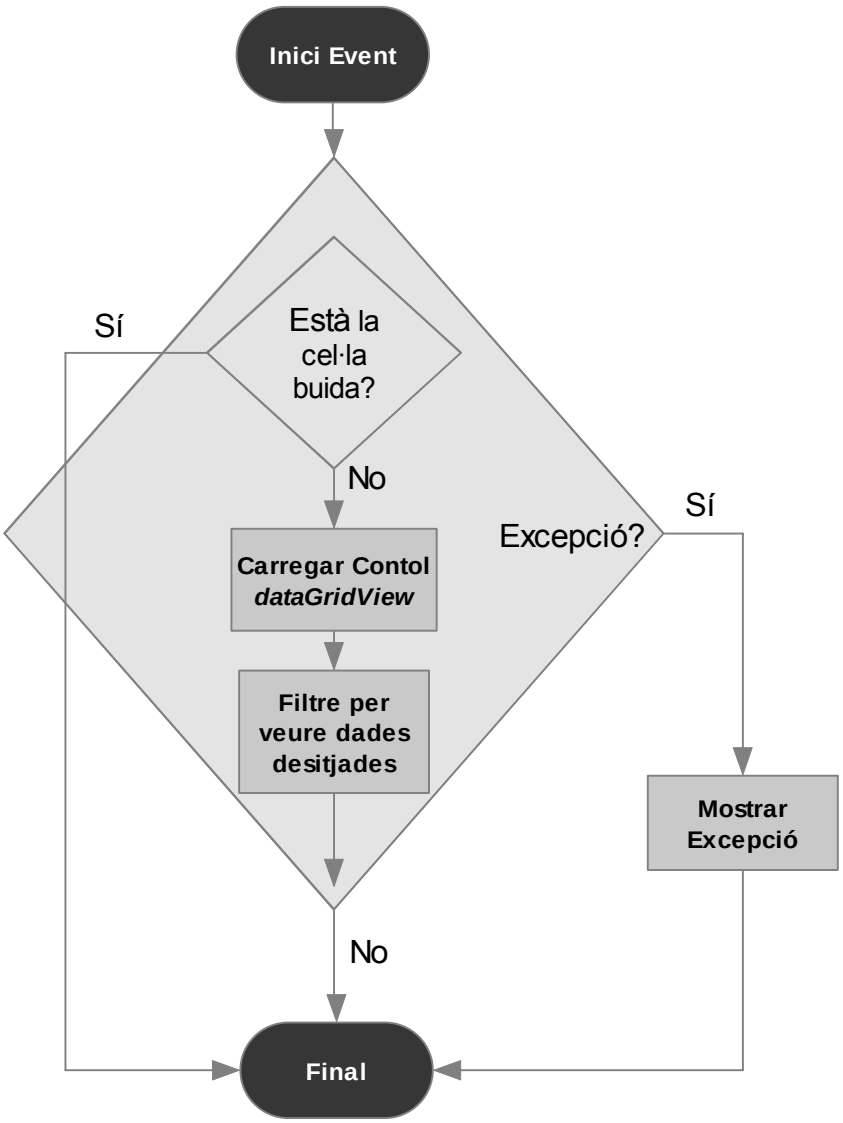


Diagrama 3.5.2 - Diagrama de funcionament de l'event *dataGridViewEtiquetes_CellMouseClick.*

Per fer això es segueix el mateix procés que per visualitar les dades dels altres controls *dataGridView.* En aquest però, hi ha una petita diferència, que és que només s'han de visualitzar els camps que te configurats la caixa sobre la que s'ha fet el clic, en comptes de tota la taula.

Com s'acaba de comentar, la primera part és igual a les altres i consta de les següents instruccions. Primer, la propietat *DataMember* de l'objecte de la classe *bindingSource* s'iguala a la taula de l'objecte de la classe *dataSet* que conté la informació. Una vegada fet això, la propietat *DataSource* del control "*dataGridView*" s'iguala a l'objecte de la classe *bindingSource* que s'acaba d'inicialitzar.

Amb això s'aconsegueix el mateix que als altres dos controls "*dataGridView*", veure tota la taula que hi ha dins l'objecte *dataSet.* Per aquest motiu s'ha de fer un filtre que només deixi veure el que interessa.

El filtre consisteix en afegir al control, totes aquelles files que a la taula que conté els camps de les caixes, hi ha el mateix identificador sobre el que s'ha fet el clic.

El problema que tenia aquest event és que l'última fila d'aquest control no hi ha text, per tant, quan s'hi fa un click no té què buscar. Per solucionar aquest fet es va posar una condició *if* que fa que si la cel·la està buida, la part de codi que hi ha dins l'event no s'executa i en surt sense fer res.

Totes les instruccions que es troben dins d'aquest event estan dins d'una instrucció try/catch, ja que és bastant possible que hi hagi excepcions durant el procés que d'aquesta manera es poden controlar.

3.5.2.12.buttnCarregaDades_Click

Aquest event s'executa en el moment de prémer el botó encarregat de recollir totes les dades necessàries. Depenent de quines dades són, les envia al làser o les col·loca a la seva posició.

Per poder-ho fer cal crear dos objectes, un de la classe *LaserXML* i l'altre de la classe *LaserCO2.*

Una vegada creats, amb el mètode *AgafarDades* de la classe *LaserXML* es carreguen les dades del fitxer de dades a una taula que serà amb la que es treballarà a partir d'aquest moment. Per aconseguir-ho se li passa el text que hi ha al control "*comboBoxFitxers*", que és on l'operari ha seleccionat el fitxer. A més a més, també se li passa el text del control "*txtAdreça*" que és on hi ha la ruta del directori on es troben els fitxers de dades.

Amb la taula que conté la informació del fitxer de dades plena, ja es pot visualitzar al control "*listBox*" que es troba a la segona pestanya de la interfície gràfica. Abans d'omplir-lo amb el contingut de la taula però, cal buidar-lo mitjançant el mètode *Clear* per si anteriorment aquest pas ja s'hagués fet amb un altre fitxer. D'aquesta manera, passant la taula al mètode *AddRange*, el contingut de la taula apareix al control "*listbox*".

Seguidament, amb el mètode *BuscarDinsXML* de la classe *LaserXML* es busca dins la taula anterior la quantitat de caixes que s'han de marcar i es posa al control "*textBox*" on es localitza aquesta informació.

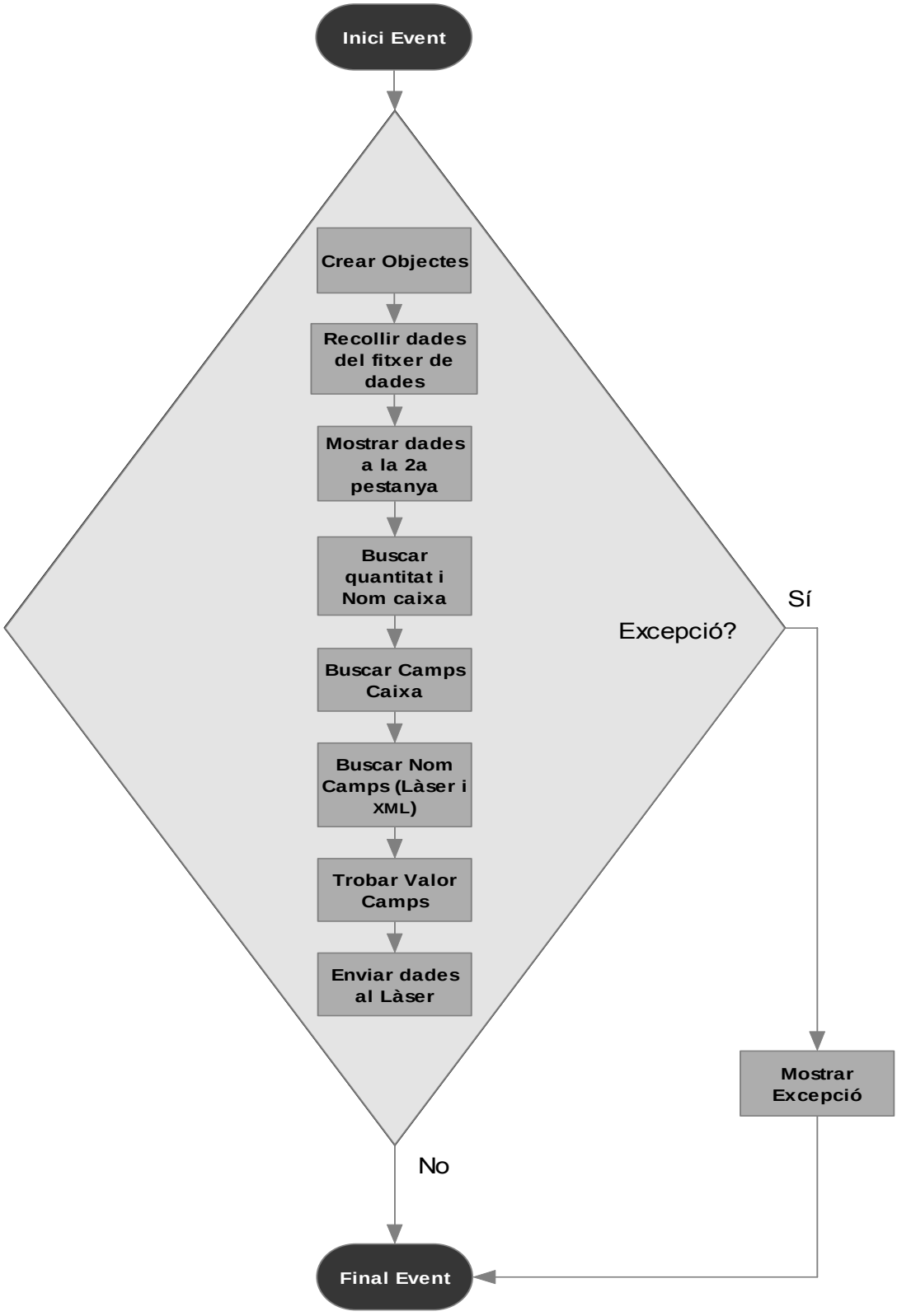


Diagrama 3.5.3 - Diagrama de funcionament de buttnCarregaDades_Click.

El següent pas consisteix en buscar els camps que la caixa té configurat que se li marquin. Per fer-ho primer es busca el nom de la caixa a la taula anterior, mitjançant el mètode *BuscarDinsXML* de la classe *LaserXML.* Una vegada ja es té aquesta informació, es passa per paràmetre juntament amb l'objecte de la classe *dataSet* al

mètode *BuscarCampsConfig*. Aquest mètode omple una taula dada membre de la classe *LaserXML* amb els camps que aquella caixa té configurats.

La següent dada que es necessita és el nom amb el que la caixa està guardada a la memòria interna del làser, informació que es troba a l'objecte de la classe *dataSet.* Per aconseguir-la, es torna a buscar el nom de la caixa a la taula de dades de la mateixa manera que a la instrucció anterior. Aquesta vegada però, es passa com a paràmetre al mètode *AgafarNomEtiqLaser* que també és de la classe *LaserXML.* En el moment en que retorna el nom que es buscava, aquest és posat al control de text on es localitza aquesta informació.

Una vegada es disposa de tota la informació, s'han d'omplir les dues taules que contenen els noms amb els que els camps es troben tant al làser com al fitxer de les dades. Per omplir la que contindrà els noms que estan al làser es fa amb el mètode *AgafarNomsCampsLaser* de la classe *LaserXML*, passant-li com a paràmetre l'objecte de la classe *dataSet.* Per omplir l'altra taula dada membre amb els noms que els camps estan guardats al fitxer de dades es fa amb el mètode *AgafarNomsCampsXML*, al que també se li ha de passar l'objecte de la classe *dataSet* per paràmetre.

L'últim pas de la recollida de dades consisteix en trobar i guardar el valor que tenen aquests camps, que serà el que es marcarà a les caixes. Per aconseguir-ho només cal cridar el mètode *AgafarValorCampsXML* de la classe *LaserXML*. Degut al fet que les dades que aquest mètode necessita es troben a les dades membre, no cal passar-li cap variable. Tampoc retorna res ja que la informació recollida també es guarda a una dada membre. En el moment que tota la informació necessària s'ha recollit, el que es fa és buidar tots els controls de text que informen de l'estat de la seqüència de marcatge. Aquests són els que hi ha dins el "groupBox" amb el nom "Seqüència de Marcatge" de la pestanya Control de l'aplicació.

Finalment, l'única cosa que falta fer és enviar les dades recollides al làser i per fer-ho s'utilitza el mètode *Init* de la classe *LaserCO2*, al que se li han de passar quatre paràmetres. El nom del port sèrie pel que ha d'enviar i rebre les dades. El nom amb el que la caixa que ha de marcar està guardat a la seva memòria interna. Una taula amb els noms que els camps estan guardats a la seva memòria i per últim, els valors d'aquests camps.

Com a primer paràmetre se li passa el text que hi ha al control "comboBox" que conté el port sèrie. Pel segon se li passa el text que hi ha al control al que s'hi ha posat el nom que la caixa té al làser i pel tercer i quart paràmetre se li passen dues taules dada membre de la classe *LaserXML.* Aquestes taules contenen els noms amb els que els camps estan guardats al làser i els seus valors.

Per acabar, per informar a l'operari que la recerca i transferència de dades ha estat correcte es posa el botó que hi ha sota aquest que s'està explicant de color verd.

3.5.2.13.OnDiInterrupt

Aquest event és el que s'executa cada vegada que l'objecte de la classe *advDIO* informa que hi ha hagut una interrupció a l'entrada configurada.

El codi que conté aquest event és molt simple ja que l'única cosa que fa és incrementar una variable que inicialment es troba a zero. D'aquesta manera per saber si hi ha hagut una interrupció o no només cal mirar el valor d'aquesta variable. Si encara està a zero vol dir que no hi ha hagut el flanc desitjat, mentre que si és major a zero significa que ha passat per aquí degut al flanc.

3.5.2.14.Timer1.Tick

Per mitjà de l'entorn de programació es crea un objecte de la classe *Timer* que s'activa cada 100ms com a mínim.

Aquest event s'ha creat per introduir-hi una espècie de màquina d'estats de la seqüència de marcatge i depenent de l'estat en el que es troba la seqüència fa una cosa o en fa una altra.

Aquesta màquina d'estats s'ha creat a partir d'una instrucció *switch* que depenent de la variable *iEstat* entra a un cas o en un altre.

Dins d'aquest event es crea un altre objecte de la classe *advDIO* que és amb el que es llegeix el valor de les diferents entrades de la targeta d'entrades i sortides digitals.

Tots els casos que hi ha tenen la mateixa estructura bàsica, que és que per mitjà d'una condició *if* es comprova l'estat del làser. Si aquest està correcte i no informa de cap error es fa el que s'ha de fer en aquell cas, pel contrari, si el làser informa d'error es para la seqüència i s'informa a l'operari que el làser té un error.

Cas 1

Aquest primer cas consisteix en comprovar que la màquina no té cap error i es troba apunt per marcar. Alguna situació en la que la màquina no donaria la senyal correcte podria ser, per exemple, que l'operari s'hagués deixat la porta oberta o que en aquell moment la màquina està retirant una caixa marcada.

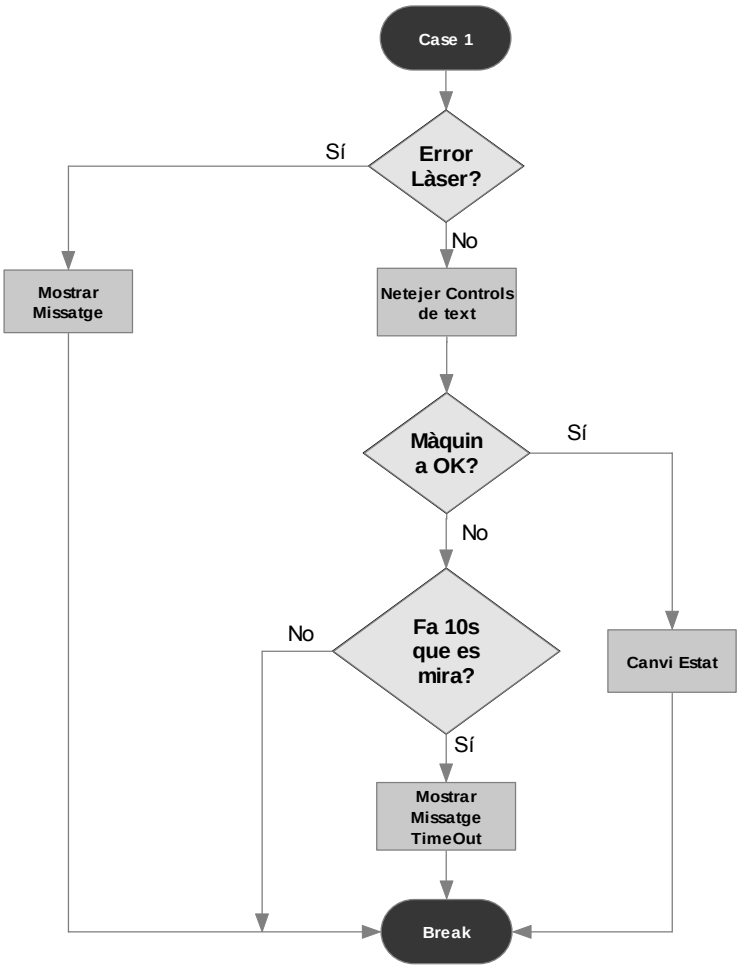


Diagrama 3.5.4 - Diagrama de funcionament del cas 1.

Aquest procés de comprovació de l'estat de la màquina es porta a terme per mitjà d'una condició *if*. Amb el mètode *ReadDiChannel* de la classe *advDIO* es llegeix el valor de l'entrada i si el nivell al que està significa que està apunt es canvia el valor de la variable de la instrucció *switch.*

Pel contrari, si no està apunt s'entra a una altra condició *if* que per mitjà d'un comptador, si fa gaire estona que la màquina no està apunt para l'objecte *Timer* i informa a l'operari que hi ha hagut un error de temps a la màquina. En aquest cas però, la variable de la instrucció *switch* no varia i continua estant a 1.

Cas 2

El segon cas és molt semblant al primer ja que el que fa és comprovar si el làser està apunt per marcar o no. Normalment, si no hi ha incidents l'únic moment de la seqüència que el làser informa que no està apunt és mentre està marcant.

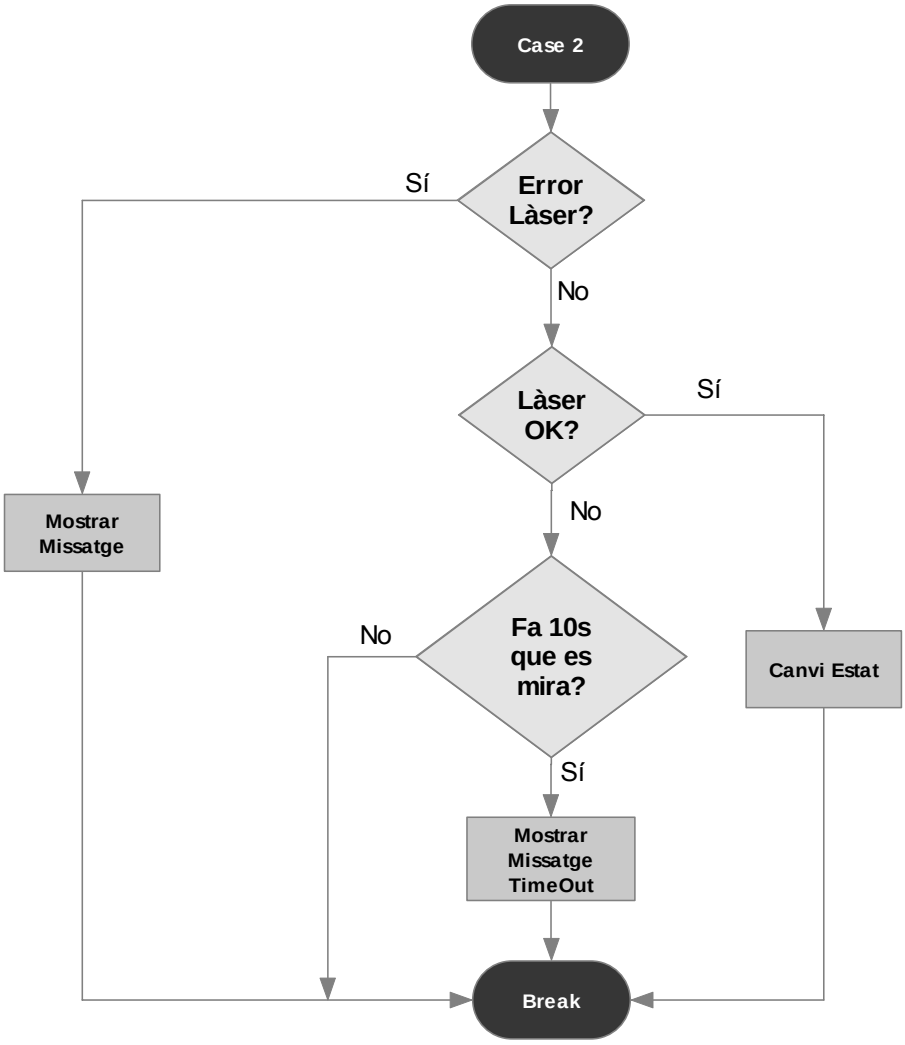


Diagrama 3.5.5 - Diagrama de funcionament del cas 2.

Per comprovar l'estat al que es troba el làser també es fa amb el mètode *ReadDiChannel* de la classe *advDIO*. Si es rep el nivell que significa que el làser està apunt, s'entra dins la condició i es canvia el valor de la variable de la instrucció *switch*. Si pel contrari no es rep aquest valor, amb un altre comptador es dóna un cert temps a que el làser passi a estar apunt. Si el temps expira, igual que al cas anterior es para l'objecte *Timer* i s'informa a l'operari que hi ha hagut un error de temps al làser i la variable de la instrucció switch tampoc canvia.

Cas 3

Al tercer cas de la seqüència el que es fa és generar un pols de 30ms que faci que el làser marqui el que en aquell moment té configurat.

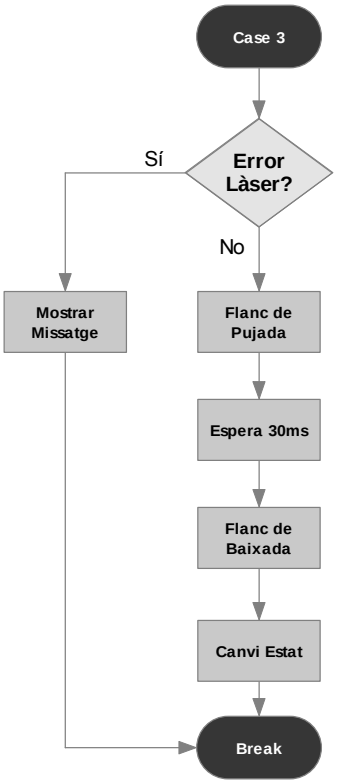


Diagrama 3.5.6 - Diagrama de funcionament del cas 3.

Per mitjà del mètode *WriteDoChannel* de la classe *advDIO* es genera el flanc de pujada del pols. Una vegada fet, l'aplicació s'espera 30ms i llavors crea el flanc de baixada amb el mateix mètode que s'ha creat el de pujada.

Tant bon punt el pols ha estat creat es canvia el valor de la variable de la instrucció *switch* perquè la propera vegada entri al següent cas.

Cas 4

Aquest cas número quatre consisteix en comprovar si el làser ha acabat de fer el marcatge de la caixa i si ha acabat, configurar de nou les interrupcions per poder ser utilitzades en una altra entrada.

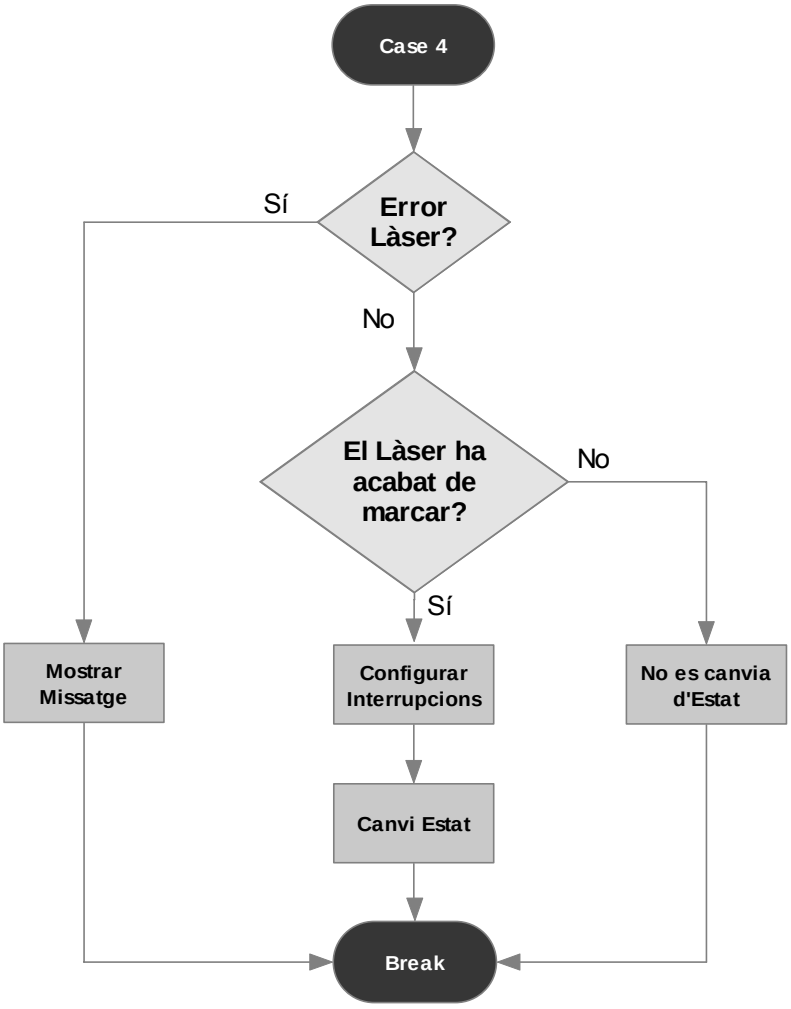


Diagrama 3.5.7 - Diagrama de funcionament del cas 4.

El primer que es fa és mirar a partir d'una condició *if* si hi ha hagut alguna interrupció. Per fer-ho es mira la variable que s'incrementa cada vegada que hi ha una interrupció i si és major que zero significa que hi ha hagut la interrupció.

Si aquesta condició és certa es torna a inicialitzar aquesta variable a zero. Això es fa perquè la propera vegada que es consulti segur que seria més gran que zero i probablement s'estaria cometent en un error. Després d'això es canvia el valor de la variable de la instrucció *switch* perquè no torni a entrar a aquest cas i es reconfiguren les interrupcions perquè funcionin per una altra entrada. Per configurar-les es fa de la

mateixa manera que a l'apartat 3.5.2.2. de l'event *Form1_Load*, però amb la particularització que en aquest punt ja estan funcionant i això fa que primer s'hagin de desactivar i una vegada configurades s'hagin de torna a posar en marxa. Les altres dues coses que varien són el canal d'entrada que es vol detectar el flanc i si ha de ser flanc de pujada o de baixada.

Si pel contrari aquesta condició no fos certa, no es canvia el valor de la variable de la instrucció *switch*, cosa que fa que la propera vegada que es torni a donar aquest event, el programa torni a entrar dina d'aquest cas.

Cas 5

Aquest cas és molt semblant al cas 3 ja que en aquest també es genera un pols. Aquest però és de 100ms i serveix per informar a la màquina que retiri la caixa ja que ja s'ha acabat de marcar.

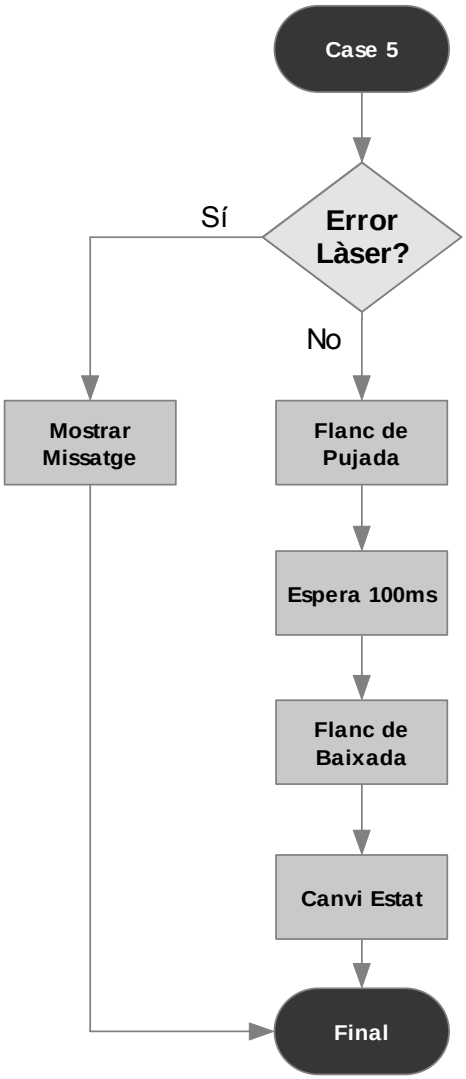


Diagrama 3.5.8 - Diagrama de funcionament del cas 5.

Per fer-ho s'utilitza el mètode *WriteDoChannel* de la classe *advDIO*. Primer es crea el flanc de pujada, llavors s'espera els 100ms i seguidament es crea el flanc de

baixada amb el mateix mètode. Una vegada això ha estat fet, només falta canviar el valor de la variable de la instrucció *switch* perquè quan torni a entrar ho faci al següent cas.

Cas 6

L'últim cas de la seqüència serveix principalment per comprovar que la caixa marcada s'ha extret correctament. També, al ser al final de la seqüència es comprova si la caixa que s'ha retirat és l'última que s'ha de marcar i si així és, s'acaba la seqüència. A més a més d'això, també és on es comprova si s'han acabat les caixes verges.

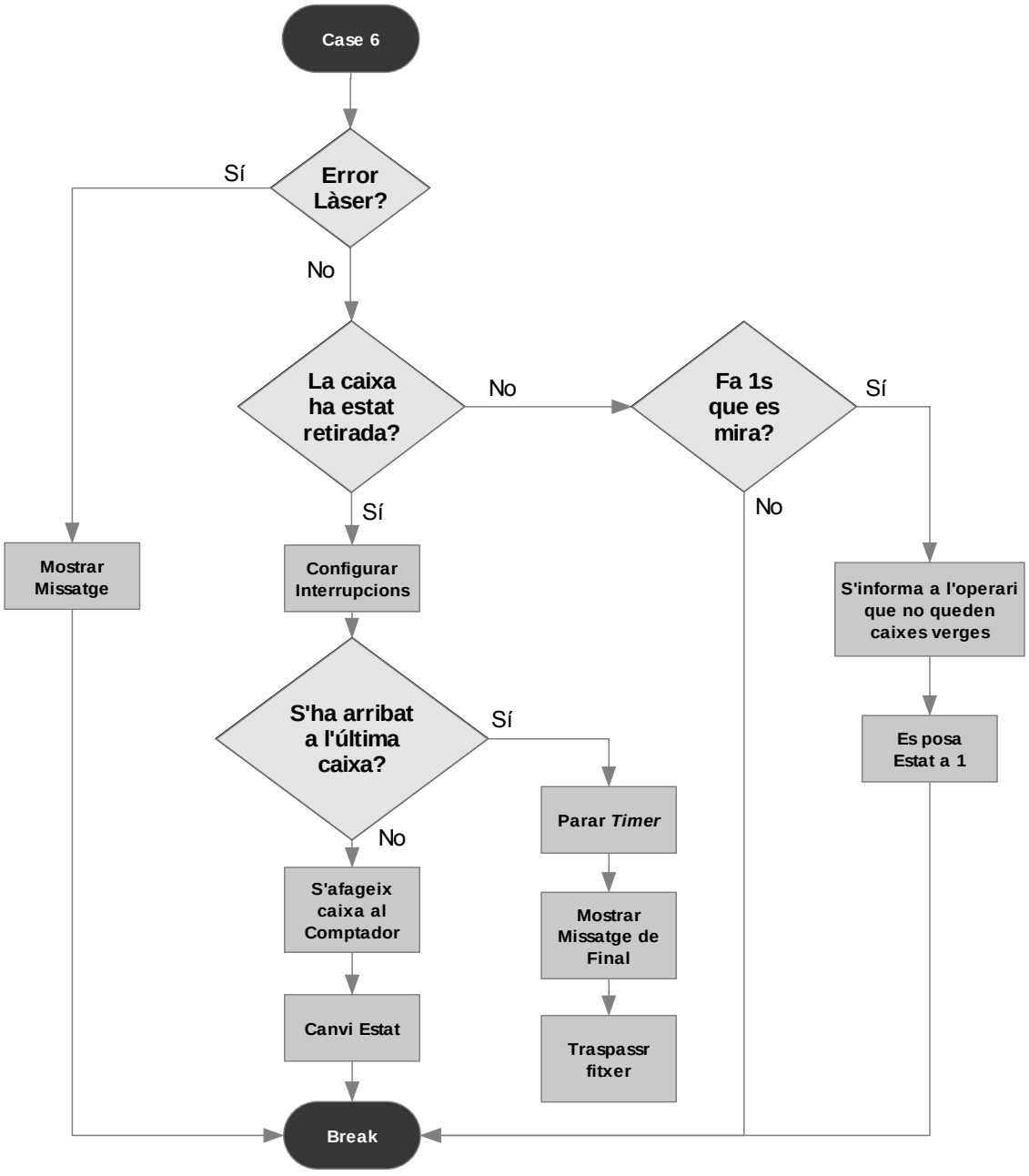


Diagrama 3.5.9 - Diagrama de funcionament del cas 6.

Primerament, mitjançant una condició *if* es comprova si hi ha hagut una interrupció, cosa que significaria que la caixa ha estat retirada correctament. Si no s'ha produït, la variable de la instrucció *switch* no varia per així tornar-ho a comprovar fins que hi sigui.

Si després de comprovar-ho un segon i mig el flanc no arriba, significa que la màquina ha anat a buscar la caixa marcada i ha tornat sense res. En aquest moment es para la seqüència i s'informa a l'operari que ha de posar més caixes verges. En el moment en que l'operari n'ha col·locat més es torna a posar la seqüència en marxa. El que sí que es fa però, és posar la variable de la instrucció *switch* a 1 perquè torni a començar el procés des del principi del marcatge d'una caixa.

Si com és habitual hi ha caixes i apareix la interrupció, el que es fa és comprovar si la quantitat de caixes que s'han marcat és inferior a la quantitat de caixes que s'han de marcar. Si no ho és es para la seqüència amb el mètode *Stop* de la classe *Timer* i es mostra un missatge informant que hi ha un error en la quantitat de les caixes. Pel contrari, si tot va bé i la condició és certa, amb una altra condició *if* es mira si aquesta és l'última caixa i si ho és també es para l'objecte de la classe *Timer* i s'informa que ja s'han marcat totes les caixes. En aquest moment, al ser clar que el marcatge de totes les caixes ha estat correcte, es trespassa el fitxer de dades de directori afegint-hi la data i l'hora.

Si pel contrari, no era l'última caixa, s'afegeix una caixa al comptador de caixes marcades, la variable de la instrucció *switch* es posa a 1 per tornar a començar des del principi amb la següent caixa i es configuren les interrupcions de la mateixa manera que al cas 5.

Finalment, referent a tot l'event, cal dir que el codi que conté es troba dins una instrucció try/catch. Això fa que si hi ha alguna excepció durant el procés la recollirà i la mostrarà per pantalla. També, com tots els errors possibles dins la seqüència, para l'objecte de la classe *Timer* abans de mostrar el missatge.

3.6. Classe LaserCOM

Aquesta classe està creada per poder portar a terme tot el que fa referència al port sèrie. Com ja s'ha comentat, la transferència de dades amb el làser es fa amb aquest tipus de comunicació, per tant, era d'una importància considerable tenir aquest punt ben definit i organitzat.

Per portar a terme aquesta classe, la qual havia de contenir un objecte de la classe "SerialPort", es va haver d'afegir un enllaç a un *workspace* que no contenia el projecte anomenat "System.IO.Ports". Afegint aquest enllaç, l'usuari pot crear un objecte de la classe nombrada anteriorment i els seus mètodes permeten establir una comunicació mitjançant el port sèrie.

3.6.1. Dades membre de la classe LaserCOM

- Objecte de la classe "SerialPort".

- Constant de temps que l'aplicació espera fins a rebre resposta del làser.

- Taula de cadenes de caràcters on es guarda el que contesta el làser.

3.6.2. Mètodes de la classe LaserCOM

3.6.2.1. ObrirPort

És un mètode al que se li passa una variable de tipus byte que conté el número de port sèrie de l'ordinador amb el que es pretén establir la comunicació i no retorna res.

La primera cosa que es fa en aquest mètode és mirar si l'objecte de la classe *SerialPort* està inicialitzat o no i si ho està s'utilitza el mètode *Dispose* per alliberar-lo.

Una vegada està clar que aquest objecte no correspon a res, s'inicialitza amb el número del port que es passa per paràmetre al mètode. A més a més també se li assignen els valors de les dades que necessita per poder ser inicialitzat establint la comunicació desitjada i finalment s'obre.

3.6.2.2. TancarPort

Mètode al que no se li passa res i tampoc retorna res.

El que fa és mirar si l'objecte de la classe *SerialPort* correspon a algo i si no correspon a res, no fa res, però si pel contrari correspon a algun objecte inicialitzat anteriorment, el port es tanca i s'allibera.

3.6.2.3. SendMsg

Mètode al que se li passa una variable de tipus string i que no retorna res.

A partir de l'objecte de la classe *SerialPort* crida el mètode *ReadExisting* per buidar el buffer de recepció del port sèrie i tot seguit amb el mètode *Write* envia el missatge pel port. Aquest missatge és el paràmetre que s'ha passat al mètode. Al final del missatge però, s'hi afageixen els caràcters "\r", ja que el làser ho requereix així.

3.6.2.4. WaitAnswer

És un mètode al que se li passa una variable de tipus int i retorna una taula de cadenes de caràcters.

Aquest mètode permet llegir la informació que el làser envia una vegada la nostra aplicació l'hi ha enviat la informació necessària.

Primer de tot es crea una taula de cadenes de caràcters que serà on es guardarà tota la informació que enviï el làser. Com que a priori no se sap la quantitat de frases que el làser pot enviar, la taula es crea de cap posició i a mesura que vagi arribant cada frase la taula s'anirà allargant una posició.

La variable que s'ha passat a aquest mètode és la dada membre que conté el temps d'espera de resposta i és en aquest punt en el que a l'objecte de la classe *SerialPort* se li assigna aquest temps d'espera. Aquest pas es fa per mitjà del mètode *ReadTimeout* de la mateixa classe.

Seguidament es declara una variable de tipus cadena de caràcters que serà la que es formarà cada vegada que el làser enviï una frase. Amb aquestes cadenes de caràcters s'anirà omplint la taula de cadenes que s'ha definit a l'inici d'aquest mètode.

Una vegada fetes les declaracions de les variables necessàries, comença el procés de lectura del port sèrie. Aquesta lectura es fa caràcter a caràcter i aquest es va afegint a la cadena de caràcters declarada anteriorment. Mentre el caràcter sigui diferent de "\r" o la cadena passada a majúscules sigui diferent de "SL1>" s'aniran llegint caràcters i augmentant la cadena.

Això es fa perquè pel làser els caràcters "\r" signifiquen final de línia i "SL1>"[3] final de comunicació.

Una vegada hagi arribat "\r" o bé "SL1>" el que es fa és analitzar l'última cadena de caràcters rebuda. Es mira el primer caràcter d'aquesta i si aquest és "-" significa que el làser ha enviat un error[4] ja que enviant un nombre negatiu és la manera amb la que el làser informa d'això. En aquest cas el que es fa és passar la cadena de caràcters a número, es busca aquest número a la enumeració dels possibles errors i se n'extreu un missatge informant-ne a l'usuari.

En el cas que la cadena no comenci per un signe negatiu o guió, s'allarga la taula de cadenes de caràcters una posició i s'hi afegeix l'última cadena rebuda a la penúltima posició.

Aquesta seqüència de llegir caràcters amb els que es crea una cadena es va produint mentre no arriba la cadena "SL1>". Una vegada arriba aquesta cadena informant que el làser ja no enviarà res més es deixa d'escoltar el port sèrie, es copia la taula de cadenes de caràcters local a la taula també de cadenes de caràcters que és dada membre i es retorna la taula local.

Tot aquest procés, des de després de declarar les variables locals fins que es deixa de llegir el port sèrie es fa dins d'una instrucció try/catch. Això es fa d'aquesta manera perquè si pel motiu que sigui, alguna línia de codi dóna una excepció, aquesta serà controlada i mostrada per pantalla.

3.6.2.5. SendAndWait

Mètode al que se li passa la cadena de caràcters que es vol enviar al làser i no retorna res.

És el mètode amb el que s'envia la informació cap al làser i per fer-ho utilitza els dos últims mètodes que s'han descrit a aquesta classe *LaserCOM*. Primerament crida el mètode *SendMsg* i se li passa la cadena de caràcters que se li ha passat al propi mètode *SendAndWait*. Aquest envia les dades pel port sèrie, i una vegada aquestes han estat enviades es fa una crida al mètode *WaitAnswer*. A aquest se li passa la constant de temps que és dada membre i espera el que li contesta el làser.

[3] Mirar el protocol de comunicació del làser a l'annex 1.
[4] Mirar el protocol de comunicació del làser a l'annex 1.

3.7. Classe LaserCO2

Aquesta classe és la que més té a veure amb el làser, per això del seu nom i serveix bàsicament per configurar-lo.

3.7.1. Dades membre de la classe LaserCO2:

Objecte de la classe *LaserCOM*. Degut al fet que la comunicació amb el làser és mitjançant el port sèrie, es necessita un objecte d'aquesta classe per poder establir la comunicació.

3.7.2. Mètodes de la classe LaserCO2:

3.7.2.1. Init

És un mètode al que se li passen quatre variables que són les següents, una cadena de caràcters on hi ha el número de port sèrie pel que s'establirà la comunicació. Una altra cadena de caràcters que conté el nom de la caixa que el làser ha de seleccionar de la seva memòria i dues taules de cadenes de caràcters on hi ha els noms dels camps que el làser també ha de seleccionar i a l'altra el contingut de cadascun d'aquests camps.

Primer de tot agafa la cadena de caràcters on hi ha el número del port i el retalla fins a quedar-se només amb el valor. Això s'ha de fer perquè quan es busca els ports sèrie que disposa l'ordinador, els dóna afegint-hi les lletres "COM" al davant, per exemple, "COM1". Una vegada es disposa del valor del port aquest es passa a tipus byte ja que el mètode per obrir el port així ho precisa.

En aquest moment es crida el mètode *Deinit* d'aquesta classe que s'està descrivint. Aquest fet és necessàri degut a que si l'objecte dada membre està inicialitzat de processos anteriors, es desinicialitza.

Després d'això el que es fa és cridar el mètode *ObrirPort* de la classe *LaserCOM* i amb el mètode *SendAndWait* de la mateixa classe s'envia la cadena de caràcters "LMODE". Amb aquest fet s'aconsegueix que el làser contesti informant del mode en el que es troba en aquell moment. Si es troba en mode "demo"[5] se n'informa a l'operari perquè actuï degudament i sinó amb el mateix mètode amb el que s'ha enviat la cadena "LMODE" s'envia la cadena "LMODE STANDBY".

Aquesta cadena serveix per posar el làser en mode "STANDBY", que és el mode en el que ha d'estar per poder ser configurat correctament. Una vegada es troba en aquest mode i no ha informat de cap error, l'aplicació s'atura durant mig segon per donar temps al làser de canviar de mode, si és el cas. Tot seguit es procedeix a enviar al làser la informació pertinent amb el mateix mètode amb el que se li ha enviat informació fins al moment.

Primer de tot se li envia la informació dels camps que s'han d'omplir i es fa enviant-li una cadena de caràcters per cada camp del següent format:

[5] Mirar el protocol de comunicació del làser a l'annex 1.

```
("DBS " + '"' + NomDelCampAlLaser + '"' + ',' + '"' + ValorDelCamp + '"')
```

Amb aquesta cadena de caràcters el làser busca el nom del camp al lloc on té memoritzats els camps i llavors li dóna el valor que se li ha enviat.

Una vegada enviats tots els valors dels camps que conté la caixa que s'ha de marcar, al làser se li envia una altra cadena de caràcters per informar-lo de quina és la caixa que ha de seleccionar de dins la seva memòria. Aquest pas també es fa amb el mètode *SendAndWait* de la classe *LaserCOM* i la cadena de caràcters que se li envia té el següent format:

```
("ACTJOB " + NomEtiqueta)
```

La cadena de caràcters "ACTJOB" és la que informa al làser que els caràcters que van després de l'espai en blanc són els que porten la informació de la caixa que ell ha de seleccionar, la qual té uns camps assignats que són els que s'han modificat al pas anterior.

Tant bon punt al làser se li han enviat totes les dades necessàries, se'l posa en mode de marcatge, la qual cosa es fa enviant-li la cadena de caràcters "LMODE RUN". D'aquesta manera, les senyals d'entrada i sortida del làser treballaràn de la manera que s'espera, fent l'acció de marcar quan se li demani i donant les senyals de *Laser Ready*, *Laser On* i *Error* correctament.

Instrucció	Funció
LMODE[6]	Serveix per canviar l'estat al que es troba el làser. Si s'envia sense anar seguit del mode al que es vol canviar, contesta informant del mode al que està.
DBS[7]	Serveix per informar al làser que se li envia el nom d'un camp i el seu contingut.
ACTJOB[8]	Serveix per informar al làser que se li envia el nom de la caixa que ha de seleccionar.

Taula 3.7.1 - Taula informativa de les instruccions que s'envien al làser.

En aquest punt el làser ja es troba totalment inicialitzat i apunt per marcar, per això, degut a que la comunicació sèrie ja no serà necessària fins que s'hagin acabat de marcar totes les caixes d'aquesta comanda, es tanca el port sèrie. Per fer-ho s'utilitza el mètode *TancarPort* de la classe *LaserCOM* i d'aquesta manera finalitza la comunicació.

Cal dir que pràcticament tot el codi d'aquest mètode es troba dins d'una condició "if", la qual permet que s'executi tot el mètode si el número del port sèrie amb

[6] Mirar pàgina 101 de l'annex 1.
[7] Mirar pàgina 102 de l'annex 1.
[8] Mirar pàgina 100 de l'annex 1.

el que es pretén treballar és positiu o 0. En cas de ser negatiu no s'executaria i informaria a l'operari amb un missatge que el port sèrie no és vàlid.

3.7.2.2. Deinit

Mètode al que no se li passa cap paràmetre ni en retorna cap.

El primer que fa aquest mètode és mirar si l'objecte de la classe *LaserCOM* que és dada membre d'aquesta classe *LaserCO2*, està inicialitzat o no fa referència a res. En cas de no fer referència a res aquest mètode no actua, però si es troba inicialitzat, envia al làser la cadena de caràcters "LMODE STANDBY" amb el mètode *SendAndWait* de la classe *LaserCOM*. D'aquesta manera el làser es queda amb un mode en el que podria ser inicialitzat i també s'hi pot estar mentre no fa res. Una vegada fet això, es tanca el port amb el mètode *TancarPort* de la classe *LaserCOM* i s'iguala l'objecte d'aquesta classe que és dada membre de *LaserCO2* a "null" perquè a partir d'aquest moment no fagi referència a res.

3.7.2.3. ConfigAplicacio

Mètode al que no se li passa cap paràmetre i retorna una taula de cadenes de caràcters. Aquest mètode permet llegir el fitxer de text de configuració de l'aplicació i agafar-ne les dades.

Primer de tot es crea un objecte de la classe *StreamReader* i amb el mètode *ReadLine* d'aquesta classe es guarda la línia del fitxer de text a una cadena de caràcters. Com que a aquest cas només hi ha tres línies, per llegir la segona i la tercera es fa de la mateixa manera i es guarda a dues variable del mateix tipus.

Una vegada es tenen les tres variables plenes, es declara una taula de cadenes de caràcters i s'hi guarden a les tres primeres posicions.

Aquesta taula de cadenes de caràcters és la que aquest mètode retorna.

3.8. Classe LaserXML

Aquesta classe és la que treballa amb els dos fitxers que es troben en format xml. Els seus mètodes permeten tractar totes les dades que es troben en ells.

3.8.1. Dades membre de la classe LaserXML

- Taula de cadenes de caràcters on es guarda el contingut del fitxer de dades.

- Taula de cadenes de caràcters on es guarden els camps que corresponen a una certa caixa.

- Taula de cadenes de caràcters on es guarden els noms que els camps de la cadena anterior tenen a la memòria interna del làser.

- Taula de cadenes de caràcters on es guarden els noms que els camps tenen al fitxer de dades.

- Taula de cadenes de caràcters on es guarden els valors que els camps tenen al fitxer de dades.

3.8.2. Mètodes de la classe LaserXML

3.8.2.1. BuscarFitxer

És el mètode que permet seleccionar d'un directori concret els fitxers que interessen a aquesta aplicació.

Per fer-ho es declaren dues taules de cadenes de caràcters. A una s'hi guarden tots els fitxers que hi ha al directori seleccionat i a l'altra s'hi guarden només els noms dels fitxers que interessen. Com que no se sap la quantitat de fitxers que hi poden haver en qualsevol de les dues taules, aquestes es declaren de zero posicions i cada vegada que se n'ha d'afegir un, la taula s'allarga una posició.

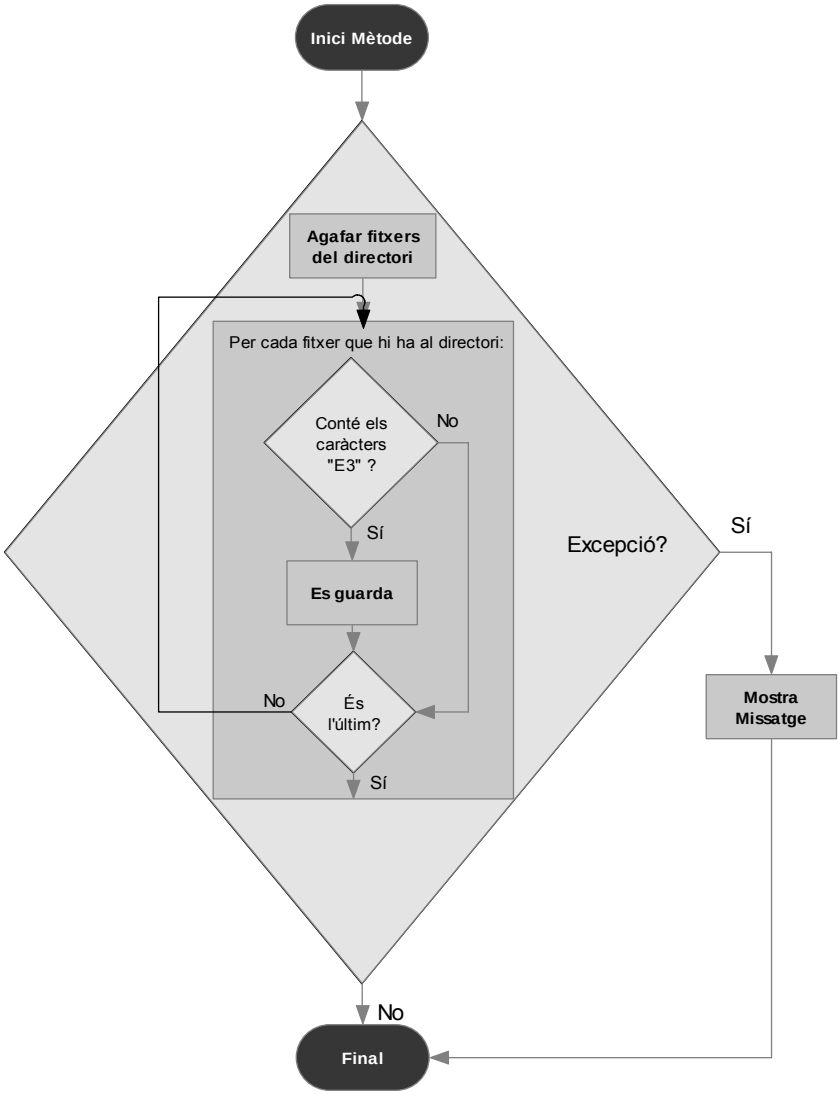


Diagrama 3.8.1 - Esquema de funcionament del mètode *BuscarFitxer*.

Mitjançant un bucle *foreach* s'agafa cadascun dels noms dels fitxers de la primera taula i amb una condició *if* i el mètode *Contains* de la classe *string* es comprova si és un dels fitxers que interessa a l'aplicació o no. Si ho és es retalla la cadena de caràcters fins a quedar només el nom del fitxer i l'extensió i es guarda a la segona taula de cadenes de caràcters que s'ha declarat. L'extensió és molt important que també hi sigui ja que quan es vulgui obrir el fitxer, si no hi ha l'extensió, no es podrà obrir.

Una vegada s'ha omplert la segona taula amb els fitxers que són d'interès, aquesta és retornada pel mètode.

El codi que forma aquest mètode es troba dins d'una instrucció try/catch per controlar les possibles excepcions que hi hagi durant el procés.

3.8.2.2. AgafarDades

És un mètode al que se li passen dos paràmetres i no en retorna cap. Els dos paràmetres que se li passen son dues cadenes de caràcters on una de les quals conté el nom d'un directori i l'altre conté el nom d'un fitxer.

Per poder llegir les dades del fitxer que se li ha passat el nom per paràmetre es crea un objecte de la classe *XmlTextReader*. A aquest objecte se li ha de dir el directori al que es troba el fitxer i el nom del fitxer, cosa que s'aconsegueix unint les dues cadenes de caràcters que s'han passat al mètode.

També cal declarar una variable de tipus cadena de caràcters que serà on es guardarà el text que llegeix en cada cas.

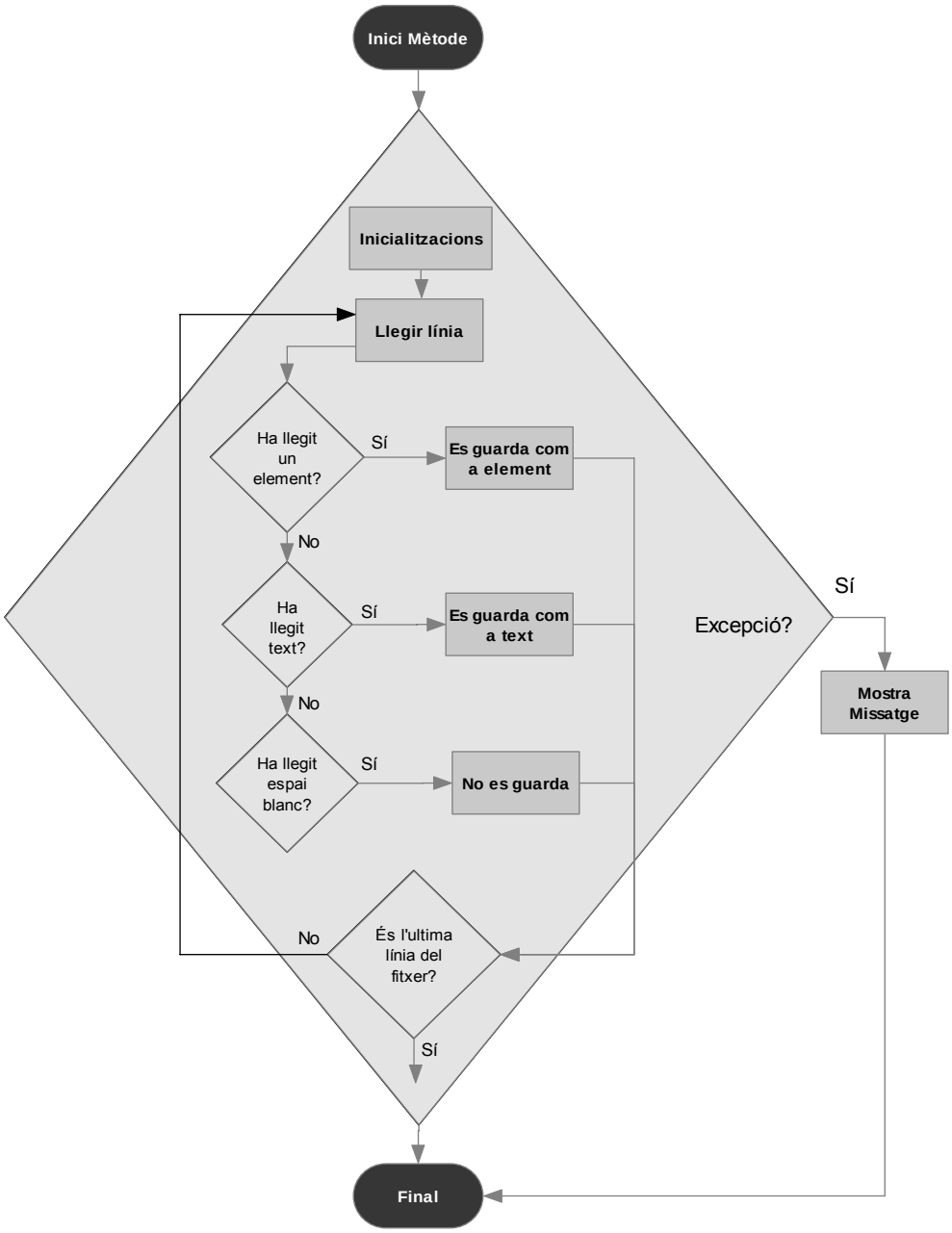


Diagrama 3.8.2 - Esquema de funcionament del mètode *AgafarDades*.

Una vegada fetes les inicialitzacions es pot procedir a llegir el fitxer. Es comença amb una seqüència *while* en que la condició donarà cert mentre no s'hagi arribat al final del fitxer. D'aquesta manera és segur que tot el fitxer serà llegit independentment de la llargada que tingui. Dins d'aquesta seqüència s'hi col·loca una sentència *switch* en la que la variable és el tipus de node que es pot trobar dins el fitxer xml.

Si es dóna el cas que el node és un element, se n'agafa el nom i es guarda a la cadena de caràcters declarada al principi del mètode afageixent-li uns caràcters que

permeten diferenciar-lo de la resta. Una vegada modificat el nom, aquest és guardat a la taula de cadenes de caràcters que és dada membre i que guarda el contingut del fitxer. Aquesta taula al principi és de zero posicions, per tant, cada vegada que s'hi vulgui guardar alguna cosa, abans s'ha d'allargar una posició i després guardar-hi la informació.

Si el tipus de node que llegeix és text, se'n gaurda el valor sense modificar a la cadena de caràcters i aquesta es guarda a la següent posició de la taula on hi ha el contingut del fitxer.

Per últim cas, si el tipus de node és un espai en blanc no es fa res.

D'aquesta manera que es llegeix i es guarda la informació, la taula queda perfectament estructurada ja que a una posició s'hi guarda un element i si te valor aquest es guarda a la següent. Així, quan s'han de recollir les dades tot és molt més simple.

Tant bon punt la condició del *while* dóna un resultat falç indicant que s'ha arribat al final del fitxer, amb el mètode *Close* de la classe *XmlTextReader* es tanca la comunicació i s'acaba el mètode.

Aquest mètode també es troba dins la instrucció try la qual si hi ha alguna excepció durant l'execució del codi, la instrucció catch la recoll i en mostra la informació per mitjà d'un missatge.

3.8.2.3. BuscarDinsXML

Mètode que recull el valor d'un cert element o camp dins la taula on hi ha tot el contingut del fitxer de dades. Per fer-ho se li passa una cadena de caràcters amb el nom del camp que se'n vol conèixer el valor i en retorna el valor.

Utilitza una seqüència *for* des de la posició zero fins al final de la taula per buscar si alguna línia conté el nom del camp que se li ha passat. En cada cas del *for*, primer llegeix la línia de la taula, la guarda a una variable local de tipus cadena de caràcters i llavors amb una sentència *if* i el mètode *Contains* de la classe *string* mira si aquella línia conté el nom del camp que se li ha passat. Si el conté el mètode retorna la cadena de caràcters que hi ha a la següent posició de la taula, ja que tal com s'havia guardat, és on es troba el valor del camp. Pel contrari, si aquella línia no és la que es buscava va entrant al bucle *for* fins que la troba.

També existeix la possibilitat que el camp que es busca no aparegui dins la taula perquè no estava al fitxer, en aquest cas el mètode retorna una cadena de caràcters informant que aquell camp no existeix al fitxer.

3.8.2.4. BuscaCampsConfig

És un mètode que se li passa el nom d'una caixa i l'objecte de la classe *dataSet* on hi ha guardada tota la informació del fitxer de configuració de les caixes. Amb aquesta informació omple la taula de cadenes de caràcters que és dada membre d'aquesta classe amb tots els camps que conté aquella caixa.

Per fer-ho, degut a que l'identificador de la caixa al fitxer de dades no concorda amb l'identificador de la caixa del fitxer de configuració de les caixes, el primer que es

fa és arreglar-lo perquè coincideixin. El problema es soluciona traient el primer caràcter de la cadena que han passat al mètode i d'aquesta manera, una caixa que tenia l'identificador "E3004" passa a tenir l'identificador "3004". No és que a partir d'ara es canviï aquest paràmetre fins al final, sinó que en aquest mètode es necessita d'aquesta manera per poder-ne buscar els camps.

Mitjançant un bucle *foreach* es mira dins de cadascuna de les files de la taula que conté la configuració de les caixes de l'objecte *dataSet.* Dins de cada fila, amb una sentència *if* es mira si la columna que conté els identificadors de les caixes concorda amb l'identificador modificat anteriorment. Si no és així mira la següent fila, i si concorda agafa el contingut de la columna que conté els camps. Aquest procés es va repetint fins al final ja que els camps que formen una caixa poden no estar en ordre ni ser consecutius.

Cada vegada que troba un valor dels que busca, com totes les dades membre d'aquesta classe, abans de guardar la informació ha d'allargar la taula una posició per després guardar-li.

3.8.2.5. AgafarNomEtiqLaser

És el mètode que retorna el nom amb el que la caixa està guardada al làser. Se li passa el nom de la caixa i l'objecte de la classe *dataSet* on hi ha la informació del fitxer de confguració de les caixes i amb això aporta el nom amb el que aquesta caixa està a la memòria interna del làser.

Primer de tot es declara una variable de tipus cadena de caràcters que serà on es guardarà el nom que es busca i la que el mètode retornarà. Una vegada fet aquest pas, igual que al mètode anterior aquí també s'ha de treure el primer caràcter de la cadena de caràcters que s'ha passat per paràmetre.

Una vegada està tot inicialitzat correctament, amb un bucle *foreach* es recorren totes les files de la taula que està guardada a l'objecte de la classe *dataSet* i que conté les caixes. Amb una sentència *if* es mira si a la columna que hi ha els identificadors de les caixes correspon amb el que s'ha passat a aquest mètode. Si és el que es busca, de la mateixa fila, s'agafa el valor de la columna on es guarda el nom que aquella caixa te el làser i és el valor que retorna el mètode. Si pel contrari no ho fos, va buscant fins que el troba.

3.8.2.6. AgafarNomCampsLaser

És el mètode que omple la dada membre que conté els noms amb els que el làser té guardats els camps de la caixa amb la que s'està treballant. Per poder-ho fer només és necessari passar-li l'objecte de la classe *dataSet,* ja que la taula que també necessita és la dels camps que conté la caixa i aquesta també és dada membre. Aquest mètode tampoc retorna res, com ja s'ha comentat, omple la taula dada membre.

Amb un bucle *foreach* es recórrer cadascun dels camps que hi ha a la dada membre que conté els camps d'aquella caixa. Dins d'aquest bucle hi ha un altre bucle *foreach* que recórrer cada fila de la taula on hi ha els camps dins l'objecte de la classe *dataSet*. En aquest punt es mira si a la columna de la taula on hi ha els noms dels

camps hi apareix el nom del camp que en aquell moment s'està buscant. Si és un altre camp es mira a la següent fila, i si coincideixen s'agafa el valor de la columna d'aquella mateixa fila on hi ha guardat el nom amb el que els camps estan guardats a la memòria interna del làser. Abans de guardar el valor a la taula, aquesta s'allarga una posició i a l'última s'hi guarda.

Aquesta seqüència de buscar dins la taula es fa per cada camp fins que a la taula on es guarden els camps no n'hi hagi més.

3.8.2.7. AgafarNomCampsXml

Aquest mètode fa exactament el mateix que l'anterior però amb dues diferències considerables una vegada ha localitzat el camp dins la taula. La primera és que en comptes de guardar la variable que hi ha a la columna on hi ha el nom que el camp té al làser, agafa la que hi ha a la columna on es guarda el nom que tenen al fitxer de dades. La segona és que en comptes de guardar aquestes dades a la dada membre on es guarda el nom que els camps tenen al làser, es guarden a la dada membre on hi ha el nom que els camps tenen al fitxer de dades.

3.8.2.8. AgafarValorCampsXml

És el mètode amb el que s'omple la dada membre que conté el valor dels camps que formen la caixa. Per fer-ho no necessita que se li passi cap paràmetre i tampoc retorna res ja que omple la dada membre esmentada anteriorment.

A partir d'un bucle *foreach* es recorre cadascuna de les posicions de la taula que conté els noms amb els que els camps estan guardats al fitxer de dades. Per cadascun d'ells allarga la taula dada membre que contindrà el valor dels camps en una posició. Després, passant el camp per paràmetre al mètode *BuscarDinsXml* d'aquesta mateixa classe busca el valor del camp i el guarda a la taula.

3.9. Enumeració LaserErrors

A l'apartat 3.6.2.4. s'explica que quan el que es rep del làser és un valor negatiu significa que aquest ha enviat un error i és en aquesta enumeració on hi ha el significat de cadascun d'aquests valors.

En aquell mateix apartat s'explica que quan això passa es mostra un missatge per pantalla informant a l'operari de l'error, doncs bé, és aquí on va a buscar el significat del valor. D'aquesta manera l'operari no ha de tenir un paper penjat de qualsevol manera per saber què vol dir el valor que l'hi ha aparagut a la pantalla. Tot i així, no significa que la informació que es mostra sigui del tot clara, però és suficient per fer-se'n una idea d'on pot estar el problema.[9]

3.10. Control d'errors de l'aplicació

Els programes poden provocar errors en temps d'execució. Els errors provocats per un comportament incorrecte són difícils de detectar, ja que es deuen a un error en

[9] Mirar el protocol de comunicació del làser a l'annex 1, pàgina 94.

el codi font del programa, però no produïxen errors ni al compilar ni a l'executar-se; simplement, el programa no funciona bé i retorna resultats incorrectes.

Existeixen altres errors que sí que són detectables i no són evitables a priori. Són errors produïts per circumstàncies alienes al programa, com per exemple un error a l'accés a un fitxer o a una base de dades, un error de comunicacions, un desbordament de pila, etc.

Els errors o excepcions, es poden gestionar en C-Sharp perquè no acabi el programa bruscament o apareguin missatges d'error del sistema, i també, per intentar solucionar l'error. Per a això, C# i .NET en general tracten les excepcions com objectes. El .NET Framework defineix una sèrie d'objectes "Exception" generals.

Les excepcions es produeïxen en instruccions concretes, ja siguin instruccions del nostre propi programa o d'un mètode que hem invocat d'una altra llibreria de classes. Per a capturar una excepció quan es produeixi, s'utilitza la instrucció try/catch.

```
try
{
        // instruccions que potencialment poden produir excepcions.
}
catch
{
        // gestió de les excepcions trobades.
}
```

Dins del bloc try s'inclouen les instruccions que poden produir errors. Si es produïx una excepció dintre del bloc, automàticament el flux d'execució passa a les instruccions del bloc catch i la resta d'instruccions del bloc try s'ignoren. Si no hi ha cap excepció, el programa continua executant-se.

En el bloc catch s'inclouen les instruccions necessàries per a controlar l'error: es pot informar a l'usuari, resoldre l'error, restaurar el flux d'execució del programa, etc.

Aquest mètode és el que s'ha utilitzat en aquest projecte per controlar les possibles excepcions que es generen durant l'execusió de l'aplicació.

4. Hardware

En aquest capítol s'expliquen detalladament tots els elements que componen el hardware d'aquest projecte i la connexió física que hi ha entre ells. D'aquest tipus d'elements n'hi ha quatre de molt diferenciats, i són els següents:

Element	Funció
Màquina manipuladora de caixes	És l'aparell que subjecta del caixes abans que aquestes siguin marcades i les enretira una vegada s'han marcat.
Làser	És l'encarregat de marcar les caixes.
Targeta entrades i sortides digitals	És una targeta d'entrades i sortides de senyals digitals que s'encarrega de rebre i generar els senyals necessaris per poder portar a terme la seqüència de marcació de les caixes.
Cablejat	És l'element del hardware encarregat d'unir físicament aquests tres elements entre si.

Taula 3.10.1 - Taula informativa del hardware que pren part en el projecte.

4.1. Manipuladora de caixes

La màquina manipuladora de caixes és l'aparell on es porta a terme tot el procés de marcatge de les caixes. Té unes dimensions de 190cm d'alçada, 80cm d'amplada i 80cm de profunditat.

La seva alçada es troba dividida en dues parts. A la part superior és on es fa tot el procés de marcatge ja que és on hi ha tots els elements que es necessiten. La part inferior per la seva part, també es divideix en dues, la part esquerra, que és on es localitza la torre de l'ordinador, i la part dreta, que és on van a parar les caixes una vegada han estat marcades.

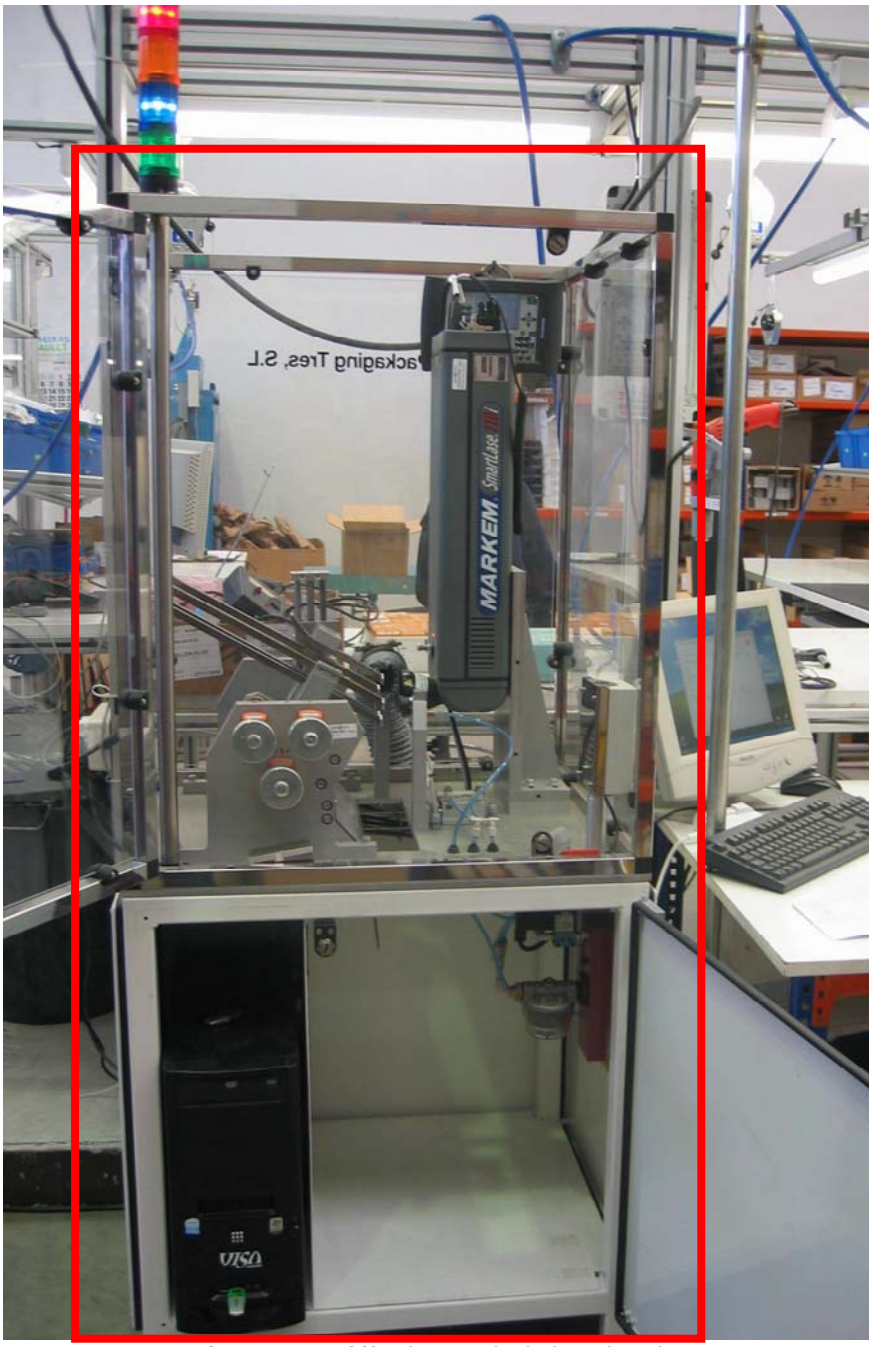


Figura 4.1.1 - Màquina manipuladora de caixes.

Aquesta màquina està feta per una empresa externa que es dedica a fer aquest tipus d'aparells, i no és una màquina fabricada en sèrie, sinó que ha estat fabricada exclusivament per aquest procés. S'alimenta a la xarxa elèctrica de 230V i també necessita aire comprimit ja que alguna de les seves tasques el precisa.

Un aspecte important d'aquest aparell són les senyals elèctriques que aporta a l'aplicació i que en rep. Perquè de bon principi es tinguin clares, s'ha realitzat un esquema gràfic que és el diagrama 4.1..

Diagrama 4.1.1 - Diagrama de les senyals I/O de la màquina.

4.1.1. Elements que formen part de la màquina

4.1.1.1. Autòmat

És l'aparell programable que actua com a unitat central de control de la màquina manipuladora de caixes. El software d'aquest aparell ha estat creat per la pròpia empresa fabricant.

Amb aquest element es generen les senyals de sortida de la màquina i es tracten les d'entrada. Depenent del nivell de les senyals internes de la màquina, a les de sortida pot trobar-s'hi un nivell o un altre. Aquest fet passa sobretot amb la senyal que informa de l'estat de la màquina. L'altra senyal de sortida, la que informa de l'estat de la fotocèl·lula no depèn de cap més ja que estarà a nivell alt en el moment de tenir una caixa molt aprop de l'òptica, sinó estarà sempre a nivell baix.

El condicionant que té la senyal d'entrada és que si la fotocèl·lula està activada no es retira la caixa, però en el moment en que es desactiva, l'autòmat procedeix a retirar la caixa marcada.

A més a més de fer aquest control de senyals, al mateix moment l'autòmat també controla un semàfor indicador que hi ha a la part més alta de la màquina. La funció d'aquest indicador és informar a l'operari de l'estat de la màquina per, d'aquesta manera poder-ho veure encara que estigui fent una altra cosa a un altre punt de l'edifici.

Figura 4.1.2 - Semàfor de la màquina manipuladora de caixes.

4.1.1.2. Dispensador de caixes

És la part de la màquina encarregada de subjectar les caixes que encara estan per marcar just davant l'òptica del làser perquè aquestes puguin ser marcades correctament.

Aquesta part no està controlada per l'autòmat intern de la màquina i no se li va assignar cap senyal elèctrica ni cap manera de detectar que queden poques caixes emmagatzemades o que ja s'han acabat. Per això l'operador ha d'estar controlant la seqüència de marcatge perquè si s'acaben les caixes i encara no se n'han marcat la quantitat que es desitjava, aquest haurà de posar-ne més al dispensador.

Per simplificar una mica aquest problema, s'ha trobat una manera que quan s'acaben les caixes el software ho detecta, para la seqüència i apareix un missatge informant que ja no queden caixes. Llavors l'operador pot anar allà, carregar de nou el dispensador i tornar a posar la seqüència en marxa des del punt que s'havia aturat per poder continuar fent el marcatge de les caixes correctament.

Un aspecte important d'aquesta part de la màquina és el fet que no totes les caixes tenen les mateixes dimensions. Tot i així però, l'aparell dispensador de caixes ha de ser capaç de subjectar-les totes amb la mateixa eficàcia.

Figura 4.1.3 - Mecanisme de col·locació de caixes davant del làser.

Per aquest motiu es disposa d'un mecanisme que permet ajustar els suports del dispensador a cada mida de caixa diferent.

Com es pot veure a la figura 4.1.3, es disposa de tres rosques que permeten desplaçar les tres guies que no són fixes. Cadascuna d'aquestes rosques és independent de les altres i depenent de en quin sentit es faci girar, fa que la guia a la que està connectada s'apropi i s'allunyi.

Aquestes rosques disposen d'un comptador de voltes cadascuna d'elles, cosa que permet que cada mida de caixa tingui uns valors assignats que són amb els que queda perfectament subjectada. D'aquesta manera, quan es canvia de mida de caixa, l'operari només ha d'ajustar les guies als valors que toca i ja es pot disposar a col·locar les caixes.

4.1.1.3. Fotocèl·lula

Aquest element serveix per comprovar que el procés de treure la caixa marcada ha estat correcte.

Com s'explica a l'apartat 4.1.1.4, el mecanisme d'extracció de la caixa es desplaça per anar a buscar la caixa i torna al seu estat inicial, que és on hi ha la fotocèl·lula. Aquesta està situada darrera la ventosa, però no es desplaça a buscar la caixa amb ella, sinó que està fixa. D'aquesta manera quan la ventosa no te res agafat la fotocèl·lula no detecta res, cosa que passa la major part del temps. Quan la ventosa va a buscar la caixa, la recull i retorna, la fotocèl·lula continua estant desactivada, però quan la ventosa arriba a la seva posició inicial amb la caixa, la fotocèl·lula s'activa. Sense tenir res a veure amb la fotocèl·lula, l'autòmat que controla la màquina fa que la ventosa es desprengui de la caixa en el moment en que ha arribat a la seva posició inicial i és en aquest punt en el que la fotocèl·lula torna a quedar desactivada.

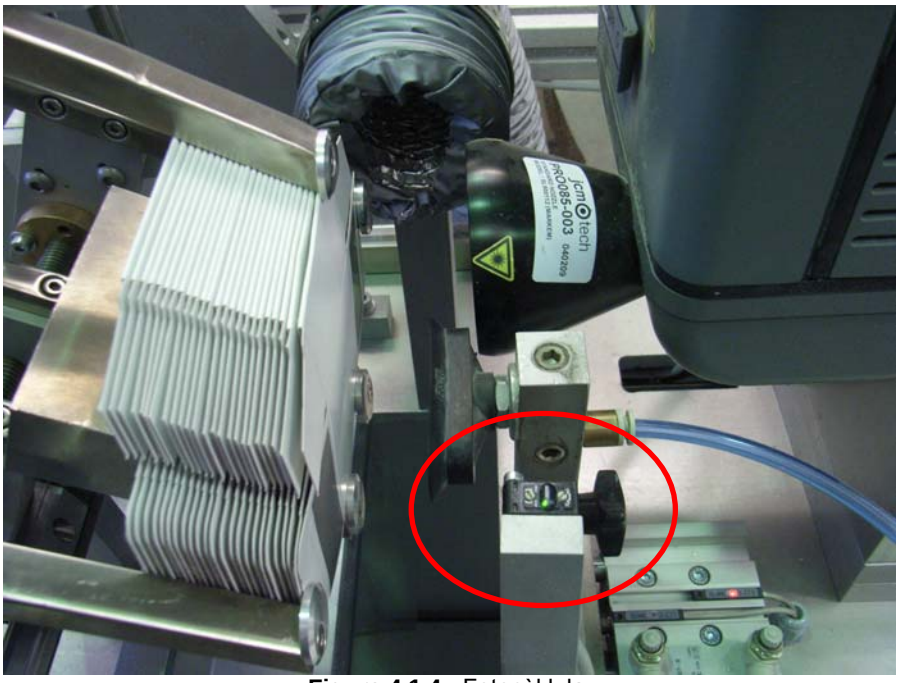


Figura 4.1.4 - Fotocèl·lula.

D'aquesta manera es pot saber si la ventosa ha enretirat la caixa marcada o no ja que durant la seqüència de marcatge de cada caixa la fotocèl·lula ha d'estar uns moments activada, tot i que és un pols minúscul.

4.1.1.4. Extractor de caixa marcada

Aquesta part és una de les més interessants del hardware ja que encara que és molt simple d'activar, té bastanta complexitat interna.

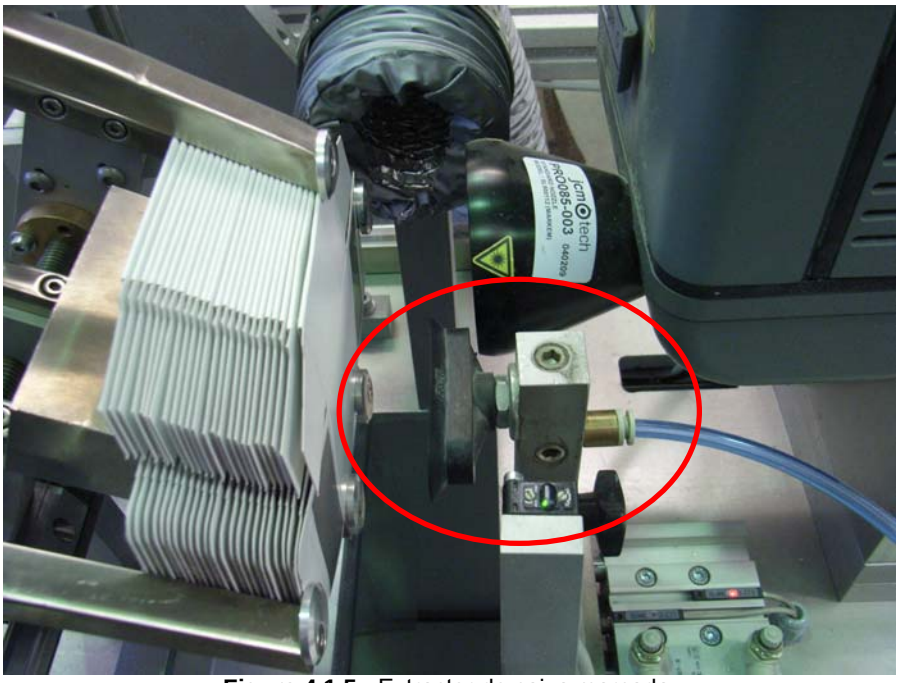


Figura 4.1.5 - Extractor de caixa marcada.

És un sistema que accionat per un pistó d'aire comprimit apropa una ventosa a la caixa marcada. La ventosa però, no és convencional ja que treballa mitjançant l'efecte Venturi, fet que fa que quan s'acosta a la caixa aquesta se li queda enganxada. El procés de retrocedir el fa amb la caixa enganxada i quan arriba a la seva posició inicial deixa caure la caixa pel seu propi pes. Totes les caixes que passen per aquí ja estan marcades.

En aquest punt la ventosa s'espera fins que se li torni a donar l'ordre de repetir aquest procés que s'acaba d'esmentar. Perquè torni a fer aquest procés només necessita que se li enviï un pols d'uns 100ms i l'autòmat intern ja fa la resta.

Aquest extractor funciona a partir de l'efecte Venturi perquè apart que és molt eficient, és menys sorollós que la resta.

4.1.1.5. Extractor d'aire

És l'aparell encarregat de d'extreure el fum produït pel fet de cremar el cartró durant el procés de marcar una caixa. Encara que aquest procés dura pocs milisegons, el fum que es genera és suficientment important com per molestar a la persona que està treballant aprop i per aquest motiu és necessari que hi sigui.

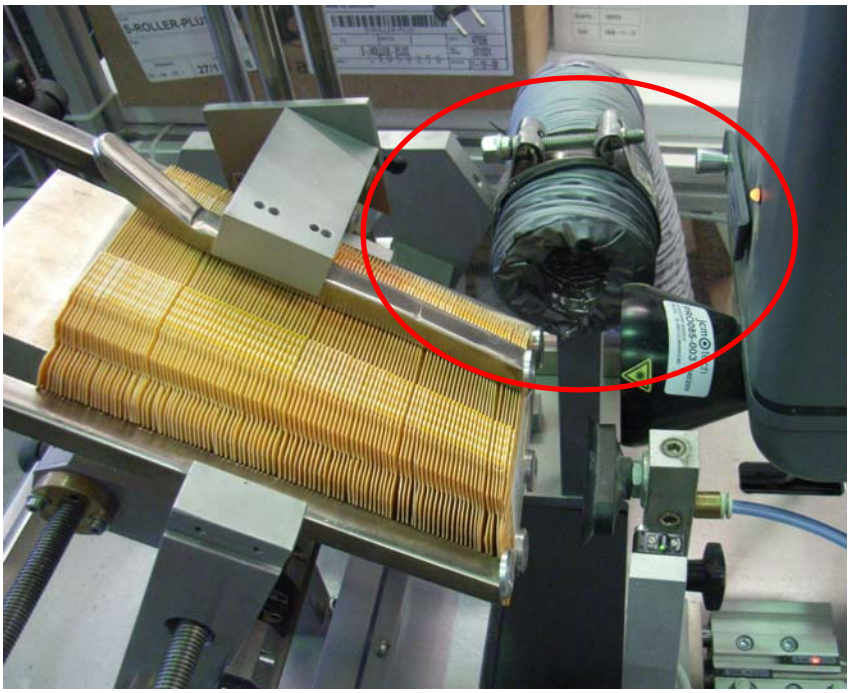


Figura 4.1.6 - Extractor d'aire.

És un extractor d'aire normal i corrent que l'operari posa en marxa en el moment que es disposa a marcar caixes. Aquest aparell però, no es troba dins la màquina manipuladora de caixes degut a l'espai que ocupa. Per aquest motiu, un tub va des d'on es genera el fum fins a l'extractor i un cop allà, aquest l'empeny fins a l'exterior.

4.2. Laser

El làser és una de les parts més importants d'aquest projecte ja que és l'encarregat de fer el marcatge de cada caixa. És de CO_2, és a dir que crema i té una potència de 10W.

Com es pot veure a la figura 4.2.1, consta de dues parts, que són una consola amb la que es configuren els paràmetres interns del làser i el propi làser. Els paràmetres interns i la consola però no són tractats en aquest projecte ja que la comunicació que es te amb el làser és per mitjà del port sèrie o bé a partir de les senyals elèctriques que es poden rebre i enviar pels altres connectors.

Figura 4.2.1 - SmartLase 110i. Làser amb el que es marquen les caixes.

Aquesta part de hardware es basa en els diferents connectors externs dels que disposa el làser. En alguns d'ells s'aprofundeix més, mentre que algun altre només és anomenat ja que per portar a terme aquest projecte no s'ha hagut de tractar.

4.2.1. Connectors

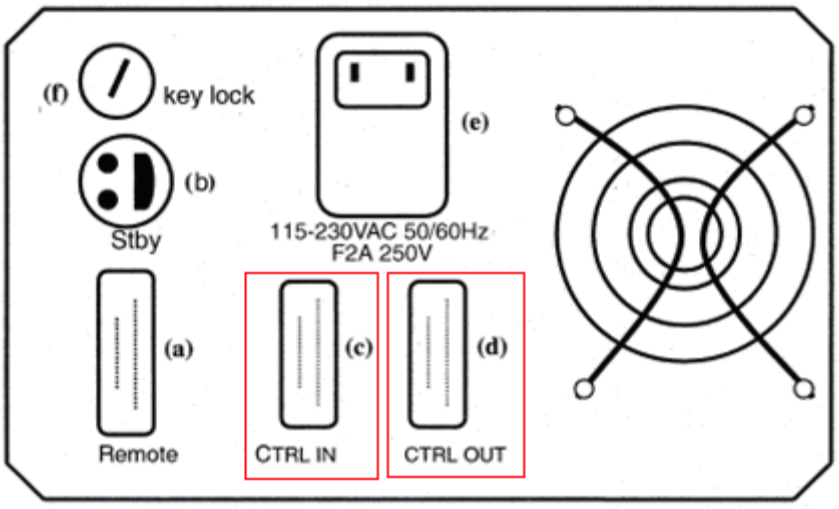


Figura 4.2.2 - Connectors del làser SmartLase 110i. En vermell els que més s'aprofundeix.

Cadascun d'aquests connectors té la seva funcionalitat i importància per poder realitzar qualsevol aplicació i es troben descrits a la següent taula.

Nom	Descripció
Remote	Connector amb el que el làser es comunica amb la consola esmentada anteriorment mitjançant un cable que ja proporciona el fabricant. En aquest projecte aquest connector no es toca.
Stby	El connector que es troba al làser és un connector BNC femella. De sèrie quan es compra el làser ve un connector BNC mascle amb un pont realitzat. Aquest pont és de seguretat ja que si no hi és el làser no funciona correctament. Per tant, el connector BNC mascle ha d'estar sempre connectat.
Control In	Connector que proporciona *Trigger* per les senyals d'entrada *Encoder i Start.* A més a més subministra 12.5Vdc. S'aprofundeix en ell una mica més endavant.
Control Out	Connector que proporciona la comunicació RS 232 i opcionalment la 485/422. A més d'això també és on hi ha les senyals de Laser On, Laser Ready i Error. També subministra 12.5Vdc.
Alimentació	Entrada d'alimentació de la xarxa elèctrica de 115-230Vac, 50-60Hz.
Key Lock	Pany per la clau que s'utilitza per engegar i parar el làser.

Taula 4.2.1 - Taula descriptiva dels connectors externs del làser *SmartLase 110i.*

De tots aquests connectors descrits a la taula 4.2.1, només s'aprofundeix en dos, el *Control In* i el *Control Out*, ja que són els dos amb els que s'ha treballat.

4.2.1.1. Control In

Tal com el seu nom indica, en aquest projecte s'utilitza aquest connector per enviar informació cap el làser. Aquesta informació però només és la que avisa el làser que ha de marcar, la transferència de dades es fa mitjançant el connector *Control Out.*

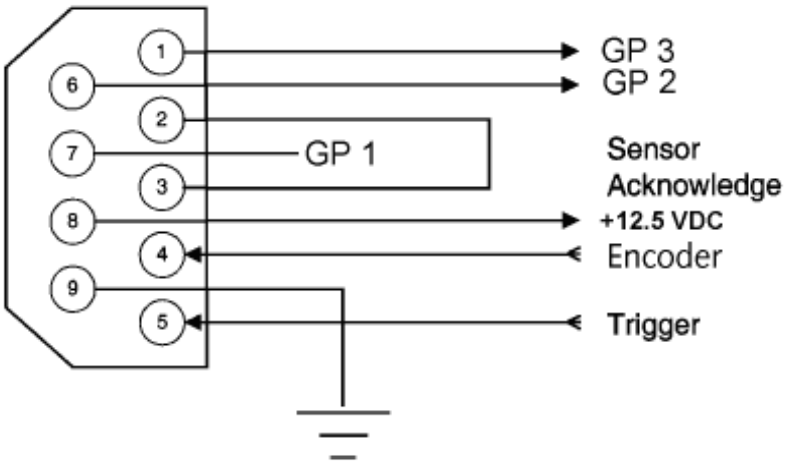


Figura 4.2.3 - Connector DB9 mascle. Pins del connector *Control In*.

A la següent taula es descriu quina és la funció de cada pin del connector *Control In.*

Nº de pin	Nom del pin	Descripció
1	GP 3	
2	Pin 2	Curt-circuitar amb pin 3 ja que així ho marca la guia de servei del làser.
3	Pin 3	Curt-circuitar amb pin 2 ja que així ho marca la guia de servei del làser.
4	Encoder	
5	Trigger	Senyal d'entrada al làser que fa que si aquest està en el mode correcte, no té cap error i aquí se li envia un pols, marca la caixa.
6	GP 2	
7	GP 1	
8	12.5Vdc	Sortida alimentació.
9	GND	Sortida senyal de referència.

Taula 4.2.2 - Taula descriptiva dels pins del connector Control In.

Mitjançant els pins d'alimentació i de *Trigger* s'envia un pols de 30ms al làser informant-lo que en aquell moment ha de fer el marcatge. És aquesta la manera en que a cada seqüència la caixa es marca.

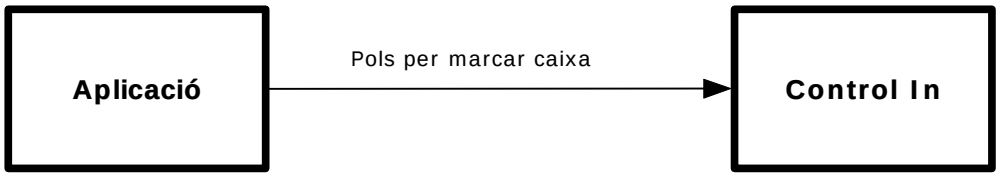


Diagrama 4.2.1 - Esquema de la comunicació entre l'aplicació i el connector Ctrl In del làser.

4.2.1.2. Control Out

Connector molt important en aquest projecte ja que és el que aporta més informació cap a l'aplicació.

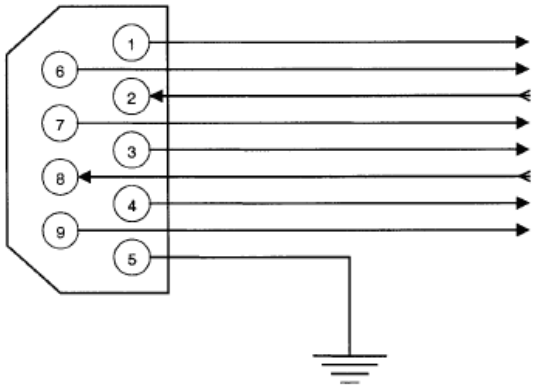


Figura 4.2.4 - Connector DB9 femella. Pins del connector *Control Out*.

A la següent taula es descriu quina és la funció de cada pin del connector *Control Out.*

Nº de pin	Nom del pin	Descripció
1	Laser On	És un senyal de sortida del làser que informa de si el làser es troba en procés de marcatge o no. Si el làser no està marcant aquesta senyal està a nivell baix, mentre que si el làser està marcant la senyal està a nivell alt.
2	RS 232TX	Pin pel que el làser envia informació cap a la aplicació. Normalment per enviar els missatges dels possibles errors que hi hagi hagut durant la transferència de dades.

3	RS 232RX	Pin pel que el làser rep la informació que se li envia des de l'aplicació. Aquesta informació és la de configuració del làser, és a dir, per on se li envien les dades que haurà de guardar per després marcar.
4	Laser Ready	També és una senyal de sortida i aquesta informa si el làser està apunt per marcar o te algun problema. A diferència de les altres aquesta es pot configurar segons els modes en els que es pot trobar el làser. Per aquesta aplicació s'ha configurat de manera que quan el làser està en mode "Run" i apunt, aquesta senyal està a nivell baix, mentre que si no hi està, que sol ser quan està marcant, està a nivell alt.
5	GND	Senyal de referència, terra.
6	RS 485/422 A	Pin pel que el làser es podria comunicar mitjançant el protocol RS 485 o RS 422 amb una aplicació. En aquest projecte però, no s'utilitza.
7	RS 485/422 B	Pin pel que el làser es podria comunicar mitjançant el protocol RS 485 o RS 422 amb una aplicació. En aquest projecte però, no s'utilitza.
8	Error	És la tercera senyal de sortida que hi ha en aquest connector. Aquesta però indica si el làser detecta un error, ja que si està nivell baix vol dir que està tot correcte, mentre que si està a nivell alt s'ha de parar i mirar què li passa.
9	12.5Vdc	

Taula 4.2.3 - Taula descriptiva dels pins del connector *Control Out*.

En aquest connector és on es troben totes les senyals que es consulten durant la seqüència de marcatge, que son la de *Laser On*, *Laser Ready* i *Error.* Apart de per la lectura d'aquestes senyals, aquest connector també és utilitzat per establir la comunicació sèrie amb el làser. Mitjançant aquesta comunicació s'envia al làser la informació que es vol que es marqui a les caixes.

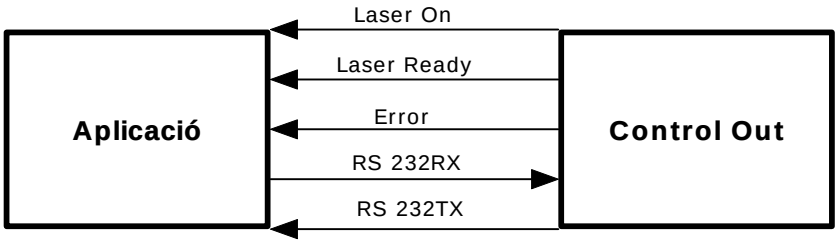


Diagrama 4.2.2 - Esquema de la comunicació entre l'aplicació i el connector Ctrl Out del làser.

4.2.2. Comunicació entre el làser i l'aplicació

Com s'ha comentant en apartats anteriors, entre el làser i l'aplicació hi ha dos tipus de comunicació. La lectura dels possibles nivells de les senyals elèctrics que el làser proporciona i la comunicació sèrie. Tots dos tipus de comunicació es descriuen a la taula 4.2.4.

Comunicació	Descripció
Simple	La comunicació simple entre el làser i l'aplicació és la que es fa durant la seqüència de marcatge de les caixes. Aquesta comunicació consisteix en llegir una sèrie de senyals que permeten saber l'estat en que es troba el làser en cada moment. A aquest apartat també s'ha donat cabuda al pols que s'envia cap al làser perquè aquest faci el marcatge. Aquestes accions es porten a terme mitjançant la targeta de l'apartat 4.3 que és l'encarregada de llegir el nivell dels senyals i amb el software processar-les i fer que la targeta enclavi un dels seus relés, que tancant un circuit permet fer el pols.
Sèrie / RS 232	La comunicació sèrie és la que s'utilitza per enviar al làser les dades que aquest ha de marcar a les caixes. També per aquí, una vegada enviada la informació, el làser contesta en el cas que alguna cosa que se li hagi enviat no sigui correcte informant de l'error que ha detectat. La informació que s'envia i que es rep durant aquesta comunicació no és tant senzilla com en el cas de la comunicació simple. Per poder portar a terme aquesta comunicació s'han hagut d'estudiar bastant detalladament les instruccions que es poden enviar al làser i rebre el que aquest pot contestar.[10]

Taula 4.2.4 - Taula informativa de les diferents comunicacions entre aplicació i làser.

[10] Informació que es troba a l'annex 1.

4.2.3. Memòria interna

L'estructura de la memòria interna del làser és molt semblant a l'estructura del fitxer de configuració de les caixes ja que també es divideix en camps i caixes. El que sí que difereix però és el nom que tenen aquestes variables per ell.

Nom variable al fitxer	Nom variable al làser
Camp	Base de dades
Caixa	Fitxer

Taula 4.2.5 - Taula informativa del nom de certes variables al làser.

Igual que passa al fitxer de configuració de les caixes, la configuració dels diferents fitxers del làser es fa a assignant-los diferents bases de dades.

4.3. Targeta d'entrades i sortides digitals

Aquesta targeta és l'encarregada de fer que les senyals de sortida de l'aplicació siguin reals i que de les diferents senyals d'entrada en llegeixi el nivell.

Figura 4.3.1 - Targeta PCI-1761 d'entrades i sortides digitals.

Característiques i aspectes principals de la targeta:

- Es connecta directament a la placa base de l'ordinador, cosa que la fa molt pràctica, ja que no ocupa espai extern.

- Disposa de 8 relés de sortida.

- Disposa de 8 canals digitals d'entrada.

- Els 8 canals d'entrada suporten fins a 50V, aspecte molt adequat per poder acceptar tant els 12 Volts que li arriben de les senyals que genera el làser com

pels 24 que li arriben de les senyals que genera la màquina manipuladora de caixes.

- Té la propietat de treballar per flancs i per nivell. Aquest aspecte és molt important ja que en aquest projecte és necessari treballar de les dues maneres, degut a que les senyals que s'han de llegir, tenen propietats bastant diferents.

Els elements necessaris per poder connectar els cables que porten les senyals del làser i de la màquina manipuladora de caixes cap a la targeta i d'aquesta als altres són els dels apartats 4.3.1. i 4.3.2..

4.3.1. Borna terminal

És l'aparell que permet connectar els cables que porten les senyals digitals cap a la targeta d'entrades i sortides, també s'hi connecten els cables de les senyals que en surten.

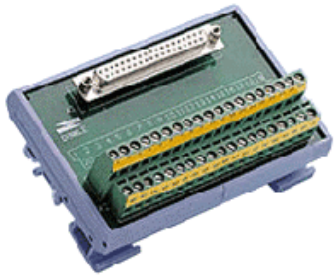


Figura 4.3.2 - Borna terminal.

Com es pot comprovar a la figura 4.3.2 hi ha bastantes entrades i sortides, cosa que fa que hi pugui haver errors a l'hora de connectar el cablejat. Per simplificar aquest fer, a la imatge 4.3.3 es pot veure l'assignació que te cadascun dels pins.

Canals	Sentit	Descripció
0, 1, 2, 3	Entrada	Es disposa del contacte normalment obert i normalment tancat del relé.
4, 5, 6, 7	Entrada	Només es disposa del contacte normalment obert del relé.
0, 1, 2, 3, 4, 5, 6, 7	Sortida	Hi ha els dos bornes pels dos cables que porten la senyal.

Taula 4.3.1 - Taula descriptiva del connexionat de les senyals al borna.

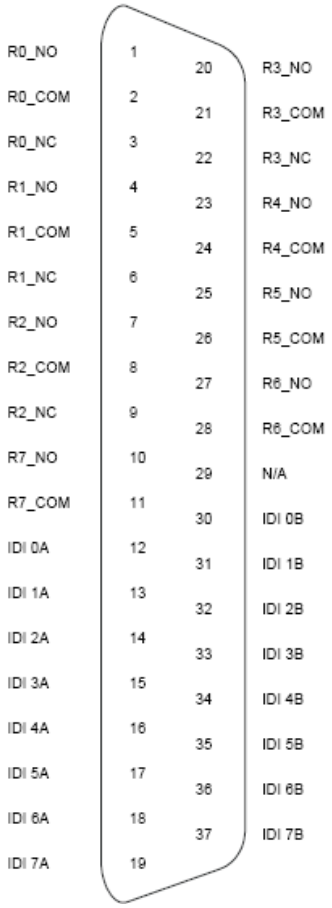


Figura 4.3.3 - Assignació dels pins del connector d'entrades i sortides.

4.3.2. Cable

Cable necessari per connectar la targeta que es troba dins l'ordinador amb el borna terminal on hi ha tots els cables degudament connectats.

Figura 4.3.4 - Cable connector.

Aquests tres elements, amb el seu software degudament instal·lat, permeten conèixer la diferència de tensió que hi ha entre dos punts i d'aquesta manera conèixer a quin nivell es troben els diferents senyals d'entrada.

Els polsos que es necessita generar per fer funcionar algun dels elements també es generen a partir d'aquí. A l'esquema del connexionat, a la figura 4.4.5, es pot veure com es generen tenint com a element un relé.

4.4. Cablejat i connexió

En aquest apartat s'explica com s'ha fet la connexió entre tots els elements hardware que formen aquest projecte.

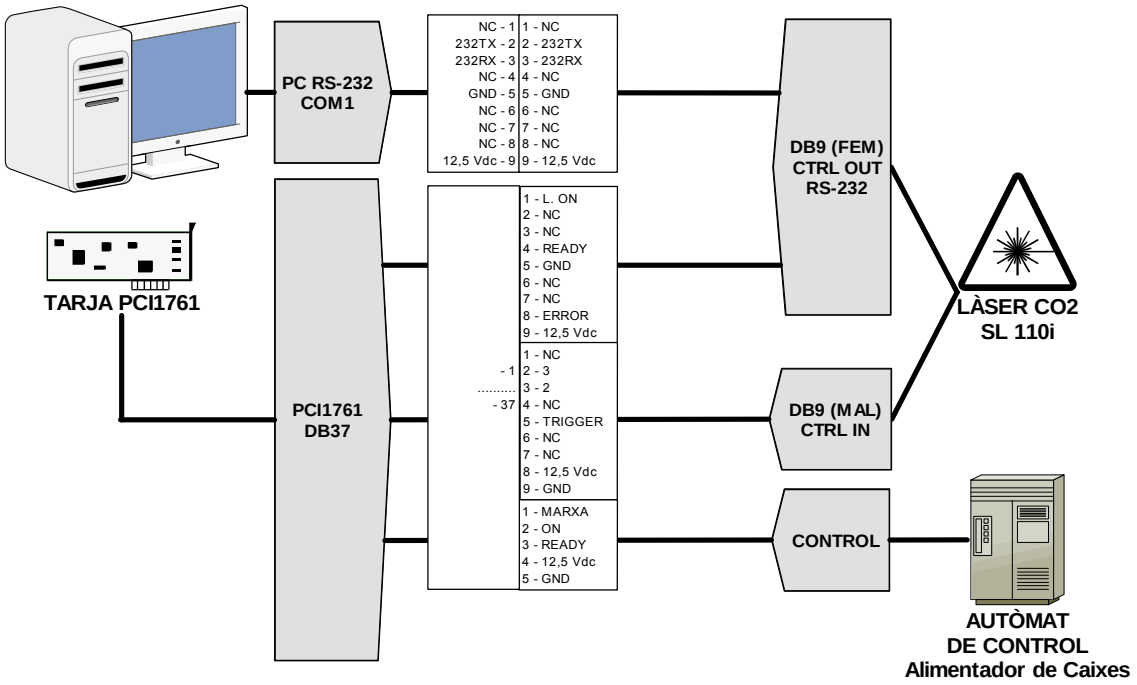


Diagrama 4.4.1 - Diagrama de les connexions entre el hardware.

Per poder portar a terme tot aquest connexionat, el primer que s'ha de fer és mirar quin és el funcionament intern de cada connector que hi pren part. Als apartats 4.4.1. i 4.4.2. s'aprofundeix en els que més complexitat tenen, ja que no només se'ls ha d'endollar.

4.4.1. Connexionat del connector *Control In*

Aquest connector del làser, tal com s'explica a l'apartat 4.2.1.1. és pel que se l'informa que ha de fer el marcatge.

Perquè aquest fet sigui possible s'ha hagut de posar una resistència de "pull-down" de 5,6KΩ a l'entrada de *trigger*. Aquest pas és degut a que la targeta d'entrades i sortides només disposa de sortides de relé. D'aquesta manera, el làser normalment veu nivell baix, mentre que quan el contacte de la targeta que connecta l'entrada de *trigger* amb alimentació es tanca, veu nivell alt. D'aquesta manera és com el làser rep que ha de fer el marcatge.

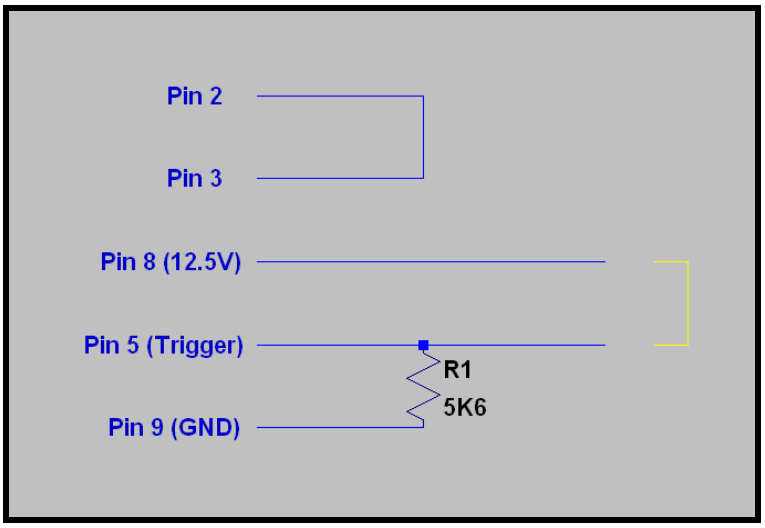


Figura 4.4.1 - Esquema del connector *Control In* del làser.

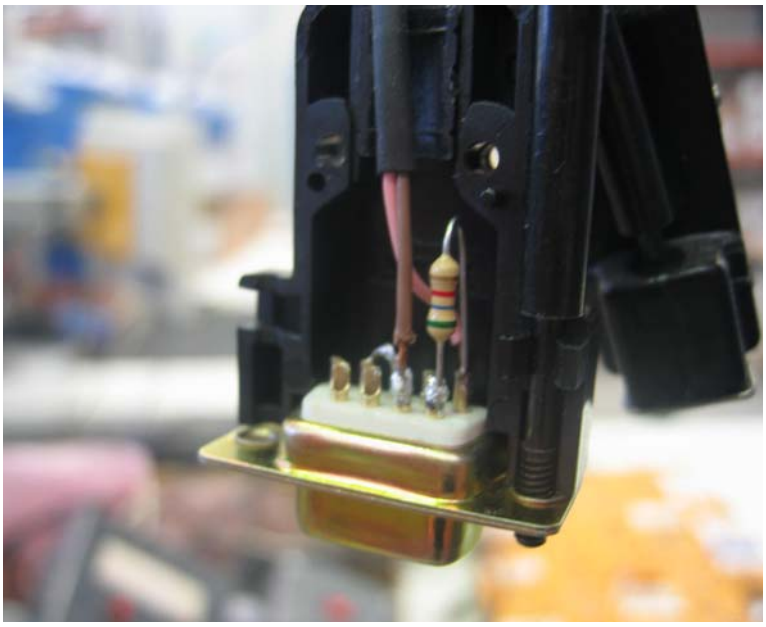


Figura 4.4.2 - Imatge del "pull-down".

4.4.2. Connexionat del connector *Control Out*

Aquest altre connector del làser, tal com s'explica a l'apartat 4.2.1.2. és per on el làser informa de l'estat en que es troba a cada moment.

En aquest cas, el que s'ha hagut de fer ha sigut posar una resistència de "pull-up" de 1KΩ a cadascuna de les senyals informatives. D'aquesta manera per llegir aquestes senyals se sap que si es troben a nivell alt és que estan activades, mentre que si el nivell que es llegeix és baix no ho estan. El fet que les resistències siguin de "pull-up" fa que per llegir el nivell de les senyals es faci entre el pin corresponent i el cinc, que és massa.

Sobre els pins dos i tres no s'hi ha hagut de fer res ja que al ser els que contenen la comunicació sèrie no es necessita que se'ls afegeixi res.

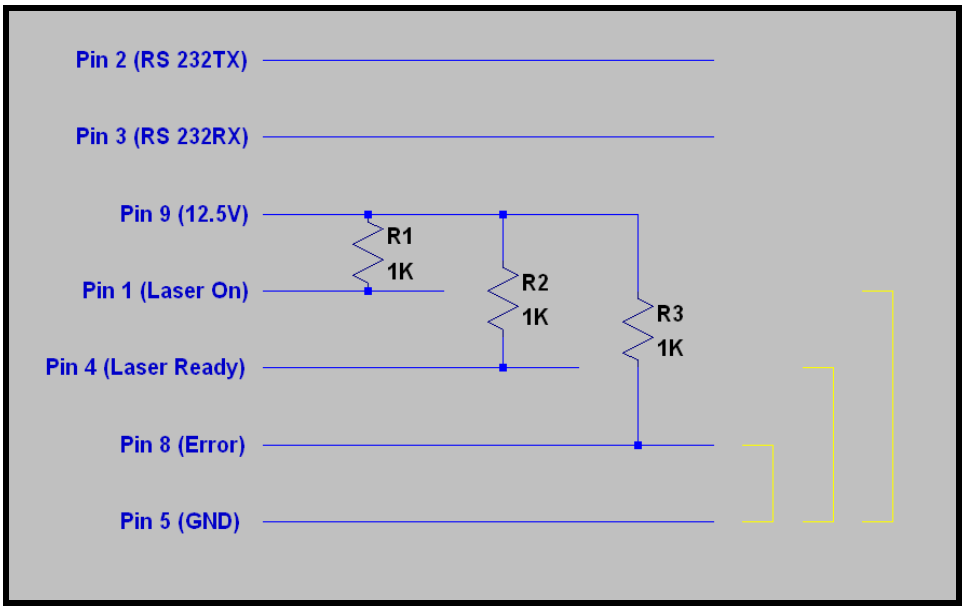


Figura 4.4.3 - Esquema del connector *Control Out* del làser.

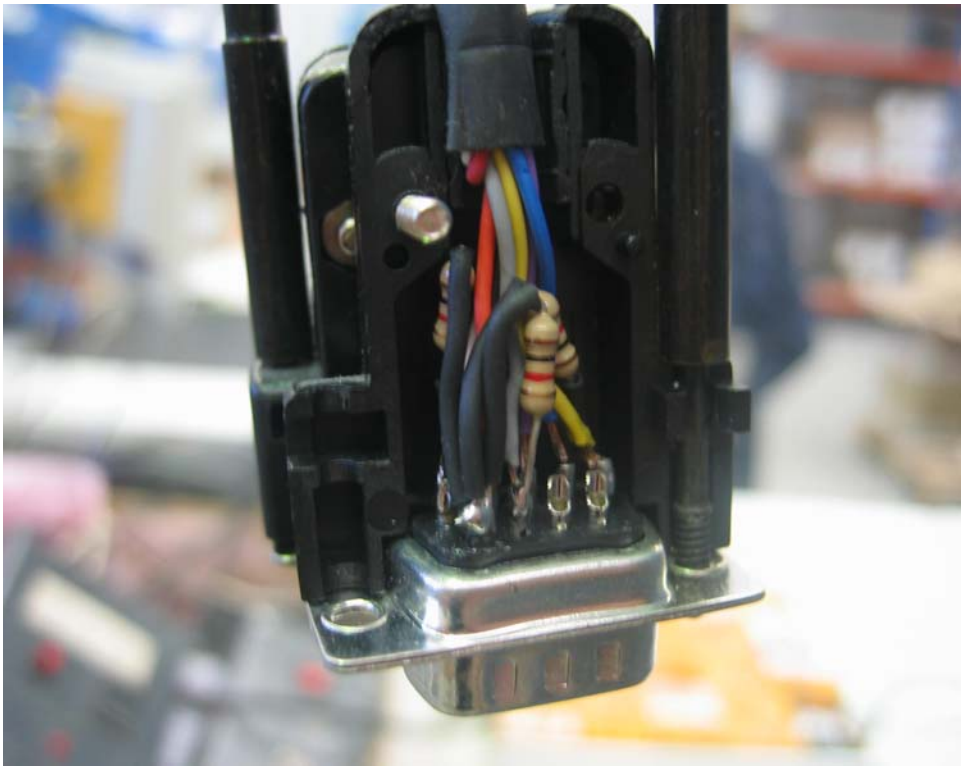


Figura 4.4.4 - Imatge dels "pull-up".

4.4.3. Connexionat i cablejat general

A la figura 4.4.5 es pot veure l'esquema de totes les connexions que hi ha entre els diferents aparells de hardware que conformen aquest projecte.

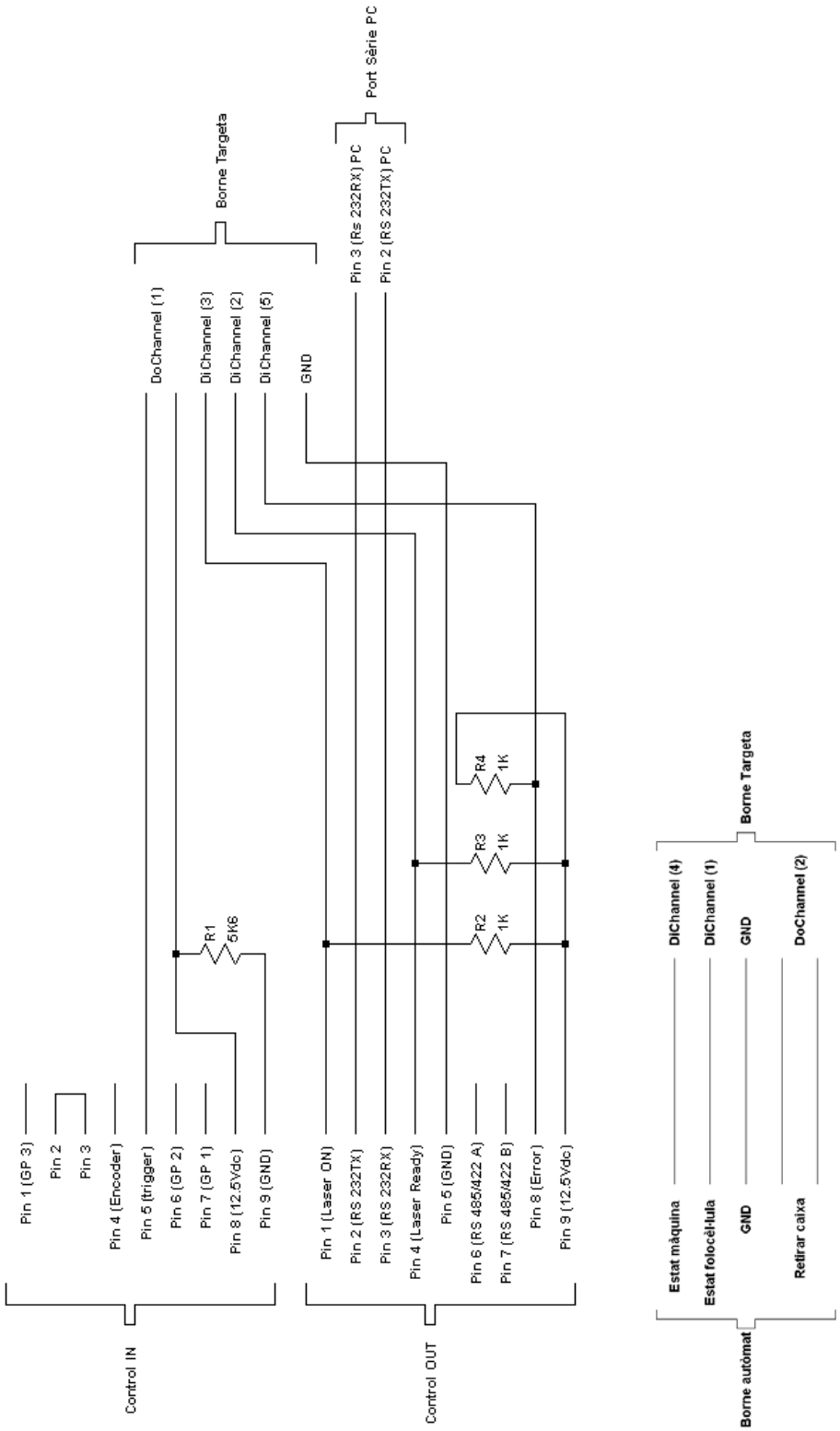


Figura 4.4.5 - Esquema de les connexions entre el hardware.

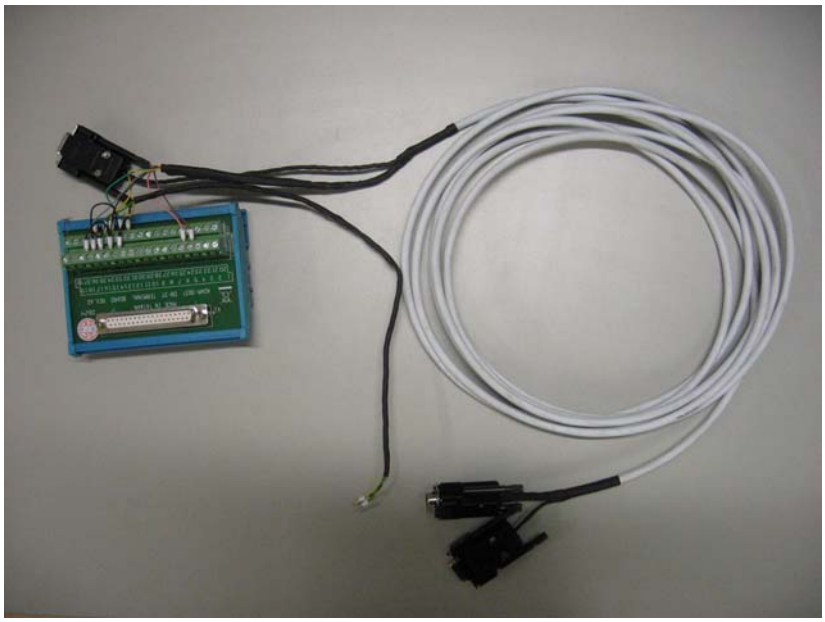


Figura 4.4.6 - Cable que interconnecta tots els components hardware.

4.5. Senyals digitals durant la seqüència de marcatge de les caixes

Els dos tipus de comunicació que hi ha entre el hardware, en cap moment es donen simultàniament. La comunicació sèrie es porta a terme durant la inicialització del làser, fet que dura pocs segons. La comunicació simple, que és la que afecta les senyals digitals, en canvi, només s'utilitza durant la seqüència de marcatge de les caixes.

A la figura 4.5.1 hi ha un esquema d'aquestes senyals durant el procés de marcatge de dues caixes on es veu el nivell al que es troba cadascuna de les diferents senyals digitals.

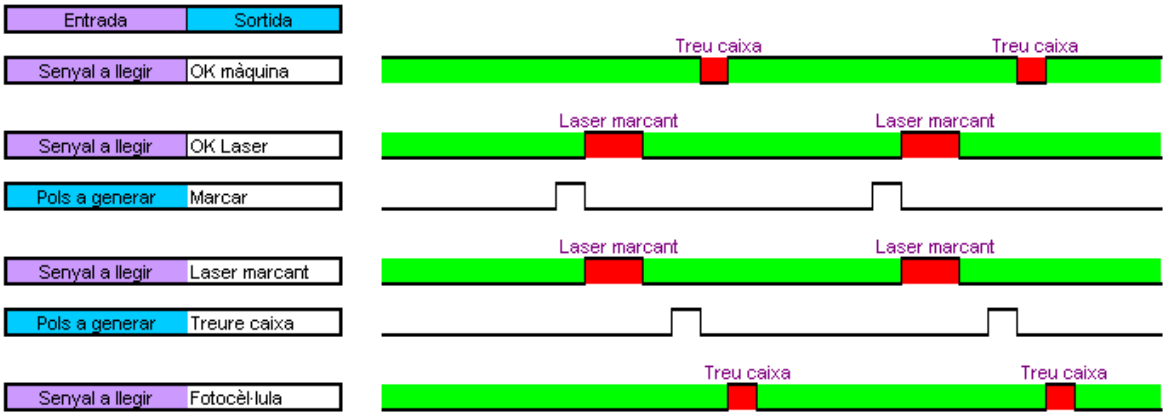


Figura 4.5.1 - Seqüència dels senyals digitals durant el marcatge.

Es pot comprovar com si no hi ha cap error i tot està correcte, la senyal que informa de l'estat de la màquina està sempre a nivell alt menys en el moment que es retira la caixa marcada. En aquest moment la programació de l'autòmat fa que doni nivell baix.

Amb la senyal que es rep del làser informant del seu estat passa una cosa semblant a la de la màquina ja que normalment està sempre correcte menys durant el temps que tarda a fer el marcatge. Aquest fet ja és correcte, ja que si està fent un marcatge, no se n'hi pot fer fer un altre.

En el moment que les dues senyals s'han llegit amb nivell correcte, s'envia el pols al làser perquè aquest marqui. Tant bon punt es detecta que el làser ha acabat de marcar s'envia el pols a la màquina perquè retiri la caixa i quan, mitjançant la fotocèl·lula es veu que la caixa s'ha retirat correctament es torna a començar al principi.

5. Resultats i conclusions

5.1. Valoració econòmica

En aquest capítol s'ha fet una valoració econòmica del procés productiu que aquest projecte ha modificat, fent una comparativa de cost entre el procés antic i l'actual.

5.1.1. Aspectes importants i estimacions

Per poder portar a terme aquesta comparativa s'han fet una sèrie d'estimacions ja que hi ha moltes dades difícils de precisar.

- Com s'explica a l'apartat 2.1.1 una comanda passa a ser una ordre de treball, que per fer aquesta valoració s'ha estimat que de mitjana són d'una quantitat de cent articles.

- El cost d'una hora de feina d'un operari en aquest procés s'ha fixat a 18€, fet que dóna un cost de 0.005€ per segon.

- El cost del software creat per l'autor d'aquest projecte no s'hi ha tingut en compte ja que és un projecte de final de carrera i s'ha realitzat fora de les hores de feina. Aquest fet fa que per l'empresa aquesta part no tingui cost.

5.1.2. Cost de preparació d'una ordre de treball

Com tot procés, tant sigui productiu com no, el primer que es fa és la preparació de tot el que es necessitarà per portar-lo a terme. Aquest fet comprèn entre el moment en que l'ordre de treball arriba al taller d'acabats fins que l'operari ha preparat tot el que necessita durant el procés.

Tot aquest procés es troba descrit econòmicament a la taula 5.1.1.

PROCÉS DE PREPARACIÓ D'UNA ORDRE DE TREBALL			
Abans		**Ara**	
Procés	**Temps**	**Procés**	**Temps**
Escollir equip	1'5s	Processar Ordre de Treball	8s
Escollir etiqueta	1'5s	Escollir Ordre de Treball	2s
Escollir quantitat	1'5s	Aprovisionament caixes	20s
Previsualització	2s	Preparació marcatge	5s
Engegar impressió etiquetes	0s	Engegar seqüència marcatge	0s

Aprovisionament etiquetes	5s		
Total	11'5s	Total	35s

Taula 5.1.1 - Procés de preparació d'una ordre de treball.

5.1.3. Cost total de producció d'una ordre de treball

Una vegada la preparació està calculada, es calculen els diferents costos de producció totals. Els resultats d'aquests càlculs es troben a la taula 5.1.2.

Abans		**Ara**	
Procés	**Preu**	**Procés**	**Preu**
Preparació	0'0575€	Preparació	0'175€
Etiquetatge	1'5€		
Cost etiquetes	0'25€		
Consumibles	0'04€		
Total	1'8475€	Total	0'175€

Taula 5.1.2 - Comparativa de cost dels dos processos.

5.1.4. Amortització

En aquest apartat es fa un càlcul aproximat del temps que es tardarà a amortitzar la nova maquinària comprada per poder renovar el procés productiu tractat en aquest projecte.

Aparell	**Preu**
Màquina manipuladora de caixes	24.600€
Làser	14.000€
Ordinador	400€
Targeta d'entrades i sortides digitals	200€

Taula 5.1.3 - Taula dels preus del hardware adquirit.

A partir dels resultats obtinguts als apartats 5.1.2 i 5.1.3 i de la informació que conté la taula 5.1.3 es pot fer un càlcul aproximat del temps que es tarda a amortitzar tots els elements comprats.

Una dada necessària per obtenir els resultats desitjats és que s'estima que durant un any es produiran 7.000 ordres de treball.

Primer de tot es calcula la diferència del cost de producció de cada ordre de treball, que és el següent:

- Cost procés vell / OT – Cost procés nou / OT = Estalvi / OT
- 1'848€ - 0'175€ = 1'673€

Una vegada se sap l'estalvi per cada ordre de treball es calcula l'estalvi de tot un any:

- Estalvi / OT * OT's per any = Estalvi / any
- 1'673 * 7000 = 11.711€/any

Com a últim pas només falta calcular el cost total de la inversió feta i dividir-la per l'estalvi que s'obté cada any. D'aquesta manera s'aconsegueix saber els anys que es tarda a recuperar la inversió.

- Total inversió / Estalvi cada any = Anys recuperar inversió
- 39.200€ / 11.711€/any = 3'35 anys

Amb el resultat obtingut en aquest apartat es pot comprovar com la inversió realitzada per l'empresa és totalment viable.

Cal afegir a aquest apartat una sèrie de factors que fan que aquesta inversió sigui encara millor. Tot i així, no s'han afegit a la part de càlculs degut al fet que són molt difícils de quantificar.

- Disminució dels costos logístics. Aquest fet és degut a que el departament de logística no ha de comprar etiquetes i tampoc ha de controlar que se'n tinguin en estoc.

-Increment de la versatilitat del procés productiu. S'evita dependre d'un material extern, com són les etiquetes, per poder portar a terme el procés.

- El procés productiu és més ràpid, degut en part, a que s'ha aconseguit fer un sistema molt fàcil de fer funcionar, configurar i inicialitzar.

- Un procés productiu més ecològic, ja que es deixa d'utilitzar paper de les etiquetes i la tinta o tóner de l'impressora que les imprimia.

5.2. Resultats

L'objectiu global d'aquest projecte ha estat complert amb èxit, cosa que fa que els resultats siguin bons. El fet que els objectius s'hagin desglossat, permet que a les conclusions també es faci.

El primer objectiu consistia en crear una interfície gràfica capaç de recollir la informació necessària, tractar-la i enviar-la al làser. Aquesta part de la interfície gràfica s'ha estat provant en diverses ocasions i s'ha conclòs amb èxit.

El segon objectiu era aconseguir que aquesta interfície gràfica tingués una simplicitat elevada per fer que l'operari emprés el mínim de temps, i fós fàcil recordar-ne el funcionament. Aquest apartat també s'ha complert ja que actualment aquesta aplicació ja es troba en funcionament, i només s'ha explicat una vegada a l'operari i està marcant una quantitat de caixes diàries bastant elevada, senes complicacions.

El tercer objectiu demanava la creació, dins la interfície gràfica d'un apartat on es pogués fer la configuració de cada tipus de caixa. Com es pot veure a l'apartat 3.2 de la pròpia memòria, la tercera pestanya de la interfície gràfica, té aquesta com a única funció. Podent, des d'allà, crear una varietat molt elevada de diferents configuracions de caixes.

Finalment, l'últim objectiu que es marcava era el de fer el control de les senyals del làser i de la màquina, per generar la seqüència de marcatge de cada caixa. Aquest objectiu, després de molt esforç també s'ha aconseguit, ja que ha estat una de les parts del projecte que més dedicació ha necessitat.

Figura 5.2.1 – Resultat final de les caixes marcades.

5.3. Conclusions

La meva valoració sobre la creació d'aquest projecte és clarament positiva, i ho és per dos motius. El primer per aconseguir portar a terme un projecte de la vida real. Personalment no em cridava gens l'atenció el fet de fer un projecte que m'hagués inventat expressament per aquesta ocasió, ja que els objectius haguessin set ficticis i no sabríem en cap moment el grau real de la seva dificultat i la seva possible aplicació real. D'aquesta manera, en canvi, tinc o he tingut una motivació addicional i la satisfacció d'haver aconseguit donar solució a una projecte d'una oficina tècnica, que està fent una quantitat de projectes important i d'una complexitat molt elevada.

El segon, per la seva part, és la intenció d'aprendre en profunditat a treballar amb un llenguatge de programació d'alt nivell, que treballi amb classes i objectes, és a dir, programació orientada a objectes. A aquest tipus de programació m'hi havia iniciat en dues assignatures optatives de la carrera, POOEG i Aplicacions d'Internet. Les dues em van agradar molta, i vaig sentir que aprenia moltíssim, i això va fer que cada vegada m'interessés més aquest tipus de programació. El fet és que quan vaig demanar per fer el projecte a l'empresa, i al cap de poc em van presentar les propostes, no vaig dubtar gens a l'hora d'escollir aquesta.

Tot i això, però, cal dir que el camí no ha estat un camí de roses ja que hi ha hagut moments molt complicats. El fet de tenir tant poca experiència en aquest tipus de programació, fa que aconseguir qualsevol funcionalitat, per simple que sigui, pugui convertir-se en un problema molt gran.

D'exemples reals en els que en aquest projecte s'ha viscut aquest fet, n'hi ha dos. El primer va ser durant la creació de la pestanya de configuració de les caixes, on em va costar molt aconseguir fer funcionar correctament els controls amb els que es mostren les diferents taules. Davant un problema d'aquest tipus comences agafant-ho com un repte, i en el cas de no trobar informació, cal força de voluntat per creure-hi i molt d'esforç per aconseguir resoldre'l.

El segon cas va ser en la modificació que es va haver de fer a la seqüència de marcatge. Primerament, totes les senyals a llegir es miraven per nivell. Aquest fet feia que el *timer* hagués d'anar molt de pressa ja que les senyals del làser marcant i de la fotocèl·lula són molt curtes. Això, provocava que la seqüència de marcatge funcionés bé a uns ordinadors, mentre que altres es bloquejaven. Per tant va caler modificar el sistema de lectura de les senyals, havent-se de detectar els flancs, per fer anar el *timer* més lent. El problema però, va arribar a l'hora de programar-los, ja que d'informació de com programar la targeta PCI-1761 per flancs no en vaig trobar enlloc. Aquest va ser l'altre moment critic de la realització del projecte. De casos d'aquests però, n'hi ha molts altres en els que, potser no a un nivell tant elevat, també s'han complicat molt les coses i han estat difícils.

El fet de crear un software, fa que les coses es puguin modificar amb certa rapidesa. Per aquest motiu, en aquest apartat no hi ha millores del projecte, ja que considero que està tant ben fet com sé. Segur que gent que conegui més bé l'entorn de programació i el llenguatge, seria capaç de fer-ho molt millor, o amb moltes menys

línies de codi font, però el fet d'anar modificant el programa tot sovint, fa que no sàpiga quines millores podria haver-hi.

La meva valoració del projecte en general però, continua sent molt positiva, ja que he après moltes més coses de les que en un principi em pensava, apart de complir el meu objectiu d'aprendre més profundament la programació orientada a objectes.

Bibliografia

- **Llibres:**
  - ARORA Geetanjali, AIASWAMY Balasubramaniam, PANDEY Nitin. *Proyectos Profesionales Microsoft C#.* Madrid: Ediciones Anaya Multimedia, 2002.
- **Pàgines web:**
  - http://java.sun.com 05/03/09
  - http://www.es-asp.net/Foro/foro-c--f.aspx 05/03/09
  - http://www.windowsforms.net/ 08/03/09
  - http://www.codeguru.com/Csharp/Csharp/cs_graphics/ 06/03/09
  - http://www.gotdotnet.com/ 15/03/09
  - http://samples.gotdotnet.com/quickstart/winforms/ 01/04/09
  - http://www.codeproject.com/index.asp 01/04/09
  - http://msdn.microsoft.com 19/03/09
  - http://www.devx.com/dotnet/Article/33748 25/04/09
  - www.markem.com/products/description.jsp?productid=5548 01/03/09
  - http://www.solovb.net/index.php/2009/01/06/datagridview/ 09/04/09
  - http://www.developer.com/net/csharp/article.php/3718691 25/04/09
  - http://www.c-sharpcorner.com/Forums/ShowMessages.aspx?ThreadID=55967 01/05/09
  - http://www.csharpfriends.com/Forums/ShowPost.aspx?PostID=59002 01/05/09
  - http://www.dreamincode.net/forums/showtopic35775.htm 28/04/09
  - www.advantech.com/ 05/05/09
  - www.termcat.cat/ 19/05/09

Annexes

Annex 1: Protocol comunicació del làser

Annex 2: Datasheet targeta entrades/sortides

ANNEX 1: Protocol de comunicació del làser

SmartLase "i" Series Command Line Communications Protocol

Prepared By	Date	QA/RA	Date	Owner	Date
Kevin Franklin Moses Derkalousdian	7/26/05				

MARKEM SmartLase 1

Document No 061021 Rev X1 Confidential Preliminary Page 1 03/06/2009

The following is a complete list of commands.

The return error codes are always given as a negative number. They map into the table on the following page(s).

Note the SL1> prompt. This is sent by the SmartLase to signal the completion of a command. If there is an error code returned, it will be sent followed by a <CR>SL1> sequence.

Josep Codina Bau

Syntax Overview

- General Syntax:

TO LASER UNIT: *Command [parameter] ... [parameter]<cr>*
FROM LASER UNIT: *<cr> [Optional Response]<cr>*
SL1>

- Command and parameters are not case sensitive.
- A single comma and/or one or more spaces can be used to separate the command and/or its parameters.
- Leading and trailing spaces are ignored.
- Every command ends with a carriage-return.
- The response to the command, if any, is sent once the SmartLase receives the carriage-return.
- If an error is detected, the response is a minus sign, followed by the ASCII digits for the error code.
- The response, or error response, is followed by a carriage-return.
- The prompt sequence, "SL1>" indicates the SmartLase is ready for the next command.
- A parameter can be one of the following types:
  - Numeric: Requires a parameter contain only numbers (0-9).
  - Real: Requires a parameter contains only number and a decimal point (0-9).
  - String: A parameter with any characters inside quotes is considered a string. A parameter without quotes that is not a Numeric or a Real is also a string.
- If a string contains non-printable ASCII values they must be passed in as a quoted string and each hexadecimal byte represented as an ASCII character with a backslash preceding it. See table below.
- Quotes must be used to force a parameter to be treated as a string when it contains special characters. See table below
- The total length of a command and its parameters must not exceed 246 bytes.

Required String Example	Proper String CLI Format	Description
123	"123"	String with only numeric. Without the quotes this will be treated as a numeric parameters
Hello World	"Hello World"	String with spaces. Without the quotes this will be treated as 2 string parameters.
Hello "World"	"Hello \"World\""	String with quotes. A quote must be embedded in a string with a backslash preceding it.
Test \123\ \Test\	"Test\\123\\" Or \Test\\	String with backslash. A backslash must be embedded in a string with a backslash preceding it. True whether or not quotes are used.
Test1 Test2	"Test1\0DTest2" Or "Test1\0DTest2"	String with non-printable characters. ASCII characters representing the Hex value of a non printable character must be used with a backslash preceding it. Up to Two ASCII hex characters after a backslash will always be treated as a hex value. **It is suggested that this value always contains 2 ASCII hex characters to avoid the possibility of misreading regular ASCII characters as part of the backslash hex value. Pad with leading zero if necessary.
Hello World □	"Hello |0174orld |4E13"	Unicode String. A string can contain all Unicode or mixed ASCII and Unicode characters. ASCII characters are sent as described above. ASCII characters representing the Hex value of a Unicode character must be used with a vertical bar preceding it. Up to four ASCII hex characters after a vertical bar will always be treated as a Unicode character. **It is suggested that Unicode characters always contain 4 ASCII hex characters even if it must be padded with zeros. This is to avoid the possibility of misreading regular ASCII characters as part of the Unicode character
Test|123| Or |Test|	"Test||23||" Or ||Test|	String with Vertical bar

```
Error definitions taken directly from the C source code.
CLI_ERR_INVALID_COMMAND                 -1
CLI_ERR_PARAMETER_TOO_LARGE             -2
CLI_ERR_PARAMETER_TOO_SMALL             -3
CLI_ERR_NOT_ENOUGH_PARAMETERS           -4
CLI_ERR_TOO_MANY_PARAMETERS             -5
CLI_ERR_NO_MATCH_OF_ENUM_STRING         -6
CLI_ERR_PARAMETER_OUT_OF_RANGE          -7
CLI_ERR_INVALID_INDEX                   -8
CLI_ERR_MIN_GREATER_THEN_MAX            -9
CLI_ERR_MAX_LESS_THEN_MIN               -10
CLI_ERR_INVALID_SECONDS                 -11
CLI_ERR_INVALID_MINUTES                 -12
CLI_ERR_INVALID_HOURS                   -13
CLI_ERR_INVALID_MONTH                   -14
CLI_ERR_INVALID_DAY                     -15
CLI_ERR_INVALID_YEAR                    -16
CLI_ERR_INVALID_ADDRESS                 -17
CLI_ERR_INTERNAL_CLOCK_ERROR            -18
CLI_ERR_NAME_CAN_NOT_BE_BLANK           -19
CLI_ERR_NAME_TOO_LONG                   -20
CLI_ERR_BLOCK_WAS_IN_PROGRESS           -21
CLI_ERR_NO_BLOCK_IN_PROGRESS            -22
CLI_ERR_MUST_BE_GREATER_THEN_0          -23
CLI_ERR_INVALID_RANGE                   -24
CLI_ERR_INTERNAL_ERROR                  -25
CLI_ERR_INVALID_PARAMETER               -26
```

CLI_ERR_INDEX_OUT_OF_RANGE	-27
CLI_ERR_INVALID_PASSWORD	-28
CLI_ERR_INVALID_PERMISSION	-29
CLI_ERR_INVALID_LINE_NUM	-30
CLI_ERR_LINE_TOO_LONG	-31
CLI_ERR_BAD_REGISTER_NUM	-32
CLI_ERR_BAD_REGISTER_DATA	-33
CLI_ERR_CANT_CHANGE_SER_NUM	-34
CLI_ERR_CANT_CHANGE_MODEL_NUM	-35
CLI_ERR_AUTO_MODE_LOCKOUT	-36
CLI_ERR_BLOCK_NUM_TOO_SMALL	-37
CLI_ERR_BLOCK_NUM_TOO_LARGE	-38
CLI_ERR_BAD_PARAMETER_TYPE	-39
CLI_CHECKSUM_ERROR	-40
CLI_ERR_XFER_BUFFER_LENGTH_EXCEEDED	-41
CLI_ERR_FONT_RESOLUTION_MISMATCH	-42
CLI_ERR_DIREC_CHANGE_WITH_IRS_ON	-43
CLI_ERR_INVALID_SPACE_NUM	-44
CLI_ERR_INVALID_SEGMT_NUM	-45
CLI_ERR_INVALID_INTERLOCK_ERROR_NUM	-46
CLI_ERR_INVALID_FONT_NUM	-47
CLI_ERR_FILE_SYSTEM_ERROR	See below
CLI_ERR_SL_TIMED_OUT	-49
CLI_ERR_BUFFER_TOO_LARGE	-50
CLI_ERR_ODD_NUM_OF_BYTES	-51
CLI_ERR_LASER_DEAD	-52
CLI_ERR_FILE_WRITE_IN_PROGRESS	-53
CLI_ERR_KEY_LOG_BUFFER_OVERRUN	-54
CLI_ERR_PLAYBACK_IN_PROGRESS	-55
CLI_ERR_PLAYBACK_FORMAT_ERROR	-56
CLI_ERR_ACTIVE_FILE_NOT_FOUND	-57

```
CLI_ERR_ACTIVE_FILE_NOT_READ        -58
CLI_ERR_OUT_OF_MEMORY               -59
CLI_ERR_CANNOT_RENAME_FILE          -60
CLI_ERR_CANNOT_UPLOAD_FILE          -61
CLI_ERR_CANNOT_DOWNLOAD_FILE        -62
CLI_ERR_CANNOT_DELETE_FILE          -63
CLI_ERR_CANNOT_ACTIVATE_USER_VARS   -64
CLI_ERR_ACTIVATE_JOB_NEW_MODE       -65
CLI_ERR_PRINT_ACTIVE_JOB            -66
CLI_ERR_DBS_KEY_NOT_FOUND           -67
CLI_ERR_DBS_NO_SPACE_LEFT           -68
CLI_ERR_DBS_KEY_EXISTS              -69
```

File system error codes:

FLASH_VOLTAGE_ERROR	-501
FLASH_LOCKING_ERROR	-502
FLASH_LOCKED_ERROR	-503
FLASH_PROGRAM_SUSPEND_ERROR	-504
FLASH_ERASE_SUSPEND_ERROR	-505
FAT_FLASH_FULL	-506
FLASH_SAVING_REENTRANCY_ERROR	-507
FLASH_BUFFER_SIZE_ERROR	-508
FILE_BUFFER_FULL	-509
FILE_FLASH_FULL	-510
SL_FILE_INVALID	-511
FILE_NOT_FOUND	-512
FLASH_RESPONSE_TIMEOUT	-513
FILE_HEADER_INVALID	-514
FILE_COPY_DELETED	-515
END_OF_DIR	-516
FILE_NAME_TOO_LARGE	-517

File type number definitions:

File Type Number	File Type String	File Type	Maximum Characters In File Name	Stored Location	Comments
1	xlb	Bitmap	10	UI	
2	dkf	Dead Key	25	UI	
3	ldf	Local Keyboard	--	UI	Must be named "Keyboard"
4	xlf	Font	25	UI/Laser	Two fonts exist in UI and must be named "16x10" & "Abomination". Laser must contain the file "Abomination".
5	var	User Variables	25	Laser	
6	hin	Hint	15	Laser	
7	tpl	Job Template	--	Laser	Must be named "Template1"
8	job	Normal Job	15	Laser	
9	pff	Pre-Defined Formats	"Formats"	Laser	Must be named "Formats"
10	udw	User Defined Day of Week	25	Laser	
11	usc	User Defined Seconds	25	Laser	
12	syr	User Defined Year	25	Laser	
13	umt	User Defined Month	25	Laser	
14	udm	User Defined Day of Month	25	Laser	
15	udy	User Defined Day of Year	25	Laser	
16	uwy	User Defined Week of Year	25	Laser	
17	uhr	User Defined Hour - 24	25	Laser	

18	shm	User Defined Hour – 12	25	Laser	
19	smn	User Defined Minutes	25	Laser	
20	sap	User Defined Am-Pm	25	Laser	
21	ssh	User Defined Shift Codes	25	Laser	
22	pdt	Power Distribution Test	--	Laser	Must be named "PDT"
23	scn	Screen	20	UI	
24	pwd	Login Password	--	UI	Must be named "Login"
25	cdt	Contrast	--	UI	Must be named "Contrast"
26	dtd	Database in Time Date	--	--	
27	gph	Graphic	15	Laser	
28	snd	Sequence Number Default	15	Laser	
29	dba	Database File	25	Laser	
30	lan	Language & Unit File	25	Laser	Name of file determines installed firmware language and unit. Name of file is Language-Unit.lan e.g. USEnglish-Imperial.lan, Chinese-Metric.lan
31	did	Device ID list	"IDFile"	UI	Name and device ID for multi-drop lasers
32	smd	User Defined Minute of Day	25	Laser	
33	sfe	System Files Installed via Firmware Image	"SystemFile"	Laser	Name unchanged

ACTJOB

Purpose:	Will load or return the name of the job currently active	
Minimum/Maximum Parameters:	0/1. Name of the job to activate. Example "Enjoy By".	
Return Parameters	Type	String
	Format	"XXXXX" ACSII characters representing the name of the job. "" If no job loaded
	Example	"Enjoy By"<CR>
Error Code Return Value(s):	0 CLI_OK -5 CLI_ERR_TOO_MANY_PARAMETERS -57 CLI_ERR_ACTIVE_FILE_NOT_FOUND -58 CLI_ERR_ACTIVE_FILE_NOT_READ -59 CLI_ERR_OUT_OF_MEMORY	
	-65 CLI_ERR_ACTIVATE_JOB_NEW_MODE	
Example:	SL1> ACTJOB<CR> "Enjoy By" // The job that is currently active SL1> SL1> ACTJOB "Batch 1"<CR> // This will activate "Batch 1" <CR> SL1>	

LMODE

Purpose:	This will set or return the current laser mode. Note, this command does not guarantee the laser will transition to the specified mode immediately. The POLL or LMODE command can be used to ensure the mode transition occurred. A mode transition that does not occur within ~40seconds is considered an error.	
Minimum/Maximum Parameters:	0/1. Name of the mode to put the laser in, case-insensitive. "RUN", "DEMO", "STANDBY"	
Return Parameters	Type	String
	Format	"XXXX" ASCII Characters representing the mode of the laser:
	Example	"RUN" or "DEMO" or "STANDBY"
Error Code Return Value(s):	0 CLI_OK -5 CLI_ERR_TOO_MANY_PARAMETERS	
Example:	SL1>LMODE "Run"<CR> // Signal the laser to transition to RUN mode <CR> SL1> LMODE<CR> // return the current mode the laser is in DEMO <CR> SL1>	

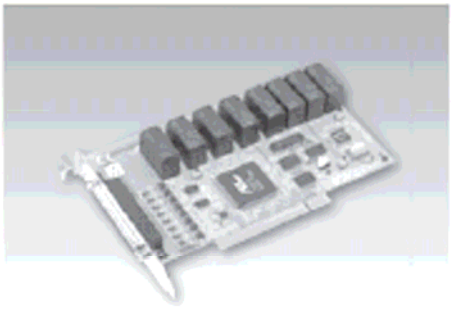
DBS

Purpose:	Will set or return a database value given a database key	
Minimum/Maximum Parameters:	1/2 String: ASCII or UNICODE key. ASCII example: "Line", UNICODE example: |12AB|A12F String: ASCII or UNICODE value: ASCII example "1234567", UNICODE example: |02BD|12EC	
Return Parameters	Type	String, ASCII or UNICODE
	Format	"XXXX" for ASCII or |XXXX|XXXX|XXXX for UNICODE (Note no spaces between Unicode values).
	Example	See above
Error Code Return Value(s):	0 CLI_OK -5 CLI_ERR_TOO_MANY_PARAMETERS -67 CLI_ERR_DBS_KEY_NOT_FOUND	
Example:	Retrieving data SL1> DBS "Database Key" <CR> Database content <*CR*> Setting data. Note the data maximum length is set at the time of the database creation. SL1> DBS "Database Key" "Database content" <CR> <*CR*>	

ANNEX 2: Datasheet targeta entrades sortides

PCI-1761

8-ch Relay Actuator/Isolated DI PCI Card



Features

- 8 relay output channels and 8 isolated digital input channels
- LED indicators to show activated relays
- 4 Form C and 4 Form A type relay output channels
- Male DB37 matching connector included
- Output status read-back
- Retained relay output values when hot system reset
- High-voltage isolation on input channels (3,750 V_{DC})
- High ESD protection (2,000 V_{DC})
- High over-voltage protection (70 V_{DC})
- Wide input range (10 ~ 50 V_{DC})
- Universal handling capability
- Board ID™ switch

Introduction

The PCI-1761 relay actuator and isolated DI card is an add-on card for the PCI bus. It provides 8 optically-isolated digital inputs with isolation protection of 3,750 V_{DC} for collecting digital inputs in noisy environments and 8 relay actuators for serving as on/off control devices or small power switches. For easy monitoring, each relay is equipped with one red LED to show its on/off status. The PCI-1761's eight optically-isolated digital input channels are ideal for digital input in noisy environments or with floating potentials.

The PCI-1761 digital input channels feature a rugged isolation protection for industrial, lab and machinery automation applications. It durably withstands voltage up to 3,750 V_{DC}, protecting your host system from any incidental harms. If connected to an external input source with surge-protection, the PCI-1761 can offer up to a maximum of 2,000 V_{DC} ESD (Electrostatic Discharge) protection. Even with an input voltage rising up to 70 V_{DC}, the PCI-1761 can still manage to work properly, albeit for only a short period of time.

When the system has a catastrophic hot reset (i.e. without turning off the system power), the PCI-1761 can either retain output values of each channel, or return to its default configuration as open status, depending on its onboard jumper setting. This function protects the system from unwanted operations during unexpected system resets.

Specifications

Isolated Digital Input

- **Channels** 8
- **Input Voltage** Logic 0: 3 V max.
  Logic 1: 9V min. (50V max.)
- **Interrupt Capable Ch.** 8
- **Isolation Protection** 3,750 V_{DC}
- **Overvoltage Protection** 70 V_{DC}
- **Opto-Isolator Response** 25 µs
- **Input Resistance** 5500 Ω
- **Input Current** 1.6 mA @ 10 V_{DC}, 8.3 mA @ 50 V_{DC}

Relay Output

- **Channels** 8
- **Relay Type** SPDT (4 x Form C and 4 x Form A)
- **Contact Rating** 250 V_{AC} @ 3 A, or 24 V_{DC} @ 3 A
- **Relay on Time** 15 ms max.
- **Relay off Time** 5 ms max.
- **Life Span** 2 x 10^7
- **Resistance**
  Contact: 50 MΩ
  Insulation: 1 GΩ min.

General

- **Bus Type** PCI V2.2
- **I/O Connectors** 1 x 37-pin D-type
- **Dimensions (L x H)** 175 x 100 mm (6.9" x 3.9")
- **Power Consumption** Typical: +5V @ 220 mA
  Max: +5 V @ 750 mA
- **Operating Temperature** 0 ~ 60° C (32 ~ 140° F) (IEC 68-2-1, 2)
- **Storing Temperature** -20 ~ 70° C (-4 ~ 158° F)
- **Storing Humidity** 5 ~ 95 % RH, non-condensing (IEC 68-2-3)

Ordering Information

- **PCL-1761** 8-ch Relay Actuator/Isolated DI PCI Card
- **PCL-10137-1** DB37 cable assembly, 1 m
- **PCL-10137-2** DB37 cable assembly, 2 m
- **PCL-10137-3** DB37 cable assembly, 3 m
- **ADAM-3937** DB37 Wiring Terminal for DIN-rail Mounting
- **PCLD-888** Universal screw terminal board