UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

---

ESTUDI DISSENY I INTEGRACIÓ DE DOS ROBOTS DE DIFERENT TOPOLOGIA DINS D'UNA ILLA DE PROCÉS

TITULACIÓ

Grau en Enginyeria Electrònica Industrial i Automàtica

ALUMNE

Xavier Pujadas i Castells

DIRECTORA DEL TFG

Rita Maria Planas Dangla

CONVOCATÒRIA DE LLIUREMENT DEL TFG

Octubre 2016

Contingut

Índex de figures

Índex de taules

1 OBJECTIUS

En aquest treball final de grau s'ha realitzat un estudi per implementar en una cel·la de treball dos robots de diferents morfologies de manera que produeixin de forma conjunta. Els dos robots estarien integrats en una cel·la on es complementarien per automatitzar un procés d'embalatge i posterior paletitzat de dispositius mòbils.

Per resoldre aquesta idea principal s'han fixat els següents subobjectius:

- Adaptació d'un control superior a un dels robots.
- Programació individual dels dos robots per la tasca a realitzar.
- Disseny i implementació del sistema de comunicació entre robots.
- Implementació d'una aplicació de supervisió i control del sistema global.
- Posta en marxa de la cel·la de treball.

2 INTRODUCCIÓ

Aquest treball final de grau està dividit en tres parts principals degut a que ha estat la continuació d'un projecte previ el que ha donat peu a la realització d'aquest estudi. Així doncs, com es podrà veure al llarg d'aquesta memòria, l'estructura seguida en els diferents apartats es basarà en l'explicació de dues aplicacions completament independents i finalment la unió d'aquestes.

La primera part d'aquest treball va ser impulsada en format de concurs per l'empresa FESTO Automation S.A.U. i la Universitat Politècnica de Catalunya. El concurs va ser batejat amb el nom "Primer concurs d'automatització E-Motion" i tenia com a objectiu desenvolupar un prototip que pogués realitzar una funció real i amb la possibilitat de ser aplicable en qualsevol àmbit.

La intenció de l'empresa era aconseguir, per part dels alumnes, uns projectes innovadors, emprenedors i amb capacitat d'aplicació en l'àmbit industrial. Utilitzant la competició com a base, l'empresa volia fomentar el treball en equip i veure com treballaven els estudiants.

Per al desenvolupament del projecte els equips participants disposaven d'un manipulador elèctric cedit per FESTO i un pressupost per part de la EET de 125€ per invertir en el prototip. A més, per facilitar la realització del projecte als equips, la universitat va destinar una aula de l'escola exclusiva per al seu desenvolupament.

Arrel del primer projecte es va proposar per part de la universitat la possibilitat de continuar en el desenvolupament d'aquest i utilitzar-ho com a treball final de grau. Així doncs, amb una decisió conjunta amb la tutora del treball es va optar per completar el procés productiu que realitzarà el Gantry. Per fer-ho s'ha utilitzat un dels braços robòtics que disposa la universitat i que es troba al laboratori de robòtica i CIM.

Per conèixer els dos elements principals amb el qual s'ha desenvolupat el treball a continuació es mostra una breu presentació dels robots utilitzats.

2.1 CONEIXENT AL MANIPULADOR MINI-H GANTRY EXCM-30

El robot que l'empresa va cedir a cadascun dels equips era un manipulador elèctric Mini-H Gantry com el que podem veure a la Figura 1. Els robots *gantries* o pòrtics són un tipus de robots inclosos en el grup dels cartesians que es caracteritzen per fer moviments de translació en els eixos de coordenades cartesianes XYZ.

El model de robot concret utilitzat en aquest treball és el EXCM-30 que és el robot més gran dintre la gama Mini-H Gantry amb unes dimensions de 300x260mm. El EXCM-30 està format per dos eixos XY controlats elèctricament i governats, a través de dos motors pas a pas, per la controladora de la pròpia casa FESTO. Aquest robot suporta una carga màxima de 3kg i pot arribar a una velocitat màxima de moviment de 0.5m/s.

Figura 1: Robot Mini-H Gantry EXCM-30 de FESTO

Aquest pòrtic en concret està pensat principalment per a aplicacions d'assemblatge de peces reduïdes com poden ser components electrònics o també per a laboratoris mèdics automatitzats. Aquestes aplicacions que consisteixen bàsicament en moure objectes d'un punt a un altre es coneixen com a aplicacions del tipus "picking" o "pick and place". El Gantry és un robot pensat per a aquest tipus d'aplicacions ja que actualment s'utilitza en laboratoris químics per manipular mostres i també per a la inserció de components en plaques electròniques.

2.2 CONEIXENT AL ROBOT IRB 140

El braç robòtic IRB 140 de la casa ABB és un robot antropomòrfic industrial articulat de 6 eixos capaç de treballar amb una càrrega de 6kg i a una velocitat màxima de 2.5m/s. Un robot que tingui moviment en els 6 eixos pot moure's en els 3 eixos XYZ i rotar sobre d'aquests. El IRB 140 destaca dins les seves especificacions per la seva rapidesa, fiabilitat, robustesa i repetitivitat de moviment entre altres.

A continuació tenim el llistat de les principals aplicacions a les quals es destina el IRB 140 i on podem veure la gran versatilitat d'aquest robot.

- Soldadura al arc
- Muntatge/Assemblatge
- Neteja
- Polvorització
- Mecanització
- Manipulació de materials
- Paletitzat

Aquest robot es caracteritza per la seva gran versatilitat ja que pot treballar muntat en qualsevol posició i disposa d'un grau de protecció IP67 cosa que el fa útil en una gran varietat d'aplicacions. El mateix IRB140 es pot trobar en diferents versions que el doten de diferents funcionalitats i proteccions depenent de quina destinació rebi el robot dins el marc del llistat anterior.

Figura 2: Controladora IRC5 amb FlexPendant i el robot IRB 140 de ABB

El laboratori on s'ha desenvolupat el treball disposa de dos robots IRB140, dels quals s'ha escollit l'Obèlix ja que està governat per una controladora més nova i una interfície HMI també més actualitzada que el seu company Astèrix. La controladora IRC5 és capaç de controlar el propi robot i fins a sis eixos externs com per exemple una cinta transportadora. Aquesta controladora ha estat dissenyada pensant en la seguretat per a la interacció amb persones i optimitzada per millorar els temps de cicle i moviments del robot. Tot això és possible gràcies al llenguatge de programació RAPID en què es basa la controladora i que també és producte de la casa ABB.

Per altra banda, la IRC5 va acompanyada per una interfície HMI anomenada FlexPendant que permet a l'usuari interactuar amb el robot a través d'un joystick 3D i una pantalla tàctil.

3 ABAST DEL TREBALL

En aquest apartat es detalla quines decisions s'han hagut de prendre al llarg del desenvolupament de l'estudi i quines han vingut imposades per diferents motius com per exemple la limitacions dels materials i elements utilitzats, o bé, per indicacions de la tutora.

Per altra banda, l'aplicació inicial desenvolupada amb el Gantry i que va donar lloc a la realització del treball, ha estat desenvolupada per un grup de tres persones arrel del concurs E-Motion. Tot i això, per a la realització del treball final de grau, aquesta primera part ha sigut modificada en diferents aspectes tant de hardware com de software per tal de millorar el seu funcionament i poder-la integrar en un sistema comú juntament amb el braç robòtic de ABB.

3.1 PROJECTE E-MOTION

3.1.1 NECESSITATS DEL MINI H-GANTRY

La controladora que incorpora el Gantry només és capaç de controlar el manipulador per si sol, movent el carro del robot en el pla XY. Per tant, en cas d'utilitzar altres elements controlables s'hauria de buscar obligatòriament un nivell superior de control per al sistema. Al ser una aplicació de desenvolupament lliure, s'ha pogut escollir el controlador que s'ha cregut més adient i també el protocol utilitzat entre els dos nivells de comunicació.

Degut a la possibilitat de complementar el robot, poder fer les modificacions necessàries i l'aplicació escollida, ha sorgit la necessitat d'incloure diferents elements al robot original. Els elements externs que complementen el Gantry per arribar a l'aplicació final també s'han escollit, muntat i fabricat amb total llibertat i decisió dels integrants del grup.

Així doncs, exceptuant el robot utilitzat, el desenvolupament de l'aplicació de packaging inclou el disseny i elecció de tots els seus components, el sistema de control i programació, l'elecció de la pròpia aplicació i la integració de totes aquestes parts en un sistema conjunt i funcional.

Dintre l'aplicació escollida s'ha exclòs del procés de packaging els auriculars i els adaptadors per a diferents mides d'orella que solen incloure aquests dispositius.

3.2 EL TREBALL FINAL DE GRAU

Com a treball final de grau es va proposar inicialment desenvolupar un sistema de supervisió i control per a l'aplicació del Gantry, però de seguida va sorgir una segona opció més atractiva per a realitzar el treball. Aquesta consistia en complementar el procés del EXCM utilitzant el braç robòtic que disposa el laboratori i d'aquesta manera fer una aplicació final conjunta entre dos robots totalment diferents. El control de les dues aplicacions sí que es basaria en un sistema de supervisió comú per a les dues aplicacions.

3.2.1 NECESSITATS DEL BRAÇ ROBÒTIC DE ABB

Degut a la complexitat de la controladora IRC5 no s'ha hagut de fer cap tipus d'adaptació al controlador del robot. A més el robot està complert físicament de manera que tampoc s'ha necessitat disposar de cap element extern que complementés el seu moviment com en el cas del Gantry.

La funció que realitza aquest robot també s'ha desenvolupat en total llibertat excepte la programació d'alguna part molt concreta del procés. Per indicacions de la tutora, per definir les posicions i facilitar la programació en futures modificacions s'ha utilitzat la programació mitjançant els objectes de treball, que es detallarà més endavant.

El software utilitzat s'ha programat a través del llenguatge de programació RAPID que és l'únic que la controladora és capaç de processar.

3.2.2 APLICACIÓ CONJUNTA

La base d'aquest estudi ha estat la interconnexió entre dos robots totalment diferents des de la topologia fins als protocols interns de comunicació controlador-màquina que tenen cadascun d'ells.

A l'hora de fer una aplicació conjunta per poder coordinar els dos robots s'ha necessitat un enllaç entre els dos controladors de cada robot. Juntament amb la tutora s'ha decidit que aquest enllaç es faria mitjançant una aplicació de supervisió i control. Com a software per a implementar el l'aplicació no hi ha hagut alternativa al Proficy iFix de GE Fanuc, que és el que disposa el laboratori on s'ha desenvolupat l'aplicació.

Per tant, independentment del software utilitzat, les funcions que realitza l'aplicació de supervisió i control s'han escollit de forma totalment lliure incloent i excloent tots els elements que s'ha cregut.

4 ANTECEDENTS

En l'àmbit de la robòtica es poden diferenciar dos grups principals com són la branca industrial i la branca de servei. La robòtica de servei inclou tots aquells robots pensats per a satisfer necessitats humanes en diferents ambients o simplement per aportar comoditat. En l'altre extrem, la robòtica industrial està dedicada exclusivament als processos productius i va ser la primera en existir.

És difícil definir un inici concret en el món de la robòtica ja que la paraula robòtica va sorgir a partir dels sistemes autònoms de control. La primera màquina autònoma productiva apareix l'any 1801 quan Jaques Jaquard inventa una màquina tèxtil programable mitjançant targetes perforades. A partir d'aquesta data comença l'evolució de diferents automatismes en diferents àmbits de la industria. Tot i això, no és fins l'any 1961 que George Devol instal·la en una cadena de muntatge a l'empresa de General Motors el que es considera el primer robot industrial fabricat en sèrie. Aquest robot articulat anomenat "Unimate" transportava planxes i les soldava a l'estructura dels vehicles.

Juntament amb la millora de materials i sistemes de control s'han pogut aconseguir robots més petits, ràpids, potents, precisos i amb una enorme quantitat de funcions. Fins a l'actualitat els robots han evolucionat en qualsevol àmbit i forma imaginable fins a arribar als robots humanoides actuals. L'exemple més actual a data d'avui d'aquests robots és "Erica", un robot d'una similitud humana extraordinària sorgit d'una col·laboració entre la universitat de Kyoto i Osaka. L'Erica no es caracteritza per el seu moviment sinó per la capacitat de mantenir una conversar amb una persona humana i la realitat de les seves expressions facials.

Figura 3: Robot industrial Unimate i robot de servei Erica

5 ESPECIFICACIONS DEL TREBALL

Com s'ha anat introduint en els apartats anteriors, la finalitat de l'aplicació desenvolupada era crear un ambient de treball comú dins una cel·la de procés on dos robots de diferents topologies treballin conjuntament i es complementin l'un amb l'altre compartint informació.

Per poder fer aquesta aplicació conjunta prèviament s'han desenvolupat dues aplicacions independents, una per cada robot, i finalment s'ha integrat en una aplicació única. Aquest conjunt formarà un sistema on un primer robot realitzarà un packaging de mòbils encapsant tots els components com són el carregador, les instruccions i el propi mòbil dins una caixa. A continuació, els mòbils arribaran a un segon robot que es cuidarà de paletitzar-los en dues línies segons el format escollit prèviament per l'usuari. Per a poder interactuar entre els dos robots s'ha desenvolupat un sistema SCADA (*Supervisory Control And Data Acquisition)* que permet a l'usuari interactuar amb les dues màquines des d'un ordinador central.

5.1 SISTEMA DE PACKAGING

L'aplicació dissenyada per al packaging de mòbils s'ha desenvolupat amb el robot Mini H-Gantry. El disseny de fabricació d'aquest manipulador permet un muntatge qualsevol sempre que es mantingui la forma de "H" que formen els eixos. Això vol dir que el podem posar tant vertical com horitzontal i amb els motors a la part superior o inferior i inclús girar-los sobre si mateixos per escollir la direcció de la sortida dels cables. En el nostre cas s'ha utilitzat un muntatge en posició horitzontal amb els motors a la part inferior com s'observa en la següent imatge.

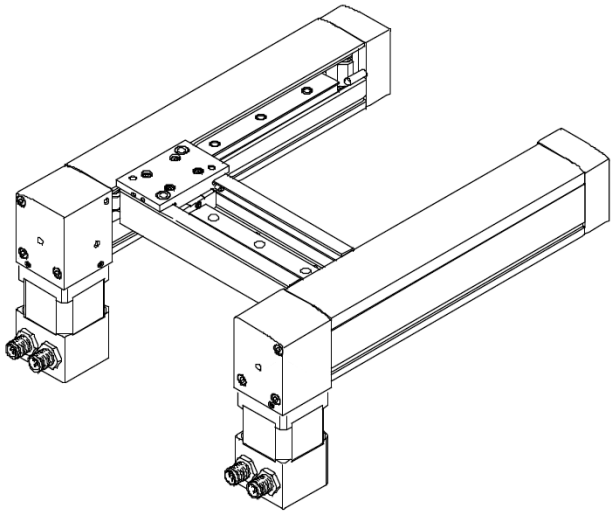


Figura 4: Muntatge del manipulador Mini-H Gantry

L'aplicació desenvoluparà un procés de packaging complert d'un dispositiu mòbil o smartphone amb les característiques següents:

- L'usuari de l'aplicació podrà iniciar o aturar el procés de packaging des del sistema de supervisió SCADA.
- El procés serà continu fins a arribar al total de dispositius demanats per el mateix sistema. Una vegada fet el packaging de la demanda l'aplicació s'aturarà automàticament.
- Si es produeix una aturada del packaging i una següent reactivació, l'aplicació seguirà el procés d'empaquetatge que realitzava en aturar-se l'aplicació.

L'aplicació del Gantry s'ha governat a través d'un microcontrolador extern a la controladora del robot anomenat Arduino. Per tant, serà aquest controlador qui gestionarà els moviments del Gantry i totes les dades del procés de packaging. A més, s'encarregarà de coordinar els moviments del robot EXCM-30 amb un tercer eix que s'ha afegit per poder manipular tots els components a encapsar.

Finalment, una vegada processat un mòbil, si no s'ha donat una ordre contrària, el robot començarà a fer el packaging del següent mòbil formant així un procés cíclic continu.

5.2 SISTEMA DE PALETITZAT

En l'aplicació de paletitzat el braç robòtic de ABB executa un paletitzat en dues línies diferents de les peces provinents del Gantry. A més, serà l'usuari el que donarà les ordres al robot indicant la quantitat de peces que vol produir en cada línia mitjançant una aplicació SCADA. Aquests tipus d'aplicacions són molt utilitzades en la industria i es caracteritzen per les funcions representades per les seves sigles en anglès *Supervisory Control And Data Acquisition*. En poques paraules és una aplicació software destinada al control d'una o vàries màquines dins una àrea productiva i permet capturar i representar dades dels sensors i actuadors d'aquestes màquines.

S'ha separat la forma de paletitzar el mòbils en dos tipus de paletitzat que contenen el mateix nombre de smartphones, la diferència entre els dos paletitzats es troba en la forma com es distribueixen els mòbils dins el palet una vegada fet el packaging.

En el primer cas s'optimitza l'espai del paletitzat al màxim compactant els mòbils entre ells evitant espais buits dintre el palet final. Aquest paletitzat consisteix en fer quatre pisos de quatre smartphones cadascun. En la següent imatge es mostra un mosaic en planta d'aquest paletitzat per fer-nos una idea de la disposició que anomenarem **quadrada**.

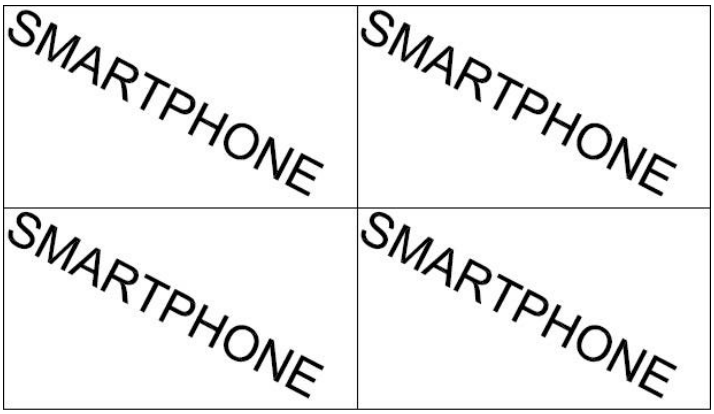


Figura 5: Mosaic en planta del paletitzat en forma quadrada

En el segon paletitzat, s'ha utilitzat una disposició similar a l'anterior permetent també apilar quatre pisos de quatre smartphones cadascun. La diferència en la disposició que podem veure a la Figura 6, deixa una columna entre mòbils enmig del paletitzat. A la columna buida s'hi podria destinar un element de protecció per assegurar el transport. Aquesta forma de paletitzat rebrà el nom de **circular**.

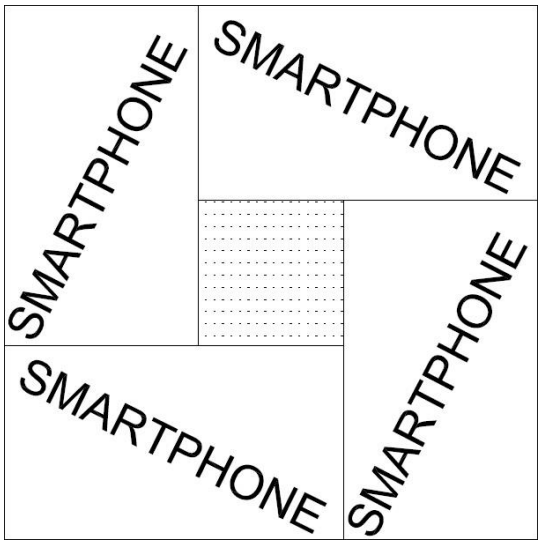


Figura 6: Mosaic en planta del paletitzat en forma circular

A part del sistema de paletitzat aquesta aplicació també permetrà al usuari interrompre el paletitzat perquè el robot enviï 2 smartphones del *transfer* d'entrada a control de qualitat. Mitjançant una altra intervenció, també indicada per l'usuari, es podrà enviar a control de qualitat el mòbil que en aquell moment es troba en l'estació del Gantry. Això significa que l'usuari pot indicar que vol enviar a control de qualitat l'smartphone que actualment es troba fent el procés de packaging. Per fer-ho el robot memoritzarà el numero de mòbil que es troba en aquell moment a l'estació del Gantry, i quant arribi l'hora del paletitzat de l'smartphone

concret el robot l'enviarà directament a control de qualitat i seguirà paletitzant la resta de dispositius.

Per últim, la mateixa aplicació també treballa amb un arxiu extern al programa on emmagatzema un seguit d'històrics del paletitzat de cada smartphone com són l'hora en que s'ha processat, la línia, el tipus de paletitzat que ha rebut i el temps que el robot ha estat manipulant cada mòbil. Aquest arxiu, al ser extern i en format Excel pot ser exportat i visualitzat per qualsevol persona que tingui accés al directori del robot que és on es guarda el fitxer.

La intenció principal d'aquesta part del treball és demostrar com es pot gestionar un procés de paletitzat de mòbils en caixes de 16 dispositius. Per dur-lo a terme s'ha dotat a l'aplicació de les funcions i característiques següents:

- El desenvolupament de l'aplicació té com a finalitat el control des d'un sistema de supervisió que serà controlat en tot moment per un usuari.
- Cada paletitzat permet un màxim de 16 dispositius amb les disposicions abans explicades.
- Una vegada finalitzat un palet de 16 dispositius es realitzarà un canvi automàtic i el procés serà indicat a l'usuari a través d'un indicador per la pantalla de l'aplicació SCADA.
- L'usuari pot demanar en qualsevol moment un control de qualitat dels dispositius del buffer i el robot agafarà i apartarà un dispositiu per a cada línia situant-lo en la posició de control de qualitat corresponent.
- L'usuari podrà indicar en qualsevol moment un control de qualitat de l'estació de packaging. El robot gestionarà l'extracció d'aquest dispositiu en funció de les peces en el buffer d'entrada de les línies. En el moment de l'extracció es deixarà l'smartphone en la situació del control de qualitat de la línia 1.
- Els dispositius enviats a control de qualitat no es gestionaran més.
- El tipus de paletitzat habitual ve determinat en funció de la línia escollida.
- Si la demanda per una de les dues línies és nul·la, es demana a l'usuari si es vol desdoblar la producció. En cas afirmatiu, la demanda es repartirà en les dues línies canviant el tipus de paletitzat de la línia a desdoblar perquè coincideixi amb la forma de paletitzat on s'ha realitzat la demanda.
- La diferència de percentatge acceptat entre les línies fa que el paletitzat s'equilibri donant prioritat a la línia amb més demanda.
- S'executarà una interrupció cíclica configurable que posarà el robot en estat de manteniment en acabar el paletitzat que es realitza en el moment de la interrupció. En

acabar el manteniment l'usuari indicarà a l'aplicació la finalització del manteniment i l'aplicació continuarà el paletitzat programat.

- En acabar la paletització actual el robot es situarà en posició d'espera per iniciar una nova demanda.

5.3 SISTEMA GLOBAL

La interconnexió entre els dos robots s'ha fet utilitzant un ordinador com a enllaç de manera que podem trobar dues comunicacions independents amb aquest ordinador. Una serà amb el controlador Arduino que governarà el Gantry i l'altra amb la controladora IRC5 del IRB 140. L'aplicació SCADA estarà dins aquest ordinador i serà la que permetrà realment l'intercanvi d'informació entre màquines. Aquest control de la cel·la des d'un únic ordinador mitjançant un SCADA ens indica que la gestió i supervisió de la producció global de packaging i paletitzat serà en forma de sistema centralitzat.

Per poder fer la comunicació independent entre cadascun dels robots i el sistema SCADA s'ha utilitzat el protocol via OPC que utilitzen aquests sistemes. La paraula OPC prové de *OLE for Process Control* o *Object Linking and Embedding for Process Control*. L'OPC és un protocol estàndard de Microsoft que permet una comunicació entre softwares individuals del mateix fabricant o no, per tal de que puguin interactuar i compartir dades. Aquest protocol és tan utilitzat degut a que ha sigut la substitució dels protocols propietaris que obligaven a tenir els *drivers* de cada producte. Actualment, pràcticament tots els fabricants ja integren l'estàndard OPC en els seus productes.

Des del punt de vista del Mini-H Gantry, aquest ha estat complementat amb un eix vertical i una pinça. Per tant, la totalitat del sistema format pel pòrtic, el tercer eix i l'element terminal està governat per un controlador Arduino ja que la controladora EXCM-30 que incorporava el Gantry no era capaç de gestionar elements externs al manipulador. Per això l'element que s'ha comunicat via OPC amb l'aplicació SCADA ha sigut el controlador Arduino i no la controladora original del Gantry. Per fer-ho s'ha instal·lat a l'ordinador el servidor *Arduino OPC* que comunica amb un protocol sèrie. Tot i això, la controladora sí que s'ha hagut de parametritzar prèviament amb una connexió directa per Ethernet amb l'ordinador principal.

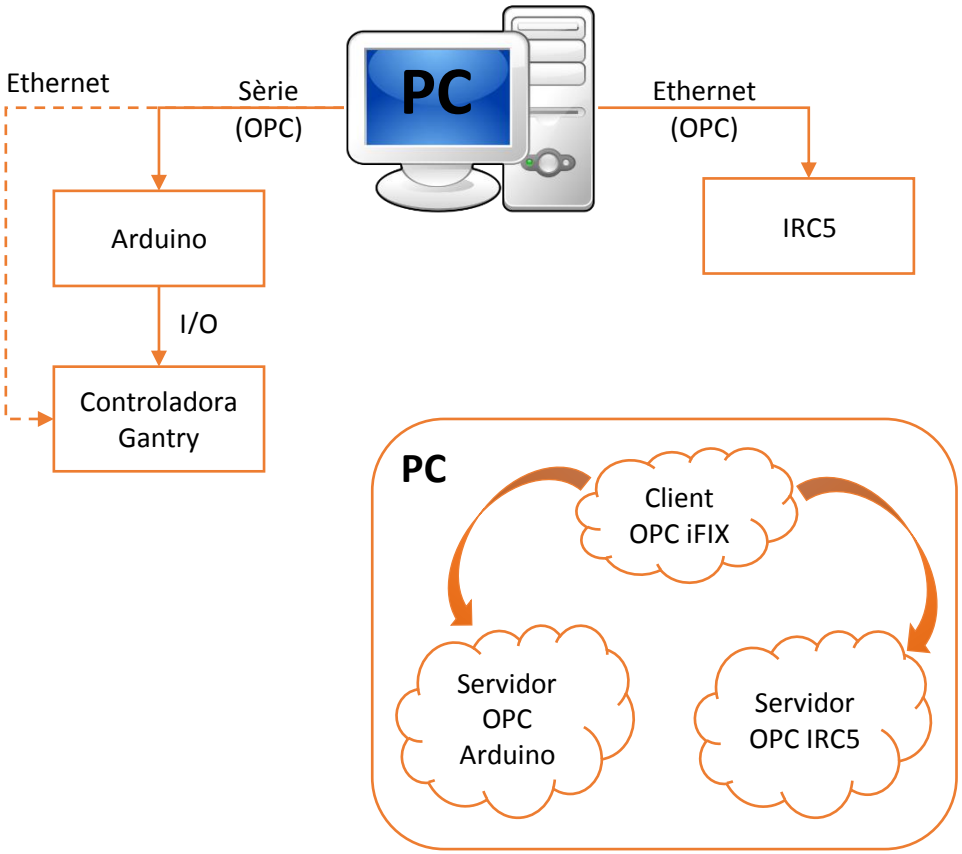
Durant el funcionament de l'aplicació, el controlador Arduino comunicarà amb la controladora del Gantry mitjançant entrades i sortides digitals.

En l'altra banda, la controladora IRC5 també s'ha connectat a l'ordinador mitjançant el protocol OPC. En aquest cas, el servidor utilitzat ha estat el de la pròpia casa ABB anomenat *ABB OPC* que comunica amb un protocol Ethernet.

Les comunicacions amb l'ordinador central permetran al sistema SCADA visualitzar i introduir les dades de producció corresponents a cada robot. Introduïdes les dades, es permetrà activar independentment cadascun dels dos sistemes de packaging i paletitzat. Tot i això, el

funcionament de les dues aplicacions es realitzarà en funció de les dades introduïdes prèviament.

L'explicació descrita anteriorment es pot resumir de forma visual en l'esquema següent:



6 ENTORN DE TREBALL

L'entorn de treball d'una aplicació defineix el conjunt d'elements que la formen. Des del punt de vista de l'usuari, l'entorn vindria a ser tot allò que envolta aquest en l'espai de l'aplicació. En un sistema automatitzat podem dividir l'espai de treball segons l'entorn físic, definit com a hardware, i per altra banda, l'entorn no tangible que seria el software.

L'entorn de programació del software es sol definir com a IDE, de l'anglès *Integrated Developement Environment*. Aquests entorns són les interfícies que permeten programar el software d'una manera més accessible per al programador.

En aquest apartat podrem veure de forma detallada tant la comunicació de hardware com de software, que coexisteix entre els diferents dispositius que formen l'aplicació. D'aquesta manera ens podrem fer una idea complerta del funcionament i la jerarquia del sistema.

6.1 ENTORN DEL MINI-H GANTRY

Com podem veure a la Figura 7, l'aplicació dissenyada pel manipulador Gantry té una jerarquia centralitzada en el controlador Arduino UNO. Aquest governarà tant a la controladora que fa moure al Mini-H Gantry com als dos servomotors que mouen l'eix vertical i la pinça. D'aquesta manera, amb les modificacions explicades, passem a tenir un robot cartesià tridimensional amb capacitat per poder agafar i moure objectes dins la superfície útil del robot.

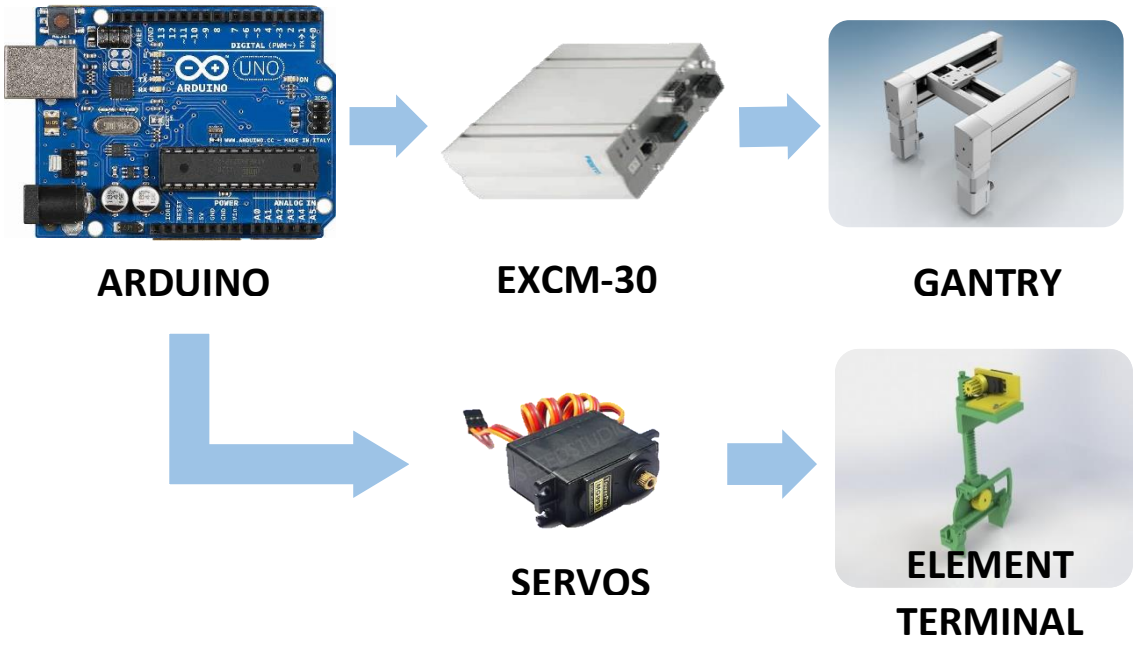


Figura 7: Esquema hardware del Mini-H Gantry

Després del muntatge, caldrà gestionar els moviments que permetran la moguda d'objectes i per això és necessari coordinar les ordres que reben els actuadors a l'hora de fer els moviments. Aquesta coordinació es pot fer programant els controladors que manipulen les parts mòbils.

La controladora EXCM-30 té un software de programació propi de la marca anomenat Festo Configuration Tool (FCT) en el qual es determinen característiques del robot com les dimensions d'aquest i el tipus de motor. Un cop parametritzades les especificacions del robot el mateix programa permet omplir una taula amb les posicions desitjades. Aquesta taula està formada, a part de la posició en XY, per la velocitat i acceleració que volem que es mogui el robot. Cadascuna de les posicions del robot amb els paràmetres que la defineixen reben el nom de **frases**. Un cop definides, s'envien les frases a la controladora del robot on quedaran guardades i posteriorment es triarà el punt al que volem que es desplaci el robot seleccionant la frase desitjada.

Com ja hem anat dient, la EXCM-30 estarà governada a un nivell superior per el nostre Arduino UNO que també té el seu propi entorn de programació Arduino. Aquest controlador seleccionarà la posició de la taula de la controladora mitjançant la comunicació per Entrades/Sortides i posteriorment donarà l'ordre de marxa per realitzar el moviment. A més, també serà aquest l'encarregat de coordinar els moviment amb l'eix vertical.

D'aquesta manera, si fem una traducció de l'esquema hardware anterior a un esquema de software per a l'aplicació del Gantry ens quedaria de la següent manera.

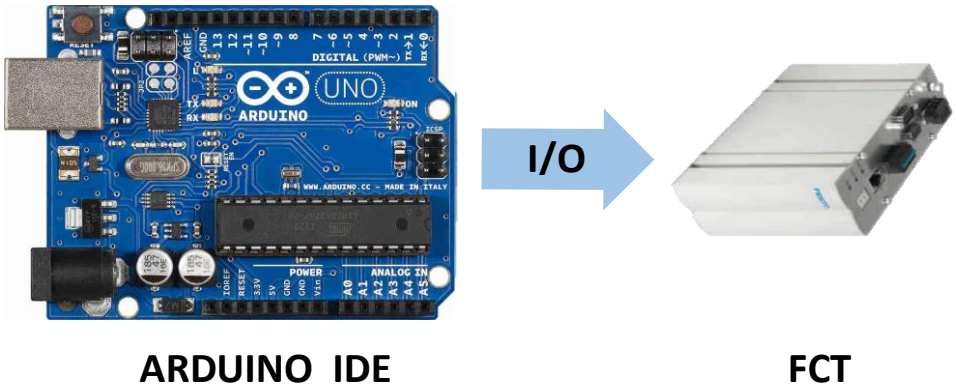


Figura 8: Esquema programadors utilitzats en l'aplicació Gantry

6.2 ENTORN DEL IRB 140

Pel que fa a l'aplicació del braç robòtic, aquesta està formada únicament per el propi robot físic, la seva controlador IRC5 i l'ordinador del laboratori que ja està destinat a aquest robot. Aquest ordinador és el que permet al programador connectar-se al robot mitjançant una comunicació via Ethernet i carregar-li els programes que després executarà. El mateix ordinador serà el centre de les dues aplicacions ja que disposarà dels programes i configuracions necessàries per comunicar amb cadascun dels robots. A més, és on s'ha desenvolupat el sistema SCADA i per tant serà on correrà l'aplicació de supervisió. En la següent imatge s'observa l'estructura hardware que segueix aquesta part del treball.

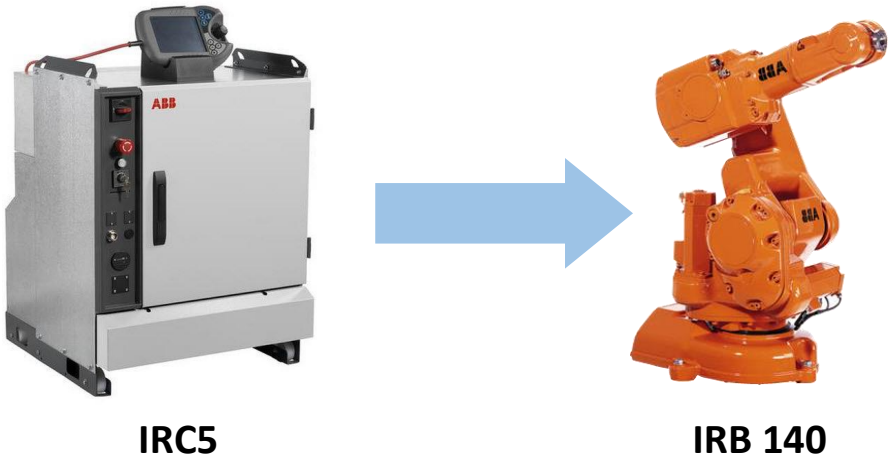


Figura 9: Esquema hardware del IRB 140

En la part de software, la controladora IRC5 d'aquest robot es pot programar des de la interfície anomenada FlexPendant que podem veure a la part superior de la controladora de la figura. Aquesta petita pantalla tàctil permet al programador interactuar amb el robot en tots els nivells, des de la programació de codi fins a la configuració interna. Tot i això, programar des de la FlexPendant no és una programació fàcil i per això s'ha optat per programar externament i a continuació carregar el programa des de l'ordinador del laboratori.

La programació externa s'ha realitzat des del simulador RobotStudio de la mateixa casa ABB. Aquest simulador permet programar i configurar el robot d'una manera més assequible i de forma offline. El simulador és capaç d'executar el programa i veure com es comportaria el robot, de manera que és molt útil a l'hora de fer proves i comprovacions de funcionament. A més de permetre visualitzar el comportament físic del robot, també pot veure el comportament intern dels motors, la força que fan, l'escalfament que pateixen, etc.

6.3 ENTORN DE L'APLICACIÓ FINAL

Una vegada coneguts tots els elements per separat podem veure com ha quedat finalment muntada l'aplicació des del punt de vista del hardware i del software en un esquema global.

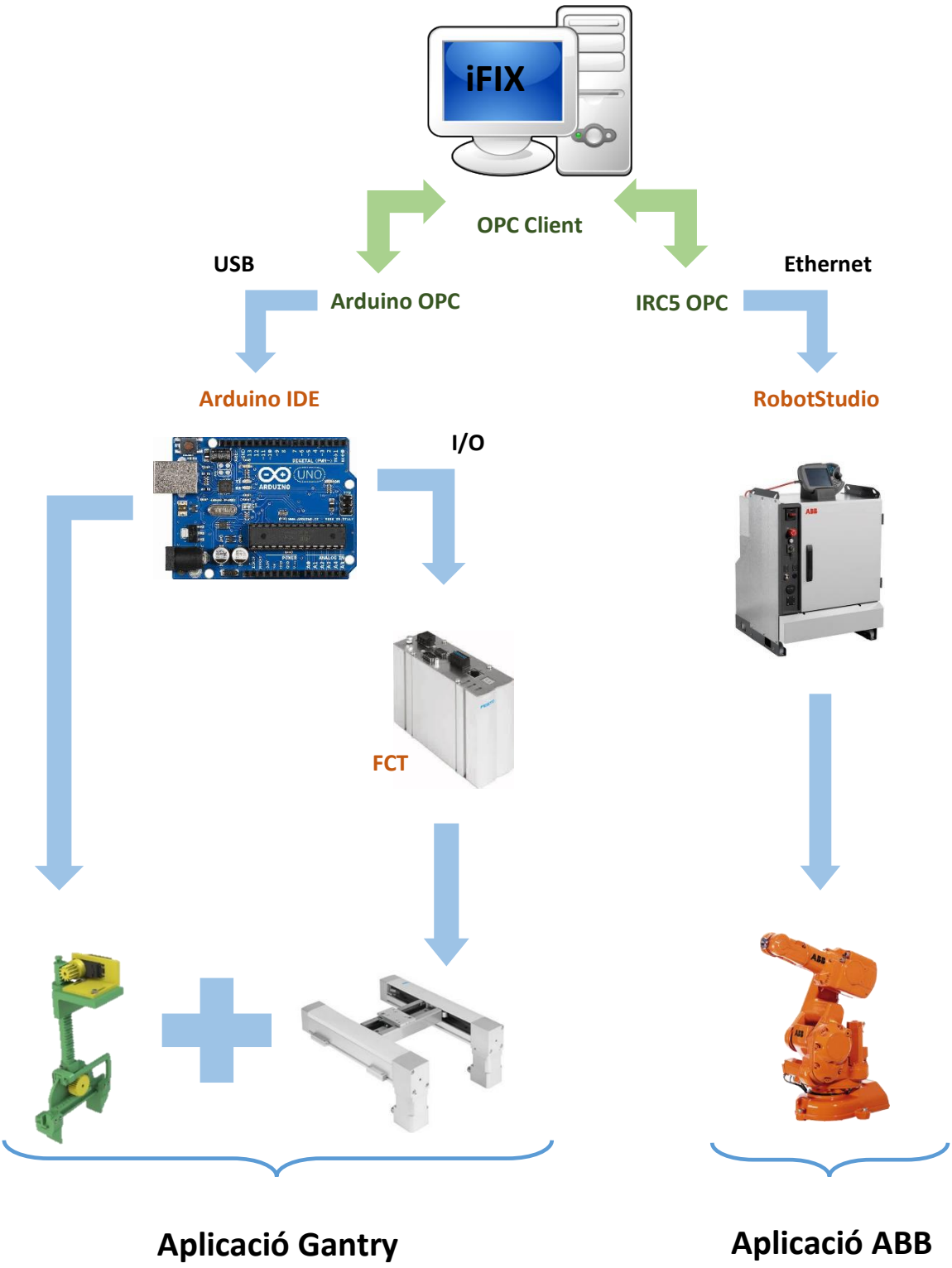


Figura 10: Esquema global de Hardware i Software

Si seguim l'esquema des de la part superior trobem inicialment el programa iFIX on s'executarà el sistema SCADA. Com és habitual en aquestes aplicacions, s'ha utilitzat una estructura Client/Servidor on el programa iFIX integrarà un OPC que serà el client, en canvi l'Arduino OPC i l'ABB IRC5 OPC interpretaran el paper de servidors. D'aquesta manera l'iFIX tindrà la capacitat d'escriure i llegir dels dos servidors que es poden veure en un segon nivell. Aquesta possibilitat bidireccional de flux d'informació converteix al sistema SCADA en el pont per a la transmissió de dades entre les dues aplicacions.

Seguidament amb color negre es mostren les interfícies de comunicació que utilitzen aquests servidors OPC per comunicar amb els controladors respectius dels que extrauran les dades. En el cas de l'Arduino s'ha utilitzat un protocol de comunicació sèrie USB i en la IRC5 el protocol Ethernet.

A continuació amb lletres taronges es mostren els IDE o interfícies de programació que s'han utilitzat en cada cas per a programar i configurar cadascun dels software que integren els components físics del sistema.

7 INICIS DEL CONCURS E-MOTION

Seguint les pinzellades fetes en els apartats anteriors a continuació s'explicaran els inicis del concurs E-Motion, i què va portar a escollir el packaging de mòbils com a aplicació final.

Es recorda que el propòsit era desenvolupar una aplicació de temàtica lliure amb el robot Mini-H Gantry entregat per l'empresa FESTO. Per altra banda, en iniciar aquest projecte encara no es coneixia la temàtica del treball final de grau i per tant, l'aplicació del Gantry estava inicialment pensada i desenvolupada de manera individual.

7.1 BASES DEL CONCURS

FESTO Automation va proposar el concurs E-Motion a través del repte anomenat "*case study*" que consistia en fer una aplicació pràctica d'àmbit i temàtica lliure amb el material facilitat a cadascun dels equips participants. El equips podien estar formats per un grup d'entre 2 i 4 persones i la única condició per a la participació era haver cursat l'assignatura de Fabricació Automatitzada i Robòtica Industrial (FARI) en els dos últims cursos. Després d'aquest filtre els equips candidats van enviar una carta de motivació i FESTO va escollir quatre d'aquests per a participar en el concurs.

Per altra banda, també es disposava de la llibertat de complementar el manipulador per a la realització de l'aplicació escollida utilitzant altres elements sempre que entressin dins el pressupost.

Com s'ha dit, l'empresa no va posar cap limitació en la temàtica i l'àmbit de l'aplicació, però tampoc podem oblidar que es tractava d'un concurs i per això sí que hi havia un seguit de característiques a tenir en compte.

- **Funcionalitat industrial i oportunitat de negoci:** L'objectiu de crear un producte és poder recuperar d'alguna manera la inversió en el desenvolupament tenint la possibilitat de vendre el producte i explotar-lo al màxim. Per tal de que això es pugui dur a terme de la millor forma possible caldrà que sigui aplicable en el mercat actual per poder modernitzar la industria i millorar cada vegades més els processos industrials.
- **Originalitat, caràcter innovador o creativitat:** Per complir amb el primer punt cal buscar idees innovadores i diferents als productes habituals. D'aquesta manera el producte nou pot tenir un millor èxit en el mercat que un altre que copiï o es basi en un producte ja existent.
- **Qualitat:** Per poder vendre una bon prototip, com és el nostre cas, s'ha de presentar amb un mínim de qualitat per tal de donar la sensació de fiabilitat i robustesa necessàries.

- **Adaptar-se al pressupost atorgat:** El pressupost és un factor molt important en tots els sectors de la industria ja que en una empresa hi ha diferents departament com per exemple producció, manteniment, investigació, compres, ventes, etc. Que es segueixin els pressupostos estimats en tots el departaments pot determinar que una empresa tingui pèrdues o beneficis a final d'any.
- **Viabilitat tècnica:** No n'hi ha prou en disposar d'una bona idea que compleixi tots els apartats anteriors si no és viable. La no viabilitat sorgeix quan una idea és impossible de reproduir-se per motius com la falta de tecnologia o l'incompliment de lleis físiques.
- **Sostenibilitat:** En el món actual la tecnologia canvia exageradament ràpid i això també fa canviar els productes i els processos de fabricació. Per això cal buscar aplicacions sostenibles en el temps i que puguin tenir una explotació a llarg termini per tal d'optimitzar al màxim el producte i la inversió.

El llistat de característiques segueix un ordre d'importància que el jurat tindria en compte a l'hora de valorar les aplicacions i determinar el grup guanyador del concurs.

7.2 PLUJA D'IDEES

L'elecció d'una idea que es pugui traduir en una aplicació per al Gantry es pot trobar sense massa complicació. Si la proposta ha de complir amb el llistat anterior, escollir una bona idea es converteix en una tasca més difícil. Per aquest motiu, s'ha utilitzat la pluja d'idees per presentar totes les opcions i poder descartar aquelles sorgides en un primer moment.

7.2.1 IDEA 1

En un primer instant, es va pensar en una aplicació on es demanés al robot un producte i aquest l'anés a buscar en una prestatgeria on el robot estaria col·locat paral·lelament. Pot semblar una idea descabellada amb el Mini-H Gantry, però hem de deixar molt clar que aquest projecte consistia en desenvolupar un prototip. Per altra banda, si la casa FESTO té un robot que es diu Mini-H Gantry es pot intuir que com a mínim n'hi haurà un altre que serà el H Gantry, i és així. La casa alemanya té diferents games d'aquest robot i el més gran, el EXCH-60, pot arribar als 2500x1500 mm. Per tant, la idea ja no queda tant fora de lloc.

El nostre equip va decidir ajustar aquesta idea en una ferreteria. El cas, és que en el món dels cargols hi ha una gran quantitat de productes diferents i es va pensar que podria ser molt interessant aquesta idea. Segons l'esbós de la Figura 11 podem observar que el robot recolliria mitjançant el seu carro (2) algun dels calaixets que estarien posats en un prestatge (1). Seguidament el robot es mouria fins a una petita plataforma (3) que baixaria per tal de que l'usuari de l'aplicació, o botiguer en aquest cas, pogués servir el material demant pel client (4).

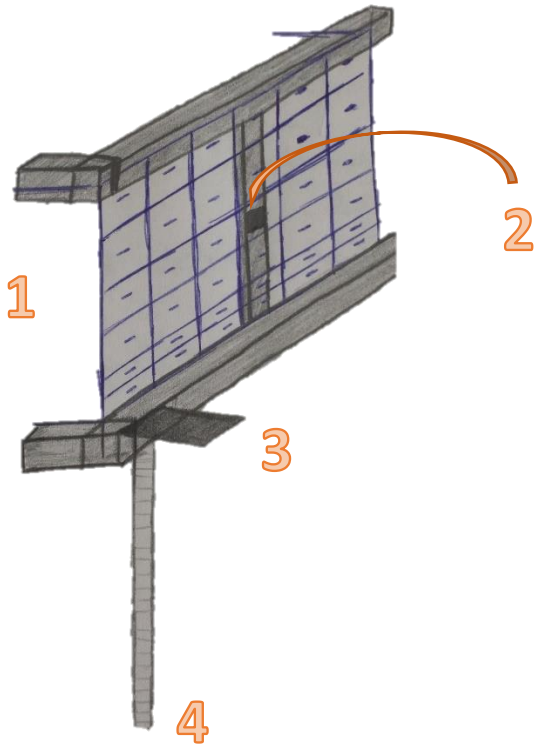


Figura 11: Esbós d'una aplicació en una ferreteria

Aquesta aplicació es basa en una idea principal que és l'ampliació de l'espai útil en una botiga de dimensions reduïdes. En una botiga de sostres alts, es podria situar el robot en la part superior i eliminar la necessitat d'utilitzar una escala per agafar el material, d'aquí el motiu de la plataforma elevadora (Figura 11, punt 3). Si la botiga no disposes de sostres alts, es podria situar per exemple en un passadís estret, on no s'hi hagués de circular, i col·locar la guia de la plataforma en posició horitzontal per arribar fins a una part del taulell.

Un dels motius principals pel qual es va descartar aquesta idea va ser la dificultat que suposaria fer un projecte a escala i que mostrés el potencial de l'aplicació. Un altre motiu, tenint presents les bases del concurs, és que fallava en els dos primers punts del llistat. El primer punt té present la oportunitat de negoci i es va creure que potser seria difícil trobar un ferreter que volgués invertir certa quantitat de diners en un aparell per servir claus i cargols. El segon punt demanava originalitat i creativitat entre altres, i en aquest cas sí que era innovadora la idea de maximitzar l'espai útil de la botiga però no automatitzar un servei per l'estil degut a que ja existeixen aplicacions molt similars en botigues de grans dimensions.

7.2.2 IDEA 2

Una vegada descartat el primer pensament l'equip va començar a pensar en el món de l'embalatge. Pensant a gran escala, es va arribar a la conclusió de que una empresa de logística que tingui una part de packaging com podria ser Amazon, té una gran varietat de productes de dimensions completament diferents els uns del altres. Per aquest motiu es va creure que podria ser interessant inventar un sistema per fer el packaging de cada producte independentment per tal de reduir el material de l'embalatge. En altres paraules, fer un empaquetatge a mida per cada producte.

Per poder desenvolupar aquesta idea s'hauria d'inventar un primer sistema per mesurar la peça i així saber quines dimensions i forma hauria de tenir la caixa per empaquetar el producte. Finalment, el robot cartesià es cuidaria de traduir aquestes mesures en una caixa tallant un patró de la forma i mides desitjades a partir d'un cartró.

El sistema per mesurar la peça tindria la forma que es pot veure a la Figura 12 i estaria format per una base giratòria (1), dos eixos mòbils col·locats un vertical i l'altre horitzontal (2) i un sensor situat en un dels eixos (3). El funcionament del sistema seria de manera que l'objecte a mesurar es situaria al centre de la base en repòs inicialment. L'eix que té el sensor aniria unit a l'eix vertical de manera que a part de la coordenada horitzontal el mateix sensor tindria llibertat verticalment. D'aquesta manera el detector faria un escàner de l'objecte en 2D i guardaria la mesura més alta, seguidament la base giraria per poder mesurar la profunditat de l'objecte.

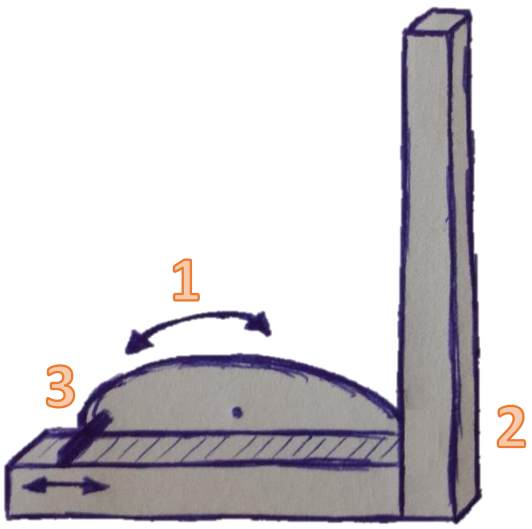


Figura 12: Esbós d'un mesurador de peces

Finalment aquestes mesures s'haurien de traduir a unes altres mesures que permetessin al robot cartesià, amb l'eina corresponent, tallar un patró de la forma adequada per a l'objecte mesurat. En funció de les dades capturades també es podria utilitzar una forma de caixa diferent en cada cas, és a dir, per ajustar més la quantitat de material es podria utilitzar una forma de caixa triangular per a un objecte d'aquesta forma en comptes d'utilitzar sempre caixes rectangulars.

Aquesta idea tenia diferents punts dèbils des del nostre punt de vista. Un exemple és que en el cas de figures molt irregulars el sistema d'escàner hauria de ser molt ràpid per tal de que l'aplicació fos eficient i es pogués observar la peça des de tots els punts de vista necessaris i determinar la distància més gran. També es va pensar que seria complicat trobar una eina capaç de tallar el contorn i marcar el cartró per fer els doblecs necessaris, ja que segurament, per fer-ho de forma correcte, es necessitaria un sensor de pressió per fer els talls i marcatges sempre iguals. També cal reconèixer que la part del sistema d'escàner se'n va molt del robot Gantry i la intenció del concurs era fer una aplicació a partir d'aquest manipulador.

Un segon punt de vista de la mateixa idea era utilitzar el Gantry per a l'escàner dels objectes, però si ho fèiem d'aquesta manera l'única funció de l'aplicació seria mesurar objectes i no hi hauria cap procés productiu físic. Per això, es va creure que aquesta segona opció no impressionaria tant com per presentar-la en un concurs.

Així doncs, per tots aquest motius es va creure que aquesta idea no era viable tècnicament degut a la seva complexitat i al poc temps que disposàvem per a desenvolupar-la.

7.2.3 LA IDEA DEFINITIVA

Una vegada introduïts en el món de l'embalatge vam arribar a la conclusió de que processos de producció per a productes de tecnologia punta, mancaven d'aquesta tecnologia per a la seva fabricació.

L'exemple més clar que es va trobar, i en qual ens vàrem centrar, va ser el packaging de mòbils. Actualment, cada any surten al marcat nous mòbils amb millors càmeres, més ràpids, més emmagatzematge i moltes més prestacions, però tot i així ha de ser una persona qui faci una feina tan repetitiva com és posar els mòbils a dintre les caixes.

Per aquest motiu vam decidir que hi havia una oportunitat industrial molt gran en aquest sector i que es podia aprofitar per fer un bon projecte. La idea era utilitzar el Gantry per empaquetar el mòbil i tots els components que es troben quan en comprem un de nou com són el carregador, el cable, les instruccions, els auriculars i el propi mòbil. Tots aquests elements s'haurien de col·locar dins una caixa que permetés anar directament al consumidor, sense necessitat d'un segon embalatge.

En la següent imatge podem observar el funcionament d'aquesta idea. El robot estaria muntat en una línia contínua de producció mitjançant una cinta transportadora (1) que faria arribar unes safates amb els elements distribuïts sempre de la mateixa manera (2). Un cop arribada a la posició del Gantry (3) la cinta s'aturaria i el robot faria el procés de packaging. Una vegada finalitzat el Package, la cinta continuaria fins a situar la següent safata sota el robot.

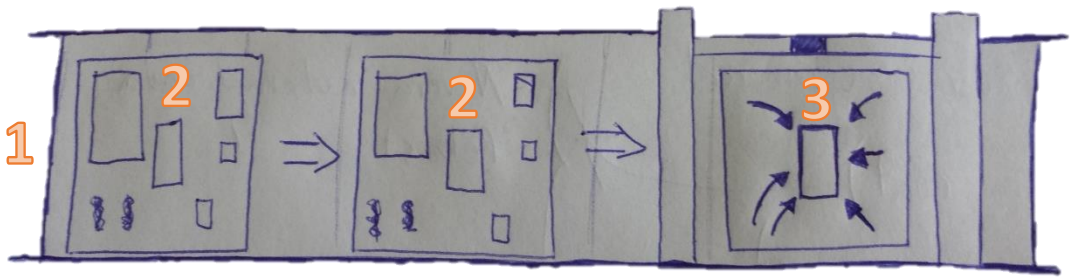


Figura 13: Esbós de l'aplicació de packaging

Aquesta aplicació requeria dues coses molt importants per al bon funcionament. La primera és que necessitàvem un element terminal que ens permetés agafar tots els components. Un segon factor imprescindible era la necessitat de que el packaging es pogués fer de forma totalment vertical, és a dir, eliminant doblecs de cartrons entre elements com tenen alguns models actuals.

Una vegada vistes totes les opcions, es va decidir que aquesta aplicació era la que tenia més potencial perquè complia satisfactòriament amb els condicionants que es demanaven.

7.3 ESTUDI DE MERCAT

Per argumentar l'elecció del packaging de mòbils s'ha fet un petit estudi de mercat per ser més conscients del món de la tecnologia mòbil i com funciona el seu procés de producció.

La informació que ve a continuació esta basada en els estudis de la *International Data Corporation* (IDC), empresa filial de *International Data Group* (IDG). Aquesta empresa és la principal proveïdora d'intel·ligència de mercat i serveis de consultoria. El seu abast es resumeix en l'àrea de la tecnologia de la informació, de les telecomunicacions i també en la tecnologia de consum. L'equip que la forma es basa en un conjunt d'analistes els quals realitzen estudis de mercat que permeten ajudar als empresaris i inversors a fer previsions del mercat i prendre decisions dins l'empresa. Dins la IDC trobem la *Worldwide Quarterly Mobile Phone Tracker* qui es cuida de gestionar les dades en el món dels smartphones de forma trimestral.

Tot i que no s'ha gaudit d'accés directe a les bases de dades, sí que s'han pogut analitzar un seguit de notícies i informes amb dades rellevants que han permès fer-nos una idea de com està el mercat.

Actualment hi ha uns 130 fabricants de mòbils distribuïts en 35 països arreu del món encara que es pot afirmar que el monopoli del sector esta governat per menys de 10 marques. A continuació es valoraran les dades dels principals venedors de smartphones de l'any 2015.

Com podem veure a la Taula 1, hi ha dos líders mundials molt coneguts al capdamunt del llistat que representen gairebé el 40% del total de la quota de mercat. El percentatge que resta ja no es divideix en marques tan conegudes com podrien ser Sony o LG degut a que aquest llistat és a nivell mundial. En l'ultima columna podem veure la xifra més significativa de la taula que indica el creixement de cadascuna de les marques respecte l'any anterior. Es pot observar que totes les marques han incrementat en vendes, tot i que no ho han fet d'acord amb la posició que ocupen en la llista. Samsung, que es troba al capdavant, té l'increment de ventes més reduït de tots amb un 2%, deu vegades menys que el segon de la llista amb menys ventes (Apple). Mentre que les posicions dos, quatre i cinc de la llista tindrien percentatges de ventes similars, Huawei despunta amb un 44% respecte el 2014.

Per descomptat no podem obviar la penúltima línia en negreta de la taula on es pot veure que la xifra total de smartphones venuts l'any 2015 és de 1.432 milions.

Top Five Smartphone Vendors, Shipments, Market Share and Year-Over-Year Growth, Calendar Year 2015 Preliminary Data (Units in Millions)					
Vendor	**2015 Shipment Volumes**	**2015 Market Share**	**2014 Shipment Volumes**	**2014 Market Share**	**Year-Over-Year Growth**
1. Samsung	324.8	22.7%	318.2	24.4%	2.1%
2. Apple	231.5	16.2%	192.7	14.8%	20.2%
3. Huawei	106.6	7.4%	73.8	5.7%	44.3%
4. Lenovo	74.0	5.2%	59.4	4.6%	24.5%
5. Xiaomi	70.8	4.9%	57.7	4.4%	22.8%
Others	625.2	43.6%	599.9	46.1%	4.2%
Total	**1,432.9**	**100.0%**	**1,301.7**	**100.0%**	**10.1%**
Lenovo + Motorola	73.9	5.16%	93.7	7.20%	-21.1%
Source: IDC Worldwide Quarterly Mobile Phone Tracker, January 27, 2016					

Taula 1: Índex de venta de mòbils al 2015

La Figura 14 recolza la taula anterior i permet fer una comparativa més visual de les vendes de les principals marques de smartphones des de l'any 2011. A grans trets podríem dir que els líders del sector tendeixen a la baixa, mentre que els fabricants menys coneguts destaquen pel seu creixement positiu.

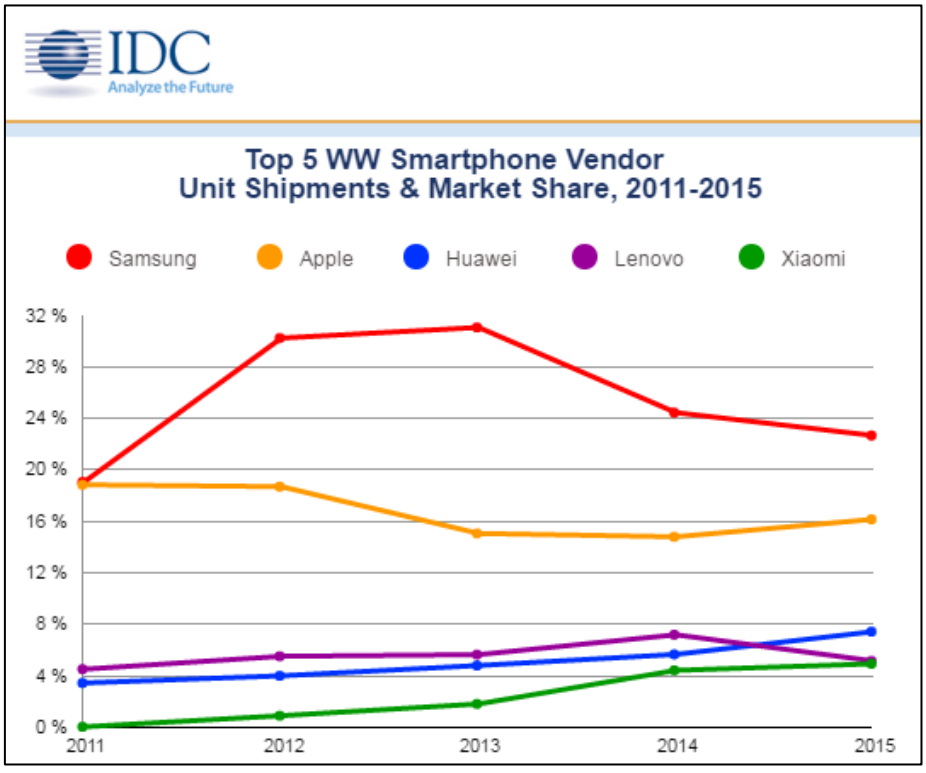


Figura 14: Gràfic de ventes dels últims 5 anys

Una vegades vistes les escandaloses xifres de mòbils que es fabriquen a l'any calia saber com es podien processar tantes quantitats de productes i la forma en què es processaven.

Els smartphones d'avui en dia representen gran part de la tecnologia més avançada que existeix. Així doncs, per fer un producte de tecnologia punta es necessiten les millors eines i els processos més avançats de fabricació. En la producció de mòbils la majoria de processos estan automatitzats, inclús una vegada fet el muntatge els encarregats de comprovar el bon funcionament del software són uns robots que manipulen el mòbil amb un element tàctil que simula el dit humà. En aquestes comprovacions es mira el funcionament global de l'aparell i els temps de resposta al navegar entre aplicacions i a les pulsacions del robot.

Així doncs, en un principi no sembla un sector on es pugui incloure una millora del procés productiu. Tot i això, una vegada acabada la cadena de muntatge falta el procés de packaging, on el mòbil s'ajunta amb els components que el complementen com són el carregador, els auriculars i les instruccions. Una vegades feta la cerca necessària s'ha comprovat que aquest procés és totalment manual i requereix gran quantitat de personal per poder assolir la demanda de smartphones dels consumidors arreu del món.

Figura 15: Procés manual d'empaquetatge

Un cop vistes les dades anteriors podem afirmar que la tecnologia mòbil és un sector en ple creixement i que representa una part molt important de l'economia de mercat. Per això creiem que depenent de l'evolució en el procés productiu, aquest àmbit podrà tenir un futur o un altre.

Si s'automatitza un procés manual com és el packaging les fàbriques podran tenir una producció constant i amb l'eliminació d'errors humans degut a la repetitivitat i monotonia del procés. Suposaria un estalvi en personal que es podria destinar a altres tasques que sí requerissin un criteri humà. També abaratiria els costos del producte ja que un cop feta la inversió inicial l'únic cost del procés seria per al manteniment i la flexibilització per a nous models. Per últim, amb un bon sistema d'empaquetat, també es podrien reduir els temps de producció sempre i quan el mercat fos capaç d'absorbir l'índex de producció.

8 OPCIONS ESTUDIADES

Una vegada triada l'aplicació a realitzar i tenint clar quin seria el funcionament d'aquesta s'ha discutit quina seria la millor opció per a desenvolupar cadascuna de les parts.

Abans de començar amb l'explicació remarco que la meva implicació personal durant el desenvolupament del concurs va ser en totes les parts però especialment la part de control i electrònica que vaig duu a terme d'una forma més autònoma.

8.1 PLANTEIG DEL SISTEMA, MATERIAL, FUNCIONAMENT DEL TERCER EIX I L'ELEMENT TERMINAL

En l'aplicació escollida, el tercer eix i l'element terminal eren les parts més delicades del projecte ja que la seva funció era agafar i col·locar dins la caixa cadascuna de les parts. Per fer-ho s'havien d'aconseguir moviments molt precisos i ajustats a les dimensions dels elements.

La primera opció que va sorgir per al tercer eix va ser unir al carro del robot una espècie de femella amb un cargol sense fi. Es muntaria un motor a l'extrem superior que permetés pujar i baixar la pinça situada a l'altre extrem. Aquesta idea requeria molta flexibilitat dels cables del motor ja que per fer pujar la pinça també hauria de pujar el motor. A més, s'havia de fixar el motor d'alguna manera que no girés sobre si mateix a causa del fregament de la femella. Per fixar tant el motor com la pinça es podia utilitzar algun tipus de guies que forcessin el moviment vertical. Per altra banda, la pinça hauria d'estar unida a l'eix de manera que permetés un gir total d'aquest i la pinça sempre estigués en la mateixa posició. Això seria possible amb un sense fi acabat en un encaix circular que permetés girar sobre la pinça i fixar aquesta igual que el motor.

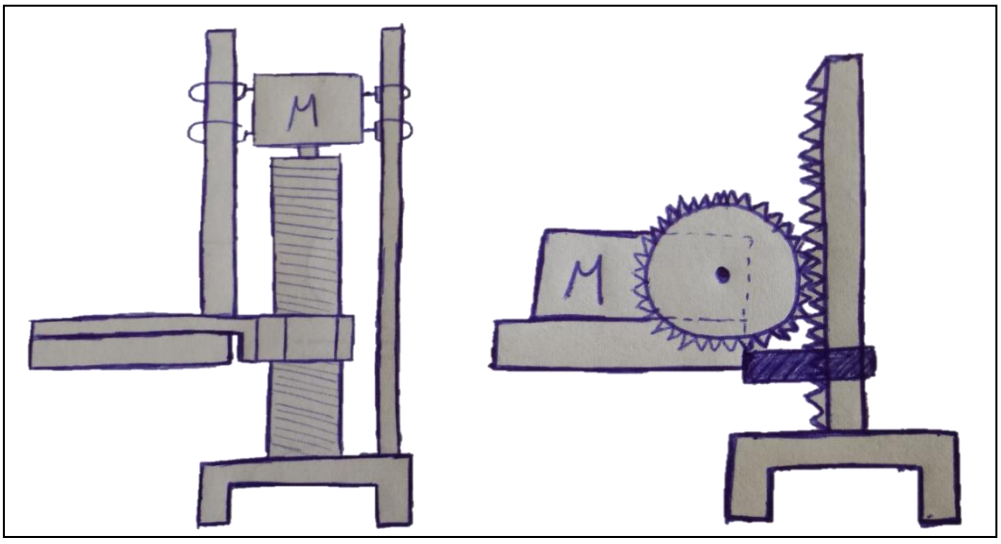


Figura 16: Esbós de les diferents opcions per implementar el tercer eix

La segona opció estudiada era un sistema típic de pinyó-cremallera on s'uneix el motor a un pinyó que tindrà la força motriu. Després s'acobla a aquest pinyó una cremallera, que seria el propi eix vertical que faria moure la pinça. L'eix amb la cremallera s'hauria de fixar de tal manera que permetés un bon contacte amb el pinyó perquè aquest no saltés els engranatges de la cremallera.

Tant per al disseny de l'eix com l'element terminal s'havia de buscar com fer-ho ja que són adaptacions concretes a dimensions concretes i que no es troben al mercat. Com hem dit, és una part de manipulació complexa ja que requereix força precisió a l'hora de col·locar els elements a una determinada posició dins una caixa. Per això, a l'hora de valorar amb quin material es podrien dissenyar els elements s'havia de buscar un material resistent però econòmic degut al pressupost concedit.

Una opció que es va pensar que donaria robustesa a l'aplicació, era dissenyar els dos elements amb un material metàl·lic com podria ser l'alumini o l'acer. L'alumini és un material que es caracteritza per pesar poc i ser flexible, a més té un rendiment a fatiga força baix, cosa que a la llarga podria perdre precisió en la repetitivitat de l'aplicació. En canvi, l'acer pesa més i al ser un material més dur donaria robustesa al conjunt. Per altra banda, també es podria buscar algun tipus d'aliatge d'aquests materials o altres que donessin unes característiques més ajustades a les necessitats.

La fabricació d'aquests elements amb alumini, acer o similar segurament necessitarien acudir a un centre de mecanitzat especialitzat en la fabricació d'aquests tipus de materials.

L'altra opció era utilitzar un material plàstic que segurament abaratiria els costos de fabricació tot i que també seria complicat trobar un material amb les característiques desitjades. Com a material plàstic es va pensar en utilitzar la tecnologia d'impressió 3D per a poder utilitzar les eines CAD conegudes i poder dissenyar personalment les peces prèviament amb ordinador. D'aquesta manera es veurien errors i possibles millores a fer abans de fabricar els components.

A part del material amb què es podrien fabricar els elements, la part més important d'aquests era el funcionament. En el cas de l'eix vertical no hi havia massa problema ja que en les dues opcions només s'hauria de coordinar la velocitat del motor amb el pas del cargol sense fi en el primer cas, i en el segon coordinar-la amb la relació del sistema pinyó-cremallera.

En el cas de la pinça inicialment es va estudiar l'opció de comprar-ne una ja que en el mercat n'hi ha de molts tipus. En la Figura 17 es mostren dos exemples de pinces metàl·liques habituals en el mercat. La primera pinça té una limitació en l'obertura deguda a les dimensions del conjunt

d'enllaços entre la pròpia pinça i l'eix del motor. Tot i això, aquesta pinça ja té una adaptació per a muntar-hi un servomotor per a l'obertura i una altra a la part superior per un segon servomotor que permetria la rotació de la pinça.

La segona pinça té un sistema semblant als clàssics obridors d'ampolles de vi amb forma de ballarina. Aquest sistema té el problema que les parts de contacte de la pinça estan sotmesos a un moviment vertical que varia depenent de l'obertura de la pinça. Aquest moviment dificultaria la col·locació dels elements a l'interior de la caixa. Igual que en el cas anterior, aquesta també esta preparada per a muntar-hi els dos motors que permetran l'obertura i la rotació de la pinça.

Figura 17: Exemple de pinces existents en el mercat

8.2 PLANTEIG DEL CONTROL DEL SISTEMA GANTRY

Com s'ha introduït en apartats anteriors, per poder controlar el Gantry s'ha hagut d'imposar un nivell de control superior. Aquest nivell de control també seria qui manipularia l'eix vertical i l'element terminal.

Abans d'escollir un controlador s'han de seleccionar quins elements s'utilitzarien per moure l'eix i la pinça. Els moviments necessaris es divideixen doncs, en el moviment vertical i l'obertura de la pinça. Per tant, els dos moviment es realitzaran a través de dos motors independents, el problema es presenta a l'hora d'escollir els tipus de motors a utilitzar.

Els motors controlables més coneguts i habituals en aquest tipus d'aplicacions són els motors pas a pas, PAP o steppers i els servomotors o servos.

Els servomotors no deixen de ser motors continus (DC) que inclouen un sistema de control, uns engranatges i un sensor de posició que sol ser un potenciòmetre. La funció dels engranatges és fer de sistema reductor de la velocitat del motor DC i augmentar el parell de l'eix. El potenciòmetre fa la funció que faria un *encoder* revelant la posició actual del motor en tot moment, és a dir, l'angle girat. Gràcies al potenciòmetre el sistema de control és capaç de mantenir la posició del motor encara que aquest rebi esforços externs.

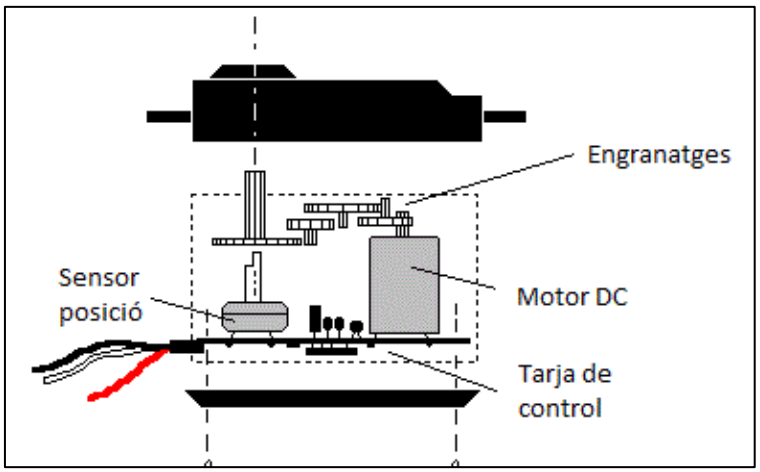


Figura 18: Interior d'un servomotor

Dintre dels servos n'existeixen de diferents tipus, els que tenen un angle de gir determinat, normalment 180º o 360º, i els que són de gir continu. Els que tenen un angle de gir determinat es solen ajudar per un limitador mecànic a part del sistema de control. Cal dir que el control d'un servomotor de gir limitat és més senzill que un de gir continu i això és deu a que els servos de gir continu són realment servos limitats modificats. Aquesta modificació es basa en treure el limitador mecànic i fer una modificació en el control. Una vegada manipulat s'obté un servo

igualment però que ara es controla per velocitat i no per posició com es feia abans. És veritat que fent càlculs entre la velocitat i el temps es pot conèixer la posició, i és així com funcionen els controls interns d'alguns servos més complerts utilitzats en aplicacions industrials.

Aquest tipus de motors, degut a la seva exactitud, s'utilitzen en aplicacions que requereixen molta precisió ja que permeten controlar la posició exacte i en tot moment gràcies a la realimentació.

Per altra banda, els PAP també s'utilitzen per a controlar la posició però aquests difereixen dels servos en el seu funcionament. Aquests motors no estan formats per un motor DC sinó que tenen els anomenats motors magnètics. Hi ha diferents tipus d'aquests motors, però la majoria tenen un nucli ferromagnètic dentat que s'alinea amb cadascun dels electroimants que té al seu voltant a mesura que aquests s'activen mitjançant un control extern. Cada alineació amb un electroimant és un pas, d'aquí ve el nom, i com més electroimants tingui més precís serà el moviment del motor.

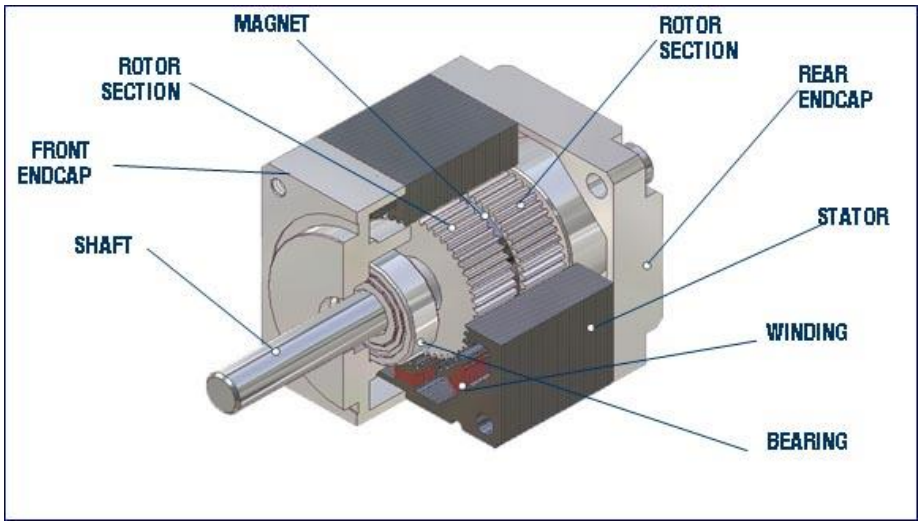


Figura 19: Interior de un motor pas a pas

Els motors a passos són més lents però no requereixen del sistema de realimentació per a conèixer la posició, ja que aquesta ve determinada per la rotació del propi motor que és incremental.

Una vegada explicat el funcionament dels tipus de motors, ja que l'aplicació no requereix una força excessiva per als elements i les dimensions del robot són reduïdes, s'ha arribat a la conclusió que independentment del tipus de motors caldria escollir-ne uns que treballessin a una tensió de 5 volts.

Aquí apareix la primera complicació important ja que ens trobem amb la controladora del manipulador treballant a 24 volts, la qual s'ha de governar des del mateix controlador que farà funcionar els dos servos de 5 volts. Tenint en compte els voltatges s'han valorat dues solucions, es podia buscar un controlador que treballés a 24v i fer una conversió per controlar els servomotors, o bé utilitzar un controlador a 5v igual que els motors i fer una conversió per a governar el Gantry. En qualsevol dels dos casos s'hauria de fer una adaptació de les tensions per comunicar els motors amb el controlador.

Coneguts els components a manipular, a l'hora d'escollir el controlador semblava que utilitzar un PLC donaria més robustesa a efectes pràctics de la aplicació ja que és un tipus de controlador molt utilitzat en la industria degut a la seva fiabilitat. Per altra banda, uns altres tipus de controladors molt coneguts i econòmics en electrònica són els de la plataforma Arduino, a més, diferents membres del grup en tenien i en coneixien el funcionament.

Un PLC, abreviatura de *Programmable Logic Controller*, és un autòmat dissenyat per a controlar maquinaria i processos en l'ambient industrial, que es comunica mitjançant un seguit de senyals d'entrades i sortides que poden ser analògiques o digitals. Aquests autòmats es caracteritzen per ser reprogramables i per això cadascun dels fabricants té el seu propi entorn de programació que permet crear i modificar programes amb diferents llenguatges de programació. Aquests softwares de programació no són econòmics i el preu depèn bàsicament de la marca de PLC escollit.

Dins la diversitat de marques de PLC hi ha dos tipus d'autòmats, els compactes i els modulars. Els compactes inclouen en un mateix bloc la CPU, la memòria, uns pins limitats d'entrades i sortides i en alguns casos la font d'alimentació. Els modulars tenen els mateixos elements que els anteriors però separats per mòduls o tarjes, excepte la memòria i la CPU que solen anar al mateix mòdul. Aquests es diferencien per la seva flexibilitat ja que es poden afegir diferents tarjes d'entrada/sortida o de comunicació per a altres dispositius. Un dels avantatges en cas de malmetre's algun mòdul només cal canviar aquell que s'ha fet malbé sense necessitat de modificacions en la resta. Els PLC de tipologia compacte, solen ser més senzills i per les seves característiques, són més econòmics que els modulars.

Tot i això, utilitzar un PLC modular permetria més facilitat en solucionar el problema de les tensions. Aquests permetrien destinar un mòdul al Gantry i un altre als servomotors treballant a 24v i 5v respectivament.

Des de l'altre punt de vista, Arduino és una placa de circuit imprès que integra un microcontrolador i un conjunt de pins per entrades i sortides que esta destinada al prototipatge.

Es programa des d'un entorn de desenvolupament o IDE que és d'adquisició lliure i el seu llenguatge de programació és el de Wiring que és una simplificació del llenguatge C/C++.

Arduino significa gran amic en italià, això es deu a que, a diferència del PLC, la intenció inicial era destinar aquesta plataforma a la creació de prototips i introduir a les persones no expertes al món de l'electrònica.

Igual que en el cas dels PLC, Arduino també té diferents models de plaques i es diferencien per la quantitat d'entrades i sortides que tenen, el processador, i les funcions que inclouen ja que les més noves permeten inclús comunicació via Wi-Fi i Bluetooth.

Figura 20: Exemple d'un PLC modular, un compacte i una placa Arduino

Per últim calia estudiar quin protocol es faria servir per a la comunicació entre la controladora del Gantry i el propi manipulador. L'EXCM-30 disposa de tres interfícies de control per comunicar la controladora amb el dispositiu de control superior:

- I/O digitals (Entrades/Sortides)
- CAN Open
- Control via Ethernet (CVE)

Tot hi haver diferents opcions per al control, degut al poc temps que s'ha tingut per investigar el funcionament de totes elles, l'equip ha decidit directament utilitzar la interfície de comunicació per entrades i sortides digitals.

En definir l'entorn de programació del Gantry s'ha comentat que es necessitaria omplir una taula amb les posicions desitjades per moure el robot mitjançant el software FCT. Aquestes posicions reben el nom de **frases** i determinen una posició, la velocitat i l'acceleració a la que s'ha de moure el robot per assolir l'objectiu. Un cop definides, s'envien les frases a la controladora del robot on quedaran guardades i posteriorment triarem el punt al que volem que es desplaci el robot seleccionant el número de la frase desitjada.

Per a la selecció, la controladora del robot disposa d'una interfície física de comunicació en la qual s'hi envien senyals de lògica codificades de forma binària. La interfície I/O és la més senzilla de totes i només permet fer moviments escollint fins a un límit de 32 frases.

Les altres dues maneres de comunicació són més complexes i no només permeten escollir una de les frases, sinó que també poden fer el que es coneix com a acció directa. L'acció directa permet parametritzar en cada instant la posició a assolir utilitzant els bytes de control i d'estat en el cas de la comunicació CAN Open, o enviant trames mitjançant el protocol TCP que utilitza Ethernet. Aquests protocols de comunicació, al permetre l'acció directa, no estan limitats en quant al nombre de posicions a diferència de la interfície I/O.

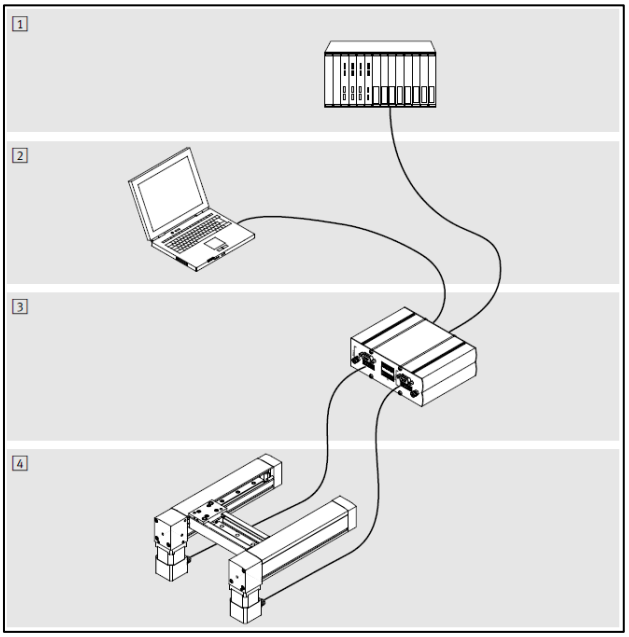


Figura 21: Nivells de control del sistema

8.3 PLANTEIG DEL SISTEMA DE PALETITZAT

El paletitzat és la part de l'aplicació conjunta que gestiona el robot IRB140 de ABB. Aquesta part no ha necessitat cap part de disseny ja que el robot estava complert per fer el paletitzat de mòbils. Així doncs, totes les decisions preses en aquest cas han tingut a veure únicament amb el software.

El IRB140 és un robot destinat, entre d'altres coses, a fer processos de paletitzat. Tot i això la casa no ofereix cap facilitat en la programació d'aquest procés. Ens referim a facilitat de programació en cas d'haver-hi algunes eines com funcions predefinides, on introduint una sèrie de paràmetres es pogués definir la posició de paletitzat.

Al no tenir aquesta facilitat la programació és totalment manual i cal indicar cadascun dels punts per on passarà el robot. Perquè ens fem una idea de la quantitat de posicions que caldria definir analitzarem un paletitzat de 4 pisos i 4 objectes a cada pis. En total són 16 objectes a paletitzar que tindrien 16 punts diferents. A més, també caldria afegir una posició de seguretat, que seria a una distància sobre el punt destí de manera que el robot deixés l'objecte de forma totalment vertical per evitar col·lisions amb els objectes ja paletitzats. Les posicions de seguretat es poden reduir a 4 si se'n utilitza una de comuna per a cada columna en comptes d'una per a cada objecte. Les 20 posicions que s'obtindrien en aquest moment encara podrien augmentar més si variéssim el nombre de pisos. Com podem veure aquest sistema és molt intuïtiu i fàcil de programar però també incòmode ja que cal definir una gran quantitat de punts.

Una opció alternativa a la de definició de punts és utilitzar posicions relatives amb l'ajuda de la funció *RelTool*. Aquest mètode treballa amb un punt fix de referència que es pot modificar sumant i restant distàncies. L'avantatge d'aquest sistema és que únicament cal definir un punt, per contra, es complica lleugerament la programació ja que cal indicar molt bé els increments a fer per a cada objecte a paletitzar.

A part del sistema de paletitzat des d'un principi va sorgir la idea d'incloure en l'aplicació l'opció d'extreure les dades dels processos de paletitzat. Aquests fitxers permetrien millorar el procés i estudiar els temps de cicle per optimitzar l'aplicació en un futur o simplement tenir un control de la producció.

Les dades extretes del paletitzat es podrien escriure en un fitxer extern de diferents formats. Un format molt habitual és en un fitxer del tipus *".txt"*. A part d'aquest format es podria intentar extreure històrics de manera que es poguessin manipular en un programa pensat per tractar dades com podria ser Excel.

Per altra banda, per simular una producció en un ambient més real i industrial s'ha inclòs la funció de parada de manteniment en el funcionament del robot. Aquest produirà una aturada de la producció passat un cert temps des de l'inici del paletitzat. El mètode més adient per a la programació d'aquesta funció és utilitzar una intervenció del programa passat el temps establert.

9 SOLUCIÓ ADOPTADA

Una vegada vist el ventall de solucions possibles per al desenvolupament de la mateixa idea, cal justificar quina d'elles és la més adequada per a poder implementar en la nostra aplicació. Per això, i perquè el resultat d'aquest estudi sigui el màxim de satisfactori, s'ha intentat escollir la solució més adient per a cadascuna de les parts. Una bona elecció dels components o dissenys individuals resultaria en una millor combinació en el resultat final.

9.1 DEFINICIO DEL SISTEMA, MATERIAL, FUNCIONAMENT DEL TERCER EIX I L'ELEMENT TERMINAL

Per a la implementació del tercer eix finalment s'ha decidit el sistema pinyó-cremallera ja que s'ha cregut que donaria un millor rendiment. El sistema mitjançant el cargol sense fi s'ha descartat ja que necessitava unes guies per fixar tant el motor com la pròpia pinça i evitar que giressin juntament amb el cargol. Primer de tot cal reconèixer que aquestes guies donarien molt de volum al muntatge final. En segon lloc al tenir col·locat el sense fi en posició vertical i la pinça a l'extrem, els pesos que estiguessin subjectes afectarien al fregament entre el cargol i la femella. Com a conseqüència, la força de fregament es traduiria en un major moment d'inèrcia entre el motor i les guies.

Per això, al ser el sistema pinyó-cremallera una solució més senzilla tot i tenir una junta entre els dos elements que la formen, només caldria dissenyar l'eix de manera que no rotés sobre si mateix. Per assolir aquesta fixació a la rotació s'utilitzaria un eix de forma no cilíndrica.

En la Figura 17 s'han presentat dos tipus de pinça que es poden adquirir al mercat per 23 i 15 euros. La pinça de ballarina s'ha descartat per culpa del moviment vertical que rep al obrir i tancar i que a més depèn del tamany de l'objecte. L'altra pinça té un sistema i un disseny que s'ajustarien molt a les necessitats, tot i això l'obertura que permeten aquests models no era suficient per a poder agafar tots els components. Per això, finalment s'ha optat per dissenyar la pinça des de zero però basant-nos plenament en aquest model.

Pel que fa al disseny de l'eix i l'element terminal finalment s'ha decidit optar per utilitzar la tecnologia d'impressió 3D. La valoració que ha decantat cap a aquesta idea va ser el seu baix cost i la proximitat i facilitat per accedir al servei d'impressió. El baix cost permetria millorar el producte final i donaria l'opció de repetir algunes peces per corregir errors. En quant a la facilitat i proximitat, la pròpia universitat disposa d'un taller anomenat **FabLab** que dóna un servei d'impressió 3D. Això donaria rapidesa a l'hora de demanar, fabricar i recollir el producte.

9.2 DEFINICIÓ DEL CONTROL DEL SISTEMA GANTRY

Després d'estudiar els diferents tipus de motors i havent vist el seu funcionament, els avantatges i inconvenients, s'ha determinat que els servomotors donarien millor funcionalitat a l'aplicació. Principalment ha estat la rapidesa que permeten els servos davant els PAP el que ha fet triar els primers com a millor opció. A més, el preu dels motors a passos és generalment més elevat ja que necessiten un control extern per a gestionar els electroimants que el fan girar.

En l'aplicació ha sorgit la necessitat d'utilitzar dos motors, per a l'eix Z i la pinça. Es podrien haver fet servir els dos motors de gir continu, però finalment s'ha implementat amb un servomotor de gir continu per a l'eix vertical i un de mitja volta per a l'obertura de la pinça. Per a l'eix vertical es necessitava un motor que tingués molt recorregut, és a dir, que pogués donar moltes voltes, per poder pujar i baixar la pinça la distancia necessària. Per altra banda, la pinça no necessitava tant recorregut i per això s'ha escollit un motor de 180º de manera que amb mitja volta pogués obrir les distancies desitjades.

Pel que fa al controlador i el problema de la diferència de voltatges s'ha optat per escollir primer el controlador a utilitzar i després solucionar l'adaptació entre voltatges.

Així doncs, s'ha escollit l'Arduino com a controlador de nivell superior del sistema per les següents raons:

- Un PLC és un controlador pensat per aplicacions industrials complexes on es requereix un control de moltes variables.
- L'Arduino té la característica de ser una plataforma oberta de software lliure, cosa que elimina les llicències per a la programació i facilita l'adquisició d'informació i llibreries.
- Arduino es programa amb el seu propi llenguatge de programació i el seu IDE basat en Wiring/Processing que són ambdós una simplificació del llenguatge C, cosa que facilitaria la programació.
- Dintre de les llibreries disponibles a l'Arduino n'hi ha que permeten el control dels servomotors.
- A part del cost que pot suposar el software per a programar un PLC el cost del dispositiu per si sol pot suposar un increment de centenars d'euros en el millor dels casos.

Per últim, el protocol de comunicació entre l'Arduino i la controladora del Gantry s'ha realitzat mitjançant la interfície I/O degut a la seva simplicitat i facilitat de programació des del dispositiu de control finalment escollit.

Per fer la conversió de tensions necessària per adaptar les senyals del controlador Arduino a la controla del EXCM-30 es va provar inicialment amb transistors BJT ja que és un component electrònic estudiat en el grau i que es comporta com a interruptor. La controladora treballa amb tensions negatives o NPN, això significa que les senyals estan normalment a una tensió positiva i es necessita una tensió negativa per activar-les. En fer la demanda de material, es van comprar per error els transistors 2N3906 que són del tipus PNP. Una vegada provat de diferents formes i veure que no funcionava es va decidir canviar de sistema.

En aquest punt es tenien dues opcions, provar amb els transistors del tipus NPN o buscar un altre component. Els transistors de tipus NPN haurien de funcionar correctament, però es temia que pogués ocasionar algun problema per la gran diferència de voltatges entre elements. Per això es va optar per utilitzar un altre component com és l'optoacoblador. Aquest component fa la mateixa funció que el transistor però treballa de forma elèctricament aïllada. A diferència del transistor té una connexió interna de 2 circuit independents per mitjà d'un sensor òptic. Això significa que en circular corrent per la part emissora de l'optoacoblador s'encén un LED infraroig i a continuació el circuit receptor es tanca gràcies a la llum captada per un fototransistor. En la Figura 22 podem veure com són externament i també internament els transistors i els optoacobladors.

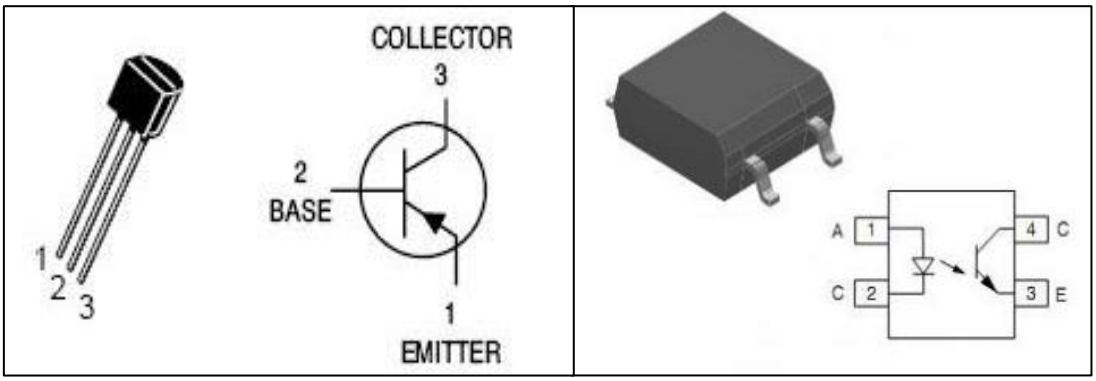


Figura 22: Exemple d'un transistor PNP i un optoacoblador

9.3 DEFINICIÓ DEL SISTEMA DE PALETITZAT

Dins les opcions disponibles per a realitzar la funció de paletitzat s'ha utilitzat com a base la idea de definir un sol punt i referenciar la resta sobre aquest punt inicial. Tot i això, no s'ha utilitzat la funció *RelTool* sinó que s'ha utilitzat un mètode pensat per a fer futures modificacions en l'aplicació. Per aquest motiu s'ha utilitzat el tipus de dada anomenat objecte de treball (WObj).

Els objectes de treball expressen el posicionament final de l'eina utilitzada pel robot respecte el sistema de coordenades definit en el WObj. A l'hora de definir un moviment a un punt en l'espai de treball del robot, es pot fer donant solament l'ordre de moviment o donar l'ordre i indicar un objecte de treball prèviament parametritzat. D'aquesta forma, la definició dels punts queda condicionada al sistema de coordenades utilitzat. Això es pot explicar de manera que si donem l'ordre al robot de moure's a un punt en l'espai, aquest punt variarà segons l'objecte de treball utilitzat.

L'avantatge d'utilitzar els objectes de treball per a referenciar els punts és la llibertat que dóna a l'hora de fer modificacions de posicionament en el futur. En el nostre cas, si fem un paletitzat referenciat a un WObj i en un futur es volgués moure la posició de palet, només caldria redefinir les coordenades de l'objecte de treball i no seria necessari reconstruir el programa. Això es tradueix en rapidesa i facilitat de programació en un ambient industrial on la continuïtat de la producció és un factor molt important.

Pel que fa a l'arxiu de dades, el format *".txt"* pot servir per visualitzar poques dades, però no és còmode de treballar i manipular d'es d'un editor de textos. Per això, finalment s'ha implementat en un fitxer de format *".xls"* per poder manipular les dades des del programa Excel. Tot i això, el llenguatge RAPID no reconeix comandes referenciades al programa de Microsoft per escriure en una fila o columna concreta. És a dir, que internament escriu en un fitxer de text qualsevol que depenent de l'extinció que se li doni s'obrirà en un programa o altra. Per aquest motiu i fent referència al programa Excel, s'ha utilitzat l'escriptura del text en diferents línies per saltar de fila, i la separació de valors mitjançant CSV (*Comma Separated Value*, de símbol ";") per diferenciar columnes. Configurant Excel adequadament, aquest caràcter permet separar en columnes independents les dades que vinguin unides en una mateixa línia.

Per a la integració de la funció de manteniment en el robot s'ha utilitzat una interrupció cíclica. Aquestes interrupcions no depenen directament de cap variable ni senyal d'entrada externa sinó que tant sols depenen d'un valor numèric que indica el temps de repetició de la interrupció. En el desenvolupament de l'aplicació sí que s'ha lligat aquest valor numèric a una variable per poder modificar el temps de cicle des del sistema SCADA.

10 DESENVOLUPAENT DE LA SOLUCIÓ

En el següent apartat s'explicaran el conjunt de passos seguits per dur a terme el desenvolupament de l'aplicació final. S'explicarà pas a pas els mètodes utilitzats en la implementació de cada part individual del treball.

10.1 EL GANTRY

10.1.1 CONDICIONANTS DE L'APLICACIÓ GANTRY

Per a poder demostrar el funcionament de l'aplicació i veure el potencial d'aquesta no ens havíem de conformar en desenvolupar la idea sinó que també calia demostrar que podia funcionar. Per això, s'ha buscat un mòbil ja existent en el mercat del qual se'n conservessin totes les parts i que a més complís amb la característica indispensable que permetés fer el packaging totalment en vertical. Per aquest motiu s'ha utilitzat un smartphone de la casa Samsung de model Galaxy SII.

Aquest model que podem veure a l'esquerra de la Figura 23, no és un smartphone d'última generació però és l'únic que s'ha trobat amb la possibilitat d'utilitzar la nostra aplicació per un assemblatge vertical. A la part dreta de la mateixa figura podem observar un model més actual de la mateixa casa però que per contra utilitza una làmina de cartró doblada entre els elements que el nostre robot no seria capaç de manipular. Aquest segon model conté tots els elements i les instruccions a la part inferior de la làmina i el mòbil a la part superior.

Figura 23: Comparativa del packaging real de dos models diferents

Per a la implementació del prototip s'ha decidit disposar del elements següents:

- Convertidor corrent de 5V
- Bateria
- Cable adaptador USB-MiniUSB
- Instruccions
- Smartphone

De tots els elements que solen incloure els smartphones actuals s'han exclòs els auriculars i els adaptadors per a diferents mides d'orella. Els motius de l'exclusió són la falta d'espai útil de treball del Gantry ja que quedava una superfície molt atapeïda de components, i per altra banda, al tractar-se d'un prototip, la representació de la intenció final quedava suficientment demostrada amb els elements inclosos en el llistat anterior.

10.1.2 IMPRESSIÓ 3D

En la fabricació de l'element terminal i el tercer eix s'ha utilitzat un programa de modelat en 3D per veure com quedaria el muntatge abans d'imprimir i poder fer les modificacions necessàries durant el procés. La simulació s'ha realitzat mitjançant el programa de disseny CAD en 3D SolidWorks que ha permès representar cadascuna de les parts dels dos components i fer un assemblatge final.

10.1.2.1 Desenvolupament de la pinça

En l'aplicació del Gantry s'ha donat prioritat al disseny de la pinça degut a que era l'element més complex i que havia de tenir més precisió. Com s'ha dit anteriorment, es necessitava una obertura determinada per part de la pinça perquè pogués agafar des del component més gran fins al component més petit. Per això, s'ha dissenyat un sistema pinyó-cremallera similar al previst per a l'eix Z.

Com es podrà veure a la Figura 24, per fer-ho s'ha unit un engranatge a l'eix del servomotor i després s'han acoblat a l'engranatge dues barres dentades unides als dos braços de la pinça. Les barres estan situades una a la part superior i l'altre a la inferior del pinyó. D'aquesta manera, quan el pinyó giri rebran moviments contraris i podran obrir i tancar la pinça mantenint sempre un punt central.

A continuació es pot veure la nomenclatura donada als components que formen la pinça i el disseny final d'aquesta.

1- Pinyó o engranatge
2- Cremalleres
3- Braços
4- Suport dels braços
5- Motor
6- Suport del motor
7- Guia de la cremallera superior
8- Guia dels braços
9- Unió amb l'eix vertical

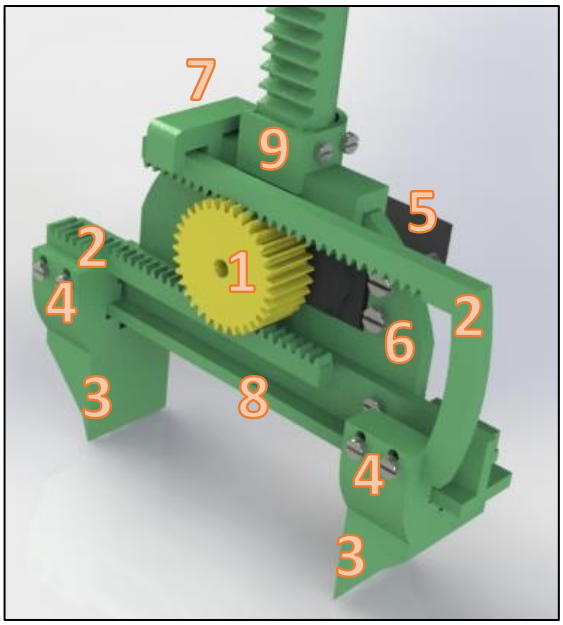


Figura 24: Nomenclatura de la pinça

Podem dir que l'obertura de la pinça és realment una conversió de l'angle de gir del servomotor que depèn de l'engranatge utilitzat. En aquest cas, s'ha dissenyat un engranatge que permetés una obertura aproximada de 10 cm per poder agafar la tapa del package de l'smartphone que és l'element més gran. El disseny de la pinça permet un muntatge desplaçat, respecte els 0º del motor, de les barres dentades que estan en contacte amb l'engranatge principal. Això significa que en el muntatge, quan el motor està en repòs, les barres es poden col·locar més separades del centre. D'aquesta manera tindrem una obertura major i a l'hora de tancar, en comptes d'arribar al tancament total, la pinça quedarà oberta un parell de centímetres, suficient per agafar el cable que és el component més estret.

Degut al muntatge de la pinça, quan el servomotor rep una consigna d'angle gran la pinça es tanca i quan rep un angle petit s'obre. Per tant, l'estat de la pinça serà normalment obert amb un angle de 0º i es tancarà en funció de la consigna donada entre aquests 0º i els 180º que indicarien un tancament total.

Per dissenyar l'engranatge principal, que és de qui dependrà la distància a obrir i tancar, s'han utilitzat els càlculs següents:

Els càlculs del pinyó s'han basat en l'obertura màxima que havia de tenir la pinça que són 10 cm. Aquests 10 cm equivalen al perímetre de la circumferència que ha de tenir el pinyó.

Sabent que el diàmetre primitiu correspon al punt de contacte de les dents i amb el qual es calculen les característiques d'un l'engranatge.

$$Perímetre = \pi * Ø_{Primitiu}$$

Per tant, calculant amb mil·límetres,

$$100 = \pi * Ø_P$$

$$Ø_P = 31.831 \cong \mathbf{32\ mm}$$

Tot i els càlculs realitzats cal recordar que estem tractant un servomotor de 180º i que per tant, el perímetre d'actuació o de contacte amb la cremallera és la meitat del calculat. Per altra banda, el muntatge de les dues cremalleres és contrari i quan una es mou en una direcció l'altra ho fa cap a l'altre. És a dir, amb el mateix gir del motor les pinces s'obren per igual cap a les dues bandes. D'aquesta manera es contraresta que el motor només giri mitja circumferència.

Un altre paràmetre que caracteritza un engranatge és la quantitat de dents (z) i el mòdul (m).

$$Ø_P = m * z$$

En aquest cas s'ha escollit un valor de mòdul normalitzat de manera que el tamany de les dents sigui suficientment gran per tenir una bona superfície de contacte amb la cremallera.

Sabent:

- Addendum o cap (a): Distància entre la circumferència primitiva i la part superior de la dent.

$$a = m$$

- Dedendum o arrel (b): Distància entre la circumferència primitiva i la part inferior de la dent.

$$b = 1.25 * m$$

- Altura total de la dent (h): Suma del cap i l'arrel.

$$h = m * (1.25 + 1)$$

Deixant un valor de mòdul unitari queda una profunditat de dent de 2.25mm, suficient per engranar correctament en l'obertura i tancament de la pinça.

Per tant, amb un valor de mòdul unitari, la quantitat de dents serà igual al diàmetre primitiu.

$$\boldsymbol{z = 32\ dents}$$

Una vegada fet l'assemblatge en simulació 3D i veure les dimensions finals de la pinça s'ha procedit a la impressió de totes les parts. Tot i això, la primera impressió no va sortir com era d'esperar i es van haver de repetir diferents components. Un cop totes les parts estaven impreses correctament es van unir utilitzant el material que es pot veure a la taula 2.

Material	Quantitat
Cargols M3x10	4
Cargols M3x16	1
Cargols M3x20	3
Cargols M3x25	1
Cargols M4x16	5
Cargols M4x20	4
Femella M3	7
Femella M4	9
Volanderes	9

Taula 2: Material d'unió utilitzat per a l'element terminal

Una de les modificacions en el disseny de la pinça durant el desenvolupament és que inicialment no incloïa la guia de la barra superior. Aquesta s'ha afegit més endavant ja que al realitzar les primeres proves es va veure que quan la pinça quedava molt oberta la cremallera superior no tenia un bon contacte. El problema era que els engranatges saltaven respecte els del pinyó quedant-se la pinça sense el moviment d'un braç. Això feia que la referència del motor es perdés ja que aquest continuava girant encara que els braços no es moguessin.

10.1.2.2 Desenvolupament de l'eix vertical

Una vegada dissenyat el conjunt de la pinça s'han pogut conèixer les mides reals d'aquesta i a partir d'aquí preveure quina altura necessitaríem aixecar la pinça per poder fer el packaging sense col·lisions entre components. A part d'aixecar la pinça una certa distància també caldria aixecar tot el robot mitjançant un suport degut a que amb l'altura que donen les potes dels motors no és suficient per no tocar a la base.

Amb l'ajuda de l'esbós de la Figura 25 podem fer-nos una idea de la distància necessària per a l'eix i per al suport. Com es pot observar, la caixa central del packaging fa 5cm, mentre que la tapa superior en fa 5.5cm. Per altra banda, la pinça muntada té una altura final de 12cm.

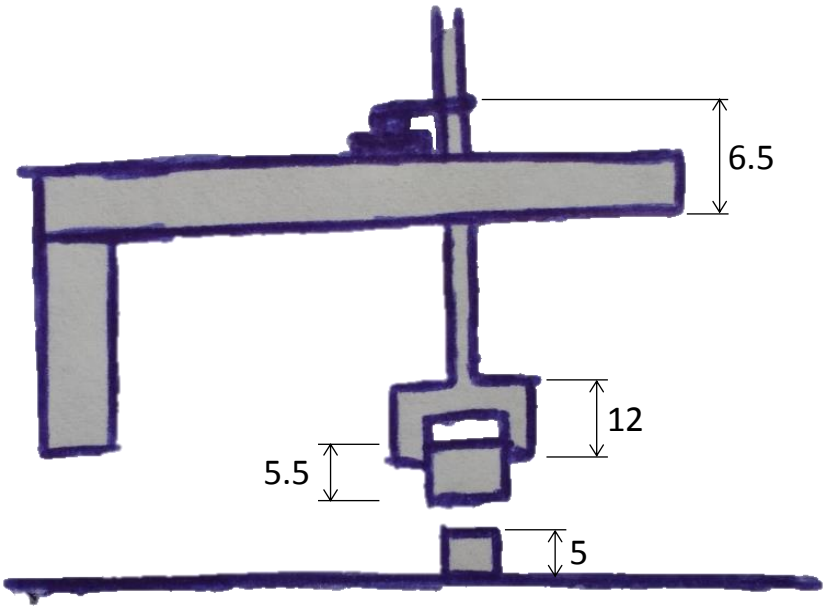


Figura 25: Esbós de l'altura del tercer eix i el suport

Això significa que el tercer eix havia d'alçar-se una altura mínima de 10.5cm. A aquesta altura mínima s'hi ha afegit una seguretat de 4cm, 2cm per l'espai que hi pugui haver entre la guia dels braços i la tapa, i 2cm més entre la tapa i la caixa central. Amb aquests espais de seguretat

l'altura mínima passa a ser de 14.5cm. A més, s'ha de sumar la distància de 6.5cm que hi ha entre la part inferior del robot i on tenim l'engranatge que fa funcionar el sistema. La impressora XYZ Da Vinci 1.0 utilitzada no era capaç de materialitzar peces més llargues de 20cm.

Degut a la llargada final necessitada pel tercer eix s'ha dividit en 2 parts per complir amb les característiques de la impressora utilitzada. Per altra banda, per la forma semicircular que se li ha donat a l'eix per estalviar material però que a l'hora no permetés una rotació, s'ha hagut de dividir cadascuna de les parts anteriors en dues més. En total 4 parts a imprimir que formaran un sol eix. El motiu d'aquesta última divisió es troba en la tecnologia utilitzada en la impressió 3D. La Da Vinci utilitza la tecnologia més coneguda actualment que és la deposició de material plàstic mitjançant la fusió. S'abrevia com a FDM per les sigles en anglès de *Fusion Deposition Modeling*. Aquesta tècnica utilitza plàstics com l'ABS o el PLA com a material mare i els extrueix depositant-los en diferents capes sobre una base sòlida. El procés d'extrusió implica l'escalfament del material fins al punt de la fusió, per això s'ha partit l'eix vertical per la meitat de la part semiesfèrica. D'aquesta manera la impressió es produiria, en cadascuna de les 4 parts, situant l'àrea més gran (pla central de l'eix) a la base de la impressora i a partir d'aquí les capes serien cada vegada d'àrea més reduïdes. Si la impressió es realitzés començant per la part de menor superfície a la base de la impressora, les capes superiors anirien depositant-se poc a poc a la base degut a la fusió del material en el procés d'extrusió i l'efecte de la gravetat.

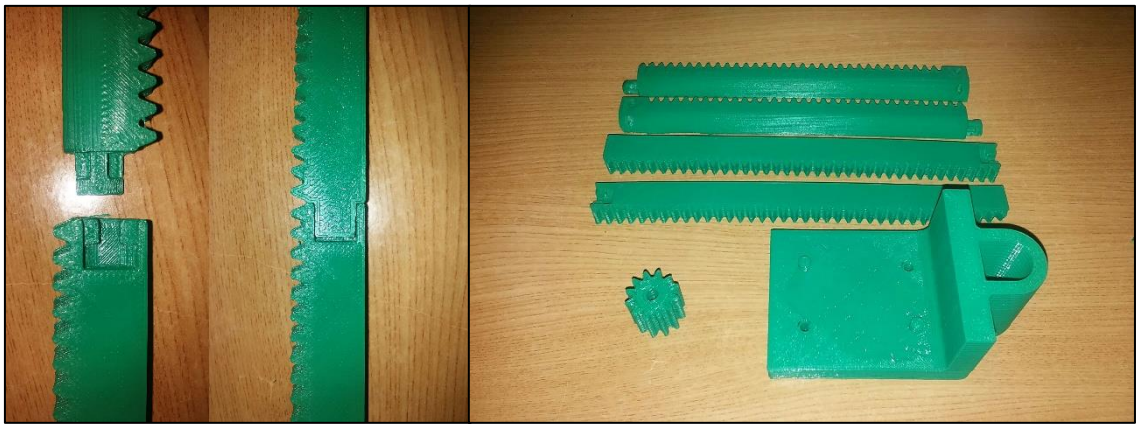


Figura 26: Impressió per parts de l'eix vertical

Una vegada impreses les 4 parts de l'eix calia unir-les de manera que quedessin suficientment fortes com per suportar els esforços de l'aplicació. Per unir les parts que separaven la semiesfera es va preveure des del disseny per SolidWorks i es va afegir una petita dent i un encaix a cadascuna de les dues parts respectivament com podem veure a la Figura 26. Amb això ja tenim l'eix vertical amb la llargada total però dividit en dues parts.

Per unir les dues meitats restants es podria haver optat per utilitzar algun tipus de cola però es temia per la corrosió que algunes coles poden causar en materials plàstics. Per això es va utilitzar una tècnica que consisteix en passar per les parts que es volen unir, un drap humit amb dissolvent. Com és d'esperar, el dissolvent desfà el material plàstic, però si es fa amb mesura, permet debilitar el plàstic de tal manera que després d'aplicar-hi pressió i assecar-se el dissolvent, queda una forta unió.

A part del tercer eix, en aquest sistema pinyó-cremallera també tenim un engranatge unit a l'eix del motor que permet moure la pinça verticalment. Aquest engranatge s'ha dissenyat amb unes dents prou profundes per assegurar una bona tracció de l'eix i evitar que la cremallera de l'eix vertical pogués saltar les dents de l'engranatge degut al pes de la pinça. A més, al poder controlar la velocitat del motor per programa, s'ha dimensionat en funció de les voltes de motor i la distància recorreguda.

A continuació es pot veure la nomenclatura donada als components que formen el sistema pinyó-cremallera.

1- Pinyó o engranatge
2- Motor
3- Suport del motor
4- Base de l'eix vertical

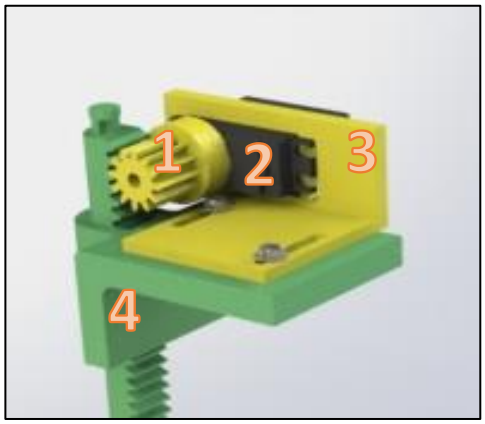


Figura 27: Nomenclatura del sistema de l'eix vertical

Tot seguit es mostren els càlculs utilitzats per al dimensionat del pinyó utilitzat.

Com s'ha dit, en l'eix vertical la característica que interessava per dimensionar l'engranatge era la profunditat de les dents.

Primer de tot s'ha escollit un valor de mòdul normalitzat de 1,5 per determinar l'altura de les dents. Amb aquest valor la profunditat de les dents (h) és prou gran per assegurar un bon contacte.

$$h = m * (1.25 + 1)$$

$$h = 1.5 * 2.25 = \mathbf{3.375\ mm}$$

En segon lloc, pel diàmetre de l'engranatge s'ha escollit un valor orientatiu que permeti la profunditat de dents calculada anteriorment. Després de fer diferents càlculs s'ha escollit un valor de diàmetre de 18 mm que permet un valor exacte de dents.

Per tant,

$$z = \frac{\emptyset_P}{m}$$

$$z = \frac{18}{1.5} = \mathbf{12\ dents}$$

A partir del diàmetre escollit es pot calcular el perímetre de l'engranatge,

$$Perímetre = \pi * \emptyset_{Primitiu}$$

$$P = \pi * 18 = \mathbf{56.548\ mm}$$

Amb els valors obtinguts i sabent que el motor pot assolir una velocitat màxima aproximada de 60 rpm, podem calcular la velocitat lineal màxima de l'eix.

$$\frac{60\ rev}{min} * \frac{1min}{60s} = 1\ rev/s$$

$$\frac{56.5\ mm}{1\ rev} * \frac{1\ rev}{s} = \mathbf{5.6\ cm/s}$$

10.1.3 CONFIGURACIÓ DEL SOFTWARE FCT

El software FCT és un software de la pròpia marca FESTO destinat a configurar els dispositius de la casa dotats de certa intel·ligència com són, en aquests cas, els manipuladors d'eixos elèctrics. Els software és molt intuïtiu i pensat per fer una configuració pas a pas per tal de no deixar-se cap paràmetre a configurar.

El nom del robot EXCM-30 és en realitat una abreviatura de EXCM-30-300-260-KF-SB-B2-E3-ES. Aquesta referència ens mostra les característiques principals del manipulador. En la següent taula s'explica breument què significa cadascuna d'aquestes referències.

Referència	Paràmetre	Significat
EXCM	Tipus	Pòrtic amb 2 eixos de moviment
30	Tamany	30
300	Carrera en l'eix X	300 mm
260	Carrera en l'eix Y	260 mm
KF	Guia	Cargol de boles
SB	Classe de motor	Pas a pas amb fre
B2	Muntatge dels motors	Inferior, sortida de cables enrere
E3	Controlador	Inclou controlador i cables dels motors (1 m)
ES	Idioma documentació	Documentació entregada (Castellà)

Taula 3: Referència completa EXCM-30

Un cop conegudes les característiques del robot el pas següent ha sigut estudiar la controladora per saber com s'havia de configurar. Conèixer bé la controladora ha comportat certa facilitat en la comunicació de senyals per donar les ordres de forma correcta al robot.

Per explicar els passos seguits d'una forma més visual s'han utilitzat diferents imatges del manual original de la controladora.

La Figura 28 mostra les diferents interfícies de comunicació que podem trobar a la controladora i que han estat breument comentades en els apartats anteriors. Recordem que en aquest cas s'ha utilitzat la interfície d'entrades i sortides digitals I/O [4] per a la comunicació amb el nivell de control superior Arduino.

Per altra banda, s'avança que per a la parametrització i configuració s'ha utilitzat la comunicació via Ethernet [5].

Una altra part important de la controladora és la filera 3 indicadors LED que donen informació del funcionament i el display de 7 segments destinat a la codificació d'errors. Es poden veure els 4 indicadors a la part esquerra del dibuix de la controladora següent.

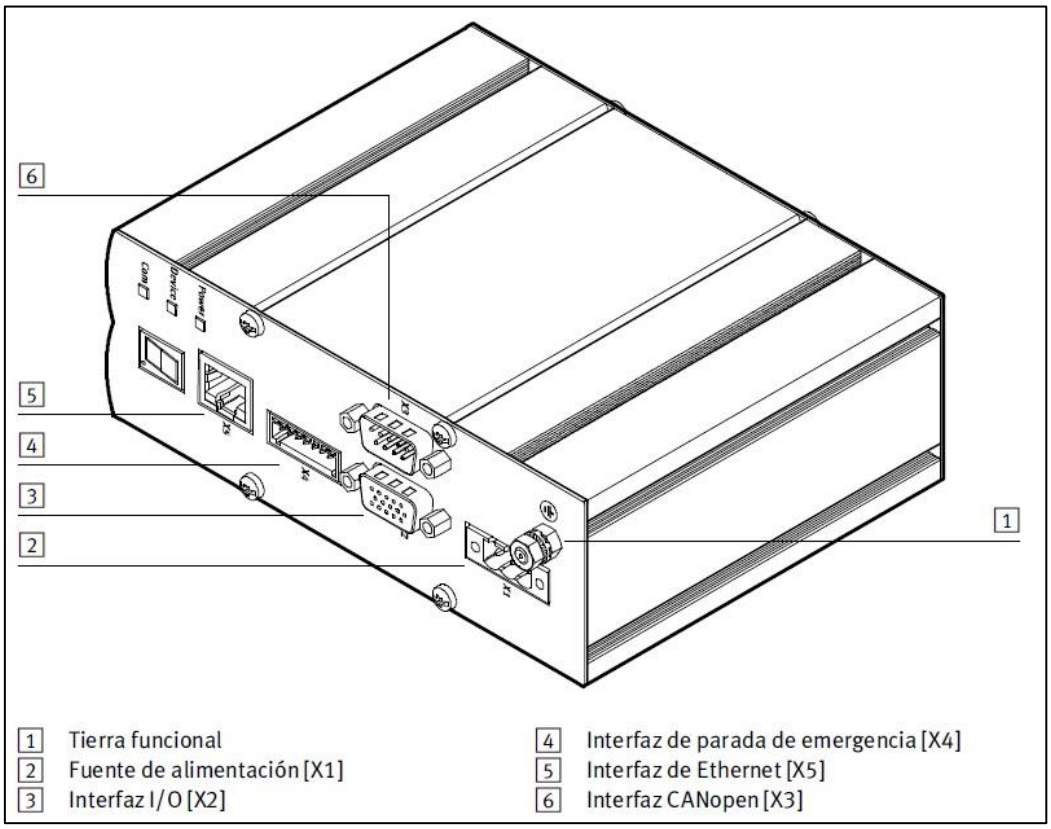
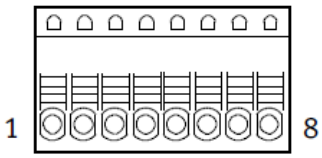


Figura 28: Interfícies de comunicació del EXCM-30

En la mateixa Figura 28 es pot observar la interfície de parada d'emergència [X4]. Tot i això, en la implementació de l'aplicació no s'ha integrat cap parada d'emergència. Aquesta es podria integrar ja que suposaria un canvi mínim en el disseny. Per altra banda, és important configurar correctament aquesta interfície ja que és la que permet donar tensió als motors i activar-los o desactivar-los. En la Taula 4 es pot veure el significat intern de cadascun dels 8 pins d'aquesta interfície.

Conexión	Pin	Función	
Interfaz en el controlador  Conector lado de conexión 	1	+24 V lógica	Salida: tensión de la lógica +24 V
	2	TO	Entrada: interrumpir la tensión de alimentación de los motores (con 0 V)
	3	ES	Entrada: activar rampa de frenado (con 0 V)
	4	RB	Entrada: soltar freno (con +24 V)
	5	FAULT	Salida: hay un fallo (con +24 V)
	6	DIAG1 (Contacto 1)	Los contactos de diagnosis están libres de potencial. El contacto de diagnosis es de baja impedancia cuando la alimentación del excitador está desconectada (contactos de diagnosis 1 y 2 puenteados).
	7	DIAG2 (Contacto 2)	
	8	0 V (GND)	Potencial de referencia

Taula 4: Interfície de parada d'emergència del EXCM-30

Els motors utilitzats en el robot tenen fre i aquest està normalment activat, per desactivar-lo i permetre un moviment lliure del robot s'ha de donar una tensió positiva a través del pin 1 als pins 2 i 4. Al pin 3 també s'hi aplica una tensió positiva per obviar la rampa de frenada que s'hauria configurat prèviament a través del FCT.

Una vegada realitzades correctament les connexions de hardware externes, s'ha configurat la controladora d'acord amb les característiques del manipulador.

Per fer-ho s'ha utilitzat el software comentat anteriorment *Festo Configuration Tool* (FCT). No es realitzarà una explicació detallada de cada paràmetre a configurar però sí que es comentaran els que han estat més destacables.

Seguint l'ordre que es mostrarà en el programa indicat, el següent llistat mostra de forma resumida les opcions disponibles i les configuracions més importants realitzades des del software esmentat.

Components

A partir de les característiques del robot utilitzat,

- **Elecció del component a parametritzar:** EXCM-30
- **Tamany:** 30
- **Mides robot:** Llargada dels eixos reals 300x260 mm
- **Fre de motor:** Disponible
- **Posició del motor:** Inferior

- **Interfície de control:** Per a la parametrització s'ha utilitzat la interfície Control Via Ethernet CVE, en l'aplicació final s'ha utilitzat el protocol I/O.
- **Port:** En el control CVE podem adjudicar un port. S'ha deixat 49700 per defecte.

Gantry

- **Limitacions de velocitat i acceleració generals**
- **Parada ràpida**
- **Posició Homming:** S'ha utilitzat l'extrem inferior esquerra del pla XY del robot.
- **Centre de coordenades:** És el 0 absolut i fa el paper de centre de coordenades, d'aquesta manera tots els punt de moviment estan referenciats a aquesta posició. En la l'aplicació el centre de coordenades i la posició de Homing coincideixen.
- **Velocitat i acceleració del Homing**

Controlador

- Amb una comunicació online amb el controlador, un botó permet capturar dades d'informació de la controladora i també l'adreça IP que té actualment. Per defecte ve una IP 192.168.178.1, per tant, s'ha configurat el port IP de l'ordinador des del Panell de Control de Windows en el mateix rang que la controladora (192.168.178.20).
- **Taula de punts:** Aquest és el punt més important de tots i s'explicarà amb detall al final de la llista.
- **Jog Mode:** Aquesta pantalla permet configurar el robot de manera que es mogui utilitzant una forma trapezoïdal. Això significa que la velocitat anirà augmentant durant el temps desitjat fins a arribar a una velocitat constant. Un cop a punt d'arribar a la posició indicada, la velocitat començarà a reduir-se en rampa fins a aturar-se just al punt de destí.
- **Llaç tancat:** Aquests són els paràmetres del PID intern i no s'han modificat.

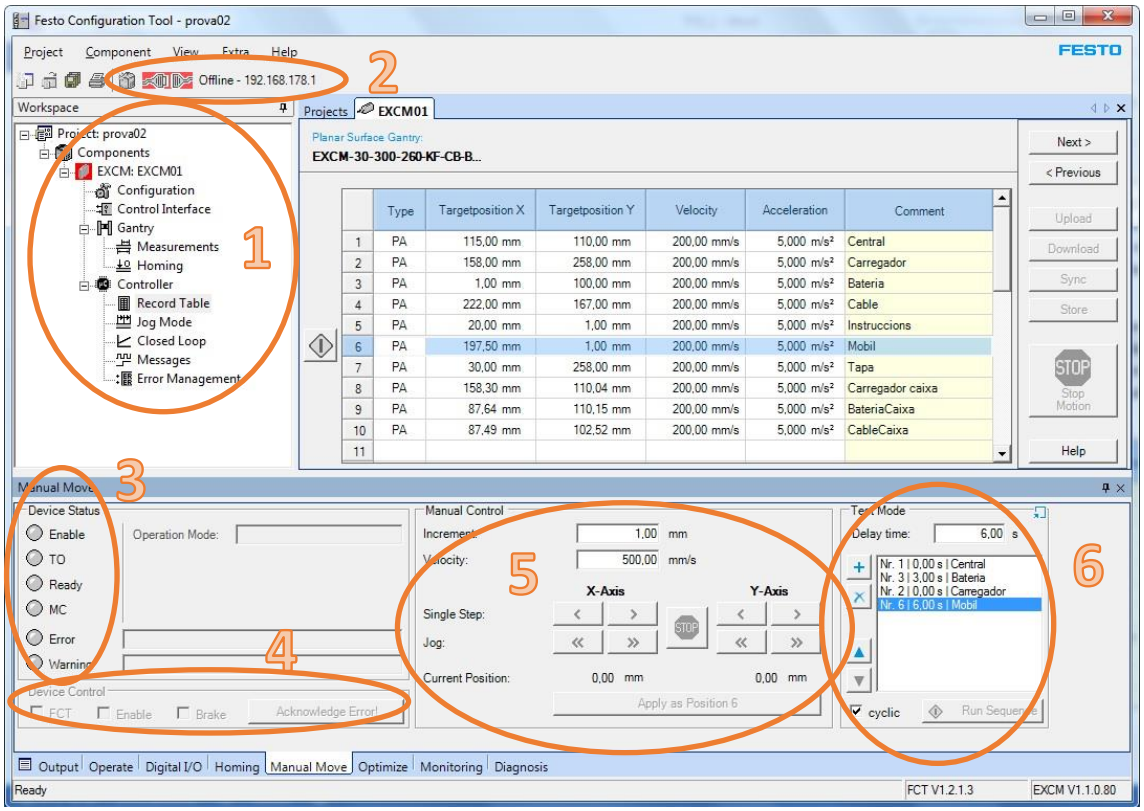


Figura 29: Captura pantalla FCT

En la captura de pantalla del FCT de la **¡Error! No se encuentra el origen de la referencia.** es mostra la taula de frases utilitzada en la l'aplicació i la seva configuració. Analitzant les columnes veiem que la primera indica el tipus de moviment que pot ser absolut respecte al centre de coordenades (PA), relatiu a la posició nominal (PRN) o relatiu a la posició actual (PRA). En l'aplicació final s'han determinat les posicions com a posicions absolutes, per tant, estaran referenciades al centre de coordenades.

Les dues segones columnes indiquen la posició del carro del robot en mil·límetres, la tercera i la quarta indiquen la velocitat de desplaçament i l'acceleració. Tant la velocitat com l'acceleració queden molt per sota de les seves possibilitats ja que es podrien augmentar fins a 500 mm/s i 10 m/s^2 respectivament. S'han fixat valors de 200 mm/s i 5 m/s^2 per la seguretat global del sistema. Finalment ens podem ajudar d'un petit comentari per saber quina és cadascuna de les posicions.

En la mateixa figura s'observa un polsador virtual que permet la comunicació Online amb la controladora via Ethernet i que a més mostra la IP configurada [2]. En la meitat inferior de la imatge es mostra una de les finestres més utilitzades en les proves que és la del moviment manual.

Dintre d'aquesta, el punt 3 mostra l'estat en què es troba la controladora. Si les connexions realitzades en la interfície de parada d'emergència abans comentada no són correctes, no es permet executar moviments i la senyal TO s'activa. Per altra banda, també indica una senyal MC (Motion Complete) que ja es veurà que ha sigut molt útil a l'hora de coordinar moviments. En la part inferior trobem els indicadors *Error* i *Warning* que mostren al programador quin és el problema mitjançant les línies de text de la part dreta del cercle 3. Aquests mateixos errors surten codificats en el display mostrat a la Figura 28: Interfícies de comunicació del EXCM-30.

El punt 4 mostra el control del dispositiu, si la controladora és controlada des del software FCT, si està controlada externament o si els frens estan activats. El botó de la part dreta permet reconèixer els errors.

El punt 5 representa el joystick manual per poder moure el manipulador utilitzant petits increments. Aquest és el que ha permès capturar totes les posicions necessàries per omplir la taula superior amb certa precisió.

Per últim, al punt 6 podem veure una petita llista molt útil per provar les diferents seqüències amb les posicions escollides incloent retards si és necessari.

10.1.4 ELECTRÒNICA DEL GANTRY

La part que comporta el control del robot ha estat la més complicada de desenvolupar degut a la seva complexitat. Això es deu principalment a la dificultat de treballar amb diferents voltatges, imposar l'Arduino com a controlador de nivell superior i el fet d'haver de controlar els dos servomotors de forma coordinada amb els moviments del robot.

Una vegada mogut el robot des del FCT era hora de provar de comunicar i fer moviments des del controlador Arduino. Per això s'ha utilitzat la interfície I/O finalment escollida per a la comunicació entre controladors. Com podem veure a la Figura 30, el connector que podem trobar a la controladora és una femella estàndard del tipus DB 15 de 3 files, similar al connector estàndard VGA.

En la mateixa es mostren el significat de cadascun dels pins de la interfície. En el nostre cas s'han utilitzat els pins 2-6 per a la selecció de frase i els pins 8 i 9 per a iniciar el moviment. Aquests són pins d'entrades a la controladora que vindran des de l'Arduino. Per altra banda, com a sortida de la controladora i entrada a l'Arduino, s'ha utilitzat la senyal de la controladora MC (Motion Complete). Aquesta permet conèixer quan el robot havia acabat un moviment i era capaç de gestionar una altra ordre.

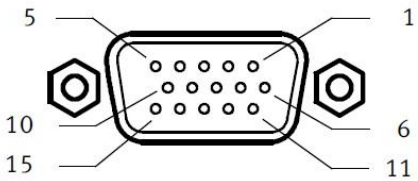
Conexión	Pin	Función	
	1	24VL	Salida: listo para la comunicación
	2	DI 1	Entradas: selección de frase
	3	DI 2	
	4	DI 3	
	5	DI 4	
	6	DI 5	
	7	6 DI	No se utiliza
	8	START	Entrada: inicio de frase
	9	ENABLE	Entrada: desbloquear regulador
	10	RESET	Entrada: validar fallo
	11	Enabled	Salida: desbloquear regulador
	12	FAULT	Salida: fallo
	13	Ack	Salida: validación
	14	MC	Salida: Motion Complete
	15	O V	Potencial de referencia

Figura 30: Interfície I/O de la controladora EXCM

Per altra banda, també s'ha utilitzat el pin 15 com a potencial de referència degut a que la controladora treballa amb lògica inversa.

El fet de treballar amb lògica negativa indica que els pins de senyal connectats a la controladora del Gantry en realitat entreguen els 24v i el que s'ha de fer és tancar el circuit de cada senyal

amb els 0v a través dels optoacobladors. En la següent imatge es mostra un exemple d'aquesta connexió a través d'un optoacoblador.

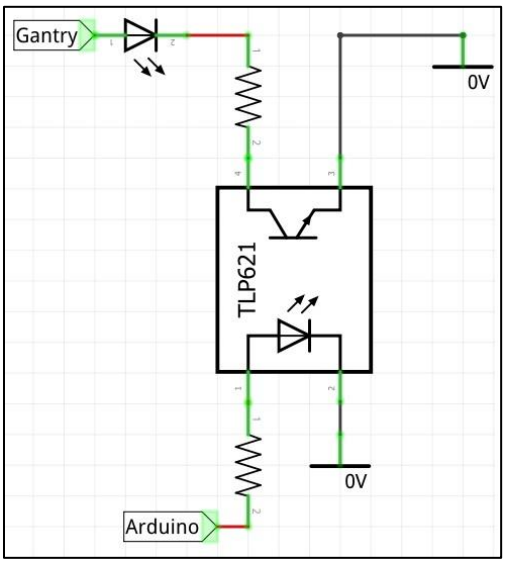


Figura 31: Exemple de la connexió d'un optoacoblador

La característica d'aquest connector, igual que el VGA, és que cadascun dels pins té l'obligació de governar una senyal concreta, permetent així, poder enviar senyals individuals pels diferents pins. El connector estàndard VGA no utilitza els 16 pins del hardware i a més alguns pins són comuns a GND. Al no utilitzar-los, els fabricants ja no connecten internament els pins innecessaris per estalviar material. Per altra banda, veiem que la disposició dels pins de la controladora és important ja que cadascun té una funció i no es pot canviar. Per tant, s'ha vist que alguns dels pins no utilitzats en el connector VGA eren necessaris per a la comunicació I/O. Per aquest motiu s'ha hagut de buscar un connector mascle DB-15 de 3 files i cablejar manualment els pins per poder comunicar amb l'Arduino.

A continuació es mostra una imatge del connector DB-15 utilitzat en el protocol VGA. En la mateixa imatge es subratllen dos dels pins necessaris (4 i 9) que no estan cablejats.

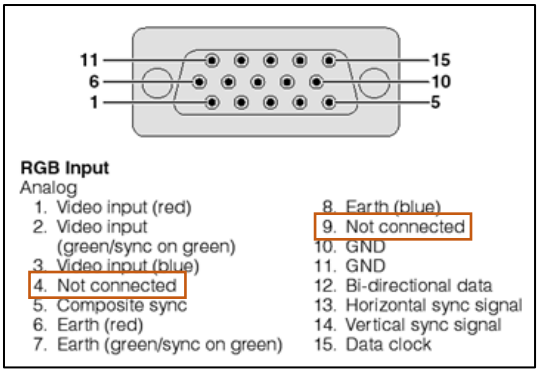


Figura 32: Cablejat del connector VGA

Per altra banda, per fer la interfície electrònica entre el controlador de 5v Arduino i la controladora pròpia del robot que treballa amb 24v s'han escollit els optoacobladors CNY74-4. Aquest component en concret està format per un conjunt de 4 optoacobladors units com si es tractés d'un circuit integrat.

A part dels CNY74-4, per poder fer una comprovació de forma senzilla i ràpida del funcionament, s'han col·locat uns díodes LED a la sortida de l'optoacoblador per saber si realment el component enviava alguna senyal a la controladora. Amb aquests LED podem veure quina senyal de les comentades en la Figura 30 està activada. D'aquesta forma tindrem 7 senyals de sortida, que faran una conversió de 5v a 24v, i 1 sola d'entrada que serà la senyal de MC i farà una conversió de 24v a 5v. Per protegir cada LED i limitar la intensitat que circula per el CNY74-4 s'han col·locat unes resistències a les entrades dels optoacobladors.

L'Arduino UNO és el controlador més venut i més conegut de la plataforma, però tot i això té certes limitacions en quant a entrades i sortides. Aquestes es divideixen en 6 entrades analògiques i 14 pins digitals dels quals 6 permeten una sortida mitjançant PWM (*Pulse Width Modulation*). Aquests últims simulen senyals analògiques generant senyals quadrades a 5v, però amb una llargada de puls que és equivalent al voltatge desitjat. Per exemple, si fem una senyal quadrada amb amplitud de 5v que duri la meitat del període com podem veure a la Figura 33, la senyal de sortida serà equivalent a una senyal de 2.5v.

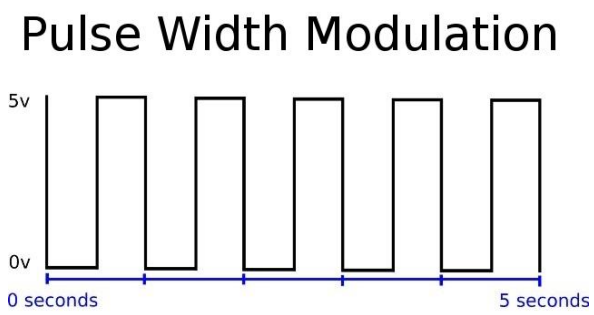


Figura 33: Exemple de senyal PWM

El pins de sortida que permeten fer servir la tècnica del PWM s'han utilitzat per a controlar els dos servomotors de la pinça i l'eix vertical.

Com es pot observar, el controlador està força limitat amb un total de 20 senyals. Per aquest motiu, des d'un principi es va pensar en estalviar la màxima quantitat de pins possible per tal de poder integrar totes les funcions que poguessin sorgir. Per això s'ha utilitzat un registre de desplaçament 74HCT595. A grans trets, aquest component permet emmagatzemar 8 bits de dades entrades en sèrie i fer posteriorment, d'acord a una senyal de rellotge, una sortida en

paral·lel d'aquestes. D'aquesta forma, utilitzant 3 senyals d'entrada el 74HCT595 permet gestionar-ne 8 de sortida.

En el cas de l'aplicació només s'han utilitzat 5 de les 8 senyals de sortida per la selecció de frases (pins 2-6, Figura 30). Inicialment es va provar d'incloure la senyal d'inici de frase, però la controladora necessita que aquesta senyal arribi més tard que la frase en sí. Per aquest motiu, la senyal d'inici de moviment s'ha col·locat en un pin individual de la placa Arduino. Per tant, primer s'envia la frase seleccionada a la controladora del Gantry mitjançant el registre de desplaçament i seguidament l'Arduino dóna l'ordre perquè el robot iniciï el moviment. Una vegada el robot arriba a la posició, la senyal de moviment complert MC rebrà un puls. Aquest puls és captat per l'Arduino i a partir d'aquí pot començar el següent cicle enviant una altre frase a la controladora.

10.1.4.1 Coordinació del robot amb el tercer eix i la pinça

Com ja sabem, els moviment del Gantry s'havien de coordinar amb els moviments que generaven els servomotors de l'element terminal.

En el cas de la pinça, al no tenir cap tipus de senyal de retorn del tancament o l'obertura s'han utilitzat retards en el programa per esperar i assegurar l'obertura i el tancament dels braços. Com que la distància d'obertura depèn del component a agafar, s'han ajustat els retards manualment en cada moviment per optimitzar al màxim el temps de cicle de l'aplicació.

En canvi, la coordinació a través de software per controlar l'eix vertical ha sigut força més complicada d'implementar com es podrà veure en el pròxim apartat 10.1.5 PROGRAMACIÓ AMB ARDUINO.

Pel que fa al hardware de l'eix, degut als problemes de control, s'ha utilitzat un detector mecànic per saber quan la pinça estava a la posició superior. Aquest detector s'ha adjuntat a la base de l'eix vertical que fa d'unió entre el carro del Gantry i el propi eix. D'aquesta manera, quan la pinça puja i arriba a la part superior, una limitador situat a un parell de centímetres sobre la pinça toca aquest detector i envia una senyal a l'Arduino perquè aturi el moviment de pujada de l'eix vertical.

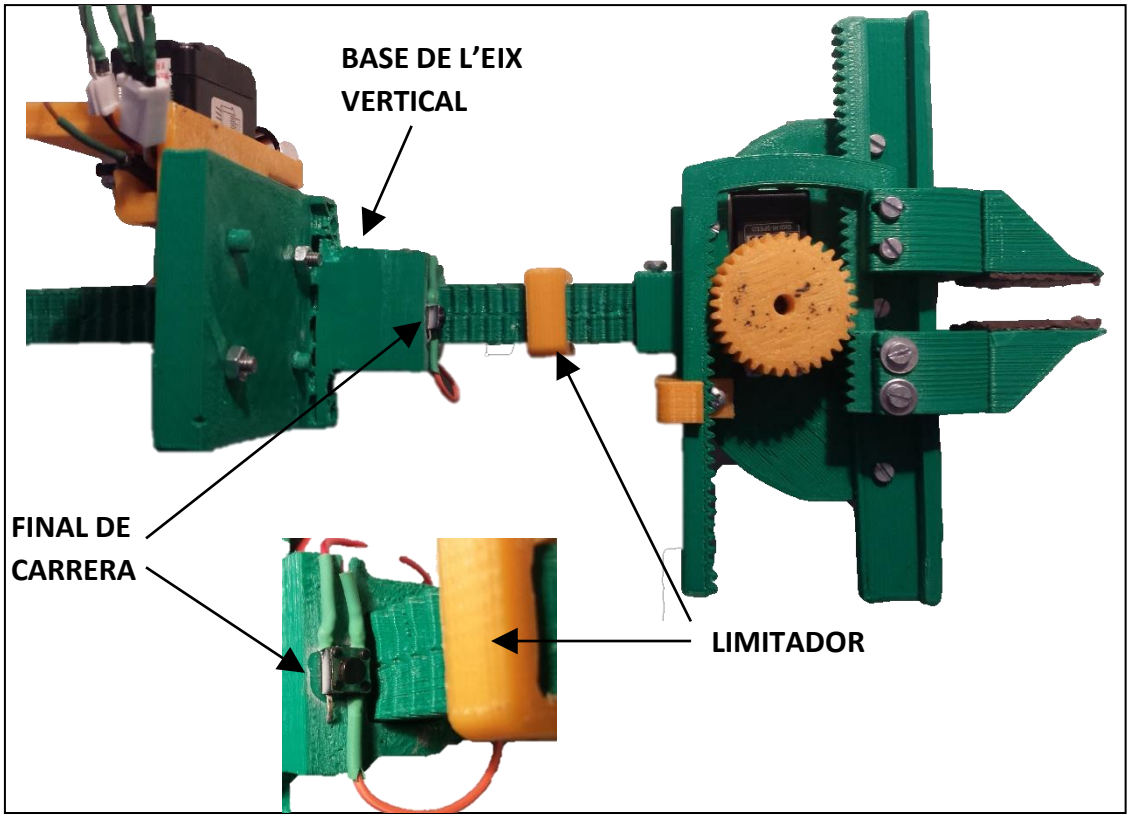


Figura 34: Situació del final de carrera de l'eix terminal

10.1.4.2 Cablejat de l'element terminal

El cablejat dels servomotors de la pinça, l'eix vertical i l'interruptor de final de carrera necessitaven una llargada aproximada d'1.5 metres per arribar a la placa de control i permetre el moviment del robot. Per aquest motiu s'ha optat per agafar cables de la mida desitjada i soldar-hi uns pins amb estany [1] simulant els pins dels cables que s'utilitzarien en una protoboard [2].

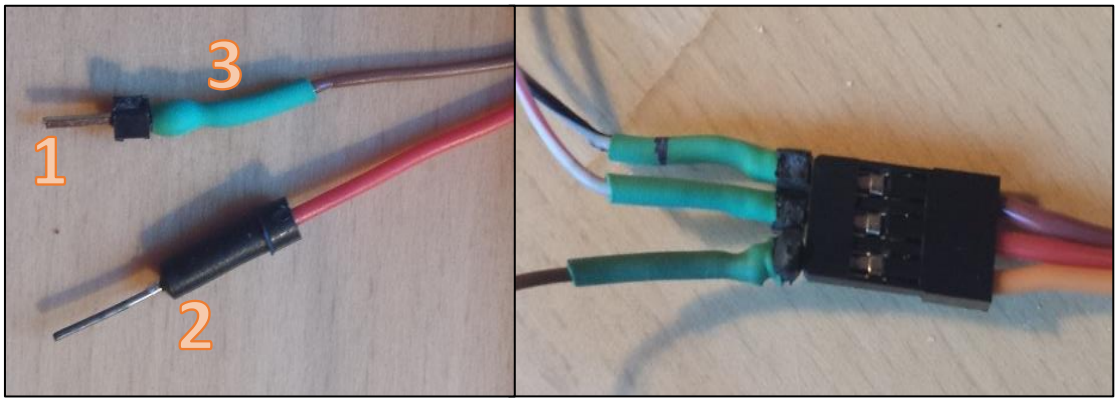


Figura 35: Exemple d'un connector manual i un de fabricat i la utilització en la connexió d'un servomotor

Si únicament es soldaven els pins quedava desprotegida la part de la soldadura i era perillós tenir pins tant propers entre ells ja que podrien tocar-se i generar falsos contactes. Per evitar aquests possibles errors s'han cobert totes les soldadures amb un tros de tub retràctil que es comprimeix en aplicar-hi calor aïllant així els contactes [3].

En total tenim 3 circuits de corrent independents a cablejar que van des de la pinça fins a l'Arduino. Els 3 circuits estan formats per alimentació entre 0 i 5v i la senyal a rebre o controlar. Aquests circuits corresponen al control dels dos servomotors i el pin de la senyal de final de carrera.

Per tant es necessiten 3 cables de 0v, 3 de 5v i 3 calbes de senyal. Per optimitzar el cablejat i gràcies a que el disseny de l'eix ho permet, s'ha adjuntat en la pròpia pinça una regleta per a unificar la part d'alimentació. D'aquesta manera, els únics cables que anirien fins al controlador serien els 2+1 cables d'alimentació i les 3 senyals corresponents a cadascun dels circuits. El significat dels 2+1 cables es detallarà en el següent apartat ja que ha sorgit la necessitat d'independitzar el cable negatiu de l'interruptor de final de carrera.

Com que els cables s'han creat manualment un a un, per facilitar la manipulació i alhora protegir-los s'ha utilitzat un organitzador de cables pel qual s'han introduït els 6 fils i d'aquesta manera arribar de forma unida fins la placa de control.

10.1.4.3 Implementació en una PCB prototip

Com s'ha presentat en l'abast del treball, el Gantry ha tingut certes modificacions per al treball final de grau i una de les més importants és el circuit electrònic. Aquest s'ha traslladat de la *protoboard* inicial a una placa de prototip PCB (*Printed Circuit Board*).

El fet d'implementar el circuit en aquest tipus de plaques li dona més robustesa i fiabilitat ja que les *protoboards* són molt útils per fer les proves però tendeixen a acabar fallant. Per fer el disseny del nou circuit s'ha utilitzat el software lliure Fritzing que permet fer el disseny dels circuits en una *protoboard*, en una PCB i per altra banda també permet extreure el circuit en format d'esquema elèctric.

Com es pot observar en l'annex 4, el disseny de la PCB ha seguit dues restriccions principals. En primer lloc les pistes han respectat els moviments de Manhattan eliminant els girs amb angles de 90º per evitar interferències. Per altra banda, s'han utilitzat les dues cares de la placa per tal d'evitar encreuaments de pistes.

Cal remarcar que aquest programa realitza el disseny d'una PCB real, no de prototip. Això significa que el disseny realitzat a través de Fritzing ha servit com a guia per al desenvolupament de la placa prototip i per tant, la distribució dels components no té perquè ser exacte al disseny.

Així doncs, seguint el muntatge de la *protoboard* inicial amb les modificacions de hardware i software realitzades, s'ha implementat mitjançant soldadura per estany en una PCB prototip a partir de l'esquema de l'annex 4.

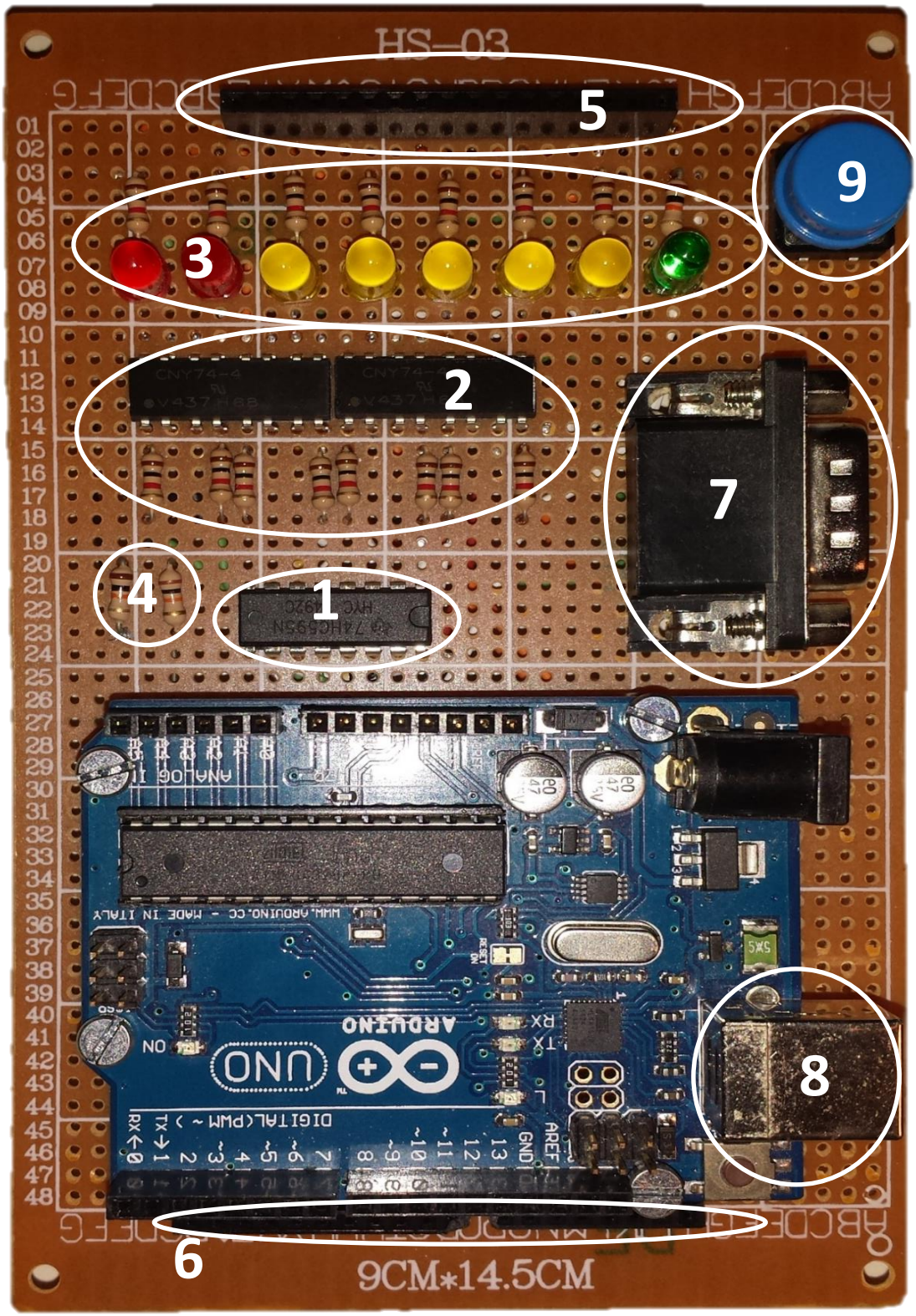


Figura 36: Implementació de l'electrònica en una PCB prototip

Tot seguit es llistaran i s'explicarà la funció principal dels components utilitzats en la placa, en referència a la numeració de la imatge.

1- Registre de desplaçament: Envia les frases seleccionades per l'Arduino en paral·lel cap als optoacobladors.
2- Dos grups de 4 optoacobladors: Realitzen la funció d'interfície entre els 5v de l'Arduino i els 24v de la controlador del Gantry. Cadascun esta cablejat amb les resistències de protecció a l'entrada.
3- Indicadors de senyal LED: Indiquen l'habilitació de la senyal corresponent, d'esquerra a dreta, amb la següent codificació:

    Vermell 1 – Senyal d'habilitació del manipulador
    Vermell 2 – Senyal d'inici de frase o marxa
    Groc 1 – Bit 1 de selecció de frase (LSB)
    Groc 2 – Bit 2 de selecció de frase
    Groc 3 – Bit 3 de selecció de frase
    Groc 4 – Bit 4 de selecció de frase
    Groc 5 – Bit 5 de selecció de frase (MSB)
    Verd 1 – Senyal MC o final de moviment

    Excepte el LED verd tots s'encenen en rebre la senyal, el bit MC està normalment encès i s'apaga momentàniament en finalitzar un moviment.

4- Resistències de les senyals de final de carrera i el polsador de marxa.
5- Interfície de senyals externes: S'hi connecten les senyals externes com els servomotors de la pinça i la senyal de final de carrera de l'eix.
6- Interfície entre l'Arduino i la PCB: S'hi connecten els pins utilitzats de l'Arduino per poder reaprofitar la placa Arduino i no soldar-la a la PCB.
7- Interfície de la controladora del Gantry: Utilitzant un cable especial es comunica amb la controladora del robot.
8- Interfície USB de l'Arduino: Utilitzat per a la comunicació sèrie amb l'OPC.
9- Polsador Marxa/Paro: Útil per a les modificacions i proves durant el treball sense la necessitat del funcionament de l'aplicació l'SCADA. Finalitzada la programació s'ha deixat com a polsador de marxa secundari.

Coneguts tots els elements utilitzats en la placa es pot aprofundir millor en els més significatius.

- **Resistències**

a) Eliminació de falsos contactes.

Per eliminar les falses senyals produïdes per soroll elèctric s'utilitzen les anomenades resistències pull-up o pull-down. Aquestes configuracions determinen respectivament que la senyal de sortida estigui connectada a un voltatge normalment positiu o a massa.

En el segon circuit de la Figura 37 es pot observar la configuració pull-down que ha estat la utilitzada per a la implementació de les senyals dels polsadors de marxa, final de carrera i MC.

Amb aquesta configuració la senyal de sortida Vout està normalment connectada a massa a través de la resistència i per tant es llegeixen 0v. Si premem el polsador S1, equivalent als polsadors utilitzats o a l'optoacoblador en el cas de la senyal MC, la corrent entregada per la font Vcc passa a la sortida Vout.

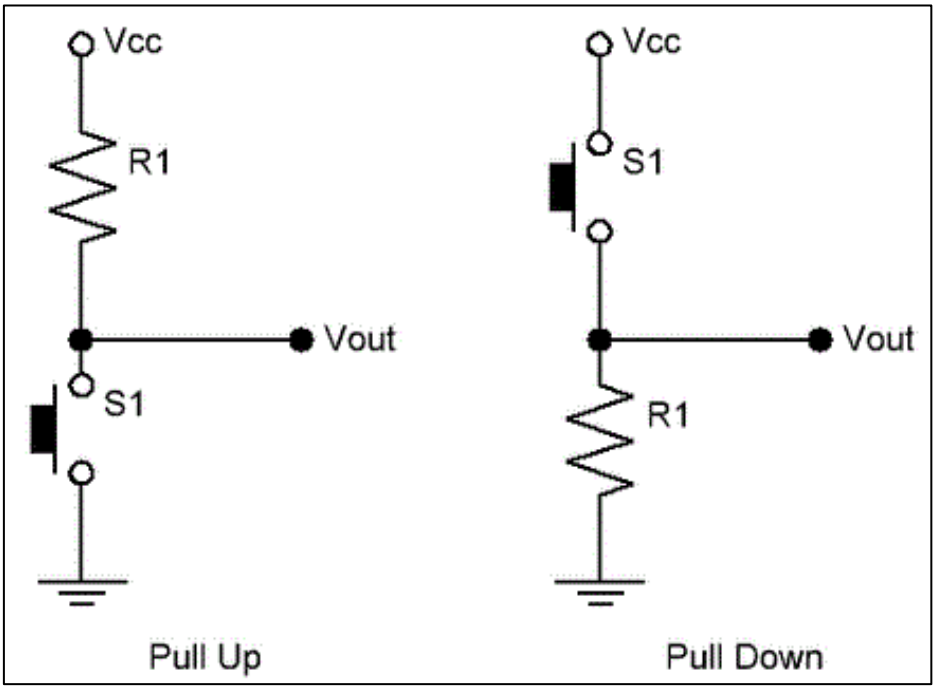


Figura 37: Resistències pull-up i pull-down

En l'apartat anterior s'ha comentat la necessitat d'independitzar el cable de 0v de l'interruptor final de carrera. El motiu és la inserció de la resistència pull-down aquí explicada.

Aquestes configuracions no demanen un valor de resistència exacte però es solen posar valor entre 1kΩ i 10kΩ. En la implementació del circuit s'han utilitzat resistències de 10kΩ.

b) Protecció de components

En el cas dels optoacobladors utilitzats, la corrent màxima d'entrada és de 60mA en estat de conducció normal i de 1.5 A en cas de pics de corrent inferiors a 10µs.

Sense la resistència, la corrent que circula pel component és aproximadament 75mA. Aquesta limitació de corrent es deu a una limitació interna de l'Arduino i depèn de la quantitat de pins activats. Degut a que ens trobem en el cas de més limitació de corrent s'ha necessitat la utilització d'una resistència per reduir la intensitat dins el rang recomanable pel fabricant. Amb una resistència de 1kΩ la corrent passa a ser de 4mA, amb un correcte funcionament del component i la disminució del consum global del circuit.

A la sortida de l'optoacoblador el fabricant recomana treballar a un valor màxim de 50mA en conducció. En aquest cas, la intensitat a controlar prové de la controladora del Gantry que treballa a 24v. A diferència de l'Arduino aquesta treballa amb una limitació de 100mA per a cada sortida i 2mA a cada entrada. Per tant, per limitacions de la controladora no seria necessari utilitzar cap resistència en les senyals d'entrada. Tot i això, per seguretat s'ha col·locat igualment una resistència de 1kΩ a cada sortida.

Tots els valors dels components s'ha comprovat i mesurat de forma experimental utilitzant un multímetre.

- **Interfície amb la controladora del Gantry**

Per enviar i rebre senyals de la controladora a través de la interfície I/O utilitzada s'ha inclòs un connector DB15 mascle per connectar mitjançant un cable propi amb la controladora. Els pins d'aquest connector estan cablejats a les sortides i l'entrada corresponent dels optoacobladors.

Pel que fa al cable utilitzat s'ha realitzat manualment mitjançant un cable multifilar de 10 fils i uns connectors DB-15 per connectar entre la controladora i el connector integrat a la placa.

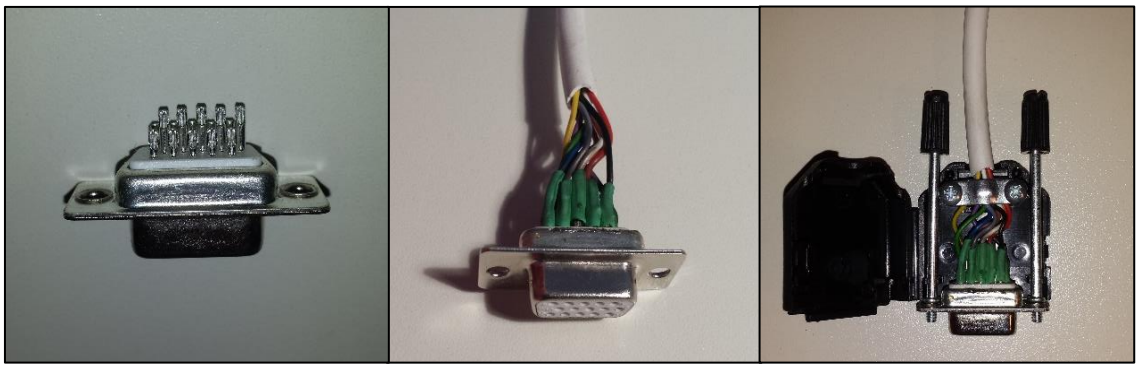


Figura 38: Fabricació del connector per a la interfície I/O

- **Interfície de senyals externes**

La regleta de connectors de la part superior de la placa permet connectar el cablejat provinent dels servomotors i de l'interruptor de final de carrera. Recordant que aquest interruptor necessita independitzar el cable de 0v per integrar-hi una resistència pull-down, la interfície de 17 connectors respon a la distribució següent d'esquerra a dreta:

1- Senyal del final de carrera
2- Cable negatiu de l'interruptor final de carrera
3- Senyal del servomotor de l'eix vertical
4- Senyal del servomotor de la pinça
5- Alimentació negativa de l'Arduino (0v)
6- Alimentació positiva de l'Arduino (5v)
7- Senyal d'habilitació del manipulador
8- Senyal d'inici de frases
9- Bit 1 de selecció de frase (LSB)
10- Bit 2 de selecció de frase
11- Bit 3 de selecció de frase
12- Bit 4 de selecció de frase
13- Bit 5 de selecció de frase (MSB)
14- Senyal de MC o final de moviment
15- No utilitzat
16- Alimentació positiva de la controladora (24v)
17- Alimentació negativa de la controladora (0v)

Com es pot observar s'han repetit les senyals utilitzades al connector DB-15 corresponents al controlador Gantry. El motiu de la repetició es deu a la possibilitat de tenir una segona opció en cas que fallés el connector de la placa o el cable fabricat manualment.

- **Interfície entre l'Arduino i la PCB**

Per poder reaprofitar la placa Arduino en un futur i no realitzar unes connexions permanents s'han integrat dues regletes per utilitzar com a mitjà per comunicar les senyals de control necessàries. La distribució que segueixen aquestes dues regletes continues d'esquerra a dreta és la següent:

1- Interrupció del polsador Marxa/Paro
2- No s'utilitza
3- No s'utilitza
4- Senyal de *data* del registre de desplaçament
5- Senyal de *Latch* del registre de desplaçament
6- Senyal de *clock* del registre de desplaçament
7- No s'utilitza
8- Senyal MC o de final de moviment
9- No s'utilitza
10- Alimentació negativa de l'Arduino (0v)
11- Alimentació positiva de l'Arduino (5v)

Segona regleta,

12- Senyal d'habilitació
13- Senyal d'inici de frase o marxa
14- No s'utilitza
15- Senyal de final de carrera
16- No s'utilitza
17- Senyal del servomotor de l'eix vertical
18- Senyal del servomotor de la pinça
19- No s'utilitza

Els pins 15, 17 i 18 estan connectats internament als pins 1, 3 i 4 de les senyals externes.

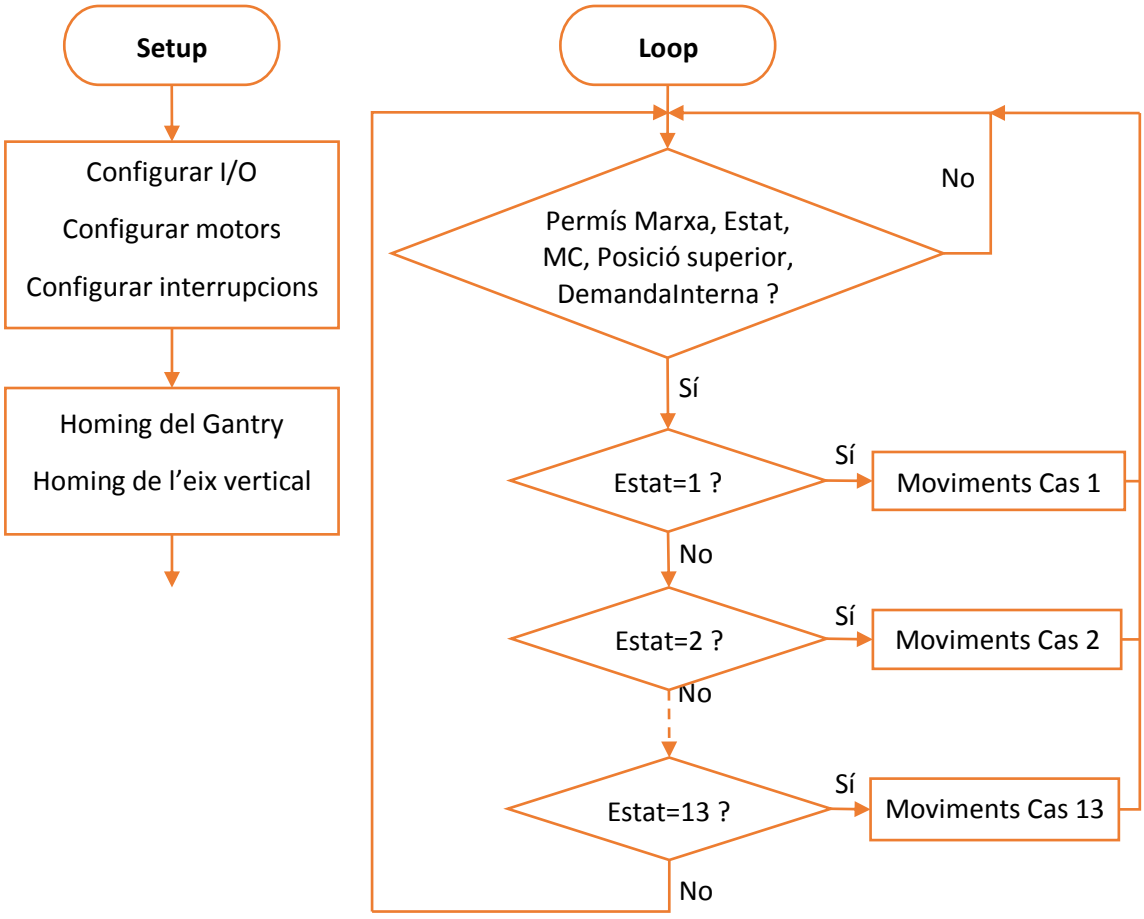
10.1.5 PROGRAMACIÓ AMB ARDUINO

Com s'ha anat dient, el cervell de tota aquesta primera part del treball és el controlador Arduino. Aquest controla des del moviment del Gantry fins al moviment de l'eix vertical i la pinça. Per tant, el programa gestiona una part de moviment i també una comptabilització dels mòbils processats. El conjunt d'aquestes dades serà comunicat via OPC i visualitzat en l'aplicació SCADA.

10.1.5.1 Funcionament del programa

A diferència del software que veurem a continuació per al robot de ABB, el controlador Arduino es caracteritza per estar organitzat en un programa d'un sol mòdul format per dues funcions principals. La funció ***Setup*** que s'executarà una sola vegada en iniciar el microcontrolador, i la funció ***Loop***, que pel seu propi nom serà una funció que es repetirà de forma cíclica formant un bucle. La primera funció sol destinar-se a realitzar una configuració i parametrització inicial per poder inicialitzar una aplicació. En el segon cas, trobarem el cos principal del programa.

A part d'aquestes dues funcions bàsiques i necessàries pel controlador, el programador en pot crear d'altres que seran cridades des de la funció cíclica principal. A continuació veurem el funcionament principal del programa a través dels diagrames de flux.



Prèviament a les funcions de **Setup** i **Loop** s'han definit i inicialitzat les variables a utilitzar al llarg de tot el programa.

La funció **Setup** s'ha dividit en dues parts diferenciades per la configuració i el moviment del robot a una posició inicial de *Homing* amb el Gantry al centre de coordenades i l'eix vertical recollit. Dins la configuració trobem la definició dels pins segons siguin entrades o sortides, els pins utilitzats com a PWM per controlar els servomotors i finalment les interrupcions. Les interrupcions suposen una pausa immediata del curs de programa principal i l'execució d'un programa secundari. Una vegada executat el secundari es torna al principal seguint el flux de programa que s'havia pausat per la interrupció.

En el cas dels servomotors cal incloure una llibreria que ofereix Arduino per poder treballar amb ordres complexes i el controlador sigui capaç de comprendre i processar.

La llibreria i l'objecte que s'utilitza es determinen de la següent manera:

- `#include <Servo.h>` — Afegim la llibreria Servo
- `Servo ServoP;` — Creem el servo que controlarà la pinça
- `Servo ServoZ;` — Creem el servo que controlarà l'eix vertical

Les interrupcions s'han definit en mode *RISING* ja que únicament interessa el flanc de pujada de la senyal d'entrada. En fer aquest canvi de voltatge s'executarà la interrupció corresponent una sola vegada encara que el pin quedi activat. En la definició cal indicar el número de la interrupció, la funció a executar quan succeeixin i el mètode d'activació.

```
- attachInterrupt(0, INTMarxaParo, RISING);
- attachInterrupt(1, INTMotionComplete, RISING);
```

La funció **Loop** comprova inicialment que els permisos de marxa i MC siguin correctes, que l'estat sigui diferent a l'anterior, que l'eix vertical es trobi en la posició superior i que la demanda total sigui inferior als dispositius empaquetats. Amb aquestes condicions assegurem que el Gantry no es mogui fins acabar els moviments de la pinça i que cada moviment s'executi una sola vegada. Si no es compleixen aquestes condicions el controlador esperarà.

Existeixen un total de 13 estats referents a cadascun dels moviments del carro del Gantry, independentment dels moviments de la pinça i l'eix.

Els diferents estats tenen una estructura molt similar excepte en l'últim cas, el tretzè. Al ser l'últim estat, un cop acabats els moviments, s'actualitzen els comptadors de smartphones empaquetats i el temporitzador de cicle.

Els estats comentats anteriorment són els següents:

Estat	Definició
1	Posició central
2	Recollir convertidor del carregador
3	Deixar convertidor del carregador a la caixa central
4	Recollir Bateria
5	Deixar Bateria a la caixa central
6	Recollir cable del carregador
7	Deixar cable del carregador a la caixa central
8	Recollir instruccions
9	Deixar instruccions a la caixa central
10	Recollir dispositiu mòbil
11	Deixar dispositiu mòbil a la caixa central
12	Recollir tapa
13	Deixar tapa sobre la caixa central

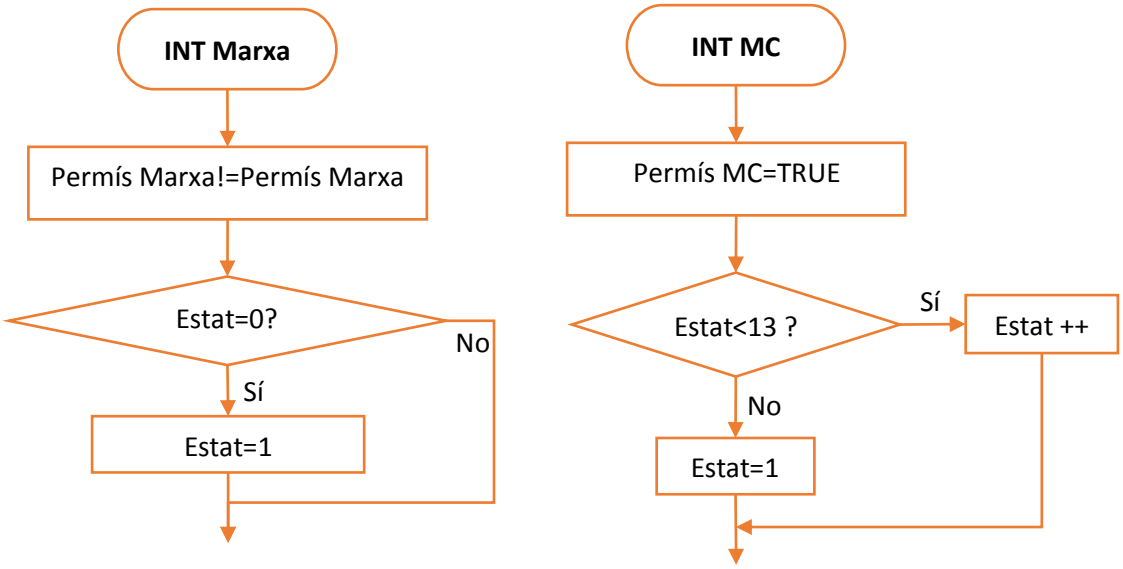
Taula 5: Definició dels diferents estats del robot

Dins la funció **Loop**, veiem que segons l'estat executem un moviment o un altre. L'estructura d'aquests moviments varia segons es tracti d'un moviment de recollida o entrega d'un component. Com que les instruccions dels dos moviments són molt similars es representa un moviment de baixada com a exemple.

- Crida de la funció de moviment del Gantry amb la posició corresponent a l'estat actual
- Obertura de la pinça
- Crida de la funció de moviment de l'eix vertical per baixar la pinça
- Tancament de la pinça a la distància programada
- Retard en milisegons perquè la pinça tanqui
- Crida de la funció Homing de l'eix vertical per tornar a la posició superior

En el cas de l'últim estat, s'hi afegeix l'actualització del comptador de dispositius empaquetats i el càlcul del temporitzador de cicle iniciat en el primer estat.

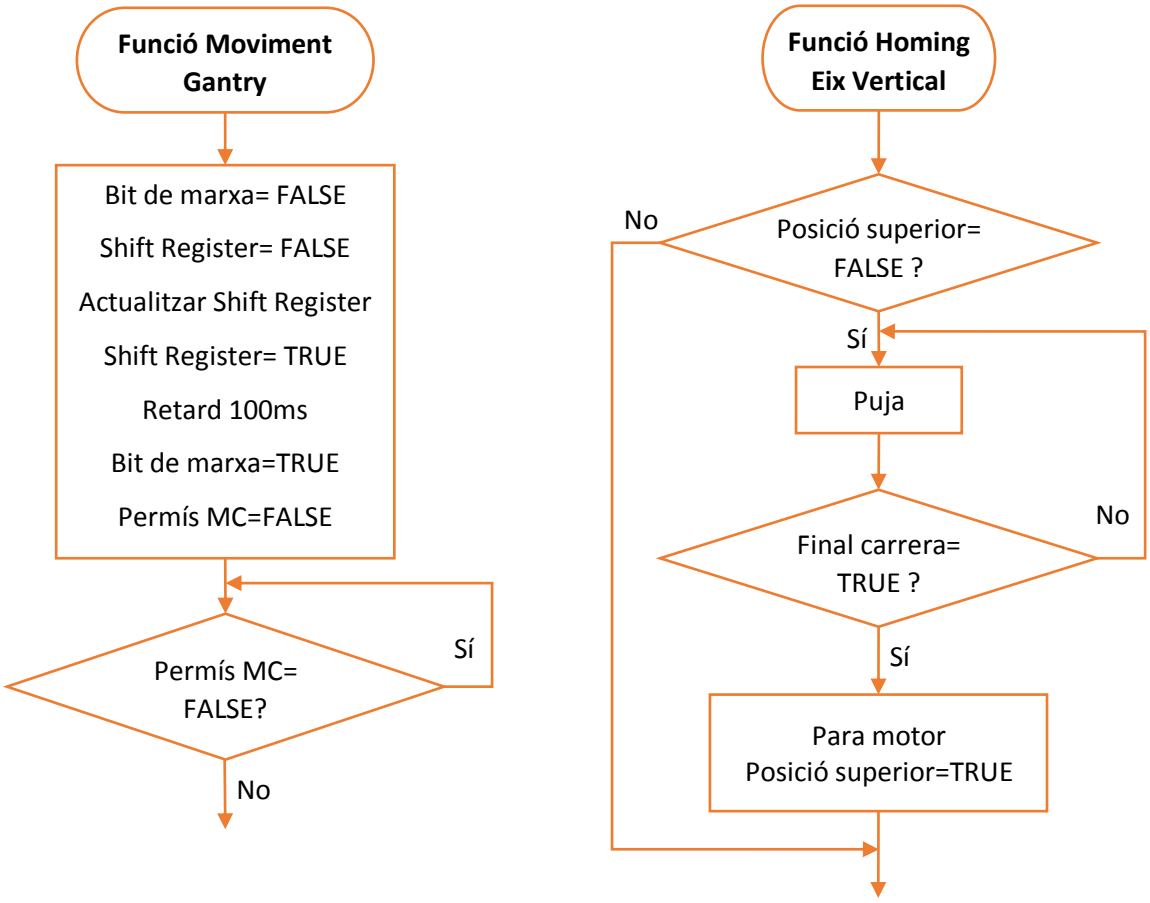
A continuació es mostren els diagrames representatius del les interrupcions utilitzades.



La funció cridada per la interrupció 0, **INT Marxa**, únicament canvia l'estat de la variable de permís de marxa en funció de l'estat anterior. Recordem que aquesta interrupció només té sentit en el polsador secundari integrat en la placa. En el funcionament des de l'aplicació SCADA la variable de permís de marxa canviarà internament a través de les variables OPC.

La funció corresponent a la interrupció 1, **INT MC**, s' executarà quan el controlador rebi un flanc positiu de l'optoacoblador provinent de la senyal Motion Complete de la controladora. En executar-se aquesta interrupció es donarà el permís de MC i es canviarà l'estat segons la posició Actual de Gantry.

A continuació es mostren els diagrames de les funcions representants del moviment del carro del robot i la recollida de l'eix vertical.



Des del punt de vista de l'Arduino, la funció de moviment del Gantry es cuida únicament de treure un nombre codificat en binari mitjançant el registre de desplaçament. Utilitzant un sol registre de desplaçament es poden codificar fins a 8 bits, per tant, un byte. Això significa que existeixen 2^8=256 possibilitats. Per altra banda, com s'ha comentat anteriorment, en utilitzar la interfície de comunicació I/O en la controladora del Gantry la possibilitat de moviments del robot queda reduïda a un total de 32 posicions. De les quals la nostra aplicació només n'utilitza 13. En definitiva, es podria disposar d'un registre de desplaçament de 4 bits. Per no limitar l'aplicació i poder sotmetre's a ampliacions futures s'ha decidit utilitzar el registre de 8 bits inicial.

Per aquest motiu, en el programa s'ha definit una variable **num** del tipus **byte**, que permet guardar les 32 combinacions necessàries en format binari (o no) com es pot veure a continuació.

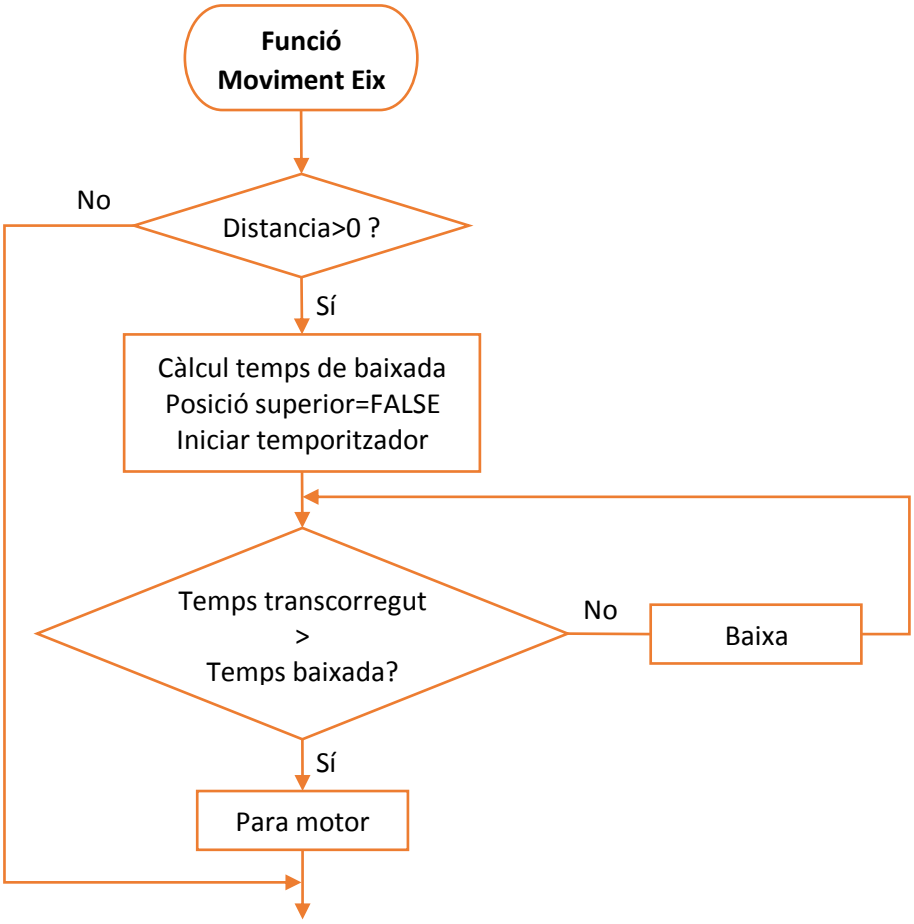
```
byte num[] = {B00000000, B00000001, B00000010, B00000011, B00000100,
B00000101, B00000110, B00000111, B00001000, B00001001, B00001010,
B00001011, B00001100, B00001101, B00001110, B00001111, B00010000,
B00010001, B00010010, B00010011, B00010100, B00010101, B00010110,
B00010111, B00011000, B00011001, B00011010, B00011011, B00011100,
B00011101, B00011110, B00011111};
```

Per transferir aquesta codificació a la controladora del robot es segueixen els passos que es veuen en el diagrama de la funció **Moviment Gantry** i que corresponen als següents.

1- Enviem un 0 al pin de marxa de la controladora del Gantry.
2- Aturem la sortida del registre de desplaçament per poder introduir el nou valor.
3- Transferim en sèrie al registre de desplaçament el nombre codificat en binari.
   `shiftOut (Data, Clock, LSBFIRST, num[posicio]);`
4- Habilitem la sortida del registre de desplaçament per mostrar les dades transferides.
5- Esperem un breu retard perquè la controladora processi la posició a fer el moviment.
6- Enviem un 1 al pin de marxa de la controladora del Gantry.
7- Anul·lem el permís de Motion Complete. El moviment del carro s'està executant.
8- Una vegada iniciat el moviment esperarem fins que la controladora retorni la senyal de MC a través de la **INT MC**.

La funció de **Homing Eix Vertical** és realment la funció de recollida de la pinça. Aquesta executa el moviment de pujada fins que el detector final de carrera dóna un puls indicant que ha arribat a la posició superior. En arribar a dalt, el motor s'atura immediatament per no malmetre els components ja que ha arribat a un límit mecànic.

Per últim es mostra el diagrama de la funció de baixada de l'eix vertical.



La funció **Moviment Eix** determina la distància que ha de baixar la pinça per arribar en cada cas a la posició correcte per recollir el component de l'smartphone. Per aquest motiu, recordant que s'ha utilitzat un servomotor de gir continu, es necessita calcular el temps que el motor ha d'estar girant en funció de la velocitat prèviament indicada. Per tant, mentre el temps transcorregut sigui inferior al temps que s'ha calculat perquè el motor giri aquest seguirà baixant, passat aquest temps el motor s'aturarà.

Aquests càlculs s'han realitzat a paper i s'han introduït en el programa mitjançant una constant amb la qual es calcula el temps d'activació del motor per una velocitat constant del motor.

En primer lloc s'ha observat el comportament del motor observant la velocitat angular en funció de la velocitat entrada per programa, obtenint la gràfica següent.

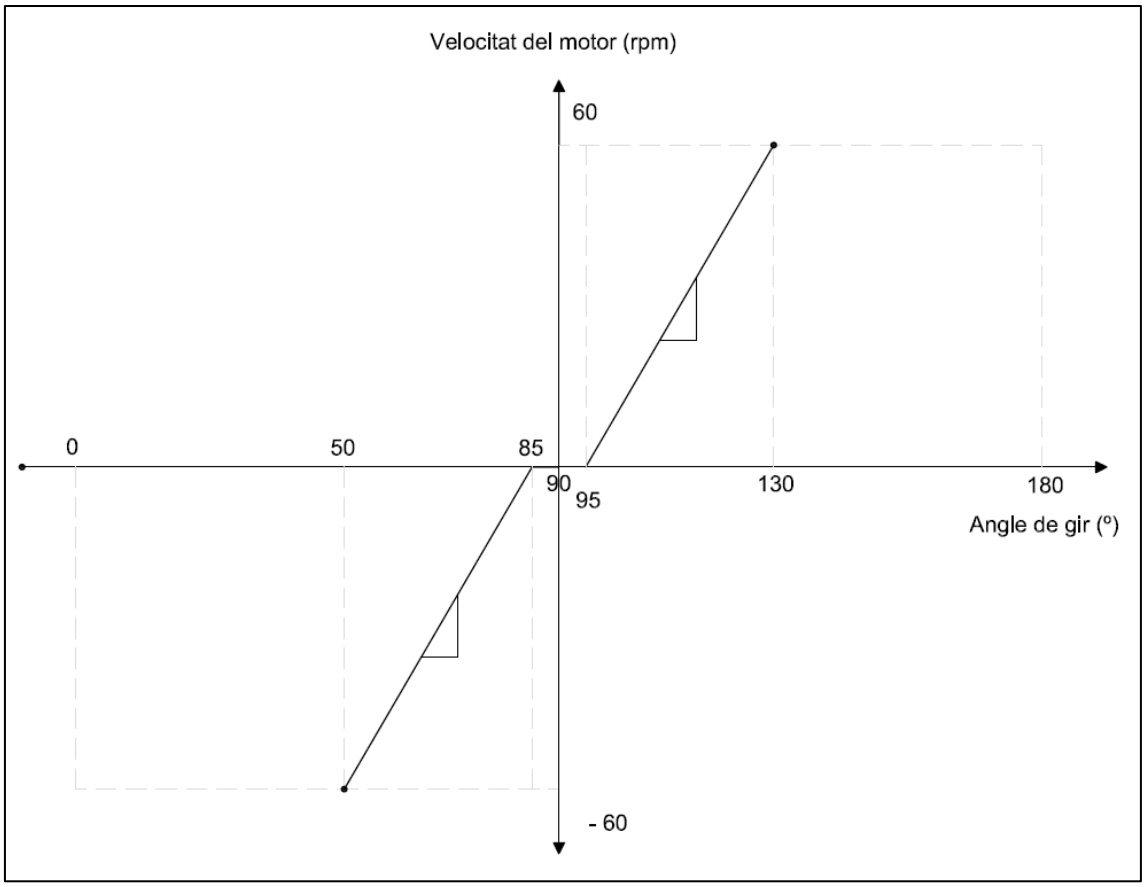


Figura 39: Comportament del servomotor vertical

Com es pot veure en la gràfica, la velocitat del motor queda molt limitada respecte l'angle introduït per programa ja que tant en un sentit com en l'altre l'interval que accepta és de 35º.

Aquest sistema només s'ha utilitzat en la baixada de l'eix ja que la pujada es realitza mitjançant la funció **Homing**. Per tant la velocitat del motor és constant fins arribar al final de carrera.

Per aquest motiu es realitzaran els càlculs en la part entre 85º i 50º de la figura, corresponents al gir negatiu del motor. Amb la gràfica obtinguda es pot observar que el canvi de gir del motor succeeix als 90º amb un offset de 5º pels dos costats.

A partir de la gràfica, sabent que l'equació d'una recta és:

$$y = mx + n$$

On *m* és el pendent de la recta

i *n* és la intercepció amb l'ordenada

S'obté,

$$m = \frac{\Delta y}{\Delta x} = \frac{(-60) - 0}{(50 - 85)} = 1.714 \qquad i \qquad n = 0$$

Si adjudiquem una velocitat constant de 75º per programa, podem extreure la velocitat angular del motor.

$$y = 1.714 * (75 - 85) = \mathbf{-17.14\ rpm}$$

Fent la conversió adient amb el perímetre de l'engranatge obtenim la velocitat lineal de l'eix en funció de l'angle introduït per programa.

$$\frac{17.14\ rev}{1\ min} * \frac{1\ min}{60\ s} = 0.286\ rev/s$$

Sabent que el perímetre de l'engranatge és la distància que es recórrer en una volta del motor obtenim la velocitat lineal.

$$Vel = \frac{0.286\ rev}{1\ s} * \frac{56.548\ mm}{1\ rev} = \mathbf{16.17\ mm/s}$$

Una vegada coneguda la velocitat a la que es mou l'eix es pot calcular el temps que aquest ha d'estar girant per arribar a una posició concreta.

Si

$Velocitat = \frac{Ditsància}{Temps}$ aleshores, $Temps = \frac{Distància}{Velocitat}$

Per tant,

$$t = \frac{d}{16.17}$$

Una vegada fets els càlculs aquest valor teòrics de velocitat s'ha ajustat pel mètode de prova i error. El motiu són les imperfeccions del material i els fregaments de les peces que afecten en el temps d'actuació. Així doncs, el valor finalment utilitzat com a velocitat i al que el sistema respon millor és una constant de valor 20.

La **constant**, s'utilitza en la funció de programa **Moviment Eix** abans explicada i concretament en la instrucció següent:

```
-  TempsBaixada = (distancia / constant) * 1000;
```

El paràmetre **distància** és el valor en mil·límetres que es passa a la funció esmentada. Finalment es multiplica el temps obtingut per passar a milisegons que és la unitat temporal de treball del controlador.

10.1.5.2 L'OPC del controlador Arduino

Com s'ha explicat en l'inici del treball Arduino és una plataforma totalment lliure i oberta a tothom que permet obtenir constantment novetats i millores en tots els sentits des de la comunicació mitjançant Wifi fins al cas de l'OPC.

A diferència de la comunicació per Wifi o Bluetooth, l'OPC d'Arduino no és una opció oferta per la plataforma Arduino sinó que està sent un desenvolupament constant per part de Ildefonso Martinez Marchena. Aquest doctor en informàtica ha desenvolupat unilateralment l'OPC d'Arduino i permet descarregar-lo a través de la seva pàgina web *Software Tools for Makers* {8}. En la mateixa pàgina s'hi inclouen les llibreries necessàries per al funcionament juntament amb alguns petits exemples.

Degut a l'entorn que es troba actualment l'OPC no existeix cap documentació suficientment complerta per justificar i entendre la configuració necessària per al seu funcionament. Per tant, aquesta part del treball ha requerit molt de temps en proves per a poder finalment transmetre i rebre de l'SCADA les dades necessàries.

Pel que fa la instal·lació del servidor OPC cal descomprimir l'arxiu de descarrega i executar l'aplicació ArduinoOPCServer. Una vegada executada apareixen dos fitxers de registre però al mateix temps s'inicia l'OPC. És molt important tancar i registrar l'OPC com a administrador perquè consti la seva existència en el PC i d'aquesta manera el sistema SCADA que actuarà com a client el pugui trobar.

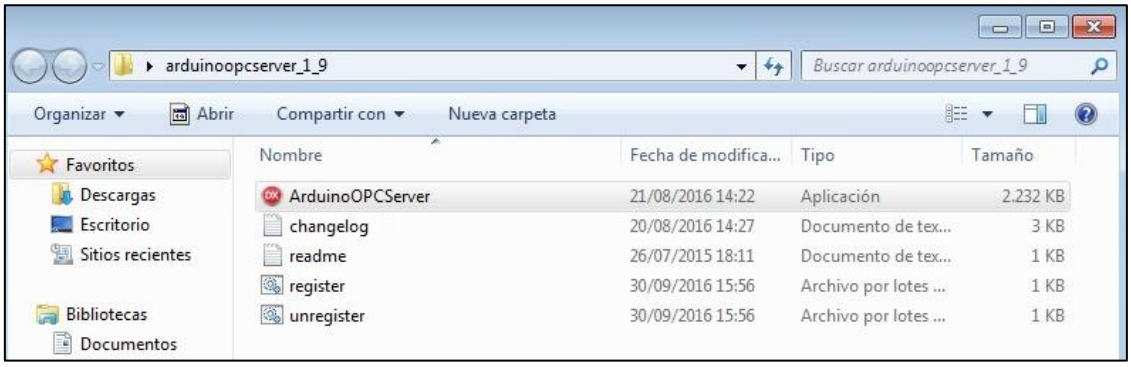


Figura 40: Instal·lació Arduino OPC Server 1.9

Una vegada registrat i tornat a executar l'OPC, a la finestra de configuració afegim un dispositiu sèrie que per defecte surt com a ArduinoSerial0. Recordem que l'Arduino comunica les dades amb l'ordinador mitjançant el mateix cable USB que en la programació, i aquest és un protocol de transmissió de dades en sèrie.

En la configuració cal seleccionar el port USB que utilitza l'Arduino i que es sol detectar de forma automàtica. La velocitat de comunicació s'ha deixat per defecte a 9600bits/s i no hi haurà control de flux. La lectura de dades es realitzarà cada 100ms per optimitzar al màxim la lectura de dades.

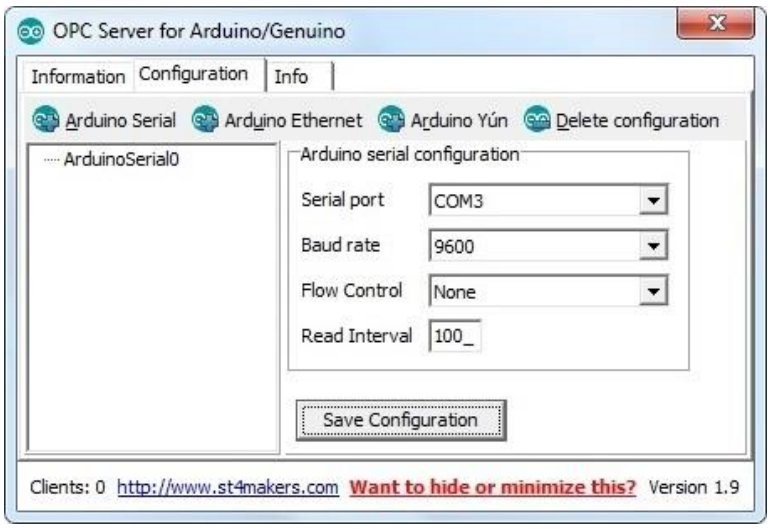


Figura 41: Configuració Arduino OPC Server

Una vegada guardada la configuració i tenint l'OPC en marxa veurem que l'indicador inferior esquerra que senyala la quantitat de clients connectats està a 0. A més, la finestra d'informació que mostra l'estat en què es troba, senyala inicialment que l'OPC està en marxa però esperant la connexió d'algun client.

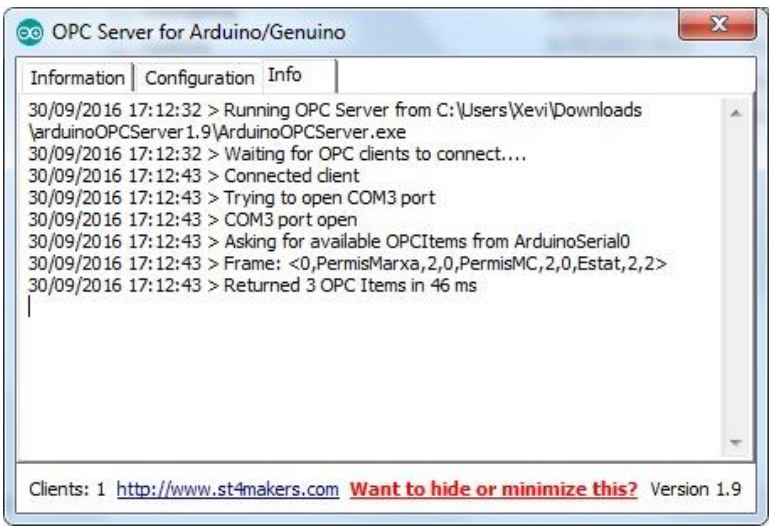


Figura 42: Funcionament Arduino OPC Server

En connectar algun tipus de client, com ara un explorador d'OPC o un sistema SCADA, l'OPC obra el port de comunicació i retorna les variables que en el programa s'ha configurat perquè es comuniquin via OPC.

En l'exemple anterior s'observa com l'OPC ha retornat les variables PermisMarxa, PermisMC i Estat. Els valors que acompanyen aquestes variables no són el valor que contenen sinó la codificació de la seva configuració que s'explicarà més endavant.

Així doncs, passem ara a veure el funcionament de l'OPC internament al programa utilitzat. En primer lloc, igual com s'ha fet amb els servomotors cal incloure la llibreria que interpreta les comandes referents a l'OPC i crear un objecte al que es farà referència en tot moment.

- `#include <OPC.h>` — Afegim la llibreria OPC
- `OPCSerial ObjecteOPC;` — Creem un objecte OPC

En la funció **Setup** on hem inicialitzat les variables i definit les interrupcions caldrà iniciar l'objecte OPC creat anteriorment i també la comunicació pel port sèrie del controlador. Aquesta comunicació sèrie es parametritzarà amb una velocitat estandarditzada de 9600 bits per segon coincidint amb la configuració de l'Arduino OPC Server mostrada a la Figura 41.

```
- ObjecteOPC.setup();
- Serial.begin(9600);
```

A continuació, encara dins la funció **Setup** es seleccionaran les variables que es transmetran a l'OPC. Els noms definits en aquests punt seran els que es veuran en escollir les variables a representar en l'OPC client, és a dir, el sistema SCADA.

La forma de declarar les variables a comunicar és similar a la crida d'una funció amb paràmetres i té la forma següent:

```
ObjecteOPC.addItem("NomItem",opc_Operacio,opc_tipusItem,FuncioRetorn);
```

- ObjecteOPC.addItem: Indica que afegirem un Item (o variable) que estarà referenciat a l'objecte OPC creat en l'inici del programa anomenat ObjecteOPC.
- "NomItem": És el nom que donem a l'Item a transmetre. Per facilitat de comprensió i programació els noms dels Items coincideixen amb les variables utilitzades en el programa.
- opc_Operacio: Indica l'accessibilitat de la variable i pot ser una operació de lectura (opc_read), escriptura (opc_write) o lectura i escriptura a la vegada (opc_readwrite).
- opc_tipusItem: Indica el tipus de variable i pot ser *bool*, *byte*, *int* o *float*.
- FuncioRetorn: Funció on es troba la variable a comunicar.

Un exemple real de la declaració d'una variable OPC del programa seria la següent:

```
ObjecteOPC.addItem("PermisMarxa",opc_readwrite,opc_bool,FuncioOPC1);
```

Com podem veure encara falta definir una funció de retorn on es treballi la variable a transmetre. Degut a que la declaració d'aquestes funcions de retorn també demanen una forma especial en la seva declaració, s'ha decidit crear una funció per a cada variable en comptes d'utilitzar funcions ja existents en el programa.

Així doncs, les funcions de retorn necessàries tenen l'aspecte següent:

```
TipusFuncio NomFuncio(const char *itemID, const opcOperation opcOP,
const TipusVariable value) {
  return VariableAretornar;
}
```

- *itemID: Introdueix el nom de la variable adjudicat en la declaració de la variables OPC.
- opcOperation opcOP: Introdueix el tipus d'accessibilitat donat en la declaració de la variable OPC.
- value: Valor que s'introdueix en la funció. Aquest només es tindrà en compte en les funcions amb opció d'escriptura.

L'estructura representada serà idèntica en aquelles variables amb una accessibilitat de la variable del tipus *opc_read* ja que la seva funció serà únicament de lectura del valor de la variable de programa. A continuació es mostra l'exemple de la variable comptador de packages totals que té únicament accés de lectura.

```
int FuncioOPC3(const char *itemID, const opcOperation opcOP, const int
value) {
  return ContadorPackage;
}
```

En el cas de les funcions que incloguin la opció d'escriptura (*opc_write* o *opc_readwrite*) caldrà captura el valor introduït per l'SCADA a través de l'OPC i modificar el valor de la variable de programa. En cas d'accessibilitat de lectura/escriptura caldrà diferenciar quina acció a realitzar en cada cas. A continuació es mostra l'exemple de la variable de lectura i escriptura de permís de funcionament.

```
bool FuncioOPC1(const char *itemID, const opcOperation opcOP, const
bool value) {
  if (opcOP == opc_opwrite) {
    PermisMarxa = value;
  }
  else
    return PermisMarxa;
}
```

Les variables que s'han transmès del controlador Arduino al sistema SCADA amb aquest mètode es mostren en la taula següent:

Variable	Tipus	Accessibilitat	Funció
PermisMarxa	Bool	Lectura/Escriptura	Permet el funcionament global
PermisMC	Bool	Lectura	Permet el moviment de l'element terminal i el canvi d'estat
ContadorPackage	Int	Lectura	Comptador de dispositius empaquetats
DispositiuActual	Int	Lectura	Número de dispositiu que s'està empaquetant
TempsCicleFloat	Float	Lectura	Temps de l'últim cicle de package
Demanda	Int	Lectura/Escriptura	Demanda de smartphones
DemandaCQ	Int	Escriptura	Demanda de controls de qualitat
Estat	Int	Lectura	Estat de l'1 al 13 en que es troba el packaging actual

Taula 6: Variables a comunicar de l'Arduino

Fins a aquest punt, el software solament ha configurat les variables a transmetre. Perquè aquestes s'actualitzin cal fer un processat continu de les variables a través de la línia de codi següent:

- `ObjecteOPC.processOPCCommands();`

Aquesta instrucció solament variaria si es modifiqués el nom de l'objecte OPC utilitzat. La seva utilitat és executar les comandes OPC i d'aquesta manera actualitzar les variables cada vegada que el programa executa la instrucció. Per aquest motiu aquesta instrucció s'ha situat al final de la funció principal **Loop**.

Com s'ha comentat en l'inici d'aquest apartat, l'OPC d'Arduino està desenvolupat per una sola persona com a repte personal i sense cap benefici a canvi. Això es tradueix en un OPC de funcionalitat senzilla comparada amb altres controladors i que demana un tractament optimitzat del programa on s'inclou.

Una de les limitacions que s'ha trobat en aquest servidor OPC és que necessita que les comandes OPC s'executin molt sovint. En cas contrari la comunicació es perd per intervals de temps i retorna un valor indeterminat de la variable a consultar.

Per aquest motiu ha calgut fer més modificacions addicionals en el programa per complir amb la condició de que la instrucció de comandes s'executés tantes vegades com fos possible.

Per solucionar-ho s'han eliminat tots aquells bucles que capturaven l'atenció del programa i no permetien l'execució de les comandes OPC perdent d'aquesta manera la comunicació. En els casos que no ha sigut possible eliminar el bucle, s'ha inclòs la instrucció dins el bucle.

Un dels casos més assenyalat és el retard en l'obertura i el tancament de la pinça realitzat mitjançant la funció *delay*.

Aquesta funció pausa el programa durant el temps indicat sense permetre cap altre tipus de lectura o execució d'instrucció. A continuació es mostra un retard de 100ms amb la funció *delay*.

```
- delay(100);
```

Per aquest motiu s'ha optat per crear una funció pròpia de retard que permetés l'execució de les comandes OPC. La funció que es mostra simplement captura el temps actual en la variable *temps1*. A continuació comprova si el temps transcorregut es inferior al retard introduït. Mentre aquesta condició és certa es processen les comandes OPC.

```
void temporitzador(long retard) {
  long temps1 = long(millis());
  while (long(millis() - temps1) < retard) {
    ObjecteOPC.processOPCCommands();
  }
}
```

La següent línia de programa mostra una crida a aquesta funció amb un retard de 100 ms. Per tant, serà la instrucció que substituirà al *delay* abans mostrat.

```
- temporitzador(100);
```

Igual que en el *delay* original també es realitza una pausa del programa, però durant la pausa es processen constantment les comandes OPC.

Un procés similar s'ha implementat en el temps d'actuació del motor de l'eix vertical, tant en el cicle de pujada com de baixada.

10.1.6 EL SUPORT I LA GOMA ANTILLISCANT

El suport que s'ha utilitzat per a subjectar i alçar el Gantry s'ha construït amb fusta pel seu baix cost i la seva facilitat de manipulació. Aquest suport permet recolzar l'EXCM-30 de manera que quedi totalment horitzontal i a més permeti un moviment lliure de la pinça en tot l'espai de treball útil del robot.

Degut a que igualment s'havia de dissenyar el tercer eix amb dues parts per poder assolir l'altura necessària, es van dissenyar les dues parts igual de grans per tenir marge d'error en les altures mesurades i el muntatge final. De la mateixa manera que el propi eix, el suport també es va dissenyar una mica més alt per tenir marge per a fer modificacions.

L'excés en els marges tant de l'eix com del suport ha permès pujar la base perquè la superfície de treball quedés més elevada i pròxima al robot. Això ha permès millorar en el funcionament de l'aplicació, reduir el temps de package i eliminar les vibracions que produïa el robot en les zones més baixes.

Com podem veure en la **¡Error! No se encuentra el origen de la referencia.** el suport ha quedat construït a dos nivells. El nivell superior és el principal i forma l'àrea de treball del robot, per tant és la superfície on es col·loquen els diferents elements a empaquetar. Perquè el robot treballés millor a l'hora d'agafar els objectes s'ha afegit una làmina d'un material antilliscant. Aquest és un material rugós que permet una major fricció amb la base i assegura la fixació dels objectes. El mateix material també s'ha enganxat a la part interior dels braços de la pinça per ajudar a la subjecció dels components manipulats.

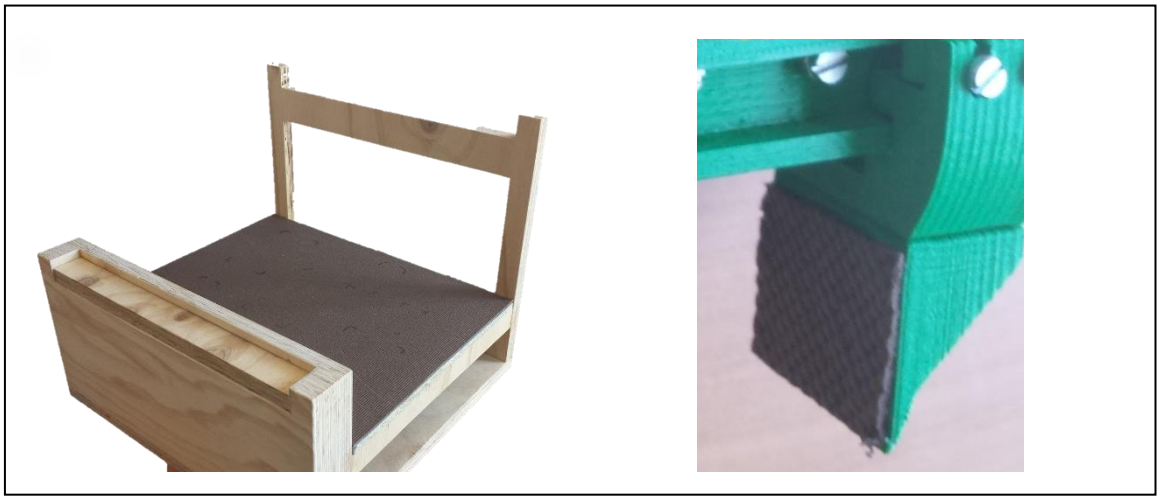


Figura 43: Suport i pinça amb el material antilliscant

En la goma antilliscant del suport s'hi ha marcat la posició dels components a empaquetar per tal de situar-los sempre en una mateixa posició. La situació dels objectes a empaquetar s'ha distribuït de forma radial a la caixa central del dispositiu.

Les marques d'alguns dels elements no fan referència directament al component sinó a uns objectes que es van utilitzar per alçar-los. D'aquesta manera els extrems dels objectes més delicats com les instruccions o la bateria queden a l'aire millorant la subjecció amb la pinça. A continuació es pot veure en detall la distribució dels components en l'espai de treball del robot.

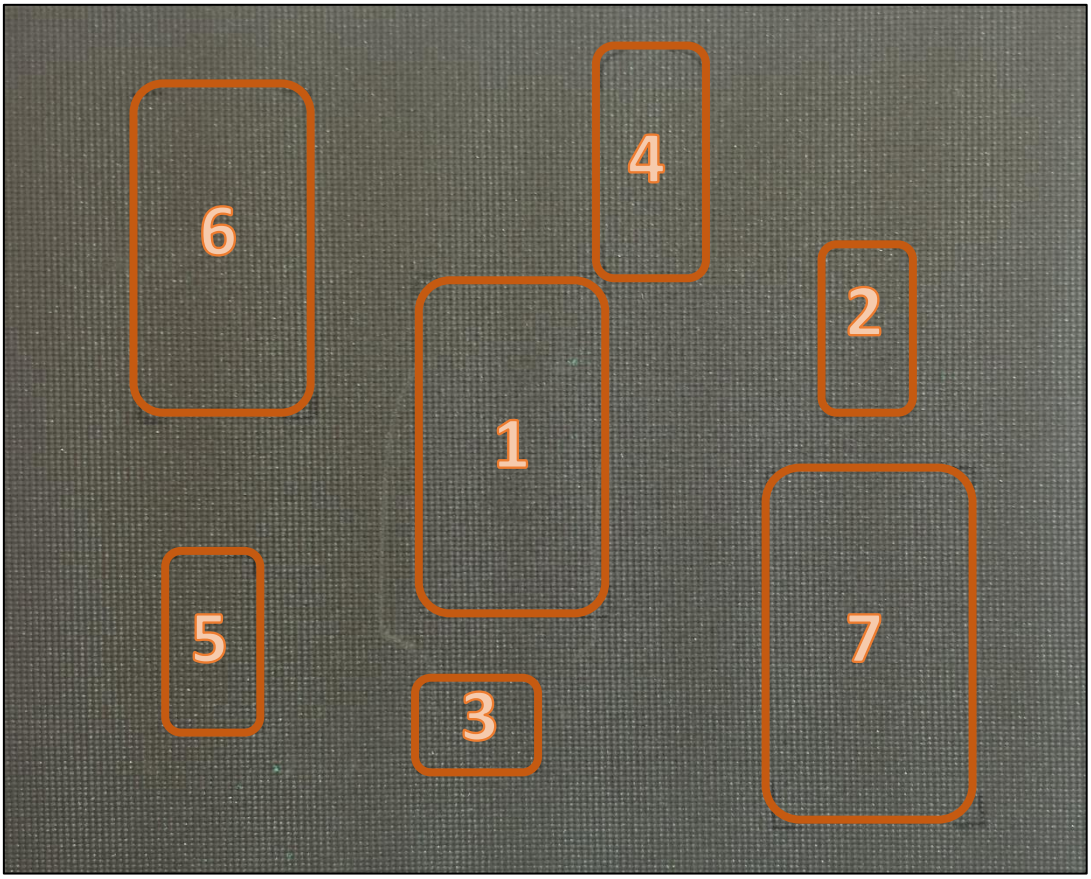


Figura 44: Distribució dels components en la base del suport

La numeració correspon a l'ordre de recollida dels objectes:

1- Caixa central i base del packaging
2- Convertidor
3- Bateria
4- Cable del convertidor
5- Instruccions
6- Dispositiu mòbil
7- Tapa del packaging

10.2 EL IRB140 I EL SISTEMA DE SUPERVISIÓ

10.2.1 APLICACIÓ DE PALETITZAT

Com s'ha anat veient, en aquesta aplicació s'han desenvolupat dos paletitzat de formes diferents, una quadrada i una circular. A continuació veurem detalladament la composició d'aquests dos paletitzats i el volum que representen.

En els dos mosaics en planta següents es pot observar una numeració en cada mòbil representat per les lletres de *SMARTPHONE*. Aquesta numeració defineix l'ordre de paletitzat que seguirà el robot en cadascun dels casos que anirà en sentit contrari a les agulles del rellotge.

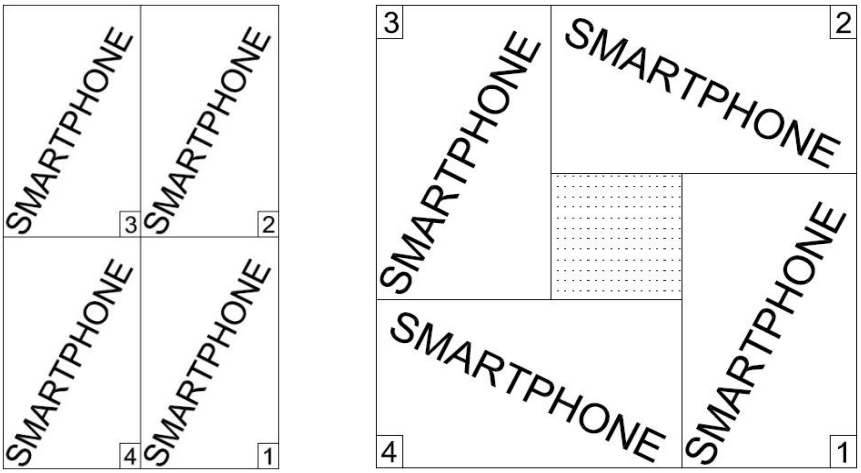


Figura 45: Distribució dels paletitzats quadrat i circular

Per a realitzar els càlculs de volum ens basarem en les mesures reals del dispositiu una vegada fet el packaging que podem veure en la següent imatge. A través d'aquests càlculs veurem que hi ha una certa diferència entre les dues formes de paletitzat.

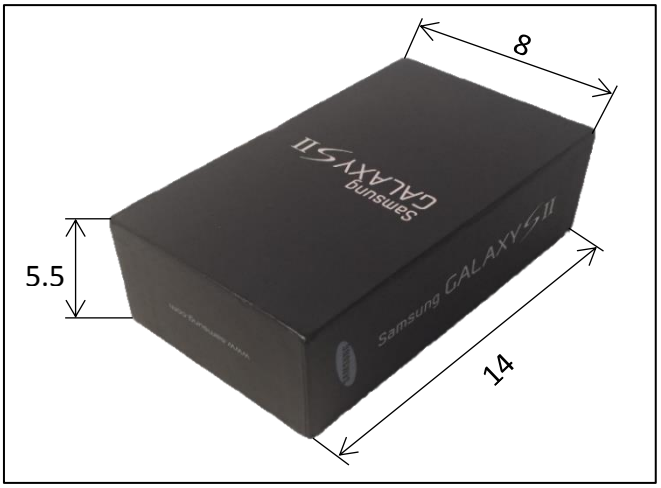


Figura 46: Model Samsung Galaxy SII acotat

Paletització quadrada:

$$Volum\ individual = 14 * 8 * 5.5 = \mathbf{616\ cm^3}$$

$$Volum\ total\ material = Volum\ individual * nº\ mòbils\ per\ pis * nº\ pisos$$

$$Volum\ total\ material = 616\ * 4 * 4 = \mathbf{9856\ cm^3}$$

$$Volum\ total\ palet = Base * alçada$$

$$Volum\ total\ palet = 28 * 16 *\ 5.5 * 4 = \mathbf{9856\ cm^3}$$

Paletització circular:

$$Volum\ individual = 14 * 8 * 5.5 = \mathbf{616\ cm^3}$$

$$Volum\ total\ material = Volum\ individual * nº\ mòbils\ per\ pis * nº\ pisos$$

$$Volum\ total\ material = 616\ * 4 * 4 = \mathbf{9856\ cm^3}$$

$$Volum\ total\ palet = Base * alçada$$

$$Volum\ total\ palet = (14 + 8)^2 * 5.5 * 4 = \mathbf{10648\ cm^3}$$

$$Volum\ no\ utilitzat = Quadrat\ central = 6 * 6 * 5.5 * 4 = \mathbf{792\ cm^3}$$

A continuació podem veure l'estalvi que suposa el paletitzat en forma quadrada davant la circular.

$$Estalvi\ (\%) = 100 - \frac{9856 * 100}{10648} = \mathbf{7.438}\%$$

L'espai no utilitzat en la forma circular es reduiria si les caixes dels mòbils fabricats tinguessin una forma més quadriculada. Aquesta optimització de l'espai útil seria un factor a tenir en compte pel fabricant des del punt de vista del cost en el transport.

En la Figura 47 es representa a través del simulador RobotStudio el robot del laboratori i les posicions on executa els moviments de paletitzat principals. En aquesta imatge es veu a quin costat equival cadascuna de les línies i on són les posicions de **Recollida** i **Home** del robot. Si ens hi fixem en detall, els quatre punts de les circumferències taronges s'observa que són lleugerament diferents. Això es deu a que la línia 1 té un paletitzat habitual en forma quadrada mentre que a la línia 2 el paletitzat habitual serà el circular.

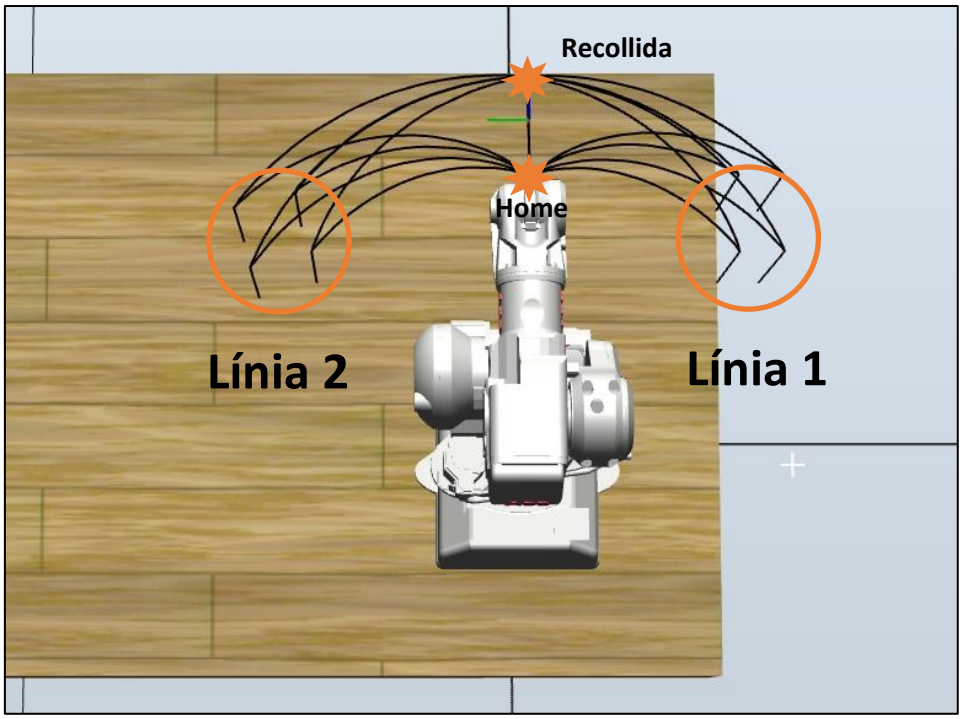


Figura 47: Posicions de Home, recollida i paletitzat del robot

Parlem de paletitzat habitual en els dos casos ja que l'aplicació permetrà, segons hi hagi la possibilitat, desdoblar les línies de paletitzat. Això permet a l'aplicació ser capaç de processar el mateix tipus de paletitzat en ambdues línies.

Les característiques del procés de paletitzat determinen que l'usuari ha de demanar mitjançant l'SCADA, la quantitat de mòbils i posteriorment donar l'ordre per iniciar el paletitzat.

Per entendre millor el procés d'introducció de dades al robot en iniciar l'aplicació, s'adjunta un llistat que mostra ordenadament aquest procés fins al començament de la paletització.

- Demanda de la quantitat de dispositius a produir per cadascuna de les línies.
- Si el nombre de dispositius totals a paletitzar és 0 o negatiu el programa ho indica a l'usuari amb un missatge i es reinicia la demanda.
- Si el nombre de dispositius a paletitzar és 0 en una de les dues línies de paletitzat, el programa demana si la línia amb la demanda es vol desdoblar en les dues línies. En aquest cas, la línia desdoblada rebrà el mateix tipus de paletitzat que la original.
- Si el nombre de dispositius és diferent en cada línia es farà la producció de les dues línies paral·lelament, cadascuna amb el seu tipus de paletitzat habitual.
- Si les dades anteriors són correctes l'aplicació espera un permís per iniciar el paletitzat.

Una vegada començat el procés de paletitzat segons la quantitat de mòbils demanats a cada línia el robot es comportarà d'una manera o d'una altra.

10.2.2 PROGRAMACIÓ AMB RAPID

Tots els robots de la casa ABB es caracteritzen per utilitzar el llenguatge de programació creat en el seu dia per la pròpia marca anomenat RAPID (*Robotics Application Programming Interactive Dialogue*). És un llenguatge de programació d'alt nivell amb estructures similars a altres llenguatges més coneguts com podria ser el C/C++ però pensat exclusivament per al moviment de robots ABB.

En el llenguatge RAPID es poden diferenciar 3 formats de variables com poden ser constants, variables o persistents. Les **constants** són les que mantenen el mateix valor en tot el programa des de la declaració i inicialització de la variable. Les **variables** poden modificar el valor tantes vegades com es vulgui durant el programa independentment del valor d'inicialització. Per últim, les variables **persistents** són una mescla entre les variables ja que permeten modificar el seu valor però una vegada es finalitza el programa l'últim valor queda guardat com a valor d'inicialització pel següent inici de programa.

Les variables en format persistent són les variables que s'han hagut d'utilitzar per poder fer la comunicació via OPC ja que són les úniques que ho permeten. Les variables que no s'han volgut mostrar o interactuar amb l'aplicació SCADA o que s'utilitzen per fer càlculs interns en el programa s'han deixat en el format habitual de constant o variable.

Per altra banda, la distribució del programa de l'aplicació s'ha dividit en 4 mòduls que contenen diferents rutines. Aquestes rutines poden ser de 3 tipus segons siguin **funcions**, **procediments** o **tractament d'interrupcions**. Les funcions executen una part de programa que retorna un tipus de dada concreta que dependrà d'uns valors d'entrada. Els procediments serien la forma senzilla de les funcions ja que permeten executar una part de programa sense retornar cap valor ni depenent de cap valor d'entrada. Per últim, un tractament d'interrupció és la part de programa que només s'executa en activar-se la senyal corresponent a aquella interrupció.

A diferència de les interrupcions, les funcions i els procediments, poden ser cridats des de qualsevol part del programa i també des d'una altra rutina.

Per veure el funcionament i la distribució del programa de paletitzat s'explicaran les característiques principals dels diferents mòduls en què s'ha distribuït el programa per ajudar a la seva organització.

- **Mòdul de variables:**

En aquest primer mòdul únicament es declaren els tipus de variables i s'inicialitzen amb els valors o paràmetres desitjats. La declaració i inicialització de variables només es realitza una vegada en l'inici de programa.

Com que el programa està pensat per funcionar de forma cíclica s'ha creat una funció per tal de que quan aquesta sigui cridada les variables siguin inicialitzades altre vegada.

- **Mòdul principal:**

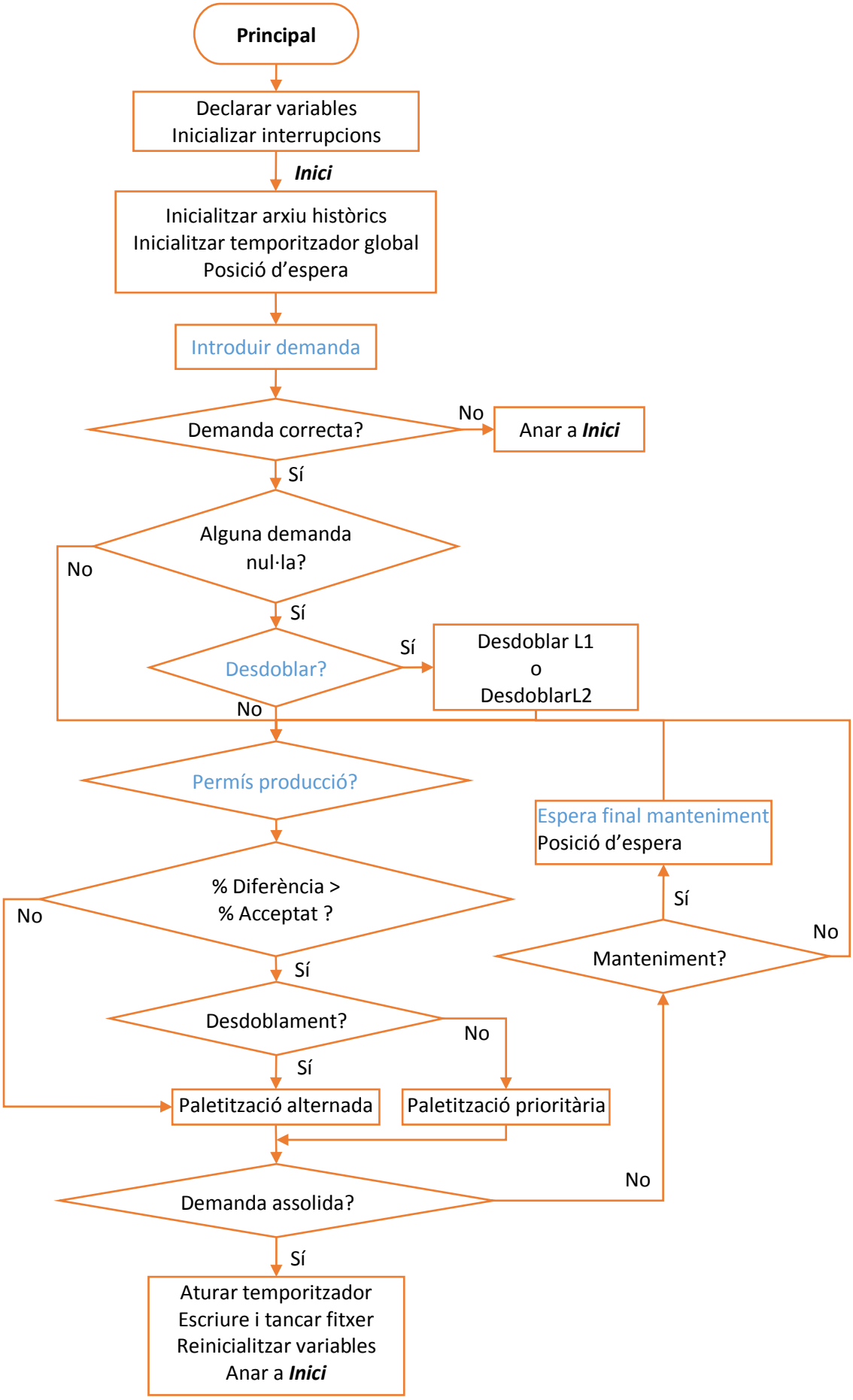
El mòdul principal inclou la funció del programa principal encarregat de la gestió dels moviments del robot amb el suport de la resta de mòduls. Inicialment, en aquesta funció principal es defineixen les 4 interrupcions que permetrà l'aplicació, la direcció on es crearà l'arxiu per a guardar els històrics i la inicialització de la capçalera del mateix. Finalment, trobem un bucle infinit que serà la base del nostre programa.

En l'inici d'aquest bucle s'espera que l'usuari introdueixi les dades de paletitzat. Introduïda la demanda i en cas que correspongui, també es demanarà si es vol fer un desdoblat de la línia de paletitzat o si les dades introduïdes no són vàlides. A continuació el robot es posicionarà a la posició de Home, en cas que no ho estigui, esperant el permís de producció del robot.

A partir d'aquí comença la part principal del programa on inicialment es comprova si s'ha demanat el desdoblament d'alguna de les línies. En cas afirmatiu es reparteix la demanda entre les dues línies i també es canvia el tipus de paletitzat habitual de quadrat a circular o viceversa de la línia a la que s'envien la meitat de les peces. Per altra banda, s'anul·la l'equilibri entre línies ja que en aquest cas s'ha repartit de forma igual i es produirà alternativament un paletitzat per cada línia.

En cas de que no existeixi un desdoblament de les línies el programa marcarà un límit en el procés de producció de la línia que tingui més demanda. Aquest límit ve determinat pel valor del percentatge de línies acceptat. En el procediment d'actualitzar variables que veurem més endavant existeix un càlcul que compara les peces restants de les dues línies i treu un percentatge de la diferència de les restants totals.

Quant el percentatge que marca la diferència és superior al percentatge acceptat es paletitza prioritzant la línia de més demanda. Per altra banda, mentre la diferència entre paletitzats restants sigui inferior al percentatge límit o s'hagi demanat un desdoblament, la paletització es farà alternant les dues línies.


Principal
Declarar variables
Inicializar interrupcions
Inici
Inicialitzar arxiu històrics
Inicialitzar temporitzador global
Posició d'espera
Introduir demanda
Demanda correcta?
No
Anar a Inici
Sí
Alguna demanda nul·la?
No
Sí
Desdoblar?
Sí
Desdoblar L1 o DesdoblarL2
No
Permís producció?
Espera final manteniment
Posició d'espera
% Diferència > % Acceptat ?
No
Sí
Manteniment?
Sí
No
Desdoblament?
No
Sí
Paletització alternada
Paletització prioritària
Demanda assolida?
No
Sí
Aturar temporitzador
Escriure i tancar fitxer
Reinicialitzar variables
Anar a Inici

En el diagrama de flux anterior es poden observar les funcions principals del mòdul principal. A més, també s'ha inclòs simbòlicament el mòdul de variables en la inicialització i reinicialització de variables. El mateix diagrama mostra en blau les pauses que rep el programa fins a rebre una resposta de l'usuari a traves de l'aplicació SCADA.

En la paletització alternada es realitza un paletitzat continu per a cada línia mentre no s'hagi assolit la demanda. En canvi, la paletització prioritària paletitza la línia de més demanda fins que aquesta s'equilibri a la de menys demanda, un cop assolida la igualtat es procedeix a la paletització alternada.

Per últim, al final del mòdul principal, una vegada tancat el bucle es programen les interrupcions.

1- Interrupció de Permís de marxa

   Activació d'un bit que permet el funcionament o la pausa del programa i el moviment del robot.

2- Interrupció de control de qualitat de les dues línies

   Activa una petició de control de qualitat que es comprovarà en el mòdul de paletitzat.

3- Interrupció de control de qualitat del procés de packaging del Gantry

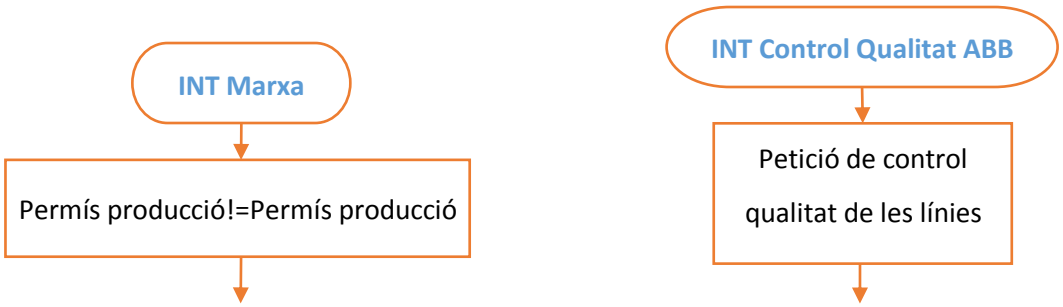
   Activa una petició de control de qualitat que es comprovarà en el mòdul de paletitzat.

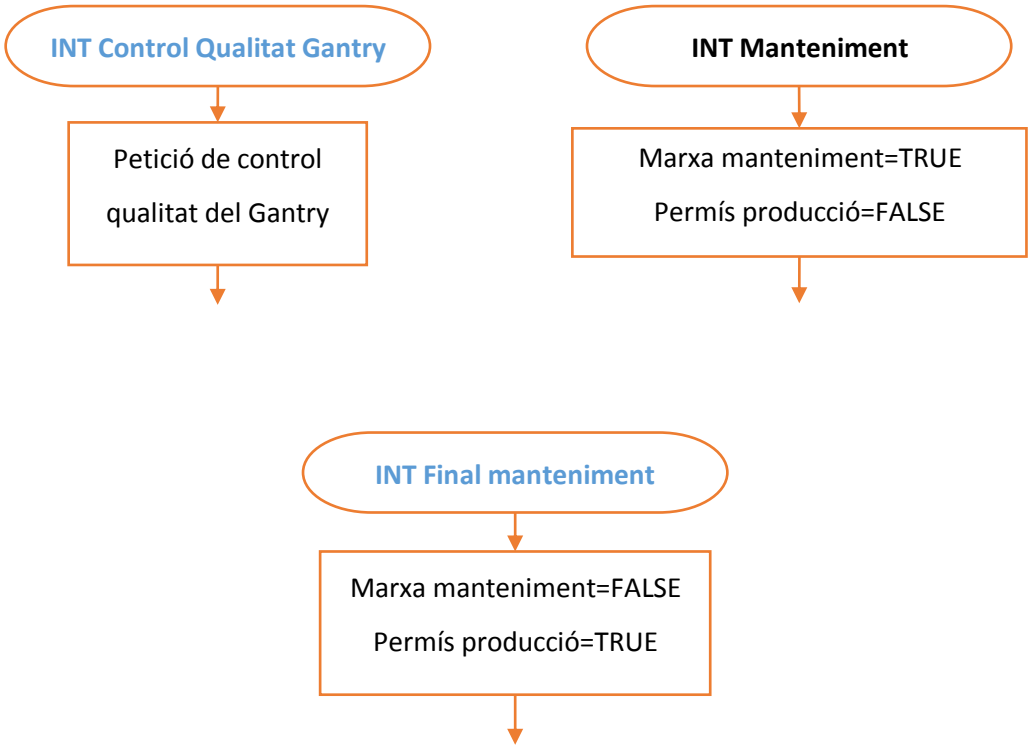
4- Interrupció de manteniment

   Interrupció temporal del robot que s'executa cada cert temps i atura el paletitzat deixant el robot en la posició d'espera.

5- Interrupció de final de manteniment

   La pausa del moviment del robot causada per la interrupció anterior no és arrancada fins que aquesta interrupció s'executa.





Igual que en el diagrama general del programa principal, les interrupcions també mostren en blau les crides realitzades des de la pantalla del sistema de supervisió.

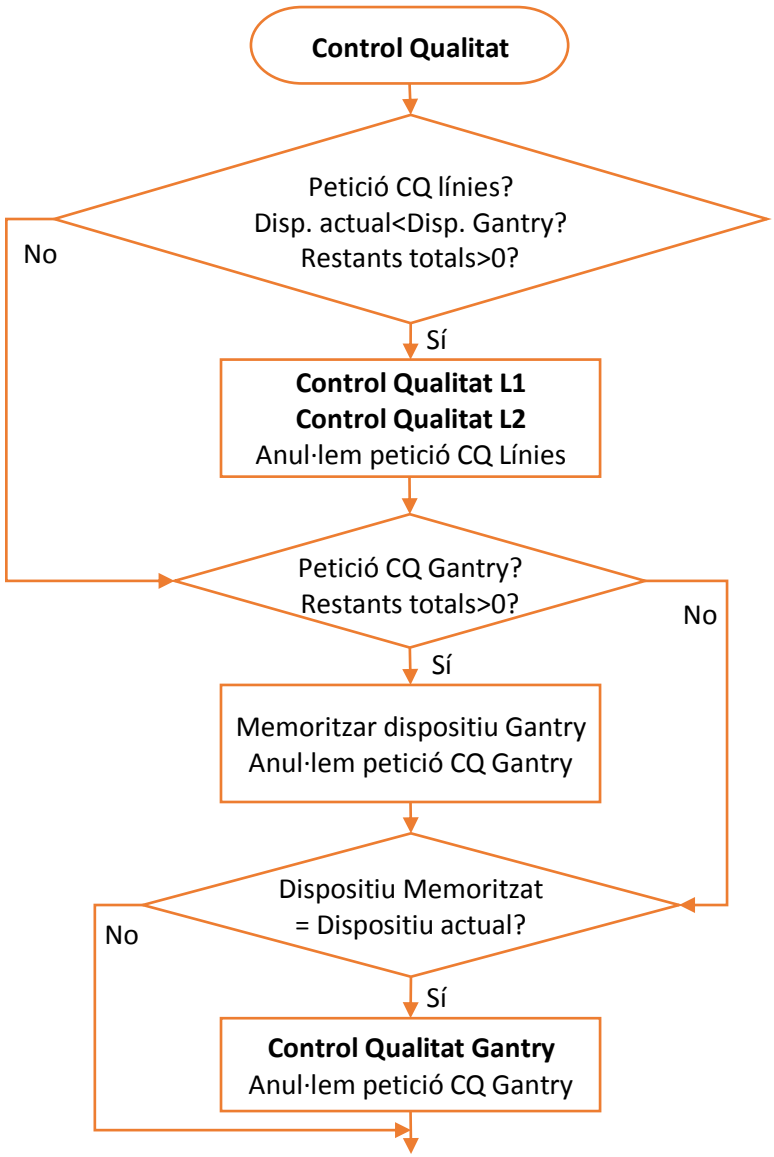
Mòdul de paletitzat:

Aquest mòdul executa els moviments del robot a partir de les crides que es fan des del programa principal i es divideix en diferents funcions o procediments. A continuació s'explicarà el funcionament de cada funció i es representarà en forma de diagrama de flux. En el mateix diagrama es podrà comprovar que també hi haurà crides entre les diferents funcions utilitzades.

1- Control de qualitat

Aquest procediment s'executa en cada nova paletització i comprova si hi ha alguna petició de control de qualitat tant de les línies com de l'aplicació del Gantry.

En cas que hi hagi una demanda en les línies, s'executa un control de qualitat per cada línia. Si la petició de control de qualitat és en el Gantry, el robot memoritza el dispositiu actual del packaging. Quan aquest smartphone arribi a la posició de recollida, s'executarà el moviment de control de qualitat per la línia 1.



2- Control de qualitat en la línia 1

Procediment que executa el moviment corresponent en cas que el procediment 1 hagi detectat una petició de control de qualitat per la línia 1.

3- Control de qualitat en la línia 2

Procediment que executa el moviment corresponent en cas que el procediment 1 hagi detectat una petició de control de qualitat per la línia 2.

4- Control de qualitat del Gantry

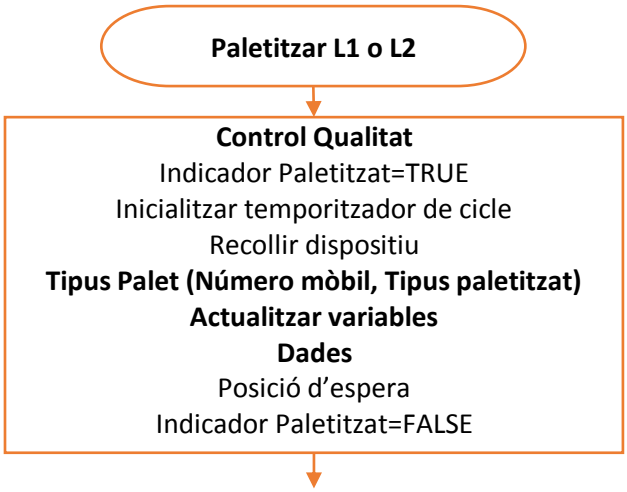
Procediment que executa el moviment corresponent en cas que el procediment 1 hagi detectat una petició de control de qualitat en el sistema de packaging.

5- Paletitzar L1

Procediment que executa el paletitzat per la línia 1 i és cridat des del programa principal.

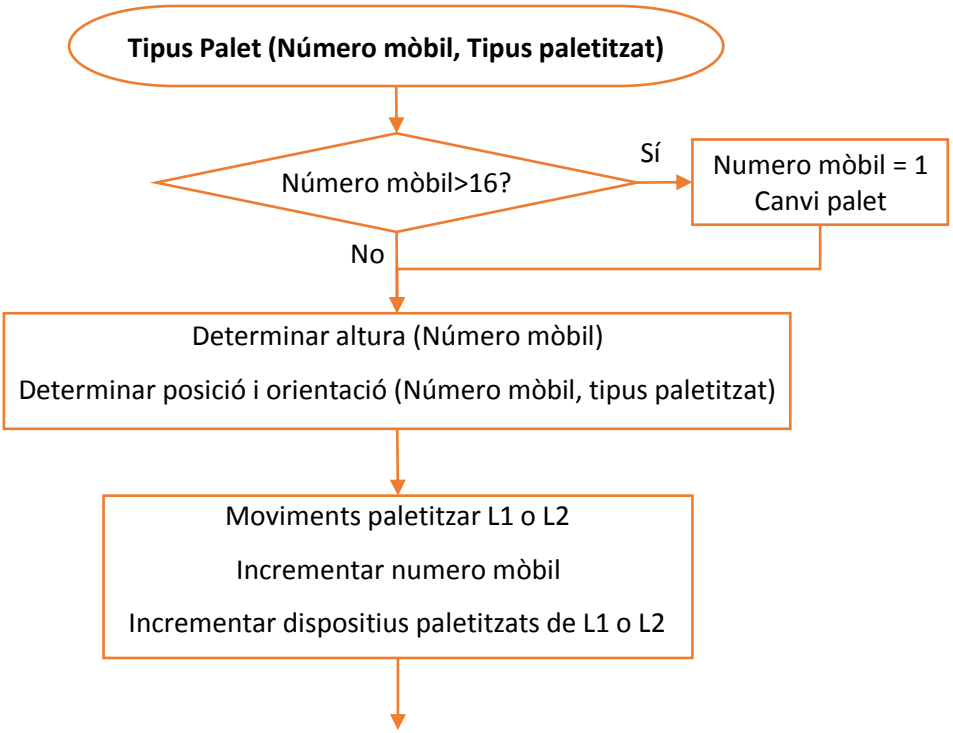
6- Paletitzar L2

Procediment que executa el paletitzat per la línia 2 i és cridat des del programa principal.



7- Tipus palet

Funció cridada per les funcions 5 i 6 i el seu propòsit és calcular i posicionar el robot per a cada paletitzat en funció del tipus (quadrat o circular) i el numero de smartphone actual.

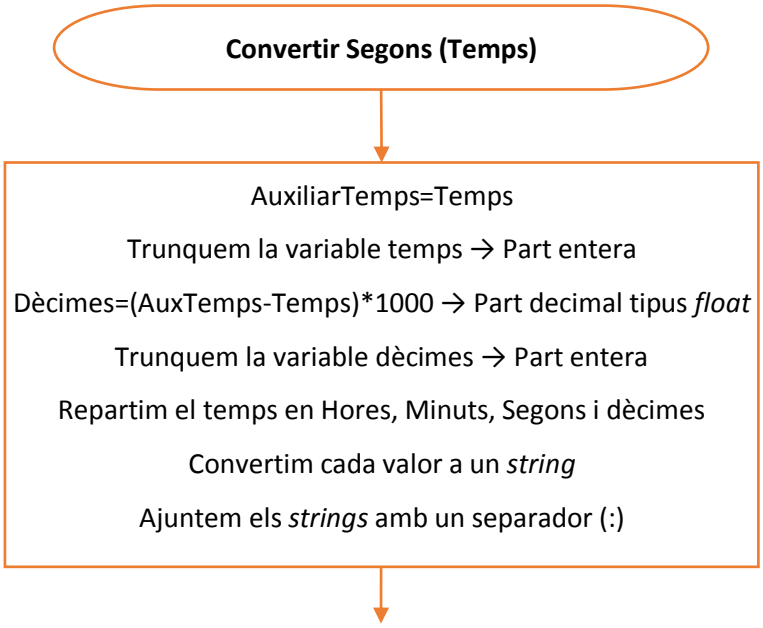


- **Mòdul de dades:**

La intenció d'aquest mòdul és fer un tractament de les dades i no executa cap moviment del robot. Dins aquest mòdul també podem trobar diferents rutines.

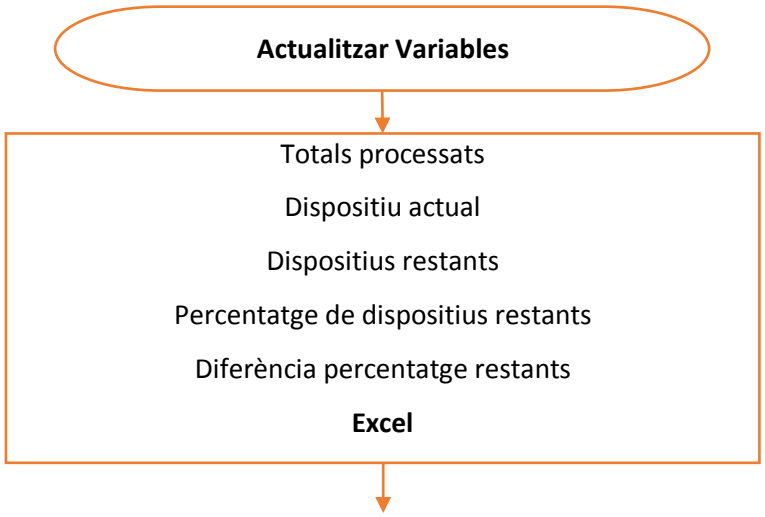
1- Convertir segons

Durant el funcionament del programa s'activen i desactiven diferents temporitzadors per calcular temps de cicles de paletitzat o bé el temps de funcionament total de l'aplicació. El resultats d'aquestes temporitzacions ve donat amb un format del tipus 123.45 segons. Aquesta representació dificulta la lectura quan els temps de processat són majors d'un minut. Per aquest motiu s'ha optat per crear aquesta funció que agafa aquest valor numèric, el separa i l'ajusta a un format h:m:s:ds. Cadascun d'aquests valors es calcula independentment i a continuació es fa una conversió d'un format numèric a un format escrit (*string*) ajuntant tots els valors amb un separador. Finalment es retorna aquesta cadena de caràcters.



2- Actualitzar variables

Procediment que actualitza totes les variables després de cada moviment de paletitzat del robot calculant els paletitzats restants, el total de processats i el percentatge de paletitzats restants que determinarà l'equilibri entre les dues línies.



3- Dades

Aquest procediment mostra un recull de les dades més importants calculades anteriorment per la TeachPendant.

4- Excel

És l'últim procediment del programa i escriu en l'arxiu Excel el conjunt de dades que s'ha cregut més important. Per tant, en cada moviment del robot s'escriu a l'arxiu les següents dades que corresponen a la capçalera inicialitzada en el mòdul principal:

"Hora;Numero de peça;Procés;Tipus paletitzat;Temps de procés;"

Com podem veure en la capçalera, tots els valors escrits en l'arxiu Excel van acompanyats d'un punt i coma. Aquest és el caràcter CSV que separa totes les variables per ajustar-se a la columna corresponent.

10.2.2.1 Arxiu de dades extern

L'arxiu de dades extern permet a l'usuari fer un seguiment del procés realitzat pel robot des de que ha iniciat la producció fins que l'ha finalitzat.

Degut a que l'aplicació està pensada per a un funcionament continu i per millorar la cerca d'un dispositiu concret, es varia automàticament el nom de l'arxiu segons el dia i hora de l'inici de la paletització. Una limitació d'aquesta numeració és que el robot no llegeix el dia en el format habitual dd/mm/aaaa sinó que utilitza el dia de la setmana numerat de 1 a 7 per indicar el dia de dilluns a diumenge. Aquest canvi del nom de l'arxiu es produirà cada nou cicle de paletitzat.

Com s'ha pogut comprovar en l'apartat anterior el caràcter CSV és el que permet la diferenciació de columnes en l'arxiu Excel. Tot i això, cal indicar al programa que les dades introduïdes segueixen aquesta estructura. En cas contrari, les dades queden mostrades de la següent manera.

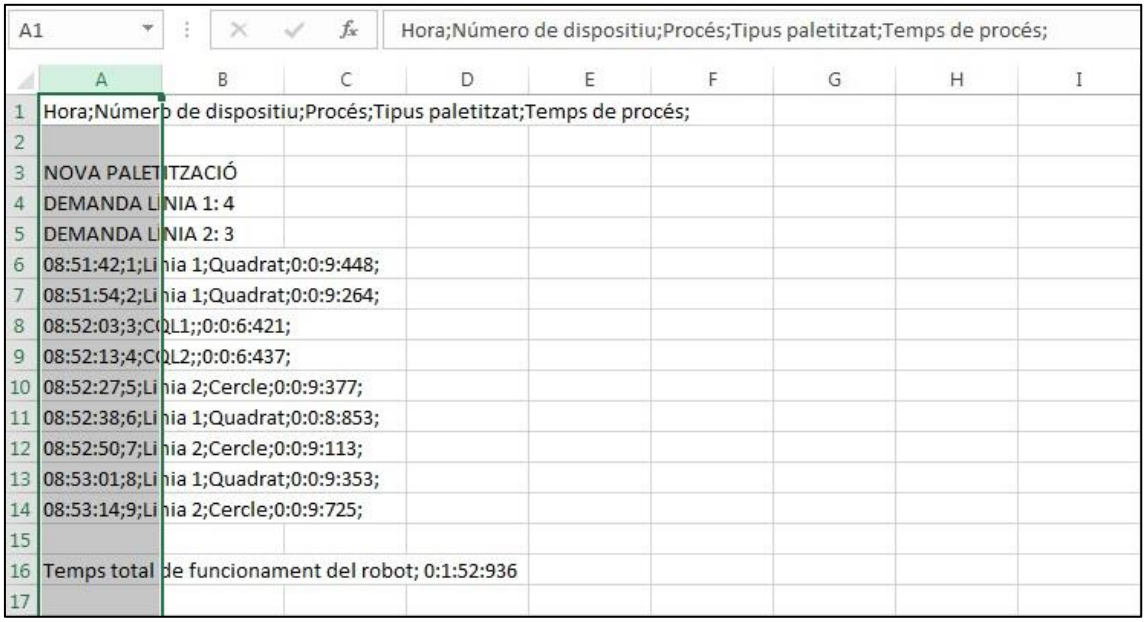


Figura 48: Excel-Exemple d'un arxiu d'històrics en format CSV

Per fer la traducció de dades simplement cal seleccionar la primera columna que és on es troben totes les dades en un principi. Una vegada seleccionada fem clic a la pestanya *Datos* del programa on apareixerà un desplegable del qual seleccionarem la icona *Texto en columnas*.

Figura 49: Excel-Conversió de text a columnes 1

Apareixerà una pestanya com la següent on seleccionarem la primer opció.

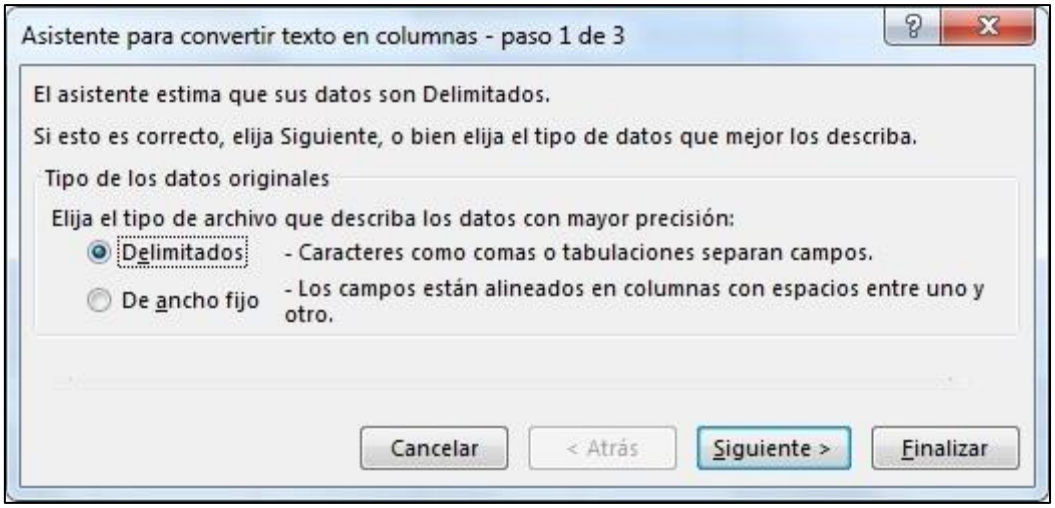


Figura 50: Excel-Conversió de text a columnes 2

En la finestra següent marcarem l'opció *Punto y coma* ja que és el caràcter utilitzat com a separador de dades.

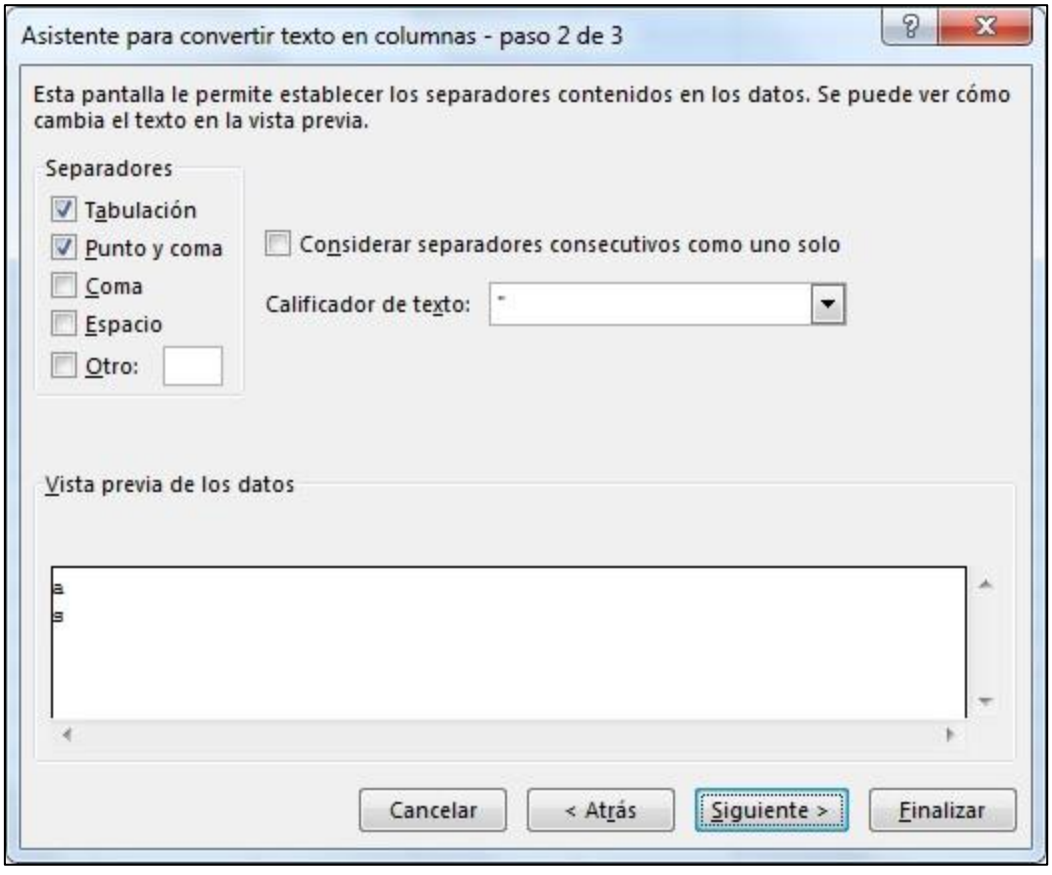


Figura 51: Excel- Conversió de text a columnes 3

En prémer següent i finalitzar les dades es separen automàticament en columnes permetent així una lectura molt més lleugera. A continuació podem veure un exemple dels arxius creats.

	A	B	C	D	E
1	Hora	Número de dispositiu	Procés	Tipus paletitzat	Temps de procés
2					
3	NOVA PALETITZACIÓ				
4	DEMANDA LÍNIA 1: 4				
5	DEMANDA LÍNIA 2: 3				
6	8:51:42	1	Linia 1	Quadrat	0:0:9:448
7	8:51:54	2	Linia 1	Quadrat	0:0:9:264
8	8:52:03	3	CQL1		0:0:6:421
9	8:52:13	4	CQL2		0:0:6:437
10	8:52:27	5	Linia 2	Cercle	0:0:9:377
11	8:52:38	6	Linia 1	Quadrat	0:0:8:853
12	8:52:50	7	Linia 2	Cercle	0:0:9:113
13	8:53:01	8	Linia 1	Quadrat	0:0:9:353
14	8:53:14	9	Linia 2	Cercle	0:0:9:725
15					
16	Temps total de funcionament del robot	0:1:52:936			
17					

Figura 52: Excel-Exemple d'un arxiu d'històrics

Segons la imatge de l'arxiu Excel es pot observar en la primera fila la capçalera que indica el tipus de dada de cada columna.

A continuació s'indica un nou procés de paletitzat i la demanda que hi ha per a cada línia. Tot seguit es mostren les dades corresponents a cada moviment realitzat. Si ens fixem en l'últim número de dispositiu paletitzat veurem que no concorda per 2 amb la demanda. Això es deu a la petició d'un control de qualitat de les línies que correspon als dispositius 3 i 4.

Finalment es pot observar el temps total de funcionament del robot des de que s'ha iniciat la producció amb un format de hores, minuts, segons i centèsimes de segon.

En iniciar una nova paletització es generaria un nou arxiu del mateix estil que l'exemple explicat amb les dades corresponents a una nova demanda.

10.2.2.2 Utilització de l'eina

Per fer una simulació completa del que seria el procés real s'ha inclòs en el programa una eina de treball com la doble ventosa que inclou el robot del laboratori. Aquesta ventosa permetria subjectar els smartphones empaquetats per a fer el procés de paletitzat.

Els moviment del robot van determinats al que s'anomena TCP (*Tool Center Point*) que seria el punt d'aplicació de l'eina utilitzada. Això significa que en donar una ordre de moviment a una posició XYZ el que fa el robot és moure el TCP a aquesta posició. Si no es defineix cap eina o *Tool* el robot agafa com a TCP l'eix 6 del robot que és la part més externa del braç.

En adjuntar una eina com la que podem veure en la Figura 53 fa que es modifiqui el TCP del robot i a partir d'aquest moment tots els moviments es faran referenciats a partir d'aquest nou TCP. En aquest cas concret, el centre de l'eina es troba a 7.5cm del sisè eix en línia recta i, al ser una eina simètrica, correspon al centre de les dues ventoses.

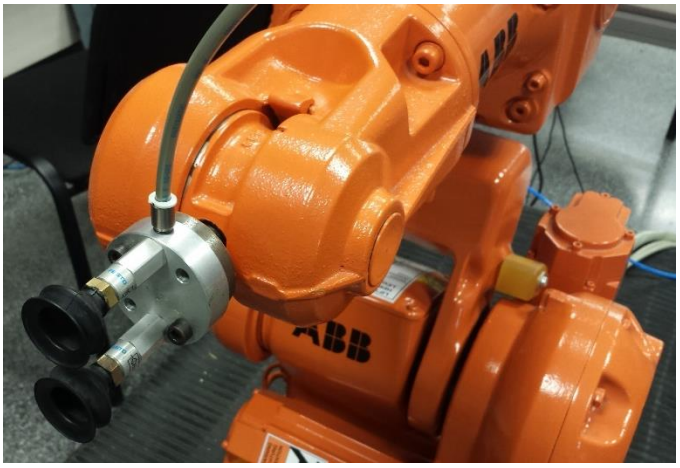


Figura 53: Eina utilitzada i la seva orientació

L'estat habitual de la ventosa és en posició horitzontal cosa que suposava un problema en l'aplicació ja que s'havia suposat que els mòbils arribarien en posició vertical. Si s'hagués deixat l'eina en horitzontal i els smartphones arribessin en vertical la distància entre extrems de la ventosa quedaria molt justa a les dimensions de la capça del robot i no es podria realitzar una subjecció segura. Per aquest motiu s'ha girat 90º la ventosa per tal de que quedi de forma vertical com es pot veure en la imatge anterior i permeti subjectar correctament les caixes dels dispositius.

Per no haver de definir la rotació de l'eina en cada posició s'ha configurat la rotació en la definició de la pròpia eina de treball. A continuació podem veure la línia de programa que defineix i configura aquesta eina que anomenarem Ventosa.

PERS tooldata Ventosa:=[TRUE,[[0,0,75],[0.707,0,0,-0.707]],[0.2,[0,0,50],[1,0,0,0], 0,0,0]];

- TRUE: Significa que el robot està subjectant l'eina.
- [[0,0,75],[0.707,0,0,-0.707]]: El primer paràmetre són les coordenades del TCP (75 mm en l'eix Z) i el segon indica la rotació de l'eina utilitzant uns paràmetres anomenats quaternions.
- [0.2[0,0,50],[1,0,0,0], 0,0,0]: Indiquen respectivament el pes de la *tool* en kg, la posició en XYZ del centre de gravetat, els eixos del moment, i el moment. Aquests dos últims paràmetres s'ha deixat per defecte ja que s'ha considerat com una carga puntual.

Definició de quaternió:

L'orientació d'un sistema de coordenades pot ser descrita utilitzant una matriu de rotació que defineix la direcció dels eixos de coordenades d'aquest sistema basant-se en un sistema de referència.

Els quaternions tenen la mateixa utilitat que la matriu de rotació, de fet, es calculen a partir d'aquesta però es representen d'una forma més compacta ja que permet definir una orientació amb 4 valors.

Degut a que s'han utilitzat aquests elements per orientar l'eina, a continuació es mostren els càlculs necessaris per girar l'eina un angle de 90º perquè quedi de forma vertical.

Quant es vol fer una modificació de l'orientació s'ha de tenir molt clar quins eixos de coordenades s'utilitzen i com estan configurats originàriament.

Tot i que en el simulador no es pot mostrar l'eina físicament, en la primera imatge de la Figura 54 es mostra el sistema de coordenades original que es troba a 75mm del sisè eix. En la part inferior esquerra es poden veure en petit uns eixos de coordenades diferents als de l'eina. Aquests eixos són fixes i són els eixos de coordenades món que es troben 30cm sota la base del robot ja que s'ha alçat per representar el muntatge de la plataforma del laboratori.

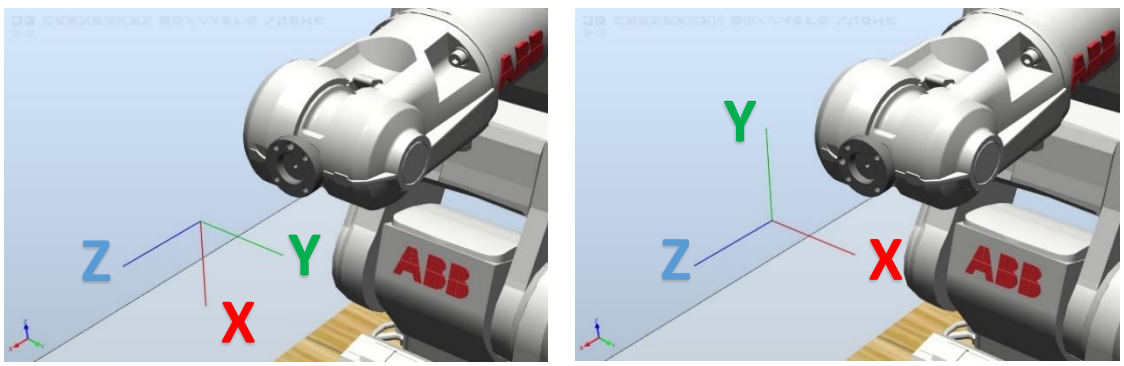


Figura 54: Modificació en l'orientació del TCP

Per l'altra banda, en la imatge de la dreta es mostra la orientació que es vol aconseguir on podem veure que s'ha fet una rotació de 90º sobre l'eix Z.

Una vegada coneguts els eixos actuals i la posició que es vol aconseguir s'ha de fer una projecció dels nous eixos sobre els eixos de referència per extreure la matriu de rotació abans comentada. Per fer-ho cal desglossar cada nova component en els eixos del sistema de referència original. En fer aquesta operació s'obté cada eix nou com a un vector de tres components en el sistema de referència antic. Dit d'una altra manera, si sobreposem el sistema de coordenades rotatiu de la Figura 55 en el sistema de coordenades de referència, l'eix X es podrà descomposar en tres vectors (X_1, X_2, X_3) sobre els eixos XYZ del sistema de referència. El mateix passarà amb l'eix Y i Z que obtindrien uns vectors Y_1, Y_2, Y_3 i Z_1, Z_2, Z_3 respectivament.

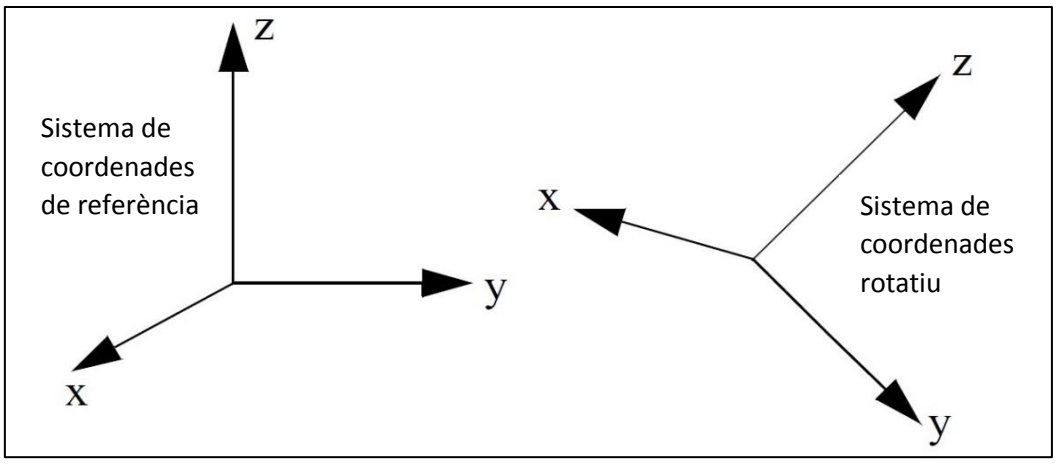
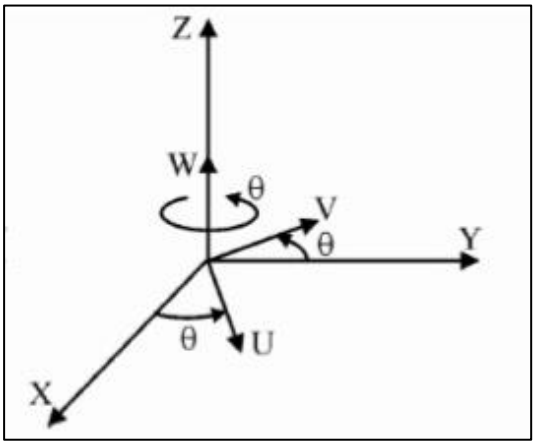


Figura 55: Rotació del sistema de coordenades

Per tant, si s'expressa el sistema de coordenades rotatiu des del sistema de referència s'obté una matriu com la següent:

$$\begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix}$$

En els casos pràctics, per fer la projecció del nou sistema es mantenen les lletres XYZ per al sistema de referència i es canvien per UVW en el sistema projectat. En el nostre problema, procedim a fer la projecció dels eixos en una rotació sobre l'eix Z com es mostrava en la Figura 54.



$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} U \\ V \\ W \end{pmatrix}$$

Si θ=90º,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} U \\ V \\ W \end{pmatrix}$$

Una vegada obtinguda la matriu de rotació només queda extreure el valor dels quaternions mitjançant unes formules que proporciona el mateix manual de ABB {1 i 2}.

$$q1 = \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2}$$

$$q2 = \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2} \qquad signe\ q2 = signe\ (y_3 - z_2)$$

$$q3 = \frac{\sqrt{y_2 - x_1 - z_3 + 1}}{2} \qquad signe\ q3 = signe\ (z_1 - x_3)$$

$$q4 = \frac{\sqrt{z_3 - x_1 - y_2 + 1}}{2} \qquad signe\ q4 = signe\ (x_2 - y_1)$$

Fent la substitució de valors corresponent:

$$q1 = 0.707$$

$$q2 = 0$$

$$q3 = 0$$

$$q4 = -0.707$$

Amb els valors dels quaternions obtinguts l'eina es mantindrà girada 90º respecte la posició habitual. D'aquesta manera, la ventosa queda de forma vertical i és capaç d'agafar correctament els smartphones a paletitzar.

10.2.2.3 Utilització dels objectes de treball

Com s'ha explicat en la definició de la sistema de paletitzat (Apartat 9.3) per a definir la posició concreta a la que s'havia de moure el robot s'han utilitzat els objectes de treball. Al tenir dues línies de paletitzat s'ha trobat convenient independitzar cadascuna de les línies i per aquest motiu s'ha utilitzat un objecte diferent per a cada una. Aquest fet permet modificar una de les línies sense que l'altre es vegi afectada.

Els objectes de treball venen definits per una sèrie de paràmetres. Tot seguit es mostren els dos objectes de treball utilitzats (PosPalet1 i PosPalet2) i la configuració que se'ls hi ha donat.

```
PERS wobjdata PosPalet1:=[FALSE,TRUE,"",[[200,-500,0],[1,0,0,0]],[[140,80,0],[1,0,0,0]]];
PERS wobjdata PosPalet2:=[FALSE,TRUE,"",[[200,350,0],[1,0,0,0]],[[110,110,0],[1,0,0,0]]];
```

- FALSE: El robot no subjecte l'objecte de treball sinó que manipula l'eina.
- TRUE: El sistema de coordenades de l'usuari és fix. No s'utilitzen eixos externs.
- "": Nom de la unitat mecànica que es coordina amb el moviment del robot, és a dir, la referència dels eixos externs. Al no utilitzar-ne es deixa en blanc.
- [[200,-500,0],[1,0,0,0]]: Sistema de coordenades de l'usuari, definint la posició amb XYZ en el primer paràmetre i la orientació amb quaternions en el segon. Aquestes coordenades fan referència al sistema de coordenades de la base.
- [[140,80,0],[1,0,0,0]]: Sistema de coordenades de l'objecte, definint la posició amb XYZ en el primer paràmetre i la orientació amb quaternions en el segon. Aquestes coordenades fan referència al sistema de coordenades de l'usuari.

En la següent imatge es pot observar que el sistema de coordenades de l'objecte de treball està referenciat al sistema de coordenades de l'usuari.

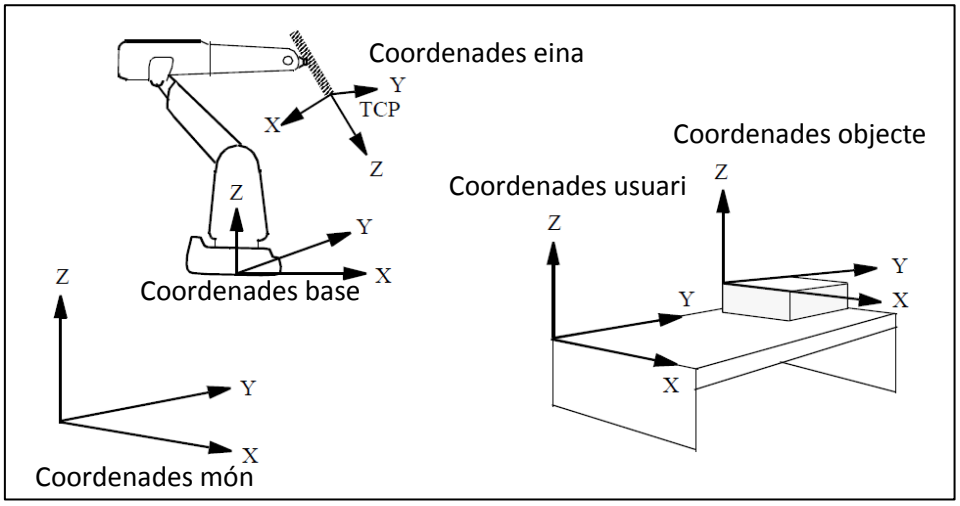


Figura 56: Definició dels sistemes de coordenades

Una vegada coneguts tots els sistemes de coordenades existents cal remarcar que les coordenades usuari estan referenciades a les coordenades de la base. Per tant, com hem pogut comprovar en l'objecte de treball PosPalet1 al estar situat a la dreta del robot queda amb una coordenada Y negativa.

Per altra banda, les coordenades de l'objecte fan referència a les coordenades usuari, i no a la base. Per aquest motiu, com que l'objecte queda a la part esquerra es defineix amb coordenades positives.

Per fer una traducció de la imatge anterior ens basarem en la Figura 57. Partint de la base que l'objecte de treball coincideix en el pla XY de les coordenades d'usuari, no hi haurà diferència de posicionament en l'eix Z.

Durant el desenvolupament s'ha decidit escollir com a centre de les coordenades d'usuari l'extrem inferior dret del palet i com a centre de les coordenades de l'objecte el centre d'aquest palet.

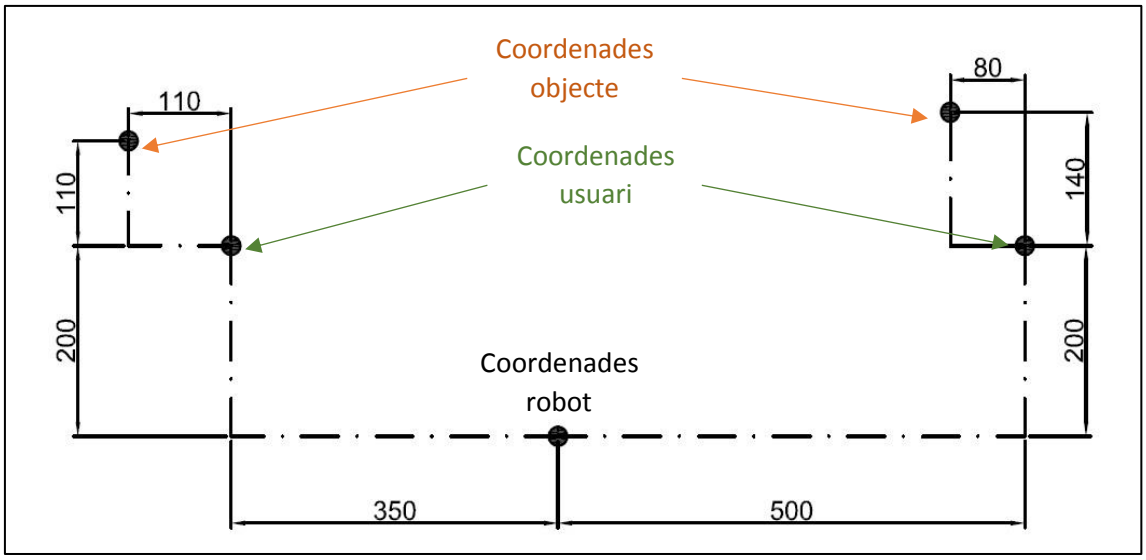


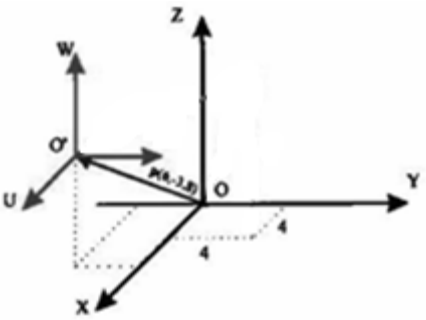
Figura 57: Sistemes de referència del paletitzat

Aquesta elecció és deguda a les previsions de futur per adaptar-se als nous models de dispositiu més grans com indica la tendència, al paletitzat de tauletes tàctils o simplement a un paletitzat de més quantitat de dispositius. El fet de situar el centre de coordenades usuari en l'extrem inferior dret i seguir una paletització en sentit contrari a les agulles del rellotge fa que aquest centre de coordenades actuï com a limitador del paletitzat.

Des d'aquest punt es permetria créixer en escala en direcció contraria adaptant el centre de coordenades de l'objecte al paletitzat d'un altre dispositiu.

En aquest cas, els sistemes de coordenades dels objectes no han rebut cap modificació en la orientació ja que s'ha mantingut el sistema de referència de la base del robot. Per altra banda, com és evident, sí que han rebut una translació respecte aquest.

Les translacions dels eixos de coordenades, igual que les rotacions, també es poden representar utilitzant matrius.



La translació d'un sistema es fa a partir d'un un vector *t* (t1,t2,t3) que determina el centre del nou sistema de coordenades UVW.

Per determinar les coordenades dels punts a representar en el nou sistema de coordenades cal aplicar la matriu de les transformacions geomètriques.

$$T = \begin{bmatrix} R_{3x3} & P_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} \cdot p_{4x1}$$

On, R_{3x3} és la matriu de rotació que si no existeix es substitueix per la matriu identitat.

P_{3x3} és el vector de translació que si no existeix es substitueix per un vector unitari.

f_{1x3} és el vector de la perspectiva d'un sistema de visió que serà nul si no s'utilitza.

w_{1x1} és el valor de l'escala utilitzada.

p_{4x1} és el vector del punt a representar format per (x,y,z,1)

Com es pot observar, el vector del punt és de 4 components degut a que la matriu de les transformacions geomètriques és una matriu 4x4. Per això cal representar el vector en coordenades homogènies, on un punt x,y,z en coordenades cartesianes equivaldrà a un punt x',y',z',n en coordenades homogènies sempre que es compleixi:

$$x = x'/n$$

$$y = y'/n$$

$$z = z'/n$$

Per tant, en el cas que ens ocupa:

$$T = \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_1 \\ p_3 \\ n \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

Si apliquem la matriu de transformació obtinguda en el cas del sistema de coordenades usuari de la línia de paletitzat 1, podrem obtenir el punt del sistema de coordenades de l'objecte respecte el sistema de les coordenades del robot.

- El sistema de coordenades d'usuari serà el vector de translació [200,-500,0]
- El sistema de coordenades de l'objecte serà el punt a traslladar [140,80,0]

Per tant,

$$T = \begin{bmatrix} 1 & 0 & 0 & 200 \\ 0 & 1 & 0 & -500 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 140 \\ 80 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 340 \\ -420 \\ 0 \\ 1 \end{bmatrix}$$

Degut a que aquest cas és molt senzill i amb nombre exactes, mitjançant l'anterior Figura 57 es pot comprovar que el resultat obtingut és correcte i que per tant la transformació també és correcte.

En el cas del TCP també es podria haver aplicat la matriu de transformacions geomètriques substituint per la matriu de rotació obtinguda i el vector de translació per un vector unitari. D'aquesta forma es coneixeria un punt concret des del sistema de la base del robot però la intenció era justificar i demostrar el funcionament dels quaternions.

10.2.2.4 Definició d'un punt amb l'eina i l'objecte de treball

Per a la definició d'un punt concret durant el procés de paletitzat, segons el que s'ha explicat anteriorment, ve determinat per unes coordenades que faran referència a un objecte de treball i a una eina concreta.

A continuació es pot veure la instrucció que determina la posició de paletitzat i els paràmetres necessaris per situar el robot.

```
MoveL Caixa, vel, fine, Ventosa, \WObj:=PosPalet1;
```

- MoveL: Indica un moviment lineal del robot
- Caixa: Posició de paletitzat variable en funció de la posició a ocupar dins el paletitzat
- vel: Variable global que indica la velocitat de moviment del robot
- fine: Determina que la posició és assolida de forma exacte
- Ventosa: Eina utilitzada i que determina el TCP
- \WObj:=PosPalet1: S'afegeix l'objecte de treball utilitzat

La posició *Caixa* es calcula en funció de l'objecte de treball i recordem que en qualsevol dels dos casos el centre de coordenades es troba en el centre de la base del palet. Aquesta posició indicada com a variable tindrà internament unes coordenades i una orientació concreta expressada en forma de quaternions.

Per determinar la posició i orientació final del TCP per a la posició s'ha de conèixer l'objecte de treball actual i quina orientació li volem donar respecte aquest.

Sabent que el sistema de coordenades de l'objecte es troba com mostra la imatge següent, es pot calcular l'orientació a donar al TCP respecte aquest perquè quedi de la forma desitjada.

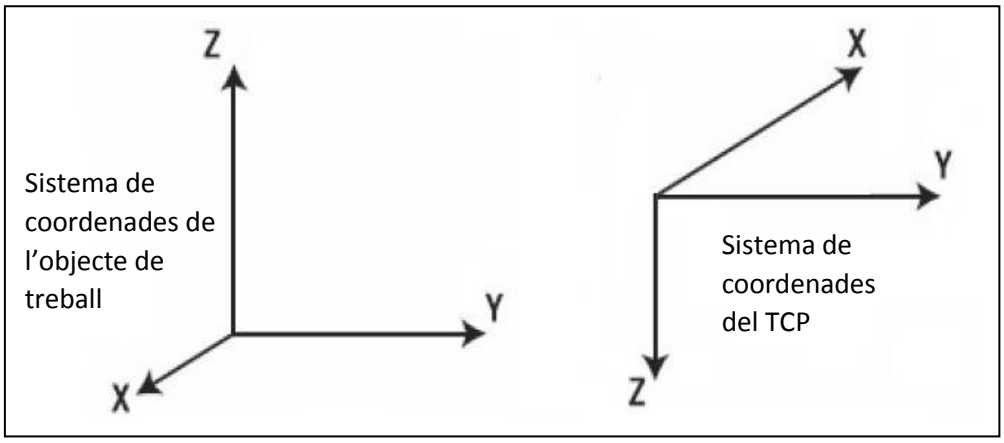


Figura 58: Orientació del TCP respecte un objecte de treball

Que el sistema de coordenades del TCP miri al revés és molt important ja que el paletitzat es fa per la part superior. Dibuixar els sistemes de coordenades ens permet fer les projeccions corresponents i extreure la matriu de rotació. Posteriorment s'obtenen els quaternions per incloure en el programa RAPID.

Per tant, seguint el procediment explicat en els apartats anteriors:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} U \\ V \\ W \end{pmatrix}$$

Si l'angle de rotació és de 90º,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} * \begin{pmatrix} U \\ V \\ W \end{pmatrix}$$

Fent la substitució de valor en les fórmules per extreure els quaternions,

$$q1 = \frac{\sqrt{-1+1-1+1}}{2} = 0$$

$$q2 = \frac{\sqrt{-1-1+1+1}}{2} = 0 \qquad signe\ q2 = signe\ (0-0)$$

$$q3 = \frac{\sqrt{+1-0-1+1}}{2} = 1 \qquad signe\ q3 = signe\ (0-0)$$

$$q4 = \frac{\sqrt{-1+1-1+1}}{2} = 0 \qquad signe\ q4 = signe\ (0-0)$$

Amb aquesta configuració el TPC queda mirant avall, amb l'eix Z mirant al terra, i la orientació desitjada per a deixar els dispositius correctament.

Aquesta configuració seria correcte per a 6 de les 8 posicions de paletitzat, 4 de la forma quadrada i 4 de la forma circular. En el paletitzat circular 2 de les posicions requereixen una rotació de 90º per encaixar amb la forma de paletitzat dissenyada.

Per a poder fer aquesta rotació en l'orientació de la caixa, caldria fer una rotació de 90º graus sobre l'eix Z de la orientació obtinguda anteriorment. Visualment es mostraria de la següent manera:

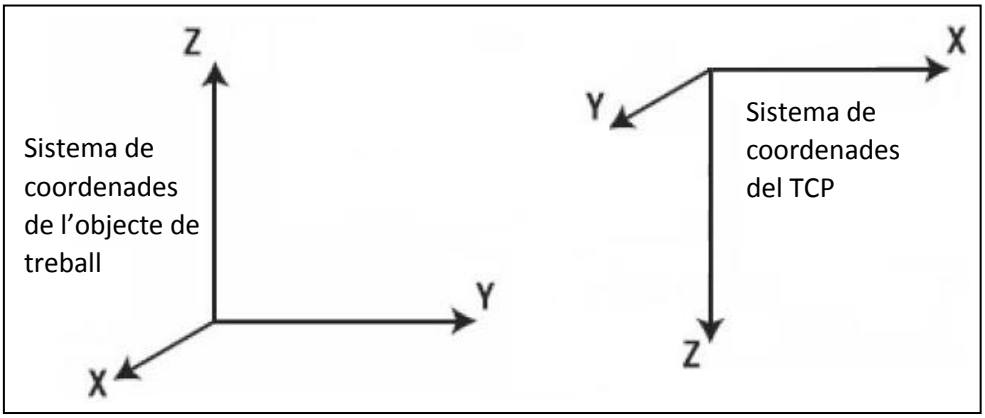


Figura 59: Orientació i rotació del TCP respecte un objecte de treball

En aquest segon cas la matriu de rotació obtinguda queda de la forma següent:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} * \begin{pmatrix} U \\ V \\ W \end{pmatrix}$$

Si calculem el valor del quaternions,

$$q1 = 0$$

$$q2 = 0.707$$

$$q3 = 0.707$$

$$q4 = 0$$

Aquesta orientació permetrà la rotació dels paletitzats 2 i 4 del primer pis del paletitzat circular i els corresponents dels pisos superiors.

Per al canvi de la rotació en aquestes posicions concretes s'utilitza la instrucció següent:

```
Caixa.rot:=[0,0.707,0.707,0];
```

10.2.1 CONFIGURACIÓ DEL CLIENT OPC UTILITZANT EL PROGRAMA iFIX

De la mateixa forma que s'ha fet en l'Arduino, per a la controladora IRC5 també cal definir i configurar un servidor OPC per a comunicar les dades. En aquest cas però, l'ordinador del laboratori que s'ha utilitzat per a l'aplicació SCADA ja té configurat aquest OPC.

En la següent imatge es pot observar aquest OPC servidor i la seva configuració.

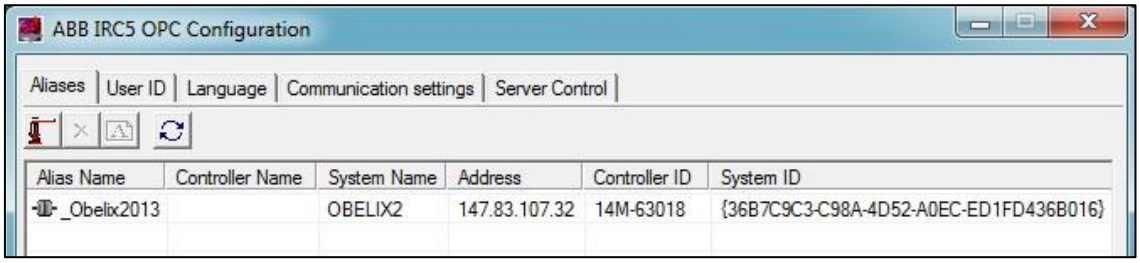
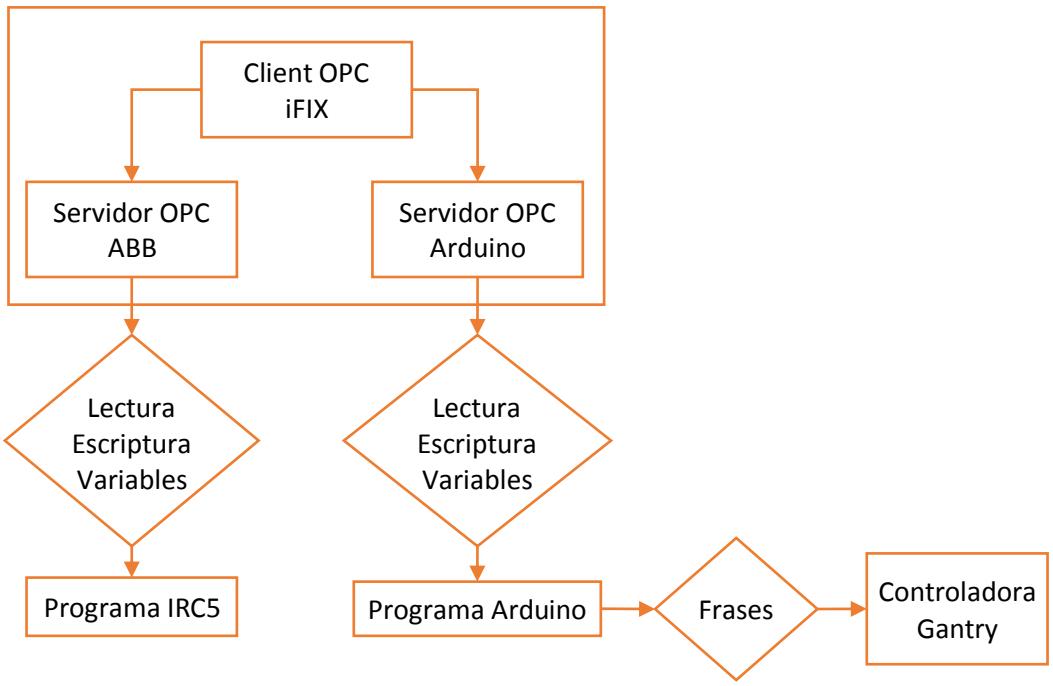


Figura 60: ABB IRC5 OPC

Recordant que la comunicació entre la controladora i el PC es fa utilitzant Ethernet, en la configuració de l'OPC es pot veure que l'adreça IP (147.83.107.32) correspon a la del robot.

Així doncs, una vegada configurats els dos servidors OPC és hora de definir qui serà el client que interactuarà amb cadascun d'ells i permetrà un intercanvi de dades. La definició i configuració de l'OPC client s'ha realitzat utilitzant el propi programa del sistema SCADA Proficy iFIX. Abans però, es mostra el diagrama que seguirà l'intercanvi de dades entre dispositius.



En iniciar el programa s'obra una finestra com la de la Figura 61: Finestra inicial però amb un directori *SCU File* que ve per defecte. La configuració del client OPC només cal realitzar-la en iniciar una nova aplicació, les altres vegades només s'ha d'indicar el directori on hi ha guardada la configuració a través del *SCU File* i executar l'iFIX.

Per procedir a la configuració inicial de l'OPC client s'ha d'anar a la tercera icona de la finestra dins l'apartat SCU (System Configuration Utility).

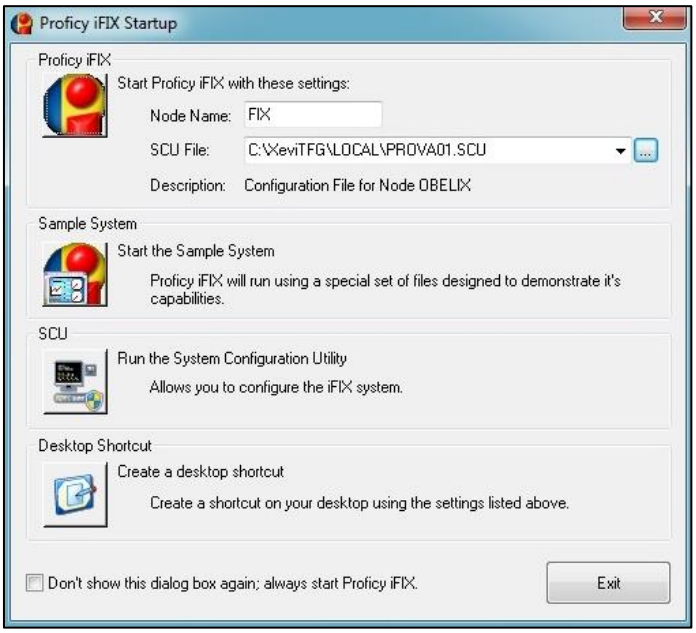


Figura 61: Finestra inicial iFIX

La icona ens porta a una finestra secundaria com la que es pot veure a continuació. Aquesta finestra serà la base de la configuració del client i hi caldrà modificar diferents apartats. El primer, serà el directori principal on voldrem guardar el nostre projecte i en el qual també es guardarà l'aplicació SCADA que s'anirà desenvolupant.

Figura 62: Configuració SCU-Pantalla principal

Per fer-ho premem en la primera icona de la part inferior representada per unes carpetes.

A continuació, es deixa per defecte la base i la llengua del projecte en l'apartat *Location of System Software and Data Files*. En el segon apartat, *Application Paths In Current Project Configuration*, canviarem el directori del projecte (*Project*) a una carpeta on voldrem guardar tot el projecte. A continuació premem *Change Project* perquè els directoris que queden sota *Project* es canviïn automàticament. Al ser la primera vegada demanarà si es volen crear les carpetes necessàries, acceptem.

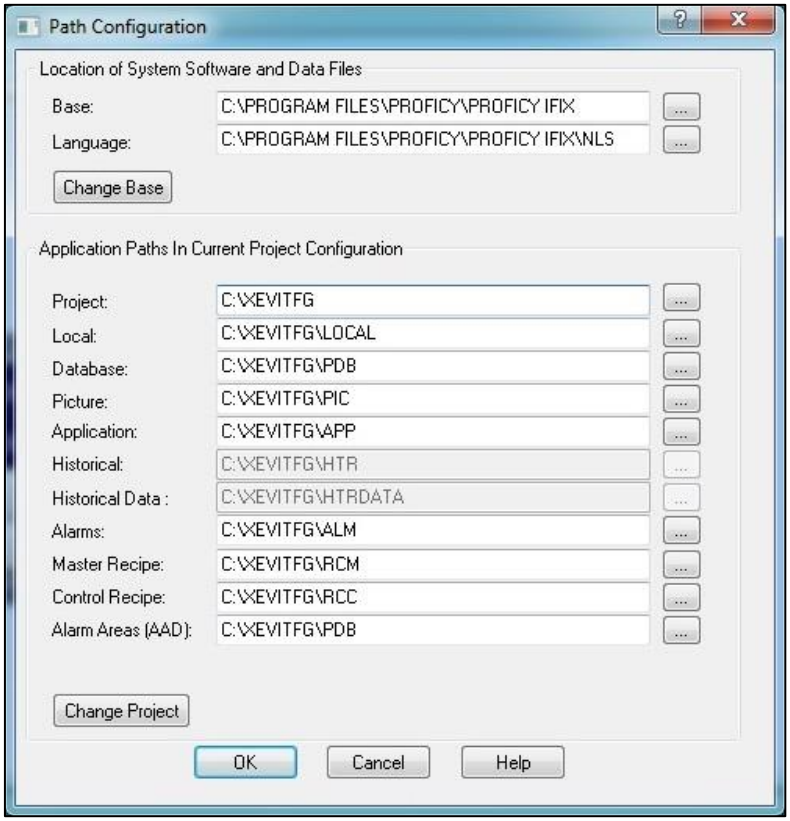


Figura 63: Configuració SCU-Directori del projecte OPC client

Per altra banda, també demanarà si volem deixar el directori escollit com a directori per defecte. Al ser un ordinador públic de l'escola no acceptarem la petició. Aquest és el motiu pel qual cada vegada que vulguem inicial l'aplicació caldrà indicar el directori on es troba el projecte.

Una vegada tancada la finestra de la configuració del directori seleccionarem la quarta icona de la pantalla principal SCU. Aquí es configuren els divers que tindrà accés al sistema SCADA, és a dir, els OPC servidors. Perquè sigui possible cal habilitar l'opció de *SCADA Support*. A continuació inserim un *I/O Driver Name* del tipus *OPC-OPC Client v7.42a* utilitzant el desplegable de la dreta i el botó *Add*.

En mostrar-se la llista, el seleccionem i premem *Configure...*

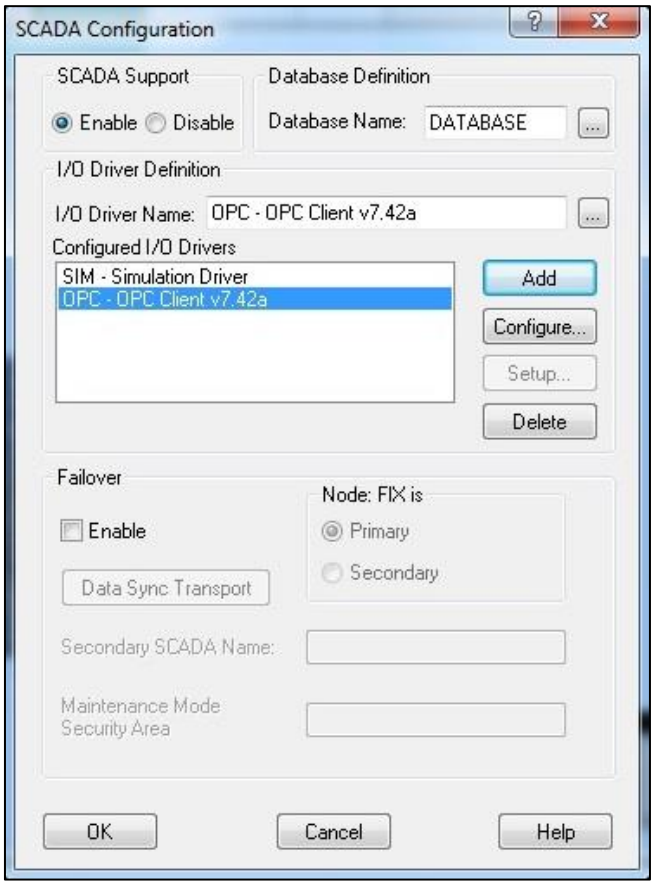


Figura 64: Configuració SCU-SCADA

Apareixerà una finestra on s'haurà d'indicar que volem utilitzar un OPC local seleccionant *Use Local OPC Client Driver*.

Tot seguit s'executarà un subprograma anomenat Power Tool. En aquest afegirem els dos OPC servidors i seleccionarem de les variables disponibles, les que vulguem monitoritzar a través de l'aplicació de supervisió.

Per fer-ho seleccionem afegir un nou dispositiu mitjançant la icona del connector que podrem veure a la part inferior esquerra de la pantallaFigura 65. Apareixerà un llistat dels OPC instal·lats en l'ordinador. Per a l'explicació d'aquest apartat agafarem l'OPC de ABB, en el cas de l'Arduino el procés que segueix seria exactament el mateix seleccionant l'OPC d'Arduino.

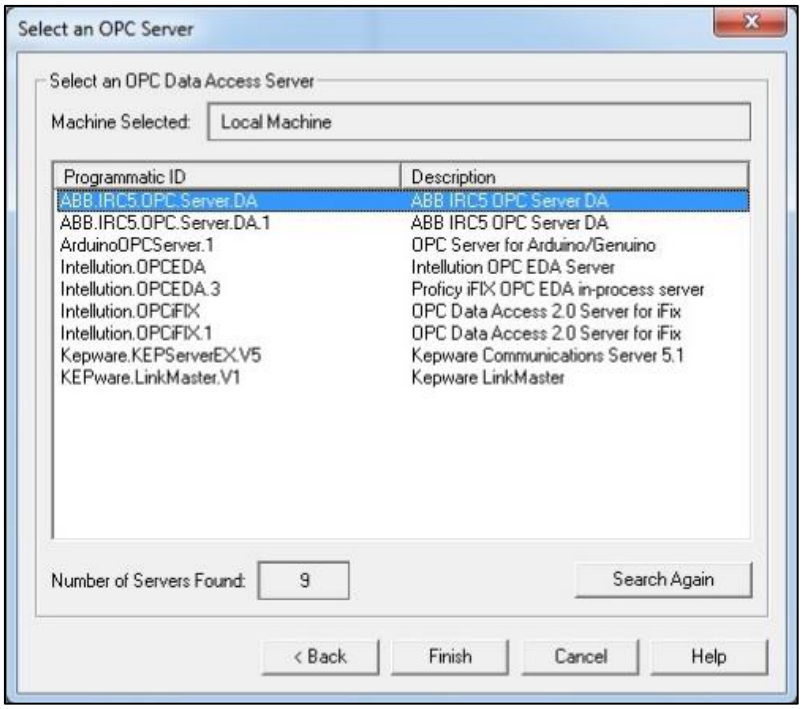


Figura 65: Power Tool-Selecció del servidor OPC

Així doncs, seleccionem *ABB.IRC5.OPC.Server.DA* i premem *Finish*. En la finestra de la Power Tool apareix un dispositiu al que se li ha donat el nom de **IRC5OPC** i en el cas de l'Arduino, **ArduinoOPC**. La única modificació que s'ha fet en el dispositiu i que es repetirà en els següents punts és activar la casella *Enable* que surt a la part superior dreta de les característiques en seleccionar el dispositiu. Si no es selecciona, les dades no es transmetrien ja que es deshabilitaria l'entrada de dades de la IRC5.

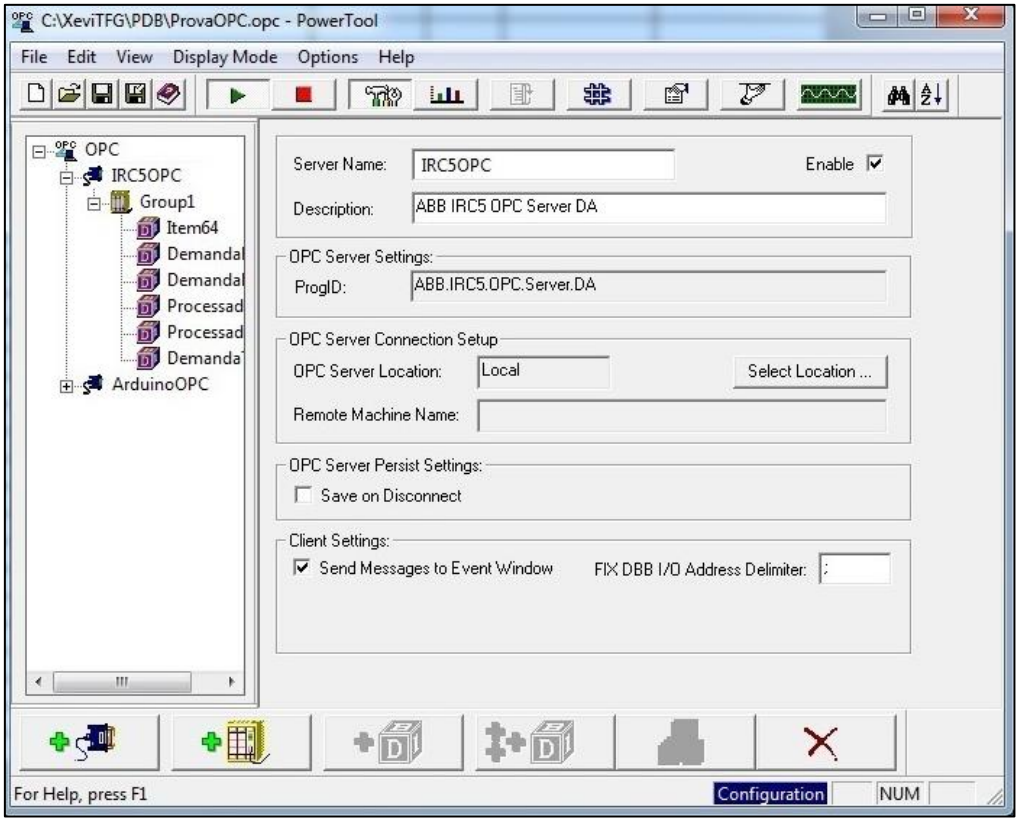


Figura 66: Power Tool-Configuració d'un dispositiu

Dins del dispositiu es poden crear diferents grups de variables, en el nostre cas se n'ha creat un de sol on hi seran totes. Per fer-ho cal prémer en la icona de dispositiu que s'ha habilitat a la part inferior.

Aquí tampoc s'han fet modificacions de la configuració per defecte, però igual que en el cas del dispositiu s'ha habilitat el grup de variables.

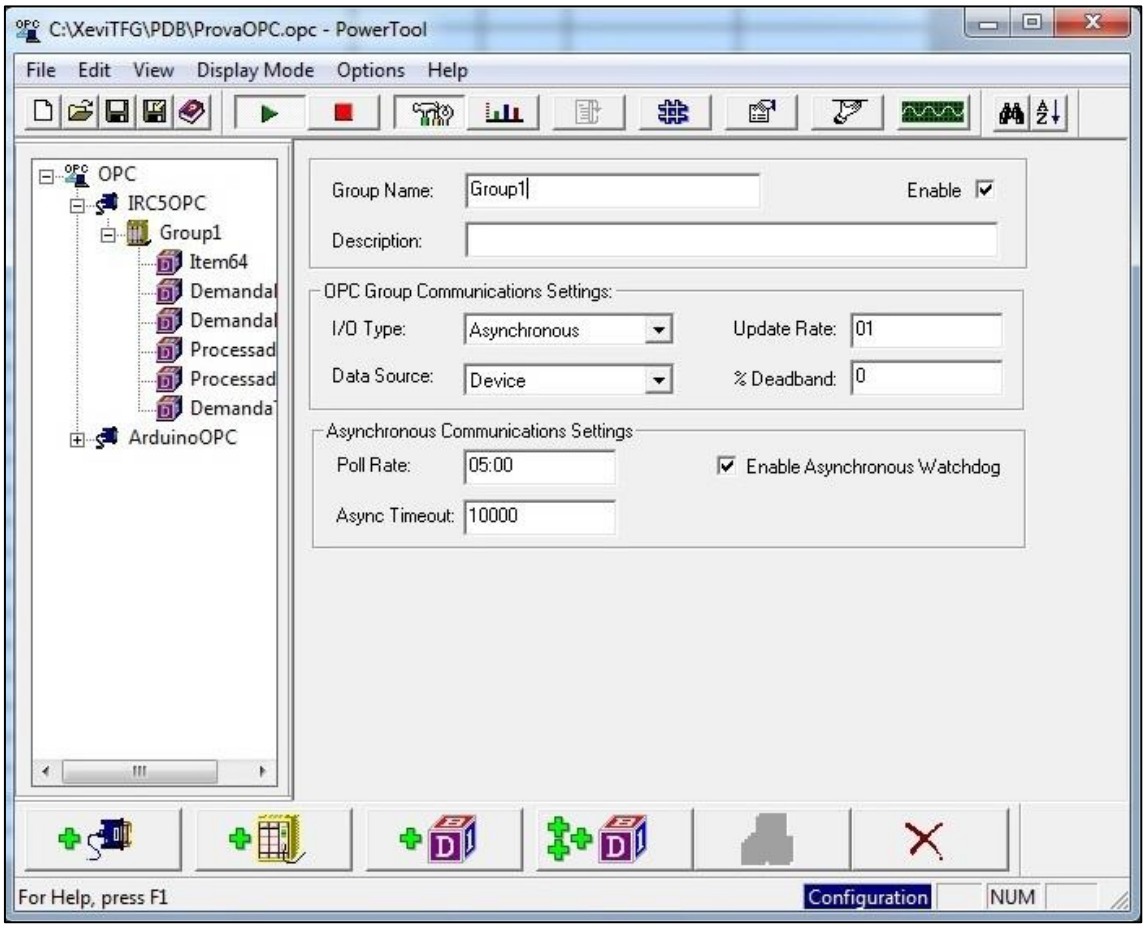


Figura 67: Power Tool-Configuració d'un grup de variables

Una vegada creat el grup cal inserir les variables desitjades. En aquest cas es pot fer de dues maneres a través de la selecció individual de variables o de múltiples variables. Les variables escollides s'hauran d'anar a buscar dins el directori que inclou el programa del robot. Degut a que únicament hem creat un grup i que totes les variables del robot es troben declarades en el mateix mòdul de programa, s'ha escollit la selecció múltiple mitjançant l'últim botó habilitat.

Com es pot observar a la Figura 68 caldrà desplegar el directori fins a arribar als mòduls de programa. Dintre d'aquests, les variables disponibles es mostraran en el mòdul on s'han declarat que en el programa implementat serà dins el modul **Variables**.

ABB.IRC5.OPC.Server.DA → _Obèlix2013 → IOSYSTEM → RAPID → T_ROB → Variables.

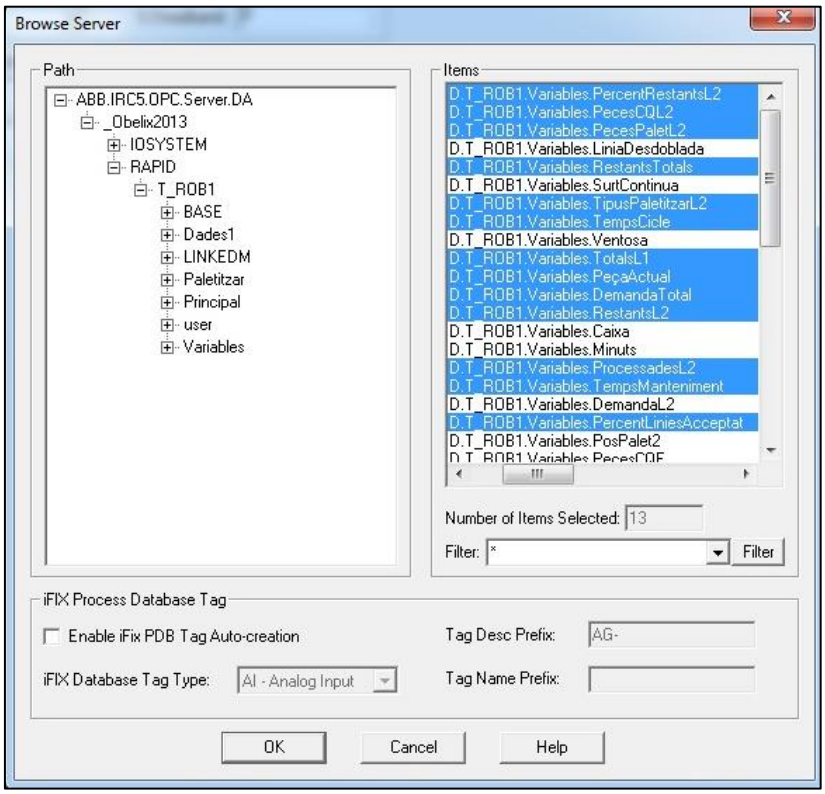


Figura 68: Power Tool-Selecció de múltiples variables

Si es selecciona alguna de les variables seleccionades també ens apareixen les característiques d'aquesta i igual que en la creació de dispositiu i de grup, habilitem totes les variables que vulguem monitoritzar.

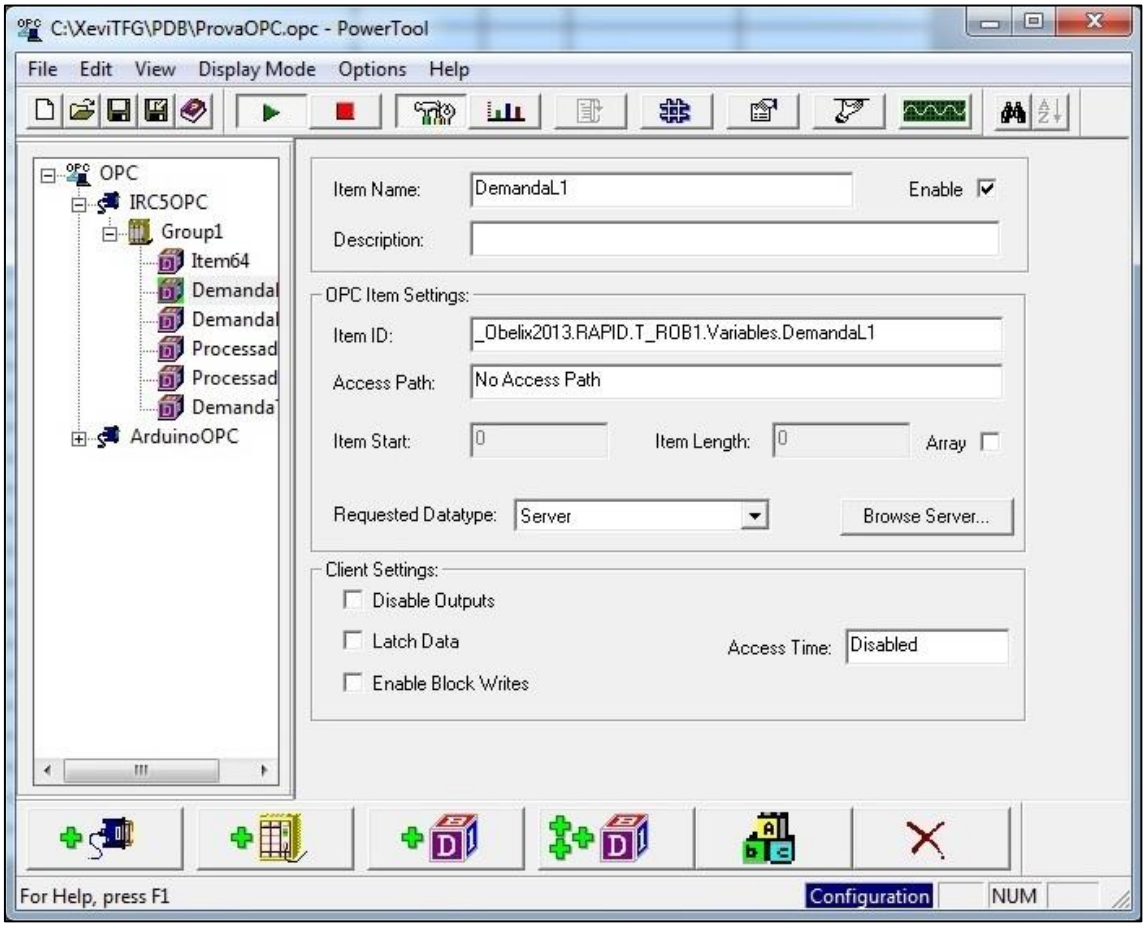


Figura 69: Power Tool-Configuració de variables

Una vegada finalitzada la selecció de les variables en els dos dispositius, cal prémer la icona amb els símbol *PLAY* de color verd que podem trobar a la part superior perquè arranqui la captura dels dispositius configurats.

Per últim s'ha guardat la configuració dels OPC en la carpeta PDB creada automàticament en la creació del projecte OPC client, que es troba dins la carpeta *XeviTFG*.

Una vegada guardat el projecte cal incloure a la carpeta PDB els divers OPC que utilitza l'iFIX per entendre i processar el direccionament de variables. Aquests divers formen part del pack software adquirit per l'escola i es troben dins la carpeta PDB original en el directori de l'ordinador del laboratori.

- C:\Archivos de programa\Proficy\Proficy iFIX\PDB

A continuació cal copiar els arxius d'aquesta carpeta i guardar-los a la carpeta PDB del projecte.

- C:\XeviTFG\PDB

En la carpeta del projecte ja s'hi hauran creat alguns arxius en guardar la configuració de l'OPC feta a través de PowerTool. Tot i això, es copien en aquesta carpeta tots els arxius de la carpeta PDB original. En fer-ho Windows detectarà que alguns arxius ja existeixen i ens demanarà si els volem sobreescriure, li direm que no.

En finalitzar, tornarem a la pantalla principal de la configuració SCU. Seleccionarem la cinquena icona *Task Configuration*, obrint-se la següent finestra:

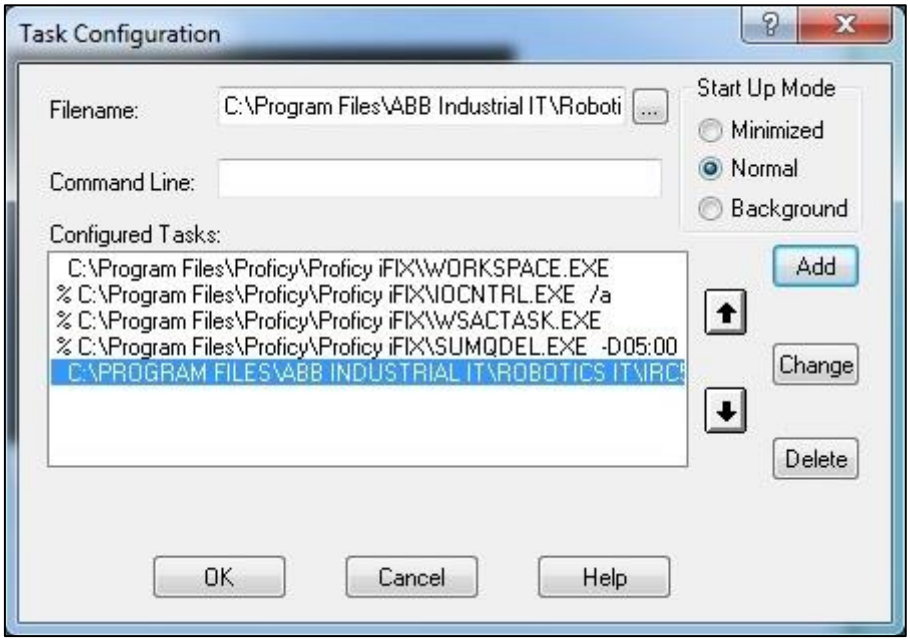


Figura 70: Configuració SCU-Executables

Revisarem que hi ha els executables WORKSPACE i IOCONTROL, si no hi són els buscarem a la carpeta Proficy iFIX original. A més, perquè l'OPC de ABB s'inicii en obrir l'iFIX afegirem l'executable RobOPC.exe del directori:

- C:\Archivos de programa\ABB Industrial IT\Robotics IT\IRC5 OPC Server

Per últim, sortim i guardem l'arxiu SCU des de la pantalla principal.

10.2.2 CREACIÓ DE L'APLICACIÓ SCADA A TRAVÉS D'iFIX

Finalitzades les configuracions anteriors s'ha pogut iniciar la programació de l'aplicació del sistema SCADA que permet la interacció conjunta amb els dos robots.

En inicialitzar l'iFIX altra vegada torna a aparèixer la finestra inicial de la Figura 61. Per entrar al programa de l'SCADA i que aquest estigui referenciat a les configuracions prèvies cal canviar el directori de l'arxiu SCU creat anteriorment. Una vegada canviat, s'inicia el programa prement la icona de *Proficy iFIX*.

Una vegada iniciat es pot comprovar el funcionament de l'ABB IRC5 OPC Configuration executant el programa i observant la finestra Server Control. Si l'OPC està funcionant només es podrà prémer el polsador *Stop*.

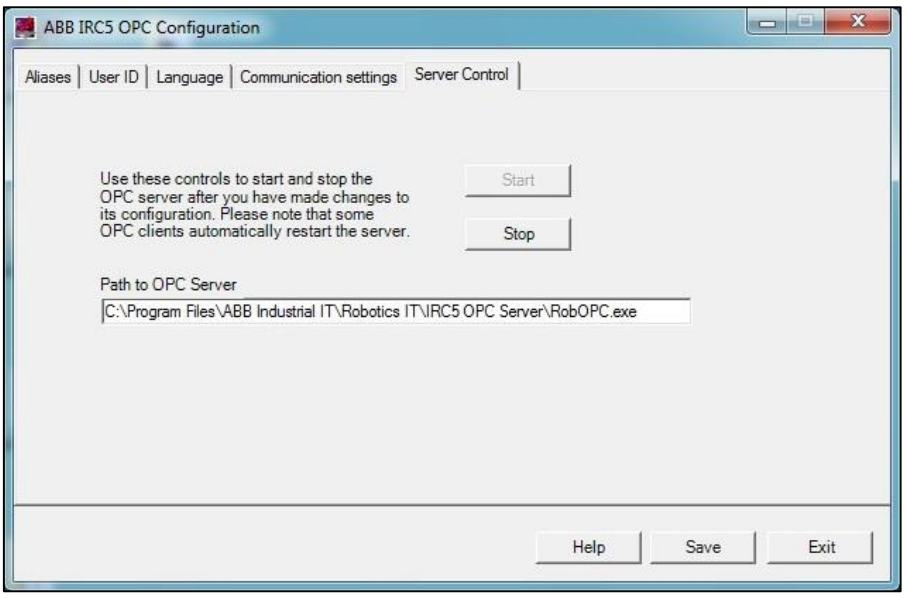


Figura 71: ABB IRC5 OPC-Inicialització

S'obrirà per defecte una pantalla en blanc anomenada **untitled1** de la que es pot canviar el nom o se'n pot crear una de nova. Per al desenvolupament del treball hem utilitzat una finestra amb el nom de **Principal**. En aquesta finestra s'han col·locat tots els valors, indicadors o representacions per mostrar a l'usuari de l'aplicació.

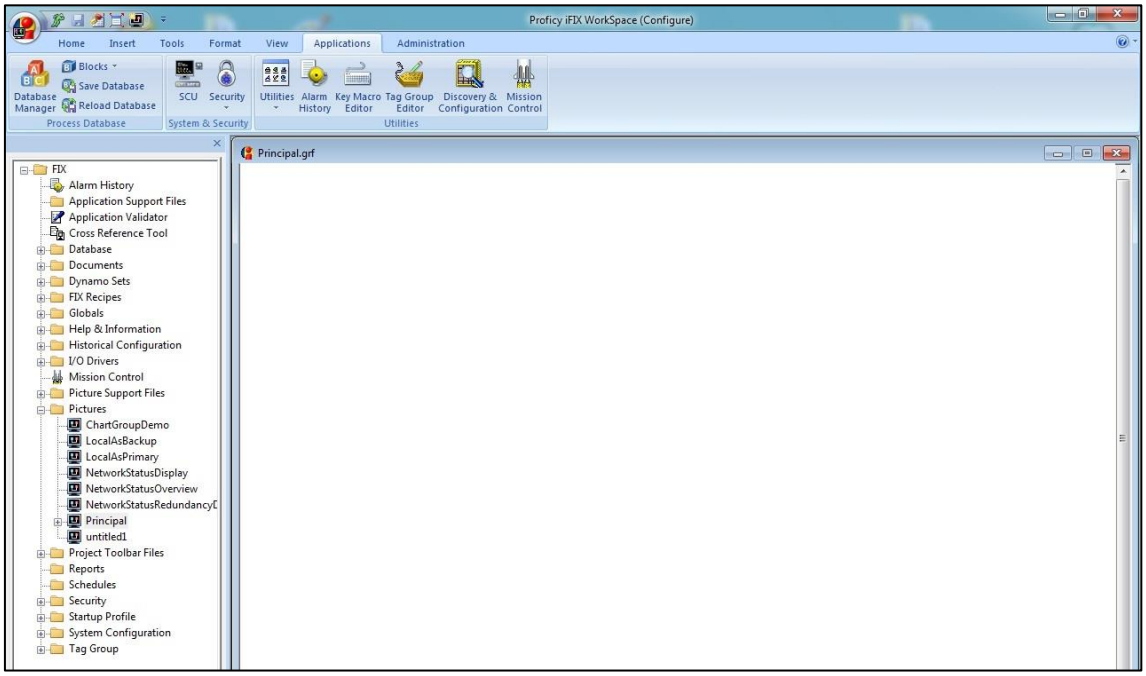


Figura 72: iFIX-Pantalla principal

Per poder mostrar qualsevol valor, cal primer declarar aquesta dada dins l'aplicació a través de la base de dades. Per fer-ho anem a la pestanya *Applications* i seleccionem *Database Manager* com es pot observar en la figura anterior.

En entrar a la base de dades es veurà una interfície com la de la Figura 73. El programa demanarà si es vol utilitzar una connexió local de la base de dades o buscar-la dins la xarxa. En el nostre cas, com que ens trobem dins una estructura centralitzada en el PC del laboratori i totes les comunicacions van a parar a aquest ordinador, seleccionarem que el node de la base de dades és local.

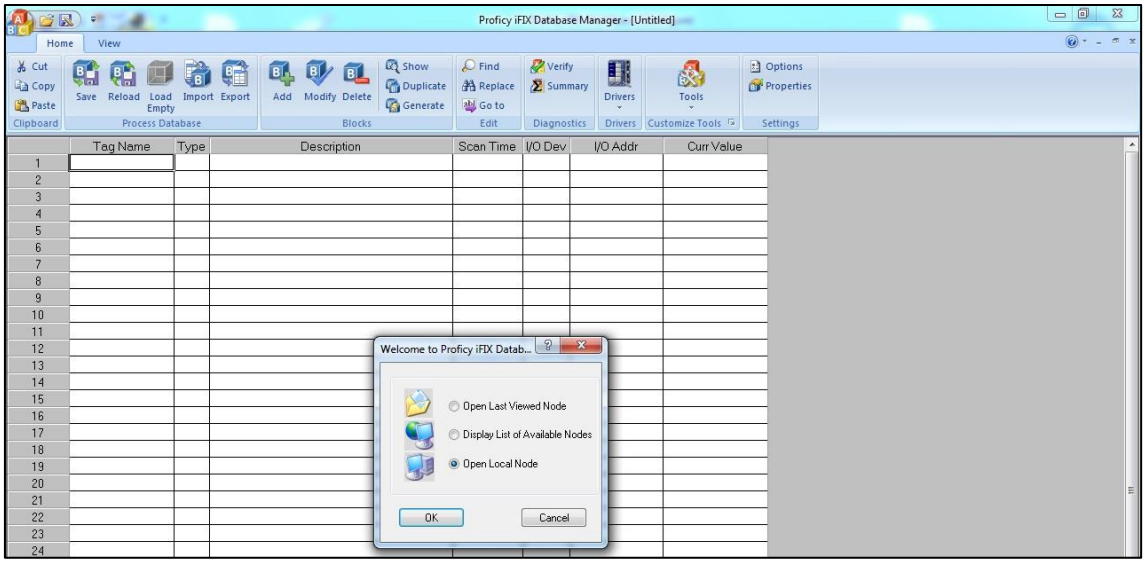


Figura 73: iFIX-Database Manager

Per afegir una nova variable a monitoritzar podem fer doble clic en qualsevol lloc de la taula de Tags que tenim de fons. Els tags són l'enllaç entre la variable de l'SCADA i la variable real de l'OPC.

El primer que es demana és seleccionar el tipus de tag que es vol afegir del llistat següent:

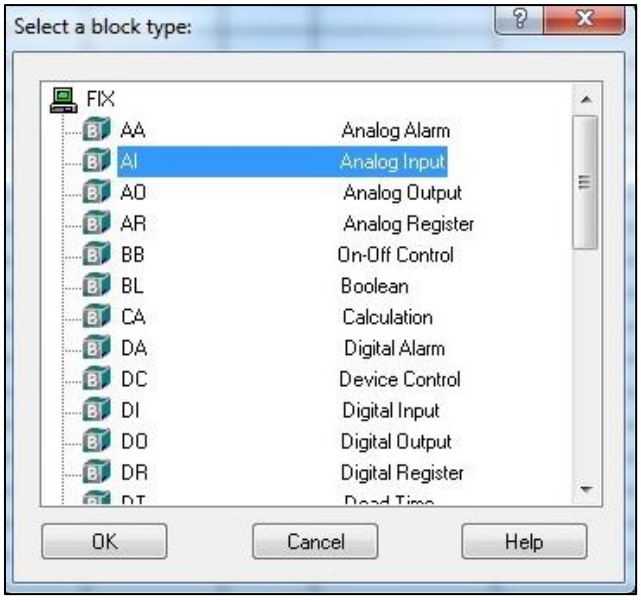


Figura 74: iFIX-Selecció del tipus de variable

De les diferents opcions disponibles només s'han utilitzat el llistat que es mostra a continuació:

- **Analog Input**: Entrada de valors al sistema SCADA
- **Analog Output**: Sortida de valors del sistema SCADA
- **Analog Register**: Valors interns
- **Digital Input**: Entrada de senyals al sistema SCADA
- **Digital Output**: Sortida de senyals del sistema SCADA
- **Text**: Cadenes de caràcters

Una vegada seleccionat el tipus de variable apareixerà una finestra on es demanarà el nom que se li vulgui donar al Tag. Per enllaçar amb la variable de l'OPC configurat seleccionem *OPC Client v7.42a* en l'apartat *Driver*.

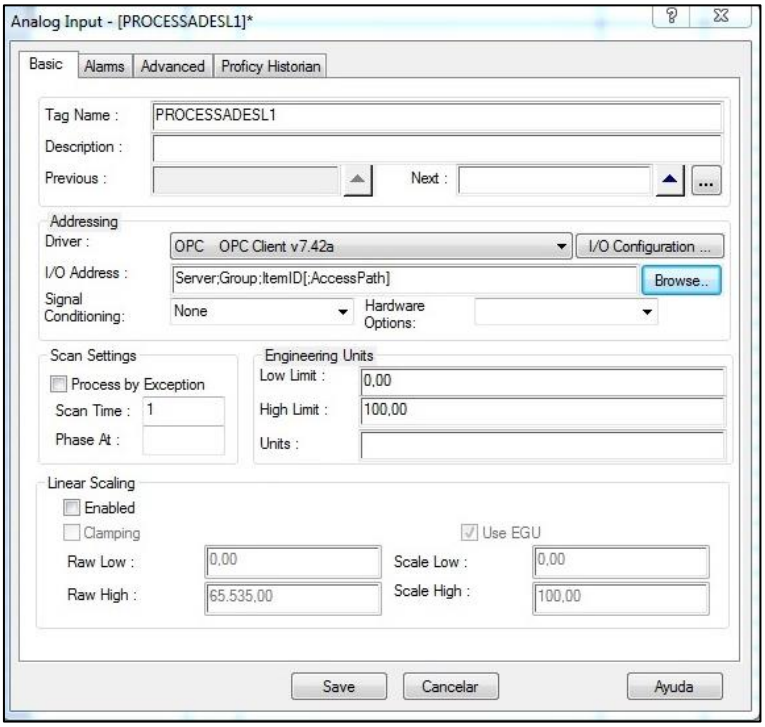


Figura 75: iFIX-Configuració d'una variable

Per seleccionar la variable a monitoritzar premem el polsador *Browse* de *I/O Adress*.

S'obrirà una finestra amb una interfície similar a la PowerTool i escollirem la variable desitjada.

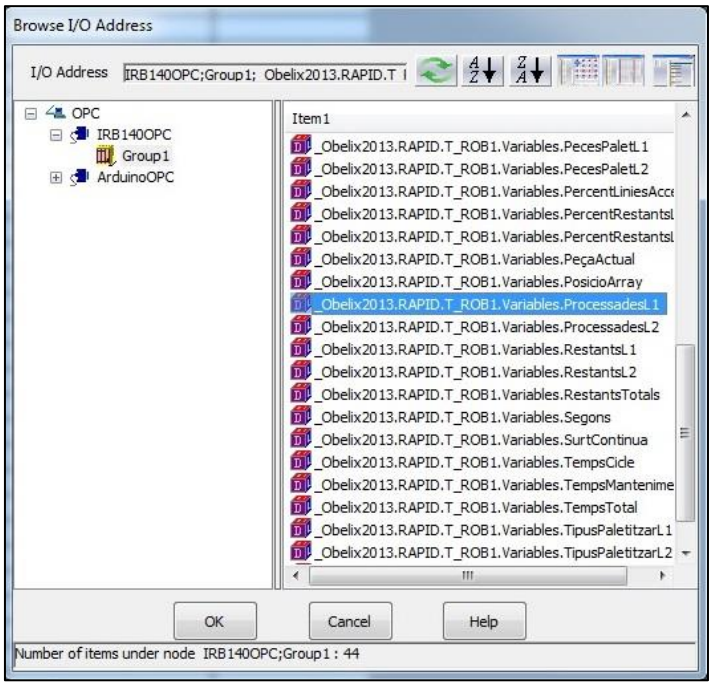


Figura 76: iFIX-Selecció d'una variable

Un cop seleccionada, tornarem a la finestra anterior on es podran configurar diferents paràmetres com ara límits o escalat. Aquests paràmetres varien en funció del tipus de variable. Prement el polsador de *Save* la variable queda guardada i tornarem a la pantalla principal de Database Manager.

En aquesta pantalla caldrà prémer el botó de *Save* i *Reload* per guardar i comprovar la base de dades del projecte.

El procés seguit es repetirà per cadascuna de les variables a mostrar i el procediment serà idèntic tant per les variables provinents de la IRC5 com per a les de l'Arduino.

Una vegada inclosos els tags a la base de dades es pot escollir des de la pantalla principal quines dades es volen mostrar. Per fer-ho ens situem a la pestanya superior *Insert*, seleccionem *Objects/Links* i es desplegaran totes les opcions disponibles.

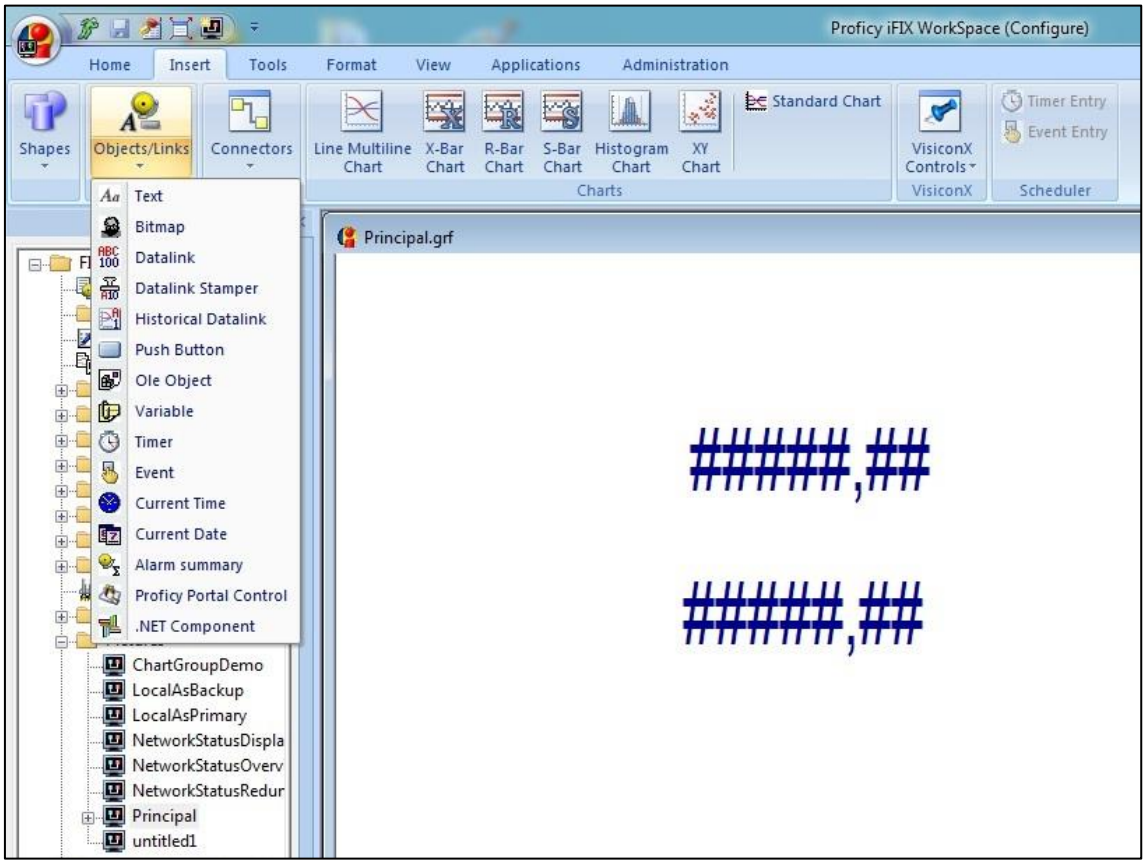


Figura 77: iFIX-Incloure objectes

Si per exemple seleccionem un *Push Button* s'afegirà un polsador a la pantalla que estiguem en aquell moment. A aquest polsador se li pot escriure un text al damunt fent doble clic sobre ell mateix i d'aquesta manera donar-li un nom. Prement el botó dret apareixen les opcions disponibles de les que destaquen *Animations* per afegir funcions relacionades al polsador i també *Edit Script*. Aquest últim permet programar en Visual Basic accions addicionals que es realitzaran quan l'aplicació estigui en funcionament i es premi el polsador. Al final d'aquest mateix apartat es mostrarà un exemple de la programació amb VisualBasic.

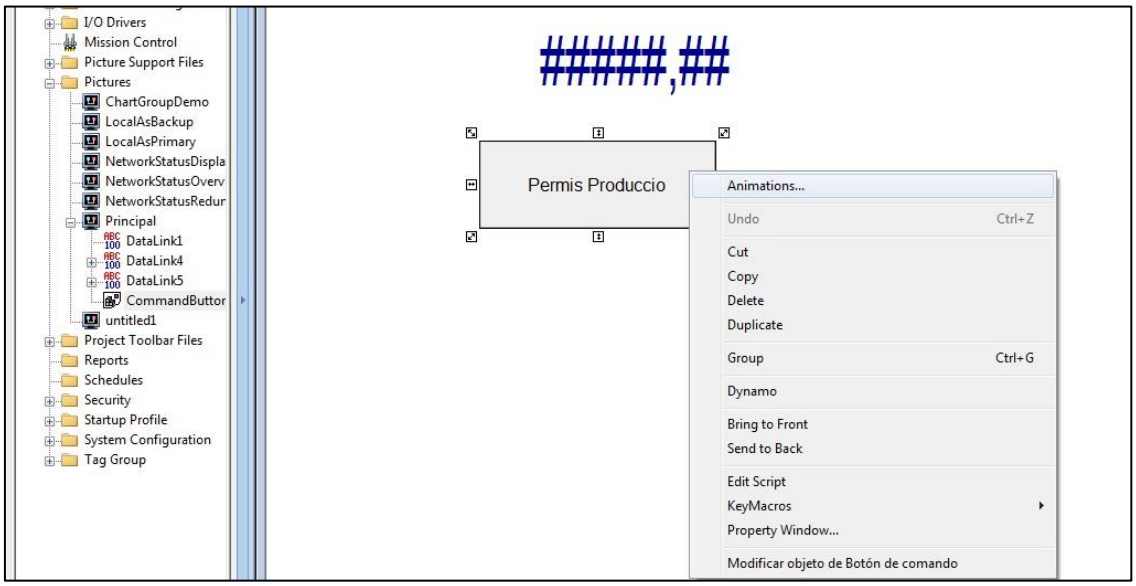


Figura 78: iFIX-Afegir objecte de tipus Push Button

Per donar una funció en prémer el botó seleccionem *Animations* i es desplegarà la finestra *Basic Animation Dialog* que podem veure a continuació. En aquesta finestra apareixen totes les opcions de visualització disponibles. Si ens hi fixem en detall veurem que algunes d'aquestes no es poden seleccionar apareixent en gris. Això és degut a que el tipus d'objecte que estem programant va obligatòriament relacionat a un tag del tipus booleà. Per aquest motiu, el programa bloqueja les visualitzacions que anirien relacionades a una variable de tipus analògica.

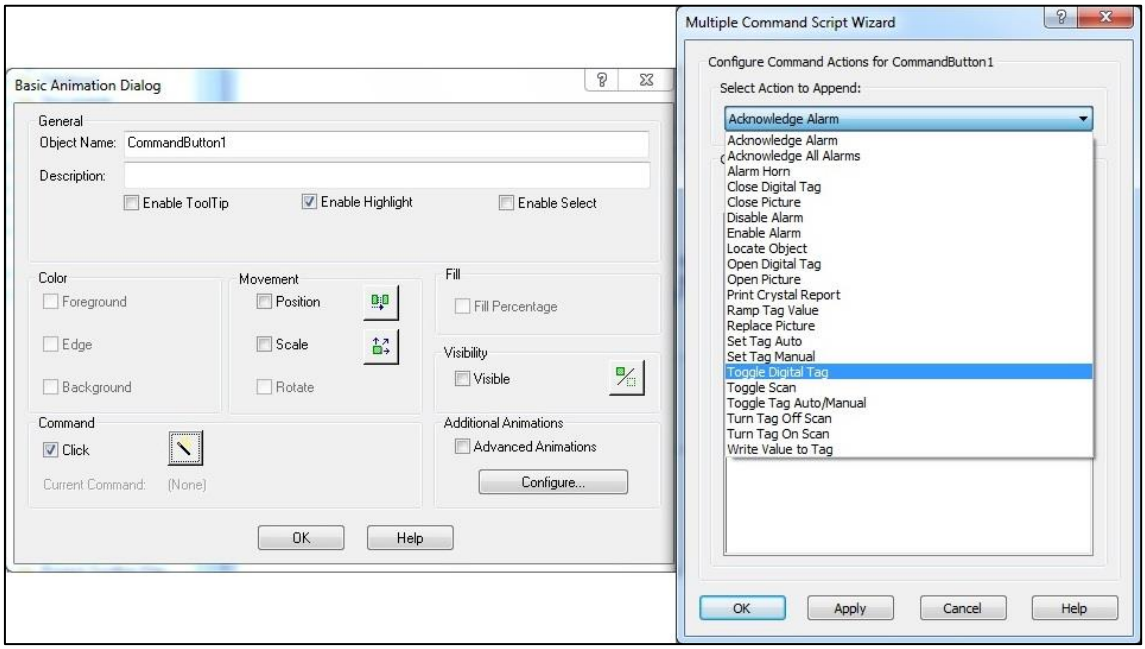


Figura 79: iFIX-Configuració dels objectes Push Button

Per realitzar una acció en prémer el polsador seleccionem la comanda *Click* de la part inferior. Apareixerà la finestra de la dreta on podrem seleccionar quina funció li volem donar de les disponibles. En el cas de la variable que estem programant, Permís de producció del IRB140, volem que en prémer el boto s'activi el permís i quedi activat fins que es torni a prémer. Per fer-ho seleccionem *Toggle Digital Tag*. Un cop seleccionada la funció a realitzar per l'objecte apareix la finestra de selecció de tag. Recordem que per seleccionar un tag aquest ha d'estar prèviament afegit en la base de dades.

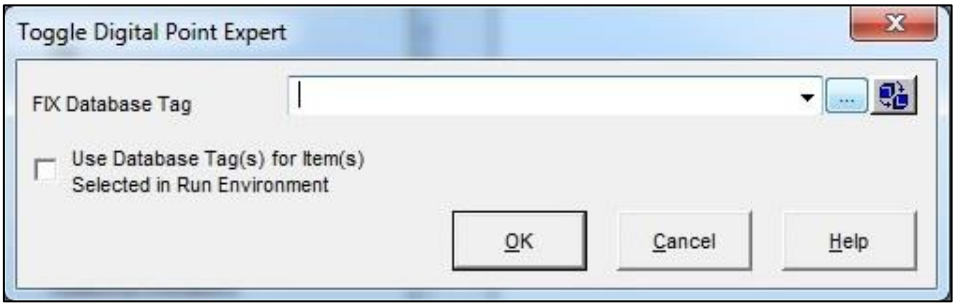


Figura 80: iFIX-Finestra de selecció del tag

Per a la selecció apareixerà una finestra on escollirem la variable que desitgem dins la llista disponible de tags. En la llista *Field Names* seleccionarem el tipus de representació *F_** corresponent als valors del tipus *float* o reals i concretament *F_CV* de *Current Value*. Els llistats de *A_** i *E_** corresponen a *Alarms* i *Events* respectivament. Només en el cas de les cadenes de caràcter s'ha utilitzat el format A_CV.

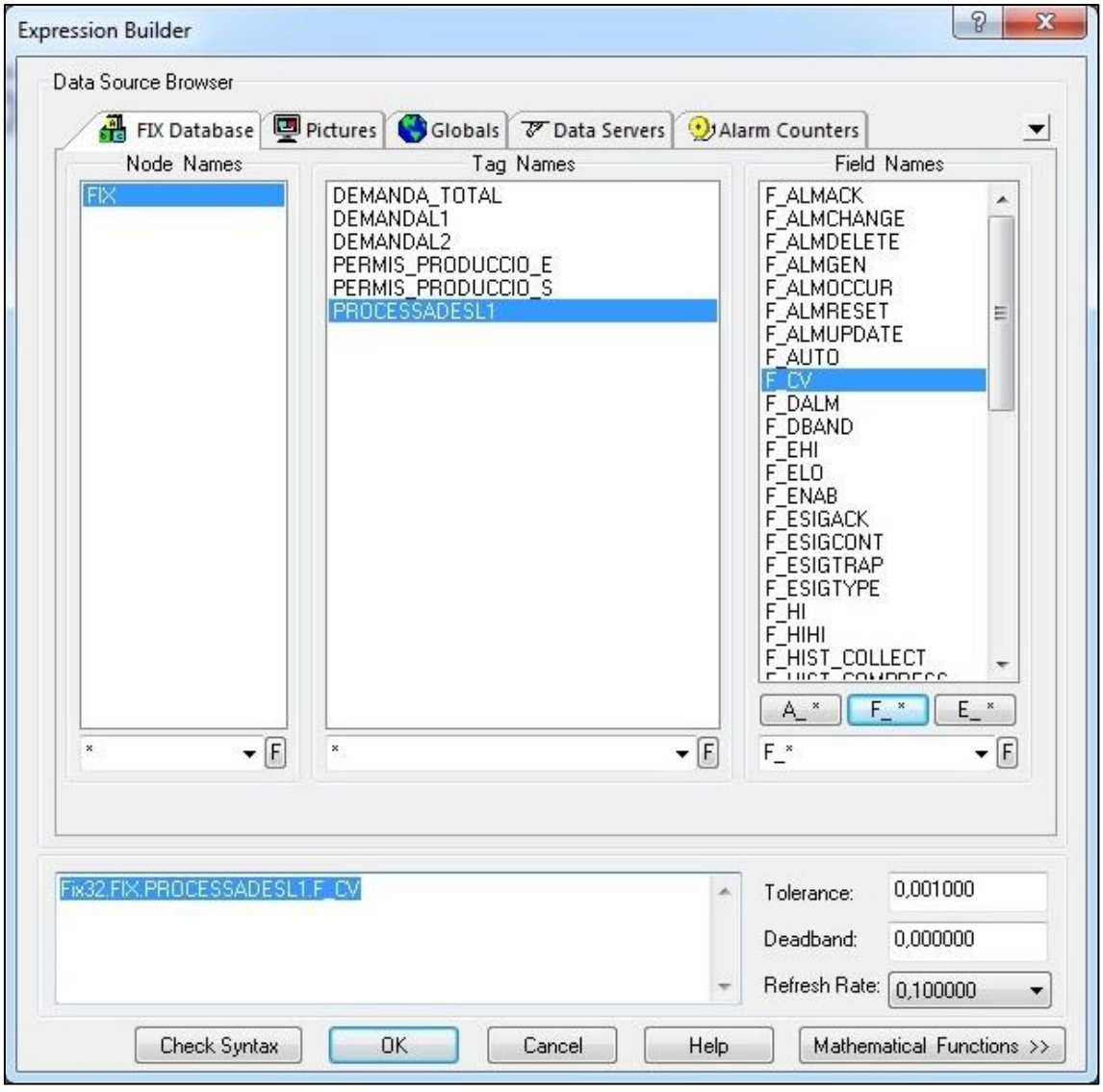


Figura 81: iFIX-Selecció del tag i tipus de representació

Si en comptes d'un objecte *Push Button* s'inclou un objecte del tipus *Datalink* el primer que apareix és la finestra de la Figura 82. En primer lloc seleccionarem el tag a monitoritzar i en segon lloc, configurarem el valor a mostrar. Si el tag és una cadena de caràcters podrem seleccionar la quantitat de línies amb les que volem que es mostri i quants caràcters per línia utilitzant l'opció *Formatting*. En cas que es tracti d'un valor analògic podrem modificar la quantitat de xifres enteres i decimals a mostrar.

Si l'objecte Datalink ha de permetre entrar a valors per l'usuari seleccionarem el tipus d'entrada com a *In-place* a traves de l'opció *Data Entry*.

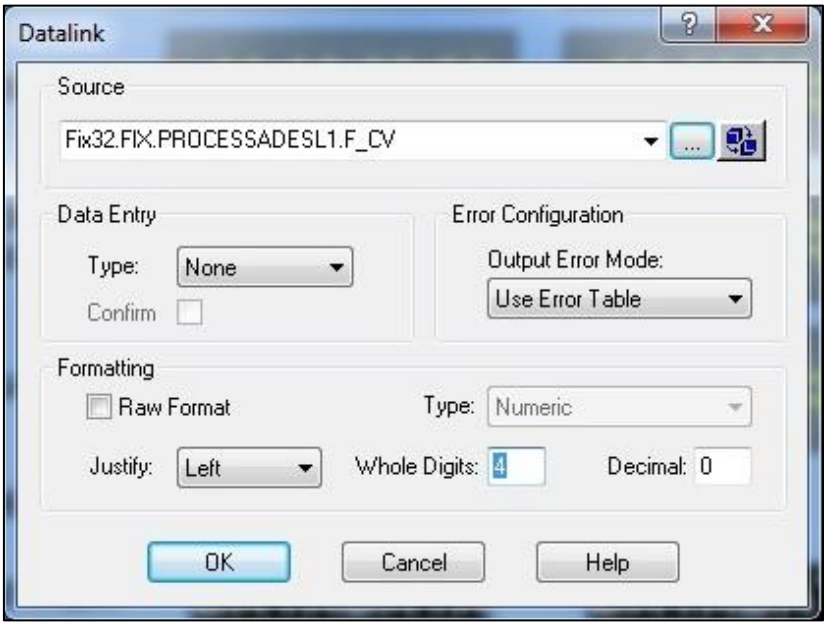


Figura 82: iFIX-Configuració dels objectes del tipus Datalink

A part de l'activació de bits o modificació de dades, els objectes inclosos també poden realitzar canvis en l'aspecte, la forma, la visualització i la posició entre altres. En la inserció del *Push Button* hem vist que aquestes característiques estaven bloquejades. En el cas del text "LINIA 2" de la següent imatge s'ha programat perquè la opció del fons del text *Background* faci una intermitència quant aquesta línia estigui en procés de paletització.

Per fer-ho s'ha condicionat a l'activació de la variable booleana *PaletitzemL2* que es mantindrà activa mentre s'estigui realitzant el procés de paletitzat. Quant aquest bit estigui actiu el fons del text canviarà alternativament de gris a groc.

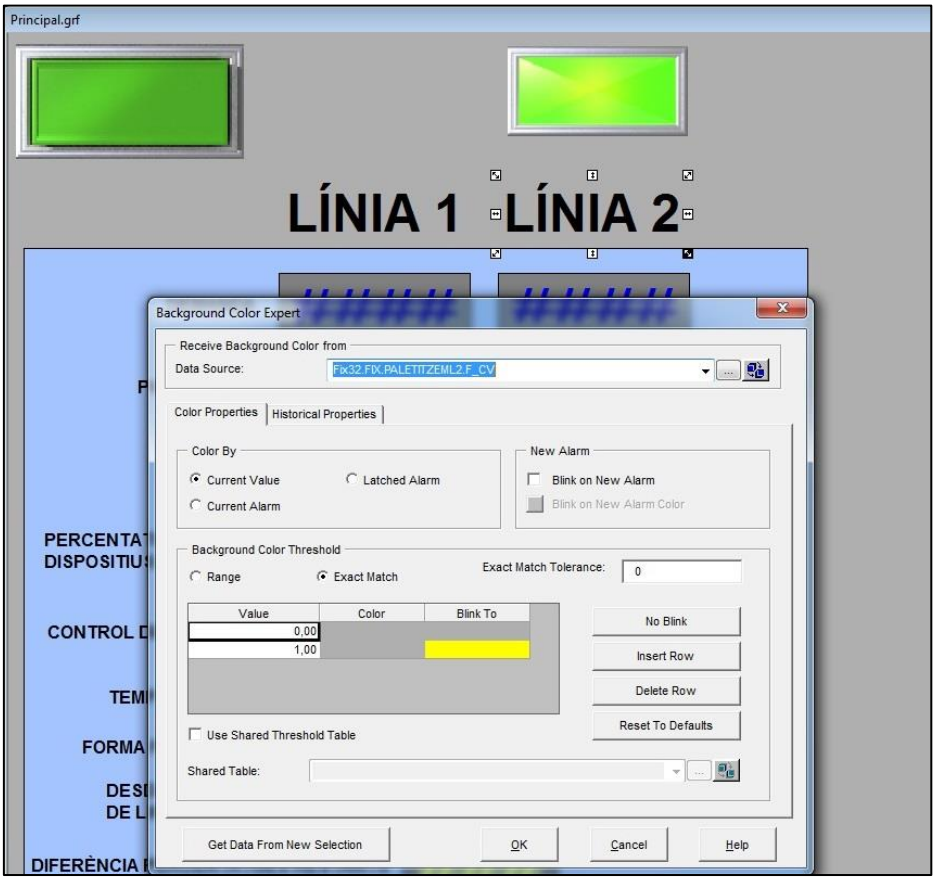


Figura 83: iFIX-Configuració d'una visualització intermitent

En la mateixa pantalla també s'han inclòs botons, indicadors i representacions reals d'objectes que podem trobar en les llibreries d'iFIX i que donen a l'aplicació una interfície més realista.

Per fer-ho ens situem en l'arbre de projecte i concretament a Dynamo Sets. Es desplegarà un llistat de grups que inclouen aquests tipus d'objectes. Si seleccionem per exemple un dels grups *Pilot* apareixerà una finestra amb tots els elements que inclou com podem veure en següent imatge. Per incloure'l en la pantalla només cal copiar-lo i seguir amb les indicacions anteriors com si es tractés d'un objecte o link.

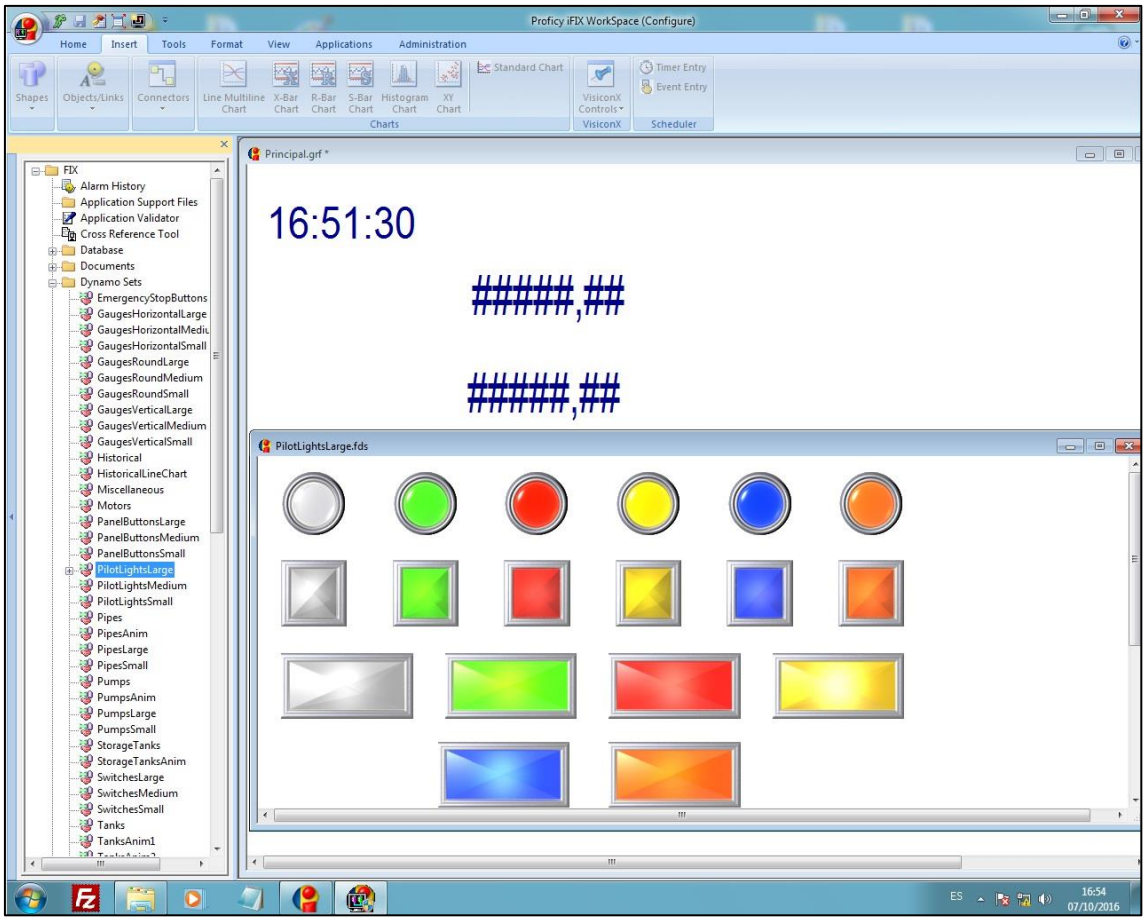


Figura 84: iFIX-Objectes de llibreries

Amb la combinació adient de tots els elements explicats s'ha pogut arribar a un sistema SCADA amb les característiques que es determinaran en el següent apartat.

En l'inici d'aquest apartat s'ha comentat la possibilitat de la programació mitjançant Visual Basic a través dels *Scripts*. En l'aplicació s'ha utilitzat per un costat per programar petites alternatives a les comandes ofertes en incloure un objecte o link. Per altra banda, també s'ha utilitzat a l'hora de voler executar unes instruccions quan succeeixin unes condicions concretes o per exemple cada vegada que una variable canviï de valor.

Aquests tipus d'instruccions s'implementen a través dels *Events* o objectes d'esdeveniment. Els esdeveniments no deixen cap rastre en la pantalla on s'executen, simplement és una part de programa que s'executa en segon pla. Aquest programa pot incloure una animació que es representi a la pantalla mitjançant l'activació d'un indicador lluminós per exemple.

A continuació es mostren els passos que s'han seguit per introduir a la demanda de smartphones de l'Arduino la demanda total calculada per la IRC5.

Per incloure un nou esdeveniment utilitza el llistat del desplegable Object/Links en el mateix lloc on hem seleccionat el *Push Buttons* i el *Datalink*.

En incloure l'esdeveniment apareix una finestra com la següent on seleccionarem en primer lloc la variable que l'activarà. A continuació es donarà un nom a l'Script i finalment es triarà el tipus d'esdeveniment, es a dir, cada quan s'executarà. En aquest cas serà cada vegada que la variable canviï.

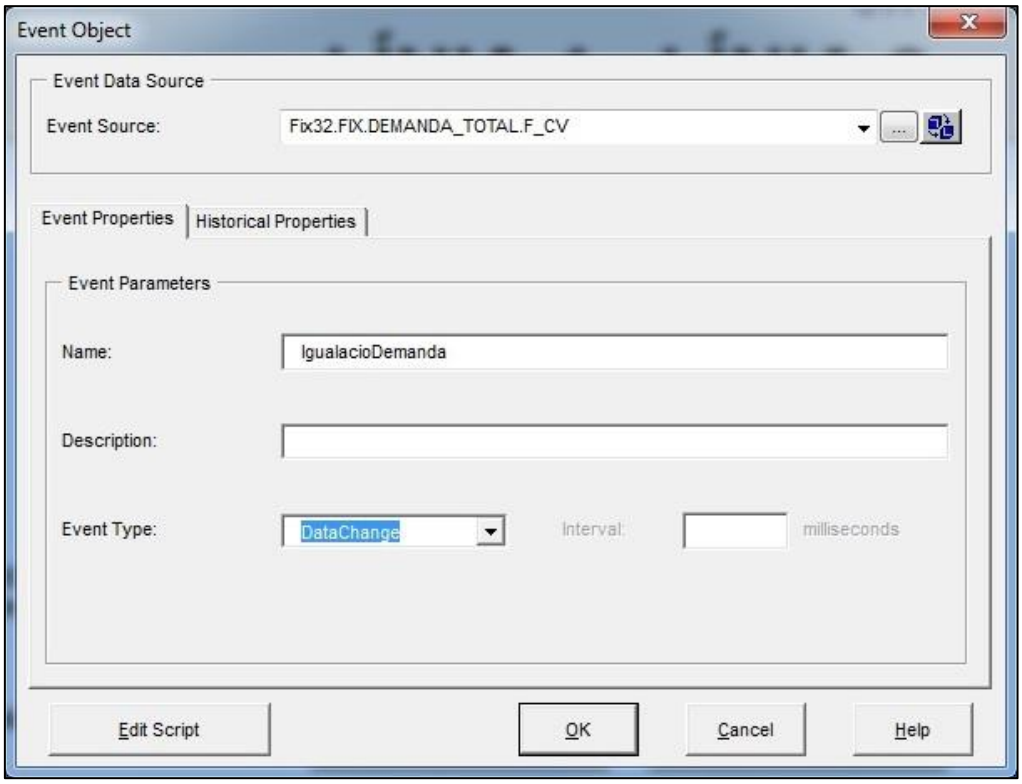


Figura 85: iFIX-Incloure objectes d'esdeveniment

Una vegada configurat l'esdeveniment s'obra una finestra amb l'script complert de la pantalla on es poden observar tots els elements inclosos en format Visual Basic i la funció que realitzen.

```
IgualacioDemanda                                                    DataChange
    OpenPicture "Password.grf", "", "", ""
    'WizardEnd


End Sub


Private Sub DesdoblarLinia_DataChange(ByVal DataValue As Variant, ByVal TimeStamp As Date, ByVal Transition As I
        'OpenPicture "Pregunta.grf", "", "", ""

End Sub


Private Sub IgualacioDemanda_DataChange(ByVal DataValue As Variant, ByVal TimeStamp As Date, ByVal Transition As
    Fix32.Fix.DEMANDA_GANTRY.F_CV = Fix32.Fix.DEMANDA_TOTAL.F_CV
End Sub

Private Sub PilotLightLgRctGreen_Edit()
    frmPilotAnim.InitializeDynamo PilotLightLgRctGreen
    frmPilotAnim.Show
End Sub


Private Sub PanelBtnLgRctGreen_Click()
    '***** Scripts Authoring Tool *****
    'The Comments below have been added automatically.
    'Any changes could cause adverse effects to the functionality
    'of the Script Authoring Experts.
    'WizardLast=Wizard1
    'WizardEditing=Wizard0
    'WizardName=ToggleDigitalPoint

    'Wizard1=ToggleDigitalPoint
    'Property1=Fix32.FIX.PERMIS_MARXA_GANTRY.F_CV
    'Property2=False
    'PropertyDescription=ToggleDigitalPoint: Property1=Database Tag, Property2=Select Tag in Run mode
    ToggleDigitalPoint "Fix32.FIX.PERMIS_MARXA_GANTRY.F_CV"
    'WizardEnd

End Sub

Private Sub PanelBtnLgRctGreen1_Click()
    '***** Scripts Authoring Tool *****
    'The Comments below have been added automatically.
```

Figura 86: iFIX-Script parcial de la pantalla principal

En l'script general de la pantalla s'afegirà una funció amb el nom que li hem donat a la configuració de l'esdeveniment amb uns paràmetres d'entrada que estan lligats al tag seleccionat. A aquesta funció se li escriuran les comandes que vulguem realitzar.

En el cas que ens ocupa copiem el valor de la variable DEMANDA_TOTAL de la IRC5 a la variable DEMANDA_GANTRY corresponent a l'Arduino.

```
Private Sub IgualacioDemanda_DataChange(ByVal DataValue As Variant, ByVal TimeStamp As Date,
    Fix32.Fix.DEMANDA_GANTRY.F_CV = Fix32.Fix.DEMANDA_TOTAL.F_CV
End Sub
```

Figura 87: iFIX-Exemple d'esdeveniment

Amb aquesta igualació, la suma de la demanda individual introduïda per a cada línia es converteix en la demanda de dispositius que ha d'empaquetar el Gantry per servir al paletitzat.

11 PROVES I RESULTATS

En aquest apartat es comentaran algunes de les modificacions més importants que s'han realitzat durant el desenvolupament del projecte. Més endavant es mostrarà el resultat de la integració total mitjançant l'aplicació SCADA.

11.1 PROCÉS DE PACKAGING

a) Interrupcions

Un dels canvis que s'ha realitzat per al treball final de grau ha estat la modificació d'una de les interrupcions que controla l'Arduino.

Una de les limitacions del controlador Arduino UNO és que solament permet dues interrupcions. Inicialment es considerava que hi havia 3 senyals critiques en el programa de packaging, la de marxa, la de moviment complert MC i l'indicador de final de carrera. Per a l'entrega del concurs s'havien programat les interrupcions de MC i final de carrera i s'havia deixat el polsador de marxa com a lectura d'un pin que es feia una vegada cada cicle de programa. El problema d'aquesta configuració era que en el programa hi havia diversos retards per obrir i tancar la pinça ja que no es disposava de cap senyal de retorn d'aquesta. Per tant, si es premia el polsador de Marxa/Paro durant algun d'aquests retards l'Arduino no responia ja que realment estava en un bucle de temps tancat i no gestionava cap més senyal.

Per aquest motiu, la parada de l'aplicació s'ha programat mitjançant una de les dues interrupcions que permet l'Arduino UNO.

Tot i que el polsador de marxa és secundari a l'aplicació final ja que s'inicia per l'SCADA s'ha programat com a interrupció. Per altra banda, com que la detecció de l'interruptor final de carrera només es pot produir en el moviment de pujada, s'ha permès aquest moviment sempre que el detector no estigui polsat.

b) Control del tercer eix

Primer de tot hem de recordar que s'ha utilitzat un servomotor de gir continu, i aquests no controlen posició sinó velocitat. Inicialment es va provar de programar-ho en funció de la distància a recórrer, el temps d'activació i el dimensionat de l'engranatge. Quan es va posar en pràctica, degut a les diferències entre els càlculs de disseny i la realitat es va veure que el sistema no era precís. Per altra banda, la programació en funció de tantes variables era realment complicada.

Per tant, es va optar per realitzar els càlculs a mà i a partir d'aquí ajustar aquests valors en funció del comportament de l'eix.

Aquests sistema funcionava força bé però igualment els paràmetres ajustats depenien de valors temporals i hi havia una diferència entre la pujada i la baixada de l'eix. Aquesta diferència es deu al fregament, al pes de l'element terminal i al moviment del suport del motor.

El cas és que en la pujada el motor havia de fer més força que en la baixada degut al pes de la pinça i al fregament de l'eix amb la seva guia. Això feia que el suport del motor flexionés una mica i en conjunt feia que amb els mateixos paràmetres que en la baixada la pujada fos més lenta i la distància recorreguda menor. Per solucionar aquestes diferències es va decidir separar d'alguna manera la pujada de la baixada. Al tenir una baixada força aconseguida i ràpida es va optar per utilitzar el detector de final de carrera en la posició superior i determinar la posició de baixada en funció dels càlculs obtinguts.

Per tant, amb els càlculs mostrats en l'apartat 10.1.5.1 s'ha parametritzat el moviment de baixada mentre que el de pujada s'ha fixat a una velocitat constant.

c) Integracio de la placa PCB

La realització del circuit electrònic que controla tots els components i que està governat per la placa Arduino es va implementar inicialment en una placa de proves anomenada també *protoboard* o breadboard. Aquest tipus de plaques són molt utilitzades en electrònica per a fer prototips de petits circuits. Com s'observa en la Figura 88 la *protoboard* dóna molta llibertat a la modificació ja que sol anar acompanyada amb cables que tenen un tipus de pin connector que encaixa perfectament en els forats de la placa.

Per altra banda, en el mateix muntatge es poden observar connexions molt poc segures en el cas del connector DB-15 de la interfície d'entrades i sortides. A part del connector, les connexions dels díodes LED i les resistències deixen molta superfície conductora lliure. Això pot causar que una senyal correcta doni corrent a un circuit proper si les potes d'algun dels elements es toquen entre elles.

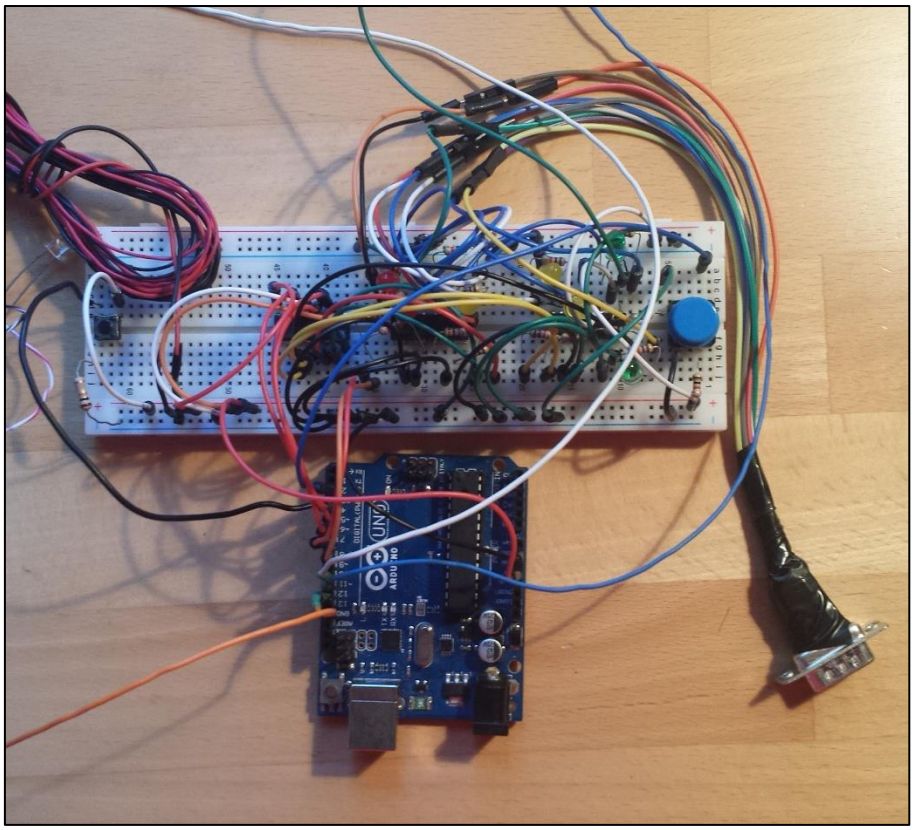


Figura 88: Implementació de l'electrònica en una protoboard

El canvi de funcionament de l'aplicació d'una placa a una altra s'ha notat des d'un primer moment evitant errors de contacte de components en la placa i en el connector de comunicació amb la controladora del EXCM.

El connector que es mostra en la imatge era mascle i es connectava directament a la controladora. El connector integrat a la PCB final també ho és però al no poder connectar-se directament pel fet d'estar fix a la placa s'ha necessitat fabricar manualment el cable allargador.

Per altra banda, els cables que inclou la placa són cables per treballar sobre la pròpia *breadboard* o molt a prop d'aquesta com és el cas de l'Arduino i per aquest motiu, aquests cables no solen superen els 10 o 15 cm de llargada. Això suposava un problema ja que també calia cablejar els servomotors de la pinça, l'eix vertical i l'interruptor de final de carrera. Per aquest motiu s'han fabricat els 6 cables que van de la pinça a la placa PCB final i s'han cobert amb un organitzador de cables.

d) L'OPC d'Arduino

En l'explicació del funcionament de l'OPC d'Arduino s'han argumentat els passos a seguir per poder enviar les comandes OPC correctament i optimitzar al màxim aquesta comunicació.

Tot i les modificacions en el software per millorar el comportament inicial de la comunicació de variables encara no s'ha pogut arribar a una comunicació completament continua i sense talls.

A continuació es mostra una imatge de l'explorador d'OPC **Matrikon OPC Explorer** que s'ha utilitzat per a les proves i que permet visualitzar en temps real les variables compartides en el protocol OPC.

En aquesta imatge es poden observar talls alternatius de la comunicació d'algunes variables.

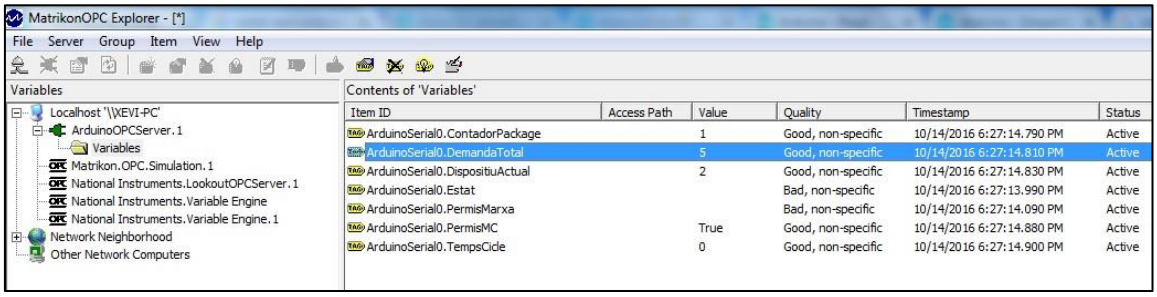


Figura 89: Comprovació de la comunicació OPC mitjançant l'explorador d'OPC Matrikon

Aquests petits talls de comunicació poden causar una pèrdua del control del packaging induint a un mal funcionament del sistema.

11.2 PROCÉS DE PALETITZAT

L'aplicació del procés de paletitzat ha tingut una evolució de millora constant amb l'addició de noves funcions i correcció d'errors fins a arribar a l'aplicació final.

L'evolució general que ha seguit el programa es pot resumir en:

1- Procés general de paletitzat.
2- Addició de funcions com desdoblar el paletitzat o la demanda de control de qualitat de les línies i del packaging del Gantry.
3- Paletitzat en equilibri per les dues línies.
4- Addició de la ventosa com a eina de treball.
5- Recull d'històrics de paletització en un arxiu extern.
6- Interrupció temporal de manteniment preventiu.
7- Càlcul dels temps de cicle i creació de la funció *ConvertirSegons*.
8- Separar els arxius d'històrics per a cada nova paletització.
9- Adaptació per a la comunicació de dades amb l'aplicació SCADA.

En el procés per adaptar el programa a l'aplicació SCADA s'han modificat les interrupcions que s'havien programat inicialment. Els canvis no han afectat en el codi de la interrupció pròpiament sinó a la crida d'aquesta interrupció.

Durant les proves, quan l'usuari es comunicava amb el robot utilitzant la TeachPendant, s'utilitzaven senyals digitals per activar les interrupcions. Aquestes senyals no es podien activar des del sistema SCADA i per això calia lligar la interrupció a una variable que el sistema de supervisió tingues accessibilitat. Per tant, el canvi ha estat la referència d'una senyal digital a una variable persistent de tipus booleana en tots els casos. D'aquesta manera, en produir un canvi en la variable des del sistema SCADA s'executaria la interrupció.

11.3 EL RESULTAT FINAL

11.3.1 SISTEMA DE PACKAGING

El muntatge de les parts del Gantry argumentades al llarg del treball han resultat en el muntatge del prototip que es mostra en la següent imatge.

Aquesta maqueta representa el packaging complert d'un dispositiu mòbil integrant tots els components especificats al llarg del treball. La comprovació del sistema demostra que el temps de cicle mitjà del procés és de 1 minut i 23 segons en un packaging continu.

El temps de cicle de treball s'ha vist molt augmentat per el moviment de l'eix vertical ja que és la part més critica i limitada de l'aplicació.

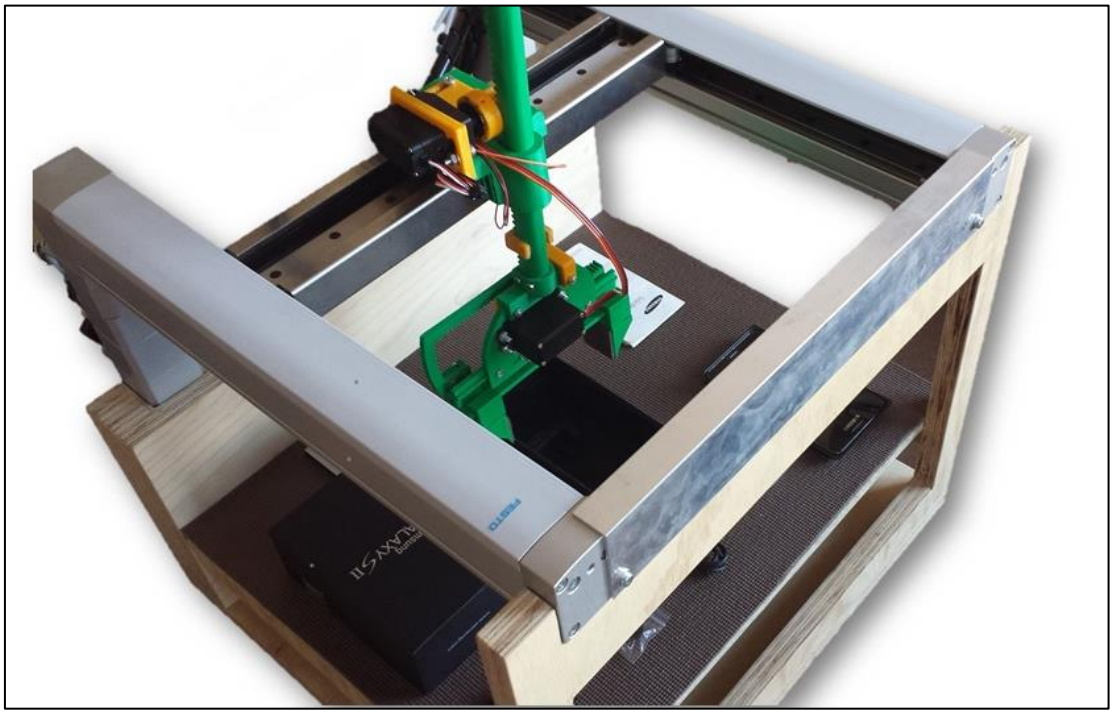


Figura 90: Muntatge final del prototip de packaging

Pel motiu abans explicat s'ha anul·lat el moviment de l'eix vertical per a les demostracions del sistema global permetent així un funcionament més dinàmic de l'aplicació de la cel·la de treball.

Anul·lant aquest moviment, el temps de cicle d'un procés d'embalatge és aproximadament de 19 segons.

11.3.2 SISTEMA SCADA

A continuació es mostraran tres exemples per veure el comportament global de l'aplicació de supervisió implementada amb iFIX.

En el primer inici de l'aplicació es mostrarà la pantalla tal i com es veu en la imatge següent, amb una demanda negativa en ambdós casos. El requadre blau de les dades del Gantry realitzarà una intermitència fins que s'iniciï el paletitzat mitjançant el polsador verd de la part superior esquerra. El mateix polsador verd permetrà parar l'aplicació en qualsevol moment sempre que s'hagi acabat el procés de paletitzat que s'estigui realitzant.

Sense haver indiciat la demanda es mostra la forma del paletitzat habitual de les dues línies i el nom de l'arxiu d'històrics on es guardaran les dades del procés a iniciar.

Figura 91: SCADA-Inici de l'aplicació

11.3.2.1 Exemple de funcionament del packaging

En el primer exemple veiem una demanda de 4 i 3 peces que equivalen a la demanda total del Gantry mostrada a la dreta de la imatge. En aquesta imatge s'està duent a terme el procés de packaging ja que es pot observar l'indicador lluminós superior dret encès i les lletres de PACKAGING fent una intermitència amb un fons groc.

La barra d'estat de procés indica en quin estat es troba el packaging actual. Per fer-ho, s'omple d'esquerra a dreta amb una barra de color groc.

En la mateixa imatge s'observen els talls parcials de comunicació amb la variable de temps de cicle de packaging en aquest cas.

En acabar les 7 peces de demanda total s'aturarà l'estació de packaging i s'esperarà fins que s'iniciï el procés de paletitzat.

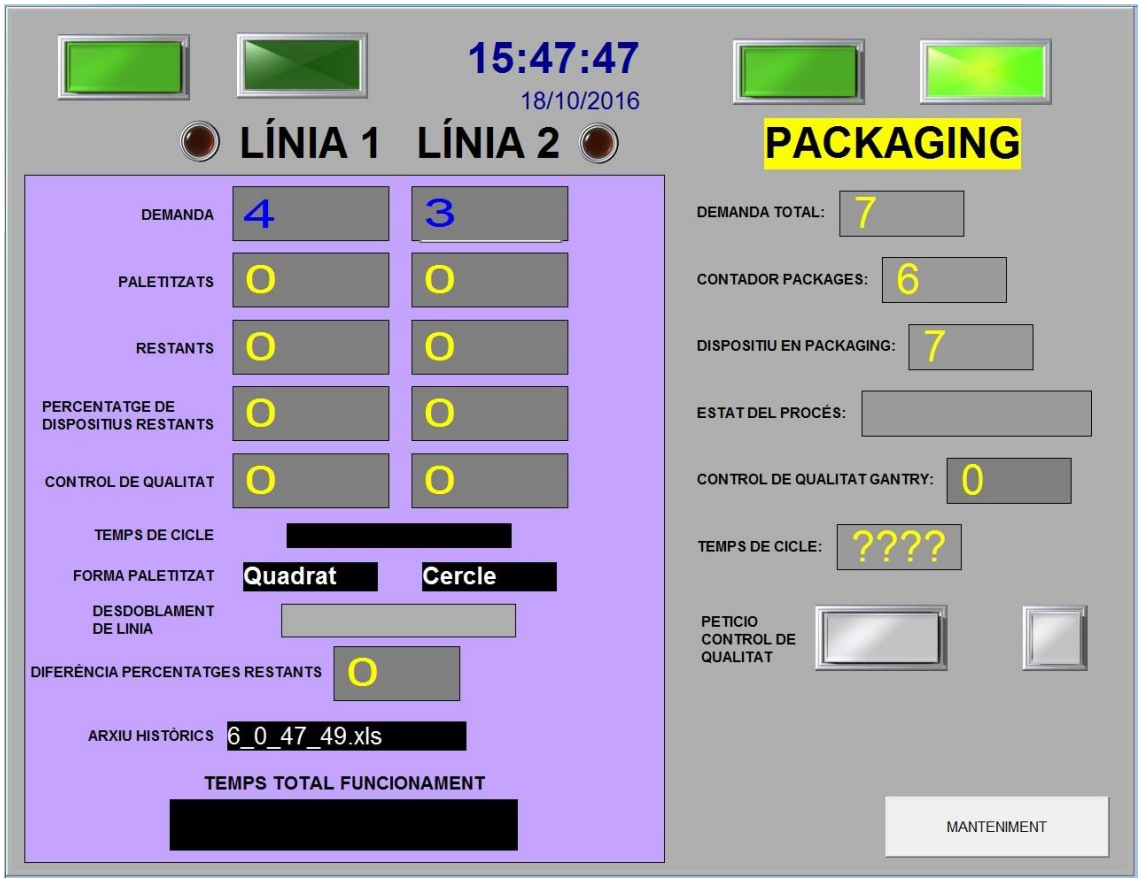


Figura 92: SCADA-Procés de packaging

11.3.2.2 Exemple de paletitzat habitual

Finalitzat el procés d'embalatge, s'ha iniciat el paletitzat de la demanda introduïda de forma alterna per les dues línies.

Igual que en el procés de packaging, també es mostra el funcionament de l'aplicació mitjançant l'indicador lluminós superior esquerra. En aquest cas, al existir dues línies de processat de mòbils la intermitència del fons groc variarà en funció de la línia que s'estigui paletitzant.

A part de les dades de procés, en la mateixa imatge es poden veure el temps de cicle de l'últim dispositiu i el temps total transcorregut des de l'inici de l'aplicació.

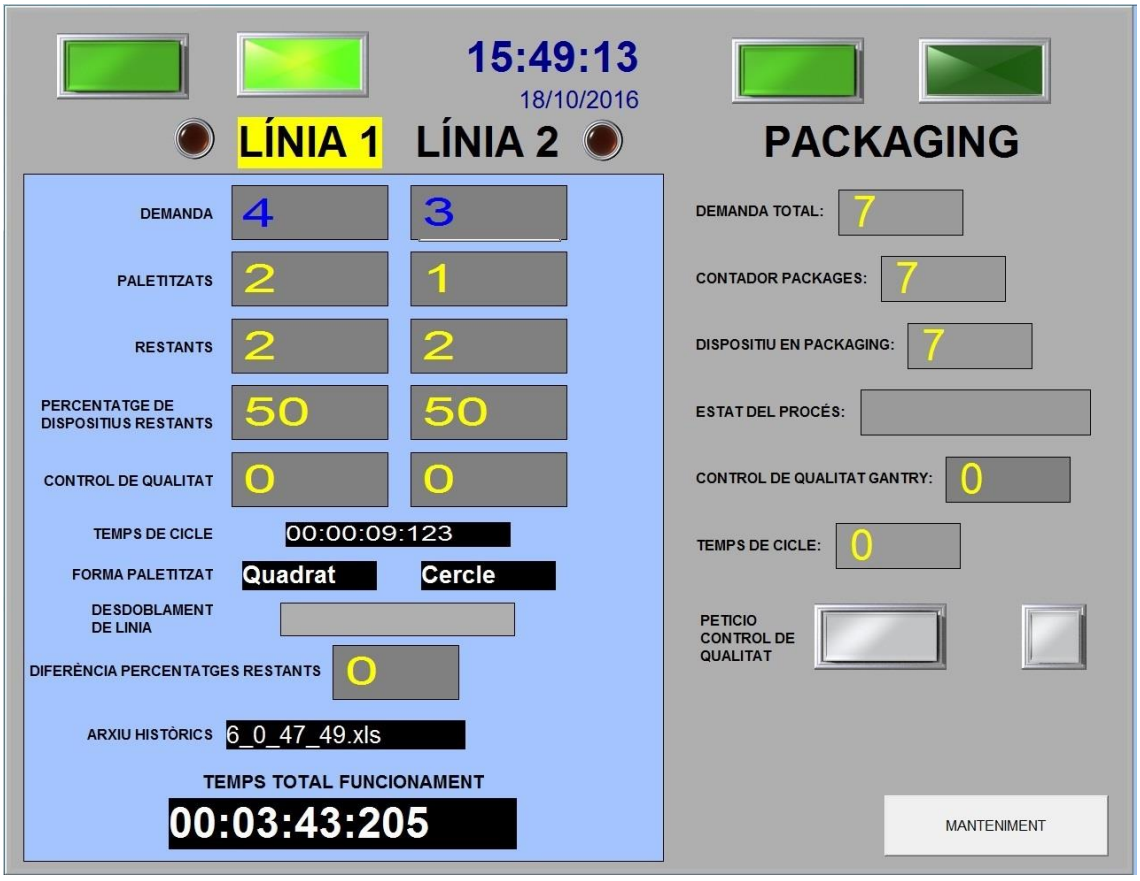


Figura 93: SCADA-Paletitzat habitual

11.3.2.3 Exemple amb desdoblament de línia

En aquest cas, la demanda introduïda és de 0 dispositius en la línia 1 i 5 dispositius en la línia 2. Per aquest motiu l'aplicació obra una finestra d'avisos demanant si es vol procedir a un desdoblament de la línia amb tota la demanda.

En aquest exemple s'ha escollit l'opció de desdoblar la producció de la línia 2 a la línia 1.

Figura 94: SCADA-Paletitzat amb desdoblament de línia 1

En escollir una opció es retorna a la pantalla principal i s'indica el desdoblament de la línia amb una intermitència del text LINIA 2 DESDOLADA. De la mateixa manera que es mostra que s'està produint un desdoblament es pot observar que la forma de paletitzat de la línia 1 ha canviat al format de la línia desdoblada.

Per altra banda en iniciar la paletització encara no s'han actualitzat les variables i per tant, les peces que resten per processar es troben totes en la línia 2. Per aquest motiu, la paletització d'una línia desdoblada sempre començarà per la línia on s'ha introduït tota la demanda inicialment.

Figura 95: SCADA-Paletitzat amb desdoblament de línia 2

Seguint amb l'exemple anterior, una vegada fet el primer cicle les variables s'actualitzen i les peces restants ja s'ha repartit entre les dues línies.

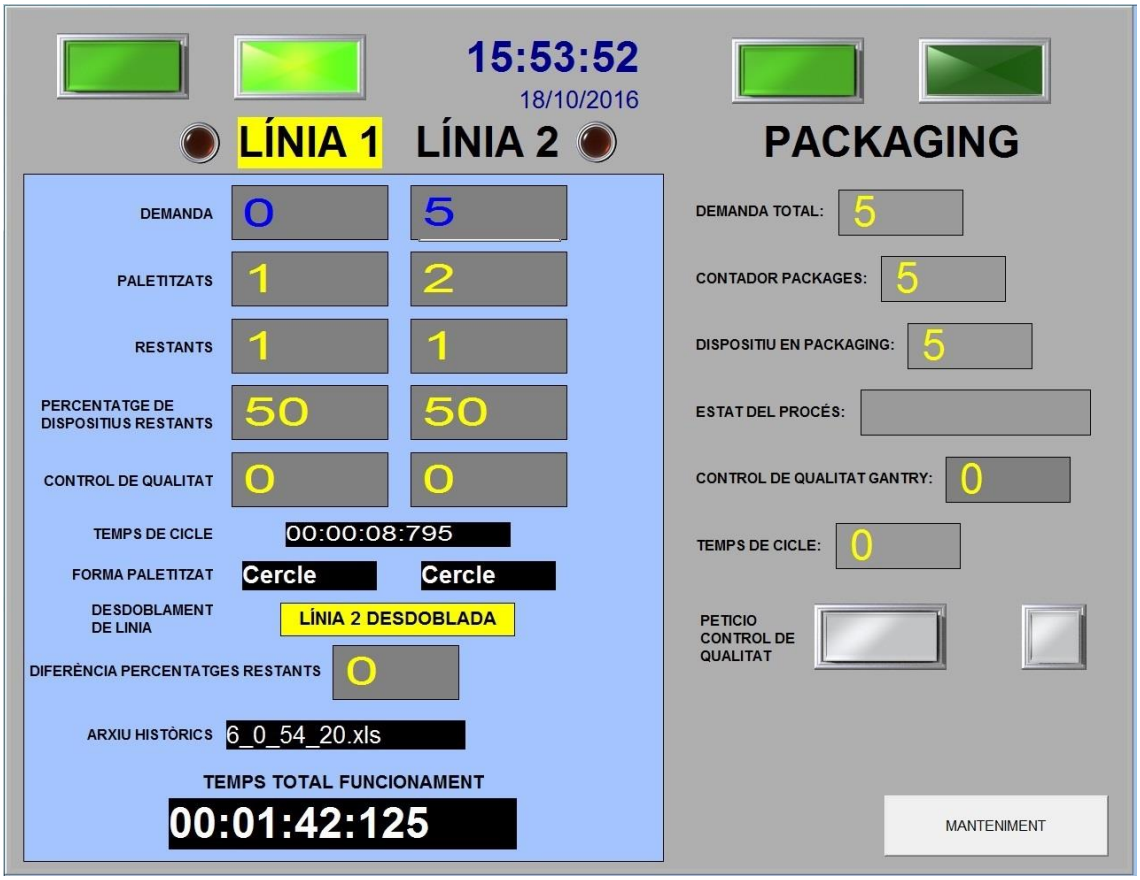


Figura 96: SCADA-Paletitzat amb desdoblament de línia 3

11.3.2.4 Exemple amb demanda de control de qualitat

En l'últim exemple es mostra la demanda d'un control de qualitat indicat per una intermitència en blau del requadre on es troben les quantitats de dispositius enviats a control de qualitat. Aquesta demanda de control de qualitat a les línies es realitza a través del polsador gran de la part inferior dreta de la pantalla. El polsador més petit faria la mateixa funció per al procés de packaging.

Recordem que aquesta petició de qualitat realitza una recollida per a cada línia i per aquest motiu, una vegada finalitzat el control a línia 1 es procedirà automàticament a la línia 2.

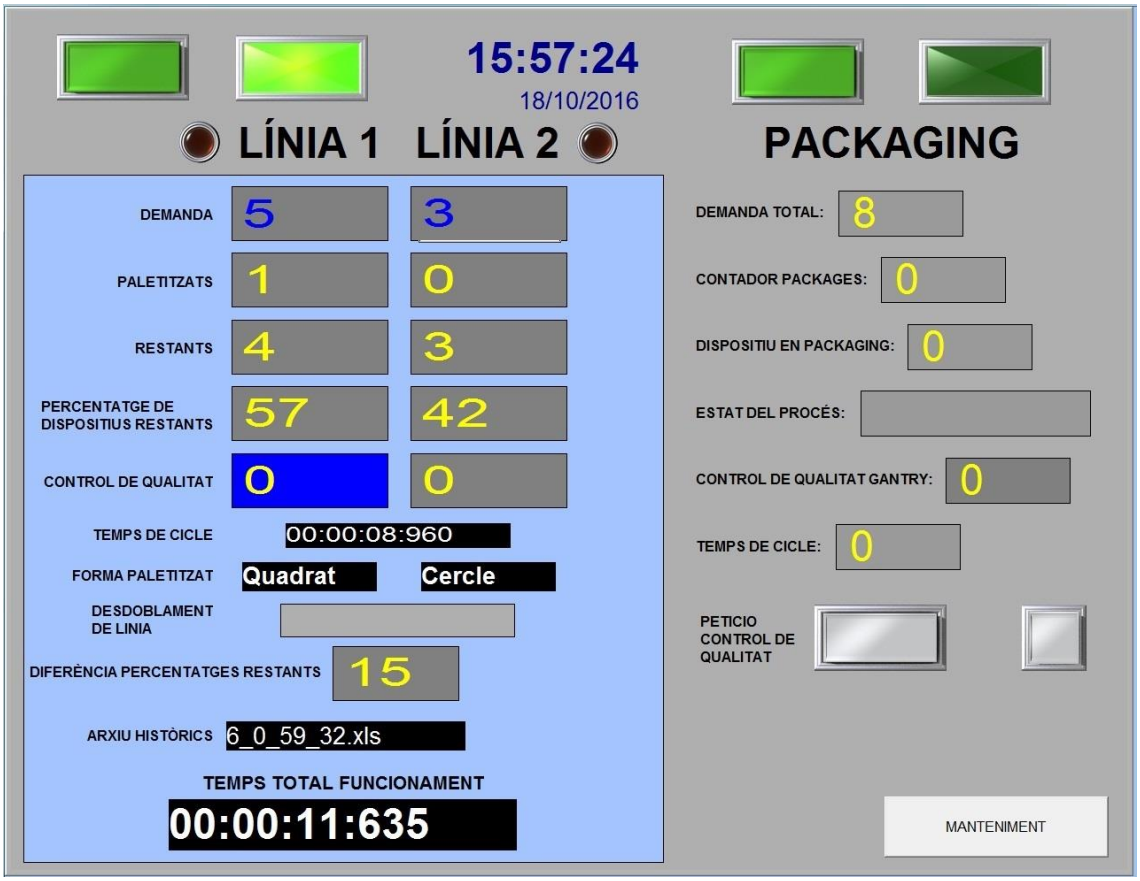


Figura 97: SCADA-Paletitzat amb petició de control de qualitat

En qualsevol moment es podrà accedir a la pantalla de manteniment des del polsador inferior dret de la pantalla principal. En aquesta finestra, es podran canviar paràmetres com el límit de percentatge acceptat o el temps de cicle de la interrupció de manteniment.

12 COCLUSIONS

En base als objectius fixats i mostrats en l'inici del treball:

- S'ha aconseguit l'adaptació un control superior a un dels robots.
- S'ha implementat la programació individual dels dos robots per la tasca a realitzar.
- S'ha dissenyat i implementat el sistema de comunicació entre robots.
- S'ha implementat una aplicació de supervisió i control del sistema global.
- S'ha comprovat i posat en marxa l'aplicació de la cel·la de treball.

Tot i haver complert amb els subobjectius de forma individual satisfactòriament, les expectatives inicials del funcionament global no han estat les esperades. Concretament no s'ha aconseguit una comunicació totalment fiable entre l'aplicació d'empaquetat i el sistema de supervisió.

13 TREBALLS FUTURS I MILLORES

A continuació es mostrarà un llistat de les millores que es podrien implementar en el sistema:

- Redissenyar l'element terminal amb un altre material que permeti augmentar la robustesa i la rapidesa del sistema de packaging.
- Incloure polsadors de parada d'emergència.
- Implementar les comandes OPC de l'Arduino de manera que s'evitin els talls en la comunicació amb el sistema de supervisió, per exemple utilitzant algun tipus d'interrupció.
- Establir una gestió d'errors i alarmes en les dues aplicacions que es pogués representar i interactuar des de l'aplicació SCADA.
- Establir un control d'usuaris en l'aplicació SCADA per accedir a la pantalla de manteniment.
- Visualització d'històrics des del sistema SCADA.

14 BIBLIOGRAFIA

MANUALS

{1} ABB – Technical reference manual. RAPID Instructions and data types

{2} ABB – ABB Flexible Automation AB. Guia de referència RAPID On-line 3.0

{3} FESTO – Instal·lació mecànica GDCE-EXCM-30-ES

{4} FESTO – Posta a punt GDCE-EXCM-SY-ES

PÀGINES WEB

{5} ABB (2016)

http://new.abb.com

{6} FESTO (2016)

https://www.festo.com/cms/es_es/index.htm

{7} Arduino/Genuino (2016)

https://www.arduino.cc/

{8} Software Tools for Makers (2016)

http://www.st4makers.com/

{9} PLC SCADA (2016)

http://plcescada.com/automacao/opc-server-para-arduino/

{10} Mad-science. Wonder how to. Fritzing

http://mad-science.wonderhowto.com/how-to/create-practically-anything-part-1-fritzing-circuit-boards-0135002/

{11} GE Fanuc. Proficy Software Documentation

http://help.geautomation.com/

15 ANNEX

1. PROGRAMA IRC5

A. MÒDUL VARIABLES

```
MODULE Variables
        CONST robtarget Home1:=[[512,0,710],[0.707,0,0.707,0],[-
1,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Home:=[[500,0,650],[0.707,0,0.707,0],[-
1,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Espera:=[[434.50,388.67,479.86],[0.0470704,0.360353,-
0.931544,0.0124621],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Recollida:=[[600,0,350],[0,0,1,0],[-
1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Recollida2:=[[600,0,450],[0,0,1,0],[-
1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget CQ:=[[-50,560,440],[0.00637311,-0.653642,0.756685,0.0118069],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget CQ2:=[[-50,560,540],[0.00637311,-0.653642,0.756685,0.0118069],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   VAR robtarget CQ_INV;
   VAR robtarget CQ2_INV;

   PERS wobjdata mirror:=[False,True,"",[[0,0,0],[0.707,0.707,0,0]],[[0,0,0],[1,0,0,0]]];
   PERS wobjdata PosPalet1:=[FALSE,TRUE,"",[[200,-500,0],[1,0,0,0]],[[140,80,0],[1,0,0,0]]];
   PERS wobjdata PosPalet2:=[FALSE,TRUE,"",[[200,350,0],[1,0,0,0]],[[110,110,0],[1,0,0,0]]];
   PERS robtarget Caixa:=[[-40,-70,500],[0,0,1,0],[-
1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   PERS tooldata Ventosa:=[TRUE,[[0,0,75],[0.707,0,0,-0.707]],[0.2,[0,0,50],[1,0,0,0],0,0,0]];

   VAR intnum Produccio;
   VAR intnum Produccio2;
   VAR intnum ControlABB;
   VAR intnum ControlFesto;
   VAR intnum MantenimentAcabat;

        VAR intnum Marxa;
        VAR intnum ControlQABB;
        VAR intnum ControlQF;
        VAR intnum Manteniment;
        VAR intnum SortirManteniment;

        PERS bool IniciMarxa;
        PERS bool IniciCQABB;
        PERS bool IniciCQF;
        PERS bool FiManteniment;

   PERS bool ProduccioContinua:=FALSE;
   PERS bool AplicacioIniciada:=FALSE;
   VAR iodev ArxiuDadesExcel;
   CONST speeddata vel:= v200 ;
   PERS bool NovaProduccio:=FALSE;
   PERS num PercentLiniesAcceptat:=10;
   PERS num Array {5}:=[0,0,0,0,0];
   PERS num PosicioArray:=1;
   PERS num ProcessadesL1:=0;
   PERS num ProcessadesL2:=0;
   PERS num TotalsL1:=0;
   PERS num TotalsL2:=0;
```

```
PERS num PeçaActual:=0;
PERS num NumPeçaF:=0;
PERS num DemandaL1:=-1;
PERS num DemandaL2:=-1;
PERS num DemandaTotal:=0;
PERS num RestantsTotals:=0;
PERS num RestantsL1:=0;
PERS num RestantsL2:=0;
PERS num PercentRestantsL1:=0;
PERS num PercentRestantsL2:=0;
PERS num DifPercentLinies:=0;
PERS num PecesCQL1:=0;
PERS num PecesCQL2:=0;
PERS num PecesCQF:=0;
PERS num PecesPaletL1:=0;
PERS num PecesPaletL2:=0;
PERS num LiniaDesdoblada:=0;
PERS string TipusPaletitzarL1:="Quadrat";
PERS string TipusPaletitzarL2:="Cercle";
PERS bool CanviPaletL1:=FALSE;
PERS bool CanviPaletL2:=FALSE;
PERS bool PermisProduccio:=FALSE;
PERS bool PeticioCQABB:=FALSE;
PERS bool PeticioCQF:=FALSE;
PERS bool PaletitzemL1:=FALSE;
PERS bool PaletitzemL2:=FALSE;
PERS bool ControlQualitatL1Marxa:=FALSE;
PERS bool ControlQualitatL2Marxa:=FALSE;
PERS bool ControlQualitatFMarxa:=FALSE;
VAR bool Marca:=FALSE;
VAR clock TemporitzadorTotal;
VAR num TempsTotal:=30.376;
PERS string TempsTotalString:="00:03:29:839";
VAR clock TemporitzadorCicle;
VAR num TempsCicle:=2.304;
PERS string TempsCicleString:="00:00:08:319";
PERS string ArxiuActual:="4_8_45_58.xls";
VAR string ArxiuActual1:="15";
VAR string ArxiuActual2:="20";
VAR string ArxiuActual3:="7";
VAR string ArxiuActual4:="7";
VAR num ArxiuActualNum1:=15;
VAR num ArxiuActualNum2:=20;
VAR num ArxiuActualNum3:=7;
VAR num ArxiuActualNum4:=7;

PERS num TempsManteniment:=3000;
VAR num Hores:=0;
VAR num Minuts:=0;
VAR num Segons:=30;
VAR num Decimes:=375;
VAR string Horess:="";
VAR string Minutss:="";
VAR string Segonss:="";
VAR string Decimess:="";
CONST string SeparaHora:=":";

PERS bool MarxaManteniment:=FALSE;
PERS bool FinalManteniment:=FALSE;
```

```
        PERS bool DemandaL1Nula:=FALSE;
        PERS bool DemandaL2Nula:=FALSE;
        PERS num DesdoblarProd:=0;
    PERS bool PeticioNoValida:=FALSE;

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!    RESET VARIABLES    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC ResetVariables()
    AplicacioIniciada:=FALSE;
    Marca:=FALSE;
    Array:=[0,0,0,0,0];
    PercentLiniesAcceptat:=10;
    PosicioArray:=1;
    ProcessadesL1:=0;
    ProcessadesL2:=0;
    TotalsL1:=0;
    TotalsL2:=0;
    PeçaActual:=0;
    DemandaL1:=-1;
    DemandaL2:=-1;
    DemandaTotal:=0;
    RestantsTotals:=0;
    RestantsL1:=0;
    RestantsL2:=0;
    PercentRestantsL1:=0;
    PercentRestantsL2:=0;
    DifPercentLinies:=0;
    PecesCQL1:=0;
    PecesCQL2:=0;
    PecesCQF:=0;
    PecesPaletL1:=0;
    PecesPaletL2:=0;
    LiniaDesdoblada:=0;
    TipusPaletitzarL1:="Quadrat";
    TipusPaletitzarL2:="Cercle";
    CanviPaletL1:=FALSE;
    CanviPaletL2:=FALSE;
    PermisProduccio:=FALSE;
    PeticioCQABB:=FALSE;
    PeticioCQF:=FALSE;
    PaletitzemL1:=FALSE;
    PaletitzemL2:=FALSE;
    Reset Do10o1;
    Reset Do10o2;
    Reset Do10o3;
    Reset Do10o4;

    MarxaManteniment:=FALSE;
    FinalManteniment:=FALSE;

    DemandaL1Nula:=False;
        DemandaL2Nula:=False;
        DesdoblarProd:=0;
    PeticioNoValida:=FALSE;
ENDPROC

ENDMODULE
```

B. MÒDUL PRINCIPAL

```
MODULE Principal
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   PROGRAMA PRINCIPAL   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC MAIN()
        CONNECT Marxa WITH TrapMarxa;
  IPers IniciMarxa,Marxa;
        CONNECT ControlQABB WITH TrapControlQABB;
  IPers IniciCQABB,ControlQABB;
        CONNECT ControlQF WITH TrapControlQF;
  IPers IniciCQF,ControlQF;
  CONNECT Manteniment WITH TrapManteniment;
  ITimer TempsManteniment, Manteniment;
        CONNECT SortirManteniment WITH TrapSortirManteniment;
  IPers FiManteniment, SortirManteniment;

  MOVEJ Home1,vel,z30, tool0;

  Inici:
  TPWrite "Esperant inici de produccio INT3";

  ArxiuActualNum1:=GetTime(\WDay);
  ArxiuActualNum2:=GetTime(\Hour);
  ArxiuActualNum3:=GetTime(\Min);
  ArxiuActualNum4:=GetTime(\Sec);
  ArxiuActual1:=NumToStr(ArxiuActualNum1,0);
  ArxiuActual2:=NumToStr(ArxiuActualNum2,0);
  ArxiuActual3:=NumToStr(ArxiuActualNum3,0);
  ArxiuActual4:=NumToStr(ArxiuActualNum4,0);
  ArxiuActual:=ArxiuActual1+"_"+ArxiuActual2+"_"+ArxiuActual3+"_"+ArxiuActual4+".xls";

  Open "C:/Users/Xevi/Dropbox/Documents/UPC" \File:=ArxiuActual, ArxiuDadesExcel;
  !!Open "/hd0a/OBELIX2/HOME/PFCs/Xevi" \File:=ArxiuActual, ArxiuDadesExcel;
  Write ArxiuDadesExcel, "Hora;Número de dispositiu;Procés;Tipus de paletitzat;Temps de procés;";

  CQ_INV:=MirPos(CQ,mirror);
  CQ2_INV:=MirPos(CQ2,mirror);

  ResetVariables;
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
  WHILE (true) DO
    IF PermisProduccio=FALSE AND AplicacioIniciada=FALSE THEN
      NovaProduccio:=FALSE;
    ENDIF
    IF PermisProduccio=FALSE AND (ProcessadesL1+ProcessadesL2)<DemandaTotal and
AplicacioIniciada=TRUE THEN
      WaitUntil PermisProduccio;
    ENDIF
    IF PermisProduccio=FALSE and MarxaManteniment=FALSE AND (NovaProduccio=TRUE OR
AplicacioIniciada=FALSE) THEN
              WaitUntil DemandaL1<>-1;
              WaitUntil DemandaL2<>-1;
              NovaProduccio:=FALSE;
      ClkReset TemporitzadorTotal;
                ClkStart TemporitzadorTotal;
      Write ArxiuDadesExcel, " ";
      Write ArxiuDadesExcel, "NOVA PALETITZACIÓ";
      Write ArxiuDadesExcel, "DEMANDA LÍNIA 1: "\Num:=DemandaL1;
      Write ArxiuDadesExcel, "DEMANDA LÍNIA 2: "\Num:=DemandaL2;
      DemandaTotal:=DemandaL1+DemandaL2;
```

```
        IF DemandaTotal<=0 OR DemandaL1<0 OR DemandaL2<0 THEN
          TPWrite "!!! PETICIÓ NO VÀLIDA !!!";
          Write ArxiuDadesExcel, "!!! PETICIÓ NO VÀLIDA !!!";
          PeticioNoValida:=TRUE;
          MOVEJ Home,vel,z30,Ventosa;
          Marca:=FALSE;
          Close ArxiuDadesExcel;
          WaitTime(3);
          GOTO Inici;
        ENDIF
        IF DemandaTotal>0 AND (DemandaL1=0 OR DemandaL2=0) THEN
          IF DemandaL1=0 THEN
            TPWrite"La demanda per la línia 1 és 0.";
                              DemandaL1Nula:=True;
          ELSEIF DemandaL2=0 THEN
            TPWrite"La demanda per la línia 2 és 0.";
                              DemandaL2Nula:=True;
          ENDIF
          IF DemandaTotal>1 THEN
                              WaitUntil DesdoblarProd>0;
            IF DesdoblarProd=1 AND DemandaL1>DemandaL2 THEN
              LiniaDesdoblada:=1;
              TipusPaletitzarL1:="Quadrat";
              TipusPaletitzarL2:="Quadrat";
            ELSEIF DesdoblarProd=1 AND DemandaL2>DemandaL1 THEN
              LiniaDesdoblada:=2;
              TipusPaletitzarL1:="Cercle";
              TipusPaletitzarL2:="Cercle";
            ENDIF
          ENDIF
        ENDIF
        TPWrite"Esperant permís de producció...";
        WHILE PermisProduccio=FALSE DO
          MOVEJ Home,vel,z30, tool0;
        ENDWHILE
        AplicacioIniciada:=TRUE;
                  NovaProduccio:=FALSE;
      ENDIF

      ActualitzaVariables;

      WHILE PermisProduccio=TRUE and MarxaManteniment=FALSE DO
        WHILE PermisProduccio=TRUE and MarxaManteniment=FALSE and
DifPercentLinies>PercentLiniesAcceptat DO
          WHILE PermisProduccio=TRUE and MarxaManteniment=FALSE and
DifPercentLinies>PercentLiniesAcceptat and DesdoblarProd=1 AND (RestantsL1>0 OR RestantsL2>0)
DO
            IF RestantsL1>RestantsL2 THEN
              IF (DemandaL1 MOD 2=0) AND (ProcessadesL1+ProcessadesL2=0) THEN
                DemandaL1:=DemandaTotal/2;
                DemandaL2:=DemandaTotal/2;
              ELSEIF (ProcessadesL1+ProcessadesL2=0) THEN
                DemandaL2:=(DemandaTotal/2)-0.5;
                DemandaL1:=(DemandaTotal/2)+0.5;
              ENDIF
            ELSE
              IF (DemandaL2 MOD 2=0) AND (ProcessadesL1+ProcessadesL2=0) THEN
                DemandaL2:=DemandaTotal/2;
                DemandaL1:=DemandaTotal/2;
              ELSEIF (ProcessadesL1+ProcessadesL2=0) THEN
```

```
                DemandaL1:=(DemandaTotal/2)-0.5;
                DemandaL2:=(DemandaTotal/2)+0.5;
              ENDIF
            ENDIF
            PercentLiniesAcceptat:=100;
            IF RestantsL1<>0 THEN
              PaletitzarL1;
            ENDIF
            IF RestantsL2<>0 THEN
              PaletitzarL2;
            ENDIF
          ENDWHILE
          IF (DesdoblarProd=0 OR DesdoblarProd=2) AND PercentRestantsL1>PercentRestantsL2
THEN
            PaletitzarL1;
          ENDIF
          IF (DesdoblarProd=0 OR DesdoblarProd=2) AND PercentRestantsL2>PercentRestantsL1
THEN
            PaletitzarL2;
          ENDIF
        ENDWHILE
        WHILE PermisProduccio=TRUE and MarxaManteniment=FALSE and
DifPercentLinies<=PercentLiniesAcceptat and (RestantsL1>0 OR RestantsL2>0) DO
          IF RestantsL1<>0 THEN
            PaletitzarL1;
          ENDIF
          IF RestantsL2<>0 THEN
            PaletitzarL2;
          ENDIF
        ENDWHILE
        IF RestantsTotals<=0 THEN
          TPErase;
          TPWrite"Paletització acabada";
          MOVEJ Home1,vel,z30, tool0;
          TPWrite"Esperant nova paletització...";
          ClkStop TemporitzadorTotal;
          TempsTotal:= ClkRead (TemporitzadorTotal);
          TempsTotalString:=ConvertirSegons(TempsTotal);
          Write ArxiuDadesExcel, " ";
          Write ArxiuDadesExcel, "Temps total de funcionament del robot; "\NoNewLine;
          Write ArxiuDadesExcel, TempsTotalString;
          Close ArxiuDadesExcel;
          ResetVariables;
          GOTO Inici;
        ENDIF
      ENDWHILE
      IF MarxaManteniment=TRUE THEN
        MOVEJ Home,vel,z10,tool0;
        TPErase;
        TPWrite"Estació en manteniment";
        WaitUntil FinalManteniment=TRUE;
      ENDIF
    ENDWHILE
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
TRAP TrapMarxa
        PermisProduccio:=NOT PermisProduccio;
ENDTRAP

TRAP TrapControlQABB
```

```
        PeticioCQABB:=TRUE;
ENDTRAP

TRAP TrapControlQF
        PeticioCQF:=TRUE;
ENDTRAP

TRAP TrapManteniment
  MarxaManteniment:=TRUE;
  PermisProduccio:=FALSE;
  FinalManteniment:=FALSE;
ENDTRAP

TRAP TrapSortirManteniment
        MarxaManteniment:=FALSE;
  PermisProduccio:=TRUE;
  FinalManteniment:=TRUE;
ENDTRAP
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ENDMODULE
```

C. MÒDUL PALETITZAT

```
MODULE Paletitzar

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   CONTROL QUALITAT PRINCIPAL   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC ControlQualitat()
   IF PeticioCQABB=TRUE AND PeçaActual<NumPeçaF AND RestantsTotals>0 THEN
      ControlQualitatL1;
      ControlQualitatL2;
      PeticioCQABB:=FALSE;
      IniciCQABB:=FALSE;
   ENDIF

   IF PeticioCQF=TRUE AND RestantsTotals>0 THEN
      Array{PosicioArray}:=NumPeçaF;
      INCR PosicioArray;
      IF PosicioArray=5 THEN
         PosicioArray:=1;
      ENDIF
      PeticioCQF:=FALSE;
      IniciCQF:=FALSE;
   ENDIF
   FOR i FROM 1 TO 5 DO
      IF Array{i}=TotalsL1+TotalsL2 THEN
         ControlQualitatF;
         Array{i}:=0;
      ENDIF
   ENDFOR
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   CONTROL QUALITAT LINIA 1   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC ControlQualitatL1()
   ControlQualitatL1Marxa:=TRUE;
   ClkReset TemporitzadorCicle;
   ClkStart TemporitzadorCicle;
   MoveJ Recollida2, vel, z30, Ventosa;
   MoveL Recollida, vel, fine, Ventosa;
   MoveL Recollida2, vel, z30, Ventosa;
   MOVEJ CQ2_INV,vel,z10,Ventosa;
   MOVEL CQ_INV,vel,z10,Ventosa;
   MOVEL CQ2_INV,vel,z10,Ventosa;
   INCR PecesCQL1;
   ActualitzaVariables;
   Dades;
   MOVEJ Home,vel,z30,Ventosa;
   ControlQualitatL1Marxa:=FALSE;
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   CONTROL QUALITAT LINIA 2   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC ControlQualitatL2()
   ControlQualitatL2Marxa:=TRUE;
   ClkReset TemporitzadorCicle;
   ClkStart TemporitzadorCicle;
   MoveJ Recollida2, vel, z30, Ventosa;
   MoveL Recollida, vel, fine, Ventosa;
   MoveL Recollida2, vel, z30, Ventosa;
   MOVEJ CQ2,vel,z10,Ventosa;
   MOVEL CQ,vel,z10,Ventosa;
   MOVEL CQ2,vel,z10,Ventosa;
```

```
    INCR PecesCQL2;
    ActualitzaVariables;
    Dades;
    MOVEJ Home,vel,z30,Ventosa;
    ControlQualitatL2Marxa:=FALSE;
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!    CONTROL QUALITAT FESTO    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC ControlQualitatF()
    ControlQualitatFMarxa:=TRUE;
    ClkReset TemporitzadorCicle;
    ClkStart TemporitzadorCicle;
    MoveJ Recollida2, vel, z30, Ventosa;
    MoveL Recollida, vel, fine, Ventosa;
    MoveL Recollida2, vel, z30, Ventosa;
    MOVEJ CQ2_INV,vel,z10,Ventosa;
    MOVEL CQ_INV,vel,z10,Ventosa;
    MOVEL CQ2_INV,vel,z10,Ventosa;
    INCR PecesCQF;
    ActualitzaVariables;
    Dades;
    MOVEJ Home,vel,z30,Ventosa;
    ControlQualitatFMarxa:=FALSE;
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!    PALETITZAR LINIA 1    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC PaletitzarL1()
    ControlQualitat;
    PaletitzemL1:=TRUE;
    ClkReset TemporitzadorCicle;
    ClkStart TemporitzadorCicle;
    MoveJ Recollida2, vel, z30, Ventosa;
    MoveL Recollida, vel, fine, Ventosa;
    MoveL Recollida2, vel, z30, Ventosa;
    PecesPaletL1:=TipusPalet (PecesPaletL1, TipusPaletitzarL1);
    ActualitzaVariables;
    Dades;
    MOVEJ Home,vel,z30,Ventosa;
    PaletitzemL1:=FALSE;
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!    PALETITZAR LINIA 2    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC PaletitzarL2()
    ControlQualitat;
    PaletitzemL2:=TRUE;
    ClkReset TemporitzadorCicle;
    ClkStart TemporitzadorCicle;
    MoveJ Recollida2, vel, z30,Ventosa;
    MoveL Recollida, vel, fine,Ventosa;
    MoveL Recollida2, vel, z30,Ventosa;
    PecesPaletL2:=TipusPalet (PecesPaletL2, TipusPaletitzarL2);
    ActualitzaVariables;
    Dades;
    MOVEJ Home,vel,z30,Ventosa;
    PaletitzemL2:=FALSE;
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!    PALETITZACIÓ SEGONS TIPUS    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
FUNC num TipusPalet (num PecesPalet, string TipusPalet)
   VAR num DistanciaX:=0;
   VAR num DistanciaY:=0;
   VAR num DistanciaZ:=0;
   VAR num Rotacio:=0;
   Caixa.rot:=[0,0,1,0];

   Incr PecesPalet;
   IF PecesPalet>16 THEN
      PecesPalet:=1;
      CanviPaletL1:=FALSE;
      CanviPaletL2:=FALSE;
   ENDIF

   IF PecesPalet<=4 THEN
      DistanciaZ:=80;
   ELSEIF PecesPalet>4 AND PecesPalet<=8 THEN
      DistanciaZ:=80*2;
   ELSEIF PecesPalet>8 AND PecesPalet<=12 THEN
      DistanciaZ:=80*3;
   ELSEIF PecesPalet>12 AND PecesPalet<=16 THEN
      DistanciaZ:=80*4;
   ENDIF

   IF PecesPalet=1 OR PecesPalet=5 OR PecesPalet=9 OR PecesPalet=13 THEN
      IF (TipusPalet="Quadrat") THEN
         DistanciaX:=-70;
         DistanciaY:=-40;
      ELSEIF(TipusPalet="Cercle") THEN
         DistanciaX:=-40;
         DistanciaY:=-70;
      ENDIF
   ELSEIF PecesPalet=2 OR PecesPalet=6 OR PecesPalet=10 OR PecesPalet=14 THEN
      IF (TipusPalet="Quadrat") THEN
         DistanciaX:=70;
         DistanciaY:=-40;
      ELSEIF(TipusPalet="Cercle") THEN
         DistanciaX:=70;
         DistanciaY:=-40;
         Caixa.rot:=[0,0.707,0.707,0];
      ENDIF
   ELSEIF PecesPalet=3 OR PecesPalet=7 OR PecesPalet=11 OR PecesPalet=15 THEN
      IF (TipusPalet="Quadrat") THEN
         DistanciaX:=70;
         DistanciaY:=40;
      ELSEIF(TipusPalet="Cercle") THEN
         DistanciaX:=40;
         DistanciaY:=70;
      ENDIF
   ELSEIF PecesPalet=4 OR PecesPalet=8 OR PecesPalet=12 OR PecesPalet=16 THEN
      IF (TipusPalet="Quadrat") THEN
         DistanciaX:=-70;
         DistanciaY:=40;
      ELSEIF(TipusPalet="Cercle") THEN
         DistanciaX:=-70;
         DistanciaY:=40;
         Caixa.rot:=[0,0.707,0.707,0];
      ENDIF
```

```
    ENDIF

    IF PaletitzemL1=true THEN
        Caixa.trans:=[DistanciaX,DistanciaY,DistanciaZ];
        Caixa.trans.z:=500;
        MoveJ Caixa, vel, z10, Ventosa, \WObj:=PosPalet1;
        Caixa.trans.z:=DistanciaZ;
        MoveL Caixa, vel, fine, Ventosa, \WObj:=PosPalet1;
        Caixa.trans.z:=500;
        MoveL Caixa, vel, z10, Ventosa, \WObj:=PosPalet1;
        Incr ProcessadesL1;
        IF PecesPalet=16 THEN
            CanviPaletL1:=TRUE;
            TPWrite "CANVI DE PALET";
        ENDIF
    ELSEIF PaletitzemL2=true THEN
        Caixa.trans:=[DistanciaX,DistanciaY,DistanciaZ];
        Caixa.trans.z:=500;
        MoveJ Caixa, vel, z10, Ventosa, \WObj:=PosPalet2;
        Caixa.trans.z:=DistanciaZ;
        MoveL Caixa, vel, fine, Ventosa, \WObj:=PosPalet2;
        Caixa.trans.z:=500;
        MoveL Caixa, vel, z10, Ventosa, \WObj:=PosPalet2;
        Incr ProcessadesL2;
        IF PecesPalet=16 THEN
            CanviPaletL2:=TRUE;
            TPWrite "CANVI DE PALET";
        ENDIF
    ENDIF
    RETURN PecesPalet;
ENDFUNC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ENDMODULE
```

D. MÒDUL DADES

```
MODULE Dades1

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!    CONVERSIÓ SEGONS -> MINUTS -> HORES    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
FUNC string ConvertirSegons (num Temps)
   VAR string TempsProces:="";
   VAR num AuxTemps:=0;
   AuxTemps:=Temps;
   Temps:=Trunc(Temps);
   AuxTemps:=(AuxTemps-Temps)*1000;
   Decimes:=Trunc(AuxTemps);
   IF Temps<60 THEN
      Hores:=0;
      Minuts:=0;
      Segons:=Temps;
   ELSEIF Temps>=60 THEN
      Minuts:=Temps DIV 60;
      Segons:=Temps MOD 60;
   ENDIF
   IF Minuts>=60 THEN
      Hores:=Minuts DIV 60;
      Minuts:=Minuts MOD 60;
   ELSEIF Minuts<60 THEN
      Hores:=0;
   ENDIF
   Horess:=NumToStr(Hores,0);
   Minutss:=NumToStr(Minuts,0);
   Segonss:=NumToStr(Segons,0);
   Decimess:=NumToStr(Decimes,0);
   IF Hores<10 then
      Horess:="0"+Horess;
   ENDIF
   IF Minuts<10 THEN
      Minutss:="0"+Minutss;
   ENDIF
   IF Segons<10 THEN
      Segonss:="0"+Segonss;
   ENDIF
   IF Decimes<100 THEN
      Decimess:="0"+Decimess;
   ELSEIF Decimes<10 THEN
      Decimess:="00"+Decimess;
   ENDIF
   TempsProces:=Horess+SeparaHora+Minutss+SeparaHora+Segonss+SeparaHora+Decimess;
   RETURN TempsProces;
ENDFUNC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!    ACTUALITZA VARIABLES    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC ActualitzaVariables()
   TotalsL1:=ProcessadesL1+PecesCQF+PecesCQL1;
   TotalsL2:=ProcessadesL2+PecesCQL2;
   PeçaActual:=TotalsL1+TotalsL2+1;
   RestantsL1:=DemandaL1-ProcessadesL1;
   RestantsL2:=DemandaL2-ProcessadesL2;
   RestantsTotals:=DemandaTotal-(ProcessadesL1+ProcessadesL2);
   IF RestantsTotals>0 THEN
      PercentRestantsL1:=(RestantsL1*100) div RestantsTotals;
      PercentRestantsL2:=(RestantsL2*100) div RestantsTotals;
   ENDIF
```

```
  IF RestantsL1>RestantsL2 THEN
    DifPercentLinies:=PercentRestantsL1-PercentRestantsL2;
  ELSEIF RestantsL2>RestantsL1 THEN
    DifPercentLinies:=PercentRestantsL2-PercentRestantsL1;
  ELSE
    DifPercentLinies:=0;
  ENDIF
  IF RestantsTotals<>DemandaTotal THEN
    Excel;
  ENDIF
  TempsTotal:= ClkRead (TemporitzadorTotal);!
  TempsTotalString:=ConvertirSegons(TempsTotal);!

ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   DADES A MOSTRAR   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC Dades()
  TPERASE;
  IF DesdoblarProd=1 THEN
    TPWRITE "DESDOBLAMENT DE LA LINIA ACTIVAT "\Num:=LiniaDesdoblada ;
    TPWRITE "Peces processades costat 1: "\Num:=ProcessadesL1;
    TPWRITE "Peces processades costat 2: "\Num:=ProcessadesL2;
  ELSEIF DesdoblarProd=2 OR DesdoblarProd=0 THEN
    TPWRITE "Diferència %: "\Num:=DifPercentLinies;
    TPWRITE "Peces demanades L1: "\Num:=DemandaL1;
    TPWRITE "Peces demanades L2: "\Num:=DemandaL2;
    TPWRITE "Linia 1: "\Num:=ProcessadesL1;
    TPWRITE "Linia 2: "\Num:=ProcessadesL2;
  ENDIF
  TPWRITE "Peces restants: "\Num:=RestantsTotals;
  TPWRITE "Control qualitat L1: "\Num:=PecesCQL1;
  TPWRITE "Control qualitat L2: "\Num:=PecesCQL2;
  TPWRITE "Control qualitat FESTO: "\Num:=PecesCQF;
  TPWRITE TempsTotalString;
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   DADES A MOSTRAR   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROC Excel()
  Write ArxiuDadesExcel, CTime()\NoNewLine;
  Write ArxiuDadesExcel, ";"\NoNewLine;
  Write ArxiuDadesExcel, ""\Num:=PeçaActual-1\NoNewLine;
  Write ArxiuDadesExcel, ";"\NoNewLine;
  IF PaletitzemL1=TRUE THEN
    Write ArxiuDadesExcel, "Linia 1;"\NoNewLine;
    Write ArxiuDadesExcel, TipusPaletitzarL1\NoNewLine;
    Write ArxiuDadesExcel, ";"\NoNewLine;
  ELSEIF PaletitzemL2=TRUE THEN
    Write ArxiuDadesExcel, "Linia 2;"\NoNewLine;
    Write ArxiuDadesExcel, TipusPaletitzarL2\NoNewLine;
    Write ArxiuDadesExcel, ";"\NoNewLine;
  ENDIF
  IF ControlQualitatL1Marxa=TRUE THEN
    Write ArxiuDadesExcel, "CQL1;"\NoNewLine;
    Write ArxiuDadesExcel, ";"\NoNewLine;
  ELSEIF ControlQualitatL2Marxa=TRUE THEN
    Write ArxiuDadesExcel, "CQL2;"\NoNewLine;
    Write ArxiuDadesExcel, ";"\NoNewLine;
  ELSEIF ControlQualitatFMarxa=TRUE THEN
```

```
      Write ArxiuDadesExcel, "CQF;"\NoNewLine;
      Write ArxiuDadesExcel, ";"\NoNewLine;
   ENDIF
   ClkStop TemporitzadorCicle;
   TempsCicle:= ClkRead (TemporitzadorCicle);
   TempsCicleString:=ConvertirSegons(TempsCicle);
   Write ArxiuDadesExcel, TempsCicleString\NoNewLine;
   Write ArxiuDadesExcel, ";";
ENDPROC
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ENDMODULE
```

2. PROGRAMA ARDUINO

```
#include <OPC.h>
OPCSerial ObjecteOPC;

int ContadorPackage = 0;
int DispositiuActual = 0;
float TempsIniciCicle = 0;
float TempsFinalCicle = 0;
float TempsCicleFloat = 0;
float TempsCicleFloat2 = 0;
int TempsCicleInt = 0;
int Demanda = 1;
int DemandaCQ = 0;
int DemandaInterna = 1;

/* DEFINICIO DE VARIABLES PER A MOVIMENT DEL ROBOT */
const int Data = 4;       //Pin Arduino //Pin 14 del shift register
const int Latch = 5;      //Pin Arduino //Pin 11 del shift register
const int Clock = 6;      //Pin Arduino //Pin 10 del shift register
const int Habilitar = 7;  //Pin Arduino
const int Marxa = 8;      //Pin Arduino
volatile int Estat = 0;
volatile int EstatAnterior = 0;
volatile long TempsBoto = 0; //Temps d'espera per els antirebots
volatile bool PermisMC = false;
volatile bool PermisMarxa = false;
byte num[] = {B00000000, B00000001, B00000010, B00000011, B00000100,
B00000101, B00000110, B00000111, B00001000, B00001001, B00001010,
B00001011, B00001100, B00001101, B00001110, B00001111, B00010000,
B00010001, B00010010, B00010011, B00010100, B00010101, B00010110,
B00010111, B00011000, B00011001, B00011010, B00011011, B00011100,
B00011101, B00011110, B00011111};

/* DEFINICIO DE VARIABLES PER A MOVIMENT DE L'EIX VERTICAL */
#include <Servo.h>
Servo ServoZ;
const int PinServoEix = 10;     //Pin Arduino
volatile bool dalt = false;
const float constant = 20;
const int VelHoming = 115;
float AngleBaixada = 75;
float posAbs = 0;
const int PinFinalCarrera = 12; //Pin Arduino
bool FinalCarrera = false;

/* DEFINICIO DE VARIABLES PER A MOVIMENT DE LA PINÇA */
Servo ServoP;//Nomes per la comunicacio amb el monitor
const int servoPinsa = 11;  //Pin Arduino

/* DEFINICIO DE LES VARIABLES OPC */
bool FuncioOPC1(const char *itemID, const opcOperation opcOP, const
bool value) {
  if (opcOP == opc_opwrite) {
    PermisMarxa = value;
  }
  else
    return PermisMarxa;
}
bool FuncioOPC2(const char *itemID, const opcOperation opcOP, const
bool value) {
```

```
  return PermisMC;
}
int FuncioOPC3(const char *itemID, const opcOperation opcOP, const int
value) {
  return ContadorPackage;
}
int FuncioOPC4(const char *itemID, const opcOperation opcOP, const int
value) {
  return DispositiuActual;
}
float FuncioOPC5(const char *itemID, const opcOperation opcOP, const
float value) {
  if (Estat == 1) {
    TempsCicleFloat2 = TempsCicleFloat;
  }
  return TempsCicleFloat2;
}
int FuncioOPC6(const char *itemID, const opcOperation opcOP, const int
value) {
  if (opcOP == opc_opwrite) {
    Demanda = value;
    ContadorPackage = 0;
    DispositiuActual = 0;
  }
  else
    return Demanda;
}
int FuncioOPC7(const char *itemID, const opcOperation opcOP, const int
value) {
  return Estat;
}
int FuncioOPC8(const char *itemID, const opcOperation opcOP, const int
value) {
  DemandaCQ = value;
  DemandaInterna = Demanda + DemandaCQ;
  return DemandaInterna;
}

void setup() {
  ObjecteOPC.setup();
  ObjecteOPC.addItem("PermisMarxa", opc_readwrite, opc_bool,
FuncioOPC1);
  ObjecteOPC.addItem("PermisMC", opc_read, opc_bool, FuncioOPC2);
  ObjecteOPC.addItem("ContadorPackage", opc_read, opc_int,
FuncioOPC3);
  ObjecteOPC.addItem("DispositiuActual", opc_read, opc_int,
FuncioOPC4);
  ObjecteOPC.addItem("TempsCicle", opc_read, opc_float, FuncioOPC5);
  ObjecteOPC.addItem("Demanda", opc_readwrite, opc_int, FuncioOPC6);
  ObjecteOPC.addItem("Estat", opc_read, opc_int, FuncioOPC7);
  ObjecteOPC.addItem("DemandaCQ", opc_write, opc_int, FuncioOPC8);

  Serial.begin(9600);
  ServoZ.attach(PinServoEix);
  ServoP.attach(servoPinsa);
  attachInterrupt(0, INTMarxaParo, RISING);
  attachInterrupt( 1, INTMotionComplete, RISING);
  pinMode (Data, OUTPUT);
  pinMode (Clock, OUTPUT);
  pinMode (Latch, OUTPUT);
  pinMode (Habilitar, OUTPUT);
```

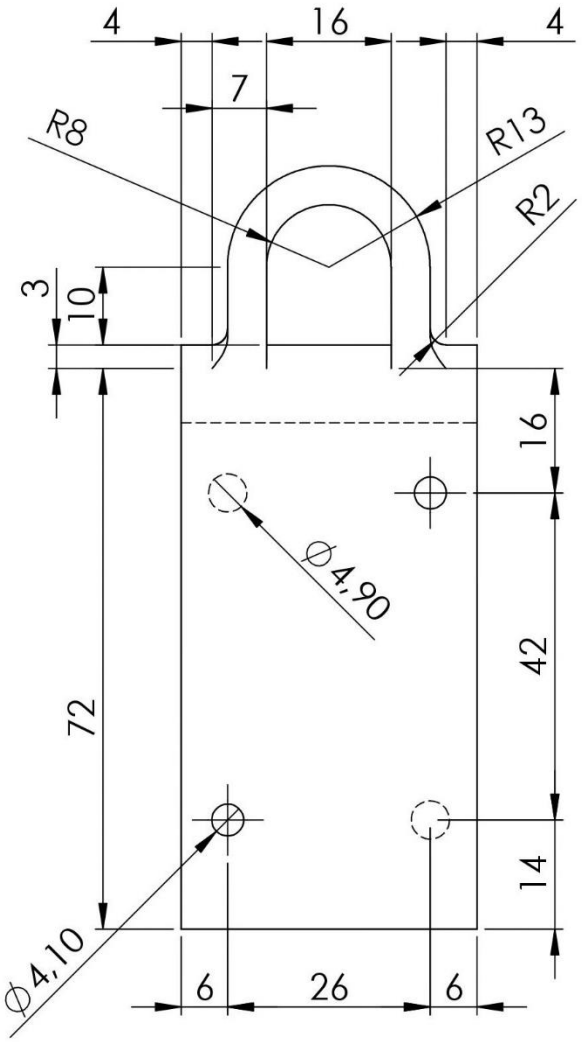
```
  pinMode (Marxa, OUTPUT);
  digitalWrite (Habilitar, HIGH);

  Homing(VelHoming);// Cal rebre senyal del final de carrera pk
continui
  delay(500);
  digitalWrite (Latch, LOW);
  shiftOut (Data, Clock, LSBFIRST, 0);
  digitalWrite (Latch, HIGH);
  delay (100);
  digitalWrite (Marxa, HIGH);
  delay(100);
  digitalWrite (Marxa, LOW);
}

void loop() {
  if (PermisMarxa == true and PermisMC == true /*and dalt == true*/
and Estat != EstatAnterior and ContadorPackage < DemandaInterna) {
    EstatAnterior = Estat;
    switch (Estat) {
      case 1://Centre
        TempsIniciCicle = float(millis());
        Moviment(num[1]);
        Homing(VelHoming);
        ServoP.write(0);
        break;
      case 2://Carregador
        Moviment(num[2]);
        ServoP.write(0);
        posAbs = 90;
        MovimentZ(posAbs);
        ServoP.write(90);
        temporitzador(800);
        Homing(VelHoming);
        break;
      case 3://CarregadorCaixa
        Moviment(num[8]);
        posAbs = 40;
        MovimentZ(posAbs);
        ServoP.write(20);
        temporitzador(300);
        Homing(VelHoming);
        ServoP.write(0);
        break;
      case 4://Bateria
        Moviment(num[3]);
        posAbs = 90;
        MovimentZ(posAbs);
        ServoP.write(100);
        temporitzador(500);
        Homing(VelHoming);
        break;
      case 5://BateriaCaixa
        Moviment(num[9]);
        posAbs = 40;
        MovimentZ(posAbs);
        ServoP.write(50);
        temporitzador(700);
        Homing(VelHoming);
        ServoP.write(0);
        break;
```

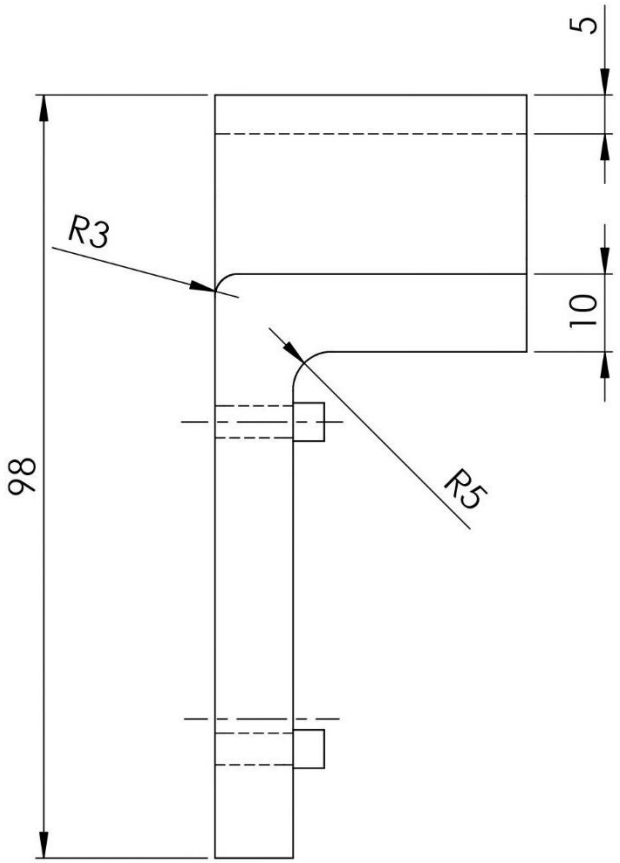
```
case 6://Cable
  Moviment(num[4]);
  posAbs = 88;
  MovimentZ(posAbs);
  ServoP.write(170);
  temporitzador(1000);
  Homing(VelHoming);
  break;
case 7://CableCaixa
  Moviment(num[10]);
  posAbs = 40;
  MovimentZ(posAbs);
  ServoP.write(100);
  temporitzador(500);
  Homing(VelHoming);
  ServoP.write(0);
  break;
case 8://Instruccions
  Moviment(num[5]);
  posAbs = 90;
  MovimentZ(posAbs);
  ServoP.write(95);
  temporitzador(500);
  Homing(VelHoming);
  break;
case 9://InstruccionsCaixa
  Moviment(num[1]);
  posAbs = 40;
  MovimentZ(posAbs);
  ServoP.write(20);
  temporitzador(500);
  Homing(VelHoming);
  ServoP.write(0);
  break;
case 10://Mobil
  Moviment(num[6]);
  posAbs = 95;
  MovimentZ(posAbs);
  ServoP.write(75);
  temporitzador(500);
  Homing(VelHoming);
  break;
case 11://MobilCaixa
  Moviment(num[1]);
  posAbs = 35;
  MovimentZ(posAbs);
  ServoP.write(15);
  temporitzador(500);
  Homing(VelHoming);
  ServoP.write(0);
  break;
case 12://Tapa
  Moviment(num[7]);
  posAbs = 80;
  MovimentZ(posAbs);
  ServoP.write(50);
  temporitzador(500);
  Homing(VelHoming);
  break;
case 13://TapaCaixa
  Moviment(num[1]);
```

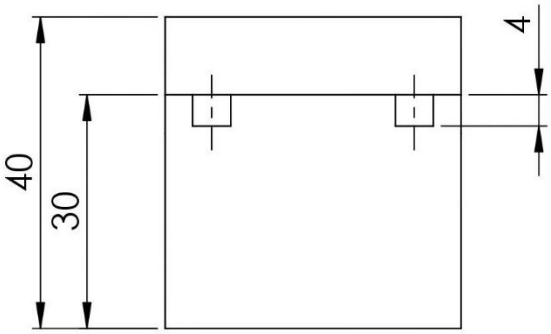
```
        posAbs = 35;
        MovimentZ(posAbs);
        ServoP.write(0);
        temporitzador(500);
        Homing(VelHoming);
        ContadorPackage++;
        TempsFinalCicle = float(millis());
        TempsCicleFloat = (TempsFinalCicle - TempsIniciCicle) / 1000;
        break;
    }
    if (Estat > 1) {
      DispositiuActual = ContadorPackage + 1;
    }
    else if (DispositiuActual == DemandaInterna) {
      DispositiuActual = ContadorPackage;
      PermisMarxa = false;
    }
  }
  ObjecteOPC.processOPCCommands();
}

void INTMarxaParo() {
  if (millis() >= TempsBoto + 250) {
    if (Estat == 0) {
      Estat = 1;
      PermisMarxa = true;
    }
    else {
      PermisMarxa = !PermisMarxa;
    }
    TempsBoto = millis();
  }
}

void INTMotionComplete() {
  PermisMC = true;
  if (Estat < 13) {
    Estat++;
  } else if (PermisMarxa == true and DispositiuActual <=
DemandaInterna) {
    Estat = 1;
  } else {
  }
}

void Moviment(byte posicio) {
  digitalWrite (Marxa, LOW);
  digitalWrite (Latch, LOW);
  shiftOut (Data, Clock, LSBFIRST, num[posicio]);
  digitalWrite (Latch, HIGH);
  temporitzador (100);
  digitalWrite (Marxa, HIGH);
  PermisMC = false;
  while (PermisMC == false) {
  }
}

void Homing(int velHom) {
  while (dalt == false) {
    ServoZ.write(velHom);
    delay(20);
```

```
    if (digitalRead(PinFinalCarrera) == HIGH) {
      ServoZ.write(90);
      dalt = true;
      posAbs = 0;
    }
    ObjecteOPC.processOPCCommands();
  }
}

void MovimentZ(float distancia) {
  float TempsAct = 0;
  float TempsBaixada = 0;
  if (distancia > 0) {
    TempsBaixada = (distancia / constant) * 1000;
    TempsAct = float(millis());
    dalt = false;
    while (float(millis()) < (TempsAct + TempsBaixada)) {
      ServoZ.write(AngleBaixada);
      ObjecteOPC.processOPCCommands();
    }
    ServoZ.write(90);
  }
  else {
    ServoZ.write(90);
  }
}

void temporitzador(long retard) {
  long temps1 = long(millis());
  while (long(millis() - temps1) < retard) {
    ObjecteOPC.processOPCCommands();
  }
}
```

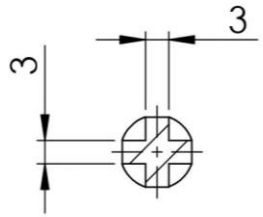
3. PLÀNOLS



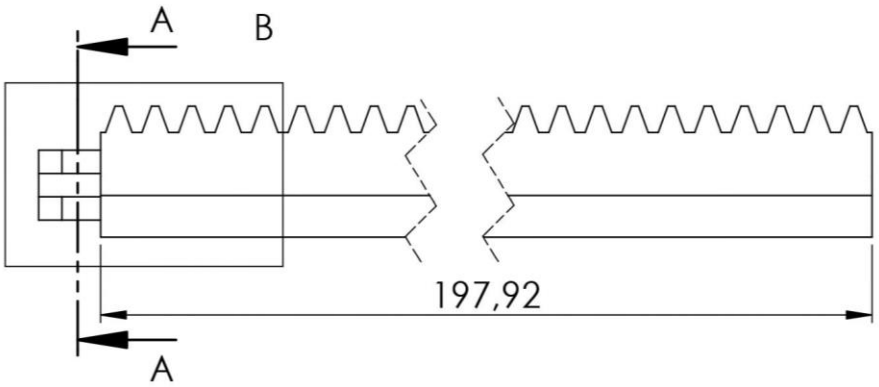


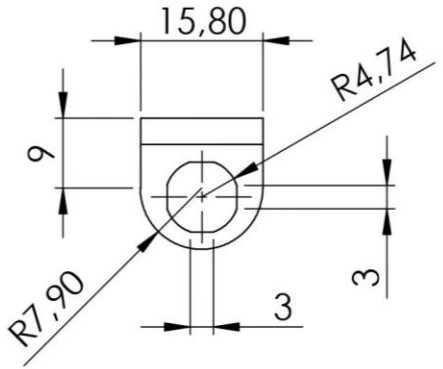


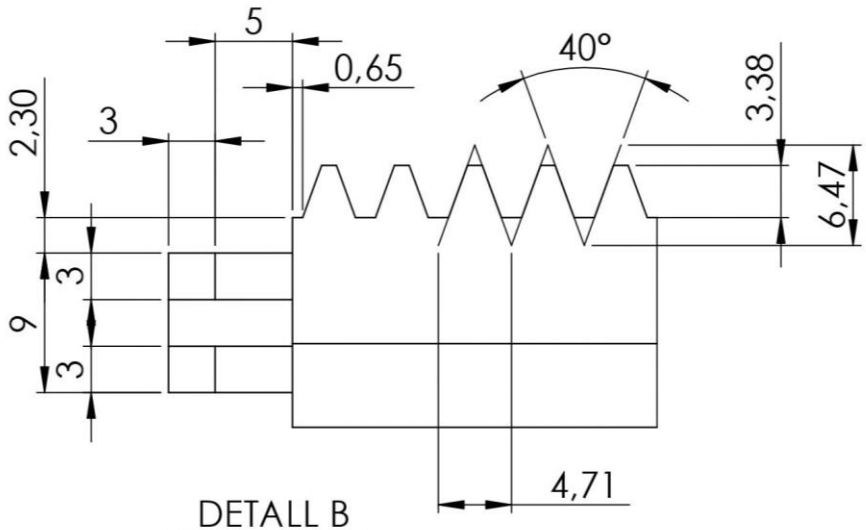
Universitat Politècnica de Catalaunya	Denominació plànol: Base eix vertical	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: E-001
	Escala: 1 : 1	Nº plànol: 01-18



SECCIÓ A-A

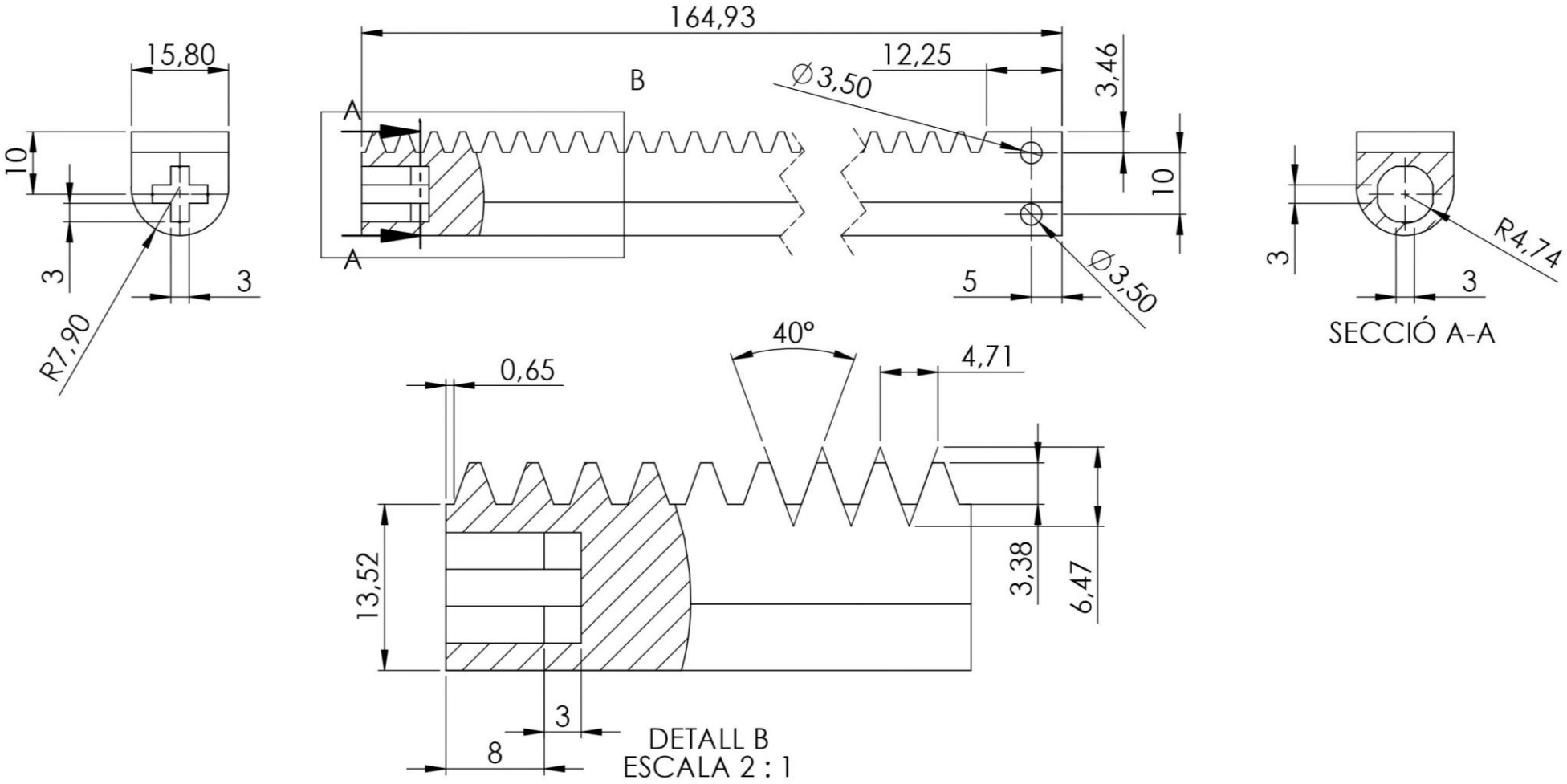




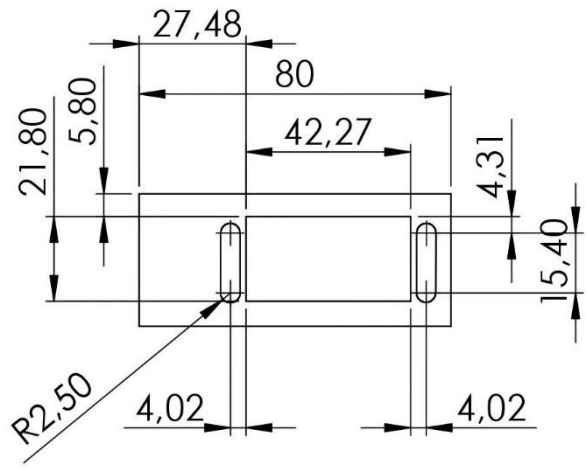


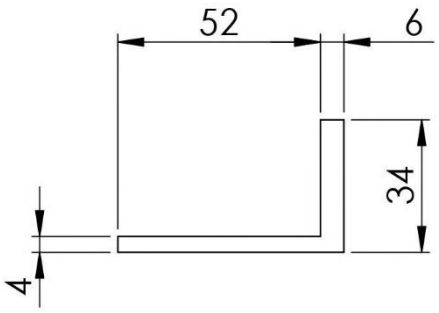
DETALL B
ESCALA 2 : 1

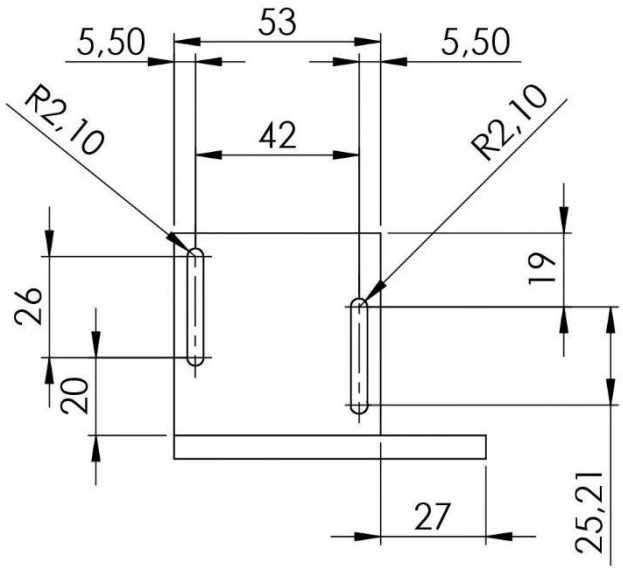
Universitat Politècnica de Catalaunya	Denominació plànol: Eix superior	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: E-002
	Escala: 1 : 1	Nº plànol: 02-18



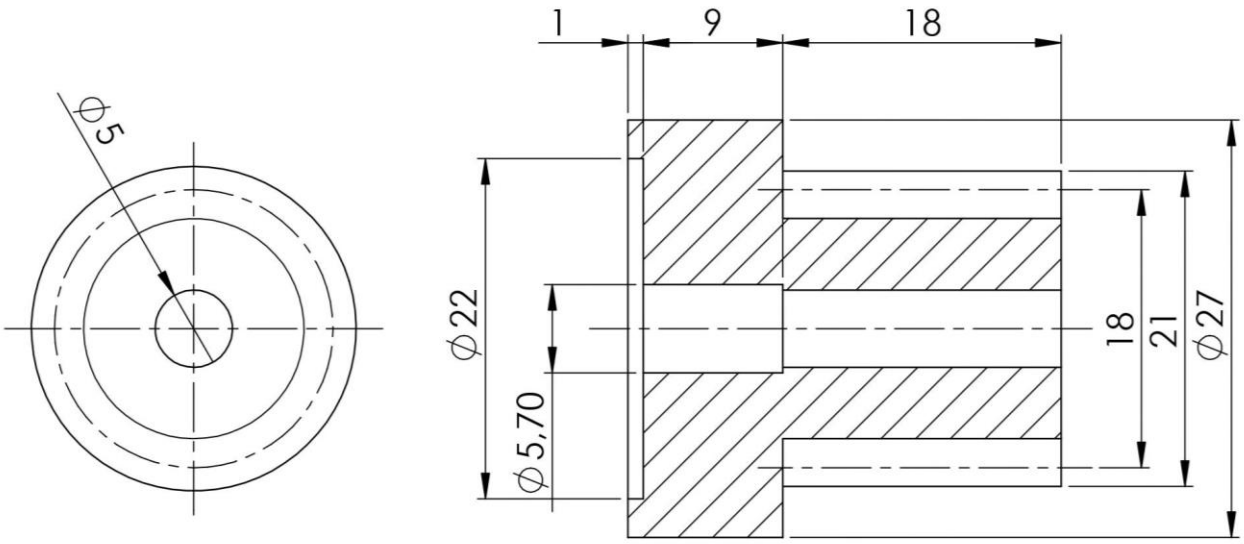
Universitat Politècnica de Catalaunya	Denominació plànol: Eix inferior	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: E-003
	Escala: 1 : 1	Nº plànol: 03-18





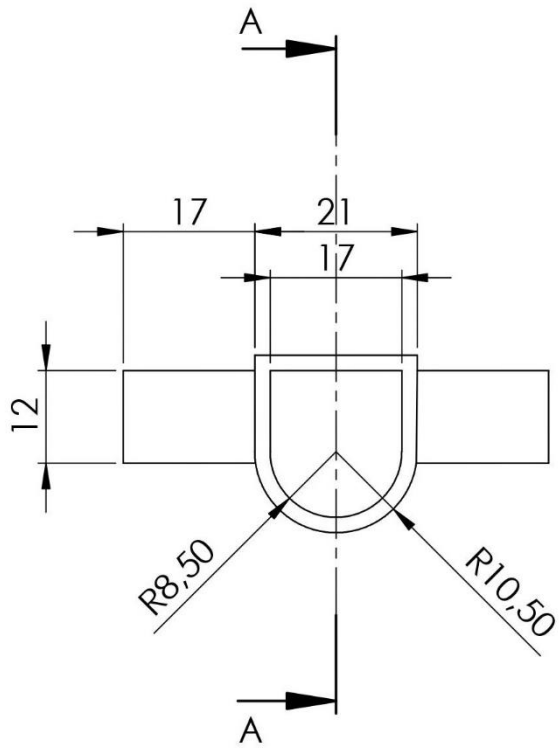


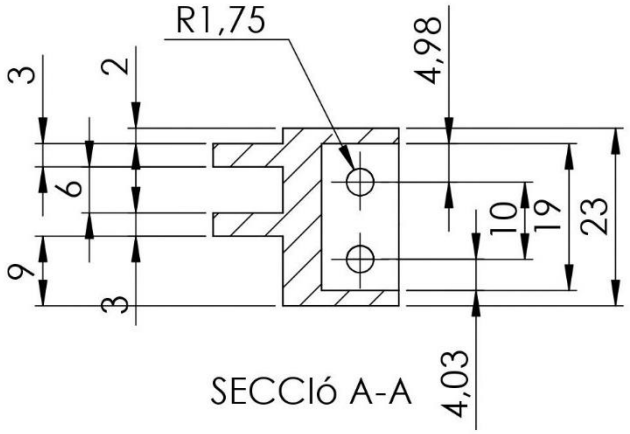
Universitat Politècnica de Catalaunya	Denominació plànol: Suport motor vertical	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: E-004
	Escala: 1 : 2	Nº plànol: 04-18



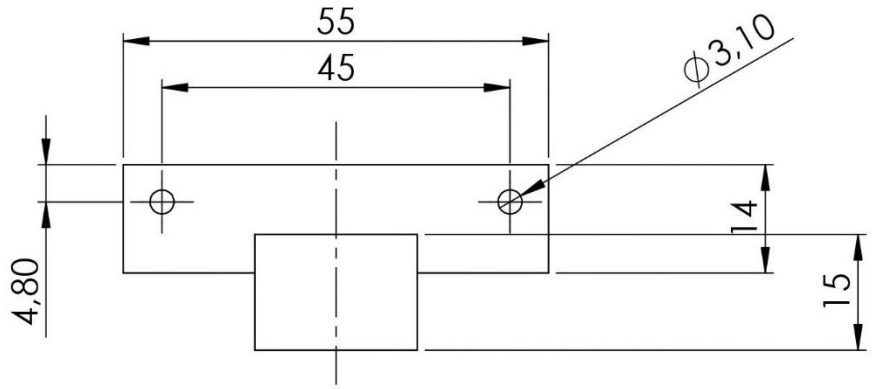
MESURA	
Mòdul	1,5
Angle de pressió	20
N° dents	12
Diàmetre primitiu	18
Diàmetre exterior	21
Diàmetre interior	14,24
Pas circular	30
Altura dent	3,38

Universitat Politècnica de Catalaunya	Denominació plànol: Pinyo eix	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: E-005
	Escala: 2 : 1	Nº plànol: 05-18

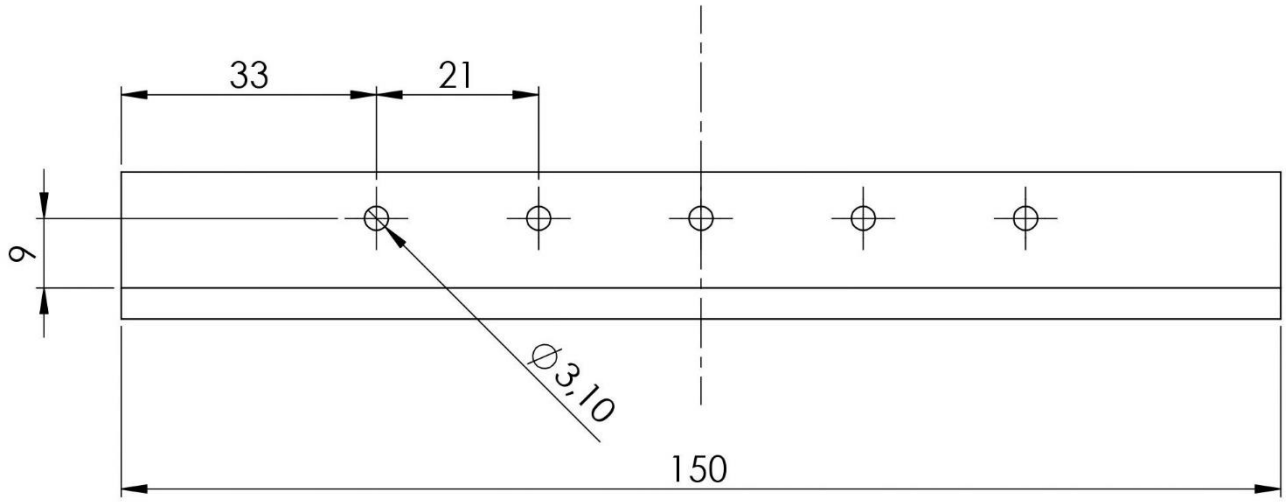


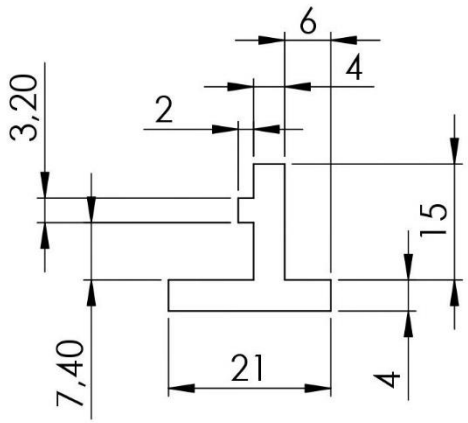


SECCIó A-A

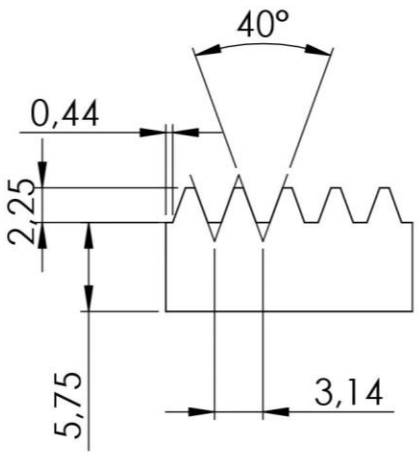


Universitat Politècnica de Catalaunya	Denominació plànol: Unio eix vertical	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: P-001
	Escala: 1 : 1	Nº plànol: 06-18

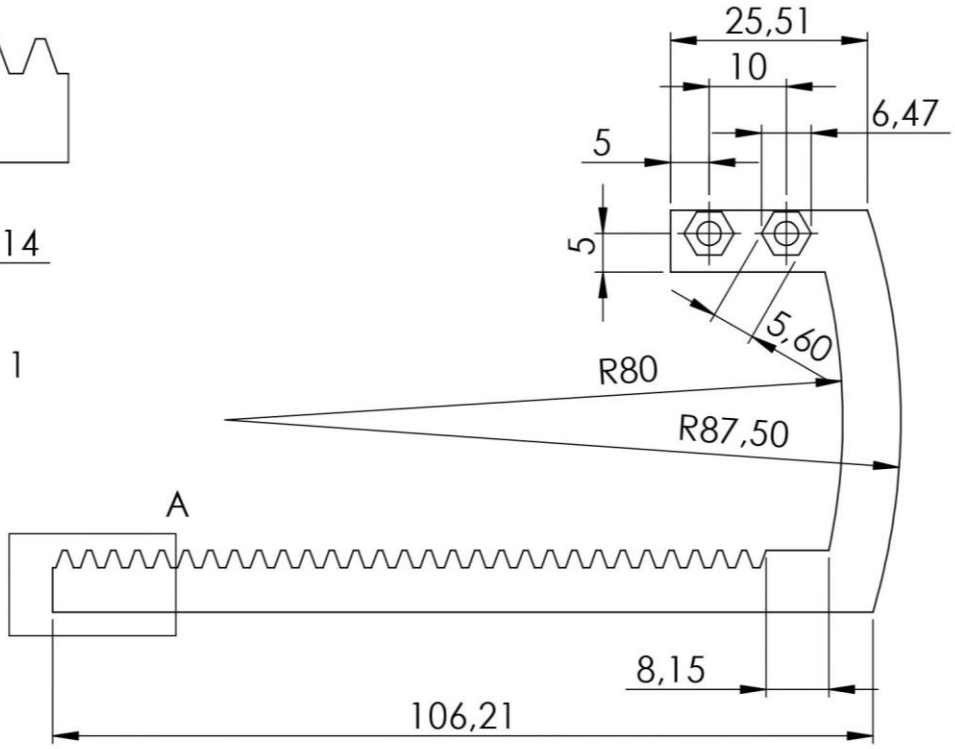


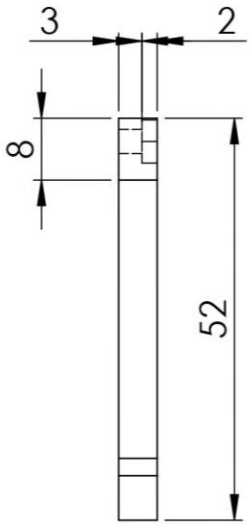


Universitat Politècnica de Catalaunya	Denominació plànol: Guia braços	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: P-002
	Escala: 1 : 1	Nº plànol: 07-18

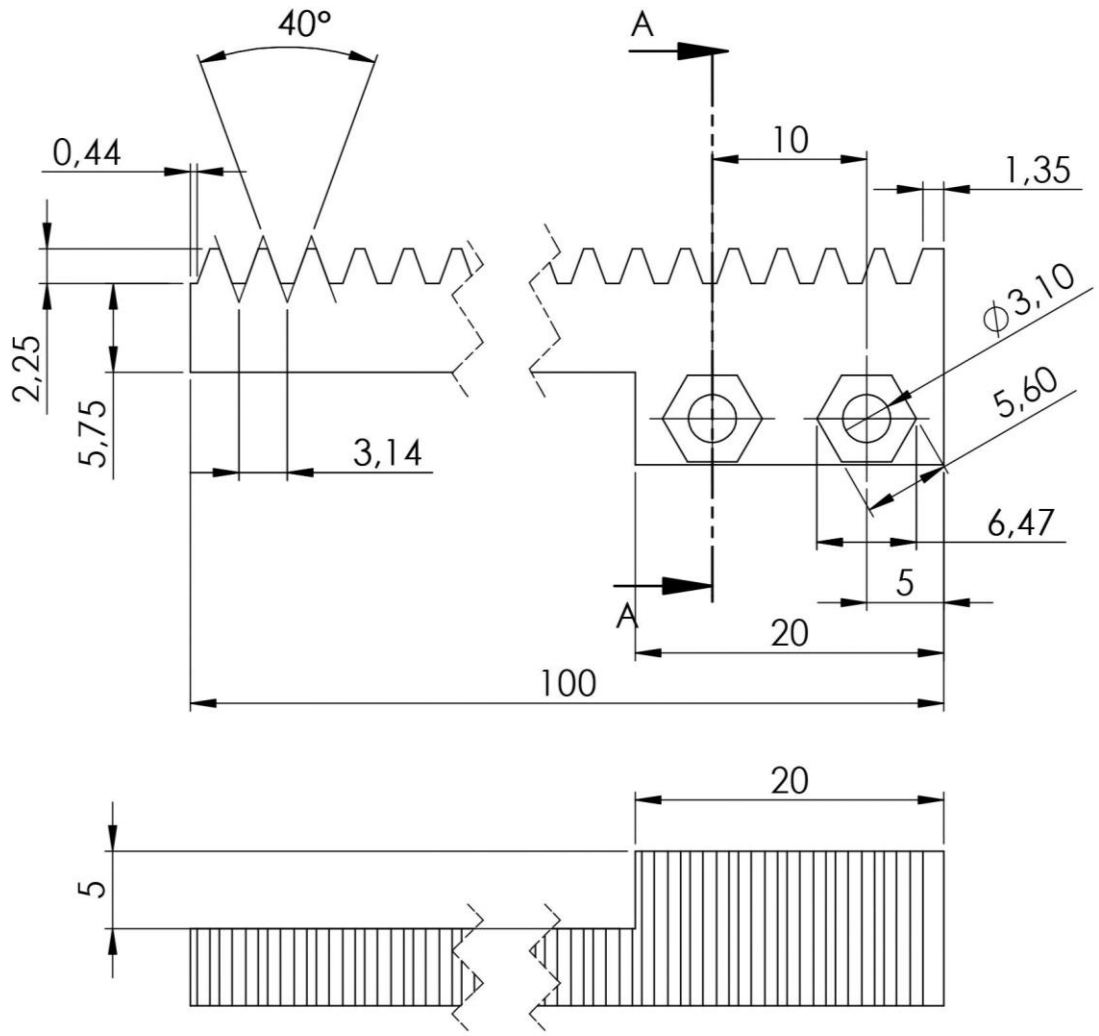


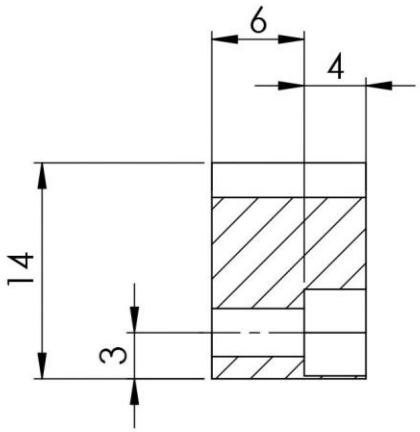
DETALL A
ESCALA 2 : 1





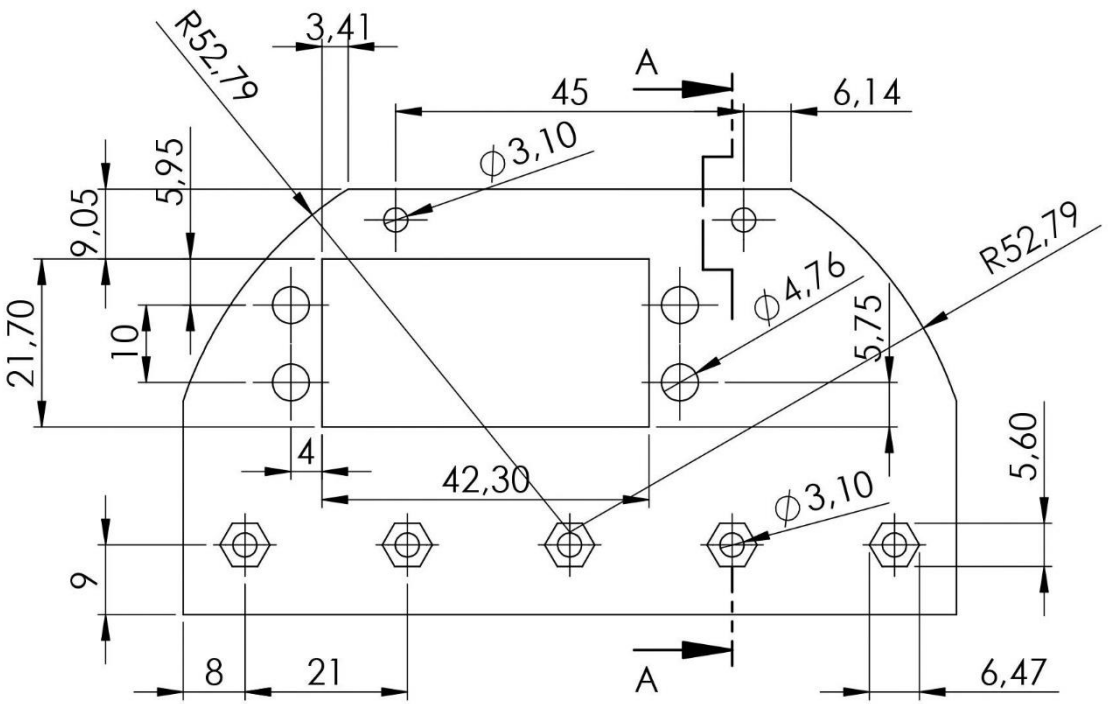
Universitat Politècnica de Catalaunya	Denominació plànol: Cremallera superior	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: P-003
	Escala: 1 : 1	Nº plànol: 08-18

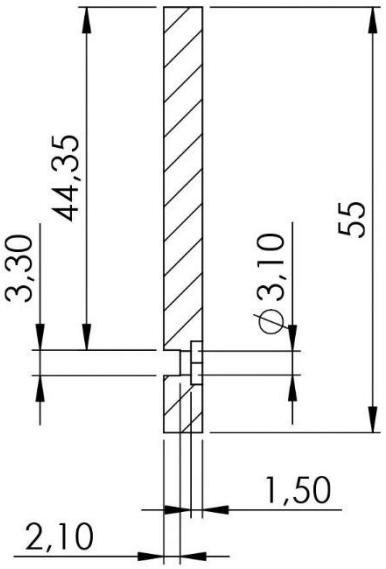




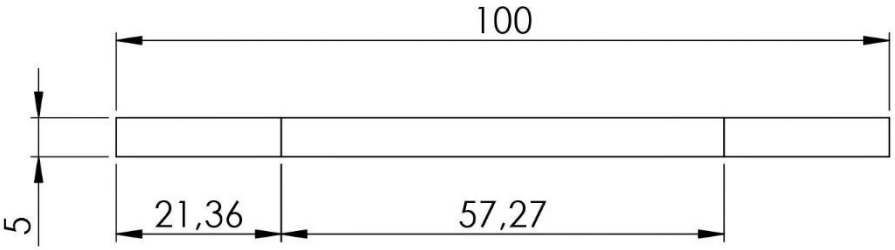
SECCIó A-A
ESCALA 2 : 1

Universitat Politècnica de Catalaunya	Denominació plànol: Cremallera inferior	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: P-004
	Escala: 2 : 1	Nº plànol: 09-18

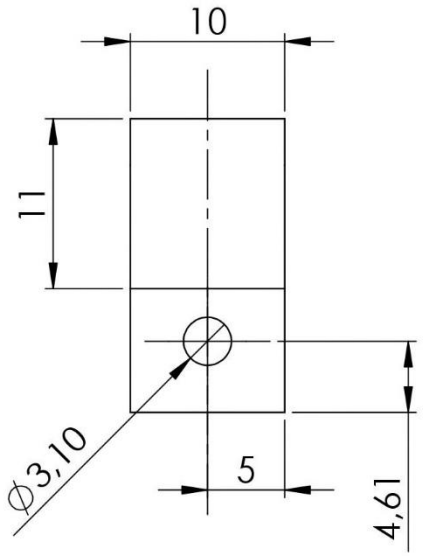


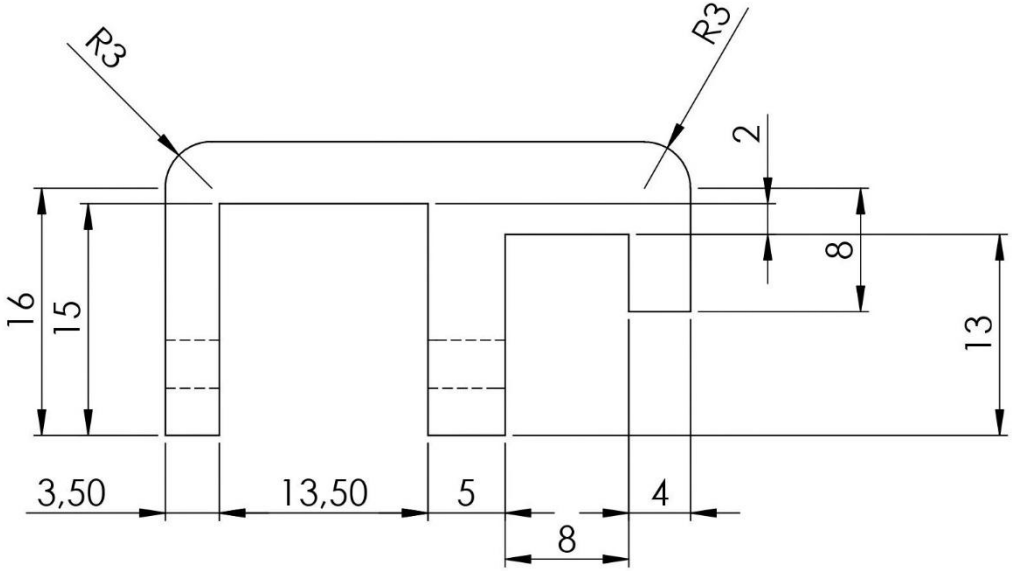


SECCIó A-A

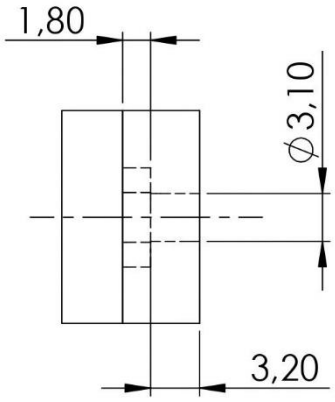


Universitat Politècnica de Catalaunya	Denominació plànol: Suport motor pinça	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: P-005
	Escala: 1 : 1	Nº plànol: 10-18

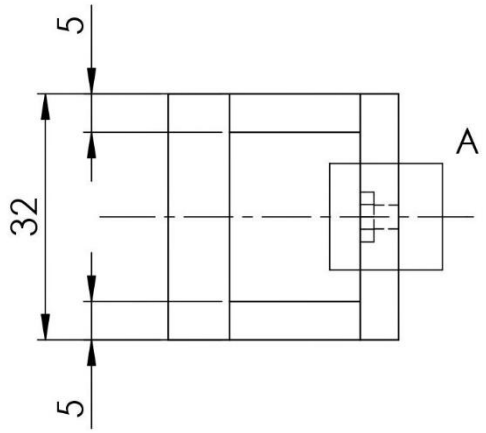


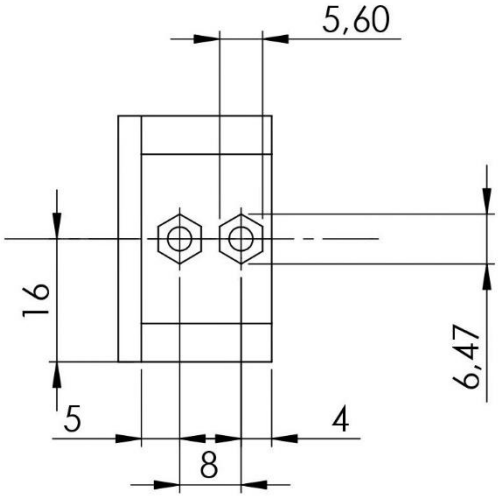


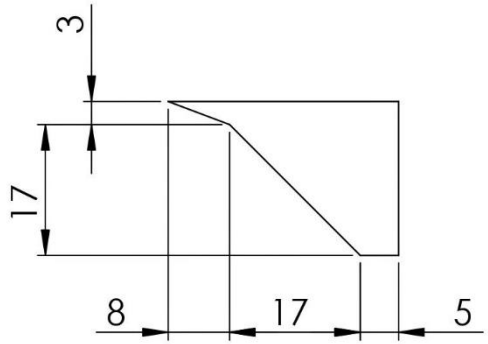
Universitat Politècnica de Catalaunya	Denominació plànol: Guia cremallera superior	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: P-006
	Escala: 2 : 1	Nº plànol: 11-18



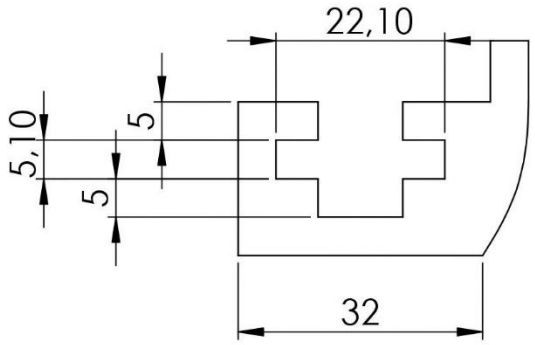
DETALL A
ESCALA 2 : 1

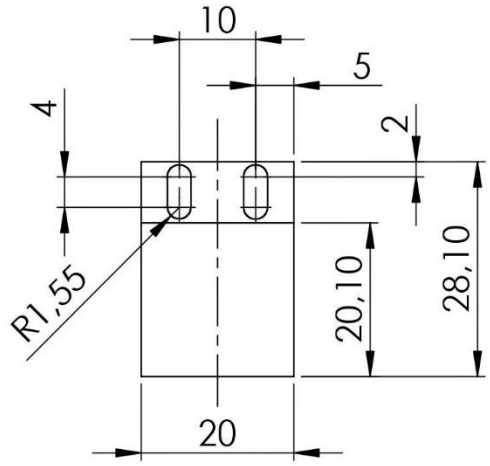


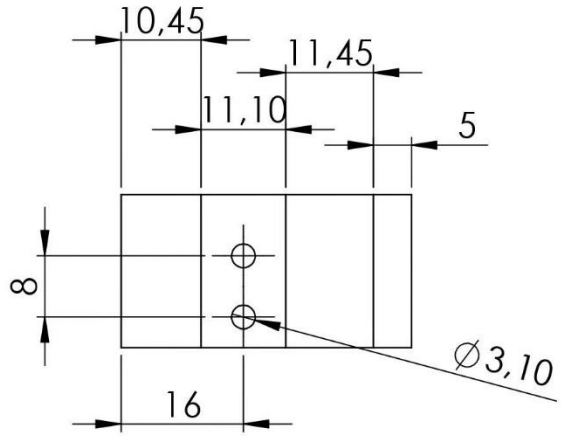




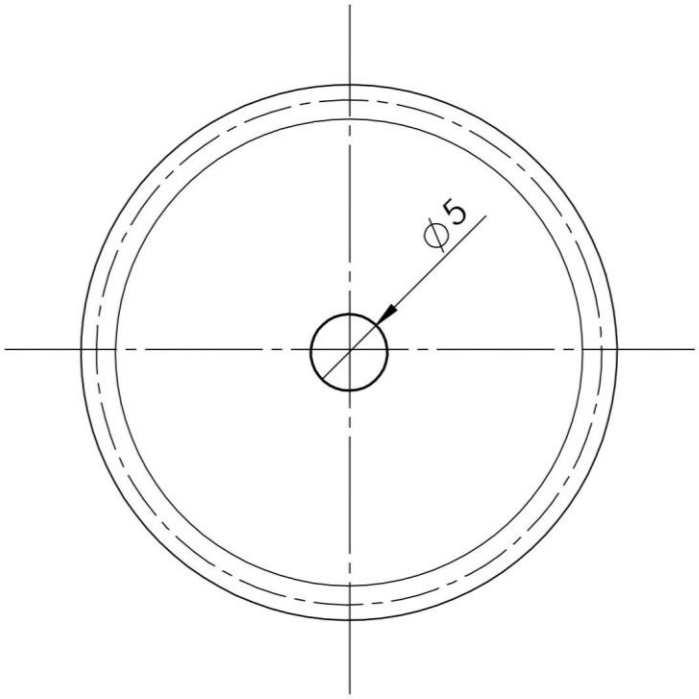
Universitat Politècnica de Catalaunya	Denominació plànol: Braços	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: P-007
	Escala: 2 : 1	Nº plànol: 12-18

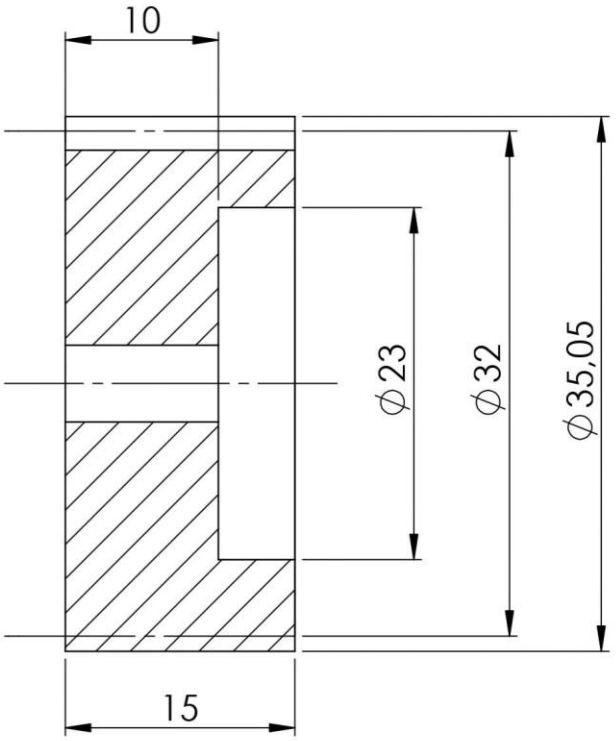






Universitat Politècnica de Catalaunya	Denominació plànol: Suport braços	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Còdi plànol: P-008
	Escala: 1: 1	Nº plànol: 13-18





MESURA	
Mòdul	1
Angle de pressió	20
N° dents	32
Diàmetre primitiu	32
Diàmetre exterior	35,05
Diàmetre interior	30,5
Pas circular	10,9
Altura dent	2,25

Universitat Politècnica de Catalaunya

Denominació plànol: Pinyo pinça

Cognoms, Nom: Pujadas Castells, Xavier

Projecte:

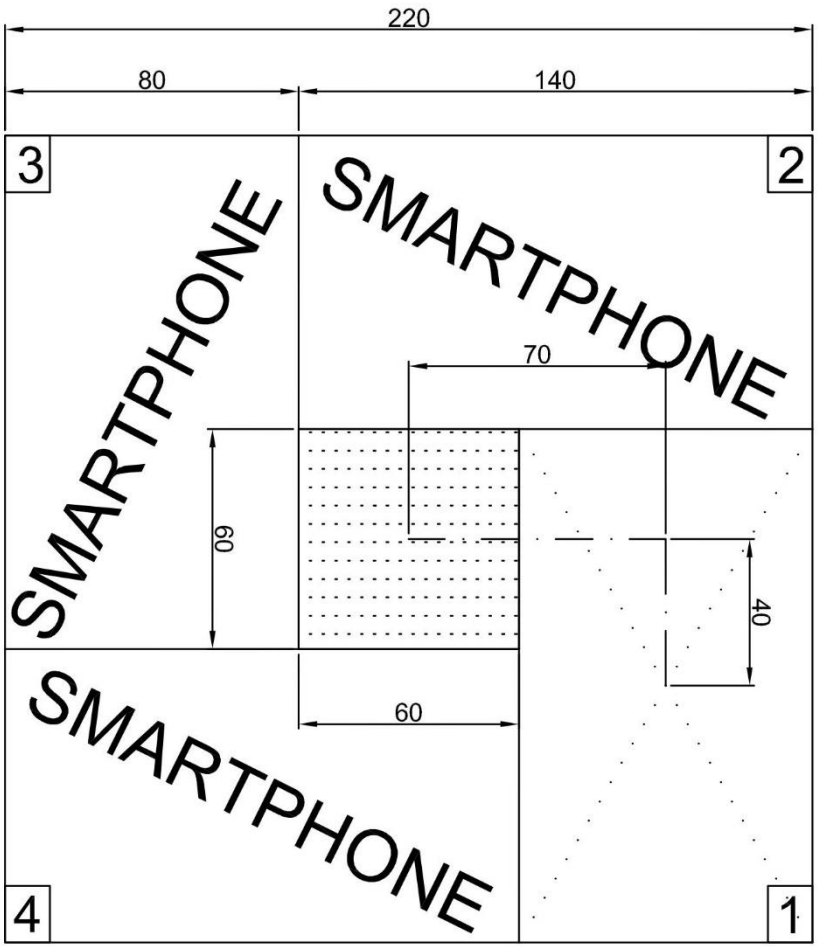
Estudi, disseny i integració de dos robots de diferent topologia dins d'una illa de procés

Data: Juliol-2016

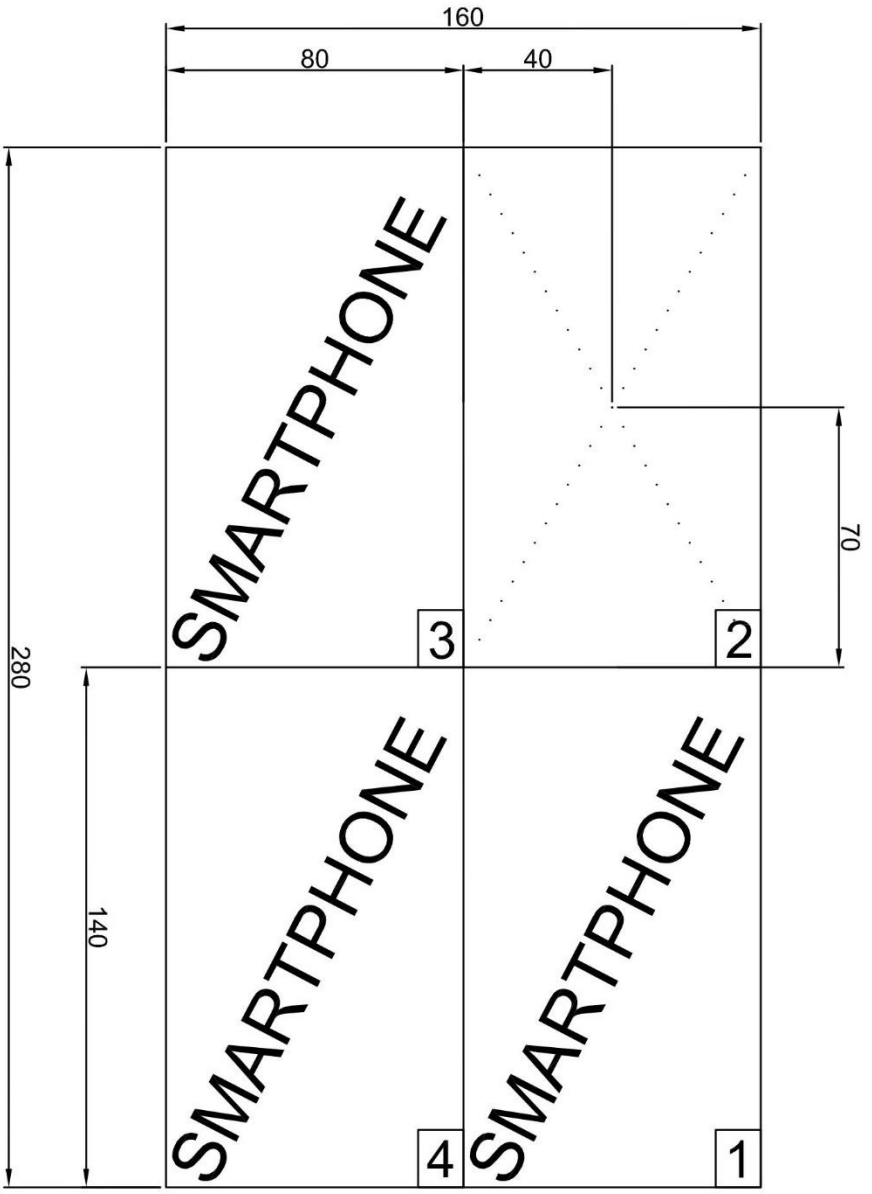
Còdi plànol: P-009

Escala: 2 : 1

Nº plànol: 14-18

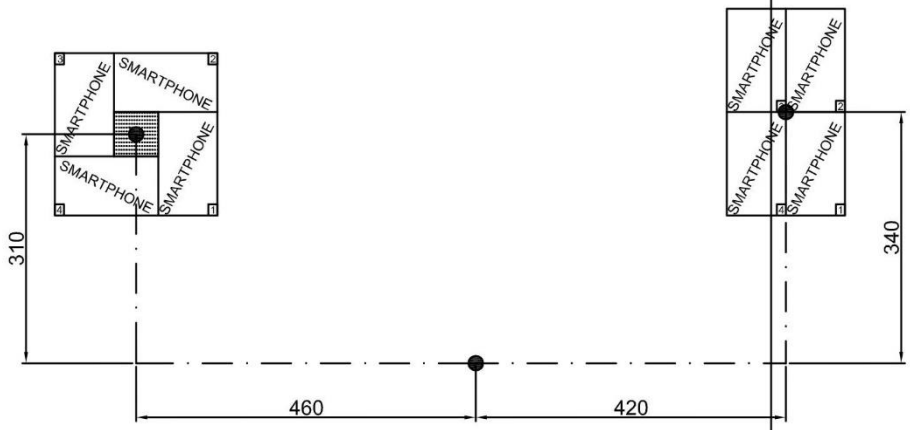


Universitat Politècnica de Catalunya	Denominació plànol: Paletitzat circular	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Codi plànol: PP-001
	Escala: 1:2	Nº plànol: 15-18

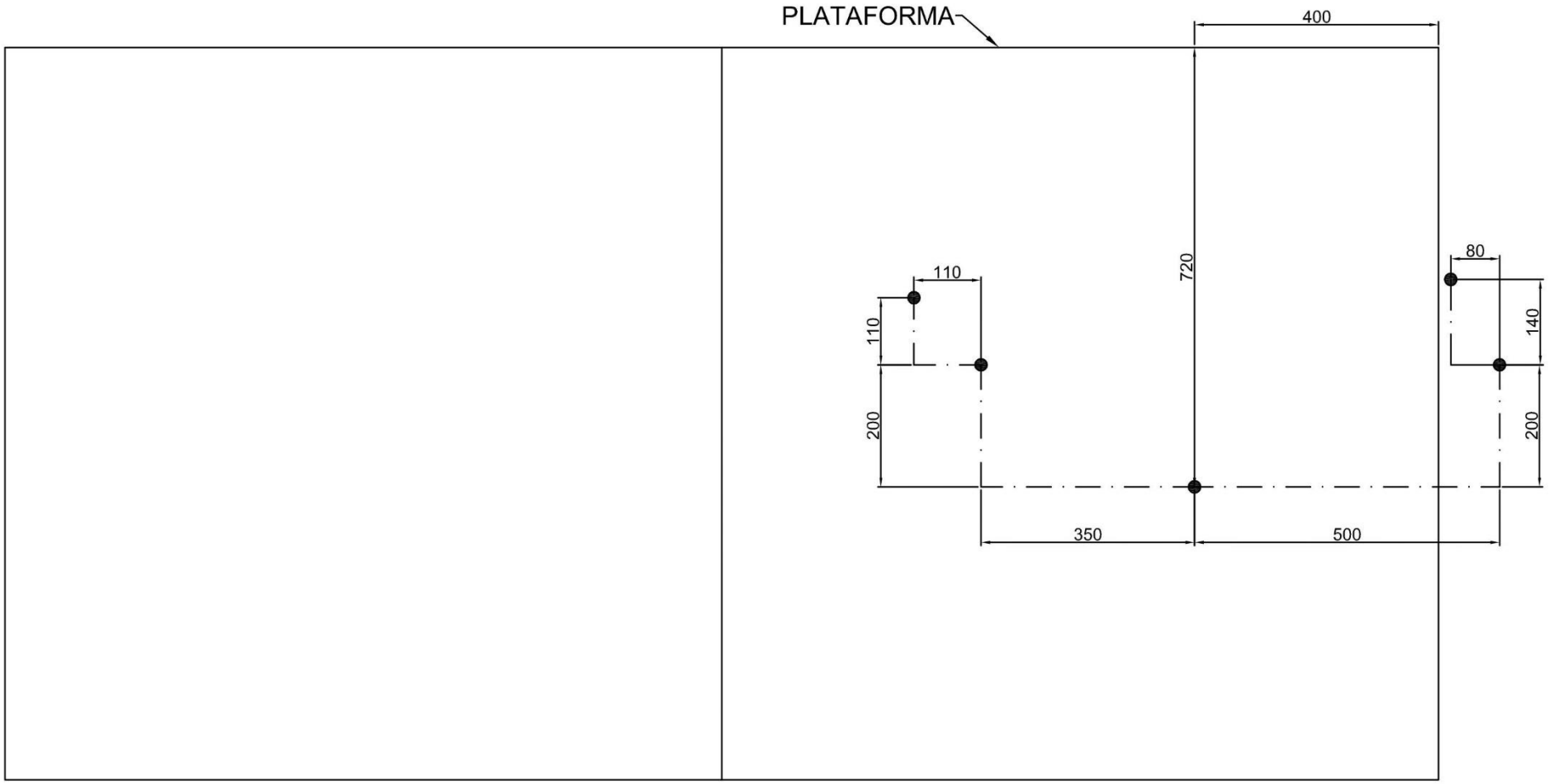


Universitat Politècnica de Catalunya	Denominació plànol: Paletitzat quadrat	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Codi plànol: PP-002
	Escala: 1:2	Nº plànol: 16-18

PLATAFORMA

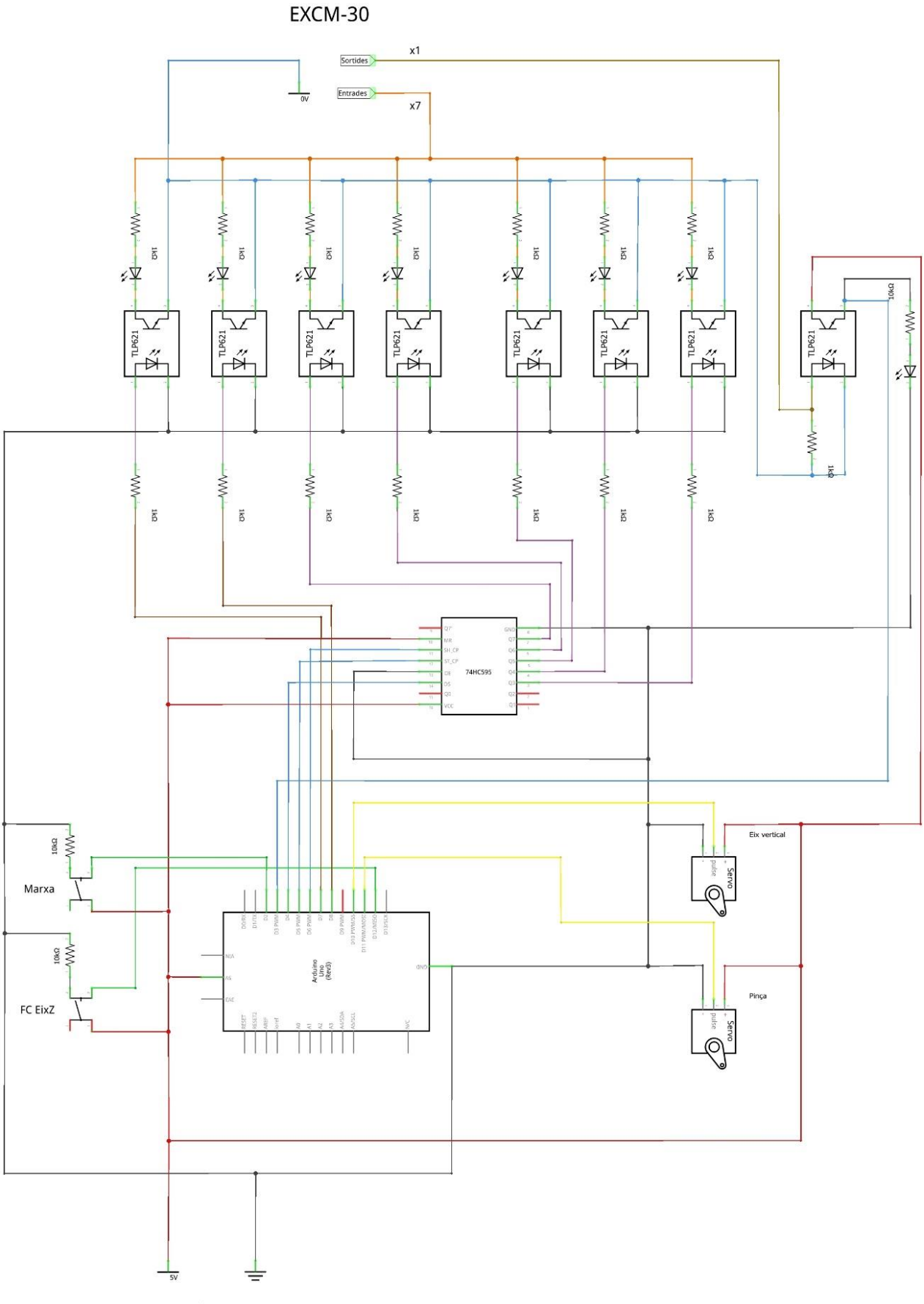


Universitat Politècnica de Catalunya	Denominació plànol: Posicions de paletitzat	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Codi plànol: PP-003
	Escala: 1:10	Nº plànol: 17-18

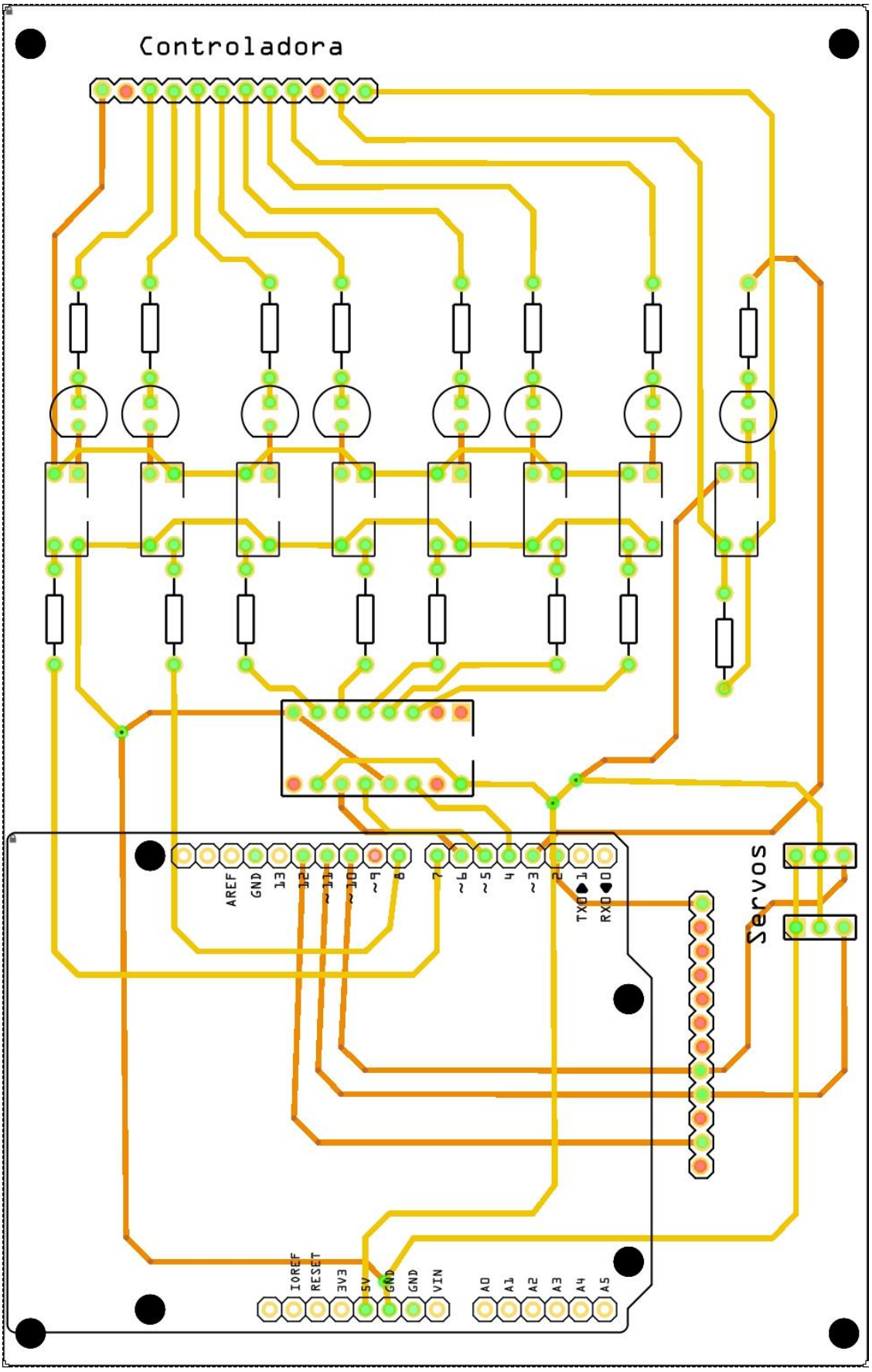


Universitat Politècnica de Catalunya	Denominació plànol: Referències del robot	Cognoms, Nom: Pujadas Castells, Xavier
Projecte: Estudi disseny i integració de dos robots de diferent topologia dins d'una illa de procés	Data: Juliol-2016	Codi plànol: PP-004
	Escala: 1:10	Nº plànol: 18-18

4. DISSENY DE LA PCB



Arduino


Controladora
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TXD 1
RXD 0
Servos
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5

5. DOCUMENTACIÓ

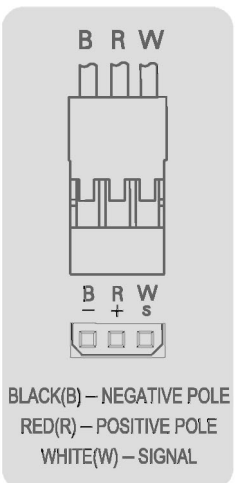
SPRINGRC®
www.springrc.com

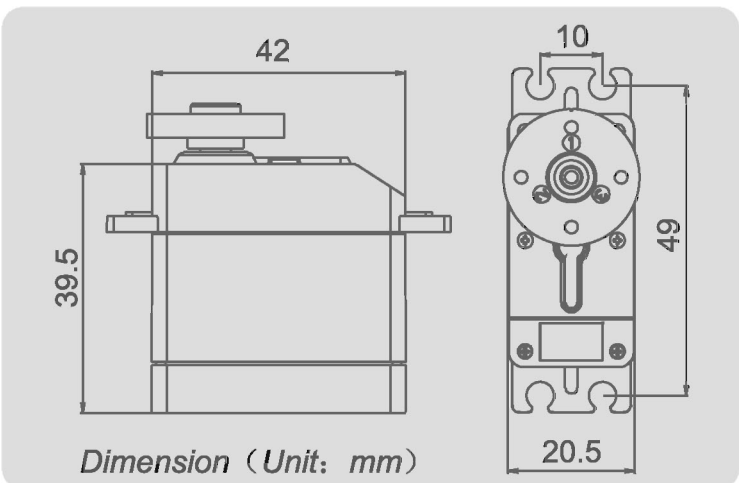
43R Servo(360° Rotation) Specification

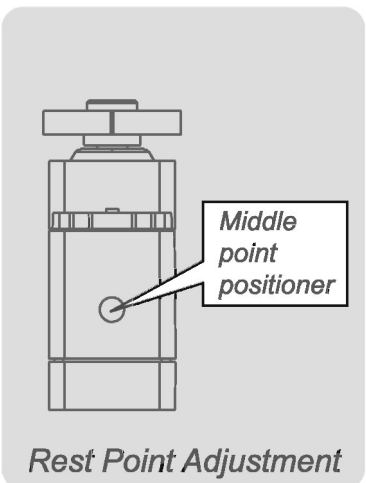
Thank you for choosing Spring Model's product

MODEL	TYPE	WEIGHT		4.8V			6V			DESCRIPTION	
		g	oz	SPEED	TORQUE		SPEED	TORQUE		GEAR	BEARING
				r/min	kg.cm	oz.in	r/min	kg.cm	oz.in		
SM-S4303R	Analog	44	1.55	60	3.3	45.8	70	4.8	66.7	1Metal Gear+ 4Plastic Gear	2
SM-S4306R		44	1.55	60	5.0	69.4	50	6.2	86.1	1Metal Gear+ 4Plastic Gear	2
SM-S4309R		60	2.12	58	7.9	109.7	49	8.7	120.8	Metal Gear	2
SM-S4315R		60	2.12	62	14.5	201.4	53	15.4	213.9	Metal Gear	2

▲ 43R Robot series servo controled via analog signal(PWM),stopped via middle point positiner.
▲ Standard interface(like JR)with 30cm wire.
▲ Rotation and Rest Point Adjustment:when analog signal inputs,servo chooses orientation according to impulse width.when intermediatevalue of impluse width is above 1.5ms, servo is clockwise rotation,conversely,anticlockwise.Rest point need use slotted screwdriver to adjust the positioner carefully.Servo stopped rotation when the input signal is equivalent to impluse width.
▲ Please choose correct model for your application.
Caution: Torque over-loaded will damage the servo's mechanism.
▲ Keep the servo clean and away from dust, corrosive gas and humid air.
▲ Without further notification when some parameters slightly amend for improving quality.







31150-MP

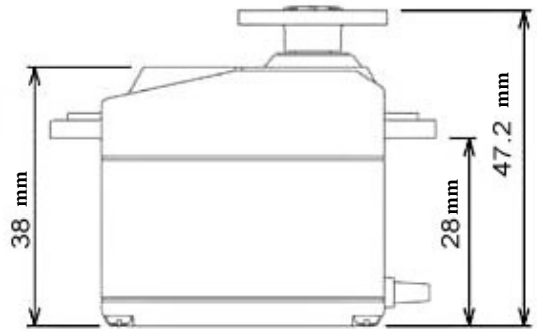
MG995 High Speed Servo Actuator

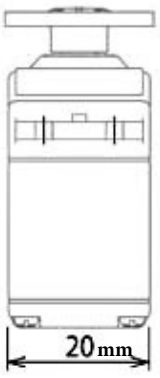
The unit comes complete with color coded 30cm wire leads with a 3 X 1 pin 0.1" Pitch type female header connector that matches most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

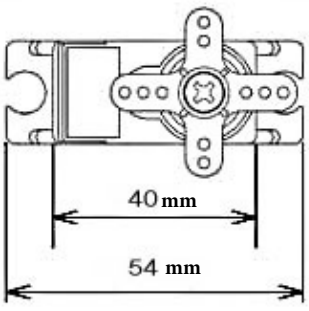
This high-speed servo actuator is not code dependant; You can use any servo code, hardware or library to control them. The MG995 Actuator includes arms and hardware to get started.

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Rotation Angle: 120deg. (+- 60 from center)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Operating voltage: 4.8 V to 7.2 V
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Metal Gears for longer life
- Temperature range: 0 ºC – 55 ºC

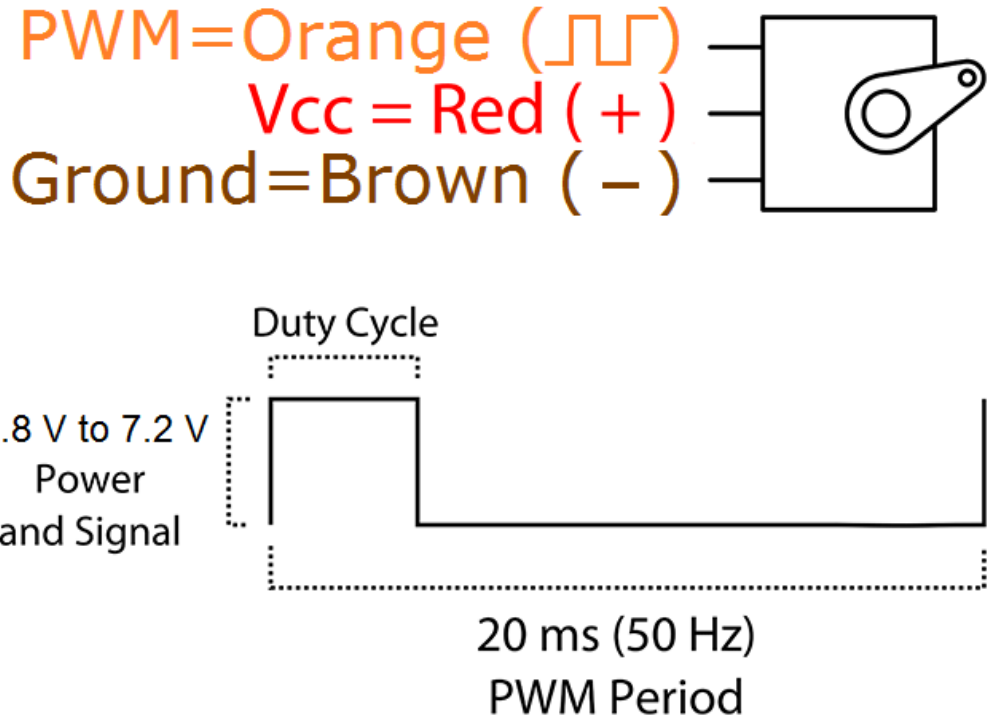






31150-MP

MG995 High Speed Servo Actuator



MARLIN P. JONES & ASSOC., INC.
P.O. Box 530400 Lake Park, Fl 33403
800-652-6733 FAX 561-844-8764
WWW.MPJA.COM

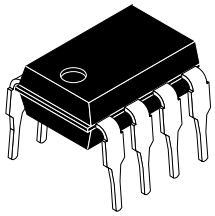
CNY74-2/CNY74-4

Multichannel Optocoupler with Phototransistor Output

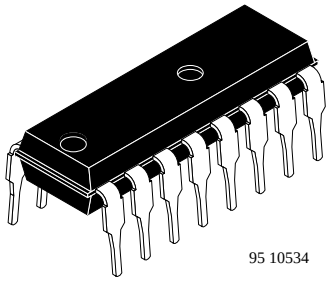
Description

The CNY74-2 and CNY74-4 consist of a phototransistor optically coupled to a gallium arsenide infrared emitting diode in a 8 lead, resp. 16 lead plastic dual inline packages.

The elements are mounted on one leadframe in coplanar technique, providing a fixed distance between input and output for highest safety requirements.



95 10828



95 10534

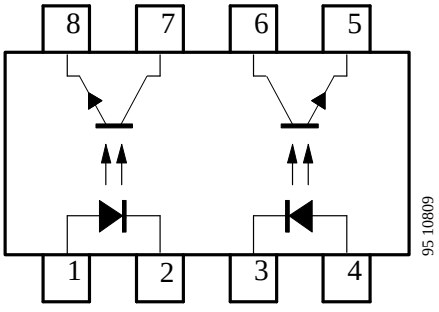
Applications

Galvanically separated circuits, non-interacting switches.

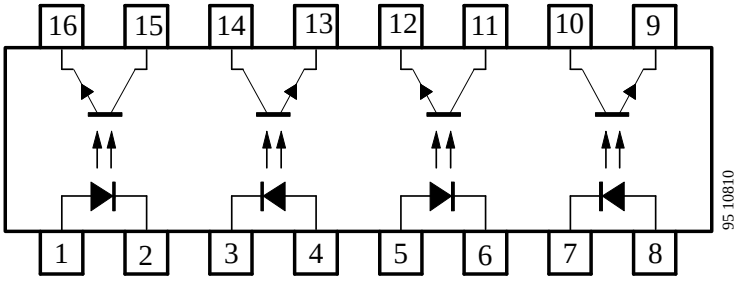
Features

- CNY74-2 includes 2 isolator channels
- CNY74-4 includes 4 isolator channels
- DC isolation test voltage V_{IO} = 2.5 kV
- Test class 25/100/21 DIN 40 045
- Low coupling capacitance typical 0.3 pF
- **C**urrent **T**ransfer **R**atio (CTR) typical 100%
- Low temperature coefficient of CTR
- Wide ambient temperature range

Pin Connections



CNY74-2



CNY74-4

Absolute Maximum Ratings

for single coupled system

Input (Emitter)

Parameters	Test Conditions	Symbol	Value	Unit
Reserve voltage		V_R	6	V
Forward current		I_F	60	mA
Forward surge current	$t_p \leq 10$ µs	I_{FSM}	1.5	A
Power dissipation	$T_{amb} \leq 25°C$	P_V	100	mW
Junction temperature		T_j	125	°C

Output (Detector)

Parameters	Test Conditions	Symbol	Value	Unit
Collector emitter voltage		V_{CEO}	70	V
Emitter collector voltage		V_{ECO}	7	V
Collector current		I_C	50	mA
Peak collector current	$t_p/T = 0.5$, $t_p \leq 10$ ms	I_{CM}	100	mA
Power dissipation	$T_{amb} \leq 25°C$	P_V	150	mW
Junction temperature		T_j	125	°C

Coupler

Parameters	Test Conditions	Symbol	Value	Unit
DC Isolation test voltage		V_{IO} 1)	2.5	V
Total power dissipation	$T_{amb} \leq 25°C$	P_{tot}	250	mW
Ambient temperature range		T_{amb}	–40 to +100	°C
Storage temperature range		T_{stg}	–55 to +125	°C
Soldering temperature	2 mm from case, $t \leq 10$ s	T_{sd}	260	°C

1) related to standard climate 23/50 DIN 50 014

Electrical Characteristics

for single coupled system, T_{amb} = 25°C

Input (Emitter)

Parameters	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Forward voltage	I_F = 50 mA	V_F		1.25	1.6	V
Breakdown voltage	I_R = 100 µA	$V_{(BR)}$	5			V

Output (Detector)

Parameters	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Collector emitter breakdown voltage	I_C = 1 mA	$V_{(BR)CEO}$	70			V
Emitter collector breakdown voltage	I_E = 100 µA	$V_{(BR)ECO}$	7			V
Collector dark current	V_{CE} = 20 V, I_F = 0, E = 0	I_{CEO}			100	nA

Coupler

Parameters	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
DC Isolation test voltage	t = 2 s	V_{IO} 1)	2.5			kV
Isolation resistance	V_{IO} = 1000 V, 40% relative humidity	R_{IO} 1)	10^{10}	10^{12}		Ω
Collector current	I_F = 5 mA, V_{CE} = 5 V I_F = 10 mA, V_{CE} = 5 V	I_C I_C	2.5 6	5 12	30	mA mA
I_C/I_F	I_F = 10 mA, V_{CE} = 5 V	CTR	0.5	1	6	
Collector emitter saturation voltage	I_F = 10 mA, I_C = 1 mA	V_{CEsat}			0.3	V
Cut-off frequency	V_{CE} = 5 V, I_F = 10 mA, R_L = 100 Ω	f_c		100		kHz
Coupling capacitances	f = 1 MHz	C_k		0.3		pF

1) related to standard climate 23/50 DIN 50 014

Switching Characteristics (Typical Values)

V_S = 5 V

Type	R_L = 100 kΩ, see figure KEIN MERKER							R_L = 1 kΩ, see fig. KEIN MERKER		
	t_d[µs]	t_r[µs]	t_{on}[µs]	t_s[µs]	t_f[µs]	t_{off}[µs]	I_C[mA]	t_{on}[µs]	t_{off}[µs]	I_F[mA]
CNY74-2/ CNY74-4	3.0	3.0	6.0	0.3	4.7	5.0	2	9	18	10

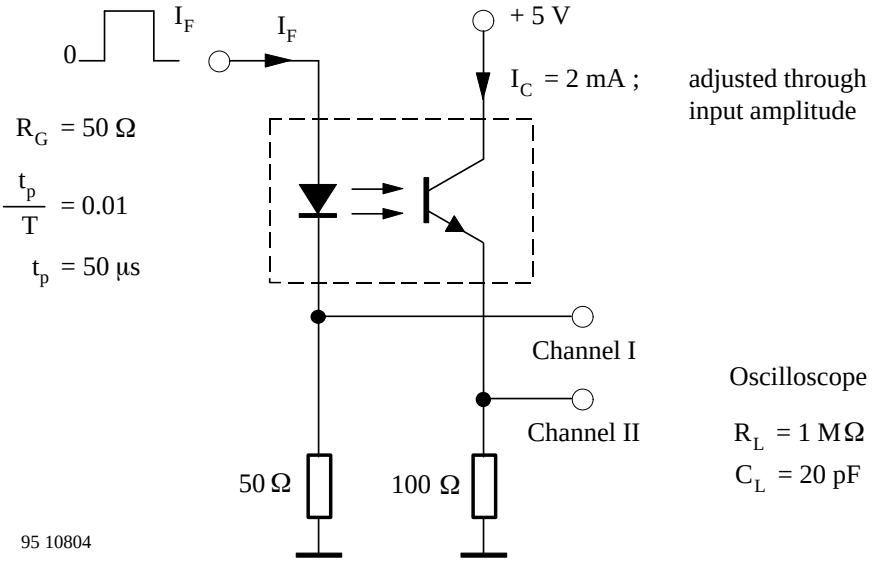


Figure 1. Test circuit, non saturated operation

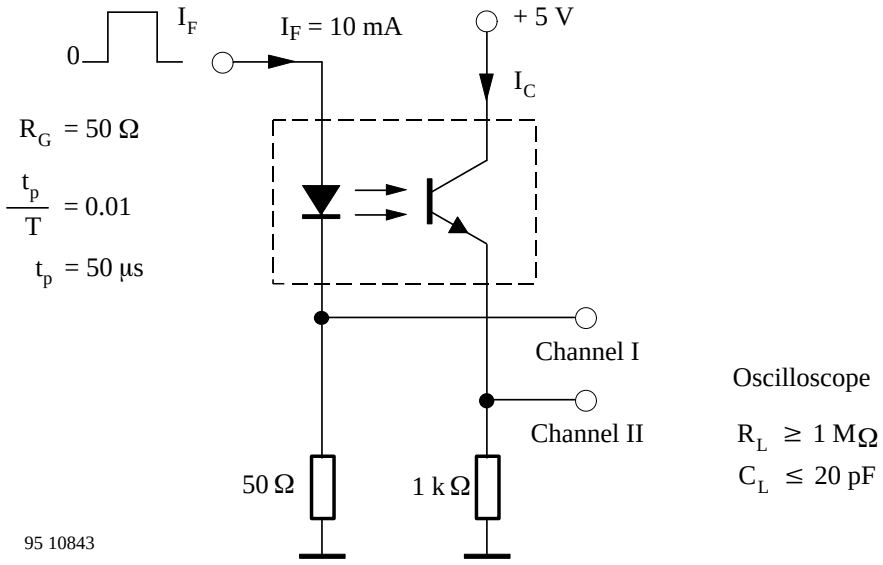


Figure 2. Test circuit, saturated operation

Ozone Depleting Substances Policy Statement

It is the policy of **TEMIC TELEFUNKEN microelectronic GmbH** to

1. Meet all present and future national and international statutory requirements.

2. Regularly and continuously improve the performance of our products, processes, distribution and operating systems with respect to their impact on the health and safety of our employees and the public, as well as their impact on the environment.

It is particular concern to control or eliminate releases of those substances into the atmosphere which are known as ozone depleting substances (ODSs).

The Montreal Protocol (1987) and its London Amendments (1990) intend to severely restrict the use of ODSs and forbid their use within the next ten years. Various national and international initiatives are pressing for an earlier ban on these substances.

TEMIC TELEFUNKEN microelectronic GmbH semiconductor division has been able to use its policy of continuous improvements to eliminate the use of ODSs listed in the following documents.

1. Annex A, B and list of transitional substances of the Montreal Protocol and the London Amendments respectively

2. Class I and II ozone depleting substances in the Clean Air Act Amendments of 1990 by the Environmental Protection Agency (EPA) in the USA

3. Council Decision 88/540/EEC and 91/690/EEC Annex A, B and C (transitional substances) respectively.

TEMIC can certify that our semiconductors are not manufactured with ozone depleting substances and do not contain such substances.

TEMIC TELEFUNKEN microelectronic GmbH, P.O.B. 3535, D-74025 Heilbronn, Germany
Telephone: 49 (0)7131 67 2831, Fax number: 49 (0)7131 67 2423

74HC595; 74HCT595

8-bit serial-in, serial or parallel-out shift register with output latches; 3-state

Rev. 6 — 12 December 2011 Product data sheet

1. General description

The 74HC595; 74HCT595 are high-speed Si-gate CMOS devices and are pin compatible with Low-power Schottky TTL (LSTTL). They are specified in compliance with JEDEC standard No. 7A.

The 74HC595; 74HCT595 are 8-stage serial shift registers with a storage register and 3-state outputs. The registers have separate clocks.

Data is shifted on the positive-going transitions of the shift register clock input (SHCP). The data in each register is transferred to the storage register on a positive-going transition of the storage register clock input (STCP). If both clocks are connected together, the shift register will always be one clock pulse ahead of the storage register.

The shift register has a serial input (DS) and a serial standard output (Q7S) for cascading. It is also provided with asynchronous reset (active LOW) for all 8 shift register stages. The storage register has 8 parallel 3-state bus driver outputs. Data in the storage register appears at the output whenever the output enable input ($\overline{OE}$) is LOW.

2. Features and benefits

- 8-bit serial input
- 8-bit serial or parallel output
- Storage register with 3-state outputs
- Shift register with direct clear
- 100 MHz (typical) shift out frequency
- ESD protection:
  - HBM JESD22-A114F exceeds 2000 V
  - MM JESD22-A115-A exceeds 200 V
- Multiple package options
- Specified from −40 °C to +85 °C and from −40 °C to +125 °C

3. Applications

- Serial-to-parallel data conversion
- Remote control holding register

4. Ordering information

Table 1. Ordering information

Type number	Package			
	Temperature range	Name	Description	Version
74HC595N	–40 °C to +125 °C	DIP16	plastic dual in-line package; 16 leads (300 mil)	SOT38-4
74HCT595N				
74HC595D	–40 °C to +125 °C	SO16	plastic small outline package; 16 leads; body width 3.9 mm	SOT109-1
74HCT595D				
74HC595DB	–40 °C to +125 °C	SSOP16	plastic shrink small outline package; 16 leads; body width 5.3 mm	SOT338-1
74HCT595DB				
74HC595PW	–40 °C to +125 °C	TSSOP16	plastic thin shrink small outline package; 16 leads; body width 4.4 mm	SOT403-1
74HCT595PW				
74HC595BQ	–40 °C to +125 °C	DHVQFN16	plastic dual in-line compatible thermal enhanced very thin quad flat package; no leads; 16 terminals; body 2.5 × 3.5 × 0.85 mm	SOT763-1
74HCT595BQ				

5. Functional diagram

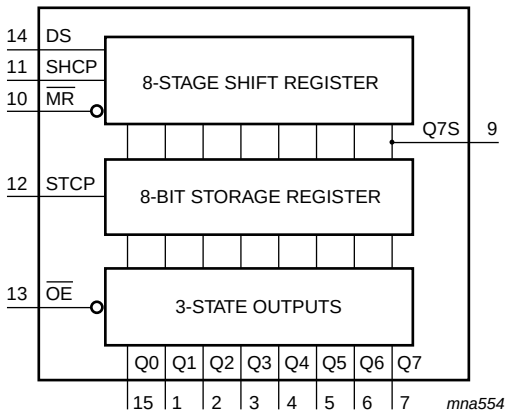


Fig 1. Functional diagram

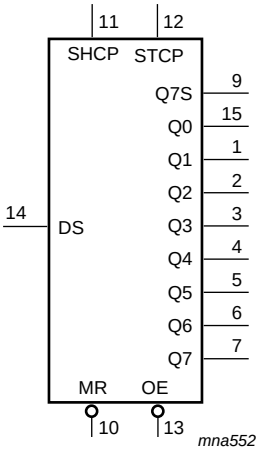


Fig 2. Logic symbol

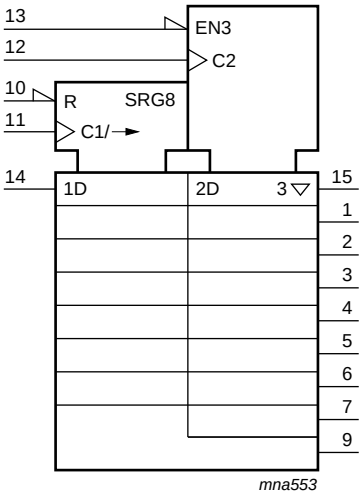


Fig 3. IEC logic symbol

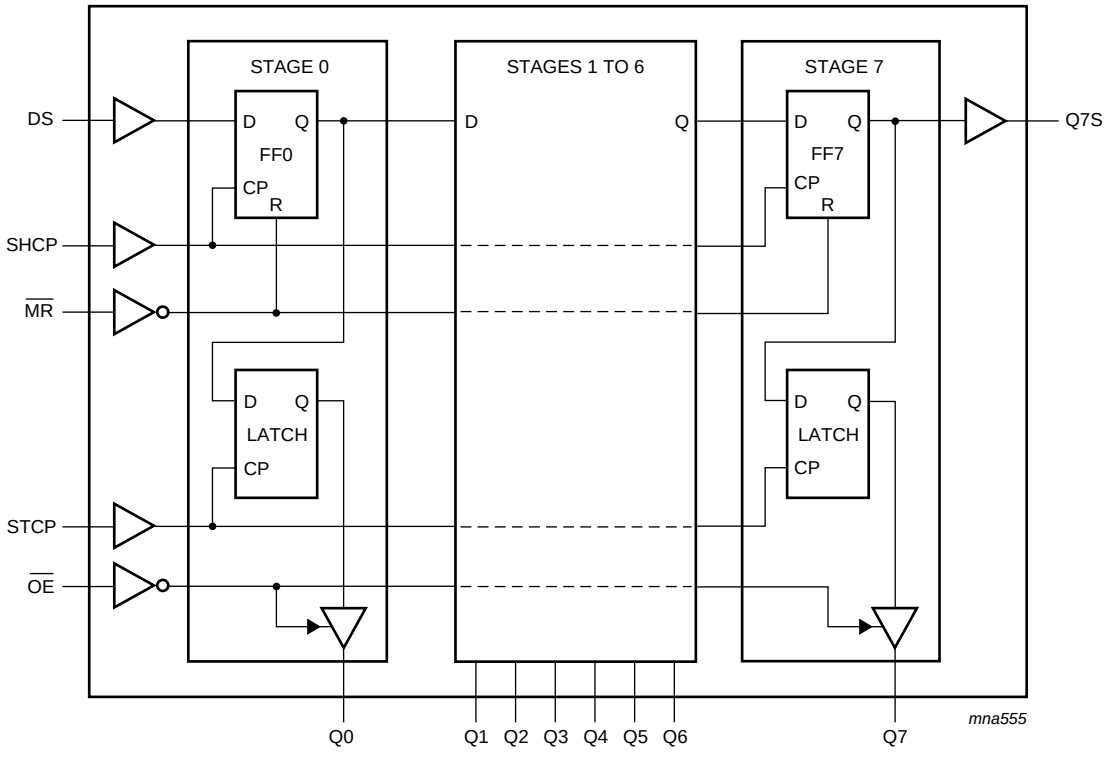


Fig 4. Logic diagram

6. Pinning information

6.1 Pinning

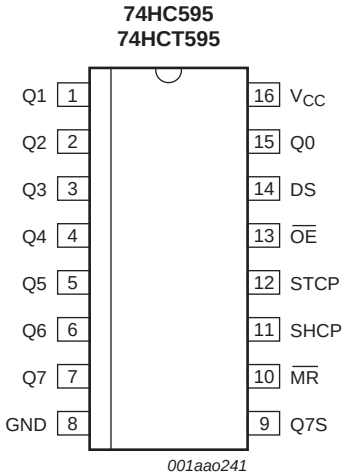


Fig 5. Pin configuration DIP16, SO16

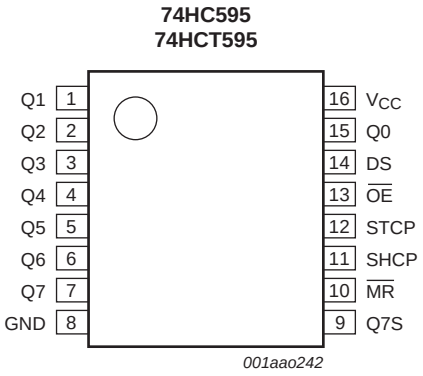
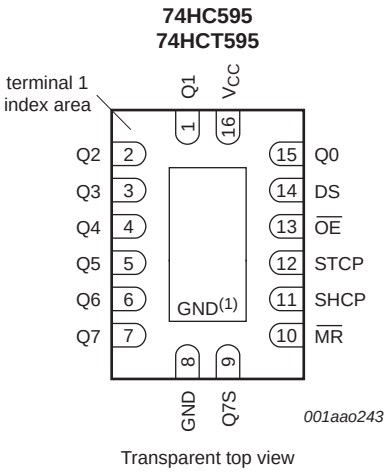


Fig 6. Pin configuration SSOP16, TSSOP16



Transparent top view

(1) This is not a supply pin, the substrate is attached to this pad using conductive die attach material. There is no electrical or mechanical requirement to solder this pad however if it is soldered the solder land should remain floating or be connected to GND.

Fig 7. Pin configuration for DHVQFN16

6.2 Pin description

Table 2. Pin description

Symbol	Pin	Description
Q1	1	parallel data output 1
Q2	2	parallel data output 2
Q3	3	parallel data output 3
Q4	4	parallel data output 4
Q5	5	parallel data output 5
Q6	6	parallel data output 6
Q7	7	parallel data output 7
GND	8	ground (0 V)
Q7S	9	serial data output
$\overline{MR}$	10	master reset (active LOW)
SHCP	11	shift register clock input
STCP	12	storage register clock input
$\overline{OE}$	13	output enable input (active LOW)
DS	14	serial data input
Q0	15	parallel data output 0
V_{CC}	16	supply voltage

7. Functional description

Table 3. Function table[1]

Control				Input	Output		Function
SHCP	STCP	$\overline{OE}$	$\overline{MR}$	DS	Q7S	Qn	
X	X	L	L	X	L	NC	a LOW-level on $\overline{MR}$ only affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6S	NC	logic HIGH-level shifted into shift register stage 0. Contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6S) appears on the serial output (Q7S).
X	↑	L	H	X	NC	QnS	contents of shift register stages (internal QnS) are transferred to the storage register and parallel output stages
↑	↑	L	H	X	Q6S	QnS	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

[1] H = HIGH voltage state;
L = LOW voltage state;
↑ = LOW-to-HIGH transition;
X = don't care;
NC = no change;
Z = high-impedance OFF-state.

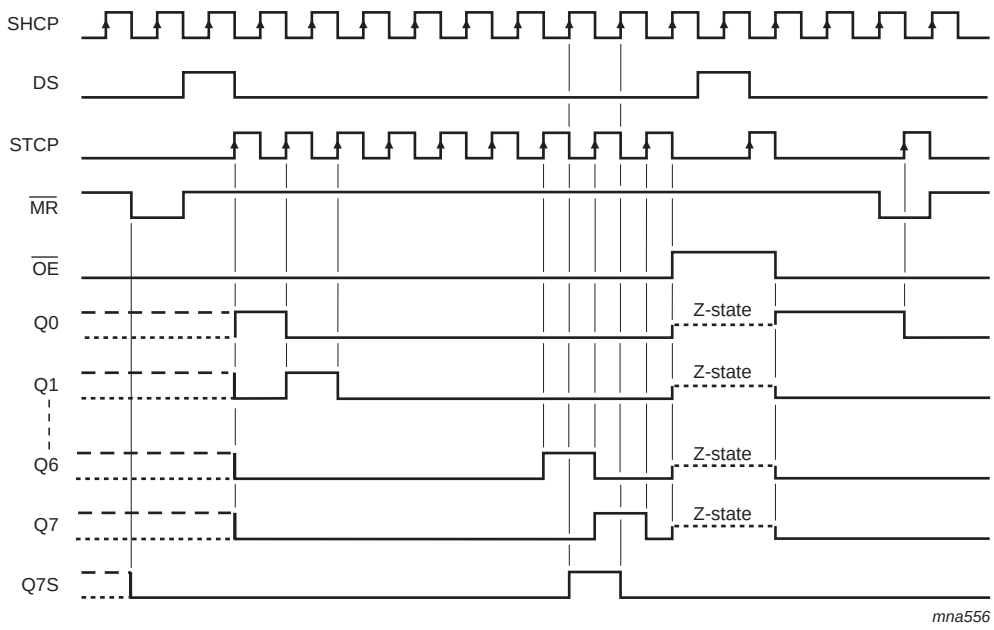


Fig 8. Timing diagram

8. Limiting values

Table 4. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134). Voltages are referenced to GND (ground = 0 V).

Symbol	Parameter	Conditions		Min	Max	Unit
V_{CC}	supply voltage			−0.5	+7	V
I_{IK}	input clamping current	$V_I < -0.5$ V or $V_I > V_{CC} + 0.5$ V		-	±20	mA
I_{OK}	output clamping current	$V_O < -0.5$ V or $V_O > V_{CC} + 0.5$ V		-	±20	mA
I_O	output current	$V_O = -0.5$ V to $(V_{CC} + 0.5$ V)				
		pin Q7S		-	±25	mA
		pins Qn		-	±35	mA
I_{CC}	supply current			-	70	mA
I_{GND}	ground current			−70	-	mA
T_{stg}	storage temperature			−65	+150	°C
P_{tot}	total power dissipation					
	DIP16 package		[1]	-	750	mW
	SO16 package		[2]	-	500	mW
	SSOP16 package		[3]	-	500	mW
	TSSOP16 package		[3]	-	500	mW
	DHVQFN16 package		[4]	-	500	mW

[1] For DIP16 package: P_{tot} derates linearly with 12 mW/K above 70 °C.

[2] For SO16 package: P_{tot} derates linearly with 8 mW/K above 70 °C.

[3] For SSOP16 and TSSOP16 packages: P_{tot} derates linearly with 5.5 mW/K above 60 °C.

[4] For DHVQFN16 package: P_{tot} derates linearly with 4.5 mW/K above 60 °C.

9. Recommended operating conditions

Table 5. Recommended operating conditions

Symbol	Parameter	Conditions	74HC595			74HCT595			Unit
			Min	Typ	Max	Min	Typ	Max	
V_{CC}	supply voltage		2.0	5.0	6.0	4.5	5.0	5.5	V
V_I	input voltage		0	-	V_{CC}	0	-	V_{CC}	V
V_O	output voltage		0	-	V_{CC}	0	-	V_{CC}	V
$\Delta t/\Delta V$	input transition rise and fall rate	V_{CC} = 2.0 V	-	-	625	-	-	-	ns/V
		V_{CC} = 4.5 V	-	1.67	139	-	1.67	139	ns/V
		V_{CC} = 6.0 V	-	-	83	-	-	-	ns/V
T_{amb}	ambient temperature		–40	+25	+125	–40	+25	+125	°C

10. Static characteristics

Table 6. Static characteristics

At recommended operating conditions; voltages are referenced to GND (ground = 0 V).

Symbol	Parameter	Conditions	–40 °C to +85 °C			–40 °C to +125 °C		Unit
			Min	Typ	Max	Min	Max	
74HC595								
V_{IH}	HIGH-level input voltage	V_{CC} = 2.0 V	1.5	1.2	-	1.5	-	V
		V_{CC} = 4.5 V	3.15	2.4	-	3.15	-	V
		V_{CC} = 6.0 V	4.2	3.2	-	4.2	-	V
V_{IL}	LOW-level input voltage	V_{CC} = 2.0 V	-	0.8	0.5	-	0.5	V
		V_{CC} = 4.5 V	-	2.1	1.35	-	1.35	V
		V_{CC} = 6.0 V	-	2.8	1.8	-	1.8	V
V_{OH}	HIGH-level output voltage	V_I = V_{IH} or V_{IL}						
		all outputs						
		I_O = –20 μA; V_{CC} = 2.0 V	1.9	2.0	-	1.9	-	V
		I_O = –20 μA; V_{CC} = 4.5 V	4.4	4.5	-	4.4	-	V
		I_O = –20 μA; V_{CC} = 6.0 V	5.9	6.0	-	5.9	-	V
		Q7S output						
		I_O = –4 mA; V_{CC} = 4.5 V	3.84	4.32	-	3.7	-	V
		I_O = –5.2 mA; V_{CC} = 6.0 V	5.34	5.81	-	5.2	-	V
		Qn bus driver outputs						
		I_O = –6 mA; V_{CC} = 4.5 V	3.84	4.32	-	3.7	-	V
		I_O = –7.8 mA; V_{CC} = 6.0 V	5.34	5.81	-	5.2	-	V

Table 6. Static characteristics *…continued*

At recommended operating conditions; voltages are referenced to GND (ground = 0 V).

Symbol	Parameter	Conditions	–40 °C to +85 °C			–40 °C to +125 °C		Unit
			Min	Typ	Max	Min	Max	
V_{OL}	LOW-level output voltage	$V_I = V_{IH}$ or V_{IL}						
		all outputs						
		I_O = 20 μA; V_{CC} = 2.0 V	-	0	0.1	-	0.1	V
		I_O = 20 μA; V_{CC} = 4.5 V	-	0	0.1	-	0.1	V
		I_O = 20 μA; V_{CC} = 6.0 V	-	0	0.1	-	0.1	V
		Q7S output						
		I_O = 4 mA; V_{CC} = 4.5 V	-	0.15	0.33	-	0.4	V
		I_O = 5.2 mA; V_{CC} = 6.0 V	-	0.16	0.33	-	0.4	V
		Qn bus driver outputs						
		I_O = 6 mA; V_{CC} = 4.5 V	-	0.15	0.33	-	0.4	V
		I_O = 7.8 mA; V_{CC} = 6.0 V	-	0.16	0.33	-	0.4	V
I_I	input leakage current	$V_I = V_{CC}$ or GND; V_{CC} = 6.0 V	-	-	±1.0	-	±1.0	μA
I_{OZ}	OFF-state output current	$V_I = V_{IH}$ or V_{IL}; V_{CC} = 6.0 V; $V_O = V_{CC}$ or GND	-	-	±5.0	-	±10	μA
I_{CC}	supply current	$V_I = V_{CC}$ or GND; I_O = 0 A; V_{CC} = 6.0 V	-	-	80	-	160	μA
C_I	input capacitance		-	3.5	-	-	-	pF
74HCT595								
V_{IH}	HIGH-level input voltage	V_{CC} = 4.5 V to 5.5 V	2.0	1.6	-	2.0	-	V
V_{IL}	LOW-level input voltage	V_{CC} = 4.5 V to 5.5 V	-	1.2	0.8	-	0.8	V
V_{OH}	HIGH-level output voltage	$V_I = V_{IH}$ or V_{IL}; V_{CC} = 4.5 V						
		all outputs						
		I_O = –20 μA	4.4	4.5	-	4.4	-	V
		Q7S output						
		I_O = –4 mA	3.84	4.32	-	3.7	-	V
		Qn bus driver outputs						
		I_O = –6 mA	3.7	4.32	-	3.7	-	V
V_{OL}	LOW-level output voltage	$V_I = V_{IH}$ or V_{IL}; V_{CC} = 4.5 V						
		all outputs						
		I_O = 20 μA	-	0	0.1	-	0.1	V
		Q7S output						
		I_O = 4.0 mA	-	0.15	0.33	-	0.4	V
		Qn bus driver outputs						
		I_O = 6.0 mA	-	0.16	0.33	-	0.4	V
I_I	input leakage current	$V_I = V_{CC}$ or GND; V_{CC} = 5.5 V	-	-	±1.0	-	±1.0	μA

Table 6. Static characteristics *…continued*

At recommended operating conditions; voltages are referenced to GND (ground = 0 V).

Symbol	Parameter	Conditions	–40 °C to +85 °C			–40 °C to +125 °C		Unit
			Min	Typ	Max	Min	Max	
I_{OZ}	OFF-state output current	$V_I = V_{IH}$ or V_{IL}; V_{CC} = 5.5 V; $V_O = V_{CC}$ or GND	-	-	±5.0	-	±10	μA
I_{CC}	supply current	$V_I = V_{CC}$ or GND; I_O = 0 A; V_{CC} = 5.5 V	-	-	80	-	160	μA
ΔI_{CC}	additional supply current	per input pin; I_O = 0 A; $V_I = V_{CC}$ − 2.1 V; other inputs at V_{CC} or GND; V_{CC} = 4.5 V to 5.5 V						
		pins $\overline{MR}$, SHCP, STCP, $\overline{OE}$	-	150	675	-	735	μA
		pin DS	-	25	113	-	123	μA
C_I	input capacitance		-	3.5	-	-	-	pF

11. Dynamic characteristics

Table 7. Dynamic characteristics

Voltages are referenced to GND (ground = 0 V); for test circuit see Figure 14.

Symbol	Parameter	Conditions		25 °C			–40 °C to +85 °C		–40 °C to +125 °C		Unit
				Min	Typ[1]	Max	Min	Max	Min	Max	
74HC595											
t_{pd}	propagation delay	SHCP to Q7S; see Figure 9	[2]								
		V_{CC} = 2 V		-	52	160	-	200	-	240	ns
		V_{CC} = 4.5 V		-	19	32	-	40	-	48	ns
		V_{CC} = 6 V		-	15	27	-	34	-	41	ns
		STCP to Qn; see Figure 10	[2]								
		V_{CC} = 2 V		-	55	175	-	220	-	265	ns
		V_{CC} = 4.5 V		-	20	35	-	44	-	53	ns
		V_{CC} = 6 V		-	16	30	-	37	-	45	ns
		$\overline{MR}$ to Q7S; see Figure 12	[3]								
		V_{CC} = 2 V		-	47	175	-	220	-	265	ns
		V_{CC} = 4.5 V		-	17	35	-	44	-	53	ns
		V_{CC} = 6 V		-	14	30	-	37	-	45	ns
t_{en}	enable time	$\overline{OE}$ to Qn; see Figure 13	[4]								
		V_{CC} = 2 V		-	47	150	-	190	-	225	ns
		V_{CC} = 4.5 V		-	17	30	-	38	-	45	ns
		V_{CC} = 6 V		-	14	26	-	33	-	38	ns
t_{dis}	disable time	$\overline{OE}$ to Qn; see Figure 13	[5]								
		V_{CC} = 2 V		-	41	150	-	190	-	225	ns
		V_{CC} = 4.5 V		-	15	30	-	38	-	45	ns
		V_{CC} = 6 V		-	12	27	-	33	-	38	ns
t_W	pulse width	SHCP HIGH or LOW; see Figure 9									
		V_{CC} = 2 V		75	17	-	95	-	110	-	ns
		V_{CC} = 4.5 V		15	6	-	19	-	22	-	ns
		V_{CC} = 6 V		13	5	-	16	-	19	-	ns
		STCP HIGH or LOW; see Figure 10									
		V_{CC} = 2 V		75	11	-	95	-	110	-	ns
		V_{CC} = 4.5 V		15	4	-	19	-	22	-	ns
		V_{CC} = 6 V		13	3	-	16	-	19	-	ns
		$\overline{MR}$ LOW; see Figure 12									
		V_{CC} = 2 V		75	17	-	95	-	110	-	ns
		V_{CC} = 4.5 V		15	6	-	19	-	22	-	ns
		V_{CC} = 6 V		13	5	-	16	-	19	-	ns

Table 7. Dynamic characteristics *...continued*

Voltages are referenced to GND (ground = 0 V); for test circuit see Figure 14.

Symbol	Parameter	Conditions		25 °C			–40 °C to +85 °C		–40 °C to +125 °C		Unit
				Min	Typ[1]	Max	Min	Max	Min	Max	
t_{su}	set-up time	DS to SHCP; see Figure 10									
		V_{CC} = 2 V		50	11	-	65	-	75	-	ns
		V_{CC} = 4.5 V		10	4	-	13	-	15	-	ns
		V_{CC} = 6 V		9	3	-	11	-	13	-	ns
		SHCP to STCP; see Figure 11									
		V_{CC} = 2 V		75	22	-	95	-	110	-	ns
		V_{CC} = 4.5 V		15	8	-	19	-	22	-	ns
		V_{CC} = 6 V		13	7	-	16	-	19	-	ns
t_h	hold time	DS to SHCP; see Figure 11									
		V_{CC} = 2 V		3	–6	-	3	-	3	-	ns
		V_{CC} = 4.5 V		3	–2	-	3	-	3	-	ns
		V_{CC} = 6 V		3	–2	-	3	-	3	-	ns
t_{rec}	recovery time	$\overline{MR}$ to SHCP; see Figure 12									
		V_{CC} = 2 V		50	–19	-	65	-	75	-	ns
		V_{CC} = 4.5 V		10	–7	-	13	-	15	-	ns
		V_{CC} = 6 V		9	–6	-	11	-	13	-	ns
f_{max}	maximum frequency	SHCP or STCP; see Figure 9 and 10									
		V_{CC} = 2 V		9	30	-	4.8	-	4	-	MHz
		V_{CC} = 4.5 V		30	91	-	24	-	20	-	MHz
		V_{CC} = 6 V		35	108	-	28	-	24	-	MHz
C_{PD}	power dissipation capacitance	f_i = 1 MHz; V_I = GND to V_{CC}	[6][7]	-	115	-	-	-	-	-	pF
74HCT595; V_{CC} = 4.5 V to 5.5 V											
t_{pd}	propagation delay	SHCP to Q7S; see Figure 9	[2]	-	25	42	-	53	-	63	ns
		STCP to Qn; see Figure 10	[2]	-	24	40	-	50	-	60	ns
		$\overline{MR}$ to Q7S; see Figure 12	[3]	-	23	40	-	50	-	60	ns
t_{en}	enable time	$\overline{OE}$ to Qn; see Figure 13	[4]	-	21	35	-	44	-	53	ns
t_{dis}	disable time	$\overline{OE}$ to Qn; see Figure 13	[5]	-	18	30	-	38	-	45	ns
t_W	pulse width	SHCP HIGH or LOW; see Figure 9		16	6	-	20	-	24	-	ns
		STCP HIGH or LOW; see Figure 10		16	5	-	20	-	24	-	ns
		$\overline{MR}$ LOW; see Figure 12		20	8	-	25	-	30	-	ns
t_{su}	set-up time	DS to SHCP; see Figure 10		16	5	-	20	-	24	-	ns
		SHCP to STCP; see Figure 11		16	8	-	20	-	24	-	ns
t_h	hold time	DS to SHCP; see Figure 11		3	–2	-	3	-	3	-	ns

Table 7. Dynamic characteristics *...continued*

Voltages are referenced to GND (ground = 0 V); for test circuit see Figure 14.

Symbol	Parameter	Conditions	25 °C			–40 °C to +85 °C		–40 °C to +125 °C		Unit
			Min	Typ[1]	Max	Min	Max	Min	Max	
t_{rec}	recovery time	$\overline{MR}$ to SHCP; see Figure 12	10	–7	-	13	-	15	-	ns
f_{max}	maximum frequency	SHCP and STCP; see Figure 9 and 10	30	52	-	24	-	20	-	MHz
C_{PD}	power dissipation capacitance	f_i = 1 MHz; V_I = GND to V_{CC} [6] [7]	-	130	-	-	-	-	-	pF

[1] Typical values are measured at nominal supply voltage.

[2] t_{pd} is the same as t_{PHL} and t_{PLH}.

[3] t_{pd} is the same as t_{PHL} only.

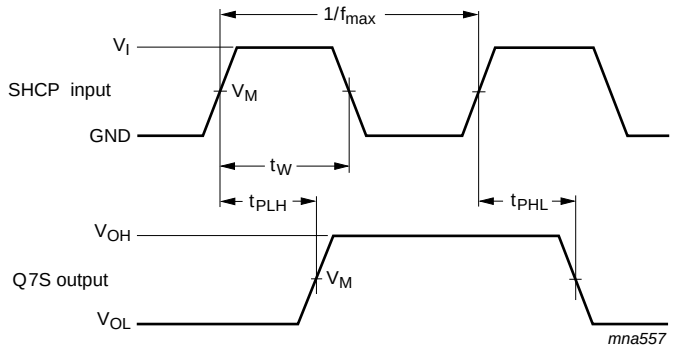
[4] t_{en} is the same as t_{PZL} and t_{PZH}.

[5] t_{dis} is the same as t_{PLZ} and t_{PHZ}.

[6] C_{PD} is used to determine the dynamic power dissipation (P_D in μW).
$P_D = C_{PD} \times V_{CC}^2 \times f_i + \Sigma(C_L \times V_{CC}^2 \times f_o)$ where:
f_i = input frequency in MHz;
f_o = output frequency in MHz;
$\Sigma(C_L \times V_{CC}^2 \times f_o)$ = sum of outputs;
C_L = output load capacitance in pF;
V_{CC} = supply voltage in V.

[7] All 9 outputs switching.

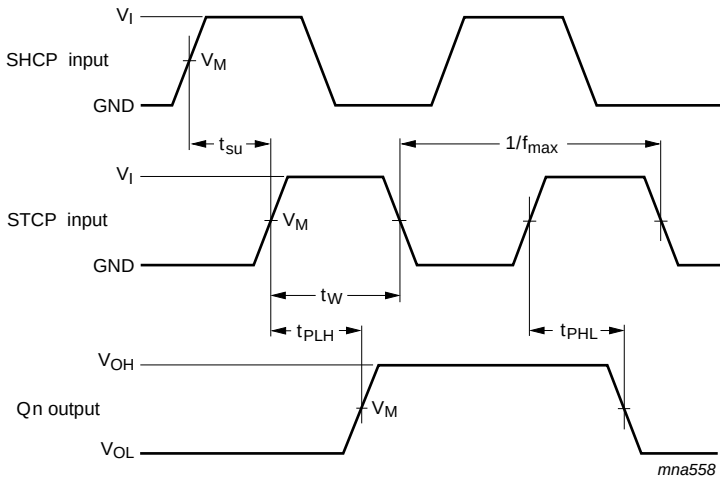
12. Waveforms



Measurement points are given in Table 8.

V_{OL} and V_{OH} are typical output voltage levels that occur with the output load.

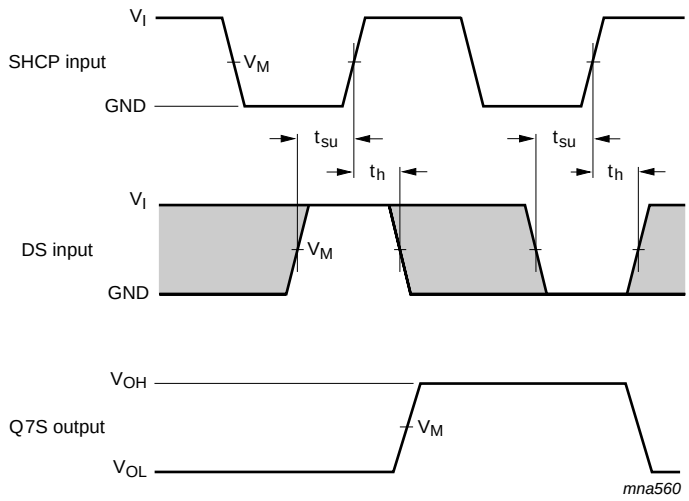
Fig 9. Shift clock pulse, maximum frequency and input to output propagation delays



Measurement points are given in Table 8.

V_{OL} and V_{OH} are typical output voltage levels that occur with the output load.

Fig 10. Storage clock to output propagation delays

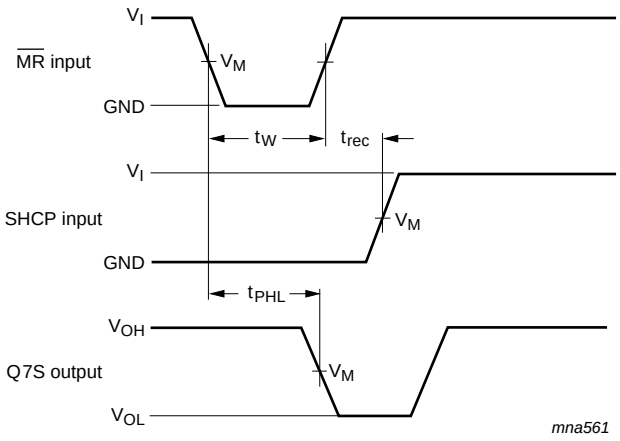


Measurement points are given in Table 8.

The shaded areas indicate when the input is permitted to change for predictable output performance.

V_{OL} and V_{OH} are typical output voltage levels that occur with the output load.

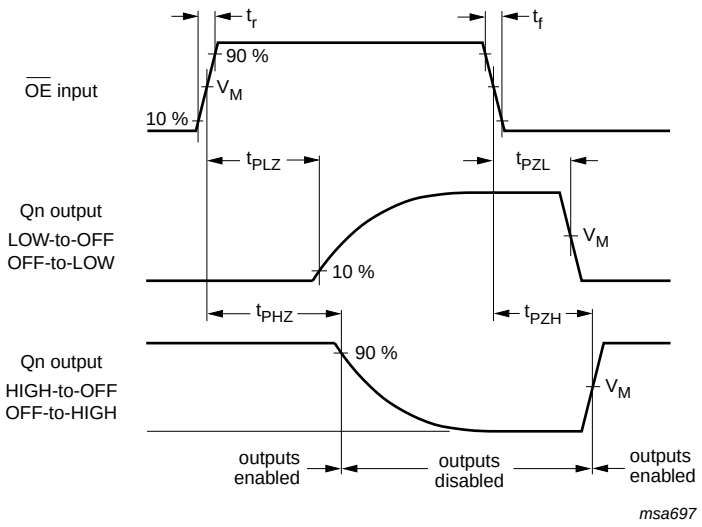
Fig 11. Data set-up and hold times



Measurement points are given in Table 8.

V_{OL} and V_{OH} are typical output voltage levels that occur with the output load.

Fig 12. Master reset to output propagation delays



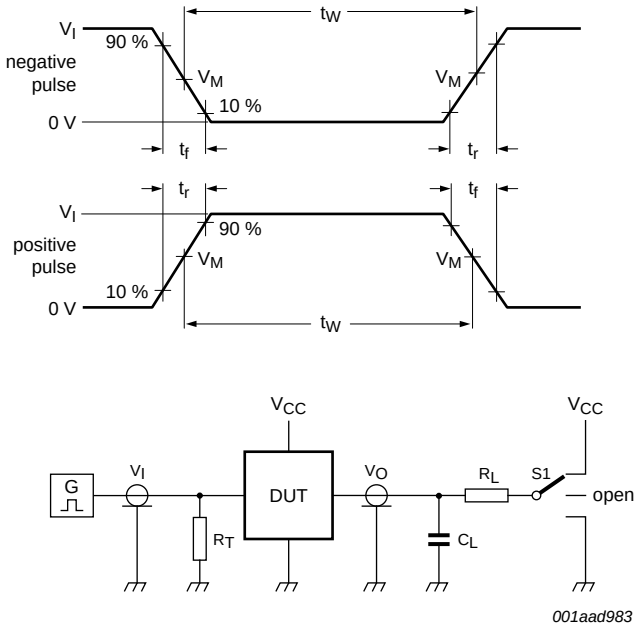
Measurement points are given in Table 8.

V_{OL} and V_{OH} are typical output voltage levels that occur with the output load.

Fig 13. Enable and disable times

Table 8. Measurement points

Type	Input	Output
	V_M	V_M
74HC595	$0.5V_{CC}$	$0.5V_{CC}$
74HCT595	1.3 V	1.3 V



Test data is given in Table 9.

Definitions for test circuit:

C_L = load capacitance including jig and probe capacitance.

R_L = load resistance.

R_T = termination resistance should be equal to the output impedance Z_o of the pulse generator.

S1 = test selection switch.

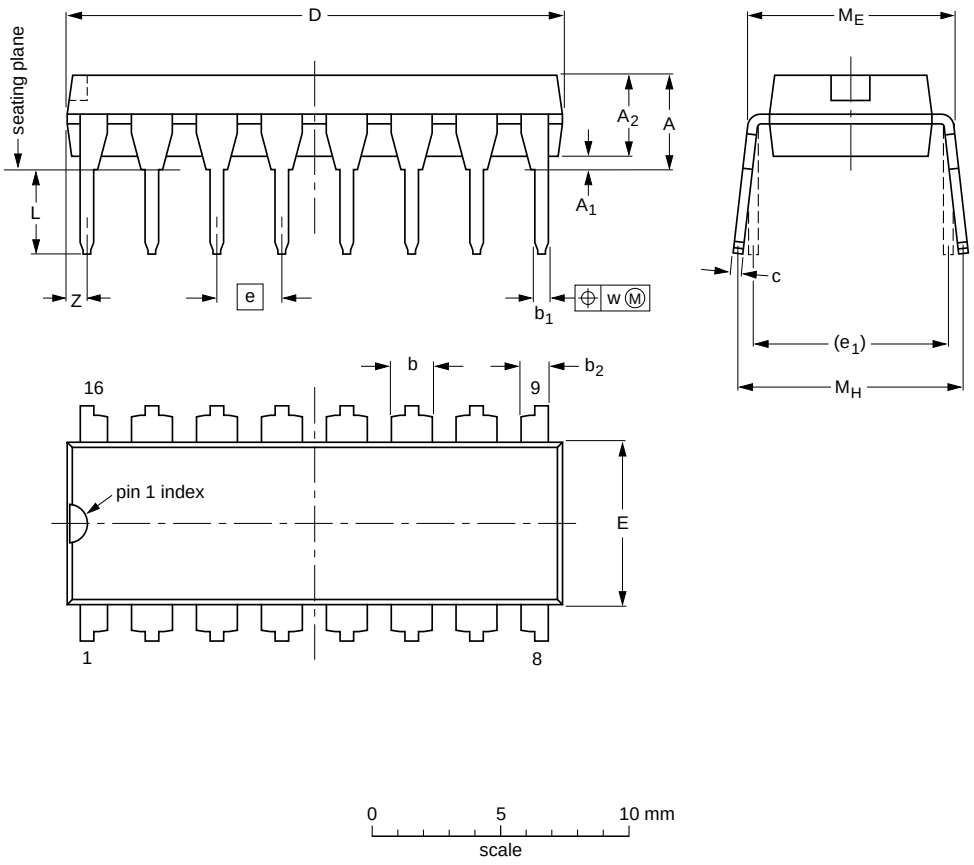
Fig 14. Test circuit for measuring switching times

Table 9. Test data

Type	Input		Load		S1 position		
	V_I	t_r, t_f	C_L	R_L	t_{PHL}, t_{PLH}	t_{PZH}, t_{PHZ}	t_{PZL}, t_{PLZ}
74HC595	V_{CC}	6 ns	50 pF	1 kΩ	open	GND	V_{CC}
74HCT595	3 V	6 ns	50 pF	1 kΩ	open	GND	V_{CC}

13. Package outline

DIP16: plastic dual in-line package; 16 leads (300 mil) **SOT38-4**



DIMENSIONS (inch dimensions are derived from the original mm dimensions)

UNIT	A max.	A_1 min.	A_2 max.	b	b_1	b_2	c	D (1)	E (1)	e	e_1	L	M_E	M_H	w	Z (1) max.
mm	4.2	0.51	3.2	1.73 1.30	0.53 0.38	1.25 0.85	0.36 0.23	19.50 18.55	6.48 6.20	2.54	7.62	3.60 3.05	8.25 7.80	10.0 8.3	0.254	0.76
inches	0.17	0.02	0.13	0.068 0.051	0.021 0.015	0.049 0.033	0.014 0.009	0.77 0.73	0.26 0.24	0.1	0.3	0.14 0.12	0.32 0.31	0.39 0.33	0.01	0.03

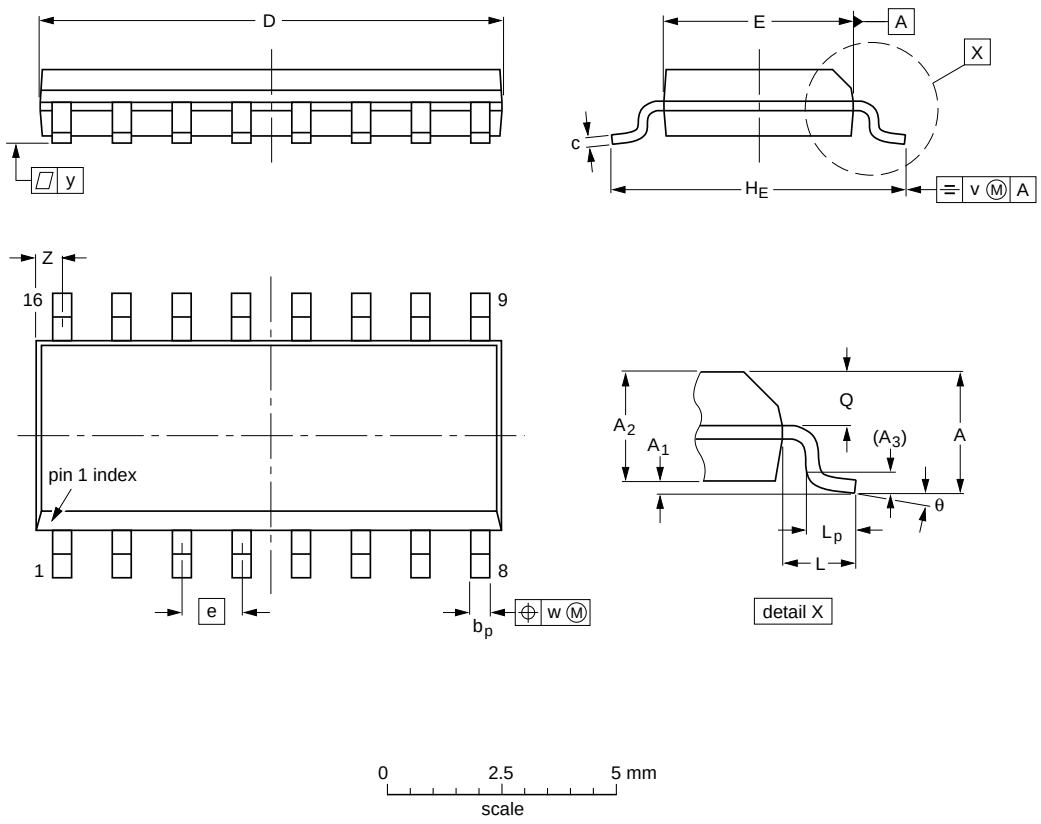
Note

1. Plastic or metal protrusions of 0.25 mm (0.01 inch) maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT38-4						~~95-01-14~~ 03-02-13

Fig 15. Package outline SOT38-4 (DIP16)

SO16: plastic small outline package; 16 leads; body width 3.9 mm **SOT109-1**



DIMENSIONS (inch dimensions are derived from the original mm dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_E	L	L_p	Q	v	w	y	$Z^{(1)}$	θ
mm	1.75	0.25 0.10	1.45 1.25	0.25	0.49 0.36	0.25 0.19	10.0 9.8	4.0 3.8	1.27	6.2 5.8	1.05	1.0 0.4	0.7 0.6	0.25	0.25	0.1	0.7 0.3	8^o 0^o
inches	0.069	0.010 0.004	0.057 0.049	0.01	0.019 0.014	0.0100 0.0075	0.39 0.38	0.16 0.15	0.05	0.244 0.228	0.041	0.039 0.016	0.028 0.020	0.01	0.01	0.004	0.028 0.012	

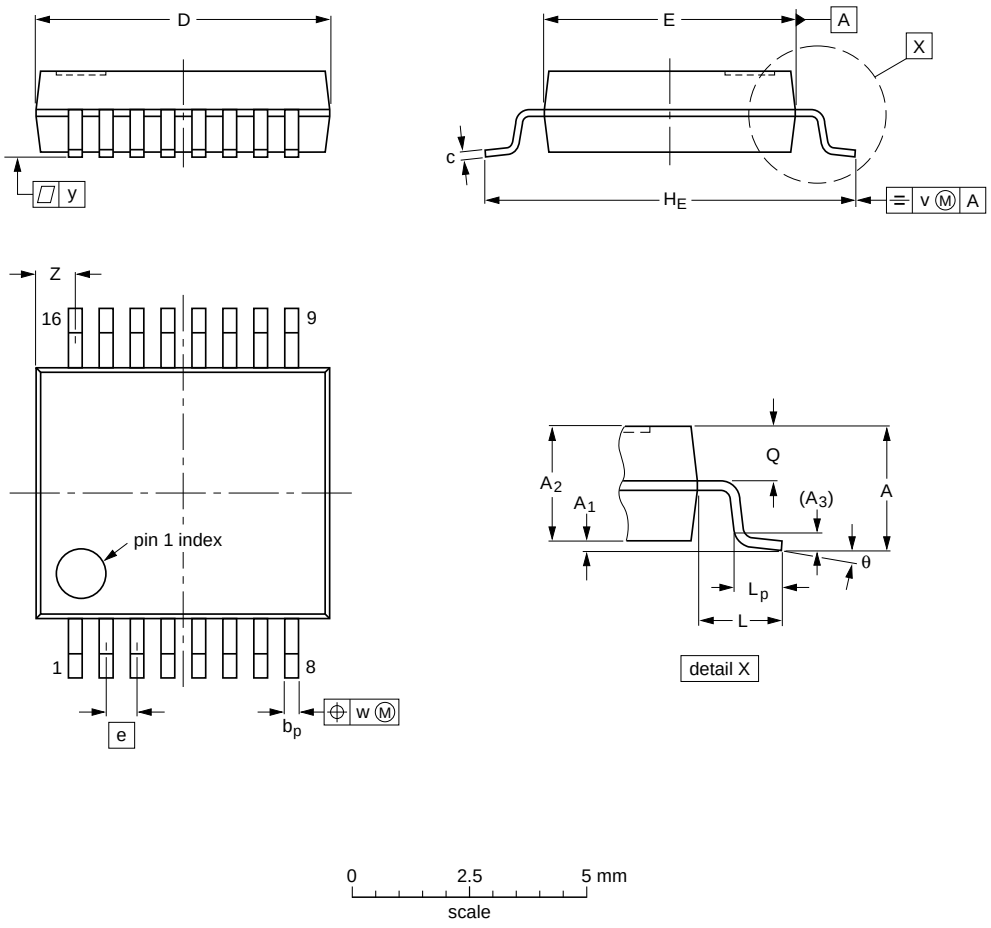
Note

1. Plastic or metal protrusions of 0.15 mm (0.006 inch) maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT109-1	076E07	MS-012				~~99-12-27~~ 03-02-19

Fig 16. Package outline SOT109-1 (SO16)

SSOP16: plastic shrink small outline package; 16 leads; body width 5.3 mm **SOT338-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	D (1)	E (1)	e	H_E	L	L_p	Q	v	w	y	Z (1)	θ
mm	2	0.21 0.05	1.80 1.65	0.25	0.38 0.25	0.20 0.09	6.4 6.0	5.4 5.2	0.65	7.9 7.6	1.25	1.03 0.63	0.9 0.7	0.2	0.13	0.1	1.00 0.55	8^o 0^o

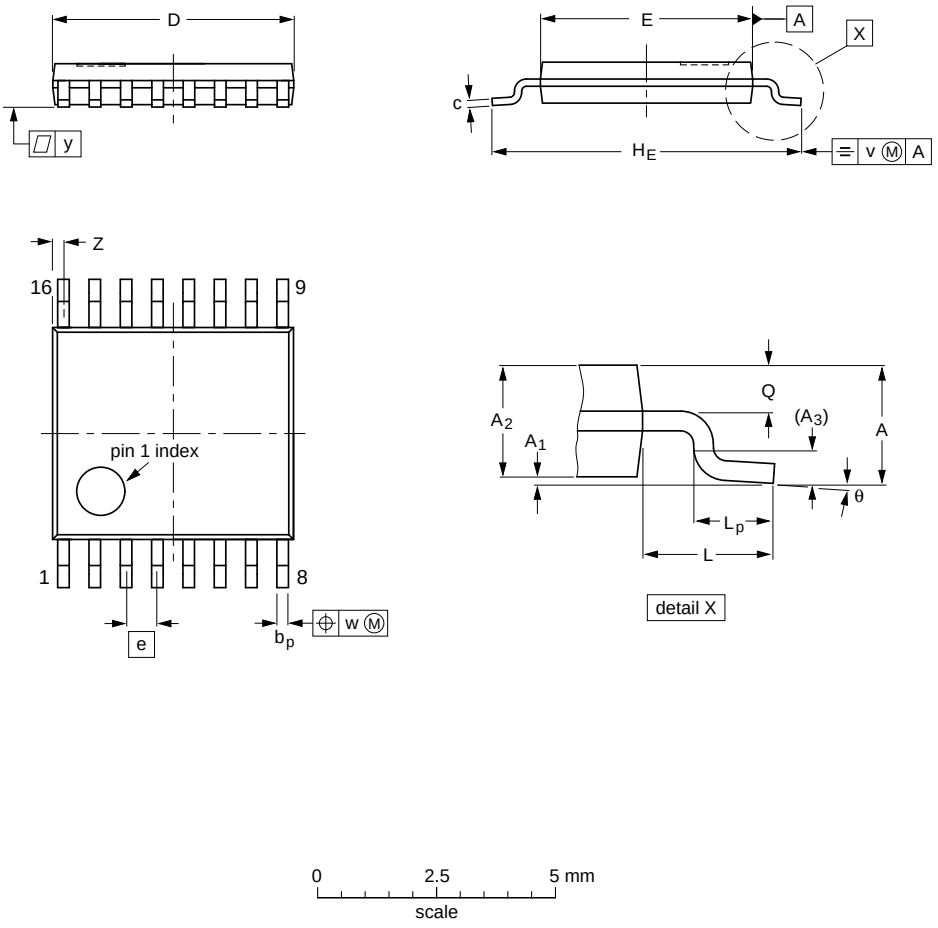
Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT338-1		MO-150				~~99-12-27~~ 03-02-19

Fig 17. Package outline SOT338-1 (SSOP16)

TSSOP16: plastic thin shrink small outline package; 16 leads; body width 4.4 mm **SOT403-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	D (1)	E (2)	e	H_E	L	L_p	Q	v	w	y	Z (1)	θ
mm	1.1	0.15 0.05	0.95 0.80	0.25	0.30 0.19	0.2 0.1	5.1 4.9	4.5 4.3	0.65	6.6 6.2	1	0.75 0.50	0.4 0.3	0.2	0.13	0.1	0.40 0.06	8^o 0^o

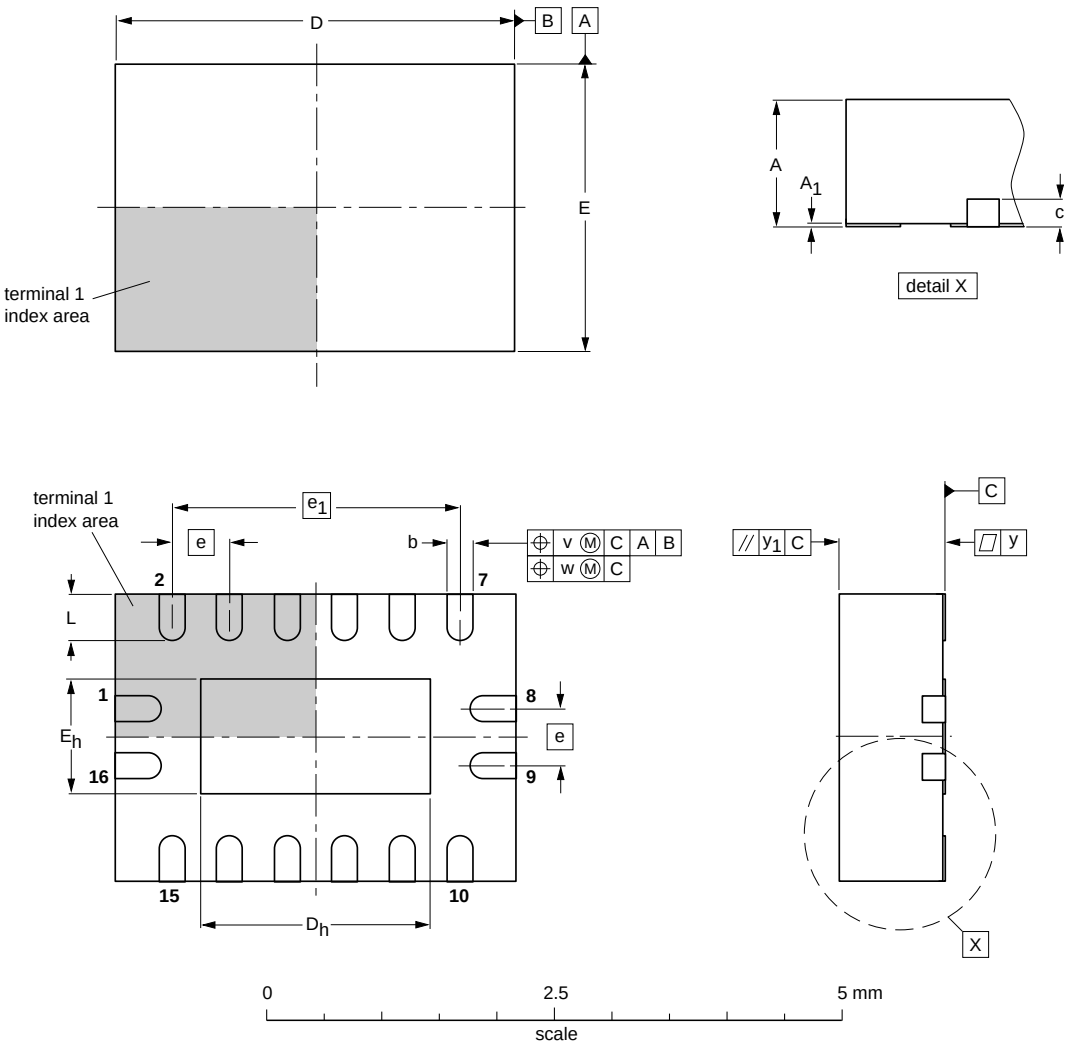
Notes

1. Plastic or metal protrusions of 0.15 mm maximum per side are not included.
2. Plastic interlead protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT403-1		MO-153				~~99-12-27~~ 03-02-18

Fig 18. Package outline SOT403-1 (TSSOP16)

DHVQFN16: plastic dual in-line compatible thermal enhanced very thin quad flat package; no leads; 16 terminals; body 2.5 x 3.5 x 0.85 mm **SOT763-1**



DIMENSIONS (mm are the original dimensions)

UNIT	$A^{(1)}$ max.	A_1	b	c	$D^{(1)}$	D_h	$E^{(1)}$	E_h	e	e_1	L	v	w	y	y_1
mm	1	0.05 0.00	0.30 0.18	0.2	3.6 3.4	2.15 1.85	2.6 2.4	1.15 0.85	0.5	2.5	0.5 0.3	0.1	0.05	0.05	0.1

Note

1. Plastic or metal protrusions of 0.075 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT763-1	- - -	MO-241	- - -			~~02-10-17~~ 03-01-27

Fig 19. Package outline SOT763-1 (DHVQFN16)

14. Abbreviations

Table 10. Abbreviations

Acronym	Abbreviation
CMOS	Complementary Metal Oxide Semiconductor
DUT	Device Under Test
ESD	ElectroStatic Discharge
HBM	Human Body Model
LSTTL	Low-power Schottky Transistor-Transistor Logic
MM	Machine Model

15. Revision history

Table 11. Revision history

Document ID	Release date	Data sheet status	Change notice	Supersedes
74HC_HCT595 v.6	20111212	Product data sheet	-	74HC_HCT595 v.5
Modifications:	• Legal pages updated.			
74HC_HCT595 v.5	20110628	Product data sheet	-	74HC_HCT595 v.4
74HC_HCT595 v.4	20030604	Product specification	-	74HC_HCT595_CNV v.3
74HC_HCT595_CNV v.3	19980604	Product specification	-	-

16. Legal information

16.1 Data sheet status

Document status[1][2]	Product status[3]	Definition
Objective [short] data sheet	Development	This document contains data from the objective specification for product development.
Preliminary [short] data sheet	Qualification	This document contains data from the preliminary specification.
Product [short] data sheet	Production	This document contains the product specification.

[1] Please consult the most recently issued document before initiating or completing a design.

[2] The term 'short data sheet' is explained in section "Definitions".

[3] The product status of device(s) described in this document may have changed since this document was published and may differ in case of multiple devices. The latest product status information is available on the Internet at URL http://www.nxp.com.

16.2 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

Short data sheet — A short data sheet is an extract from a full data sheet with the same product type number(s) and title. A short data sheet is intended for quick reference only and should not be relied upon to contain detailed and full information. For detailed and full information see the relevant full data sheet, which is available on request via the local NXP Semiconductors sales office. In case of any inconsistency or conflict with the short data sheet, the full data sheet shall prevail.

Product specification — The information and data provided in a Product data sheet shall define the specification of the product as agreed between NXP Semiconductors and its customer, unless NXP Semiconductors and customer have explicitly agreed otherwise in writing. In no event however, shall an agreement be valid in which the NXP Semiconductors product is deemed to offer functions and qualities beyond those described in the Product data sheet.

16.3 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

16.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

17. Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

18. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 12 December 2011
Document identifier: 74HC_HCT595

Robotics

IRB 140
Industrial Robot

Main Applications

Arc welding
Assembly
Cleaning/Spraying
Machine tending
Material handling
Packing
Deburring

Small, Powerful and Fast

Compact, powerful IRB 140 industrial robot.
Six axis multipurpose robot that handles payload of 6 kg, with long reach (810 mm). The IRB 140 can be floor mounted, inverted or on the wall in any angle. Available as Standard, Foundry Plus 2, Clean Room and Wash versions, all mechanical arms completely IP67 protected, making IRB 140 easy to integrate in and suitable for a variety of applications. Uniquely extended radius of working area due to bend-back mechanism of upper arm, axis 1 rotation of 360 degrees even as wall mounted.
The compact, robust design with integrated cabling adds to overall flexibility. The Collision Detection option with full path retraction makes robot reliable and safe.

Using IRB 140T, cycle-times are considerably reduced where axis 1 and 2 predominantly are used.
Reductions between 15-20 % are possible using pure axis 1 and 2 movements. This faster versions is well suited for packing applications and guided operations together with PickMaster.
IRB 140 Foundry Plus 2 and Wash versions are suitable for operating in extreme foundry environments and other harch environments with high requirements on corrosion resistance and tightness. In addition to the IP67 protection, excellent surface treatment makes the robot high pressure steam washable. Also available in white Clean Room ISO class 6 version, making it especially suited for environments with stringent cleanliness standards.

IRB 140

Specification

Robot versions	Handling capacity	Reach of 5th axis	Remarks
IRB 140/IRB 140T	6 kg	810 mm	
IRB 140F/IRB 140TF	6 kg	810 mm	Foundry Plus 2 Protection
IRB 140CR/IRB 140TCR	6 kg	810 mm	Clean Room
IRB 140W/IRB 140TW	6 kg	810 mm	SteamWash Protection
Supplementary load (on upper arm alt. wrist)			
on upper arm		1 kg	
on wrist		0.5 kg	
Number of axes			
Robot manipulator		6	
External devices		6	
Integrated signal supply	12 signals on upper arm		
Integrated air supply	Max. 8 bar on upper arm		
IRC5 Controller variants:	Single cabinet, Dual cabinet, Compact, Panel mounted		

Performance

Position repeatability 0.03 mm (average result from ISO test)

Axis movement	Axis	Working range
	1	360°
	2	200°
	3	280°
	4	Unlimited (400° default)
	5	230°
	6	Unlimited (800° default)
Max. TCP velocity		2.5 m/s
Max. TCP acceleration		20 m/s²
Acceleration time 0-1 m/s		0.15 sec

Velocity *)

Axis no.	IRB 140	IRB 140T
1	200°/s	250°/s
2	200°/s	250°/s
3	260°/s	260°/s
4	360°/s	360°/s
5	360°/s	360°/s
6	450°/s	450°/s

*) Max velocity is reduced at single phase power supply, e.g. Compact controller. Please, see the Product specification for further details.

Cycle time

5 kg Picking side	IRB 140	IRB 140T
cycle 25 x 300 x 25 mm	0.85s	0.77s

Electrical Connections

Supply voltage	200–600 V, 50/60 Hz
Rated power	
Transformer rating	4.5 kVA
Power consumption typicly	0.4 kW

Physical

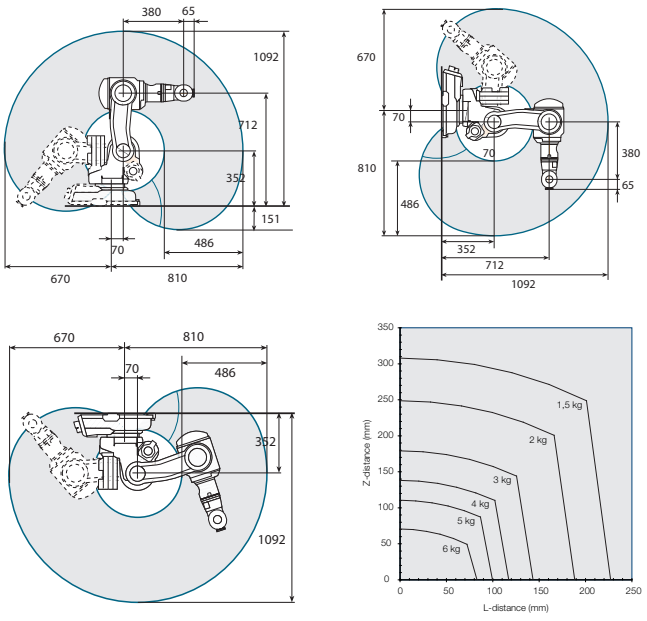
Robot mounting	Any angle
Dimensions	
Robot base	400 x 450 mm
Robot controller H x W x D	950 x 800 x 620 mm
Weight	
Robot manipulator	98 kg

Environment

Ambient temperature for	
Robot manipulator	5 – 45°C
Relative humidity	Max. 95%
Degree of protection,	
Manipulator	IP67
Options	Foundry Plus 2 SteamWash (High pressure steam washable) Clean Room, class 6 (certified by IPA)
Noise level	Max. 70 dB (A)
Safety	Double circuits with supervision, emergency stops and safety functions, 3-position enable device
Emission	EMC/EMI-shielded

Data and dimensions may be changed without notice

Working range



Robotics

IRC5

Industrial Robot Controller

Fifth generation robot controller

Based on more than four decades of robotics experience, the IRC5 sets a benchmark in the robotics industry. In addition to ABB's unique motion control it brings flexibility, safety, modularity, customer adapted user interface, multi robot control and PC tool support.

Safety

Operator safety is a central quality of the IRC5, fulfilling all relevant regulations with good measure, as certified by third-party inspections. Electronic position switches and SafeMove represent a new generation of safety, enabling more flexible cell safety concepts, e.g. involving collaboration between robot and operator.

Motion control

Based on advanced dynamic modelling, the IRC5 optimizes the performance of the robot for the physically shortest possible cycle time (QuickMove) and precise path accuracy (TrueMove). Together with a speed-independent path, predictable and high-performance behavior is delivered automatically, with no tuning required by the programmer. What you program is what you get.

Modularity

The IRC5 comes in different variants in order to provide a cost-effective solution for every need. The ability to stack modules on top of each other, put them side by side or distributed in the cell is a unique feature, leading to optimization of footprint and cell layout.

The compact variant comes with the IRC5 capabilities in a true compact format, able to control the lower end of the IRB range.

The panel-mounted version comes without a cabinet, enabling integration in any encapsulation for exceptional compactness or for special environmental requirements.

FlexPendant

The FlexPendant is characterized by its clean, color touch screen-based design and 3D joystick for intuitive interaction. Powerful customized application support enables loading of tailor-made applications, e.g. operator screens, thus eliminating the need for a separate operator HMI.

RAPID programming language

RAPID programming provides the perfect combination of simplicity, flexibility and power. It is a truly unlimited language with support for structured programs, shop floor language and advanced features. It also incorporates powerful support for many process applications.

Communication

The IRC5 supports the state-of-the-art field busses for I/O and is a well-behaved node in any plant network. Sensor interfaces, remote disk access and socket messaging are examples of the many powerful networking features.

Remote Service enabled

Remote monitoring of the robot is available through GSM or Ethernet. Advanced diagnostics allow fast investigation on failure as well as monitoring of the robot condition throughout the life cycle. Service packages include backup management, reporting and proactive maintenance activities.

RobotStudio

A powerful PC tool for working with IRC5 data on-line as well as off-line. In off-line mode, RobotStudio provides a perfect digital copy of the robot system together with strong programming and simulation features.

MultiMove

Control of up to four robots from one controller, with a compact drive module added for each additional robot. MultiMove opens up previously unthinkable operations, thanks to the perfect coordination of complex motion patterns.

IRC5

Specification

Control hardware:	Multi-processor system
	PCI bus
	Pentium® CPU
	Flash disk for mass memory
	Energy back-up power failure handling
	USB memory interface
Control software:	Object-oriented design
	High-level RAPID programming language
	Portable, open, expandable
	PC-DOS file format
	RobotWare software products
	Preloaded software, also available on DVD

Electrical Connections

Supply voltage:	3 phase 200-600 V, 50-60 Hz
	Integrated transformer or direct mains connection
	1 phase 220/230 V, 50-60 Hz (for Compact Controller only)

Physical	Size H x W x D	Weight
Single cabinet	970 x 725 x 710 mm	150 kg
Dual cabinet	1370 x 725 x 710 mm	180 kg
Control module	720 x 725 x 710 mm	50 kg
Drive module	720 x 725 x 710 mm	130 kg
Empty cabinet for customer equipment	- small 720 x 725 x 710 mm - large 970 x 725 x 710 mm	35 kg 42 kg
Panel Mounted *)		
Control module	375 x 498 x 271 mm	12 kg
Drive module small *1)	375 x 498 x 299 mm	24 kg
Drive module large *2)	658 x 498 x 425 mm	40 kg
Compact controller **)	258 x 450 x 580 mm	27.5 kg

*1) IRB 140, 340, 1600, 260

*2) IRB 2400, 2600, 4400, 4600, 6620, 6640, 6650, 7600, 660, 760

**) IRB 120, 140, 260, 360, 1410, 1600

Environment

Ambient temperature:	0-45°C (32-113°F) option 0-52°C (32-125°F)
Relative humidity:	Max. 95% non condensing
Level of protection:	IP 54 (cooling ducts IP 33)
	Panel Mounted and Compact IP 20
Fulfilment of regulations:	Machine directive 98/37/EC regulations
	Annex II B
	EN 60204-1:2006
	ISO 10218-1:2006
	ANSI/RIA R 15.06 - 1999
	UL 1740-1998

User Interfaces

Control panel:	On cabinet or remote
FlexPendant:	Weight 1 kg
	Graphical color touch screen
	Joystick
	Emergency stop
	Hot plug
	Support for right and left-handed operators
	USB Memory support

User Interfaces continued

Maintenance:	Status LEDs
	Diagnostic software
	Recovery procedures
	Logging with time stamp
	Remote Service enabled

Safety

Basic:	Safety and emergency stops
	2-channel safety circuits with supervision
	3-position enabling device
Electronic Position Switches:	5 safe outputs monitoring axis 1-7
SafeMove:	Supervision of stand-still, speed, position and orientation (robot and additional axes)
	8 safe inputs for function activation,
	8 safe monitoring outputs

Machine Interfaces

Inputs/outputs:	Up to 8192 signals
Digital:	24V DC or relay signals
Analogue:	2 x 0-10V , 3 x ± 10V, 1 x 4-20mA
Serial channel:	1 x RS 232/RS 422 with adapter
Network:	Ethernet(10/100 Mbits per second)
Two channels:	Service and LAN
Fieldbus Master:	DeviceNet™
	PROFINET
	PROFIBUS DP
	Ethernet/IP™
Fieldbus Slave:	DeviceNet™
	PROFINET
	PROFIBUS DP
	Ethernet/IP™
	Allen-Bradley Remote I/O
	CC-link
Conveyor encoder	Up to 6 channels
Integrated PLC	AC500

Sensor Interfaces

	Search stop with automatic program shift
	Seam/contour tracking
	Conveyor tracking
	Machine vision
	Force Control

Data and dimensions may be changed without notice.

Compact controller

Panel mounted controller