Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Desenvolupament d'un sistema de teleoperació pel robot Stäubli TX60 que permeti realitzar simulacions de maniobres mèdiques/quirúrgiques

Document: Memòria

Alumne: Bruno Altadill Gonzalez

Tutor: Dr. Xavier Cufi Sole

Departament: Arquitectura i Tecnologia de Computadors

Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): Setembre / 2018

ÍNDEX

1.- INTRODUCCIÓ

Des dels orígens de l'especie humana, aquesta ha fet us de l'enginy de moltes i diverses maneres. Fruit d'aquest enginy, juntament amb el pas del tems i anys d'esforç, hem aconseguit disposar de tecnologies molt avançades en diverses àrees. En l'actualitat, la tecnologia, està tan arrelada en la nostra societat que no prestem atenció al gran salt que ha fet l'especie humana. Hem passat d'anar a peu amb una llança per caçar i poder menjar, a desplaçar-nos en vehicles molt sofisticats i portar dispositius d'alta tecnologia a la butxaca en un espai de temps molt breu. Alguns exemples de les tecnologies més utilitzades en l'actualitat són:
- **Militar:** punts de mira, sensors, comunicació, armament
- **Exploració:** profunditats dels oceans terrestres, exploració espacial
- **Finances:** monitorització dels mercats de divises, transaccions financeres, moneda virtual
- **Medicina:** aparells de monitorització, anàlisi d'imatges mediques, robots cirurgians
- **Industrial:** cadenes de muntatge, automatitzacions, home automation
- **Energies sostenibles:** solar, hidràulica, eòlica
- **Videojocs:** realitat augmentada, realitat virtual, millores en els sistemes gràfics i de rendiment

El projecte està emmarcat a la Universitat de Girona (UdG), Escola Politècnica Superior (EPS), Grau d'Enginyeria Informàtica (GEINF). Aquest document recull la memòria del que pretén ser, una ínfima aportació a aquesta era d'expansió tecnològica que estem visquen.

1.1.- Descripció

Com esmentàvem en l'apartat anterior, la tecnologia és molt diversa i la podem trobar en un ampli ventall de sectors. En el nostre cas ens centrem en el sector de la medicina ajudant-nos d'eines que trobem en el sector industrial. Actualment l'EPS de la UdG està oferint el master *Medical Imaging and Applications (MAIA)* on es cursa una assignatura anomenada *Introduction to Robotics* i una altre anomenada *Computer Aided Surgery and Medical Robotics.* Cal que els estudiants tinguin la possibilitat de realitzar practiques el més realistes possibles. Per desgracia, els robots cirurgians del calibre del *DaVinci*, pel moment, escapen a l'abast de la nostra escola. Així doncs, hem decidit implementar un sistema que ofereixi l'oportunitat de simular maniobres de cirurgia robòtica amb el robot industrial del qual disposem.

1.2.- Motivacions

Des que tinc memòria, la ciència, la tecnologia, l'enginyeria, la mecànica, són aspectes que sempre m'han fet ser curiós. La robòtica ens permet fer us de tots aquest coneixements per aplicar-los en un únic punt. Arribat el moment de realitzar el treball de fi de grau, vaig decidir parlar amb diversos professors i escoltar les diverses ofertes que tenien en aquests sectors. Finalment, vaig escollir treballar amb en Xevi Cufi i realitzar un treball que podria beneficiar als companys que estudien a l'escola i establir una base per possibles futurs projectes i/o millores del sistema.

1.3.- Propòsit

El propòsit del sistema que hem dissenyat és oferir a l'estudiant del màster *Medical Imaging and Applications (MAIA)* la possibilitat de realitzar practiques que simulin, de la manera més aproximada possible, amb el material que tenim a l'escola, operacions rutinàries mediques i algunes operacions de cirurgia bàsica. També es pretén establir la base d'un sistema que podrà evolucionar per tornar-se sofisticat i poder realitzar maniobres més especialitzades, oferint d'aquesta manera a altres estudiants la viabilitat de crear millores.

1.4.- Objectius del projecte

El principal objectiu del projecte és dissenyar i implementar un sistema de control remot sobre un robot, que permeti realitzar tasques mediques rutinàries i algunes maniobres de cirurgia bàsica. Com esmentàvem, no disposem del robot adequat (robot cirurgia especialitzat), per tant, adaptarem el robot del qual disposem (braç robot industrial de 6 eixos) per realitzar un sistema que ens permeti simular les operacions necessàries.

El sistema a de permetre als estudiants del master *MAIA* realitzar les practiques que necessiten per l'assignatura *Computer Aided Surgery and Medical Robotics.* A continuació exposem amb més detall les fites que ens hem proposat assolir:

1. Maniobrar el braç robòtic industrial *(Stäubli TX60)* amb un joystick en temps real, de manera local o remota
2. Monitoritzar les maniobres amb un sensor de força/moment *(Schunk FTCL 50-80)*, amb la finalitat d'evitar possibles perills pel pacient
3. Instal·lar un sistema de visió estereoscòpica que permeti al cirurgia maniobrar sense tenir línia visual directe amb el pacient i/o el braç robòtic
4. Dissenyar, implementar i testejar les practiques que hauran de realitzar els estudiants

2.- ESTUDI DE VIABILITAT

Previ al començament d'un projecte, que comporti un grau mig/alt d'incertesa, és recomanable dur a terme un estudi de viabilitat. Aquest estudi és necessari quan és difícil fixar l'abast d'un producte/servei, i per tant se'ns fa molt complicat intentar establir un pla de projecte amb garanties de ser complert, en forma, temps i cost.

2.1.- Abast del projecte

Per tal d'evitar desviacions que ens allunyin del resultat esperat, cal definir els límits del sistema. A continuació es presenten les limitacions estudiades:

- Les practiques dels estudiants es duran a terme al laboratori de robòtica, situat a l'Escola Politècnica II de la Universitat de Girona
- El sistema implementat serà especific pel robot del qual disposa la universitat en aquest moment, el Stäubli TX60, el qual treballa amb la unitat de control CS8
- El sistema ha de funcionar amb els ordinadors del laboratori; sistema operatiu: Windows 7 i possibilitat d'usar Linux en maquines virtuals
- La comunicació entre PC i unitat de control es realitza mitjançant una xarxa Ethernet
- El robot presenta unes limitacions físiques industrials (moviments, forces, velocitats, etc), cal observar que no es tindrà la precisió dels robots quirúrgics
- Existeix la possibilitat que la resolució de sensor de força que es vol emprar no sigui prou acurada per fer lectures de valors molt petits per realitzar les maniobres desitjades
- El pressupost per adquirir material està a carreg del departament d'arquitectura i tecnologia de computadors, per tant, l'adquisició de material és limitada

2.2.- Anàlisi de la situació

Com s'ha esmentat amb anterioritat, el projecte està emmarcat dins el departament d'*Arquitectura i Tecnologia de Computadors* de l'*Escola Politècnica Superior* de l'*Universitat de Girona*. Aquest fet ens atorgo algunes avantatges i algunes dificultats en el plantejament del projecte. A continuació s'exposen les fortaleses i debilitats del plantejament inicial:

FORTALESES

- Es disposa de la major part del material necessari per realitzar les practiques, per tant, el cost econòmic del sistema a implementar serà mínim
- Treballarem amb un braç robòtic industrial (Stäubli TX60) amb el qual hem tingut un mínim contacte amb anterioritat
- La definició d'un entorn de treball tan especific (sistema operatiu dels ordinadors del laboratori, marca/model del robot, xarxa local per la comunicació, etc) fa que el sistema que cal implementar també ho sigui, facilitant la tasca

DESAVANTATGES

- El fet de que el sistema sigui tant especific, dificulta l'extrapolació a altres dominis
- Treballar amb la marca Stäubli dificulta la tasca de desenvolupament, ja que tota la informació està fortament privatitzada i cal realitzar sol·licituds de documentació. Com a conseqüència d'aquesta privatització es disposa de molt poca informació Online
- El fet de treballar amb Stäubli implica que tot el codi font desenvolupat per interactuar amb el robot ha de funcionar amb el llenguatge especific d'Stäubli anomenat VAL3
- Es improbable que s'aconsegueixi la mateixa resolució en les maniobres, que la que es podria aconseguir amb un robot destinat al propòsit medic
- L'entorn de treball és un laboratori de robòtica industrial, és probable que ens manqui o sigui de difícil accés alguns instruments mèdics i/o quirúrgics

2.3.- Avaluació de la viabilitat del projecte

Abans de prendre una decisió, cal respondre a algunes preguntes amb la finalitat de determinar si el projecte és realment viable o no.

- *Disposem de tots els elements hardware que ens calen pel desenvolupament?*

  No, ens manquen alguns elements, però després de realitzar l'estudi de viabilitat considerem que són de fàcil accés

- *El codi font d'alt/baix nivell que cal implementares pot ser desenvolupat amb els nostres coneixements?*

  Si, ja que al llarg del grau hem adquirit el saber necessari per poder dur a terme el desenvolupament del sistema

- *En cas de dificultat, disposem d'alguna font de suport?*

  Si, sempre que els dubtes siguin senzills els mestres corresponents ens podran ajudar. L'Stäubli TX60 ens permet maniobrar de moltes i diverses maneres, tot i que de moment, les practiques que s'han dut a terme en altres assignatures, són molt elementals.

- *Disposem d'alguna entitat que s'encarregui de l'adquisició del material necessari per desenvolupar el sistema?*

  Si, el departament d'Arquitectura i Tecnologia de Computadors, més concretament en Xevi Cufi, s'encarrega de l'adquisició del material requerit

- *Les resolucions de maniobra que te el robot industrial, són adaptables al sistema que es pretén desenvolupar?*

  No, s'ha deixat clar que no podrem obtenir una resolució de maniobra tan precisa com un robot especialitzat en el sector medic. No obstant, podem implementar una bona aproximació amb les eines que disposem

Finalment, després d'estudiar detingudament si el projecte és o no viable, em arribat a la conclusió que el disseny, implementació i avaluació del sistema que volem implementar **ÉS VIABLE**.

3.- METODOLOGIA

Metodologia és un mot generat a partir de tres paraules gregues: *metà* ("**més enllà**"), *odòs* ("**camí**") i *logos* ("**estudi**"). El concepte fa referencia al pla d'investigació que permet complir certs objectius en el marc d'una ciència. Per tant, es pot entendre la metodologia com el conjunt de procediments que determina una investigació de tipus tecnològic/científic.

Els següents apartats exposen breument la metodologia que s'utilitzara al llarg del disseny, desenvolupament i avaluació del sistema que cal implementar.

3.1.- Definició de la feina

Abans de començar a treballar, cal tenir molt ben definit què i com cal fer la feina. Aquest pas és molt important, ja que el temps del que disposem és limitat i no tenim marge d'error. Definir les etapes i per cada etapa, tasques concretes i breus, és millor que definir poques etapes i tasques de gran abast.

3.2.- Cronograma de treball

El fet de definir i/o limitar el temps que podem emprar per cada tasca, fa que seguim un pla de desenvolupament molt acurat. El cronograma de treball ajuda a organitzar la feina de manera visual i ens permet conèixer en quina fase del projecte estem o quina serà la següent etapa. Facilita la consecució progressiva dels objectius, ja que l'elaboració del cronograma exigeix descompassar el procés en diverses tasques que s'estructuren en petites unitats fàcilment assolibles.

3.3.- Treball per etapes

Un cop tenim definides les tasques en petits processos i hem definit un cronograma de treball, només cal començar a realitzar les tasques definides en aquest període de temps. El fet de treballar per etapes, garanteix uns mínims, ja que fins que no assolim una tasca no avancem a la següent. Per aquest motiu, és tant important definir tasques petites en uns temps d'execució realistes.

3.4.- Comunicació amb el "client"

La comunicació amb el client hauria de ser una obligatòria al llarg de qualsevol projecte. Encara que les dates d'entrega siguin clares, les tasques a desenvolupar estiguin definides i les etapes per entregar els resultats estiguin fixades, sempre hem de seguir comunicant-nos amb el client i mantinguen-lo al dia del avenços, encara que la feina ja estigui feta. Aquest fet genera seguretat i confiança en el client, i a més és probable que siguin més comprensius si sorgeix alguna complicació en el transcurs del projecte.

En el nostre cas, el "client", podríem dir que ha estat en Xevi Cufi, amb el qual hem mantingut una comunicació eficient i fluida des de l'inici del projecte.

4.- PLANIFICACIÓ

La planificació del projecte be forçament definida per la metodologia que exposàvem amb anterioritat. En aquest apartat definirem amb més detall quines són les etapes i les tasques que cal assolir, dissenyarem un cronograma de treball i finalment explicarem quin són els resultats que esperem obtenir.

4.1.- Definició de les etapes i les tasques

Hem definit el projecte en etapes i cada etapa en diverses tasques. A continuació és presenta el model definit:

- ESTUDI DELS REQUERIMENTS
  - Estudi dels requeriments funcionals
  - Estudi dels requeriments no funcionals

- ANÀLISI I FORMACIÓ
  - Analitzar els elements amb els que treballarem, entendre com funcionen i quina és la seva finalitat
  - Realitzar una mínima formació amb els elements de treball per evitar contratemps en el moment de la implementació

- DISSENY DEL SISTEMA
  - Disseny estructural del sistema
  - Disseny dels scripts/programes
  - Disseny dels protocols de comunicació
  - Disseny del sistema de visió

- IMPLEMENTACIÓ DEL SISTEMA
  - Implementació del sistema de maniobra
  - Implementació del sistema de control per forces/moments
  - Implementació del sistema de visió

- AVALUACIÓ DEL SISTEMA
  - Avaluació del sistema de maniobra
  - Avaluació del sistema de control per forces
  - Avaluació del sistema de visió

- AVALUACIÓ DE POSSIBLES MILLORES
  - Contemplar i enumerar les possibles millores que podrien dur-se a terme

4.2.- Cronograma de treball

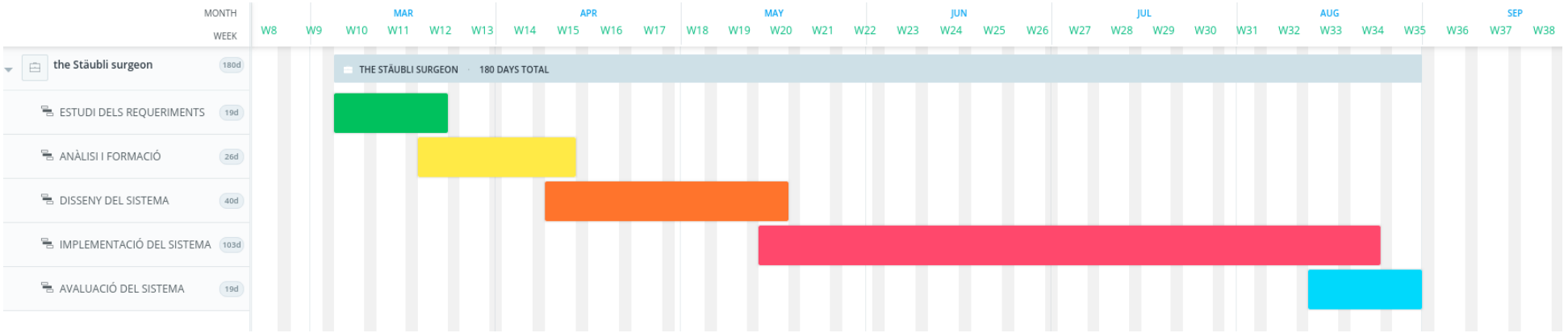


Figura 1: Cronograma temporal

- ESTUDI DELS REQUERIMENTS (19 dies)
  - 2018-03-05 a 2018-03-23
- ANÀLISI I FORMACIÓ (26 dies)
  - 2018-03-19 a 2018-04-13
- DISSENY DEL SISTEMA (40 dies)
  - 2018-04-09 a 2018-05-18
- IMPLEMENTACIÓ DEL SISTEMA (103 dies)
  - 2018-05-14 a 2018-08-24
- AVALUACIÓ DEL SISTEMA (19 dies)
  - 2018-08-13 a 2018-08-31
- AVALUACIÓ DE POSSIBLES MILLORES (19 dies)
  - 2018-08-13 a 2018-08-31

4.3.- Resultats esperats de cada etapa

- ESTUDI DELS REQUERIMENTS

  En aquesta etapa inicial esperem poder realitzar un estudi dels requeriments necessaris per desenvolupar el projecte. Podem dividir el resultats en 2 grups: els elements físics que ens calen i el programari. Per tant, obtindrem un llistat de tot el hardware que ens cal i una primera iteració del programari que caldrà implementar.

- ANÀLISI I FORMACIÓ

  Un cop tenim clar quins són els elements que intervindran en el sistema cal analitzar-los, veure quines són les seves finalitats i les limitacions que presenten. Per aconseguir-ho, en aquest etapa, llegirem manuals i farem petits tests per avaluar les capacitats dels components.

- DISSENY DEL SISTEMA

  Sempre és una bona idea tenir una idea clara del que es pretén desenvolupar abans de començar l'etapa d'implementació. Coneixent els elements amb els que treballarem, els objectius que volem assolir i el que hem aprés al llarg del grau, podem dissenyar en profunditat el sistema que és vol implementar.

- IMPLEMENTACIÓ DEL SISTEMA

  Amb un disseny inicial ja es pot començar l'etapa d'implementació. És probable que al llarg del procés d'implementació sorgeixin petits entrebancs, ja sigui perquè no s'han contemplat en el disseny inicial o per factors externs que alteren l'entorn de treball. Si ens trobem aquest entrebancs, el més recomanable és invertir una mica de temps en pensar com resoldre'l, no ho hem de veure com quelcom negatiu. És més recomanable invertir temps en pensar diverses solucions i escollir la més adient, enlloc d'implementar la primera que se'ns acudeixi i donar lloc a futurs problemes.

- AVALUACIÓ DEL SISTEMA

  Al llarg de la implementació, es van fent petites proves per verificar que cadascun dels mòduls que s'implementen funcionen adientment. Un cop tenim el sistema implementat i funcional en la seva totalitat, cal fer una avaluació complerta. En aquesta avaluació es revisa el funcionament, els temps de resposta i l'eficiència del sistema.

- AVALUACIÓ DE POSSIBLES MILLORES

  En molts casos, els projectes és veuen limitats per manca de recursos, però aquest fet no implica que no es puguin estudiar possibles millores per dur a terme en el futur.

5.- MARC DE TREBALL I CONCEPTES PREVIS

5.1.- Laboratori de robòtica de l'Escola Politècnica Superior

Com ja s'ha introduït amb anterioritat, el treball es troba emmarcat al laboratori de robòtica de l'Escola Politècnica Superior de la Universitat de Girona. A continuació presentem els components dels que disposem al laboratori:

- 9 PC pels estudiants
- 1 PC pel mestre
- 1 robot de la marca Stäubli model TX60 (braç articular de 6 eixos)
- 1 robot de la marca Stäubli model TS40 (robot scara)
- 1 fotocèl·lula de seguretat pel robot Stäubli TX60
- 4 turtle-bots per altres assignatures
- 2 bancs de treball, un per cada robot Stäubli
- Una àrea (laberint) de proves pels turtle-bot
- Joystick Logitech Extreme 3D Pro

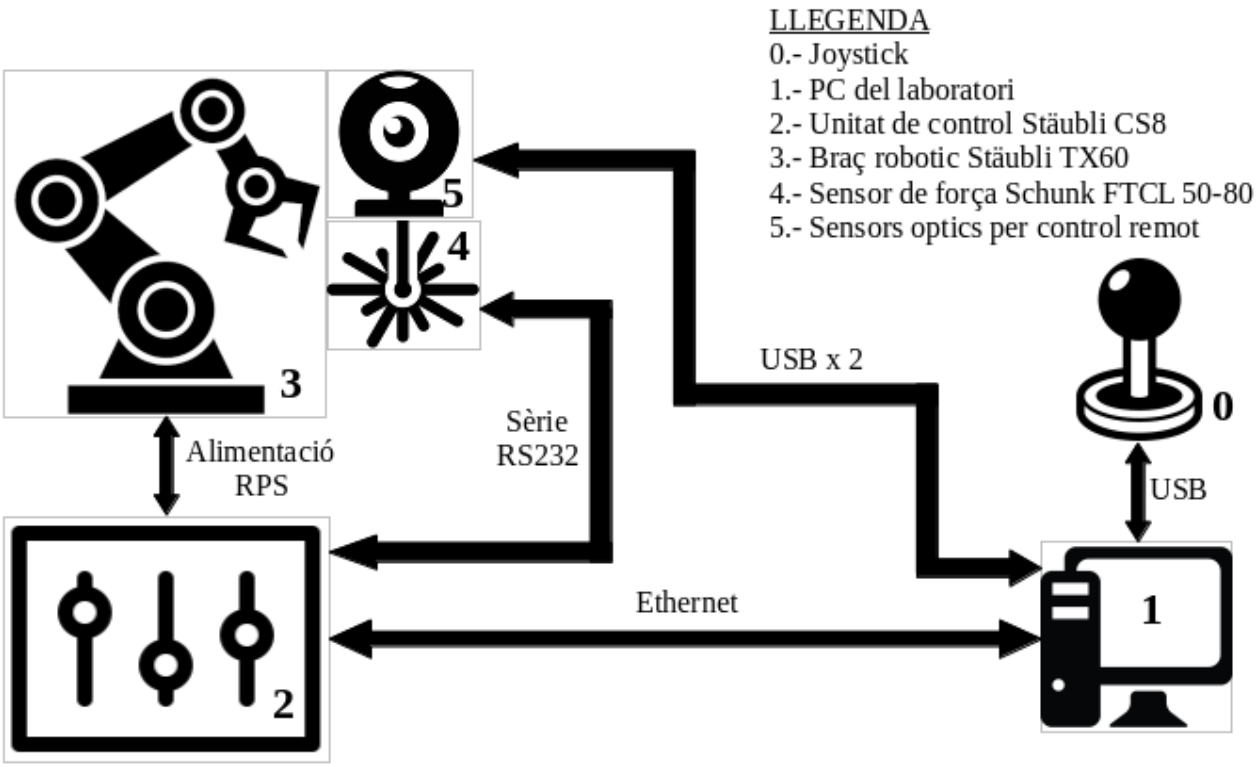


Figura 2: Esquema del sistema

La figura 2 representa el sistema que hem muntat al laboratori. El PC actua com a unitat de control superior, rebent informació del joystick per realitzar les maniobres de moviment, mostrant la informació que rebem dels sensors òptics per maniobrar sense línia directa visual, i comunicant-se amb la unitat de control CS8C. La unitat de control CS8C actua com mediador entre el PC i el braç robòtic amb el sensor de força acoblat, d'aquesta manera, el PC és qui te el control real, però delega les tasques a la unitat CS8C que és la que executa el codi necessari per realitzar les maniobres o les operacions de lectura, bloqueig, parada, etc. El fet de que el sistema de control de forces (sensor) estigui comunicat directament amb la unitat de control CS8C agilitza les maniobres de moviment, ja que no cal notificar al PC les dades llegides perquè aquest prengui una decisió.

5.2.- Fonaments teòrics

Per poder treballar amb un robot, prèviament cal entendre com funciona. En aquest apartat exposarem de manera teòrica com funciona un braç robot industrial.

La cinemàtica estudia el moviment d'un robot respecte un sistema de coordenades de referencia. Existien dos problemes principals a resoldre a la cinemàtica dels robots:

CINEMÀTICA DIRECTA

El primer, conegut com problema de la cinemàtica directa, consisteix en determinar quina és la posició {x, y, z} i orientació {roll, pitch, yaw} de l'element terminal del robot respecte un sistema de coordenades que s'agafa com a referencia, coneixent els valors articulars (q1, q2, ..., qn) i paràmetres geomètrics dels elements del robot.

CINEMÀTICA INVERSA

El segon, conegut com problema de la cinemàtica inversa, consisteix en trobar els valors articulars (q1, q2, ..., qn) que ha d'adoptar un robot per assolir una posició {x, y, z} i orientació {roll, pitch, yaw} de l'element terminal.

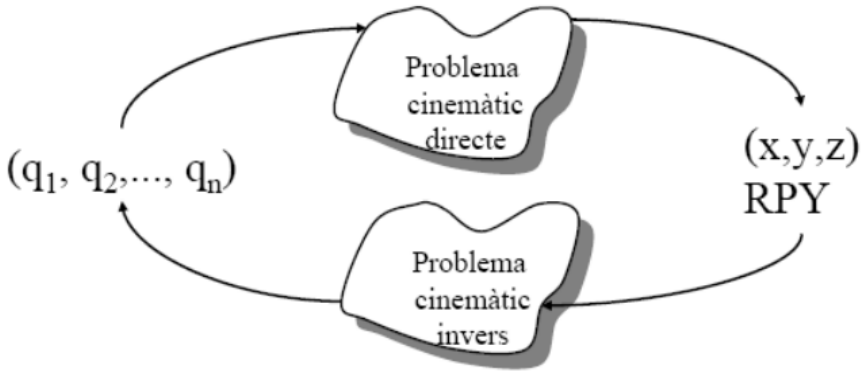


Figura 3: Esquema de la cinemàtica d'un robot

Els mètodes usats per la resolució d'aquest problema requereixen uns coneixements bàsics d'àlgebra vectorial i matricial per la localització espacial de sòlids.

5.2.1.- Cinemàtica directa

La cinemàtica directa troba la posició {x, y, z} i orientació {roll, pitch, yaw} de l'element terminal del robot a partir de les variables d'unió (q1, q2, ..., qn).

Figura 4: Esquema de la cinemàtica directa

Per saber els valors de les variables d'unió del robot s'ha de conèixer el numero d'articulacions o graus de llibertat del robot, el tipus de cadascuna de les articulacions i la mida de cadascun dels segments que formen les articulacions.

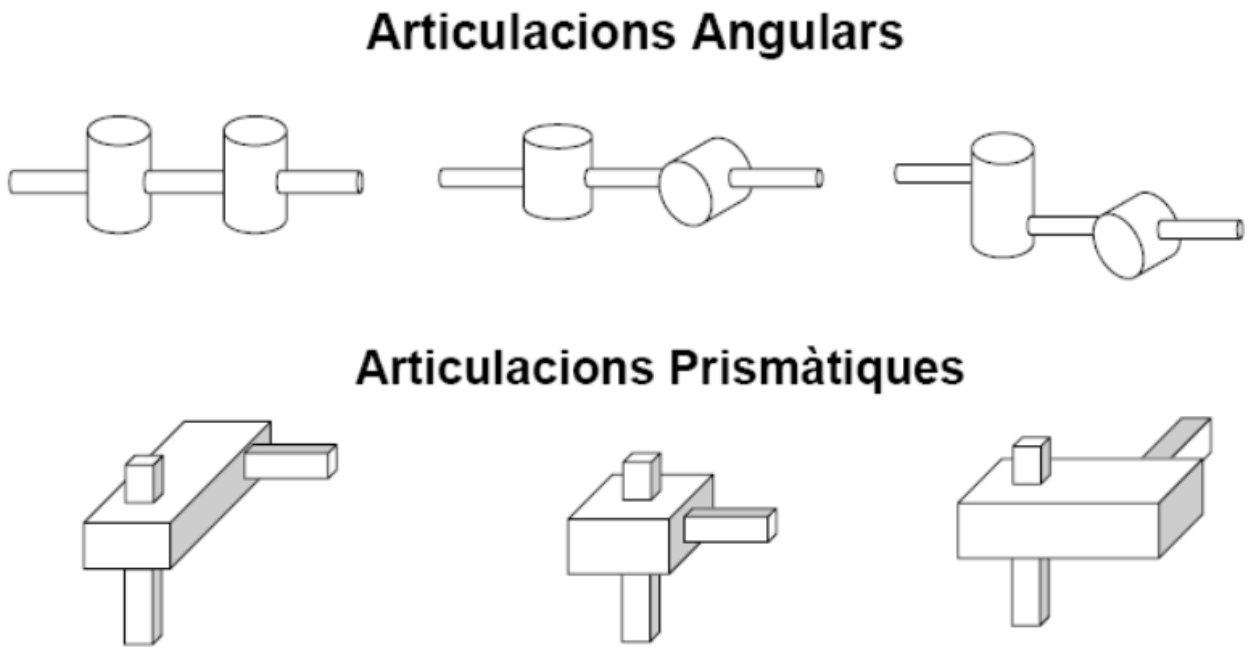


Figura 5: Tipus d'articulacions

En el nostre cas, el robot Stäubli TX60 té 6 graus de llibertat, és a dir, sis variables d'unió i aquestes sis unions són de tipus angular. Pel que fa a la mida de cadascun dels segments que formen les articulacions,. Aquestes estan definides a la figura 11 (dimensions del braç robòtic)

L'idea que descriu la cinemàtica directe és que el robot es pot veure com una cadena cinemàtica formada per sòlids rígids, l'origen dels quals es troba en cadascun dels graus de llibertat o articulacions del robot. Així si es coneix la posició i orientació de cadascun dels sòlids rígids respecte un sistema de coordenades fix, el de la base del robot, és pot trobar la posició i orientació del l'element terminal del robot.

La posició i orientació es representen mitjançant una matriu homogènia ${}^{R}T_{H}$.

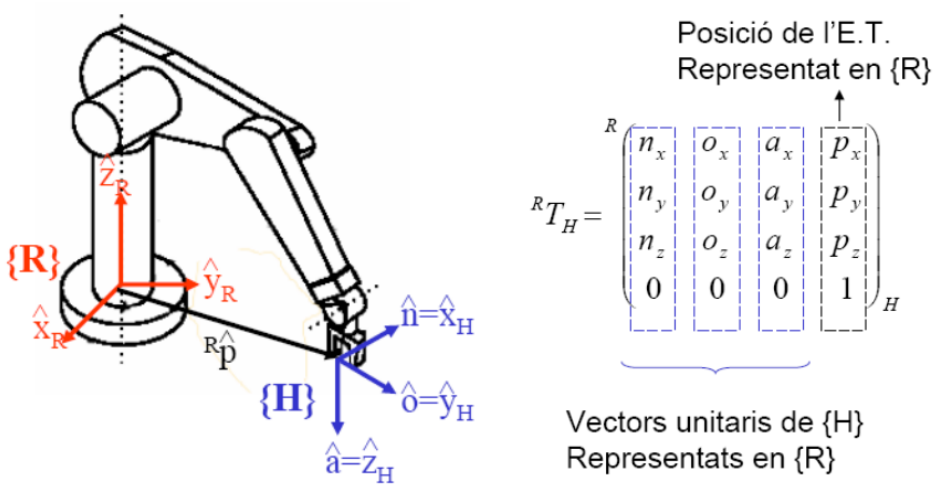


Figura 6: Matriu homogènia

La matriu homogènia de l'element terminal del robot s'obtindrà a partir de les matrius homogènies de les articulacions del robot. És a dir, en general una matriu homogènia descriu la posició i orientació de dos sòlids units a una articulació. Per tal de trobar aquesta posició es realitzen les transformacions homogènies necessàries (translacions i rotacions) als sistemes de coordenades de cada solid que forma l'articulació. Així la matriu homogènia ${}^{R}T_{H}$ serà producte de les següents transformacions homogènies:

$$T = {}^{0}ATCP = {}^{0}A_1\ {}^{1}A_2\ {}^{2}A_3\ {}^{3}A_4\ {}^{4}A_5\ {}^{5}A_6\ {}^{6}A_{TCP}$$

Per exemple, la matriu homogènia ${}^{0}A_1$ descriu la posició i orientació del sistema de referencia del primer solid o articulació respecte al sistema de referencia a la base del robot. En robòtica és usual denotar per TCP (Tool Center Point) el sistema de referencia solidari a l'extrem (element terminal) del braç.

Per resoldre el problema de la cinemàtica directa existeixen molts estudis i mètodes. En el nostre cas, em aplicat el mètode de Denavit-Hartenberg (DH), el qual és un mètode sistemàtic que serveix per descriure la geometria del robot respecte un sistema de coordenades de referencia fix.

Algorisme de Denavit-Hartenberg (DH)

0. Numereu les **unions** des de 1 fins a *n* començant per la base i acabant pel yaw, pitch i roll de l'element final (en aquest ordre).

1. Assigneu el sistema de coordenades L_0 a la base, assegurant-vos que Z_0 coincideix amb l'eix de la unió 1, i inicialitzeu k=1.

2. Feu coincidir Z_k amb l'eix de la unió k+1.

3. Poseu l'origen de L_k a la interseció dels eixos Z_k i Z_{k-1}. Si Z_k i Z_{k-1} no intersecten, utilitzeu la intersecció de Z_k amb una normal comuna a Z_k i Z_{k-1}.

4. Poseu X_k de manera que sigui ortogonal a Z_k i Z_{k-1}. Si Z_k i Z_{k-1} són paral·lels, feu X_k perpendicular a Z_{k-1}, situeu-lo al llar del link i apuntant cap a fora.

5. Seleccioneu Y_k per acabar de completar el sistema de coordenades L_k.

6. Feu k=k+1. Si k<n, torneu al pas 2.

7. Poseu l'origen de L_n a la punta de l'element final. Feu coincidir Z_n amb el vector **a** (approach), Y_n amb el vector **o** (orientation) i X_n amb el vector normal de l'element final. Feu k=1.

8. Poseu el punt b_k a la intersecció dels eixos X_k i Z_{k-1}. Si no intersecten, poseu-lo a la intersecció d'X_k amb una normal comuna a X_k i Z_k.

9. θ_k és l'angle de rotació des de X_{k-1} a X_k mesurat sobre Z_{k-1}.

10. $\mathbf{d_k}$ és la distància des de l'origen del sistema de coordenades L_{k-1} al punt b_k mesurat al llarg de l'eix Z_{k-1}.

11. $\mathbf{a_k}$ és la distància des del punt b_k a l'origen del sistema de coordenades L_k mesurat al llarg de X_k.

12. α_k és l'angle de rotació des de Z_{k-1} a Z_k mesurat sobre X_k.

13. Feu k=k+1. Si k≤n aneu al pas 8.

El resultat d'aplicar el mètode Denavit-Hartenberg per resoldre la cinemàtica directa del robot Stäubli TX60 és el següent:

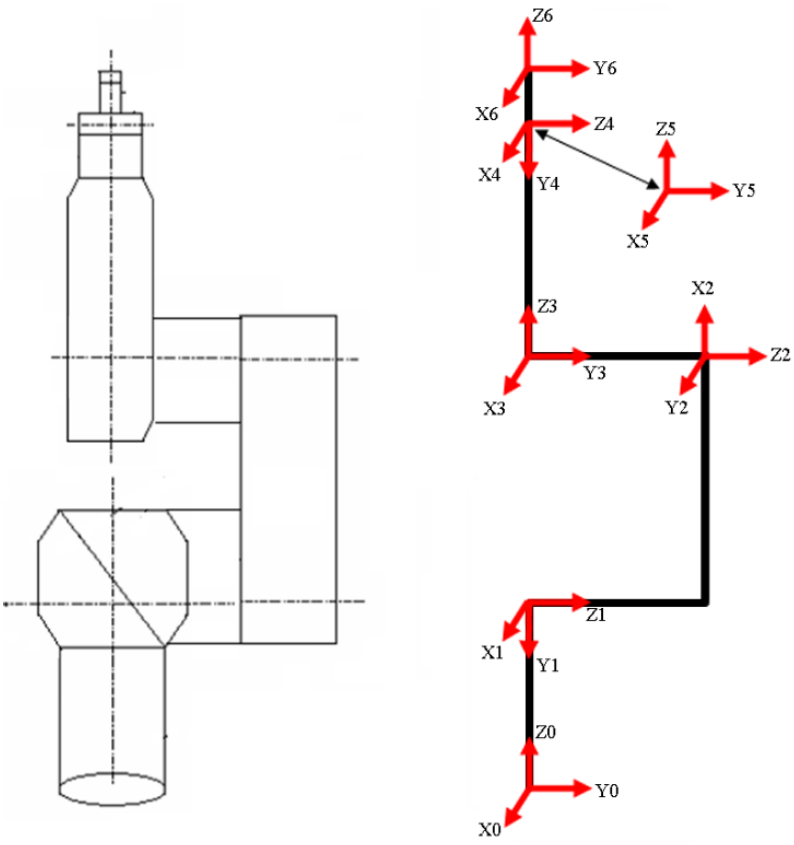


Figura 7: Cinemàtica dorecta Stäubli TX60

S'identifiquen els diferents sistemes de coordenades que formen el robot i els seus sentits de gir mitjançant el mètode (DH).

GDLL	θ_i (°)	d_i (mm)	a_i (mm)	α_i (°)	Home (°)
1	q1	375	0	-90°	0°
2	q2	142	290	0°	-90°
3	q3	-142	0	90°	90°
4	q4	310	0	-90°	0°
5	q5	0	0	90°	0°
6	q6	70	0	0°	0°

5.2.2.- Cinemàtica inversa

La cinemàtica inversa obté els valors de les variables d'unió (q1, q2, ..., qn) a partir d'una determinada posició i orientació {x, y, z, roll, pitch, yaw}.

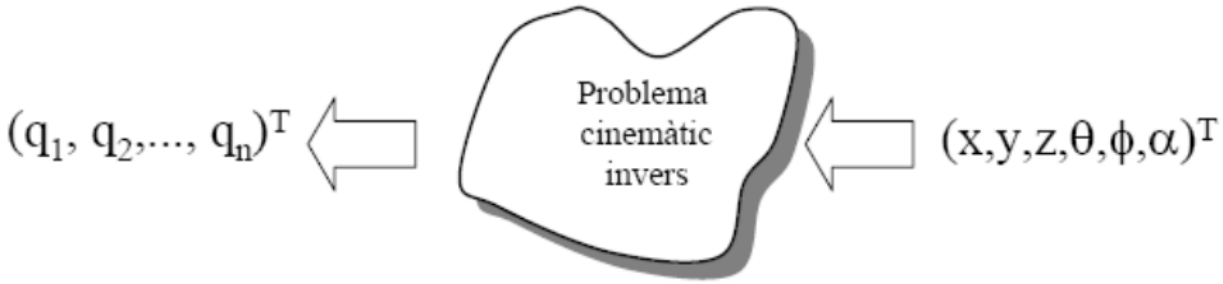


Figura 8: Esquema de la cinemàtica inversa

La posició i l'orientació han de poder ser assolides pel robot, és a dir, han d'estar dins del rang de treball i dins les restriccions electromecàniques del robot.

La cinemàtica inversa és un problema més complex que el problema de la cinemàtica directa. No existeix una metodologia tan clara, senzilla i pautada com en el problema de la cinemàtica directa, sinó que a vegades existeixen diferents solucions i algunes vegades no es pot trobar la solució. La metodologia que s'usa és igualar la matriu simbòlica a la matriu numèrica del robot.

$$ {}^{R}T_{H} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{11} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}_{H}^{R} = \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}_{H}^{R} $$

Figura 9: Matrius simbòlica i numèrica

Això ens proporciona un sistema d'equacions amb 12 equacions i n incògnites, tantes incògnites com graus de lliberta té el robot. Però aquestes 12 equacions no són totes linealment independents així que simplificant obtenim un sistema de 6 equacions i n incògnites (n = graus de llibertat del robot).

$$ {}^{0}A_{1}\ {}^{-1}T = U_2 $$
$$ ({}^{1}A_{2})^{-1}\ ({}^{0}A_{1})^{-1}\ T = U_3 $$
$$ ({}^{2}A_{3})^{-1}\ ({}^{1}A_{2})^{-1}\ ({}^{0}A_{1})^{-1}\ T = U_4 $$
$$ ({}^{3}A_{4})^{-1}\ ({}^{2}A_{3})^{-1}\ ({}^{1}A_{2})^{-1}\ ({}^{0}A_{1})^{-1}\ T = U_5 $$
$$ ({}^{4}A_{5})^{-1}\ ({}^{3}A_{4})^{-1}\ ({}^{2}A_{3})^{-1}\ ({}^{1}A_{2})^{-1}\ ({}^{0}A_{1})^{-1}\ T = U_6 $$

Quan la solució existeix no sempre és única, sinó que pot haver-hi solucions redundants. Existeixen diferents mètodes de resolució de la cinemàtica inversa, son els següents:

- Solució geomètrica
  Només aplicable a robots de pocs graus de llibertat (3 o menys). Aplicable als 3 primers graus de llibertat d'alguns robots complexos. La solució és aplicable en temps real, no és genèrica i depèn del manipulador.

- Solució simbòlica
  No sempre es pot trobar la solució. La solució és aplicable en temps real, no es genèrica i depèn del manipulador.

- Solució per desacoblament cinemàtic
  Mètode hibrid entre els dos anteriors.

- Solució numèrica iterativa
  És una solució genèrica, depèn del paràmetres del Denavit-Hartenberg, és molt lenta i no és aplicable en temps real.

La solució obtinguda pel problema cinemàtica invers del robot Stäubli TX60 és la següent:

1. Si notem com $S_i = \sin\theta_i$, $C_i = \cos\theta_i$, $S_{ij} = \sin(\theta_i + \theta_j)_i$ $C_{ij} = \cos(\theta_i + \theta_j)$, del sistema d'equacions U_2 els termes (3, 4) obtenim:

$$-S_1 P_x + C_1 p_y = -d_3$$

2. Si realitzem els canvis de variables a l'equació anterior i simplifiquem, obtenim en valor de la variable θ_1:

$$p_x = r \cdot \cos\phi$$

$$p_y = r \cdot \sin\phi$$

$$r = +\sqrt{p_x^2 + p_y^2}$$

$$\phi = \arctan\left(\frac{p_y}{p_x}\right)$$

$$\theta_1 = \arctan\left(\frac{p_y}{p_x}\right) + \arctan\left(\frac{d_3}{\pm\sqrt{(r^2 + d_3^2)}}\right)$$

3. Del mateix sistema d'equacions U_2 igualem els termes (1, 4) i (2, 4) i obtenim dues equacions que només depenen de θ_2 i θ_3:

$$C_1 p_x + S_1 p_y = d_4 S_{23} + a_3 C_{23} + a_2 C_2$$

$$-p_x = -d_4 S_{23} + a_3 S_{23} + a_2 S_2$$

4. Si resolem el sistema d'equacions i fem diverses simplificacions obtenim de forma explicita θ_3, que només depèn de θ_1 determinat anteriorment:

$$\theta_3 = \arctan\left(\frac{a_3}{-d_4}\right) + \arctan\left(\frac{d}{\pm\sqrt{(e - d)}}\right)$$

$$d = (C_1 p_x + S_1 p_y)^2 + p_z^2 - d_4^2 - a_3^2 - a_2^2$$

$$e = 4a_2^2 a_3^2 + 4a_2^2 d_4^2 (constant)$$

5. Del sistema d'equacions U_4 igualem els termes (1, 4) i (3, 4) obtenint dues equacions on les úniques variables son C_{23} i S_{23}, resolent el sistema i fent diverses simplificacions obtenim:

$$\theta_{23} = \arctan\left(\frac{\upsilon_1(C_1 p_x + S_1 p_y) - \upsilon_2 p_z}{\upsilon_1(C_1 p_x + S_1 p_y) + \upsilon_2 p_z}\right)$$

$$\upsilon_1 = a_2 C_3 + a_3$$

$$\upsilon_2 = d_4 + a_2 S_3$$

6. Ara ja podem trobar θ_2:

$$\theta_2 = \theta_{23} - \theta_3$$

7. Notem que θ_1, θ_2, θ_3 tenen totes dues solucions $\pm\theta_i$. Ara, del mateix sistema d'equacions U_4 els termes (1, 2) i (2, 3) donen les següents equacions on només apareixen θ_4 i θ_5:

$$C_4 S_5 = C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z$$

$$S_4 S_5 = -S_1 a_x + C_1 a_y$$

8. Si $\sin\theta_5 \neq 0$, podem dividir les equacions anteriors i apliquem l'arctangent:

$$\theta_4 = \arctan\left(\frac{-S_1 a_x + C_1 a_y}{C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z}\right)$$

9. Si $\sin\theta_5 = 0$, les dues equacions anteriors es compleixen per a qualsevol θ_4, diem que ens trobem en un cas degenerat, i això correspon a quan l'eix 4 i 6 es troben alineats. En aquesta situació també podem acceptar com a valor de θ_4 l'obtingut en l'equació anterior.

10. Del sistema U_5 igualem els elements (1, 2) i (2, 2) i obtenim:

$$S_6 = -C_5\{C_4\lfloor C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z\rfloor + S_4\lfloor -S_1 a_x + C_1 a_y\rfloor\} + S_5\{S_{23}(C_1 a_x + S_1 a_y) + C_{23} a_z\}$$

$$C_6 = -S_4\left\lfloor C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z\right\rfloor + C_4\left\lfloor -S_1 a_x + C_1 a_y\right\rfloor$$

11. Si dividim les equacions i aïllem θ_4 obtenim θ_4 on S_6 i C_6 es corresponen a les dues expressions anteriors respectivament:

$$\theta_6 = \arctan\left(\frac{S_6}{C_6}\right)$$

12. Si observem el conjunt de solucions angulars notem que tenim 8 solucions possibles per a una mateixa posició i orientació de l'extrem del braç, ja que fixades θ_1, θ_2, θ_3 la resta de solucions són úniques.

5.3.- Robot industrial manipulador Stäubli TX60

Segons la Associació d'Indústries Robòtiques (RIA):
"Un robot industrial és un manipulador multifuncional reprogramable, capaç de moure matèries, peces, eines o dispositius especials, segons trajectòries variables, programades per realitzar tasques diverses"

Stäubli és una de les firmes més importants en robots manipuladors, tenen distribuïdors per tot el mon i els seus robots son aptes per múltiples tasques: manipulació de carregues, tall laser, soldadura, teleoperació, alimentació, aplicacions en sala blanca, medicina, etc.

Segons la tasca o finalitat per la que s'utilitzi el robot, Stäubli proporciona el robot amb les característiques i elements terminals (o eina) pertinents. L'element terminal d'un robot és l'instrument situat a l'ultima articulació del robot, a la part anàloga a la nostra ma, i és la part que incorpora l'eina més apropiada segons la tasca a realitzar.

El robot utilitzat en el nostre projecte és el braç de geometria estàndard Stäubli TX60.

TX	6	0
(1)	(2)	(3)

(1) Braç de la família TX
(2) Radi màxim de treball entre l'eix 1 i l'eix 5, expressat en decímetres
(3) Numero d'eixos actius: 0 = 6 eixos actius

5.3.1.- Arquitectura del robot

El braç manipulador Stäubli TX60 que es pot veure a la figura 3.2 consta d'una cadena cinemàtica de sis sòlids rígids o elements units entre sí mitjançant sis articulacions de revolució (q1, q2, q3, q4, q5 i q6) que ens aporten sis graus de llibertat.

Els moviments de les articulacions del braç són generats per servomotors acoblats a sensors de posició. Cada un d'aquests motors està equipat amb un fre de servei. Aquest conjunt permet conèixer permanentment la posició absoluta del robot.

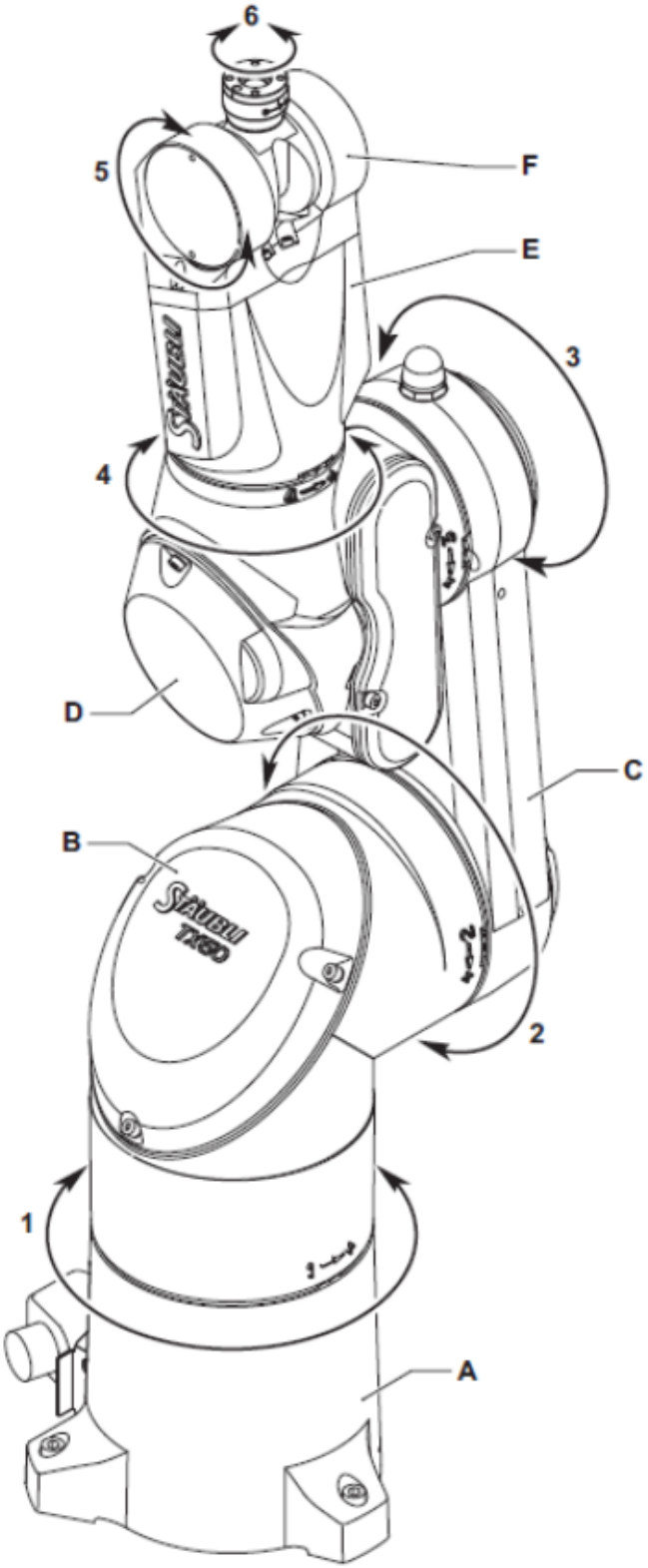


Figura 10: Braç Stäubli TX60

El braç està compost dels següents elements: la base (A), l'espatlla (B), el braç (C), el colze (D), l'avantbraç (E) i el puny (F).

5.3.2.- Dimensions

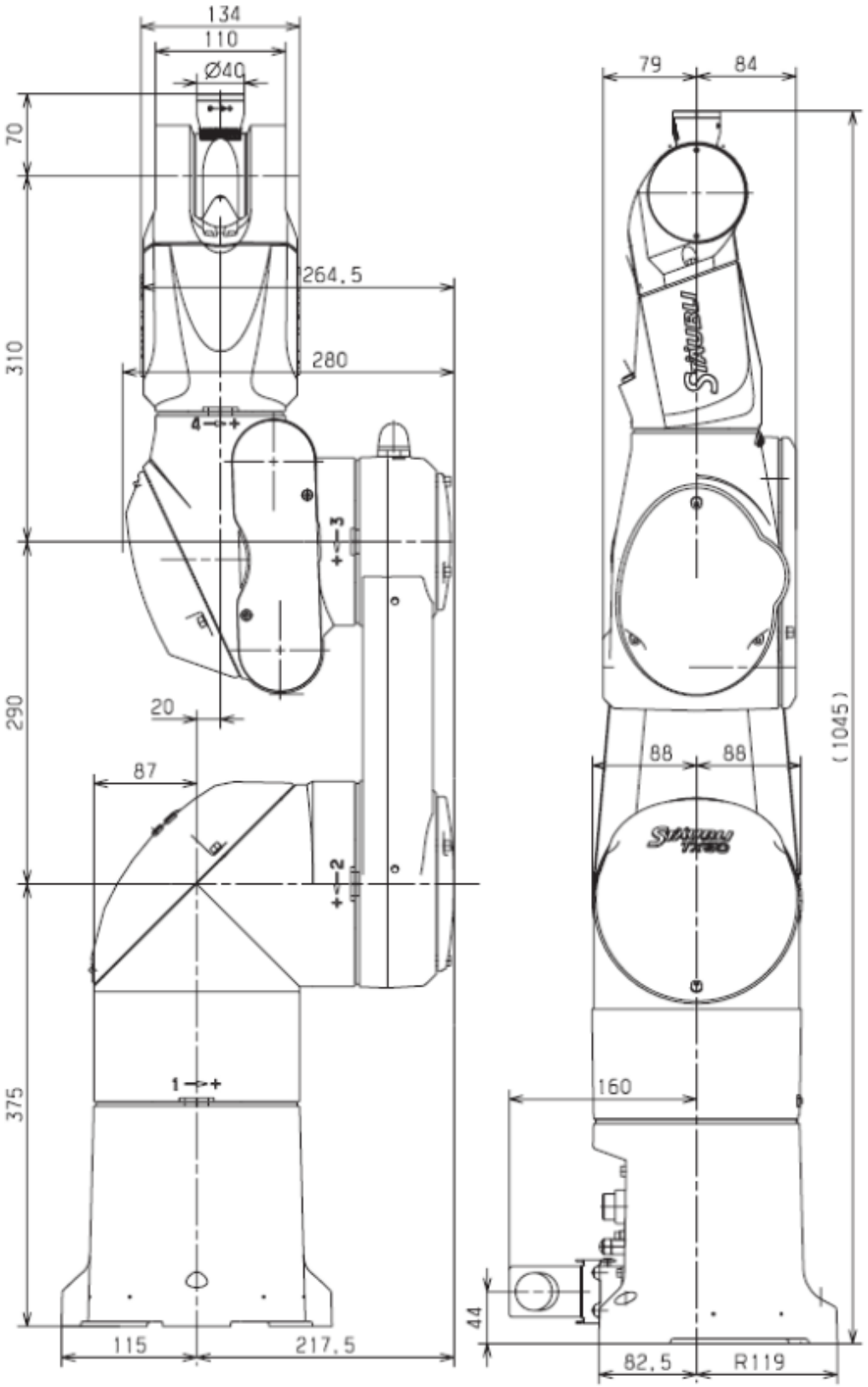


Figura 11: Dimensions del braç

5.3.3.- Pes

El pes total del robot és de 51,4 Kg.

5.3.4.- Ambient de treball

L'ambient de treball del robot ha de complir una serie de condicions pel seu correcte funcionament:

- Temperatura de funcionament de +5 ºC a +40 ºC segons la normativa directiva NF EN 60 204-1
- Humitat des de 30% a 95% (màxim sense condensació) segons la normativa directiva NF EN 60 204-1
- Altitud màxima de 2000 metres

5.3.5.- Espai de treball

L'espai o rang de treball d'un robot és el conjunt de punts accessibles per l'element terminal del robot. Aquest volum queda determinat per la grandària, forma i tipus dels elements que l'integren, juntament amb les limitacions imposades pel sistema de control.

El fet que un punt de l'espai sigui accessible pel robot no implica que ho pugui fer en qualsevol orientació. Existeix un conjunt de punts que solament podem accedir en una determinada orientació, mentre que altres punts admetran qualsevol orientació.

En el braç Stäubli TX60 l'espai o rang de treball (figura 12) queda definit amb els següents paràmetres:

- R.M radi de treball màxim entre eixos 1 i 5 — 600 mm
- R.M radi de treball màxim entre eixos 2 i 5 — 600 mm.
- R.m1 radi de treball mínim entre eixos 1 i 5 — 190 mm.
- R.m2 radi de treball mínim entre eixos 2 i 5 — 189 mm.
- R.b radi de treball entre eixos 3 i 5 — 310 mm.

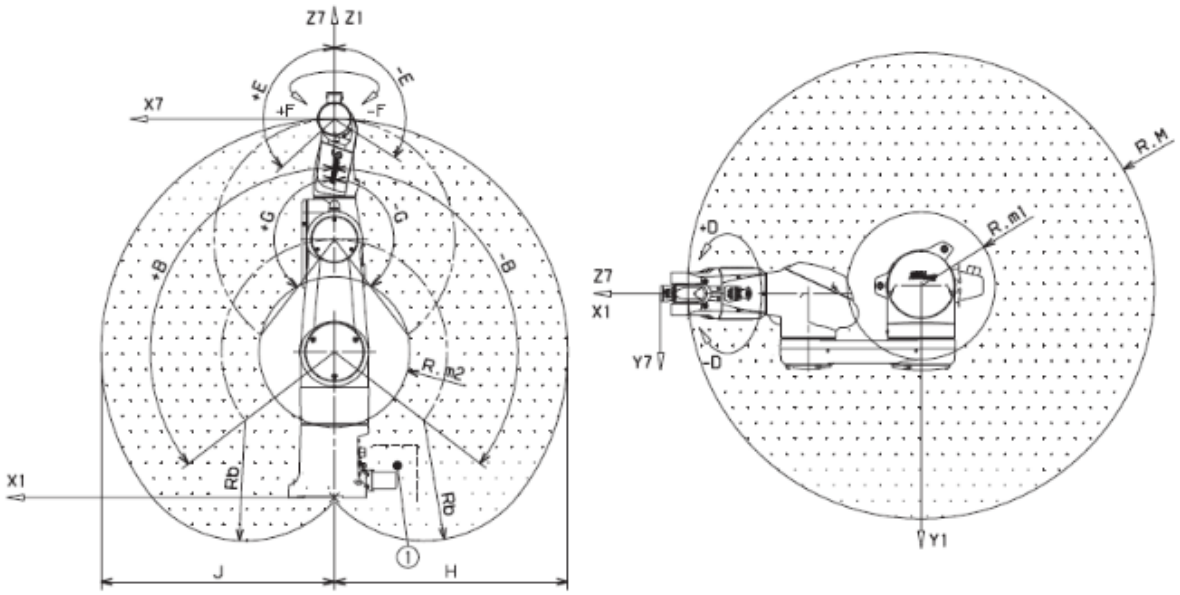


Figura 12: Espai de treball del braç

5.3.6.- Amplitud, velocitat i resolució

El braç Stäubli TX60 consta de sis articulacions i sis eixos de rotació, és a dir, de sis graus de llibertat que tenen els següents valors d'amplitud, velocitat i resolució:

Eix	1	2	3	4	5	6
Amplitud (°)	360	255	285	540	255	540
Distribució d'amplitud (°)	A ± 180	B ± 127.5	C ± 142.5	D ± 270	E + 133.5 - 122.5	F ± 270
Velocitat nominal (°/s)	287	287	431	410	320	700
Velocitat màxima (°/s)	373	373	500	968	800	1125
Resolució angular ($°.10^{-3}$)	0.057	0.057	0.057	0.114	0.122	0.172

La velocitat màxima (º/s) només es donara per condicions de carrega i inèrcia reduïda.

S'entén per resolució el mínim increment que pot acceptar la unitat de control del robot (CS8C), està limitada per la resolució dels sensors de posició, dels conversos A/D (analògic/digital) i D/A (digital/analògic), així com dels actuadors discrets.

5.3.7.- Carrega transportable

A velocitat nominal pot transportar fins a 3,5Kg i a velocitat reduïda pot transportar fins a 4,5Kg, en totes les configuracions i tenint en compte les inèrcies màximes. No obstant la càrrega màxima que pot transportar és de 9Kg.

5.3.8.- Velocitat

La velocitat a la que es pot moure l'extrem del braç d'un robot depèn de la càrrega transportada, estan inversament relacionats. Per tant la màxima velocitat del robot serà amb càrrega nul·la. La velocitat màxima en el centre de gravetat de la càrrega és de 8 m/s.

5.3.9.- Repetibilitat

La repetibilitat a temperatura constant és de ± 0.02 mm.

5.4.- Unitat de control CS8

L'armari de control CS8C o controladora és l'element que ens permet el control sobre el braç robòtic Stäubli TX60. La CS8C disposa d'un processador amb 64 MB de memòria RAM i un sistema operatiu en temps real encarregat d'executar els diferents processos de l'usuari i gestionar els moviments del robot. El llenguatge de programació per a la CS8C és el VAL3. La controladora disposa d'entrades i sortides digitals per interaccionar amb altres elements del sistema. En el nostre cas per comunicar-nos amb ella des d'un PC disposem de les interfícies de comunicacions sèrie RS232/422, CAN, USB o Ethernet.

En el nostre cas utilitzarem la xarxa Ethernet, tot i que també es podria utilitzar una xarxa dedicada. El principal motiu per utilitzar la xarxa interna recau en l'avantatge de poder utilitzar qualsevol dels PC's del laboratori on realitzem el treball. No obstant, s'ha de tenir en compte que perdem fiabilitat al sistema ja que augmentem la complexitat d'aquest. La xarxa Ethernet pot fallar degut a motius externs al nostre entorn de treball, impedint-nos el control directe sobre ella. Més endavant es comentarà detalladament el software desenvolupat per a les comunicacions entre PC i controladora CS8C.

5.4.1.- Components

L'armari de control està format per un processador (5), part intel·ligent de la instal·lació. El calculador guia al robot a través d'amplificadors de potència numèrics (1) dedicats a cada eix del braç. La conversió d'energia elèctrica l'efectuaran el bloc de potència PSM (7), l'alimentació RPS (2) i l'alimentació ARPS (3) que subministren als elements indicats més amunt, les tensions necessàries per el seu correcte funcionament a partir de la tensió subministrada per la xarxa elèctrica. Les indispensables funcions de seguretat elèctrica estan reunides a la targeta RSI (4).

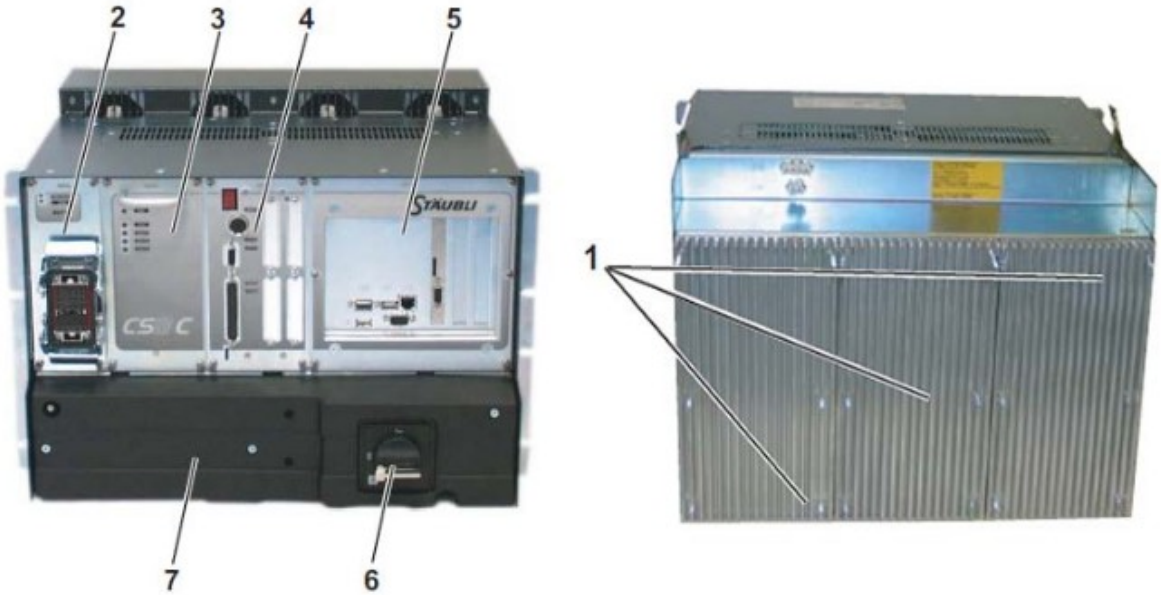


Figura 13: Armari de control CS8C

La posada fora de tensió s'obté posant l'interruptor general (6), situat a la part del davant de l'armari de control, en la posició 0. Només s'ha d'accionar quan s'ha aturat el funcionament del braç i s'ha tallat la potència del robot.

5.4.1.1.- Processador

El processador executa el programa del qual se'n deriven l'accionament del braç, la gestió d'entrades/sortides, etc. Els seus components principals són:

- Una targeta CPU sobre la qual hi ha muntat un Flash Disk de 64 MB per l'emmagatzemament del sistema VAL3 i dels programes de l'aplicació. Aquesta targeta està equipada en el panell frontal de les connexions Ethernet, USB i sèrie.

- Els ports USB "user" (J202 i J209) es poden utilitzar per claus que suportin els següents protocols: #1 reduce block commands (RBC) T10 Project 1240D, #4 UFI i #6 SCSI transparent command set.

- Una targeta STARC que assegura la interconnexió amb les transmissions en el CS8C i amb les targetes DSI al peu del braç mitjançant una comunicació numèrica. Es disposa igualment d'una entrada codificador (J305).

- Emplaçaments lliures per opcions de format PCI subministrades per Stäubli.

5.4.1.2.- Alimentació RPS 235

L'alimentació RPS235 genera la tensió contínua necessària pels amplificadors a partir d'una tensió alterna trifàsica. Quan la velocitat dels motors ha de disminuir, l'alimentació capaç d'absorbir la potència sobrant dels motors en una resistència de regeneració. L'alimentació també genera un senyal de control per la targeta RSI (estat de l'alimentació PWR-OK).

5.4.1.3.- Alimentació ARPS

L'alimentació ARPS genera tensions contínues per la part lògica de l'armari de control. Una tensió de 24 V per el processador, la targeta RSI i els amplificadors. Una tensió de 24 V per la ventilació de l'armari del robot. Una tensió de 24 V per els frens, aquesta tensió està present quan la targeta RSI subministra un senyal de comandament (BRK_REL_EN). Una tensió de 13 V per les targetes DSI situades al peu del braç. L'alimentació ARPS també genera senyals de control per la targeta RSI (estat de l'alimentació ALIM_OK, presència de la tensió a la xarxa SECTEUR_OK).

5.4.1.4.- Amplificadors

Cada amplificador està dedicat a 2 eixos i depèn dels tipus de motors accionats i de les característiques desitjades. Així, és possible que dos amplificadors siguin mecànicament idèntics, però no intercanviables.

La personalització d'una transmissió per un determinat eix està indicat per la referència escrita a la transmissió. Aquesta està alhora composta per una configuració de hardware i una configuració de software.

Els amplificadors estan alimentats a través de RPS235, per la part de potència i a través de ARPS, per la part lògica. Aquests s'associen a partir de la targeta STARC.

5.4.1.5.- Mòdul de potència (PSM)

Aquest conjunt, constituït per un transformador i proteccions associades, està situat al fons de l'armari de control. Segons el tipus d'armari de control, la connexió s'efectua en monofàsic o trifàsic, amb o sense transformador. Els fusibles de l'entrada tenen unes dimensions de 10,3 x 38 mm/500 V. Les sortides 230 V AC del transformador estan protegides per disjuntors monofàsics i trifàsics.

En el cas d'un transformador trifàsic multi tensió, l'elecció de la tensió es fa a la regleta del transformador. El disjuntor trifàsic alimenta la part de potència de l'armari de comandament a través dels contactors i de l'alimentació RPS325. El disjuntor monofàsic alimenta les part lògiques de l'armari de comandament a través de l'alimentació ARPS.

5.4.1.6.- Targeta RSI

La targeta RSI reuneix tots els elements per assegurar, en bones condicions de seguretat, la posada en tensió del braç o l'aturada d'emergència del mateix. Està equipada amb relés de seguretat redundants. Aquesta pot tenir fins a 32 entrades i 32 sortides digitals (opcions BIO). La targeta RSI assegura la continuïtat dels circuits d'aturada d'emergència, enllaçant els diferents contactes d'aturada d'emergència, l'estat dels quals està senyalitzat pel visor.

5.4.2.- Connexions

La connexió de les senyals d'entrada/sortida es fa a través dels connectors situats al panell frontal de l'armari de control (figura 3.6).

(2) Connector per la MCP
(3) Entrades/sortides ràpides
(4) Connexió amb la cèl·lula (parada d'emergència, porta, . . .)
(5) Opcions E/S digitals (BIO)
(6) Connexions Ethernet
(7) Connexions sèrie
(9) Connexions USB
(10) Braçalet antiestàtic
(11) Entrada codificador
(12) Sortida CAN per els robots SCARA

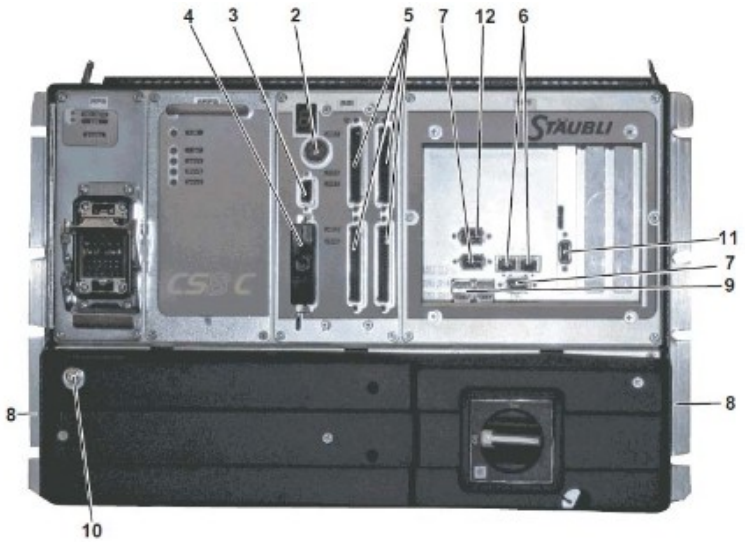


Figura 14: Panell frontal del CS8C

Les connexions de les senyals s'han de fer amb cables blindats, el blindatge dels quals ha d'estar connectat a les masses amb 2 extrems. Això és necessari tant per les senyals de parada d'emergència (J109) com per les connexions numèriques (connexions sèrie, ethernet, . . .). La fixació de l'armari de control participa també en la protecció contra paràsits. Per tant és útil que els muntants de fixació (8) estiguin connectats al pla de massa de la cèl·lula general.

5.4.2.1.- Enllaç Ethernet

La CS8C disposa de 2 ports ethernet J204 i J205. La direcció IP de cadascun es pot modificar des del panell de control. La modificació és efectiva immediatament. De fàbrica, el primer port està configurat amb la direcció 192.168.0.254 (màscara 255.255.255.0), el segon amb la direcció 172.31.0.1 (màscara 255.255.0.0). També es pot obtenir automàticament la direcció IP a partir de la xarxa mitjançant el protocol DHCP.

La controladora CS8C pot accedir a altres subxarxes ethernet mitjançant passarel·les configurables. Cada passarel·la està definida per: la direcció IP del perifèric utilitzat com a passarel·la i la direcció IP de la subxarxa que s'ha d'accedir.

L'armari de control CS8C és un servidor FTP que permet intercanviar fitxers per ethernet mitjançant el protocol FTP. N'hi ha prou en definir la direcció IP de l'armari de control del CS8C per poder accedir-hi a través de FTP, i utilitzar el login i contrasenya de xarxa corresponent a un perfil d'usuari definit al CS8C. El dret d'accés de lectura i escriptura depèn del perfil d'usuari seleccionat.

L'armari de control CS8C pot ser configurat per comunicar-se a través d'Ethernet mitjançant sockets TCP. L'armari suporta fins a 40 sockets simultanis en mode client i/o mode servidor. La configuració dels sockets Ethernet es fa des de l'aplicació panell de control (Control Panel > I/O > Socket). No es suporten els sockets UDP.

Un socket servidor s'activa (obert) en la CS8C quan l'utilitza un programa VAL3, i es desactiva (tancat) quan es desconnecta l'últim client. Quan s'arriba a la quantitat màxima de clients per un socket servidor, els altres clients que s'intenten connectar són acceptats, però la comunicació és interrompuda immediatament pel servidor.

5.4.2.2.- Port sèrie

Hi ha dues connexions sèrie disponibles a l'armari de control CS8C (J203 "COM1" i J201 "COM2") per intercanviar dades entre una aplicació VAL3 i un equip de la cèl·lula. La configuració de les connexions sèrie s'efectua des de la visualització de les entrades/sortides del panell de control i té paràmetres configurables.

5.4.3.- Consola de programació manual (MCP)

L'armari de control disposa d'una MCP (Manual Control Pendant) que permet a l'usuari el control sobre ella. La MCP està formada per una pantalla i un teclat tal com es pot observar en la figura 15. Per poder utilitzar la MCP aquesta ha d'estar col·locada en el seu pertinent receptacle, a prop del lloc de treball i fora de la cèl·lula de treball, o s'ha de mantenir polsat el botó de validació (#). Això es realitza lògicament per qüestions de seguretat.

Figura 15: Manual Control Pendant

5.4.3.1.- Pantalla de la MCP

La pantalla de la consola de programació manual (MCP) està dividida en tres zones tal com ens mostra la figura 16.

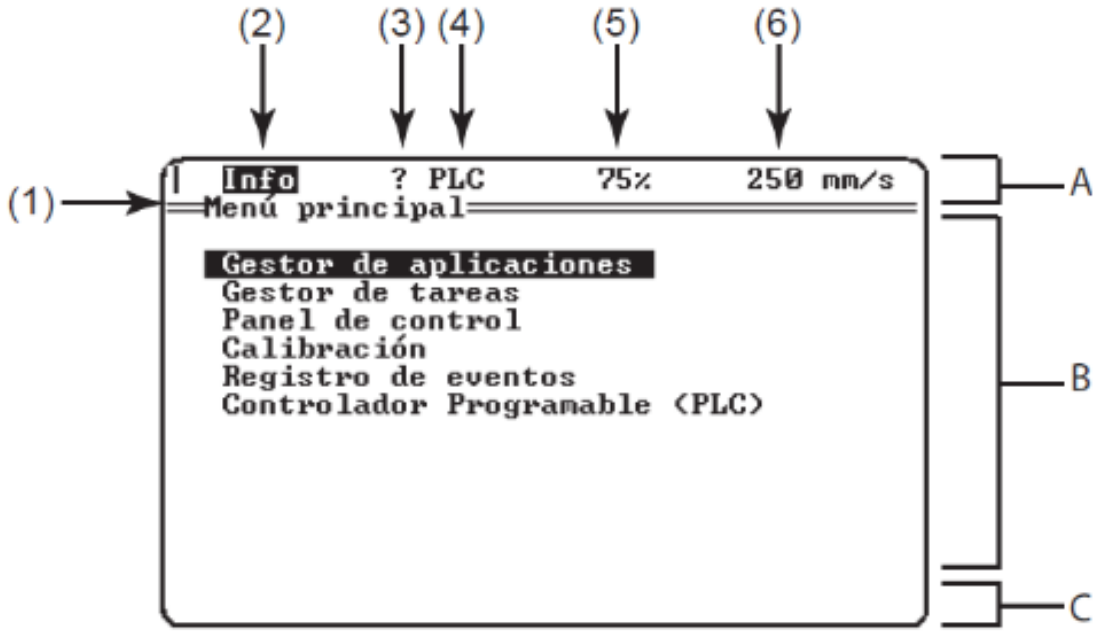


Figura 16: Pantalla de la MCP

La barra d'estat (A) proposa les informacions següents, sigui quin sigui l'estat de la navegació en curs:

(1) Indicador d'activitat del sistema: Quan l'indicador està present a la barra d'estat, el sistema no es troba disponible per l'operador.
(2) Indicador de presència de nous missatges d'informació: La seva presència indica que un o diversos nous missatges d'informació han estat emmagatzemats al registre d'esdeveniments. Aquest indicador apareix sempre parpellejant i es manté actiu mentre l'usuari no s'hagi adonat d'aquestes informacions.
(3) Indicador d'entrada: Apareix en mode parpelleig quan una aplicació VAL3 està en espera d'una entrada d'operador a la pàgina d'aplicació. Es manté actiu mentre l'aplicació estigui activa i no s'hagi efectuat l'entrada.
(4) Indicador de funcionament de l'autòmat programable (PLC).
(5) Indicador de velocitat de desplaçament del braç: S'aplica a tots els moviments (manuals i programats).
(6) Indicador de velocitat màxima en mode prova.

La pàgina de treball (B) és la zona de la pantalla situada entre la barra d'estat i la zona de menús contextuals. És damunt d'aquesta pàgina on s'intercanvien totes les informacions relatives a l'aplicació en curs (visualització, finestres d'informacions, entrades). La pàgina de treball té sempre un títol situat a la línia immediatament a sota de la barra d'estat.

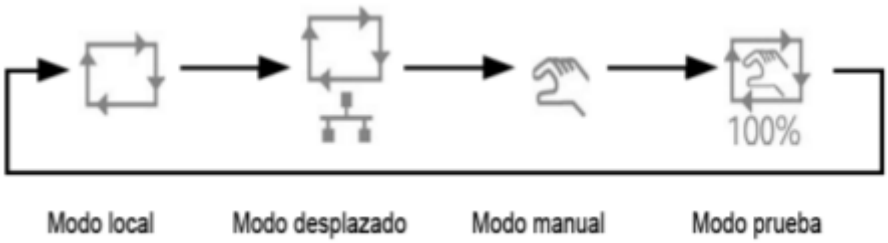
Els menús contextuals (C) permeten efectuar una acció específica a l'element seleccionat o a la pàgina de navegació. Per activar l'acció, n'hi ha prou amb prémer la tecla situada a sota l'etiqueta corresponent.

5.4.3.2.- Teclat de la MCP

A continuació es farà una petita presentació del teclat de la MCP i la seva utilitat. No obstant, no es pretén aquí mostrar de manera detallada el funcionament de la MCP, però si ensenyar al lector les seves principals característiques. Conèixer mínimament el MCP és indispensable per a la utilització del software desenvolupat en aquest projecte.

Figura 17: Teclat de la MCP

(1) Botó de selecció del mode de funcionament: Aquest botó permet seleccionar un dels quatre modes de funcionament (mode prova, mode manual, mode local i mode desplaçat). El mode seleccionat està indicat al voltant del botó davant dels pictogrames dels modes de funcionament.



(2) Botó d'aplicació de potència del braç: Aquest botó lluminós permet aplicar o tallar la potència del braç. L'indicador verd encès fix indica que l'aplicació de potència és efectiva. En mode manual o en mode prova, si la MCP no està col·locada sobre del seu suport, s'ha d'utilitzar el botó de validació (11).

(3) Botó de parada d'emergència: La parada d'emergència només s'ha d'utilitzar en cas de necessitat absoluta per una aturada no prevista en una aplicació.

(4) Tecles de moviment: Aquestes tecles estan actives en el mode manual i permeten generar moviments del braç per eix o en els punts de referència segons el mode de desplaçament escollit.

(5) Tecles d'elecció del mode desplaçament: Quan el braç està sotmès a potència i en el mode manual, cadascuna d'aquestes quatre tecles permet seleccionar el mode de desplaçament escollit (joint, frame, tool o point). L'indicador associat a la tecla indica el mode en curs.

(6) Tecla de reglatge de velocitat: Aquesta tecla permet variar la velocitat dins dels límits imposats pel mode de desplaçament seleccionat. Pot ser desactivada segons el perfil d'usuari en curs. La indicació de la velocitat es mostra a la barra d'estat de la pantalla de la MCP.

(7) Tecles de menús contextuals: S'utilitzen per validar els menús contextuals mostrats al seu damunt.

(8) Tecles alfanumèriques: Aquestes tecles serveixen per entrar dades a l'aplicació.

(9) Tecles d'interfície i de navegació: Aquestes tecles permeten navegar per la pantalla de la MCP i accedir a l'ajuda.

(10) Tecles de comandament de les aplicacions: Aquestes tecles s'utilitzen per posar en marxa/aturar una aplicació i per validar els moviments del braç.

(11) Botó de validació: Aquest botó té tres posicions i acciona contactes que estan oberts quan el botó no està activat, tancats a la posició mitja i oberts en una posició completament enfonsada que correspon a una crispació de l'usuari. Aquests contactes es mantindran oberts fins que es deixi anar el botó. Aquest botó permet autoritzar l'aplicació de potència al braç en mode manual sols quan està a la posició mitja. Les altres dues posicions impedeixen l'aplicació de potència o provoquen el tall de potència si el braç està sotmès a tensió en mode manual. En el mode automàtic, la posició del botó no es té en compte.

(12) Tecles d'activació de sortides digitals: En mode manual, aquestes tecles fan que es commuti l'estat de les sortides digitals que hi estan associades.

(13) Tecles de jog: Aquestes tecles estan actives en el mode manual i permeten generar moviments del braç, per eix o el els punts de referència segons el mode de desplaçament escollit (joint, frame, tool), amb una sola mà.

5.5.- Sensor de força Schunk FTCL 50-80

La incorporació sensorial en robots industrials ens permet dotar al robot d'una certa intel·ligència, la qual cosa ens possibilita la creació d'aplicacions amb percepció de l'entorn de treball. Els sistemes sensorials més usats en robots industrials són els que incorporen sensors de força i sensors de visió, essent el primer el més desenvolupat en la indústria. En aquest apartat no es pretén descriure els tipus de sensors que existeixen i el seu funcionament. Es pot trobar informació detallada sobre sensors en el capítol 7 del llibre, "Robótica: Manipuladores y robots móviles", comentat a la bibliografia.

Tot i això, és important saber que segons la quantitat d'informació que donen els sensors de força es poden classificar en:

- De 1 grau de llibertat: Donen informació sobre una component de força. Aquests sensors són molt econòmics i usualment es tracta de galgues extensomètriques.

- De 3 graus de llibertat: Donen informació sobre les tres components de força.

- De 6 graus de llibertat: A més de les forces, proporcionen informació sobre els moments.

5.5.1.- Característiques

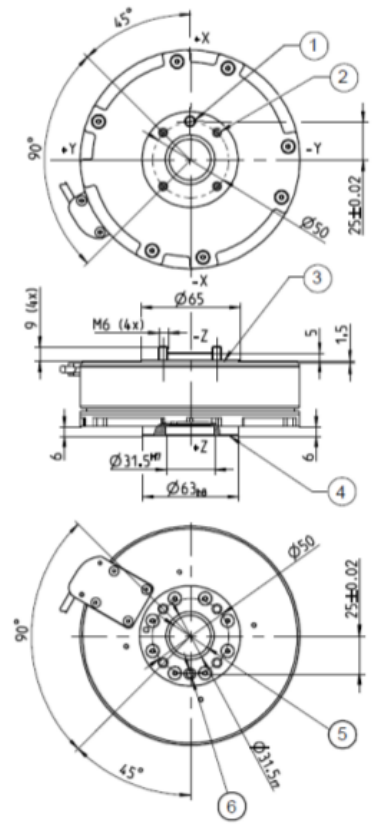
La integració sensorial al braç robòtic Stäubli TX60 es realitza mitjançant un sensor de força FTCL 50-80 de Schunk (figura 18). Aquest sensor és un sensor digital de sis graus de llibertat i s'ubica entre el puny del braç del robot i l'element terminal. En recull les forces i moments exercits sobre ell.

Figura 18: Sensor FTCL 50 - 80 de Schunk

5.5.1.1.- Dades tècniques

Pes total	0,96 kg
Temperatura de funcionament	+5 °C a 55 °C
Interfície, robot/sensor	ISO 9409-1-A50
Interfície, eina/sensor	ISO 9409-1-A50 i PZN 64 (eina)
Àrea de moviment de translació (X, Y, Z)	± 1,4 mm
Àrea de moviment de rotació (α, β, γ)	± 1,4 °
Rang de mesura de translació (X, Y, Z)	± 1,0 mm
Rang de mesura de rotació (α, β, γ)	± 1,0 °
Freqüència de mesura	1 kHz (1 ms)
Font d'alimentació	10 VDC a 26 VDC
Consum elèctric màxim	1,8 W
Interfícies elèctriques	RS232 (RS485 a demanda)
	CAN (DeviceNet a demanda)

5.5.1.2.- Dimensions



1 Centering pin Ø 6 H/ 7
2 M6 / 7 (4x)
3 Mounting surface (robot-side) with ISO 9409-1-A50 interface
4 Mounting surface (tool-side) with ISO 9409-1-A50 interface
5 Ø 26 Fully through
6 Centering pin Ø 6.5 $_{0}^{+0.2}$ / 8.5

Figura 19: Dimensions del sensor

5.5.2.- Connexió

El sensor FTCL 50-80 incorpora un cable per tal de poder connectar-se amb altres elements. No obstant, aquest cable disposa d'un terminal específic de la marca Schunk que no pot connectar-se directament a un PC, essent necessari un adaptador especial de la marca Schunk. Degut a la manca d'aquest adaptador, s'ha creat un nou cable que ens connectarà directament el sensor de força amb el la unitat de control CS8C utilitzant un terminal sèrie RS232. En el terminal sèrie RS232 hi connectarem també els cables de la font d'alimentació per poder alimentar el sensor. Les connexions entre el terminal específic del sensor, el terminal RS232 de l'unitat de control i la font d'alimentació són:

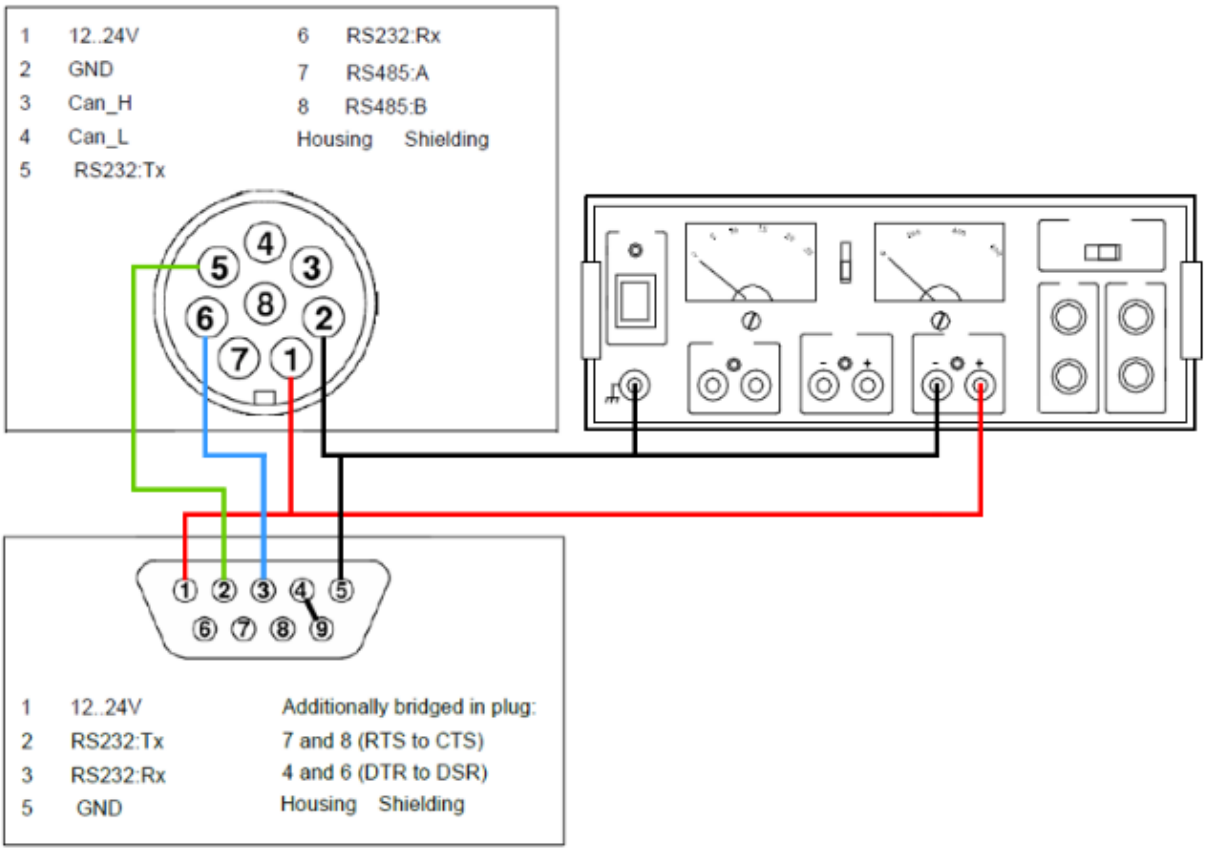


Figura 20: Connexions entre els terminals

Per comprovar que el sensor rep l'alimentació adequada (entre 10 i 26 V), el sensor disposa d'un LED d'operació que pot mostrar els següents estats:

- Verd: Sensor preparat per operar.
- Vermell: No alimentació o existeix un error en totes les interfícies.
- Parpelleig verd/vermell: Una de les dues interfícies no treballa bé.
- Parpelleig irregular vermell/verd: Firmware del sensor defectuós.
- Parpelleig irregular verd: Transferència de dades.
- No senyal: Sensor apagat o en mode flash.

5.5.3.- Sistema de referencia

El sensor FTCL 50-80 mesura les forces i moments aplicats respecte un sistema de referència que anomenem {S} i que és intrínsec al sensor (figura 21). És important tenir present aquesta referència ja que tota la informació que aportarà el sensor serà respecte {S}. El sensor mesura tres forces (fx, fy, fz) i tres moments (mx, my, mz) en les tres direccions XYZ.

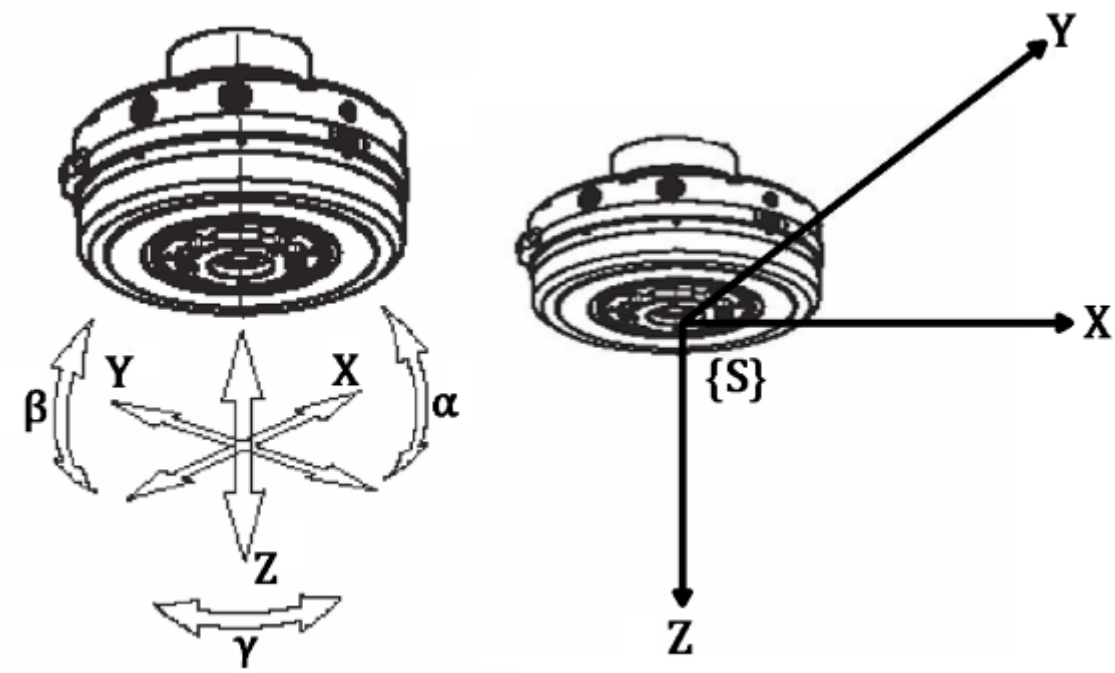


Figura 21: Sistema de refèrencia {S} del sensor

5.6.- Joystick Logitech Extreme 3D Pro

En el sector industrial és molt freqüent trobar diversos elements que es manipulen a traves de joysticks. Normalment els joysticks industrials són molt limitats, a més, la gran majoria no disposen de botonera.

En el sector dels videojocs els joystick són molt més complerts, és per això que hem decidit treballar amb material més orientat a aquest sector. Els comandaments en aquesta àrea són més complerts: 6 - 8 eixos, panells amb més de 6 - 10 botons, acceleròmetres, etc. Pel nostre sistema hem escollit el *Logitech Extreme 3D Pro* que podem observar a la figura 22.

Figura 22: Joystick Logitech Extreme 3D Pro

Un cop el sistema està executant-se a la controladora CS8C i el programa està executant-se al PC, cal observar que aquest aparell és l'element del sistema amb el qual l'usuari té més interacció, ja que mitjançant els moviments del joystick som capaços de maniobrar en temps real el braç robòtic.

Quasi tota la interacció del sistema es realitza a traves del joystick:

- Moure el braç robòtic en temps real
- Modificar el mode de moviment
- Alternar entre les diverses maniobres programades
- Canviar el llindar de forces establer inicialment
- Parar / Reprendre el moviment
- Acabar l'execució del programa

5.6.1.- Característiques

El joystick és l'element que permet la interacció entre l'usuari i el robot. Disposa de diversos eixos i botons que permeten realitzar maniobres diverses a partir de la programació de diverses tasques. Disposa d'una base ample i una palanca ergonòmica per millorar la maniobrabilitat.

5.6.1.1.- Sistema d'eixos

Aquest model de Logitech incorpora 6 eixos que treballen en el domini [-32767..32767]. A la figura 23 podem observar els eixos, dels quals trobem 3 a la palanca, 2 al HAT i el darrer situat a l'acceleròmetre.

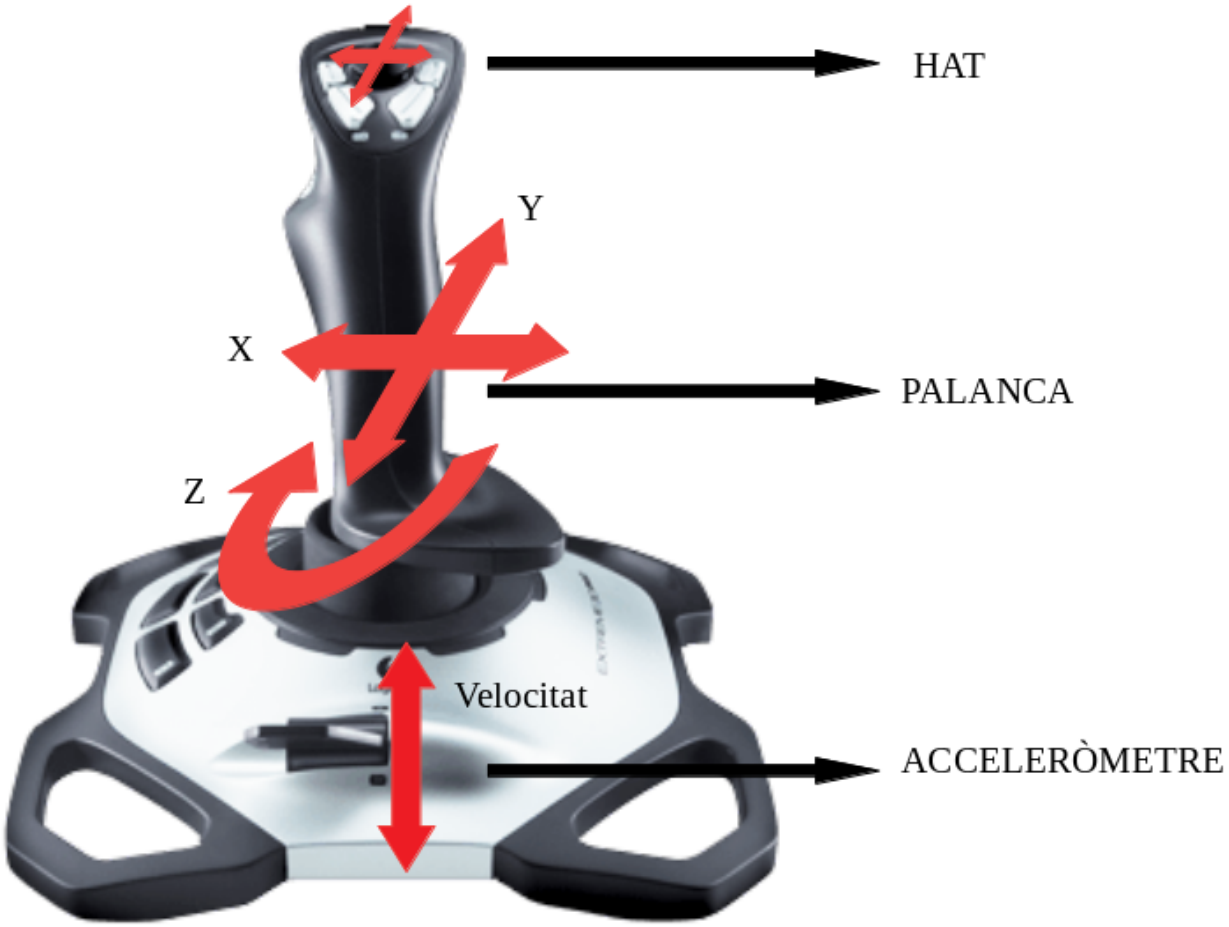


Figura 23: Configuració dels eixos del joystick

Cal remarcat, que els eixos del HAT, tot i considerar-se eixos, funcionen com botons, ja que el seu domini és [-32767, 0, 32767]. Aquest fet es degut a que en el sector dels videojocs, aquests eixos són emprats per alternar les vistes en simuladors de vol, canviar de menús/armes en jocs, entre altres funcions, per això no és necessari un major domini.

Els eixos XYZ s'encarreguen del moviment, tant en els jocs, com en el nostre sistema, mitjançant un sistema de referencia. L'acceleròmetre es emprat per modificar la velocitat de moviment a voluntat.

5.6.1.2.- Distribució dels botons

La distribució de la botonera, que està composta per un total de 12 botons, la podem observar a la figura24.

Figura 24: Configuració dels botons del joystick

El joystick distribueix els botons de la següent manera:

- Botó 1: Trigger, per parar / reprendre el moviment del robot
- Botó 2: Exit, finalitza l'execució del programa
- Botó 3 – 4: Selecció del tipus de moviment, JOINT o FRAME respectivament
- Botó 5: Modificador del llindar de les forces al eixos XYZ
- Botó 6: Alternar entre els modes de maniobrabilitat
- Botó 7 – 12: Només disponible en mode JOINT, selecció entre els 6 eixos del Stäubli TX60

5.6.2.- Connexió

La connexió del joystick al PC es fa a traves d'un cable USB. No es necessària la instal·lació de cap driver ja que es realitzara la lectura directament a traves del dispositiu.

5.6.3.- Requisits dels sistema

- *Sistema Operatiu:* Windows® 8, Windows 7 o Windows Vista® o Linux (cal intervenció de l'usuari)
- *Espai disponible al disc dur:* 70 MB
- *Connexió:* Port USB

5.7.- Sistema de visió

Els sistemes de visió són un dels més usats en el mon industrial, ja sigui per realitzar controls de qualitat, monitoritzar la producció d'una empresa, teleoperacions, etc. Existeixen diversos tipus de sistema en funció de la finalitat de l'operació:

- *Sistemes d'una dimensió (1D):* Principalment controls de qualitat
- *Sistemes de dos dimensions (2D):* Control de producció, teleoperacions, etc
- *Sistemes de tres dimensions (3D):* Localització d'objectes a l'espai, generació de volums 3D

En el nostre cas, farem servir dos sensors òptics de tres dimensions per poder obtenir una millor percepció de l'espai de treball i les maniobres que es realitzen de manera remota.

5.7.1.- Components

El sistema s'ajuda de dos sensors òptics per tal de poder teleoperar el robot sense necessitat de tenir línia visual directe. Els dispositius emprats són dos càmera Logitech C170 que podem apreciar a la figura 25.

Figura 25: Càmera Logitech C170

5.7.2.- Especificacions

Com ja s'advertia al principi del document, fem us del material disponible al laboratori, i com a conseqüència d'això ens trobem que els dispositius òptics no són els més adients pel sistema, tot i que compleixen amb els objectius establerts.

- *Resolució màxima:* 720p/30fps
- *Tipus d'enfocament:* focus fixe
- *Tecnologia de la lent:* estandard
- *Microfon integrat:* mono
- *Camp visual:* 58 º
- *Longitud del cable:* 0,95 m

5.7.3.- Tipus de sistema

L'estereoscòpia és qualsevol tècnica capaç de recollir informació visual tridimensional o de crear la il·lusió de profunditat en una imatge.

L'objectiu d'usar 2 sensors òptics pel nostre sistema és simular una visió estereoscòpica per poder distingir profunditat i volum. D'aquesta manera podríem realitzar maniobres més precises. Una de les càmeres es trobara acoblada a l'element terminal del robot, mentre que l'altre romandrà fixa, d'aquesta manera, quan els camps visuals de les dues càmeres es creuin, simularan una visió estereoscòpica oferint una millor percepció de l'entorn de treball, tot i no ser pròpiament un sistema estereoscòpic.

La figura 26 ens mostra un diagrama de com seria un sistema de visió estereoscòpic:

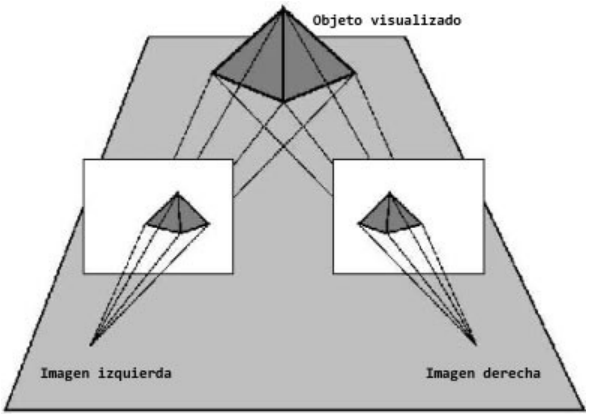


Figura 26: Esquema d'un sistema de visió estereoscòpic

Com podem observar, tenim 2 imatges d'un mateix objecte des de diferents perspectives. Les imatges per separat no ens donen informació de volum o profunditat, però si les processem conjuntament obtenim una imatge amb aquestes dos característiques.

A continuació es mostra un exemple que reflecteix l'explicació anterior:

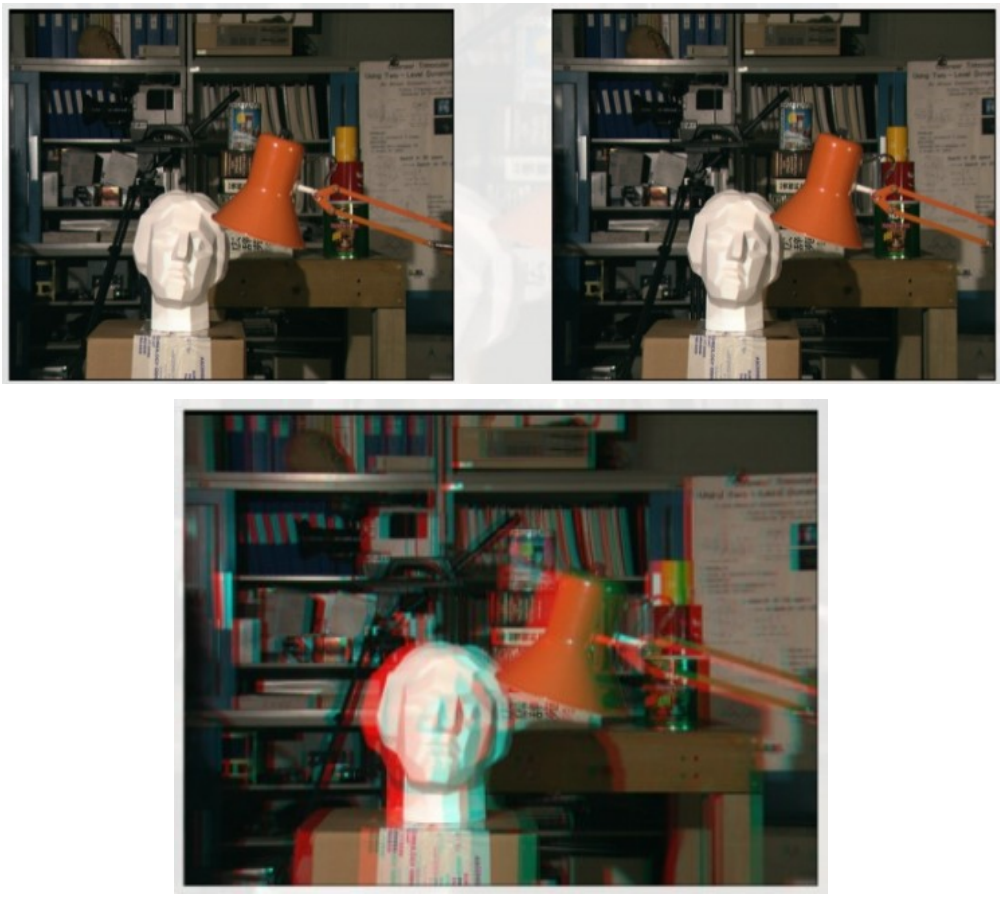


Figura 27: Resultat del processament d'imatges per un sistema estereoscòpic

5.8.- Ordinador personal

És un dels elements més importants del sistema, ja que conforma la unitat de control superior. Mitjançant el PC, l'usuari programa les diferents aplicacions en llenguatge VAL3, que posteriorment es transferiran a la controladora CS8C i finalment al robot Stäubli TX60.

L'usuari pot realitzar diverses tasques:

- Programació i modificació d'aplicacions VAL3
- Simulacions d'aplicacions
- Transferència d'aplicacions
- Altres

Totes aquests tasques es realitzen a traves del software Stäubli Robotics Studio (SRS) de gestió de robots Stäubli el qual es troba instal·lat al PC.

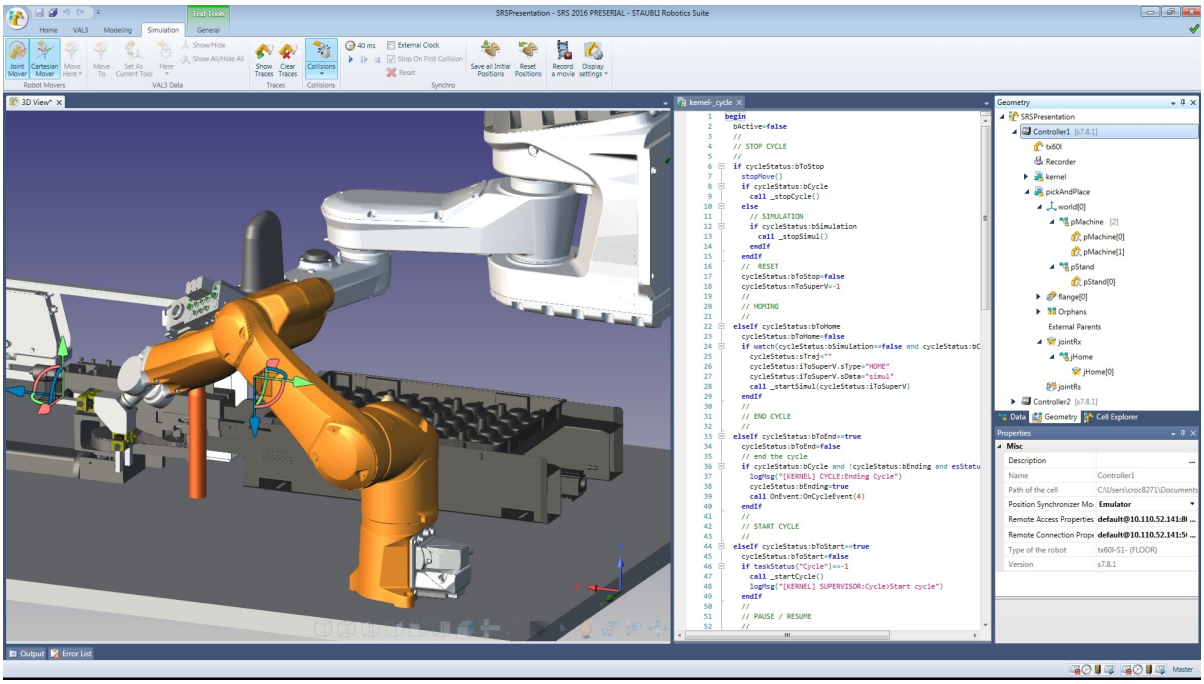


Figura 28: Stäubli Robotics Studio (SRS)

El PC es comunica amb la unitat de control CS8C via Ethernet, ja que tant el PC com la CS8C estan connectats a la xarxa local (LAN) del laboratori de robòtica de l'EPS on es desenvolupa aquest projecte. D'aquesta manera des de qualsevol PC connecta a la xarxa LAN del laboratori i amb el software requerit instal·lat pot accedir al control del robot Stäubli TX60.

Les característiques tècniques del PC i les característiques de la xarxa LAN són les següents:

- *Processador:* Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz 3.60 GHz
- *Memòria RAM:* 8.00 GB (7.87 utilitzable)
- *Sistema operatiu:* Windows 7 Enterprise Service Pack 1 (64 bits)
- *Software instal·lat:* Stäubli Robotics Studio (SRS)
- *PC connectat al port Ethernet* #1047 amb IP 84.88.129.183
- *CS8C connectada al port Ethernet* #1045 amb IP 84.88.129.201

6.- REQUISITS DEL SISTEMA

Aquest treball ha d'aconseguir implementar un sistema que permeti maniobrar un braç robòtic industrial per realitzar tasques mediques i algunes operacions bàsiques de cirurgia. En aquest apartat entrem en més detall sobre les característiques que es requereixen.

6.1.- Requisits funcionals

Un requisit funcional defineix una funció del sistema de software o dels deus components. Una funció és descrita com un conjunt d'entrades, comportaments i sortides. Els requisits funcionals poden ser: càlculs, detalls tècnics, manipulació de dades i altres funcionalitats especifiques que es suposa que el sistema deu complir.

6.1.1.- Requisits de hardware

L'apartat presenta els requeriments dels components i de les connexions necessaris pel correcte funcionament del sistema.

6.1.1.1.- Components

Com ja introduíem en apartats anteriors, el sistema està orientat a un entorn de treball molt especific, això implica l'ús de components hardware molt específics en marca, model i versió, que cal complir perquè el software que es desenvolupara funcioni correctament. A continuació s'enumeren els components i els requeriments pertinents que cal complir:

COMPONENT	MARCA	MODEL	VERSIÓ
Braç robòtic	Stäubli	TX60	Standard
Unitat de control	Stäubli	CS8C	5.4.3
Sensor de forces	Schunk	FTCL	50 - 80
Joystick	Logitech	Extreme	3D Pro
Sensors òptics	Logitech	C170	Standard

Un altre component molt important pel sistema però que no es pot emmarcar en aquesta taula, és l'ordinador. Tot seguit es presenten els requisits de hardware que ha de complir:

- Espai mínim de disc dur de 10 GB
- Memòria RAM mínim de 4GB
- Mínim 5 ports USB 2.0 o superiors
- Motherboard amb port Ethernet

6.1.1.2.- Canals físics de comunicació

Els diversos components que conformen el sistema necessiten un canal de comunicació físic per la transmissió de dades de manera eficient i fiable.

Podem dividir el sistema en 2 grups de manera física: dins i fora de la fotocèl·lula de seguretat de l'àrea de treball del robot. Dins la fotocèl·lula trobem la unitat de control CS8C, el robot Stäubli TX60, el sensor de forces Schunk FTCL 50-80 i els sensors òptics (aquest pertanyen al grup de fora la fotocèl·lula), mentre que en l'exterior de la fotocèl·lula situem l'ordinador, la consola MCP i el joystick.

- Connexions USB

  Tant el joystick com els sensors òptics es connecten al PC a traves de cables USB 2.0 o superiors. La lectura del joystick es realitza mitjançant software implementat en l'aplicació d'usuari del sistema, mentre que la lectura dels sensors òptics requereix un driver. Aquests assumptes s'exposen en el següent apartat (6.1.2) amb més detall.

Figura 29: Connexió USB

- Connexió Ethernet

  És un dels canals de comunicació més importants del sistema, ja que permet la comunicació entre la part exterior amb l'interior de la fotocèl·lula, atorgant d'aquesta manera poder a l'usuari per maniobrar el robot a voluntat. La comunicació es realitza entre l'ordinador que controla l'usuari (part externa) i la unitat de control CS8C (part interna). Per aquesta connexió fem servir la xarxa del laboratori que ja està preconfigurada (especificacions a l'apartat 5.7).

Figura 30: Connexió Ethernet

  - **ATENCIÓ!!**

    Es de vital importància assegurar que la consola MCP es troba fora de l'àrea de treball del robot (fora la fotocèl·lula) en el moment d'executar el sistema. La comunicació amb la unitat de control CS8C depèn de la xarxa del laboratori sobre la qual no tenim control directe. Si per algun factor extern la xarxa Ethernet falles i el robot acabes d'iniciar una maniobra indicada per l'usuari, no hi hauria manera possible a traves del sistema d'aturar la maniobra per evitar la causa de danys materials i/o físics. En el cas improbable de trobar-nos en aquesta situació cal actuar ràpidament executant una de les següents maniobres d'aturada, essent la primera més segura que la segona:

    - **Mètode de parada 1: Prémer el boto vermell de tall de potencia (figura 17 de l'apartat 5.4.3.2) que trobem a la part superior dreta de la consola MCP. Això talla de manera immediata el subministre de potencia al braç robòtic impedint el moviment i finalitzant la maniobra en execució.**
    - **Mètode de parada 2: Intervenir de manera física entre els sensors de la fotocèl·lula (Annex [B]). Tot i ser més perillós, ja que ens trobem més pròxims al robot, aquesta acció talla de manera immediata el subministre de potencia al braç robòtic impedint el moviment i finalitzant la maniobra en execució.**

- Connexió Serie RS232

  El sensor de força disposa d'una connexió serie RS232 que connectem directament a la unitat de control CS8C per la interacció directe en funció d'uns paràmetres definits per l'usuari. D'aquesta manera obtenim una resposta més rapida que si les dades del sensor s'haguessin de processar a la part de l'usuari i enviar el resultat a la unitat de control per executar la maniobra adient.

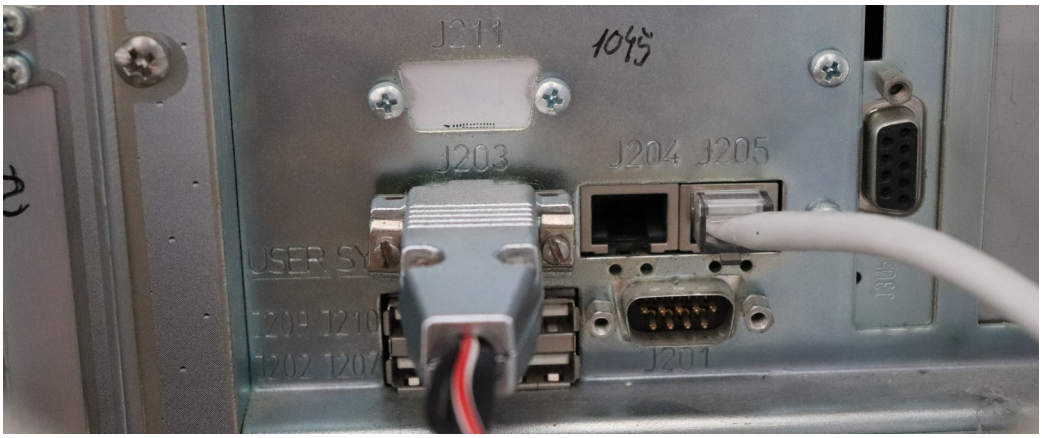
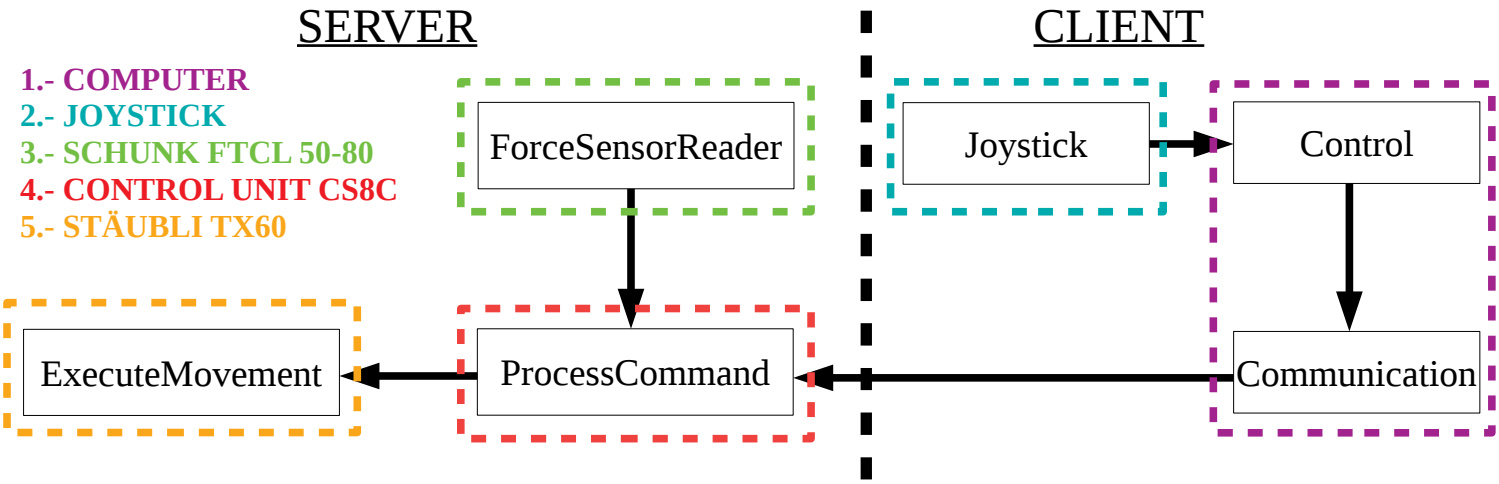


Figura 31: Connexió serie RS232 del CS8C

6.1.2.- Requisits de software

Aquest apartat ofereix una primera aproximació del sistema a nivell de software, exposant a grans trets les funcions que realitzara. El següent esquema representa de manera esquemàtica quins són els mòduls del software i com interactuen entre ells.



6.1.2.1.- Aplicació client

Per aconseguir el sistema descrit anteriorment, ens cal una aplicació de client que permeti a l'usuari interactuar amb aquest. Aquesta aplicacio es troba en l'ordinador i a de permetre a l'usuari executar les següents tasques:

- Configuració dels diversos parametres (forces, velocitats, modes de maniobra, etc)
- Maniobrar el robot de manera telematica per executar diverses operacions
- Comunicació bidirecional amb la unitat de control CS8C
- Pausar/Reprendre la maniobra de moviment del braç
- Finalitza l'execució del sistema

6.1.2.2.- Aplicació servidor

El sistema ha de disposar d'una aplicació servidor, aquesta espera rebre peticions del client per satisfer-les. Aquesta part del sistema ha de ser aliena a l'usuari, actuant com una caixa negra, per tant, no cal saber com funciona, només entendre que per certes peticions del client obtenim certes respostes del servidor. En funció de la petició, les respostes es poden manifestar de diverses maneres: moviment, senyal lumínica, inactivitat, etc. L'aplicació ha de ser capaç de realitzar les següents tasques:

- Processar les peticions rebudes pel client
- Realitzar un control de seguretat mitjançant lectures del sensor de força i actuar en funció dels paràmetres configurats
- En cas de ser necessari per algun mode d'operació, realitzar consultes a l'usuari en temps d'execució per la presa de decisions

6.1.2.3.- Sistema de maniobra

El sistema de maniobra és un dels pilars fonamentals del treball, ja que ha de permetre la maniobra eficient, en temps real i de manera remota del braç robòtic. Els robots industrials estan pensats per ser programats i executar la mateixa tasca infinitament fins que algun dels seus components es faci malbé, per això disposen d'eines que faciliten els moviments punt a punt. Per realitzar el sistema de maniobra integrem un joystick que ha de permetre executar les següents tasques:

- Moure el robot en diverses direccions/eixos
- Modificar la velocitat de moviment
- Alternar entre els modes de maniobra

6.1.2.4.- Sistema de control

El sistema de control ha d'oferir a l'usuari la possibilitat de supervisar el sistema de manera activa i/o passiva. En tots els entorns on intervé maquinaria, ja sigui de tipus mèdic o industrial, entre d'altres, existeixen sistemes de control. Els sistemes de control tenen diverses finalitats, supervisió de qualitat, monitorització de la producció, millores de la seguretat en l'entorn de treball, etc. En el nostre cas, el sistema de control té com objectius:

- Evitar malmetre l'entorn de treball
- En els moments de simular operacions de pacients, garantir la integritat d'aquests últims
- Evitar executar maniobres que excedeixin algun dels paràmetres establers
- Oferir una visió de l'entorn de treball de manera remota per si l'usuari no disposes de línia visual directe amb el robot i/o el pacient

Com podem observar, el sistema de control que es basa en el sensor de força serà el sistema actiu, ja que interactua de manera activa amb l'entorn de treball, mentre que el sistema de control basat en els sensors òptics, serà passiu, oferint només informació visual a l'usuari però sense tenir cap intervenció directe amb l'entorn de treball.

6.2.- Requisits no funcionals

Un requisit no funcional o atribut de qualitat és, a l'enginyeria de sistemes i l'enginyeria de software, un requisit que especifica criteris que poden ser usats per jutjar l'operació d'un sistema en lloc dels seu comportament específics, ja que aquest corresponen als requisits funcionals. Per tant, es refereixen a tots els requisits que no descriuen informació que guardar, ni funcions a realitzar, sinó característiques de funcionament, per això, sovint es denomine atributs de qualitat d'un sistema.

6.2.1.- Fiabilitat

Com en tots els projectes que es poden portar a terme en qualsevol àmbit, és molt important desenvolupar-los de forma fiable. En el cas d'aquest treball, la responsabilitat cau en el desenvolupador i ja que l'objectiu del treball és desenvolupar un sistema dedicat que faci de fonaments per la construcció posterior d'un sistema més gran, podríem dir que és més important que mai erigir una base el més fiable i estable possible. Per aquest motiu s'ha desenvolupat aquest projecte amb deteniment i precisió, comprovant cada modificació que es feia diverses vegades. Amb aquest mètode s'ha intentat assegurar la correctesa de tots els elements desenvolupats tot i que sempre es pot fallar i aquest treball no ha sigut una excepció, s'ha fallat en alguns casos i s'han hagut de solucionar contratemps que han sorgit durant la implementació.

6.2.2.- Eficiència

Un dels aspectes rellevants del sistema que pretenem desenvolupar és l'eficiència. Cal que l'interval de temps que transcorre entre l'acció de l'usuari i la resposta del robot sigui mínima, ja que el sistema ha de maniobrar el robot en temps real. No només les comandes de maniobra, sinó també les comandes de configuració de paràmetres, cal que s'executin i s'actualitzin en tot el sistema en el menor temps possible. La unitat de temps per processar les dades en tot el sistema hauria de ser, com a màxim i en el pitjor dels casos, de 8 mil·lisegons.

6.2.3.- Seguretat industrial

Quan es treballa amb aparells que tenen una velocitat màxima de moviment de 8 m/s cal prendre la seguretat molt seriosament, ja que els riscos poden arribar a ser mortals en els pitjors dels casos. Per sort, en el nostre cas, maniobrarem a velocitats molt reduïdes, no obstant, aplicarem els mateixos procediments de seguretat que si maniobréssim a altes velocitats. Cal que l'àrea de treball on maniobra el robot es trobi aïllada mitjançant una gàbia amb control d'accés o a traves de fotocèl·lules, que deshabilitin la potencia del braç, bloquejant tot el moviment i suprimint les tasques pendents d'execució, en cas d'accés mentre el robot realitzava maniobres.

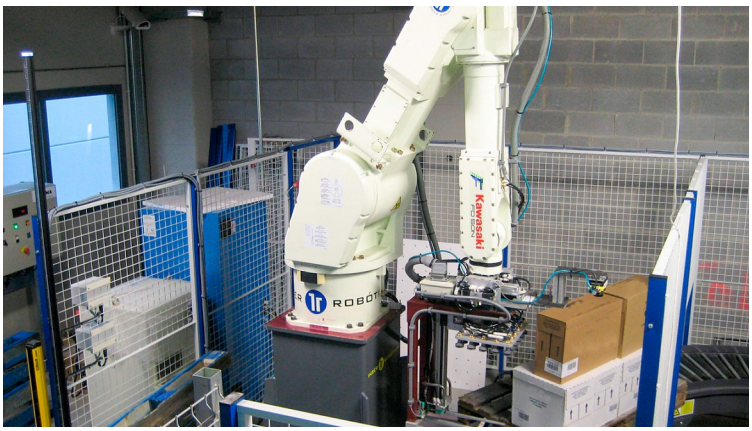


Figura 32: Sistemes de seguretat

7.- ESTUDIS I DECISIONS

L'estudi meticulós, la recerca i investigació de les eines més adients que s'utilitzaran al llarg del desenvolupament del sistema és un procés molt important, ja que pot marcar la diferencia entre assolir els objectius amb èxit o patir un fracàs rotund. L'elecció del les eines correctes comporta un estalvi de temps i esforç substancial en el moment de la implementació. Pel contrari, escollir una eina sense haver-la analitzat en profunditat anteriorment, pot comportar complicacions, endarrerint els temps establerts per cada tasca, i en el pitjor dels casos, pot comportar descartar la feina realitzada i plantejar-se el projecte des de l'inici.

7.1.- Selecció del sistema operatiu

El laboratori on es desenvolupara el projecte consta d'ordinadors hibernats que disposen de tot el software necessari per poder desenvolupar les practiques que és realitzen en aquest entorn. No obstant, tots disposen de Windows 7 com a sistema operatiu.

Pel desenvolupament d'aquest sistema, hem invertit una quantitat considerable de recursos temporals en l'estudi de quin sistema operatiu era el més adient. A continuació es presenta l'anàlisi realitzat on s'han valorat els pros i els contres d'escollir un sistema Windows o un sistema Linux.

WINDOWS

- Avantatges
  - Sistema més popular
  - Disposa d'un gran ventall de software
  - Els ordinadors del laboratori ja disposen d'aquest sistema operatiu
  - El software Stäubli Robotics Studio (SRS) només funciona amb sistema Windows
  - La instal·lació de drivers i altres programes és senzilla

- Contres
  - El cost és molt elevat
  - Les noves versions requereixen molts recursos
  - Sistema operatiu amb tendències suïcides (*pantallazo azul*)
  - La majoria dels virus estan dissenyats per atacar aquests sistemes
  - Programar la lectura d'una port físic es molt complexe sense l'ús de llibreries de tercers
  - Històricament, dels sistemes Windows, Linux i Mac, és el sistema amb un index de fallada més elevat

Figura 33: Logo de Windows

LINUX

- Avantatges
  - Gran optimització dels recursos, tant a nivell de software com de hardware
  - Disposem d'un gran ventall de software lliure per aquest sistema
  - Millor estabilitat, un clar exemple és el seu us en servidors d'alt rendiment
  - Entorn gràfic (beryl), millor que el aero de Windows
  - Existeixen distribucions de Linux per diversos tipus d'equips
  - Les vulnerabilitats són detectades i corregides més ràpidament que en qualsevol altre sistema operatiu

- Contres
  - Cal saber fer-lo anar, no es un entorn per principiants
  - La majoria dels ISP no dan suport per altres sistemes que no siguin Windows
  - No existeix molt de software comercial

Figura 34: Logo de Linux

Posteriorment a l'anàlisi de quin sistema operatiu s'ajustava més a les nostres necessitats vam escollir Windows, per comoditat, ja que era el sistema que ens imposava el fet de treballar al laboratori.

Després de començar la implementació del sistema van començar a sorgir problemes, dificultats per realitzar lectures correctes del joystick, impossibilitat d'establir comunicació amb la unitat de control Stäubli CS8C, el sistema es "penjava" quan s'executaven diverses tasques simultàniament, entre altres.

Finalment, vam optar per treballar sobre un entorn Linux. No es el més adient, ja que ens és impossible disposar d'un ordinador amb sistema operatiu Linux al laboratori, per tant, calia muntar una maquina virtual que executes Linux en un host Windows. Per sort, les maquines del laboratori ja disposen d'una maquina virtual amb la distribució Ubuntu instal·lada. Amb aquesta configuració no aprofitem el potencial d'aquest sistema operatiu, no obstant, ens va facilitar molt les tasques de desenvolupament.

En un futur caldria considerar disposar d'una maquina amb un sistema operatiu Linux, ja que aquest sistema operatiu facilita molt les tasques de desenvolupament i l'execució de software independent que no necessita llibreries de tercers ni interfícies gràfiques.

7.2.- Estudi i decisió dels dispositius

Com introduíem en apartats anteriors, el sistema consta d'una part física on intervenen diversos dispositius. La majoria dels dispositius del sistema (robot, unitat de control, sensor de força, etc) estaven imposat de manera implícita amb la proposta del projecte, facilitant d'aquesta manera l'estudi previ dels dispositius que conformen el sistema. A continuació exposem quins són els motius que fan viable la integració d'aquests elements al sistema:

- **Stäubli TX60**
  - Imposat amb la proposta de treball
  - El conjunt de moviments que pot realitzar el robot són suficients per realitzar les maniobres desitjades
  - Es pot ajustar la velocitat de maniobra en funció de les necessitats
  - Disposa d'una àrea de treball prou extensa per treballar de manera còmode

- **Stäubli CS8C**
  - Imposat amb la proposta de treball
  - El control del braç a traves d'una unitat de control fa necessari l'ús d'aquest dispositiu
  - Permet la comunicació per port serie i/o xarxa Ethernet, permetent la interacció amb aplicacions client

- **Schunk FTCL 50-80**
  - Imposat amb la proposta de treball
  - Lectures en un domini prou acurat per maniobrar de manera precisa
  - Està integrat amb el braç robòtic
  - Permet la interacció per port serie

- **Sensors òptics**
  - Connexió USB
  - Resolució suficient per permetre la maniobra remota
  - Facilitat d'usabilitat
    - Aquest dispositiu va ser aprovat per integrar-lo al sistema després de superar un estudi meticulós de funcionament i garantir que complia els requisits especificats

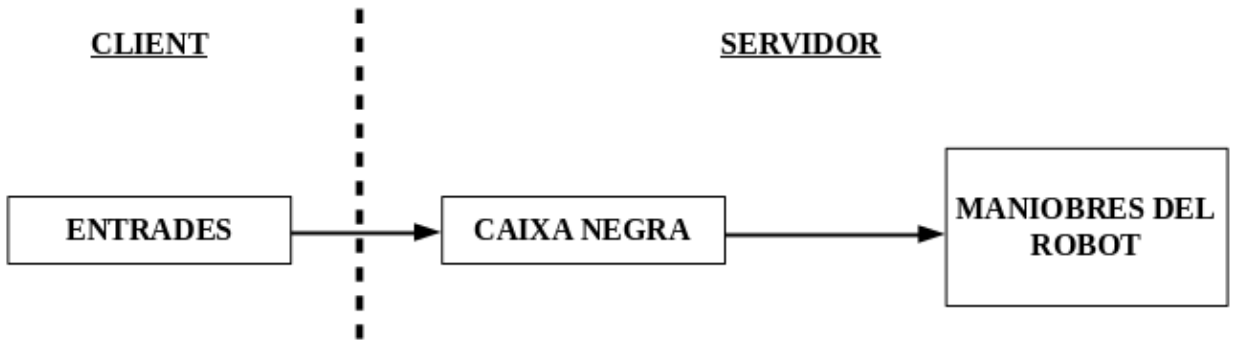
- **Joystick**
  - Connexió USB
  - Consta de 6 eixos i 12 botons
  - Resolució de lectura adequada per la finalitat del sistema
    - Aquest dispositiu va ser aprovat per integrar-lo al sistema després de superar un estudi meticulós de funcionament i garantir que complia els requisits especificats

7.3.- Aplicació client/servidor

Els sistemes industrials, més concretament els braços robòtics industrials, estan dissenyats per assolir posicions de tipus punt respecte un sistema de referencia, o assolir una posició a traves de la definició dels valors articulars. Aquest fet, fa que el disseny dels sistemes que permeten programar el robot estiguin orientats ha fer-ho des de una consola (MCP) o un ordinador (programa). La majoria de sistemes industrials, són configurats i programats per entorns de producció, repetint els moviments periòdicament sense la necessitat d'interacció humana. D'aquesta manera, obtenim un robot que es controlat per una "caixa negra" que executa moviments periòdics en funció de les entrades que rep.

El nostre sistema integra un subsistema industrial. Basant-nos en l'explicació anterior, cal adaptar-nos al fet que la part de control del robot, un cop programada, actuara com una "caixa negra", executant moviments en funció de les entrades rebudes. Aquesta situació ens força a dissenyar el sistema com un model client/servidor, on la part del servidor correspondrà al control del robot, i la part del client a les entrades que s'envien.

L'esquema que presentem tot seguit, mostra, de manera conceptual, el funcionament de la comunicació entre les parts del sistema client i servidor.



7.4.- Llenguatges de programació

Una decisió errònia en el moment d'escollir el llenguatge de programació del sistema, pot conduir a errors tals com: problemes de seguretat, ineficiència, resultats inesperats, etc. És vital encertar en l'elecció dels llenguatges de programació.

7.4.1.- Servidor

Stäubli ofereix una amplia gamma de dispositius; unitats de control, braços robòtics, sensors, etc. El fet d'oferir un ventall tan ampli de dispositius implica que cal posar a disposició dels desenvolupadors eines per poder programar aquests elements de manera intuïtiva i eficient. VAL3 és el llenguatge que unifica i defineix el mètode de programació en dispositius d'aquesta marca. Com a conseqüència directa de la integració d'elements Stäubli (unitat de control i braç robòtic) en el nostre sistema, desenvoluparem la part de servidor amb aquest llenguatge de programació. A continuació exposem algunes de les característiques del VAL3 (més informació a l'annex [C]):

- Joc de funcions robòtiques molt potent oferides per un llenguatge robòtic dedicat
- Reutilització i capitalització del coneixement gracies al seu enfocament flexible i modular
- Múltiples possibilitats de connexió: entrades/sortides digitals, comunicacions serie, bus de camp, etc
- Accés complet al robot i controladora a partir de la consola MCP

7.4.2.- Client

En apartats anteriors ja s'ha comentat de manera reiterada, que l'aplicació client ha de permetre la comunicació amb l'aplicació servidor amb la finalitat de teleoperar el robot en temps real. L'aplicació client ha de ser lleugera, ben definida i eficient en les tasques que desenvolupara al sistema. A l'apartat 6.1.2 s'ha introduït una idea inicial dels elements que conformen l'aplicació client i podem apreciar 3 tasques:

- **Joystick**
  Aquest modul està ideat per realitzar lectures de manera asíncrona del dispositiu joystick controlat per l'usuari. És important que tot moviment realitzat amb aquest dispositiu sigui processat de manera immediata i es notifiqui a la unitat de control CS8.

- **Control**
  El modul de control té com a objectiu processar les lectures del joystick i en funció dels paràmetres (mode de moviment, mode d'operació, velocitat, etc) definits per l'usuari, preparar la comanda adient per notificar a la unitat de control CS8 la voluntat de l'usuari.

- **Comunicació**
  La unitat de comunicació implementa el pont entre l'aplicació client i l'aplicació servidor, permetent la comunicació bidireccional entre ambdues parts de manera fiable i eficient.

Un cop aclarit amb més detall quines seran les tasques que ha de desenvolupar la part client cal escollir el llenguatge de programació més adient. Existeixen diversos llenguatges que ens permetrien realitzar aquestes tasques de manera fiable i eficient, com per exemple: Python, Java, C, C#, VisualBasic, etc. No obstant, hem escollit com llenguatge de programació per la part client el C++ pels següents motius:

- Compilat
- Fortament tipat
- Programació molt eficient
- Generació de codi objecte
- Administració directa de la memòria
- Estructurat combinat amb la programació orientada a objectes
- Popular per la creació de software de sistemes i per la creació d'aplicacions

En conclusió, l'elecció de C++ és deu a la capacitat de control que aquest llenguatge atorga al desenvolupador sobre el sistema.

7.5.- Stäubli Robotics Studi (SRS)

La implementació del codi VAL3 que s'executa a la part del servidor es desenvolupara amb l'eina proporcionada per Stäubli anomenada Stäubli Robotics Suite (SRS). Cal estudiar en profunditat el funcionament d'aquest software per realitzar una implementació correcta i eficient.

Aquest és un entorn de programació i simulació 3D de robots Stäubli. A més a més, té la capacitat d'emular els corresponents armaris de control. Aquesta simulació es realitza a través del software CS8 el qual està integrat dins el software Stäubli Robotics Suite 2013.

L'escassetat de material docent relacionat amb l'ús del software ha fet que la informació presentada en aquest document, sigui fruit d'una acurada investigació realitzada sobre el programari.

7.5.1.- Pestanyes d'eines

Per poder visualitzar les pestanyes d'eines cal clicar la icona superior esquerra la qual conté el dibuix d'un robot (figura 35).

Figura 35: Pestanya d'eines

La finestra resultant de clicar la icona (figura 36), conté totes les eines necessàries per poder simular una aplicació sencera.

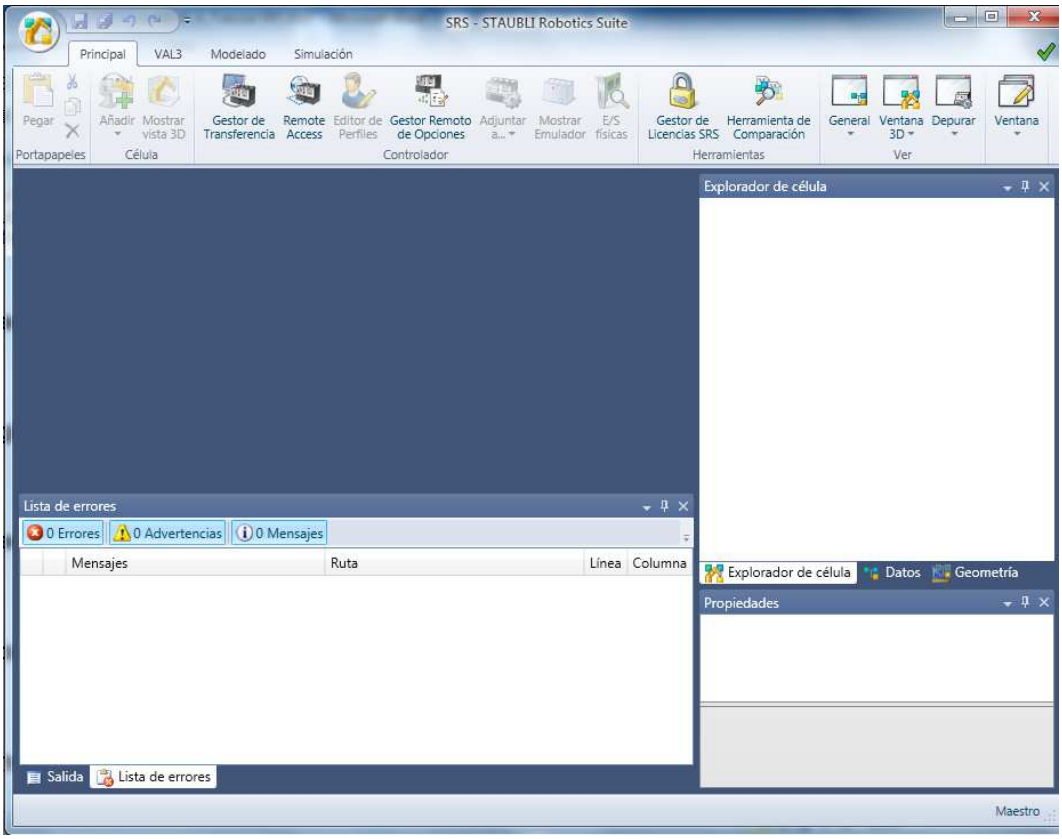


Figura 36: Entorn de treball SRS

- **Explorador de cèl·lula**
  És un explorador on s'organitzen tots els programes i biblioteques de cada una de les aplicacions obertes. Quan es crea una nova cèl·lula de treball, aquesta es mostra dins l'explorador de cèl·lula (figura 37).

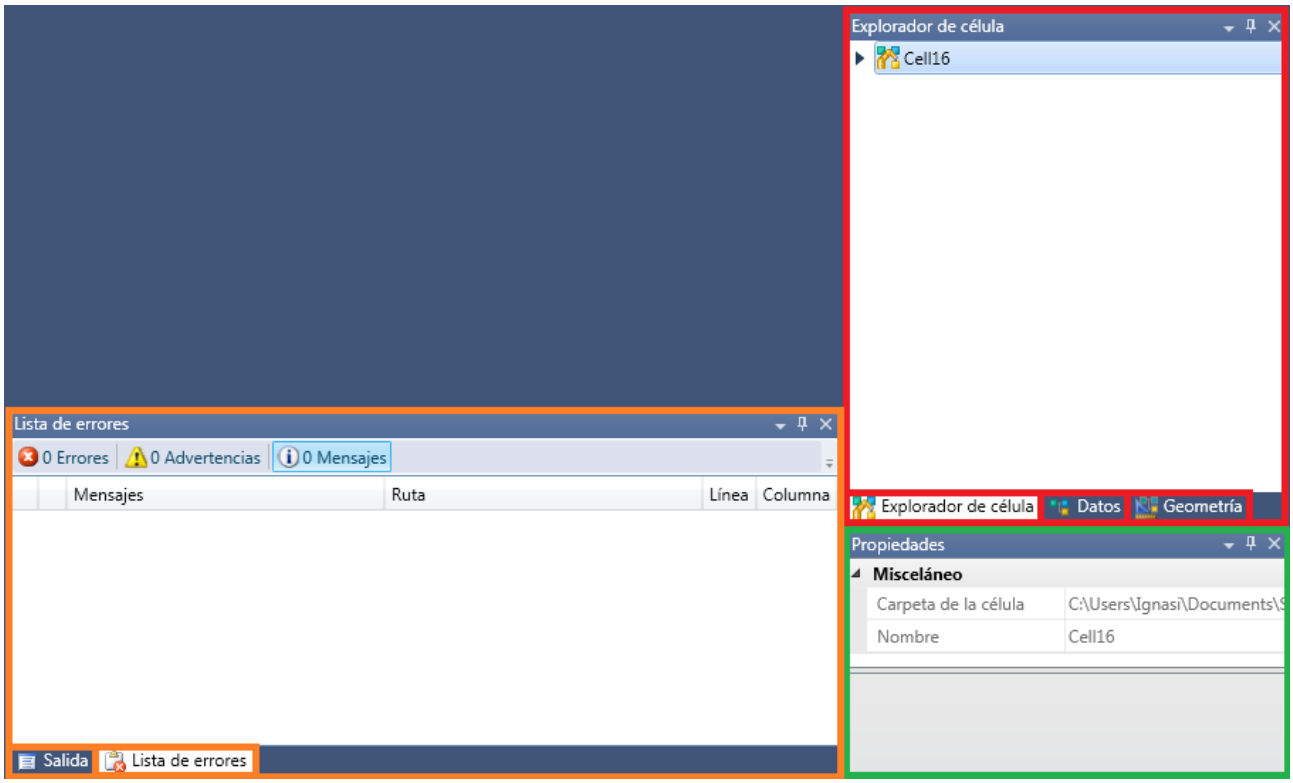


Figura 37: Explorador de cel·la

- **Dades**
  Aquesta pestanya conté totes les dades de cada aplicació. Aquests estan classificats segons el tipus de dada. Des d'aquesta pestanya es poden modificar els seus valors fent doble clic a sobre d'una d'elles. També s'hi poden crear noves dades.

- **Geometria**
  Aquesta pestanya conté tots els punts, marques i eines de la cèl·lula. És on podem trobar i editar les propietats geomètriques de totes les figures, eines, punts i sistemes de referència.

- **Propietats**
  Aquesta pestanya mostra les propietats de l'objecte que es té seleccionat.

- **Llista d'errors**
  Aquesta pestanya mostra una llista amb els missatges, errors i advertències que es generen durant la comprovació de la sintaxis, la depuració i l'execució d'una aplicació.

- **Sortida**
  Aquesta pestanya mostra els missatges emesos. Aquests estan organitzats en tres categories: general, vista 3D i emulador.

7.5.2.- Barra d'opcions

La barra d'opcions és una barra que està organitzada per defecte en 4 grups principals. Aquests són: Simulació, Modelat, VAL3 i Principal. (figura 38).

Figura 38: Barra d'opcions

En funció de la tasca específica que s'estigui realitzant en cada moment, pot ser que temporalment aparegui algun grup més.

Cada un d'aquests grups està subdividit en conjunts més petits. Aquests agrupen pestanyes que estan relacionades en funció de la seva utilitat. A la majoria de les pestanyes que apareixen en aquests grups, també s'hi pot accedir a través dels menús desplegables que apareixen al fer clic amb el botó de la dreta sobre de determinades icones.

Tot seguit anem a comentar, de manera general, cada un dels grups esmentats.

- **Principal**
  Aquest grup, tal com diu el seu nom, conté les eines principals. Aquestes permeten realitzar les accions necessàries per afegir nous robots a la cèl·lula i visualitzar-la en 3D. També gestionar el controlador i l'emulador així com totes les seves respectives connexions. A més a més, permet obrir el gestor de llicències. En cas que haguem tancat alguna de les finestres de treball de manera accidental, també ens permet desplegar les finestres que es desitgi. Finalment a través del menú "ventana", "Distribuciones de ventana", es permet gestionar el layout de pestanyes i guardar configuracions de layout específiques.

- **VAL3**
  Aquest grup permet crear i gestionar aplicacions VAL3. Conté les opcions de comprovació de la sintaxis, depuració i execució de les aplicacions, etc. Des d'aquest grup també es poden afegir llibreries.

- **Modelat**
  Aquest grup permet gestionar les geometries 3D. Es poden afegir geometries creades a través d'altres softwares CAD o generar geometries bàsiques. També permet definir les posicions d'aquestes geometries. En cas que es vulgui, el submenú "Herramientas" permet afegir una geometria com a eina, tot editant-ne les seves característiques.

- **Simulació**
  Aquest grup conté totes les pestanyes relacionades amb la simulació 3D. Des d'aquí es pot seleccionar el tipus de moviment al qual es vol sotmetre el robot i editar punts a través de la posició del robot. També permet generar un conjunt de punts per mostrar la trajectòria realitzada pel robot i fer la detecció de col·lisions. Finalment, conté el menú de sincronització que permet sincronitzar amb un mateix tren de polsos diversos controladors.

7.6.- Sistema de control òptic

Els sistemes de control en el sector de la teleoperació atorga'n a l'usuari la possibilitat de maniobrar més còmodament. Un dels sistemes més rellevants és l'òptic, ja que ofereix una aproximació a l'entorn de treball, que d'altre manera no podríem tenir per problemes de seguretat. El nostra sistema oferira aquesta aproximació, per una millor la maniobrabilitat, a traves de 2 càmeres òptiques, una fixada a l'element terminal del braç robot i l'altre fixada al banc de treball. D'aquesta manera, l'usuari obtindrà 2 perspectives diferents que l'ajudaran a realitzar les maniobres.

La marca i model de les càmeres serà Logitech C170 i aniran connectades a l'ordinador a traves de ports USB. Aquest dispositius no es controlen a traves de les aplicacions del nostre sistema, com a conseqüència, cal controlar-les externament al sistema, tot i que teòricament es podrien incloure en el marc de l'aplicació client.

7.6.2.- Aplicació ManyCam

Per facilitar encara més l'ajuda visual oferida a l'usuari, incorporarem aquest software que ens permet sobreposar la imatge de les 2 càmeres per veure-les al mateix temps. La figura 39 mostra un clar exemple del resultat tal com el veurà l'usuari.

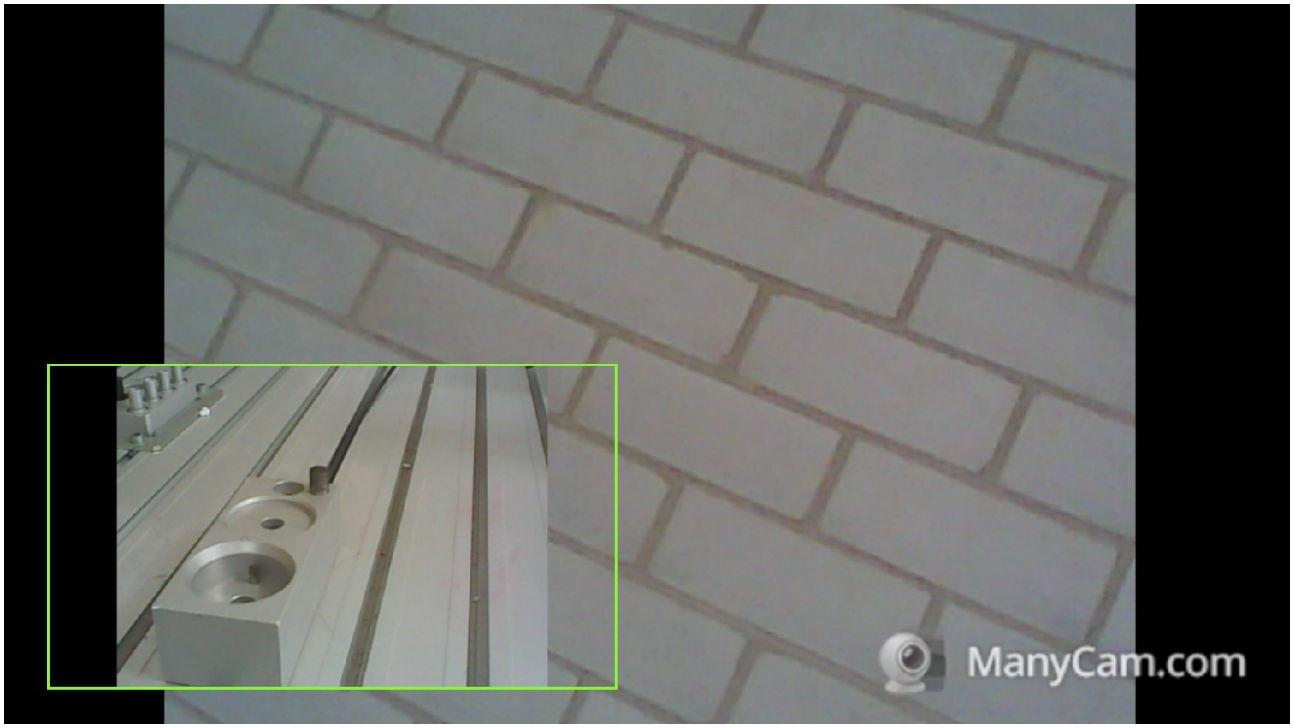


Figura 39: Vista del software ManyCam

8.- ANÀLISI I DISSENY DEL SISTEMA

La definició dels requeriments del sistema presentats amb anterioritat i l'estudi dels elements que intervenen ens permet realitzar un anàlisi en profunditat de quines seran les tasques que l'usuari desenvolupara i quina resposta obtindrem del sistema.

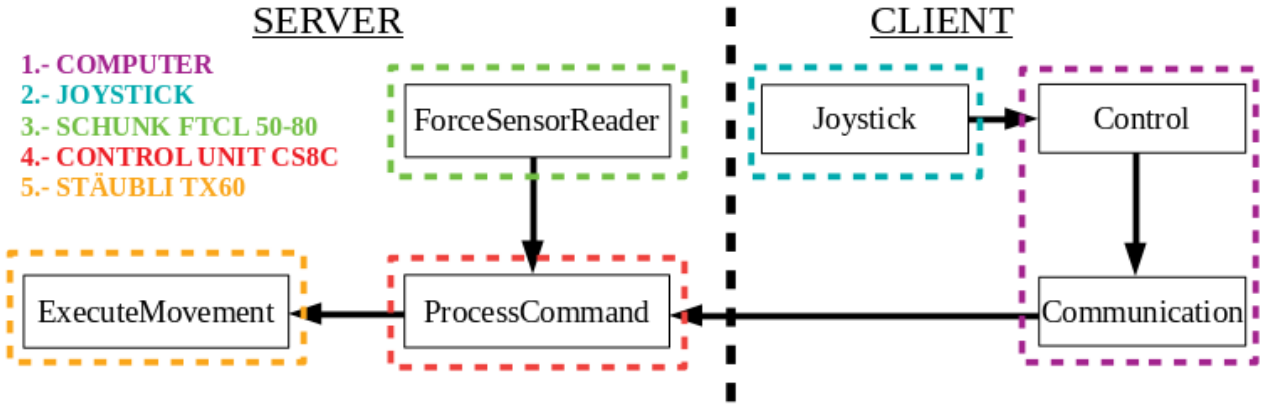
8.1.- Anàlisi del sistema

Al llarg del document hem esmentat en diverses ocasions que el sistema està format per una part física de dispositius i una altre part de programari. Dins la part de programari, distingim entre l'aplicació client i servidor. Aquesta clara diferenciació en el sistema també es veu reflectida en les etapes d'anàlisi i disseny.

Com ja exposàvem en l'apartat 7 del present treball, la majoria dels dispositius físics amb els que treballem se'ns han imposat, pel simple motiu que el sistema incorpora elements industrials complexos que ja han estat estudiats prèviament i compleixen les finalitats requerides. Per tant, l'apartat físic no requereix un anàlisi dels elements, simplement amb un estudi en profunditat de com està muntat i com funciona l'entorn del laboratori i un bon disseny dels protocols de comunicació, hauríem de tenir suficient pel sistema desitjat.

Pel contrari, el software si que requereix un anàlisi més exhaustiu per poder realitzar un bon disseny abans de la implementació. Aquest anàlisi s'aconsegueix, en la majoria dels casos, elaborant un diagrama de casos d'us.

L'anàlisi, el disseny i els requeriments d'un sistema són elements molt similars. El següent esquema, representa la diferenciació de les aplicacions client/servidor que conformen el sistema, on podem veure un disseny inicial a partir dels requeriments i l'anàlisi realitzat.



8.1.1.- Diagrama de casos d'ús

Un cas d'ús és una descripció dels passos o les activitats que deuen realitzar-se per dur a terme algun procés. Els personatges o entitats que participen en un cas d'us es denominen actors. En el context d'enginyeria del software, un cas d'us és una seqüencia d'interaccions que es desenvolupen entre un sistema i els seus actor en resposta a un esdeveniment que inicia un actor principal sobre el propi sistema. Els diagrames de casos d'us serveixen per especificar la comunicació i el comportament d'un sistema mitjançant la interacció amb l'usuari.

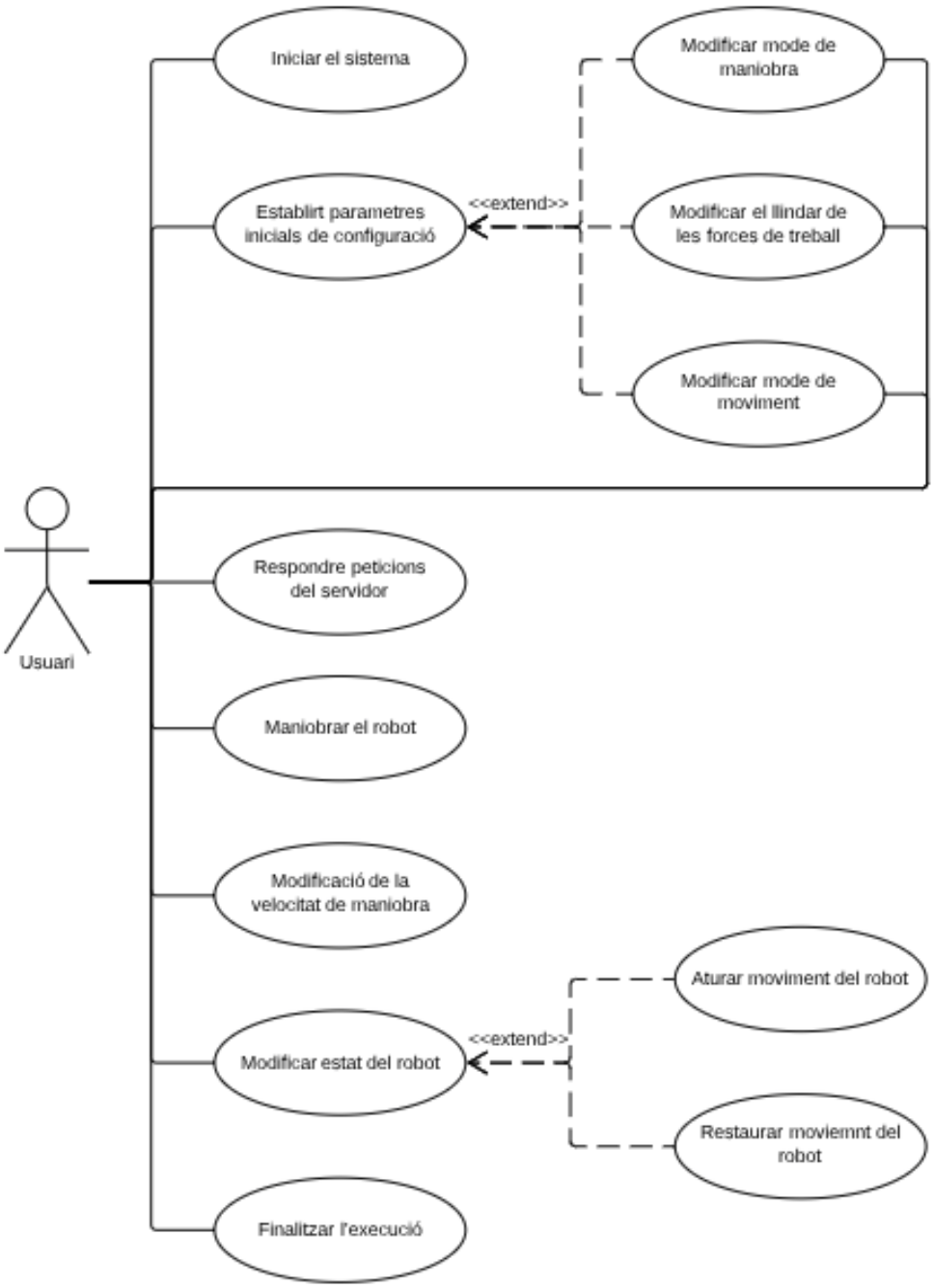


Figura 40: Diagrama de casos d'ús

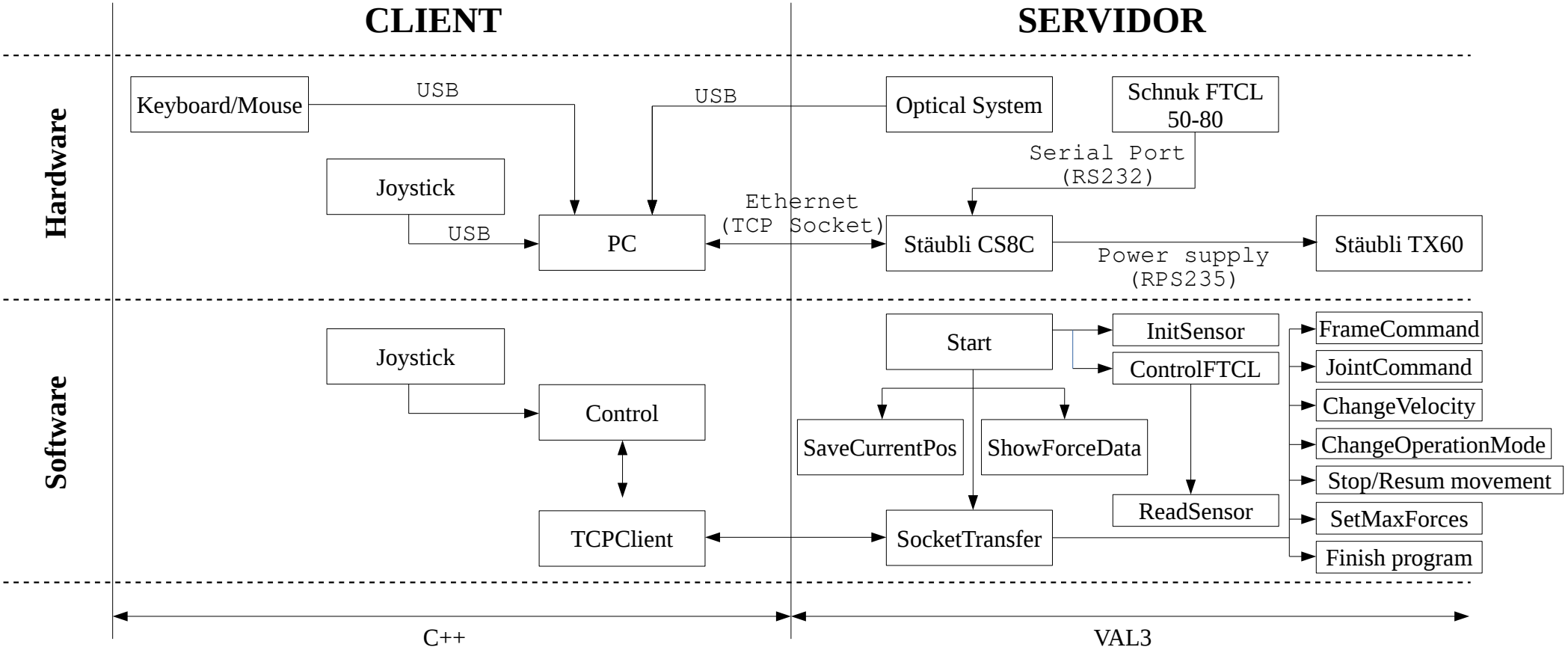
El desenvolupament del diagrama de casos d'ús ens mostra quines són les tasques que l'usuari podrà dur a terme un cop el sistema estigui implementat:

- Iniciar/Finalitzar el sistema
- Modificar el mode de maniobra
- Modificar el llindar de forces
- Modificar el mode de moviment
- Respondre peticions enviades pel servidor (CS8C)
- Maniobrar el robot
- Modificar la velocitat de maniobra

8.2.- Disseny del sistema

L'anàlisi del sistema ens brinda una primera idea de com cal dissenyar-lo, ja que ens mostra les necessitats, i de manera implícita, alguns elements estructurals.

Aquest apartat funcionara de la següent manera, tot seguit mostrarem el diagrama obtingut de la fase de disseny, on apreciarem: elements físics, canals de comunicació, mètodes de cada aplicació, etc. Un cop entès a grans trets l'esquema, desgranarem el diagrama en apartats més petits on exposarem detalladament els motius del disseny.



8.2.1.- Disseny estructural del sistema

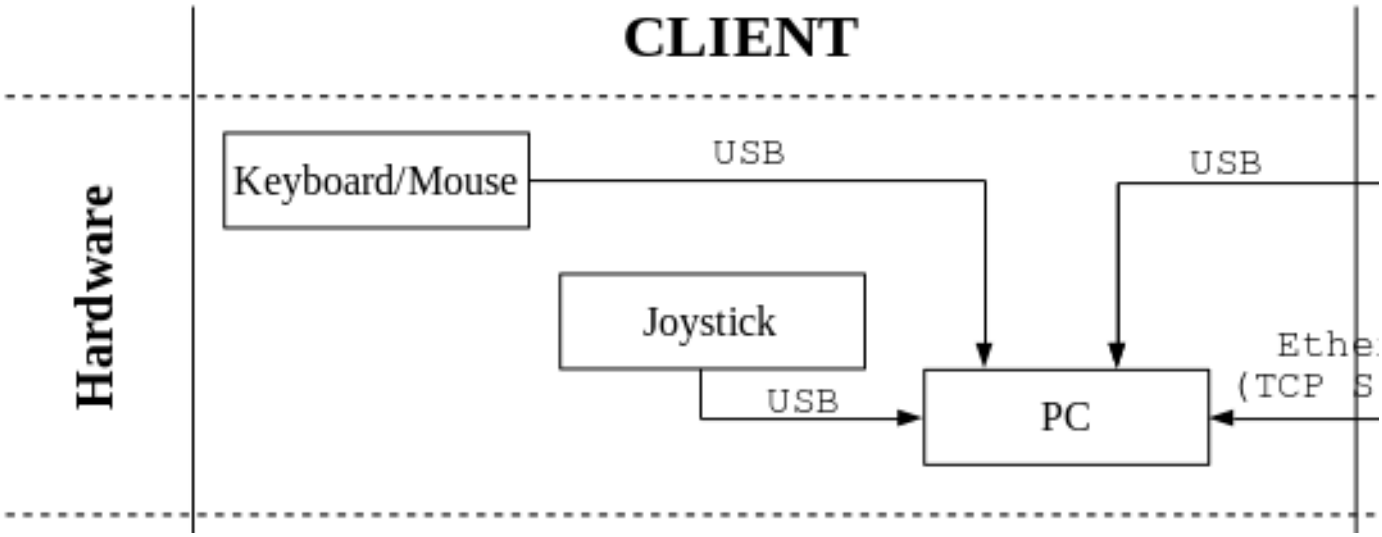
El diagrama ens permet apreciar que el sistema està estructurat en 2 blocs, el de control (client) i l'actuador (servidor).

- **CLIENT**
  La part del client permet a l'usuari tenir control sobre el sistema. Està format per un ordinador que actua com a unitat superior de control en tot el sistema que es comunicara amb la unitat de control CS8C (part servidor). L'entrada de dades al sistema de control es realitzara a traves de tres dispositius:

  - Teclat/Ratolí: Únicament s'interactuara amb aquests dispositius en el moment d'iniciar el sistema, per modificar paràmetres de configuració o per respondre peticions del servidor.

  - Joystick: Aquest dispositiu serà el millor company de l'usuari, ja que la major part del temps que es maniobri el sistema es farà a traves d'aquest element. El joystick es comunica amb l'ordinador, aquest realitza un procés de control en funció de la configuració dels paràmetres del sistema i envia les dades cap al servidor, que actua en funció de la comanda rebuda i la configuració actual dels paràmetres del servidor.

  - Càmeres òptiques: Tot i que aquests elements no necessiten programació, pròpiament dit, conformen un punt important en el sistema de control (control passiu). La interacció de l'usuari amb aquests elements és mínima. Prèviament a l'inici de l'aplicació de control del sistema, cal deixar configurades les càmeres per poder gaudir d'una millor visió de l'entorn de treball (annex [A] per més informació).

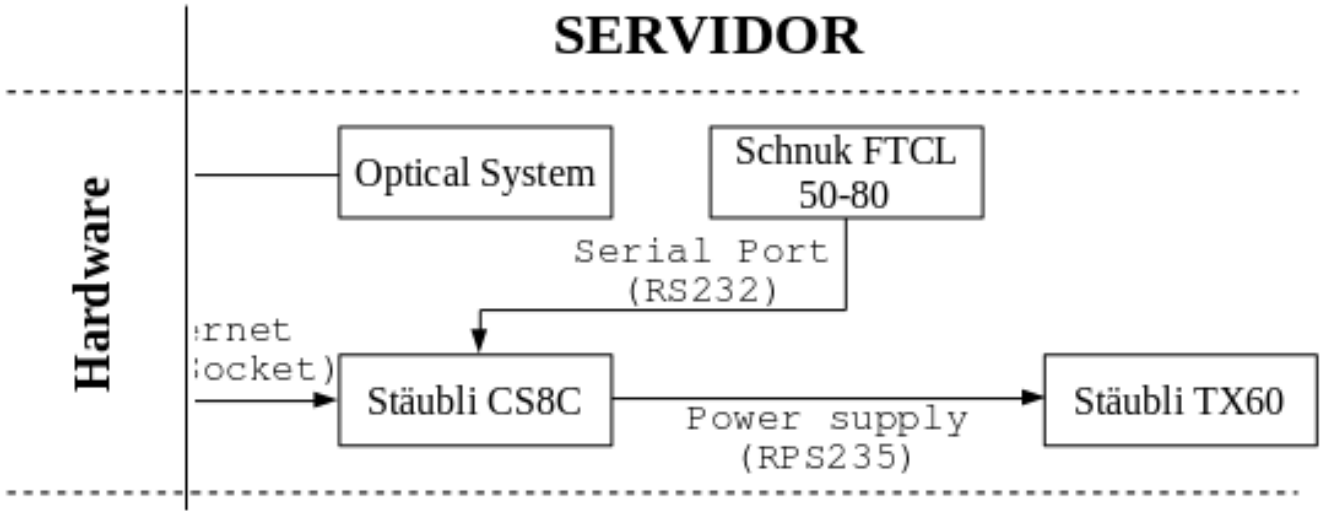


- **SERVIDOR**
  El bloc servidor, tot i tenir alguns elements de control, conforma en la seva major part, el sistema actuador. Els elements actuadors que conte aquest bloc són els següents:

  - Stäubli CS8C: La unitat de control del robot, rep aquest nom, precisament perquè és l'encarregada de controlar les accions del braç robòtic. No obstant, també s'encarrega d'enviar els senyals pertinents perquè aquest es mogui, realitzant la funció d'actuador. La CS8C es comunicara amb l'ordinador (part client) per tal de dur a terme la voluntat de l'usuari. Finalment, aquest element realitzara lectures constants del sensor de força i actua en funció de les dades rebudes i els paràmetres configurats per l'usuari.

  - Schunk FTCL 50-80: El sensor de força interactua amb el sistema de manera molt senzilla, al rebre pressió en algun dels seus eixos o captar moments en funció de la maniobra del robot, facilita les dades llegides a la unitat de control quan aquesta les sol·licita. Cal remarcar que les lectures s'han de dur a terme en espais de temps molt breus, amb la finalitat de no perdre informació que pugui conduir a accions no desitjades per l'usuari.

  - Stäubli TX60: La pinça acoblada al TX60 conforma l'element terminal del braç, de manera anàloga, el braç robòtic conforma l'element terminal del nostre sistema, essent l'objecte que ens permet interactuar amb l'entorn real. El braç en si mateix no te cap element de control, tret de les limitacions físiques. Com a conseqüència de no tenir sistema de control integrat, podem dir que és l'únic element actuador pur del nostre sistema.



8.2.2.- Disseny de les classes/scripts/programes

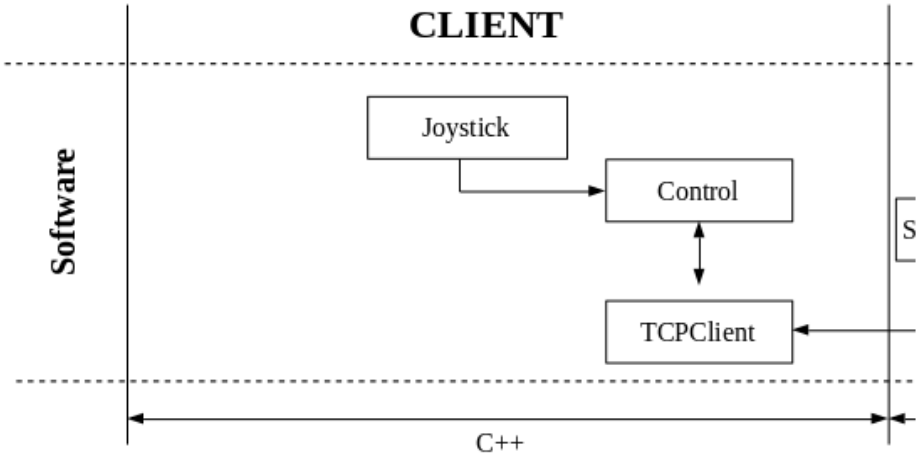
El següent pas al disseny de l'estructura és definir i dissenyar quines seran les classes, scripts i programes que haurem de desenvolupar en C++ i VAL3 pel nostre sistema.

- **CLIENT (C++)**
  Com s'ha introduït al llarg de tot el document en diversos apartats, l'aplicació client actua com a sistema de control permetent a l'usuari maniobrar el sistema. Un dels requeriments més rellevants és que el sistema s'ha de poder controlar en temps real per poder simular maniobres d'àmbit medic. Per fer-ho, hem dissenyat la part del client en les següents classes:

  - Joystick: La lectura del joystick és la prioritat principal. Cal que en qualsevol moment de l'execució del sistema, si es realitza una maniobra amb el joystick, aquesta sigui notificada immediatament a la classe de *Control* (comunicació unidireccional). Per aconseguir aquest propòsit, caldrà executar el codi corresponent a la lectura del dispositiu, en un fil d'execució separat.

  - Control: El sistema requereix un conjunt de paràmetres de configuració pel correcte funcionament. Aquesta classe és el pilar central del sistema de control, realitzant les següents tasques:
    - Modificar el valor de les variables de control del sistema
    - Rebre les dades enviades pel *Joystick*, processar-les en funció dels paràmetres de configuració, preparar la comanda adient , si escau, la comanda pertinent a la unitat de control del servidor (CS8C)
    - Esperar peticions del sistema de control del servidor (CS8C). Poden existir casos, on el servidor tingui la necessitat de consultar a l'usuari si desitja o no continuar amb la maniobra que està realitzant.
      - Aquesta tercera tasca, es du a terme de manera independent (fil d'execució separat) a la resta de funcions de la classe *Control*, ja que no sabem quin serà el factor que desencadenara la necessitat de consultar a l'usuari.

  - TCPClient: La implementació d'aquesta model, ens crea un pont de comunicació entre les 2 grans aplicacions del sistema, oferint una transmissió de dades bidireccional. La classe *TCPClient* implementara els mètodes necessaris per una comunicació via socket TCP entre la classe *Control* del client i el programa corresponent a la unitat de control CS8C del servidor.



- **SERVIDOR (VAL3)**
  El servidor representa la part d'actuació del sistema, tot incorporant alguns elements de control. La part de software del servidor està desenvolupada en VAL3 ja que l'execució té lloc a la unitat de control CS8C. Tot i que l'aplicació client conte una part important del software del sistema, podem assegurar que el gruix del codi desenvolupat es troba en aquesta secció.

  VAL3 té una manera molt peculiar de treballar (annex [C] per més informació). Només ens cal saber que s'estructura de la següent manera:
  - El sistema conte aplicacions
  - Les aplicacions contenen programes (com a mínim **Start** i **Stop**)
  - El primer programa que s'executa a l'inici del sistema es **Start**
  - El darrer programa que s'executa al finalitzar el sistema es **Stop**
  - La comanda **call nomPrograma()** permet la interacció entre programes
  - Es poden crear tasques sincrones i asincrones
  - Les variables són comunes per tota l'aplicació, però no entre aplicacions si es defineixen com privades

  Amb aquests conceptes clars hem definit l'aplicació del sistema, anomenada **"staubli-surgeon"**, i formada pels següents programes:
  - Start: Aquest programa és el punt de partida en el moment d'engegar el sistema. Es el programa principal, a partir del qual es crida a la resta de codi de la controladora CS8C. Realitzara tasques com inicialitzar el sensor de força, mostrar les dades necessàries per la MCP, crear una tasca asincrona que resti a l'espera de rebre ordres del client, entre altres.

  - InitSensor: A l'inici de l'aplicació cal realitzar un calibratge inicial del sensor de força. Si no es realitza aquesta calibratge, el sensor pot donar lectures falses i/o errònies impedint que el sistema treballi com ho hauria de fer.

  - SaveCurrentPos: És molt probable que en algun moment de la maniobra excedim la força màxima estipulada. Per aquest motiu, cada 1.5 segons guardem la posició actual del robot per si fos necessari assolir una posició anterior. Cal remarcar que aquest programa s'executa de manera asíncrona al sistema per permetre l'execució constant d'altres tasques de major prioritat.

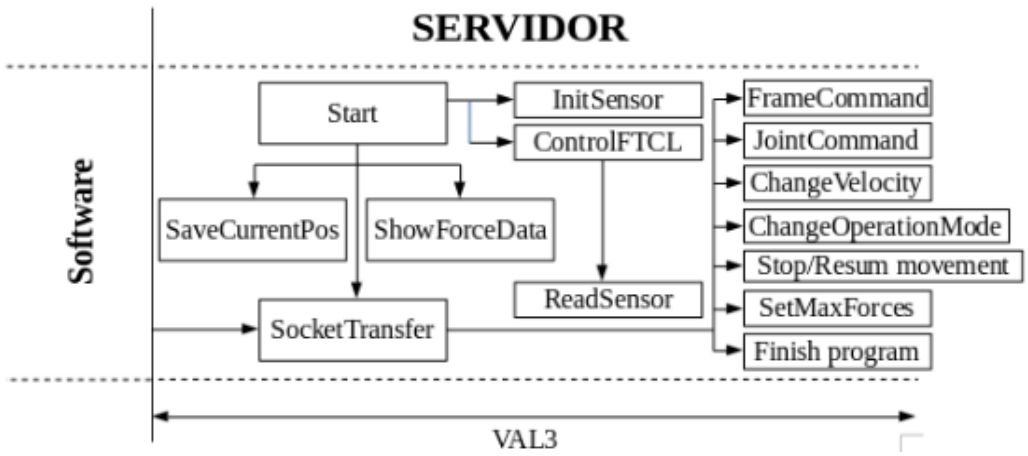
  - SocketTransfer: Aquest programa és la part anàloga al *TCPClient* de l'aplicació client. Actua com a element intermediari entre la part client/servidor. La seva única funció és romandrà a l'espera de que l'usuari enviï una comanda i executar-la en funció de la configuració actual. Aquest programa ha d'estar constantment esperant rebre comandes sense que cap altre bloc de codi impedeixi la seva execució, per tant, aquest programa també s'executa de manera asíncrona a la resta de funcions del servidor. A continuació presentem els programes que executa *SocketTransfer* en rebre una comanda del client:
    - FrameCommand: És una comanda de moviment. Com ja introduíem en l'apartat de conceptes previs, els braços robòtics es poden moure en funció de sistemes de referencia. El mode de moviment f*rame* manipula el robot basant-se en la relació que s'estableix entre els sistemes de referencia de l'element terminal (frame) i la base del robot (world).

- JointCommand: Correspon al segon mode de moviment. El TX60 es conforma de 6 graus de llibertat angulars. Com esmentàvem a l'apartat 5.3.1, el robot disposa d'un mecanisme per conèixer en tot moment la posició del robot, d'aquesta manera podem indicar al robot que assoleixi una determinada posició en funció de les configuracions individuals de cada joint.
- ChangeVelocity: Cal remarcar que les maniobres que es realitzaran s'efectuaran a velocitats molt reduïdes donat l'entorn que volem simular. No es recomanable establir una velocitat de maniobra permanent, ja que depenent de l'operació, les característiques del pacient i demes factors, pot requerir-se maniobrar a diferents velocitats. Aquesta comanda permet modificar la velocitat a la que es mou l'element terminal entre el 0,5% i el 12% de la velocitat màxima del robot.
- ChangeOperationMode: El sistema ha de permetre a l'usuari simular operacions mediques i/o quirúrgiques. Aquesta funció alterna entre les diverses opcions que presenta el sistema.
- Stop/Resume movement: Mecanisme que atorga al sistema la possibilitat de pausar / restaurar el moviment del robot en mig d'una trajectòria.
- SetMaxForces: Estableix les forces màximes que es poden aplicar en cadascun dels eixos X, Y i Z.

- ControlFTCL: Aquest programa té 2 objectius; el primer, és realitzar lectures del sensor de força cada 4 mil·lisegons i emmagatzemar les dades, el segon és realitzar l'acció pertinent, només en cas d'excedir la força definida, en funció del mode de maniobra en el que ens trobem.
  - ReadSensor: Programa que es cridat des de *ControlFTCL* i realitza la lectura i el processament de les dades que rebem del sensor per fer-les intel·ligibles, escalant i realitzant les conversions necessàries.

- ShowForceData: Mostra les forces i moments del sensor de força a traves de la pantalla de la MCP. Com les variables són comunes per tota l'aplicació, i les lectures es realitzen a gran velocitat, es inviables mostrar les dades per pantalla cada 4 mil·lisegons, ja que no seriem capaços d'apreciar-les, per aquest motiu aquest programa s'executa com una tasca asíncrona.

- Stop: Es el darrer programa que s'executa abans de finalitzar l'aplicació. S'encarrega d'eliminar totes les tasques que s'han creat, de restablir la posició inicial del braç i mostrar un missatge per pantalla indicant la parada del sistema.



8.2.3.- Disseny del protocol de comunicació

En informàtica i telecomunicacions, un protocol de comunicacions és un sistema de regles que permeten que dos o més entitats d'un sistema de comunicació es comuniquin entre ells per transmetre informació a traves de qualsevol tipus de variació d'una magnitud física. Es tracta de les regles o l'estendard que defineix la sintaxis, semàntica i sincronització de la comunicació. Els protocols poden ser implementats per hardware o per software, o per una combinació d'ambdós.

El sistema que implementem consta de 2 entitats que requereixen un mode de comunicació. És necessari el disseny d'un protocol ben definit per obtenir un sistema eficient i fiable. Abans de dissenyar el protocol de comunicació cal entendre com funciona el canal de comunicació i la codificació/descodificació que es fa en ambdós extrems.

- La classe *TCPClient* implementa un socket en C++, que funciona convertint les dades, independentment del tipus, en bytes i enviant-les o revent-les a traves del canal configurat prèviament (adreça, velocitat, paritat, *checksum*, etc).
- L'extrem del servidor funciona a traves de variables tipus SIO que cal configurar (consultar annex [C] per més informació). A grans trets, el funcionament és:
  - Enllaçar una variable del sistema amb una entrada/sortida serie del sistema. L'efecte de l'enllaç romandrà actiu fins la finalització de l'execució actual, permetent l'enviament i la rebuda de dades.
  - Les lectures del port ens retornen valors numèrics corresponents a caràcters de la taula ASCII, de tal manera que si hem rebut un 'F;1;', els valor que llegirem seran:70, 59, 49, 59. És fa necessari realitzar una conversió de les dades rebudes per tal de processar-les i realitzar les accions pertinents.
  - L'enviament de dades funciona de la mateixa manera, només podem enviar valors numèrics que representin valors de la taula ASCII. Si volguéssim enviar 'A;' hauríem d'enviar els valors: 65, 59.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Figura 41: Taula ASCII

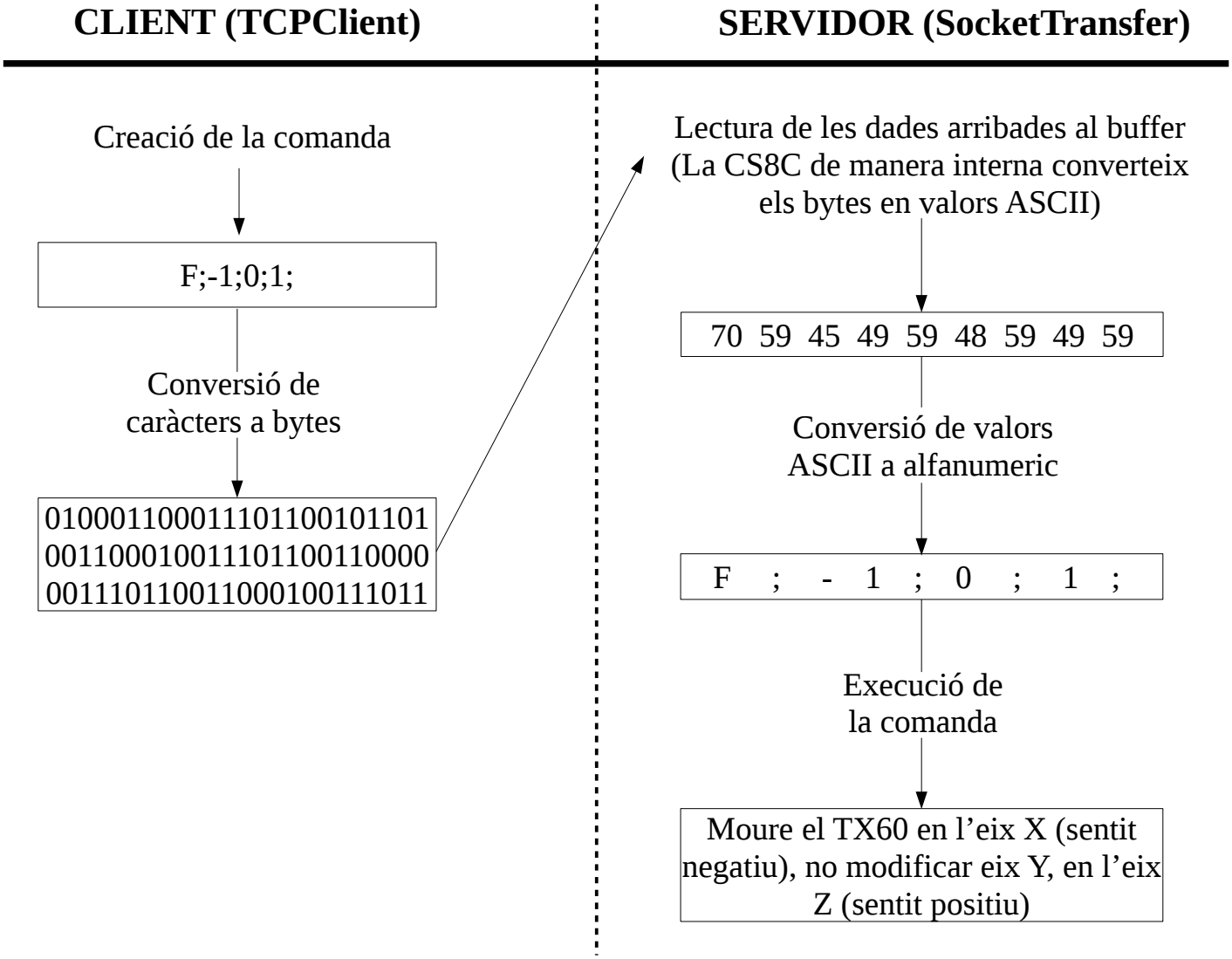
Despres de l'exploració del funcionament dels dos extrems de comunicació, podem dissenyar el protocol de comunicaió que posteriorment implementarem.

El protocol de comunicació consistira en l'intercanvi de comandes que tindran el següent format:

Command	Separator	Data	Separator	Data	Separator	Data	End
X	;	a	;	b	;	n	;

- Valor alfabetic indicant el tipus de commanda
- Separador de dades
- Valors alfanumeriques corresponent a la dada

El seguent esquema mostra l'interacció dels dos extrems quan el client envia una comanda i el servidor la processa. En cas de que el servidor envies la comanda i el client la reves, el procediment seria invertit.



9.- IMPLEMENTACIÓ, PROVES I RESULTATS

Aquest apartat del present document correspon a l'explicació detallada del procediment que s'ha realitzar al llarg del desenvolupament del sistema. Per tal d'estructurar-lo seguirem la planificació de tasques, iteracions i etapes definides al quart apartat d'aquest document.

Per cadascuna de les parts del sistema, exposarem els algorismes i els mètodes emprats, els problemes que ens hem trobat i les solucions aplicades i finalment les proves realitzades i els resultats obtinguts.

Cal remarcar que la major part d'aquest apartat no tractara els aspectes físics del sistema, ja que en apartats anteriors s'han realitzat explicacions prou detallades per entendre la part hardware del sistema. A l'annex [D] podem trobar tot el codi font de l'aplicació.

9.1.- Sistema de moviment

El sistema de moviment, tot i semblar senzill, probablement sigui amb molta diferencia, un dels objectius més importants que calia assolir en aquest treball, ja que representa el fonament de tot el sistema. Sense sistema de moviment, la resta de components del treball no tenen significat. Per aquest motiu s'ha invertit una gran quantitat de temps en desenvolupar un sistema de moviment molt acurat i amb un temps de resposta "immediat".

9.1.1.- Algorismes i mètodes

A continuació es presentaran tots els elements de software que intervenen en el sistema de moviment del Stäubli TX60. Realitzarem un recorregut des que l'usuari inicia una acció, fins que aquesta es manifesta al sistema.

9.1.1.1.- Joystick (C++)

- **string identifyDeviceName ()**
  El dispositiu Josystik es connecta al PC mitjançant un port USB. Depenent d'altres factors externs al sistema, és possible que el nom que representa al dispositiu físic variï. Aquest mètode realitza un anàlisi del fitxer ***/proc/bus/input/devices*** i en funció de l'id de venedor (**046d**) i l'id del producte (**c215**), selecciona el nom del dispositiu corresponent. És important que només hi hagi un joystick d'aquest tipus connectat al PC, d'altre manera el software no podrà iniciar el sistema ja que no el detectara. El nom del dispositiu te el següent format: */dev/input/jsx*, on X representa un valor numèric.

- **void selectMovementMode ()**
  El disseny del sistema indica que els modes de moviment són 2: *joint* i *frame*. Aquesta funció mostra un menú a l'inici de l'aplicació que permet a l'usuari escollir el mode de moviment. En cas de seleccionar el mode *joint* mostra un segon menú on cal escollir quina de les 6 articulacions desitgem maniobrar.

- **void changeJoint (int button)**
  El mètode mostra un menú per pantalla que permet a l'usuari seleccionar una de les 6 articulacions del TX60 per maniobrar-la. L'execució d'aquesta funció i la selecció del joint pertinent es realitzen a traves dels controls del joystick. La variable **button** correspon al boto del joystick que s'ha premut.

- **void changeMode (int button)**
  Mostra un menú per pantalla que permet a l'usuari alternar entre els modes de moviment *joint* i *frame*. Cal remarcar que un cop iniciat el sistema aquesta funció es realitza a traves dels controls del joystick. La variable **button** correspon al boto del joystick que s'ha premut.

- **bool getFinishState ()**
  Retorna l'estat d'una variable booleana que indica quan l'usuari desitja finalitzar l'execució del sistema.

- **void readJoystick ()**
  Aquest mètode és el nucli de la classe, ja que la major part dels processos es controlen aquí. El primer que cal saber es com s'estructuren aquests events d'aquest tipus en un sistema Linux. La següent imatge (figura 42) ens mostra el model que s'utilitza:

```
// STRUCTS
struct js_event {
    unsigned int time;      // event timestamp in milliseconds
    short value;            // value
    unsigned char type;     // event type
    unsigned char number;   // axis/button number
};
```

Figura 42: Model de dades de joystick a Linux

D'aquestes 4 característiques només ens interessen 3, ja que la marca de temps no la fem servir. El valor ***type*** ens indica si l'event és de tipus boto (JS_EVENT_BUTTON) o eix (JS_EVENT_AXIS), ***number*** ens informa de quin boto o eix està actiu, mentre que ***value*** correspon al valor llegit. El domini dels 12 botons que te el joystick és [0, 1], premut o deixat anar i el domini dels 6 eixos és [-32767..32767].

La definició de la interacció amb els elements del joystick (botons/eixos) la podem trobar a l'apartat 5.6.1 del present document.

9.1.1.2.- Control (C++)

- **void setMovementMode (int mode, int joint)**
  Guarda el valor del **mode** i **joint** en atributs de la classe. El **mode** correspon a joint (0) o frame (1) i el **joint** és un valor numèric compres entre 1 - 6 que correspon a un joint del braç.

- **void finishInitMode ()**
  Aquest mètode és simplement per controlar l'inici de l'aplicació. Estableix el valor de la variable booleana **init** a cert. Aquest control és necessari degut a que en el moment d'iniciar el sistema cal que l'usuari configuri uns paràmetres bàsics (mode de moviment, forces, etc) i fins que aquests no estan configurats no es poden executar altres parts del sistema.

- **void suspendResume ()**
  El mètode s'encarrega de notificar a la unitat de control CS8C que l'usuari vol parar la trajectòria. En cas que la trajectòria ja estigui aturada, indica que vol reprendre el moviment.

  Aquesta funció només s'executara si no ens trobem en algun estat que bloquegi el canal de comunicació, com per exemple seria estar en forma *selecció mode d'operació* o mode *establiment de forces*.

  La comanda que s'envia al servidor és: **S;**

- **void notifyExit ()**
  Quan l'usuari vol finalitzar l'execució del sistema, aquest mètode notifica al servidor que cal finalitzar els programes en execució. Aquesta funció només s'executa si no ens trobem en estat de suspensió.

  La comanda que s'envia al servidor és: **E;**

- **void movement (int mode, int joint, int * axis)**
  A la classe joystick, el mètode *readJoystick()* era el principal, de la mateixa manera, aquest mètode és el principal d'aquesta classe.

  Podem diferenciar entre 2 operacions principals controlades per aquesta funció:
  - Canvis de velocitat
    - La variable **axis** és un punter a un array de valors numèrics on l'index de cada posició correspon a un eix del joystick i el contingut és un valor dins del domini dels eixos. Processem les dades de l'eix pertinent al control de velocitat i construïm la comanda per notificar el canvi de velocitat. Si la comanda és diferent a la darrera enviada al servidor, es notifica el canvi de velocitat.

      La comanda que s'envia al servidor és: **V;x;**
      On **x** pot prendre els valors: 12.0, 10.0, 7.0, 5.0, 2.5, 2.0, 1.5, 1.0, 0.5

  - Comandes de moviment en mode JOINT/FRAME
    - En funció de la variable **mode** i **joint** construïm la comanda de moviment pertinent. El **mode** correspon a joint (0) o frame (1) i el **joint** és un valor numèric compres entre 1 - 6 que correspon a un joint del braç. Un cop construïda la comanda, s'envia al servidor, nomes si és diferent a la darrera que s'ha enviat.

      La comanda que s'envia al servidor pel mode joint és: J**;n;x;**
      On **n** pren un valor del domini [1..6] en funció del joint que volem maniobrar
      On **x** pren un valor del domini [-1, 0, 1] en funció del sentit en que volem moure el joint (-1 o 1) o si volem aturar el moviment (0)

      La comanda que s'envia al servidor pel mode frame és: **F;x;y;z;**
      On **x**, **y** i **z** prenen un valor del domini [-1, 0, 1] en funció del sentit en que volem moure l'eix (-1 o 1) o si volem aturar el moviment en aquell eix (0)

9.1.1.3.- TCPClient (C++)

- **bool sendData (string data)**
  Funció molt dedicada que només té com a finalitat enviar els valors de la variable **data** convertits en bytes pel socket configurat prèviament. Retorna un valor booleà en funció de si s'ha pogut enviar la informació exitosament.

9.1.1.4.- Staubli-Surgeon (VAL3)

- **socketTransfer ()**
  El programa executa un bucle infinit esperant rebre comandes pel socket, que han estat enviades des del client. Quan rep una comanda, independentment del tipus, la processa com s'ha exposat a l'apartat 8.2.3 d'aquest document. Un cop la comanda està convertida a mode text, analitzem el valor de la primera posició, que sempre correspondrà a un valor alfabètic en majúscula que ens indicara de quin tipus de comanda es tracta. En funció del tipus de comanda, executarem un altre programa que rebrà com paràmetre els valors de la comanda inicial.

- **stop ()**
  Aquesta funció s'executa com darrera instancia abans de finalitzar l'execució del sistema. Les tasques que realitza abans de finalitzar són:
  - Parar la trajectòria en execució
  - Aturar i esborrar totes les tasques creades
  - Moure el braç a la posició inicial
  - Deshabilitar la potencia del braç
  - Mostrar un missatge de final d'execució
  - Netejar la pantalla de la MCP

  Aquest programa pot executar-se per dos motius:
  - L'usuari ha indicat que vol finalitzar l'execució del sistema
  - En algun moment de l'execució del sistema, aquest ha fallat o ha estat necessari realitzar una parada d'emergència i com a conseqüència s'executa el programa stop()

- **startStop ()**
  Inverteix el valor de la variable booleana que controla l'aturada o la restauració del moviment del braç robòtic.

- **changeVelocity (command)**
  El paràmetre **command** correspon a una cadena de caràcters que representa un valor numèric. Realitzem una conversió de cadena de caràcters a valor numèric i l'assignem a la variable pertinent per establir la velocitat al valor desitjat. Tot aquest procés s'ha de realitzar aturant el moviment del robot i restaurant-lo al finalitzar.

- **jointCommand (command)**
  El paràmetre **command** conté el valor del joint que volem moure/parar i en quin sentit ho volem fer, dreta/esquerra. Com sempre, la variable es troba representada com una cadena de caràcters, cal realitzar una conversió a valor numèric.

  Un cop obtenim el joint i el sentit de moviment, cal calcular el valor màxim que pot assolir aquell joint en aquell sentit. Aquest calcul ens retorna un valor numèric que ens indica la posició que el joint ha d'adquirir.

  Finalment, si el sentit de moviment és diferent de 0 (aturar moviment en joint), executem la comanda de moviment per tal de que el braç es comenci a moure cap a la posició desitjada.

  Aquesta comanda només s'executara si el sistema no es troba en mode suspensió.

- **frameCommand (command)**
  El paràmetre **command** conté el valor de l'eix que volem moure/parar i en quin sentit ho volem fer. Com sempre, la variable es troba representada com una cadena de caràcters, cal realitzar una conversió a valor numèric.

  Un cop obtenim l'eix i el sentit de moviment, comprovem que no vulguem aturar el moviment en un o més eixos. Si es dones el cas, caldria aturar completament el moviment del robot.

  A continuació calculem el punt màxim dins el rang de treball del robot que pot assolir l'element terminal en cada eix (X, Y i Z) considerant el sentit i si cal o no moure aquell eix.

  Finalment, executem la comanda de moviment perquè el robot es desplaci a la nova posició.

  Aquesta comanda només s'executara si el sistema no es troba en mode suspensió.

9.1.2.- Problemes i solucions

Com ja s'ha mencionat, el sistema de moviment conformava la base del sistema. Sense teleoperació del braç en temps real, ni podíem avançar ni teníem sistema. Tot i l'estudi, l'anàlisi i el disseny inicial del sistema de moviment, no han estat pocs els problemes que han sorgit al llarg de la implementació.

9.1.2.1.- Estrebades

En alguns punts del document s'ha introduït la idea que els sistemes industrials estan dissenyats per assolir posicions, realitzar maniobres i reproduir un conjunt de moviments preconfigurats.

En el nostre primer intent de moure el robot de manera fluida, vam realitzar una implementació que consistia en incrementar/decrementar X unitats el valor de l'eix corresponent al punt actual de l'element terminal en funció de la comanda rebuda. D'aquesta manera, si ens trobàvem a la posició {10, -20, 87} i la comanda rebuda indicava que volíem moure'ns en sentit positiu a l'eix X, incrementàvem el valor en aquest eix X unitats (en funció de la velocitat establerta) obtenint el nou punt que calia assolir, {15, -20, 87} i iniciant el moviment al punt.

El primer problema va ser que les comandes arribaven i es processaven tan ràpid que el software detectava que el robot havia sortit de rang de treball abans de fer-ho. Aquest fet és deu a que VAL3 quan executa una comanda de moviment continua amb l'execució del codi abans que el robot assoleixi la posició sol·licitada per la unitat de control.

VAL3 té una comanda anomenada **waitEndMove()**. Aquesta comanda força l'aturada seqüencial del sistema fins que l'element terminal ha assolit la posició desitjada. D'aquesta manera es corregia el problema anterior, però en sorgia un de nou. Com la comanda forçava l'aturada d'execució del codi, el nou moviment no es processava fins que l'anterior havia finalitzat, provocant estrebades i sotracs en ens moviments del robot.

9.1.2.2.- Alteració de la trajectòria

En veure que la idea inicial presentava problemes, vam tornar a revisar el manual del llenguatge VAL3 i vam apreciar una secció anomenada "10.2 ALTER: CONTROL EN TIEMPO REAL DE UNA TRAYECTORIA". Aquest apartat ens indicava com podíem modificar una trajectòria en temps real mentre s'estava executant.

És a dir, imaginem que volem assolir el punt {10, -20, 87} però a mig camí, mentre la trajectòria s'està executant, cal realitzar una correcció i alterem el punt de destí perquè sigui {10, -20, -87}. Aquest era l'objectiu de treballar amb aquestes comandes, i d'aquesta manera els moviments serien fluïts i sense estrebades.

Després d'estudiar el funcionament de les comandes especificades en l'apartat, vam realitzar un seguit de proves, totes amb resultats infructuosos. Les comandes d'alteració de trajectòria no funcionaven com el manual especificava i no ens va ser possible esbrinar el motiu pel qual no actuaven segons la definició del manual.

9.1.2.3.- Llibreries/plug-ins

En un intent de trobar una solució al problema anterior, vam indagar a la xarxa i vam descobrir que Stäubli oferia un plug-in que permetia moure, de manera constant, l'element terminal en una direcció concreta.

Semblava que havíem trobat la solució, però un cop més no era així. Els pluguins requerien un usuari i una contrasenya per accedir a la database de Stäubli. Vam realitzar diversos intents de contacte amb el responsable del sistema Stäubli del laboratori, tots inútils, ja que no obteníem resposta.

Finalment, després de molts de dies i moltes proves frustrades, vam rebre una resposta del responsable per reunir-nos i mirar de trobar una solució. Ens va proposar un parell d'idees que podien ser viables i vam continuar realitzant proves.

9.1.2.4.- Escalar valors

Aquest va ser un problema menor que va quedar resolt amb l'anterior. L'obstacle era que no podíem trobar uns valors d'escalada entre el joystick i els moviments del robot, ja que la modificació de la velocitat ens alterava constantment els valors definits, fent que el braç es mogues massa poc a poc o molt ràpid.

Al modificar el sistema de moviment, aquest problema va quedar resolt automàticament.

9.1.2.5.- Manca d'informació

Com s'ha recalcat al llarg del document, Stäubli és una gran marca, no obstant, compren una part petita del mercat de robòtica industrial i el fet de no ser una de les marques més populars dificulta la cerca d'informació.

A més la política d'empresa aposta més pel servei tècnic que per una millor documentació. Aquest factor fa que la informació online sigui molt escassa i de difícil accés.

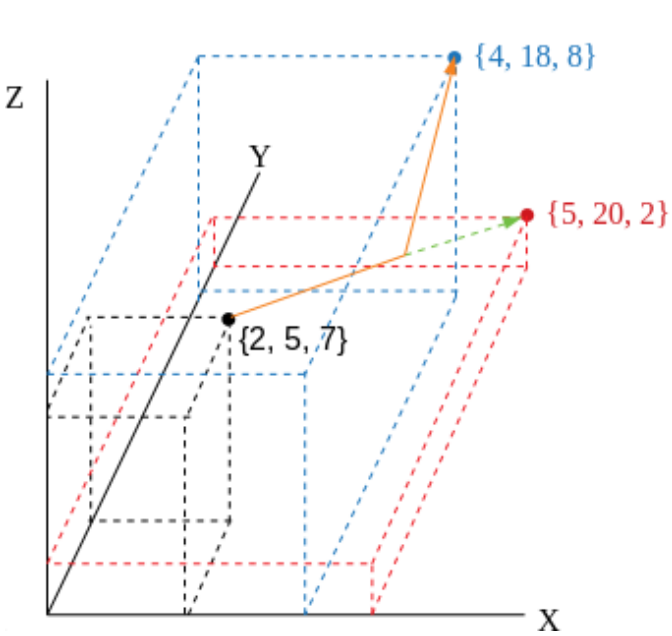
Un anècdota que ens va passar, va ser que mentre intentàvem solucionar el problema del moviment vam realitzar algunes consultes en un fòrum de robòtica i per sorpresa nostra, en el moment de cercar a Google informació sobre el moviment del Stäubli TX60, les primeres entrades que ens apareixien eren les consultes que havíem fet al fòrum. Aquest detall demostra la manca d'informació que existeix sobre aquesta marca.

9.1.2.6.- Solucions

Després de moltes proves sense resultat, se'ns va acudir realitzar una petita prova i calcular el temps d'execució per verificar si era viable. Vam realitzar un programa que calculava el punt més llunya que podia assolir l'element terminal en funció de en quins eixos ens volíem moure i en quin sentit ens volíem desplaçar. El temps de calcul va ser tant ràpid que no vam apreciar cap retard.

Finalment vam trobar la solució. Cada cop que s'executes una comanda de moviment, es calculava el punt més llunya per aquella configuració: eixos que volíem moure, en quin sentit i a quina velocitat. D'aquesta manera, vam obtenir un moviment fluït i en temps real que permetia maniobrar el robot còmodament i ens donava pas al desenvolupament de les següents etapes del projecte.

El següent gràfic mostra de manera conceptual el funcionament de la solució implementada.



- El punt negre són les coordenades de la posició actual

- Calculem el punt maxim que podem asolir donada una configuració, d'eix, direcció, sentit i velocitat, obtenint les coordenades del punt vermell

- Iniciem el moviment del punt negre al vermell.

-Si en mig de la trafectoria volem realitzar una modificació en la configuració eix, direcció, sentit i velocitat, recalculem el nou punt maxim (blau)

-De manera molt rapida, parem el moviment del robot i iniciem una nova comanda de moviment cap al nou desti (punt blai), creant l'ilusió de que el robot ha realitzat una correcció en la trajectoria

Figura 43: Representació de la solució

9.1.3.- Proves i resultats

A l'apartat de problemes i solucions, ja s'ha introduït algunes de les proves que s'han realitzat al llarg del desenvolupament en aquesta etapa del projecte. El calcul del punt més llunya que podíem assolir per aquella configuració del braç és el sistema que finalment s'ha desenvolupat i calia provar-lo en profunditat per assegurar que no presentava altres problemes no contemplats.

Les proves realitzades i resultats obtinguts del desenvolupament en aquesta etapa del projecte han estat:

- **Verificar correcció en el calcul del punt més llunya**
  - Comprovar que el calcul del punt més llunya fos correcte, donada una configuració de l'eina. Per realitzar aquesta prova, vam seguir aquests passos:
    - Mostrar per pantalla en tot moment els càlculs realitzats
    - Executar el nostre sistema de control de moviment
    - Amb el joystick assolir el punt més llunya que el sistema ens permetes fins que el braç quedes aturat
    - Canviar el mode de control des de la consola MCP, de mode desplaçat (mode de funcionament del nostre sistema) a mode manual (control del braç amb la MCP)
    - Activar la potencia del braç, ja que al canviar de mode es desactiva automàticament
    - Establir el mode de moviment (joint o frame) corresponent a la configuració introduïda per l'usuari
    - Intentar moure el robot per verificar que no avançava més en la direcció desitjada

  ✔ Després d'ajustar el calcul, ja que en alguns casos no arribava al punt més llunya, vam verificar que el calcul del punt més llunya es realitzava de manera correcte.

- **Maniobrabilitat a traves del dispositiu de control joystick**
  - Existeixen moltes maneres de configurar un dispositiu de tipus joystick per maniobrar un braç robòtic industrial. Cal que l'assignació que fem dels eixos del joystick sigui adequada i s'adapti als sistemes de referencia del robot.

    Aquesta prova va ser totalment empírica, ja que vam establir diverses configuracions del joystick per moure el robot i vam invertir algunes hores en maniobrar-lo per verificar quina era la configuració més adient.

  ✔ Finalment, després de provar diverses configuracions, consultar amb alguns company i entesos en aquests sistemes, vam optar per la configuració especificada a l'apartat 5.6.1 del present document.

- **Anàlisi del temps de resposta**
  - Un dels requeriments del sistema era poder moure el robot en temps real. Aquesta era una prova important. Per veure el temps de reacció vam fer servir dos mètodes, el calcul de temps transcorregut des que executàvem una ordre fins que es movia el robot, i observar a ull nu si existia algun tipus de retard en el moviment del braç.

  ✔ El temps de resposta era de pocs mil·lisegons i la percepció de delays a simple vista es feia impossible, per tant, podem dir que el sistema de moviment permet el control, en temps real, del TX60.

9.2.- Sistema de control per forces/moments

El fet de poder teleoperar el robot ja ha estat un gran avenç en el projecte, però els requeriments del projecte feia evidents la necessitat d'integrar un sistema de control que impedís a l'usuari realitzar moviments inapropiats.

Imaginem que estem en mig d'una operació rutinària, l'usuari teleopera el robot de manera natural, però per factors extres al sistema (estornut, un cop d'algun company, pèrdua de control de la motricitat, etc) comença a realitzar maniobres imprudent, donant lloc d'aquesta manera a que el pacient i/o l'entorn de treball pateixi danys.

9.2.1.- Algorismes i mètodes

Cal indicar que la major part del sistema de control de forces es desenvolupa en l'aplicació servidor, mentre que l'aplicació client només intervé per establir els llindars de treball.

A continuació es presentaran els elements de software que intervenen en el sistema de control per forces/moments.

9.2.1.1.- Joystick (C++)

- **void readJoystick ()**
  Quan polsem el boto 5 del joystick (veure apartat 5.6.1), iniciem un event que permet a l'usuari modificar el llindar de treball de les forces.

9.2.1.2.- Control (C++)

- **void setForces ()**
  Aquesta funció es cridada des de la classe joystick, mostrant un menú per pantalla que ofereix a l'usuari la possibilitat de modificar les forces de treball.

  A continuació mostrem les tasques que realitza el mètode:
  - Bloquegem la transmissió de comandes de moviment cap al servidor, ja que mentre s'està modificant el valor de les forces màximes no podem maniobranr.
  - En cas que el robot estigues realitzant una maniobra (en moviment), notifiquem al servidor que cal suspendre, de manera temporal la trajectòria, en execució.
  - Mostrar el menú de selecció de forces:
    - Standard (1): Aquesta opció estableix els valors màxims en -10 N (Newtons) per les forces en els eixos X, Y i Z
    - Custom (2): Aquesta opció permet establir valors de força diferents per cada eix. Si els valors introduïts no són vàlids, s'estableix el valor estàndard (-10 N)
  - Preparem la comanda i l'enviem al servidor
  - Restaurem el mode de teleoperació i reprenem la darrera trajectòria en execució

  La comanda que s'envia al servidor és: **P;x;y;z;**
  On **x**, **y** i **z** són els valor de força màxima que es pot exercir en cada eix

9.2.1.3.- TCPClient (C++)

- **bool sendData (string data)**
  Funció molt dedicada que només té com a finalitat enviar els valors de la variable **data** convertits en bytes pel socket configurat prèviament. Retorna un valor booleà en funció de si s'ha pogut enviar la informació exitosament.

9.2.1.4.- Staubli-Surgeon (VAL3)

- **initSensor ()**
  Aquest metode realitza dos tasques:
  - Inicialitzar el sensor
    El sensor pot presentar offsets en la mesura degut al seu propi pes i canvis en l'orientació.

    Aquest mètode calibra el sensor posant a zero tots els valors de forces, moments, posicions i rotacions en una orientació determinada.

    S'executa a l'inici del sistema, un cop el robot ha assolit la posició configurada per començar a teleoperar.

    La comanda que s'envia al sensor és: **122**
    Es necessari tractar el resultat que rebem després d'enviar la comanda, ja que podria ser que el sensor no l'hagues processat com cal per algun motiu. La id de resposta indicant que tot ha anat correctament és **123**. (Per més informació consultar el manual del sensor, especificat a la bibliografia)

  - Definir el llindar de forces de treball
    Inicialment establim el llindar de forces màximes que podem exercir a -10 N per tots tres eixos. Un cop el sistema s'hagi iniciat, l'usuari les pot modificar si ho creu convenient.

- **controlFTCL ()**
  Aquest mètode s'executa de manera asíncrona a la resta de codi de l'aplicació servidor.

  La finalitat d'aquest programa en el sistema de control de forces és realitzar lectures del sensor cada cuatre mil·lisegons. En cas que la força excedeixi els límits dels paràmetres de configuració és procedira de la següent manera:
  - Inhabilitació del control teleoperat
  - Suspensió de la trajectòria en execució
  - Assolir el darrer punt valid registrat
  - Restaurar el control teleoperat

  Només s'executara en cas de que no ens trobem en estat d'esperar una resposta de l'usuari a una pregunta enviada pel servidor. En aquest cas, no cal realitzar lectures del sensor de força, ja que el control teleoperat està inactiu fins que el servidor rep la resposta.

- **readSensor ()**
  Aquest mètode permet la comunicació directa amb el sensor de força. Aquesta comunicació es realitza a traves del port serie RS232 de la unitat de control CS8C.

  Les tasques que s'executen per realitzar una lectura completa de les forces/moments són:
  - Enviar al sensor la comanda **76** que indica que estem sol·licitant les forces i moments. (Més informació al manual d'usuari esmentat a la bibliografia).

  - Processar la resposta que els sensor ens envia. Si la resposta és un **77** vol dir que tot seguit ens enviara els valors numèrics pertinents a les forces i els moments.

  - Un cop rebem les dades de forces i moments del sensor, realitzem un processament que ens permet expressar les forces en Newtons i els moments en Newtons per metre, segons les indicacions del manual.
    - El processament consisteix en dividir els valors de les forces entre 32 i els dels moments entre 1024. Procediment realitzat seguint les indicacions del manual del sensor Schunk FTCL 50-80.

  - Finalment, emmagatzemem els valors llegits en variables al servidor per realitzar el control de moviment del robot.

- **saveCurrentPos ()**
  En el programa *controlFTCL()* ja introduíem que en cas de que el robot exercís més pressió de la permesa per les variables d'entorn, el sistema de control de forces aturava les maniobres teleoperades per assolir un punt anterior i alliberar aquestes pressions.

  Aquest programa s'encarrega de guardar aquesta posició anterior que en els casos especificats caldrà assolir.

  Només executarem aquest programa si l'aplicació servidor no es troba a l'espera de resposta per part del client. Cal remarcar que la posició que guardem s'actualitza cada 1.5 segons, únicament si no s'excedeix la pressió definida a les variables d'entorn en cap dels tres eixos.

- **socketTransfer ()**
  En aquest cas, el programa *socketTransfer()* funciona exactament igual que pel sistema de control de moviment: rep una comanda, fa la conversió a text, processa la primera lletra per veure de quin tipus de comanda es tracta i executa el programa corresponent.

  En aquest cas, la comanda que rebem és: **P;x;y;z;**
  On **x**, **y** i **z** corresponen als valors màxims de forces, establers per l'usuari, que el sistema pot suportar

- **setMaxForces (command)**
  Aquest programa assigna els valors màxims que poden adquirir els eixos processant els calors de la variable **command**. Aquest valors han estat definits per l'usuari.

9.2.2.- Problemes i solucions

La fiabilitat i l'eficiència són els dos factors que més s'han treballat en aquest sistema de control per forces/moments. Tots els sistemes, i més els sistemes que incorporen gran maquinaria requereixen sistemes de control fiables (no pot deixar de funcionar a mitja execució) i eficients (temps de resposta quasi immediats).

Al llarg de la implementació del sistema de control del nostre projecte han anat apareguen errors, dels quals, la gran majoria s'han pogut resoldre de manera rapida i eficient, ja que no comportaven conflictes amb el disseny inicial.

9.2.2.1.- Manca de canal de comunicació físic

A l'apartat 5.5, realitzàvem una explicació del sensor de força, on s'havia alterat la configuració de fabrica que portava el sensor per poder-lo connectar a un port serie RS232.

El primer problema va ser que cap dels ordinadors del laboratori disposaven d'una connexió serie RS232. Vam intentar aconseguir alguna PCI per poder connectar el sensor als PCs, però ens va ser impossible.

Figura 44: PCI connexió port serie RS232

Finalment vam aconseguir connectar el sensor a un dels ordinadors del laboratori mitjançant un conversor de port serie RS232 a port USB, resolvent d'aquesta manera el problema de connexió física.

Figura 45: Conversor de port serie RS232 a port USB

9.2.2.2.- Comunicació amb el sensor Schunk FTCL 50-80

Havent resolt el problema de la connexió física del sensor a l'ordinador, vam disposar-nos a realitzar lectures de forces/moments. Per fe-ho vam desenvolupar el codi necessari seguint les indicacions del manual, no obstant, els resultats no van ser els desitjats.

El sensor de força disposa d'un indicador lluminós que pampalluga en diversos modes amb els colors verd/vermell. A l'apartat 5.5.2 d'aquest document, podem trobar una descripció dels estats en que podem trobar el sensor, en funció dels senyals lluminosos.

En el nostre cas, independentment de la comanda que enviéssim al sensor, sempre obteníem com resposta pampallugues de color ver/vermell. Aquest mode indicava que alguna de les dues interfícies no funcionava de manera adient.

Després d'estudiar exhaustivament el codi implementat, vam descartar que fos error de la programació realitzada.

Finalment, vam atribuir l'error de comunicació, entre el PC i el sensor, al conversor de port serie RS232 a port USB, ja que era l'únic element del sistema que no controlàvem ni podíem analitzar.

9.2.2.3.- Lectures negatives

Després de resoldre el problema de la connexió i la comunicació (s'explica a l'apartat 9.2.2.5), vam començar a interactuar amb el sensor, inicialitzant-lo, realitzar lectures en diversos modes, etc.

Però hi havia quelcom que no acabava d'encaixar amb el sistema, els valors de les lectures, en alguns casos eren negatius.

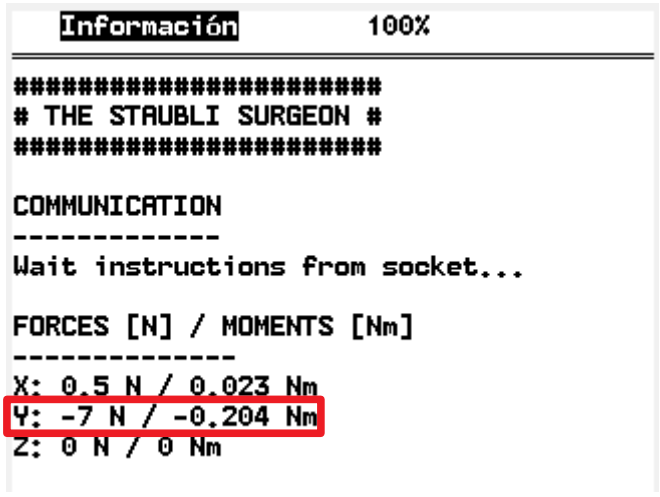


Figura 46: Representació de les forces

9.2.2.4.- Correlació entre els sistema de referencia

Aquest ha estat un error que ha passat desapercebut al llarg de tot el desenvolupament del sistema. Va aparèixer de manera casual mentre realitzàvem maniobres rutinàries.

El problema estava en que el sistema de referencia del sensor, el sistema de referencia del joystick i el sistema de referencia de l'element terminal del robot no coincidien.

La següent imatge (figura 47) ens mostra la relació que existeix entre els sistemes de referencia dels tres elements:

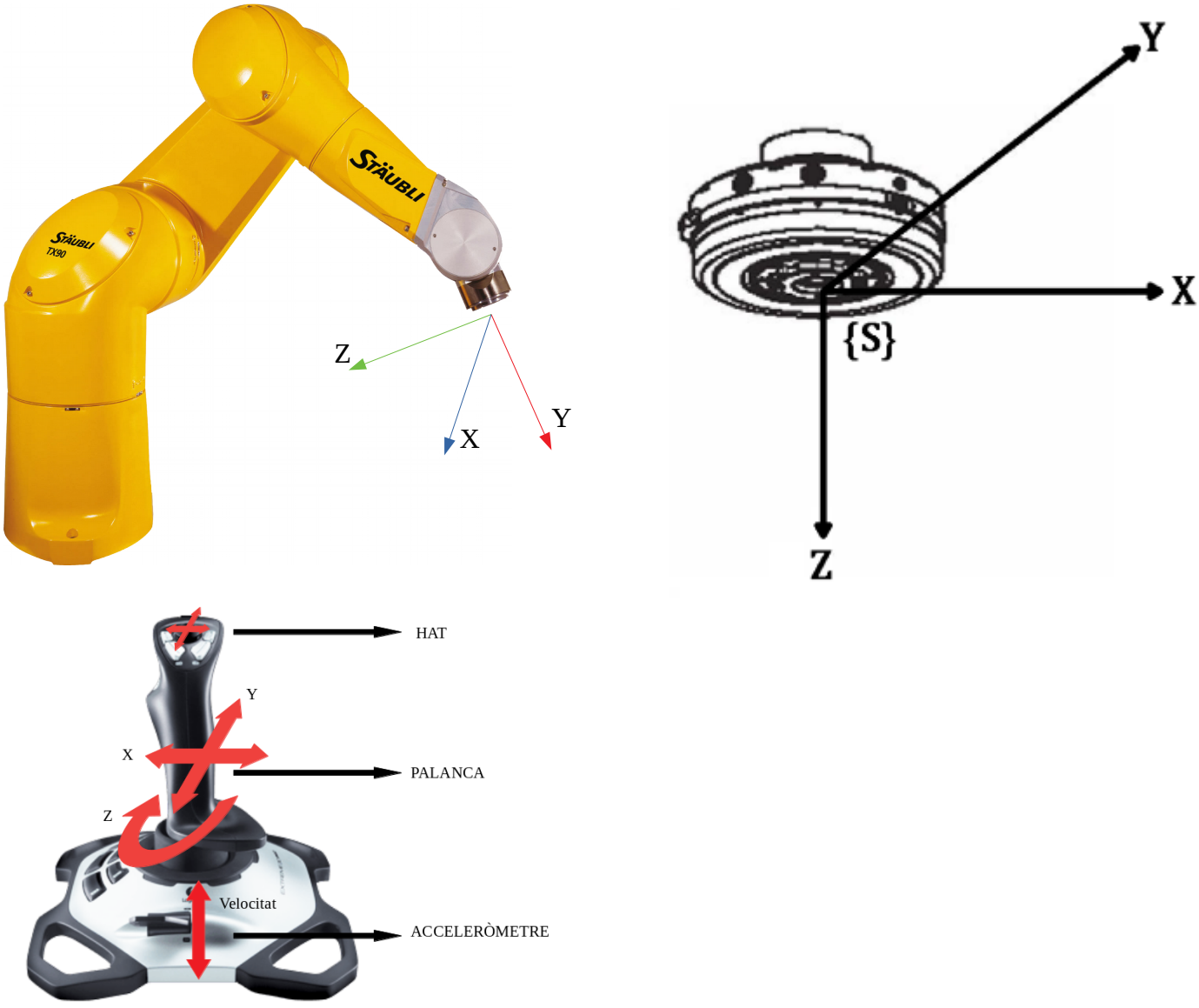


Figura 47: Correlació de sistemes de referència

Com podem apreciar, la relació entre el joystick i el TX60 és la que vam definir en el moment de dissenyar el sistema de moviment. Pel contrari, el sensor de força no es correspon amb el sistema de referencia del joystick i el robot.

9.2.2.5.- Solucions

Com ja s'ha esmentat, al llarg del desenvolupament del sistema de control han aparegut diversos contratemps, alguns més rellevants que d'altres.

- **ERRORS GREUS**
  - Connexió física

    L'error de la connexió física el vam resoldre després de moltíssims contratemps. La solució va esser connectar el sensor directament a la unitat de control Stäubli CS8C.

    El fet de connectar el sensor directament a la controladora oferia avantatges i inconvenients.
    - El principal avantatges que ens oferia és que tot el control de forces es realitzava a la part servidor, minimitzant les tasques que calia desenvolupar a la part client. D'aquesta manera, el sistema de control actuava com una caixa negra, on, en funció de les lectures realitzades, executava unes o altres maniobres.
    - El major inconvenient, era que el VAL3 no oferia les mateixes possibilitats de control que el C++ en el moment d'interactuar amb el sensor. Tot i poder realitzar les mateixes lectures, el codi resultat era molt mes complexe en VAL3 que en C++.

  - Comunicació inviable

    Com ja hem dit, aquest fet es devia al conversor de port serie RS232 a port USB. Un cop connectat a la unitat de control CS8C, vam ser capaços de comunicar-nos de manera efectiva, tot superant algun entrebanc que se'ns va aparèixer.

- **ERRORS IRRELLEVANTS**
  - Lectures negatives

    Després d'algunes proves vam descobrir que les lectures en alguns moments eren negatives perquè el sensor estava orientat de manera inversa, és a dir, apuntant cap al terra.

  - Correlació entre els sistemes de referencia

    El problema de la correlació entre els sistemes de referencia, en si mateix no era un problema, ja que tot el sistema es controlava des de la part servidor. L'únic que va caldre modificar, va ser el programa *setMaxForces()* per mapejar cada eix del sistema de referencia del joystick i l'element terminal, en el corresponent eix del sensor de força.

    D'aquesta manera, vam obtenir la següent correlació:
    - Eix X (joystick & terminal element) ⟶ Eix Z (force sensor)
    - Eix Y (joystick & terminal element) ⟶ Eix X (force sensor)
    - Eix Z (joystick & terminal element) ⟶ Eix Y (force sensor)

9.2.3.- Proves i resultats

Les proves realitzades en el sistema de control han estat orientades a verificar la fiabilitat i l'eficiència del sistema de control, i no tant en verificar el seu funcionament, ja que aquest darrer factor es va verificar en el moment de resoldre els problemes apareguts.

Les proves realitzades i resultats obtinguts del desenvolupament en aquesta etapa del projecte han estat:

- **Verificació de les lectures del sensor**
  - Comprovar que la lectura de les dades oferides pel sensor fos correcte. Per realitzar aquesta prova, vam seguir aquest passos:
    - Mostrar per pantalla en tot moment les lectures realitzades. Les lectures es realitzaven cada 0.5 segons per poder apreciar els canvis, mentre que en producció, es realitzen cada 4 mil·lisegons
    - Executar el sistema de control per forces/moments
    - Aplicar pressió en un eix de manera manual, es a dir, prement l'element terminal amb la ma en una direcció concreta.

  ✔ Després de realitzar diverses proves, aplicar diferents pressions, en els diferents eixos, vam verificar que les lectures de les forces/moments es realitzaven satisfactòriament.

- **Demostrar que el sistema és fiable i el sistema de control sempre s'executa**
  - Aquesta prova, en comptes de ser una verificació de funcionament, va esser una demostració de la fiabilitat del sistema.

    El sistema de control ha d'actuar per sobre de qualsevol altre tasca del sistema, a excepció d'una aturada d'emergència. Al el disseny ja varem introduïr que algunes tasques del sistema s'executen de manera asíncrona, independents del fil d'execució principal. D'aquesta manera garantim que sempre s'està executant el sistema de control.

    La prova en si mateixa, va consistir en estressar al sistema. Es van realitzar moviments molt ràpids, en diverses direccions, mentre es realitzaven altres tasques.

  ✔ El resultat va ser més del que s'esperava. El sistema de control no només es va executar tot el temps, sinó que va respondre de manera molt rapida a les maniobres que realitzàvem.

- **Demostració de l'eficiència del sistema a diverses velocitats**
  - Aquesta prova també consistia en realitzar una demostració. L'eficiència d'un sistema de control es basa en el temps de resposta, quan més petit sigui aquest temps, millor sistema de control obtenim.

    Igual que en la demostració de fiabilitat, aquí també vam estressar al sistema, però d'una manera encara més *"hard"*. Es van deshabilitar les restriccions de velocitat programades i vam realitzar tests a grans velocitats per verificar que el sistema de control era capaç d'actuar per sobre dels paràmetres de configuració que havíem establert.

  ✔ El resultat va superar les expectatives, ja que el sistema de control responia de manera efectiva, rapida i amb una configuració de velocitats superior a l'estipulada.

9.3.- Sistema de maniobra

Al llarg de tot el document hem introduït conceptes com, *"sistema de moviment"*, *"sistema de maniobra"*, *"les maniobres que realitza l'usuari"*, etc. És molt important distingir entre el **sistema de moviment** i el **sistema de maniobra**, tot i que en diverses ocasions ens hem referit al moviment com una maniobra.

9.3.1.- Descripció

Al llarg de tot el projecte s'han fet referencies a l'entorn mèdic, ja que aquesta practica té com a objectiu, simular operacions bàsiques de cirurgia. El sistema de maniobra, el que ofereix són tres modes de treball orientat a les simulacions que podran realitzar els estudiants un cop el sistema estigui implementat:

- **Standard:** Permet a l'usuari realitzar tasques de moviment sobre el braç robòtic. Realitza un control de forces, i en cas de superar la força màxima permeasa en algun eix, recula a una posició anterior valida, on la pressió exercida no superi els límits establers pel sistema.

- **Follow irregular path:** En la majoria dels casos, en la medicina, més concretament en la cirurgia, cal realitzar trajectòries rectes sobre una superfície irregular. Aquest mode permet a l'usuari realitzar una trajectòria recte sobre una superfície sense superar les forces definides al sistema.

- **Click the vein:** Una altre acció comuna en les cirurgies són les puncions. Aquest mode de maniobra permet a l'usuari realitzar puncions, com per exemple, simular la punció d'un anàlisis de sang.

ATENCIÓ!

Aquesta apartat, *"Sistema de maniobra"*, no consta en les etapes d'anàlisi i disseny.

La proposta inicial del projecte consistia en implementar un sistema que realitces les següents tasques:

- Implementar un sistema de control de moviment de manera remota i en temps real que mermetes maniobrar el Stäubli TX60

- Incorporar un sistema de control per forces/moments que impedis fer malbe l'entorn de treball

- Implementar un sistema de visió que ajudes a l'usuari a maniobrar el robot en cas de no disposar de linia visual directe amb aquest

Per diversos motius que ja s'exposaran en apartats posterior, no va ser viable complir amb la implementació del sistema de visió dissenyat inicialment. No obstant, el projecte si incorpora un sistema de visio, però no és tan sofisticat com el dissenyat inicialment.

El fet de no poder dur a terme el sistema de visió dissenyat incialment, ens va atorgar uns recursos temporals que vam invertir en desenvolupar aquest petit sistama de maniobra per facilitar les practiques que podran realitzar el alumnes del master *MAIA*.

9.3.2.- Algorismes i mètodes

El sistema de maniobra en si mateix no conforma una gran part del sistema, ja que només controlem en quin mode de maniobra ens trobem perquè el servidor pugui realitzar les accions pertinents.

A continuació es presentaran els elements de software que intervenen en el sistema de maniobra.

9.3.2.1.- Joystick (C++)

- **void readJoystick ()**
  Quan polsem el boto 6 del joystick (veure apartat 5.6.1), iniciem un event que permet a l'usuari modificar el mode de maniobra del sistema.

9.3.2.2.- Control (C++)

- **void setOperationMode ()**
  Aquesta funció es cridada des de la classe joystick, mostrant un menú per pantalla que ofereix a l'usuari la possibilitat d'escollir el mode de maniobra.

  A continuació mostrem les tasques que realitza el mètode:
  - Bloquegem la transmissió de comandes de moviment cap al servidor, ja que mentre s'està modificant el mode de maniobra no podem realitzar moviments.
  - En cas que el robot estigues realitzant un moviment, notifiquem al servidor que cal suspendre, de manera temporal la trajectòria, en execució.
  - Mostrar el menú de selecció de mode de maniobra:
    - Standard (1)
    - Follow irregular path (2)
    - Click the vein (3)
  - Preparem la comanda i l'enviem al servidor
  - Restaurem el mode de teleoperació i reprenem la darrera trajectòria en execució.

  La comanda que s'envia al servidor és: **O;n;**
  On **n** és un valor numèric del domini [1..3] que indica el mode de maniobra seleccionat.

- **void monitorizationSocket ()**
  El flux normal del sistema és on el client envia les diferents comandes al servidor perquè aquest les executi. No obstant, existeix un cas molt concret en el qual el servidor necessita consentiment de l'usuari per realitzar una tasca.

  En el mode de maniobra **"clickTheVein"** quan s'ha excedit la força definida als paràmetres del sistema, el servidor envia una comanda al client perquè aquest mostri per pantalla si l'usuari vol realitzar la punció. L'usuari pot respondre (y/n), en qualsevol cas es prepara la comanda de resposta i s'envia al servidor perquè realitzi o no la punció.

  La comanda que envia el servidor al client (pregunta) és: **A;**
  La comanda que envia el client al servidor (resposta) és: **D;x;**
  On **x** pren per valor **y/n** en funció de la resposta de l'usuari.

9.3.2.3.- TCPClient (C++)

- **bool sendData (string data)**
  Funció molt dedicada que només té com a finalitat enviar els valors de la variable **data** convertits en bytes pel socket configurat prèviament. Retorna un valor booleà en funció de si s'ha pogut enviar la informació exitosament.

- **string read ()**
  Funció molt dedicada que només té com a finalitat rebre les dades que el servidor ens envia pel socket prèviament configurat. Converteix els bytes que rep en una cadena de caràcters i la retorna.

9.3.2.4.- Socket-Surgeon (VAL3)

- **socketTransfer ()**
  En aquest cas, el programa *socketTransfer()* funciona exactament igual que pel sistema de control de moviment i pel sistema de control: rep una comanda, fa la conversió a text, processa la primera lletra per veure de quin tipus de comanda es tracta i executa el programa corresponent.

  En aquest cas, la comanda que rebem és: **O;n;**
  On **n** és un valor numèric del domini [1..3] que indica el mode de maniobra seleccionat.

- **controlFTCL ()**
  Aquest mètode s'executa de manera asíncrona a la resta de codi de l'aplicació servidor.

  Aquest programa controla l'acció que es dura a terme en el moment que l'usuari excedeixi les forces establertes al sistema. L'acció que realitzara serà diferent depenent del mode de maniobra en el que ens trobem.
  - Standard
    - El sistema inhabilita la maniobra remota per part de l'usuari, assoleix l'ultima posició coneguda on la força no excedia cap paràmetre del sistema i restaura el control remot.

- Follow irregular path
  - El sistema comprova que la força no s'excedeixi en l'eix Y.
  - Si fos el cas, deshabilitaríem el control remot de l'usuari, retrocediríem 5 unitats, ja que a la velocitat que maniobrem no es necessari allunyar-nos més, i habilitaríem el control remot.
  - Comprovaríem que la força no excedeix en cap dels altres 2 eixos (X, Z). Si fos el cas, deshabilitaríem el control remot, retrocediríem en l'eix Y una unitat de manera continua mentre les forces no es trobessin en els llindars definits.
  - Finalment, habilitaríem el control remot.

- Click the vein
  - Aquest mode de maniobra requereix consultar a l'usuari si desitja o no realitzar la punció.
  - Si escau, s'atura la trajectòria en execució, es deshabilita el control remot, es bloqueja la lectura de forces i s'envia al client una consulta per saber si cal o no realitzar la maniobra.

Només s'executara en cas de que no ens trobem en estat d'esperar una resposta de l'usuari a una pregunta enviada pel servidor. En aquest cas, no cal executar aquest programa, ja que la força, o bé ja l'estem excedint i estem esperant per realitzar la punció, o no realitzem la punció. De totes maneres, l'acció final, independentment del que l'usuari respongui, és alliberar la pressió exercida.

- **clickTheVein (command)**
  Dels tres modes de maniobra definits, el *"click the vein"*, és l'únic que te un programa a part, ja que les tasques que es realitzen són lleugerament mes complexes que alliberar la pressió exercida.

  Aquest programa es crida des de *socketTransfer()*, en el moment que el client respon la pregunta de si desitja o no realitzar la punció.

  La variable **command** representa una **y** o una **n**, en funció de l'opció escollida per l'usuari.

  En cas que la resposta de l'usuari sigui afirmativa, el servidor executa les següents accions:
  - L'element terminal del robot avança 3 unitats en l'eix Y per realitzar la punció
  - Esperem 5 segons (temps orientatiu per simular que estem extraient sang o agafant alguna mostra)
  - Assolim el darrer punt guardat al sistema on les forces de cap eix eren excedides
  - Habilitem el control remot perquè l'usuari pugui continuar maniobrant-lo

  En cas que l'usuari no desitgi realitzar la punció, simplement assolim el punt anterior on les forces no eren excedides i habilitem el control remot.

9.3.3.- Problemes i solucions

Aquest ha estat un dels sistemes desenvolupats que no ha presentat cap problema aparent al llarg de la implementació. La senzillesa en el funcionament ha permès que tot funciones correctament sense cap inconvenient major.

L'única petita dificultat, que no va ser un problema, va estar pensar com simular la punció, és a dir, quines eren les tasque que devíem realitzar, quants mil·límetres devíem avançar, quin temps devíem deixar transcorre mentre es realitzava la maniobra, etc.

Després d'indagar una mica i realitzar algunes consultes, vam definir el mètode esmentat anteriorment solucionava la present dificultat.

9.3.4.- Proves i resultats

En aquest cas, les proves van ser molt fàcils de realitzar. Com no se'ns va presentar cap problema, i el sistema era molt senzill vam realitzar les següents proves per verificar el correcta funcionament:

- Vam verificar que el canvi de mode de maniobra funciones correctament

- Vam comprovar que operant de manera remota el robot, realitzava les accions desitjades en funció del mode de maniobra en el que ens trobàvem.

- Vam verificar que si en mig d'un moviment s'intentava modificar el mode de maniobra, aquest es modificava satisfactòriament

✔ El sistema de maniobra compleix els requisits definits, permetent a l'usuari alternar entre els diferents modes, i garantint que depenen del mode seleccionat, el servidor dura a terme les accions programades.

9.4.- Sistema de monitorització

El entorns industrial, molt sovint incorporen molts paràmetres de sistema que permeten realitzar una configuració personalitzada en funció de l'ús que li vulguem donar. Alguns d'aquest paràmetres podrien ser; velocitats, posicions, rotacions, activació/desactivació de components, canvis de mode de treball, etc.

El nostre sistema, tot i estar enfocat a un entorn mèdic, conte elements industrials que requereixen paràmetres de configuració. Per facilitar la interacció amb el sistema, s'ha implementat una petita capa de software que mostra per pantalla els paràmetres del sistema i els manté actualitzats.

9.4.1.- Algorismes i mètodes

El sistema de monitorització és el més simple de tots els que conformen el projecte. Simplement, mostrem per pantalla els diversos paràmetres que ens permeten configurar el sistema.

A continuació es presentaran els elements de software que intervenen en el sistema de monitorització.

9.4.1.1.- Control (C++)

- **void showData ()**
  Aquesta funció mostra per la pantalla de l'ordinador tots els paràmetres del sistema. S'actualitza cada cop que un paràmetre es modifica. Només s'executa després de que el sistema s'hagi iniciat per primera vegada i l'usuari hagi establir els paràmetres necessaris.

Figura 48: Monitorització dels parametres

9.4.1.2.- Socket-Surgeon (VAL3)

- **showForceData ()**
  El programa mostra, per la pantalla de la MCP, cada 0.5 segones les forces i moments que rebem del sensor Schunk FTCL 50-80.

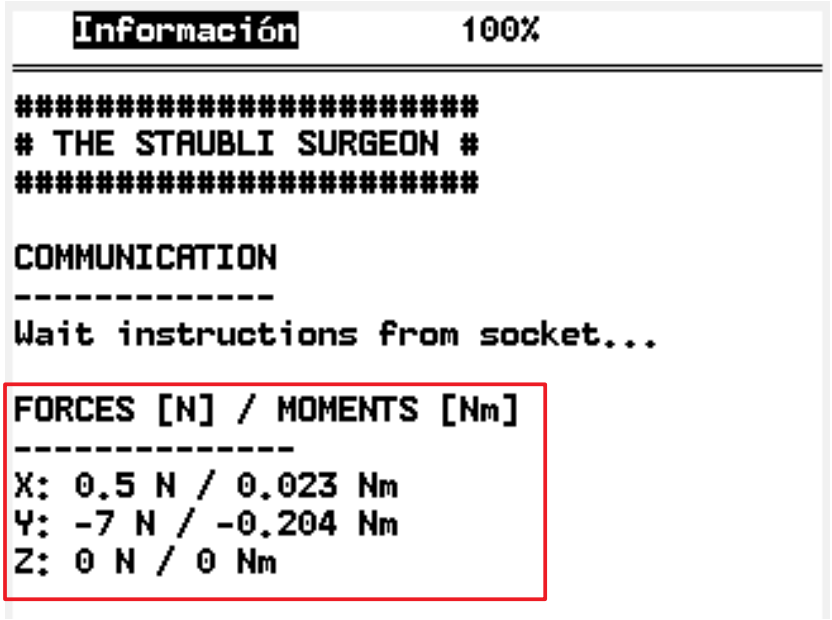


Figura 49: Monitorització de les forces

9.4.2.- Problemes i solucions

Si la monitorització presentes algun problema, no afectaria de manera directa al sistema, ja que no hi ha acció immediata en les tasques que aquest desenvolupa. En el pitjor dels casos, l'usuari podria estar confós per apreciar unes dades que no es corresponen amb l'estat o les accions del sistema.

A continuació s'exposa el principal problema que ha sorgit en el moment d'implementar la monitorització.

9.4.2.1.- Mostrar a l'ordinador les forces/moments en temps d'execució

En el disseny inicial del sistema, el sensor de força es comunicava amb l'ordinador de laboratori, aquest processava les dades, i en funció de la configuració dels paràmetres del sistema i les maniobres de l'usuari, indicava a l'aplicació servidor quines eren les accions que calia realitzar. D'aquesta manera, les lectures de les forces eren conegudes per l'aplicació del client i es podien mostrar per pantalla.

Com ja s'ha exposat en l'apartat d'implementació del *"sistema de control per forces/moment"*, el sensor de força va acabar connectant-se al la unitat de control CS8C.

Aquesta modificació en el disseny implicava que calia enviar les lectures del sensor de força del servidor al client. L'enviament d'aquesta informació implicava que el canal de comunicació romangues ocupat massa temps, impedint que algunes de les comandes arribessin a temps al servidor.

9.4.2.1.- Solució

De tots els problemes que s'han solucionat, aquesta solució és la que menys ens convenç, tot i que el problema queda resolt.

El fet d'enviar les dades del servidor al client, tot i ser un procés farragós (calia convertir caràcter a carates les lectures de les forces que volíem enviar a valors ASCII), impossibilitava la maniobra remota en temps real, ja que algunes comandes de moviments vitals pel sistema arribaven amb unes dècimes de segon de retard.

La decisió final, va estar mostrar els valors de les forces/moments per la pantalla de la consola MCP, d'aquesta manera el canal de comunicació quedava lliure de manera quasi immediata.

El fet de mostrar les dades separades en 2 pantalles diferents (ordinador i MCP) no ens acaba de convèncer, ja que el disseny inicial va ser centralitzar tota la informació del sistema perquè només un element actues com a unitat de control superior al sistema.

9.4.3.- Proves i resultats

Les proves realitzades en aquest sistema tenien com finalitat demostrar que les dades que mostrem són les correctes i s'actualitzen de manera rapida. A traves de les proves realitzades, va ser com vam descobrir el problema exposat anteriorment.

Les proves realitzades i resultats obtinguts del desenvolupament en aquesta etapa del projecte han estat:

- **Verificar la transmissió de les forces/moments entre servidor i client**
  - Per realitzar aquesta prova vam codificar els metodes pertinents, tant a la part del servidor com a la part del client. Una vegada implementada la part del servidor que enviava les dades i la part del client que les rebia, calia verificar que les dades eren correctes.
    - El servidor realitzava les lectures del sensor de força
    - Convertia les dades llegides, caràcter a caràcter, en mode ASCII i les enviava al client
    - El client rebia caràcter a caràcter els valors de les lectures realitzades, corresponents a les forces/moments en els tres eixos (moltíssima informació), els processava i els mostrava per pantalla

  ✔ El procediment d'interacció entre el servidor i el client permetia que les dades es mostressin a la pantalla de l'ordinador de manera correcta

- **Calcular el temps de transmissió de la informació**
  - Aquesta prova ens va revelar l'existència d'un problema que no havíem detectat anteriorment.

    El calcul del temps consistia en iniciar un comptador des que es mostrava un missatge per la pantalla de la MCP fins que les dades apareixien a la pantalla de l'ordinador. El temps de transmissió rondava el segon.

  ✗ Tot i que la prova es va realitzar correctament, el sistema de transmissió de les dades del sensor no era factible, per aquest motiu es va solucionar mostrant les dades per la MCP.

9.5.- Sistema de visió

El sistema de visió, és el darrer que incorpora el nostre sistema, oferint a l'usuari una aproximació a l'entorn de treball.

9.5.1.- Descripció

Teleoperar un robot de les magnituds del Stäubli TX60 amb un joystick, pot arribar a ser una tasca molt complicada. El no poder realitzar les maniobres a una distancia propera, per motius de seguretat, fa que sigui quasi impossible realitzar moviments precisos com els que necessitem simular.

Aquí és on entra en joc el sistema de visió, oferint una aproximació a l'entorn de treball. D'aquesta manera, tot i no tenir línia visual amb el robot, podem realitzar operacions molt precises.

El disseny inicial del sistema de visió consistia en muntar 2 càmeres en mode estereoscòpic per oferir a l'usuari una imatge amb volum i profunditat i que pogués apreciar encara millor l'entorn.

Per desgracia, aquest sistema no s'ha pogut dur a terme com s'esperava, pels següents motius:

- Els problemes al *"sistema de moviment"* i al *"sistema de control per forces/moments"*, ens han obligat a invertir més recursos temporals dels estimats inicialment

- Per motius de tercers, ens va mancar material per poder muntar el sistema de visió

No obstant, s'ha implementat un sistema de visió senzill que ofereix una petita millor en la maniobra del sistema.

9.5.2.- Disseny implementat

A l'apartat 5.7 d'aquesta memòria, ja introduíem el sistema de visió que implementaria el sistema.

Les següents imatges (figura 50) ens mostres com s'ha implementat finalment el sistema de visió.

Figura 50: Connexió de les càmeres

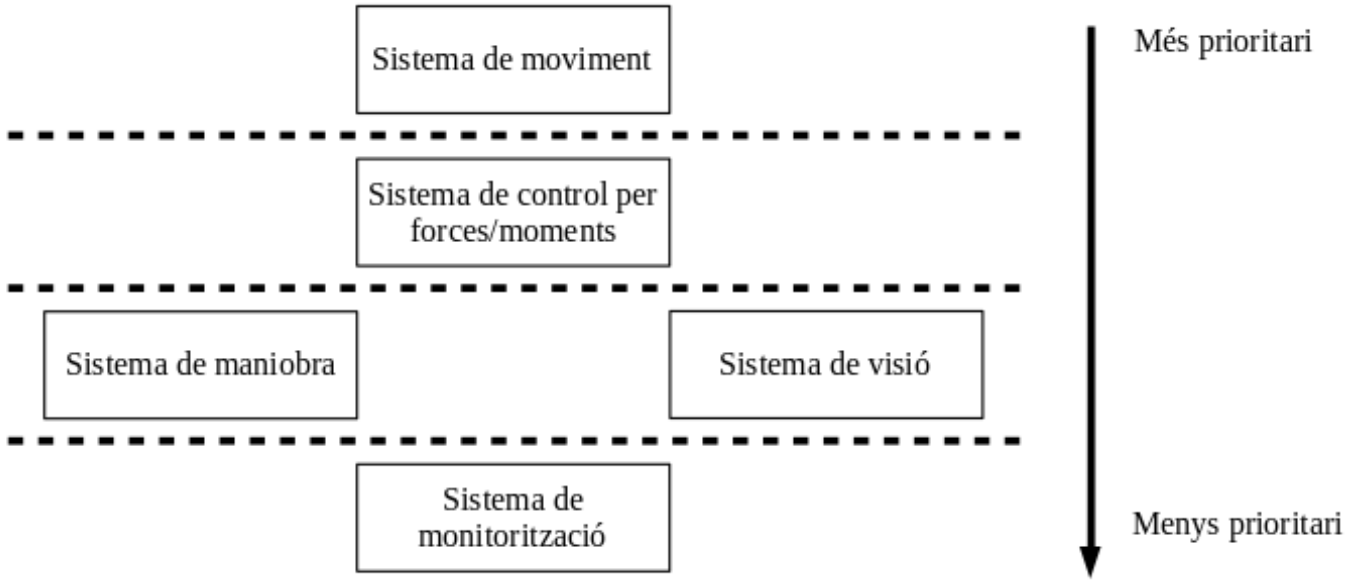
9.6.- Resultat d’assoliment dels objectius

En aquest apartat exposarem el grau d’assoliment del objectius. Per fer-ho analitzarem tots els sistemes que formen part del projecte, tot indicant quins han estat els objectius assolits.

Classificarem cadascun dels sistemes amb un del següents símbols per indicar el grau d’assoliment dels objectius:

- ✔ Tots els objectius del sistema s’han assolit satisfactòriament
- ✗ Cap dels objectius del sistema s’han assolit satisfactòriament
- \- Aquest sistema no compleix tots els objectius especificats inicialment
- \+ Aquest sistema no estava en el disseny inicial, però s’ha implementat i forma part del sistema

La següent imatge mostra la jerarquia dels sistemes que formen part del projecte resultant:



9.6.1.- Sistema de moviment (✔)

El sistema de moviment ha estat l’objectiu principal al llarg de tot el proces de desenvolupament del projecte. Tot i els entrebancs que se’ns han presentat, hem aconseguit assolir el 100% dels objectius establerts inicialment:

- Implementació d’un sistema de moviment per Stäubli TX60 amb un joystick
- Maniobrar el braç robotic en temps real
- Maniobrar el braç a diferents velocitats
- Maniobrar en 2 modes de moviment diferents (JOINT/FRAME)
- Maniobrar de manera fluida realitzant correccions en la trajectoria en execució

9.6.2.- Sistema de control per forces/moments (✔)

El sistema de control per forces/moments, ha estat el segon objectiu més important de tot el projecte. Finalment, s'ha aconseguit assolir el 100% dels objectius establerts inicialment:

- Realitzar lectures de les forces/moments del sensor Schunk FTCL 50 – 80
- Permetre que l'usuari estableixi els llindars de treball referent a les forces que es poden aplicar
- Establir un sistema de control en funció del llindars de treballs definits per evitar malmeses en l'entorn de treball

9.6.3.- Sistema de maniobra (+)

El sistema de maniobra no constava en el disseny inicial. Al llarg del desenvolupament del projecte ens vam adonar de la necessitat d'implementar-lo. A continuació es presenten els objectius assolits:

- Definir diversos modes de maniobra *(Standard, Follow irregular path, Click the vein)*
- Permetre a l'usuari alternar entre els modes de maniobra
- En el moment que s'excedeixi la força establerta als paràmetres del sistema, actuar en funció del mode de maniobra actual
- Realitzar tasques mediques/quirúrgiques de nivell molt bàsic

9.6.4.- Sistema de monitorització (+)

Aquest sistema, igual que l'anterior, tampoc constava en el disseny inicial del projecte. Vam decidir realitzar-lo, ja que consideràvem que era una necessitat, era senzill i calia invertir els recursos temporals obtinguts, ja que el sistema òptic no va sortir com s'esperava. A continuació es presenten els objectius assolits:

- Mostra tots els paràmetres del sistema a la pantalla de l'ordinador i a la MCP
- Actualitza l'estat dels paràmetres de manera instantània si l'usuari els modifica

9.6.5.- Sistema de visió (-)

Per desgracia, aquest sistema no s'ha pogut dur a terme en la seva totalitat, ja que diversos factors ho han fet inviable. A continuació presentem els objectius del disseny inicial que s'han assolit:

- Integració d'un sistema de visió que permet a l'usuari apreciar l'entorn de treball de més a prop
- Aproximació a un entorn de visió estereoscòpic (només en alguns moments de l'execució)

9.6.6.- Objectius implícits

Al llarg del desenvolupament del projecte, s'han assolit objectius que no poden ser emmarcats dins dels sistemes anterior. Aquest objectius són:

- Establir un canal de comunicació, fiable i eficient, entre client/servidor
- Implementació d'un sistema robust que impedeixi errors no controlats

10.- CONCLUSIONS

Aquest treball ens ha permès aprendre una gran quantitat de conceptes relacionats amb el sector informàtic industrial i ha fet que ens introduíssim en el món de la medicina/cirurgia a nivell molt bàsic. El sector industrial, com molts altres sectors tecnològics, avança cada vegada més ràpid, oferint noves positivitats d'expansió en els sectors que ja es troba present, i brindant la possibilitat d'introduir-se en els que encara no ho està.

La realització d'aquest treball, m'ha aproximat una mica més al sector de la informàtica industrial, permetent-me oferir aquest sistema com punt de partida d'un projecte que podria ser quelcom més gran en un futur no gaire llunya.

Per establir una bona conclusió de projecte, a continuació, analitzem el grau d'assoliment que hem realitzat en el desenvolupament del projecte, analitzarem els entrebanc que han sorgit i realitzarem una comparativa amb altres sistemes similars.

10.1.- Assoliments

Considerem que la feina realitzada assoleix de manera molt correcte els objectius del sistema que es va dissenyar inicialment.

- Creació d'un sistema de moviment fluït, de velocitat variable, intuïtiu, en temps real, fiable i eficient

- Hem integrat al sistema de moviment, un sistema de control a traves d'un sensor de força/moments que impedia al sistema malmetre l'entorn de treball

- Hem definit i preparat 5 practiques bàsiques pels estudiants del màster *MAIA*

- Hem dotat al sistema d'una monitorització per controlar les variables d'entorn

- Hem integrat un sistema de visió bàsic per facilitar a l'usuari la maniobra remota del robot

Valorant els motius presentats, creiem que hem assolit clarament l'objectiu principal del treball, el qual era la creació d'un sistema que permetes als estudiants del màster *MAIA*, realitzar simulacions mèdiques/quirúrgiques bàsiques amb el robot Stäubli TX60.

10.2.- Entrebancs

Com en tot bon projecte, el fet de realitzar un estudi en profunditat, realitzar un anàlisi exhaustiu i dissenyar un bon sistema, no impedeix que apareguin problemes i/o entrebancs.

El nostre projecte no ha estat cap excepció, ja que han sortit diversos problemes, alguns sense importància, però d'altres força més greus, obligant-nos a reestructurar part del sistema, modificar connexions físiques, alterar protocols de comunicació, entre altres que ja s'han exposat en apartats anterior del present document.

Tot i els entrebancs apareguts, estem contents amb la feina feta, ja que considerem els problemes que han sorgit, com errors constructius que ens ajudaran en un futur a millorar la metodologia de treball

10.3.- Comparativa

En el sector de la medicina existeixen robots d'alta tecnologia que permeten realitzar maniobres amb una precisió micromètrica, amb sistemes de control molt avançats que ofereixen suport al cirurgia en temps d'execució, entre altres factors que escapen al nostre coneixement actual sobre medicina.

A l'inici del projecte, ja es va deixar clar, que el sistema que implementàvem no podria comparar-se, ni de lluny, amb la tecnologia de robòtica mèdica que es trobava al mercat actualment. No obstant, tot i no poder arribar als nivells de resolució de robots del calibre del *DaVinci*, cal remarcar que moltes de les tasques que fan aquest robots mèdics, també les pot fer el nostre sistema. Per exemple, incisions, puncions, estudis mitjançant la pressió sobre diverses superfícies, etc.

Amb aquesta informació, podem valorar que el nostre sistema, tot i no poder competir amb els sistemes mèdics, ofereix un ampli ventall de possibilitats per realitzar simulacions en aquest entorn.

11.- TREBALL FUTUR

Arribats a aquest punt, disposem d'una primera versió estable del sistema de control remot en temps real del robot Stäubli TX60 mitjançant un joystick.

En coneixement del sistema, del errors i problemes resolts, podem veure moltes possibles maneres de continuar amb el desenvolupament del sistema oferint més possibilitats de simulació.

- **Centralització de la informació**
  Amb una mica més de temps, esforç i investigació, estem segurs que es podria centralitzar tota la informació en un únic punt del sistema, oferint d'aquesta manera una única unitat de control que governes el sistema de manera encara més eficient. Aquesta millora també ens oferiria certa independència de la MCP, ja que les forces/moments es podrien veure per la pantalla de l'ordinador.

- **Millorar el sistema de visió**
  Un millor sistema de visió, amb càmeres d'alta resolució, una il·luminació adequada i el correcta processament de les dades òptiques, atorgaria ulls al sistema, permetent una teleoperació del robot més còmoda i eficient.

- **Adaptar l'element terminal del TX60**
  L'adaptació de l'element terminal del robot per una base on es poguessin acoblar diversos estris (xeringa, pinça, bisturí, etc) ampliaria l'oferta de simulacions que ofereix el sistema.

- **Integració d'un sistema de realitat augmentada**
  Una de les millores més visuals, seria implementar un sistema de realitat augmentada, on digitalment poguéssim incloure un òrgan (pulmó, fetge, cor, cervell, etc) a l'entorn de treball, però que físicament no existís. Aquesta millora donaria molt de realisme al sistema ajudant a l'estudiant a introduir-se en un entorn més mèdic que industrial.

- **Realitzar consultes a personal mèdic**
  Al llarg del projecte, vam estar a punt, en 2 ocasions, de reunir-nos amb personal mèdic per realitzar una demostració del sistema implementat i que el poguessin valorar, realitzant critiques constructives per millorar-lo. Per diversos factors, aquestes reunions mai es van dur a terme.

  Seria interessant mostrar el sistema a personal d'aquest sector, amb coneixements en les maniobres que volem realitzar, perquè ens oferissin una critica constructiva amb la finalitat d'ajustar i millorar el sistema inicial.

Existeixen moltes altres maneres d'ampliar i millorar el sistema. Aquest robot, Stäubli TX60, ofereix moltíssimes possibilitats en molts de sectors, i crec que la universitat no està extraient tot el profit que en podria treure.

Espero que en un futur no gaire llunya, aquest sistema s'amplii, es millori i arribi a ser quelcom que tot estudiant vulgui fer servir per realitzar simulacions mediques en el seu pas per la Universitat de Girona.

12.- BIBLIOGRAFIA

- Stäubli (2005). *Manual de instrucciones: Brazo – familia TX Serie 60*. Stäubli Faverges. D28060205B.

- Säubli (2006). *Manual de instrucciones: Armario de control CS8C*. Stäubli Faverges. D28062905A.

- Säubli (2006). *Manual de usuario: Stäubli Robotics Studio (SRS)*. Stäubli Faverges. D28061605B.

- Säubli (2006). *Manual de referencia VAL3: Version 5.3*. Stäubli Faverges. D28062805A.

- Schunk (2008). *Assembly and Operating Manual: Force-torque Sensor Type FTC/FTCL*. Schunk GmbH & Co.

- Ollero, A. (2001). *Robótica: Manipuladores y robots móviles.* 1ª edició. Marcombo. ISBN 84-267-1313-0

- Yadav, R. (2008). *Client/Server Programming with TCP/IP Sockets*. 1a edició. DevMentor.org.

- Rahul Sreedharan. (7 Juny 2015). *Serial Port Programming on Linux*. Recuperat de: http://xanthium.in/Serial-Port-Programming-on-Linux

- Vladimir Iachimovici (18 Octubre 2007). *How to connect to a serial port in Linux using c++?*. Recuperat de: https://softexpert.wordpress.com/2007/10/18/how-to-connect-to-a-serial-port-in-linux-using-c/

- Keith Lantz. (5 Octubre 2011). *A linux c++ joystick object*. Recuperat de: https://www.keithlantz.net/2011/10/a-linux-c-joystick-object/

- Manohar P. Kuse. (19 Desembre 2015). *Programming with Joystick on Linux*. Recuperat de: https://kusemanohar.wordpress.com/2015/12/19/programming-with-joystick-on-linux/

- Wikipedia. *Joystick*. Recuperat de: https://en.wikipedia.org/wiki/Joystick

13.- ANNEXOS

13.1.- ANNEX [A]: Manual d'usuari

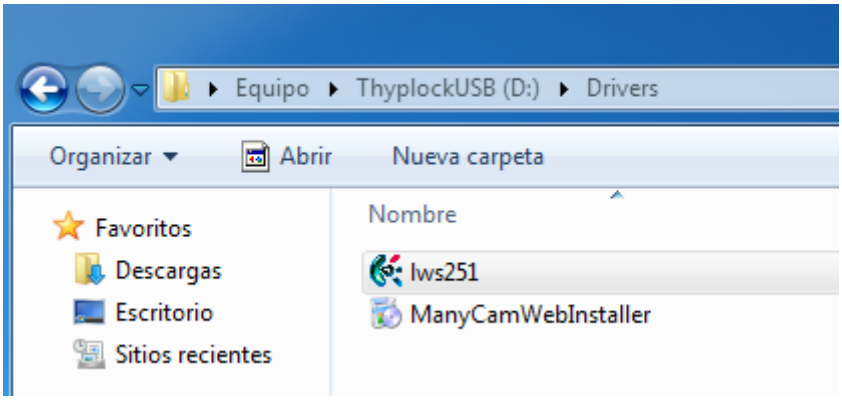
1. **Engegar el PC del laboratori**

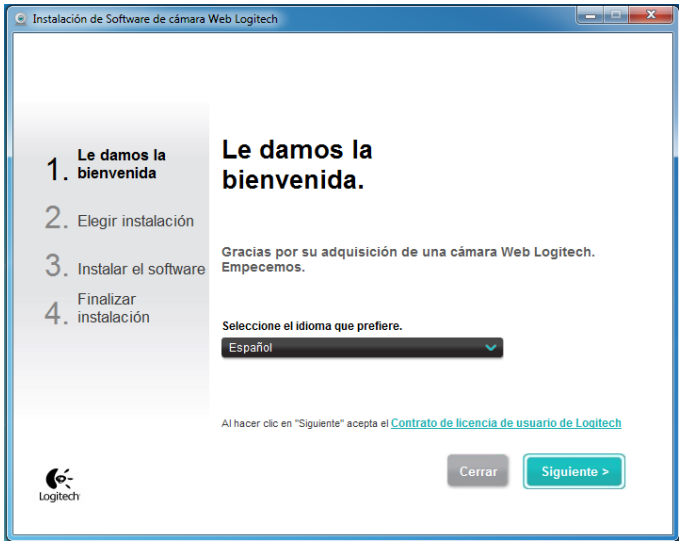
2. **Un cop el sistema Windows està inicial, connectar el dispositiu Logitech Extreme 3D Pro (joystick) al PC**

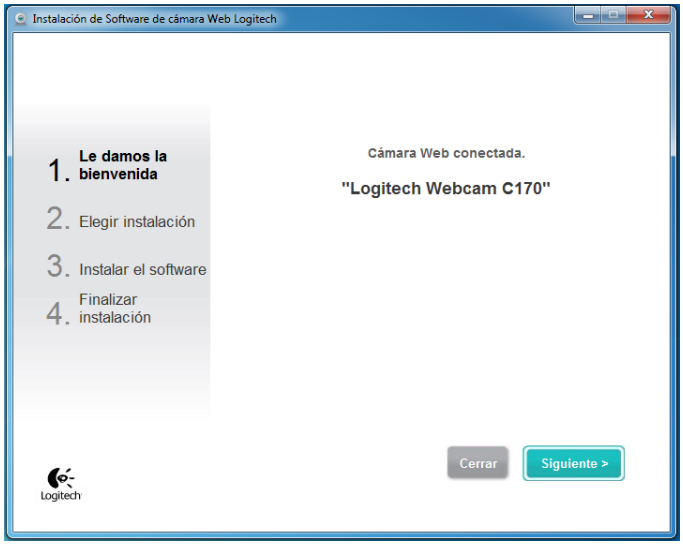
3. Connectar les càmeres Logitech C170 al PC

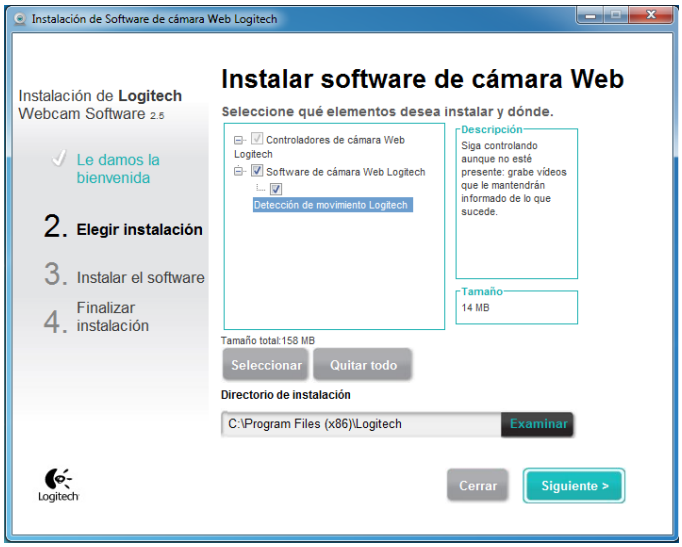
4. Instal·lar els drivers de les càmeres Logitech C170

Per realitzar la instal·lació del driver de les càmeres, només cal iniciar l'executable indicat a la següent imatge i seguir els passos que ens indica.

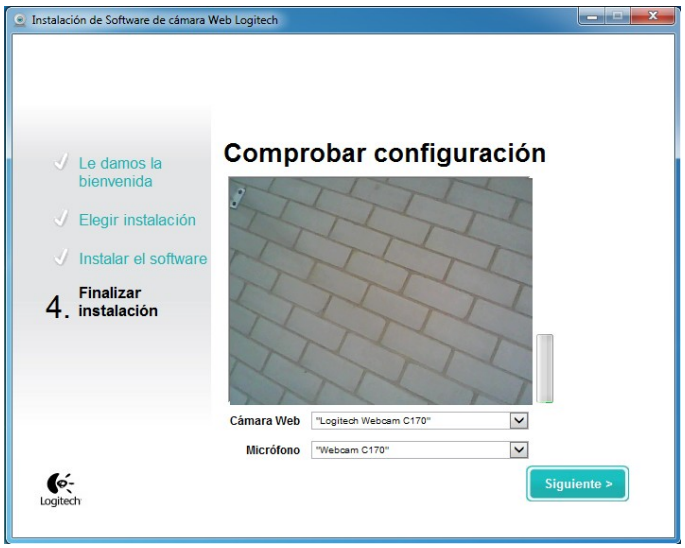


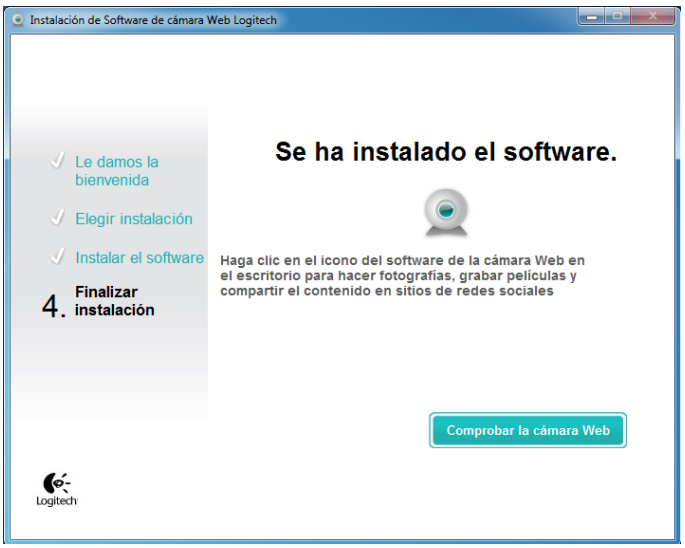





Instalación de Software de cámara Web Logitech
Instalación de Logitech Webcam Software 2.5
Instalar software de cámara Web
Seleccione qué elementos desea instalar y dónde.
Le damos la bienvenida
2. Elegir instalación
3. Instalar el software
4. Finalizar instalación
Controladores de cámara Web Logitech
Software de cámara Web Logitech
Detección de movimiento Logitech
Descripción
Siga controlando aunque no esté presente; grabe vídeos que le mantendrán informado de lo que sucede.
Tamaño
14 MB
Tamaño total:158 MB
Seleccionar
Quitar todo
Directorio de instalación
C:\Program Files (x86)\Logitech
Examinar
Cerrar
Siguiente >
Logitech


Instalación de Software de cámara Web Logitech
Instalando software
Espere mientras se instala el software
Le damos la bienvenida
Elegir instalación
3. Instalar el software
4. Finalizar instalación
E-mail
Edit Video
Facebook
YouTube
Haga fotografías. Grabe vídeo. Envíe correo electrónico o realice cargas en Facebook® o YouTube™ con una sola pulsación.
Instalando Logitech Webcam Software
Progreso total(12%)
Tiempo restante: 04:37
Logitech

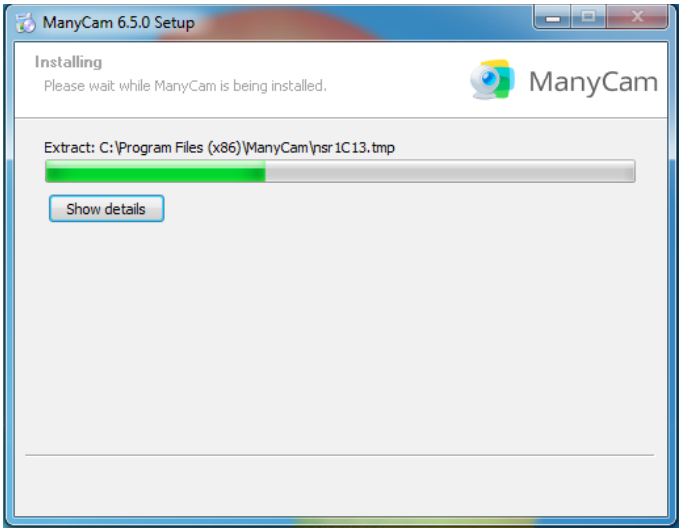

Instalación de Software de cámara Web Logitech
Comprobar configuración
Le damos la bienvenida
Elegir instalación
Instalar el software
4. Finalizar instalación
Cámara Web
"Logitech Webcam C170"
Micrófono
"Webcam C170"
Siguiente >
Logitech

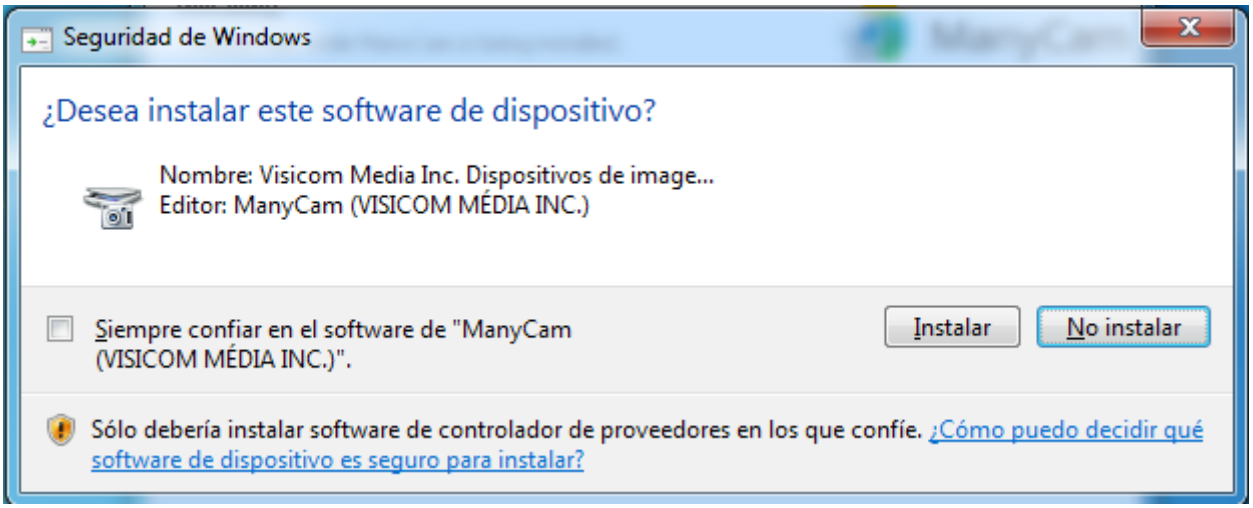

Instalación de Software de cámara Web Logitech
Se ha instalado el software.
Le damos la bienvenida
Elegir instalación
Instalar el software
4. Finalizar instalación
Haga clic en el icono del software de la cámara Web en el escritorio para hacer fotografías, grabar películas y compartir el contenido en sitios de redes sociales
Comprobar la cámara Web
Logitech

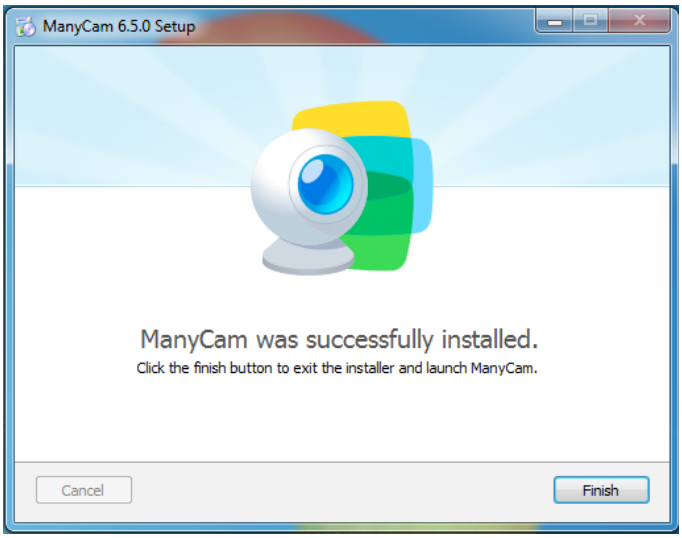

Logitech® Webcam Software
Preferencias
Ayuda
Webcam C170
Cambiar cámara Web >
Captura rápida
Tomar fotos y grabar vídeos.
Galería de cámara Web
Vea y comparta las imágenes que haya capturado.
Realizar una videoconferencia o más cosas
Controlador de cámara Web
Detección de movimiento
Logitech

5. Instal·lar el software ManyCam

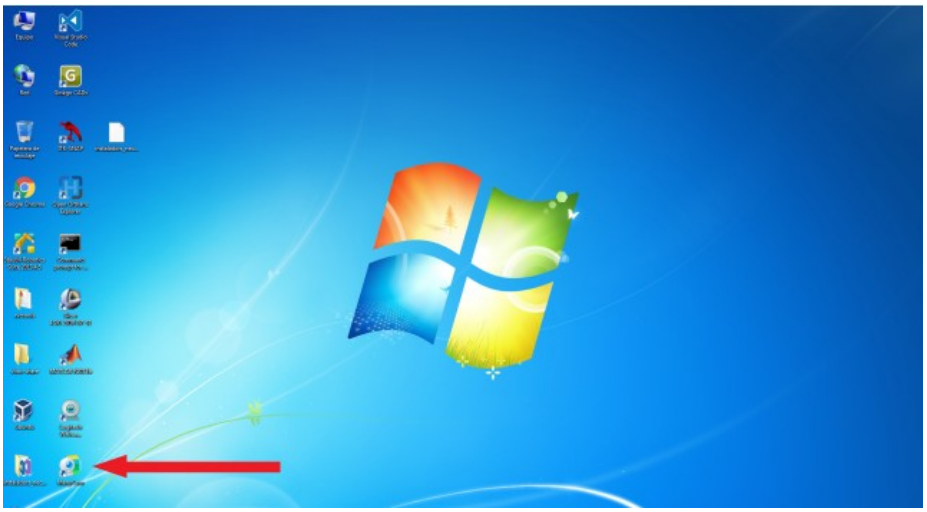
Per realitzar la instal·lació del software ManyCam, només cal iniciar l'executable indicat a l'anterior imatge i seguir els passos que ens indica.

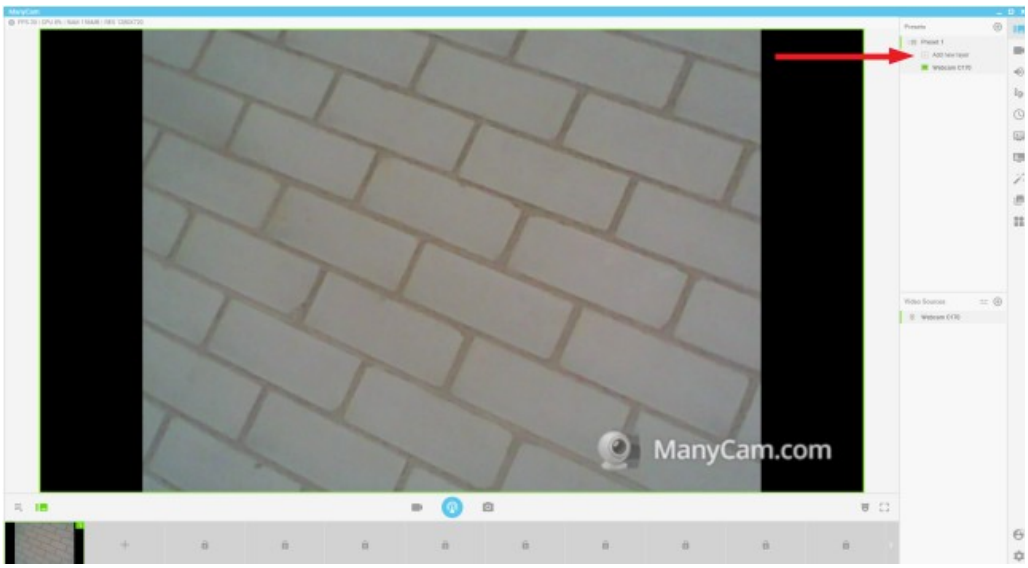


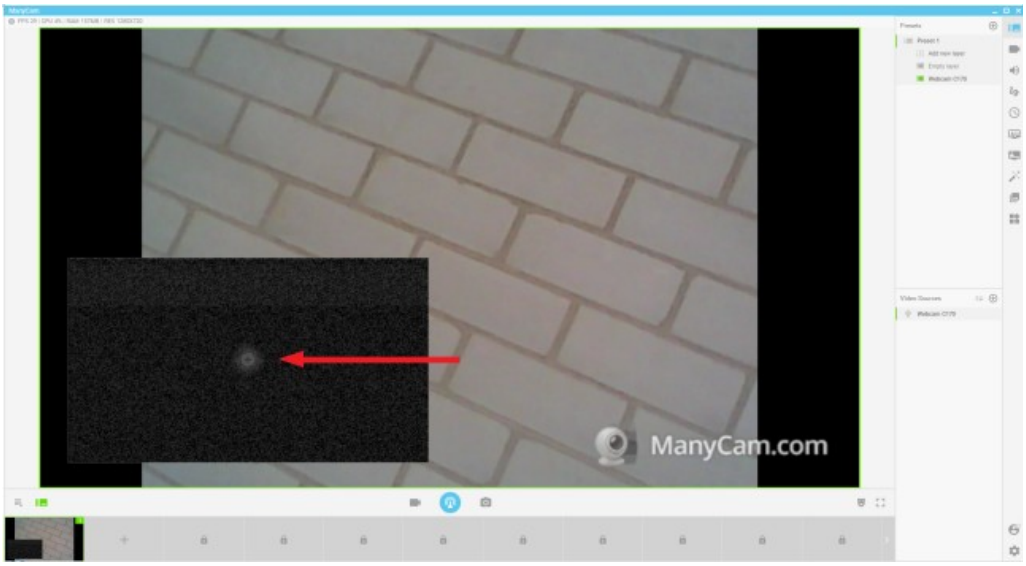


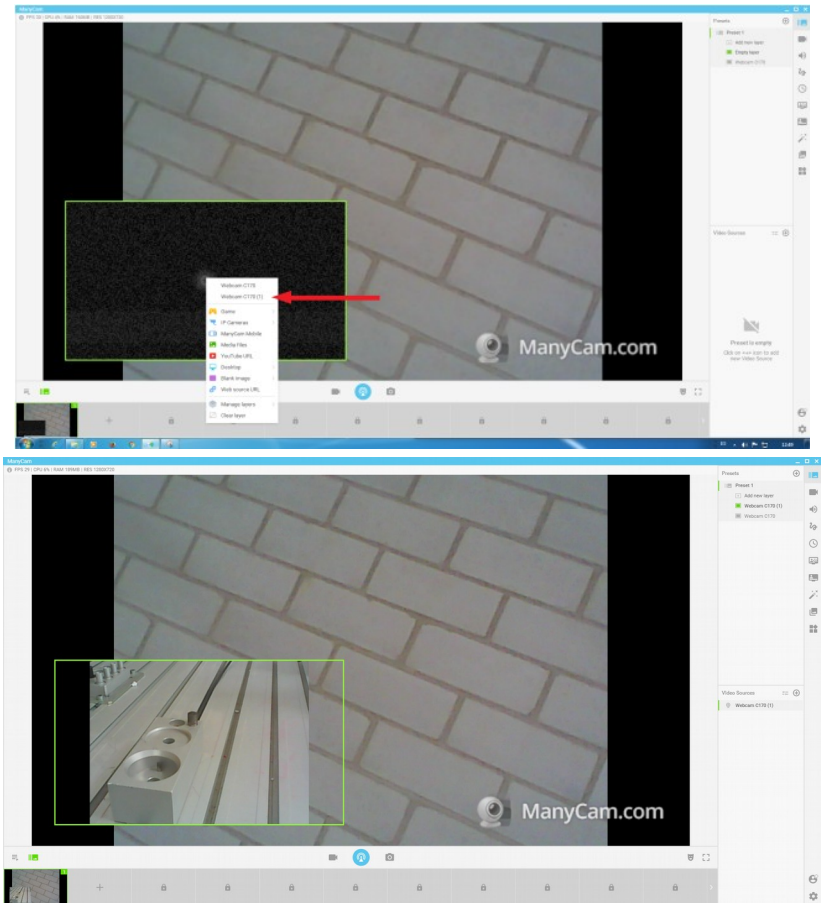


- **Preparar el layout de ManyCam per poder veure les dues càmeres simultàniament**
  Un cop instal·lar el software ManyCam cal realitzar la configuració pertinent per poder veure les imatges de les 2 càmeres. Cal seguir els següents passos

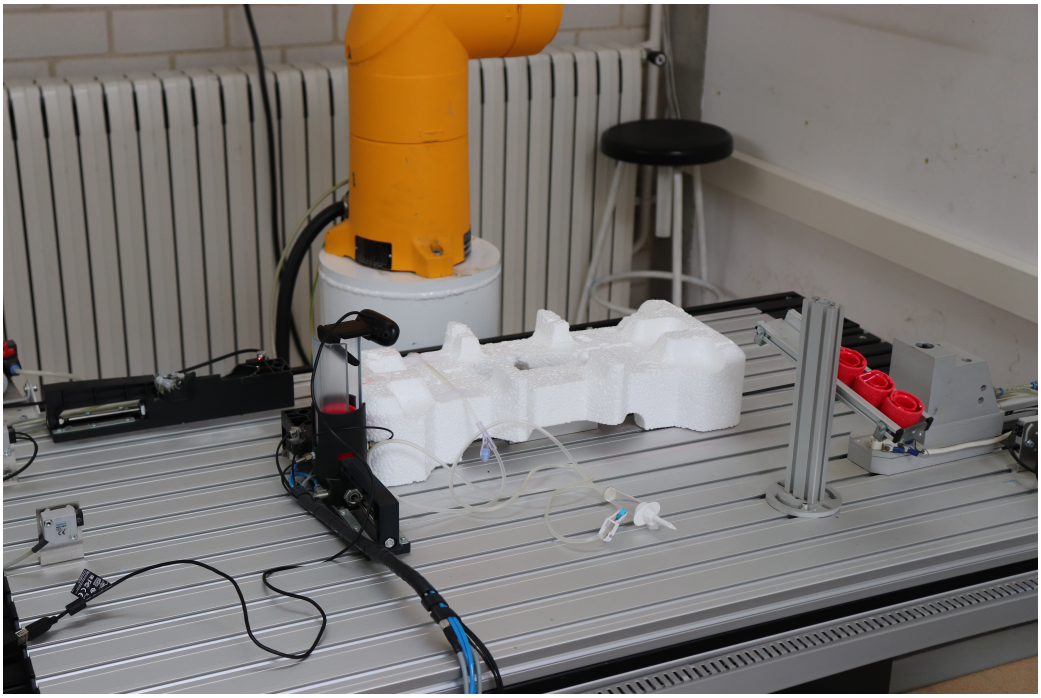




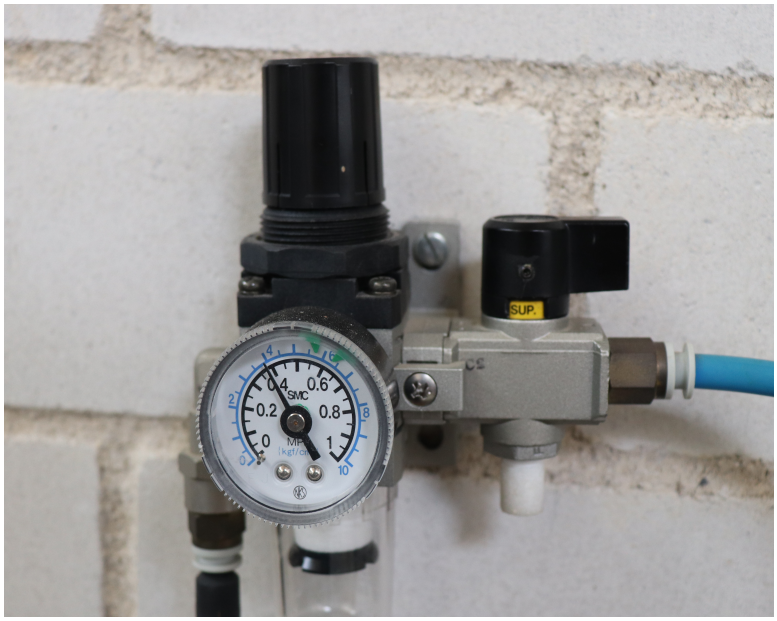




6. **Preparar la taula de treball (col·locar el material necessari i canviar l'eina si cal)**
   En aquesta etapa prepararem la taula de treball en funció de la simulació que es desitgi realitzar.



7. **Obrir l'aixeta de l'aire a pressió (possiblement calgui obrir les 2)**
   Les següents imatges mostren l'aixeta que acciona la pinça hidràulica del robot. Cal que es trobin en la mateixa posició que les imatges, ja que aquesta es la posició oberta.



8. **Engegar controladora CS8C**
   Simplement cal posar el boto en mode ON.

9. Situar la consola MCP fora de l'area de treball del robot

Com es pot aprecia a la imatge, la consola MCP es troba fora de l'àrea de treball del robot.

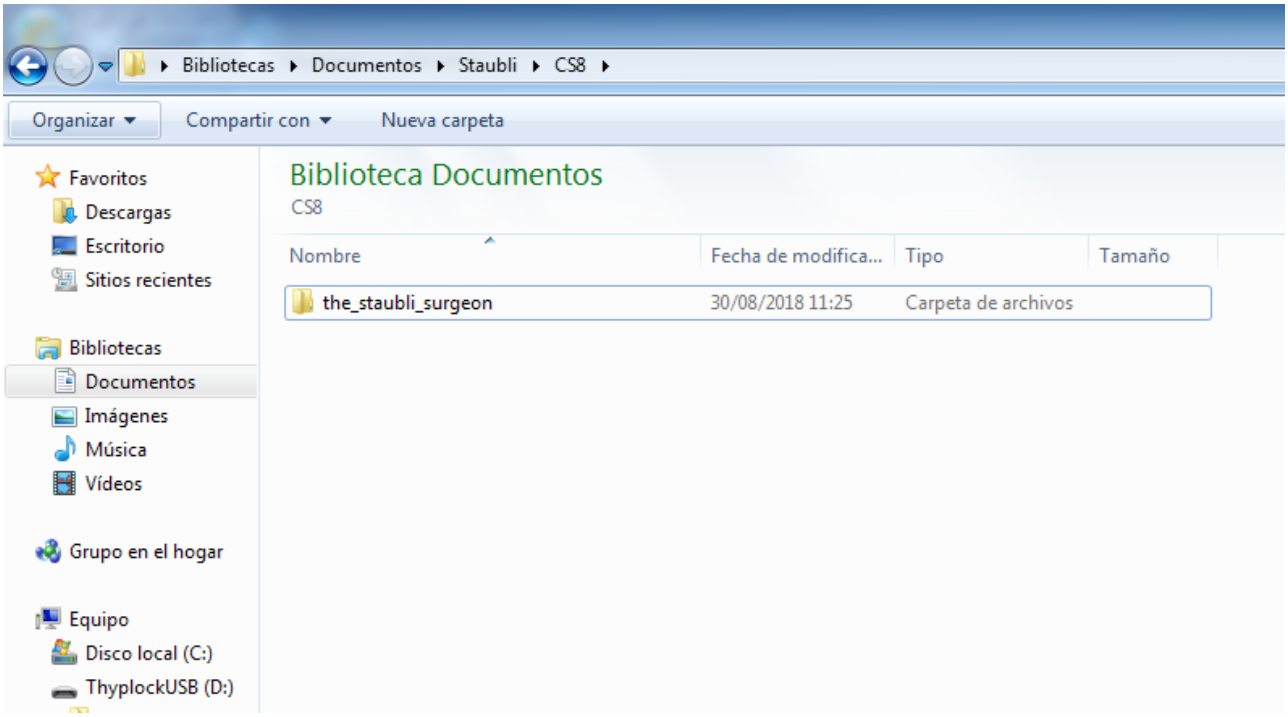
10. Activar les fotocèl·lules que permeten assegurar la zona de treball del robot

Aquest procediment es molt important, ja que representa una part molt vital en la seguretat del sistema. El procediment és el següent: verificar que no hi ha ningú dins la cel·la, mantenir premut 1 segon el boto negre, després prémer 1 segon l'altre boto negre i verificar que els leds indicadors de les fotocèl·lules es toven en estat ver (fotocèl·lula activa).

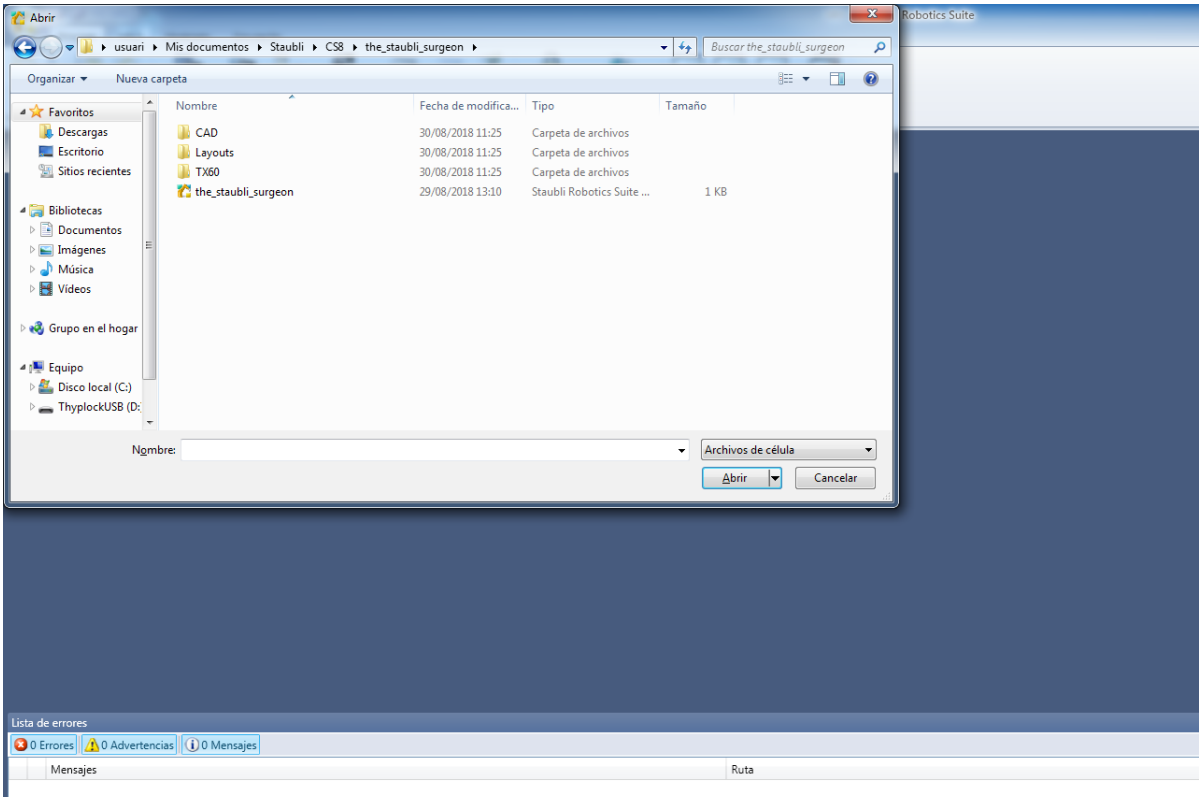
11. Obrir SRS per transferir l'aplicació a la CS8C

- Crear directori: **Documentos/Staubli/CS8/** (si no està creat)

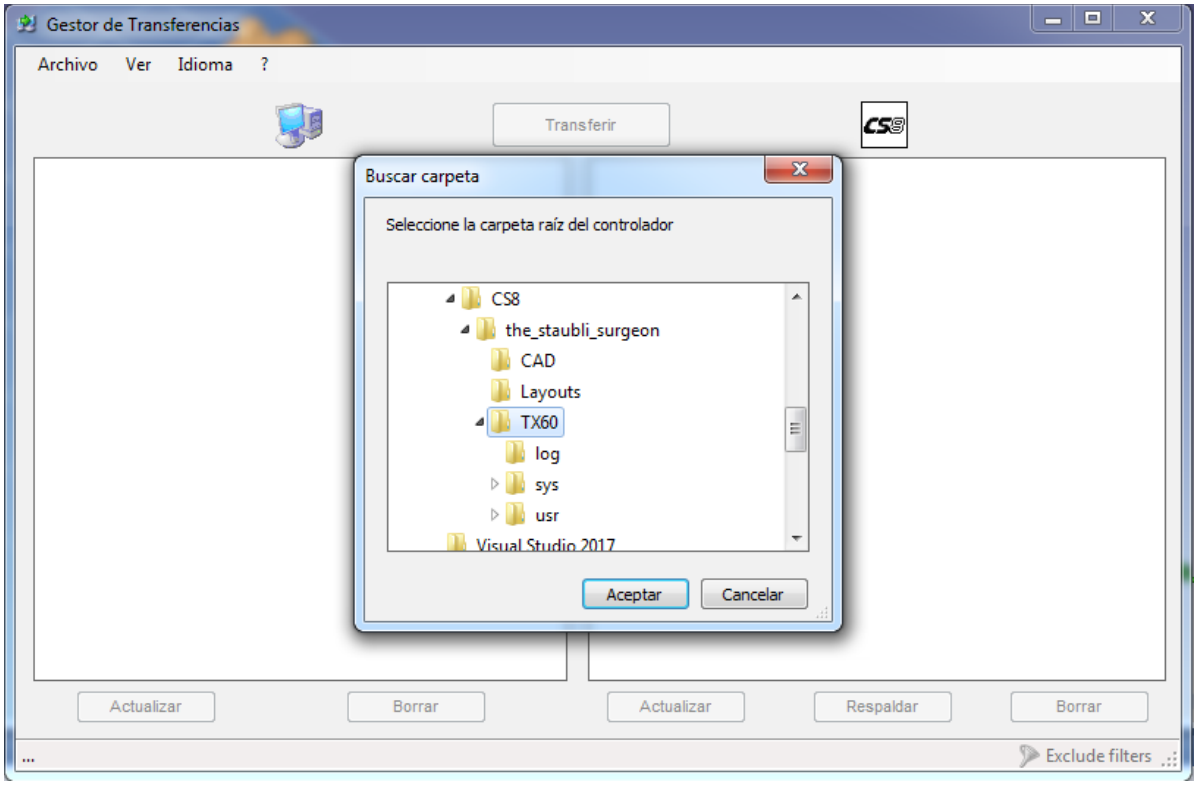
- Dins el directori anterior afegir l'aplicació **"the_staubli_surgeon"**



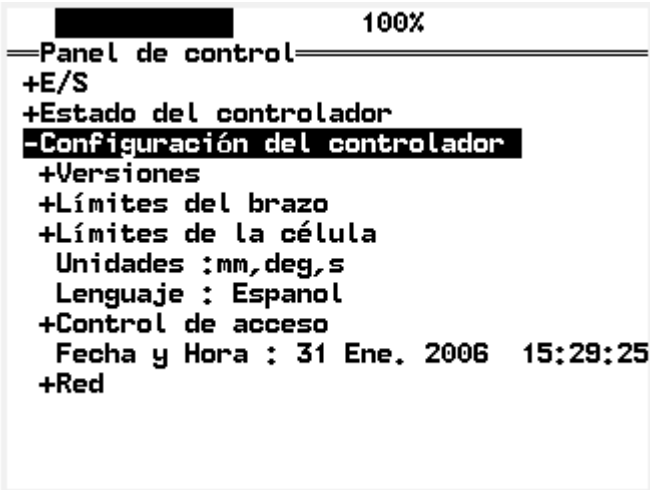
- A SRS "Abrir" i seleccionar aplicació **"the_staubli_surgeon"**

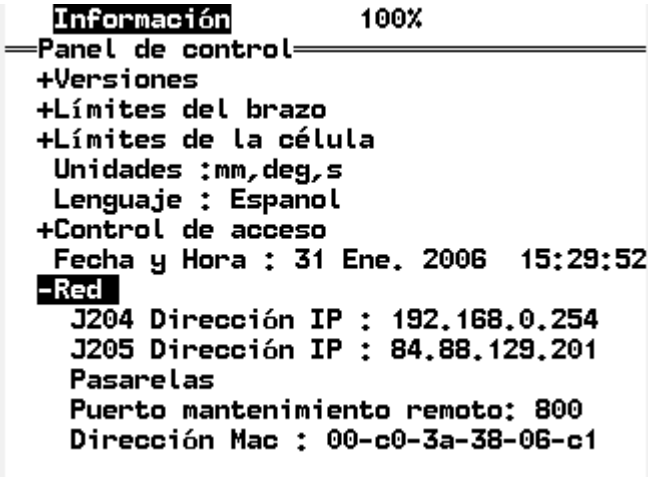


- Obrir el gestor de transferència

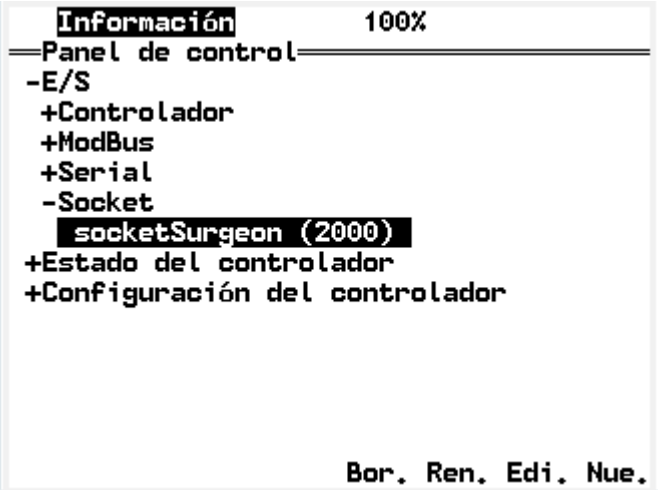


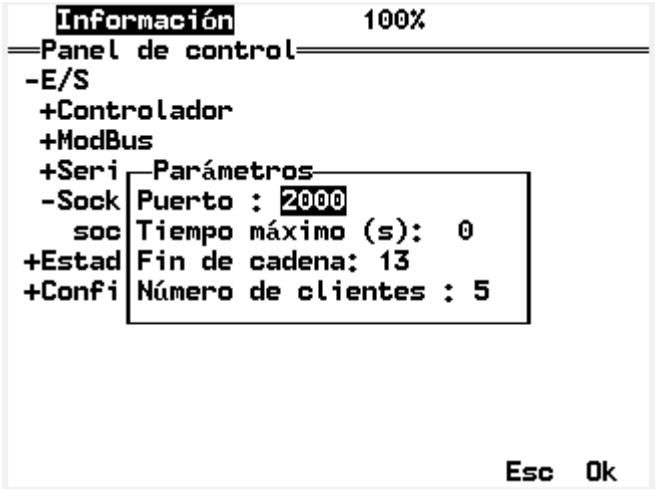
- Comprovar IP de la CS8 a la MCP.



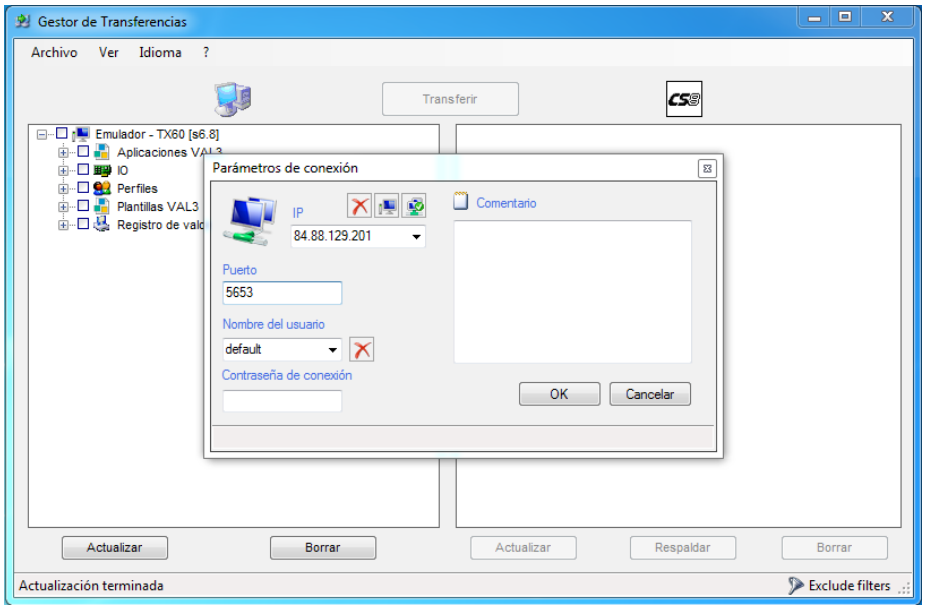


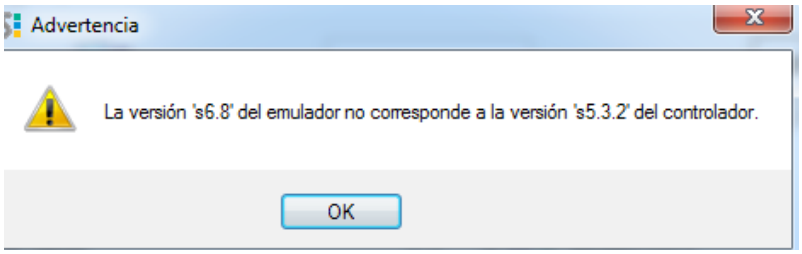
- Crear el socket a la consola MCP (només si no està creat)

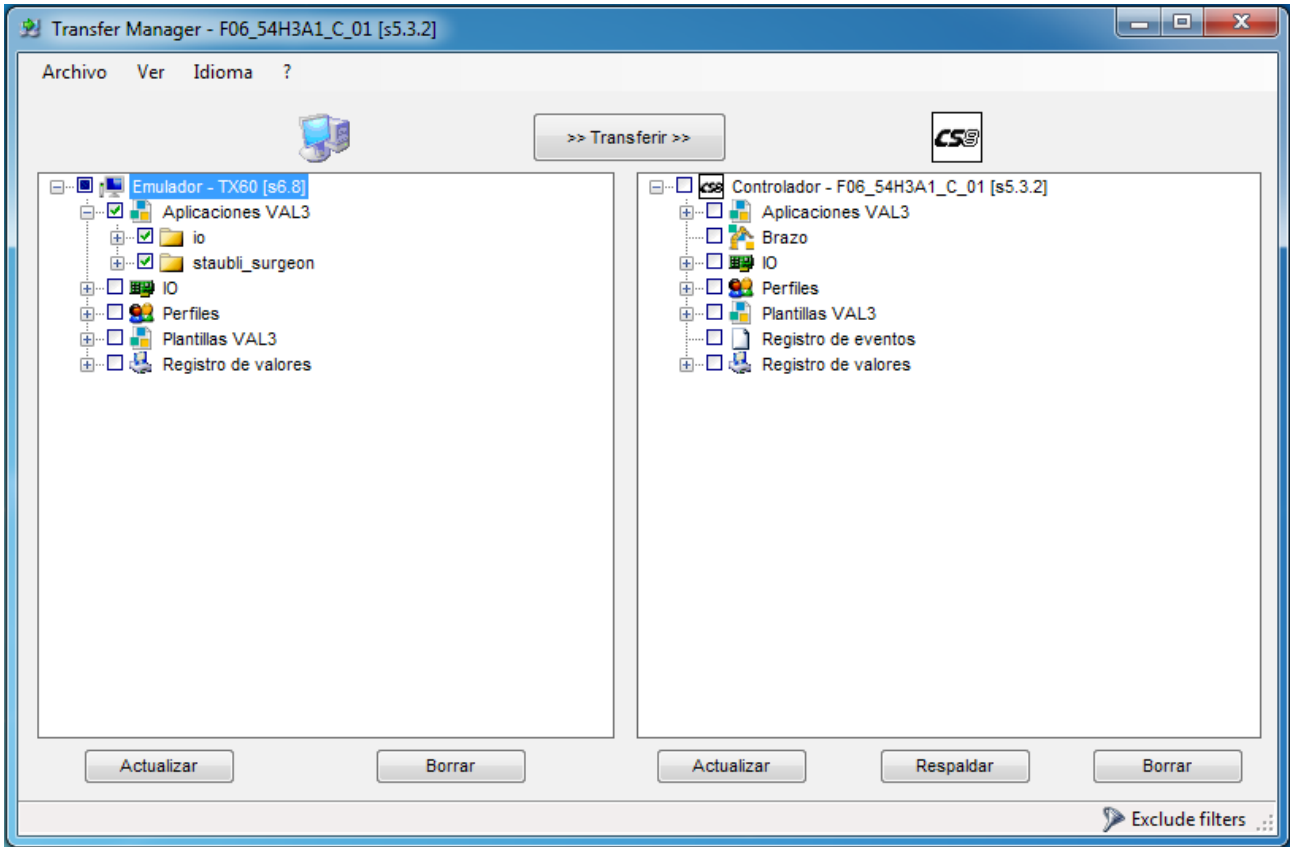


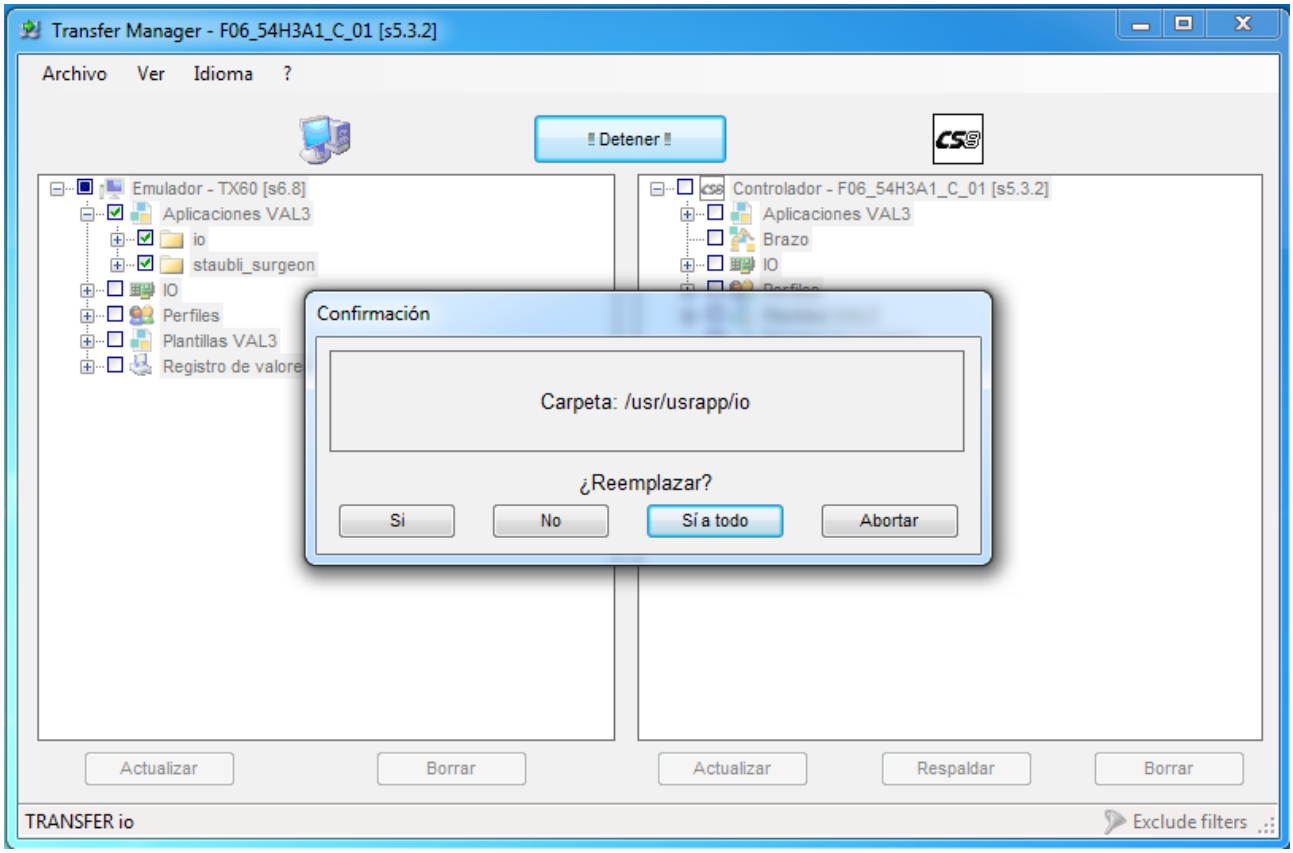


- Un cop anotada la IP (hauria de ser 84.88.129.201) procedir a realitzar la transferència seguint aquests passos:



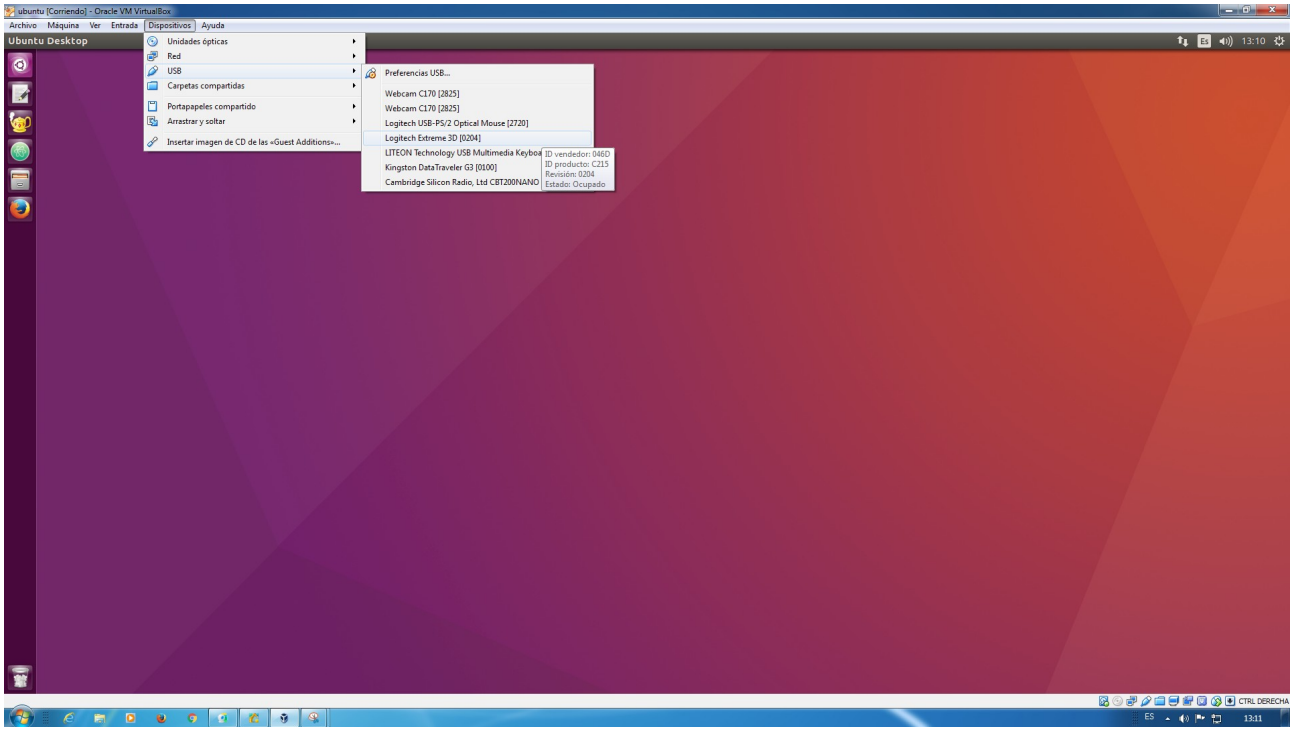



Transfer Manager - F06_54H3A1_C_01 [s5.3.2]
Archivo
Ver
Idioma
?
>> Transferir >>
Emulador - TX60 [s6.8]
Aplicaciones VAL3
io
staubli_surgeon
IO
Perfiles
Plantillas VAL3
Registro de valores
Controlador - F06_54H3A1_C_01 [s5.3.2]
Aplicaciones VAL3
Brazo
IO
Perfiles
Plantillas VAL3
Registro de eventos
Registro de valores
Actualizar
Borrar
Actualizar
Respaldar
Borrar
Exclude filters


Transfer Manager - F06_54H3A1_C_01 [s5.3.2]
Archivo
Ver
Idioma
?
!! Detener !!
Emulador - TX60 [s6.8]
Aplicaciones VAL3
io
staubli_surgeon
IO
Perfiles
Plantillas VAL3
Registro de valores
Controlador - F06_54H3A1_C_01 [s5.3.2]
Aplicaciones VAL3
Brazo
IO
Confirmación
Carpeta: /usr/usrapp/io
¿Reemplazar?
Si
No
Sí a todo
Abortar
Actualizar
Borrar
Actualizar
Respaldar
Borrar
TRANSFER io
Exclude filters

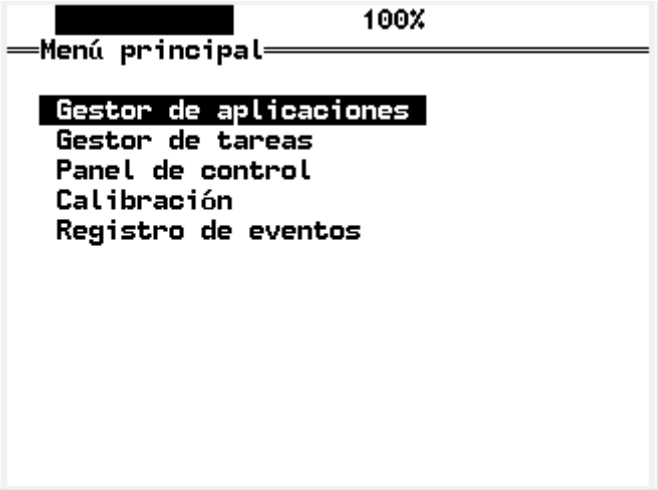
12. **A l'escriptori obrir la maquina virtual que es diu Ubuntu**

13. **Un cop el la maquina virtual hagi engegat, anar a la pestanya *Dispositivos* i seleccionar el *Logitech Extreme 3D Pro***

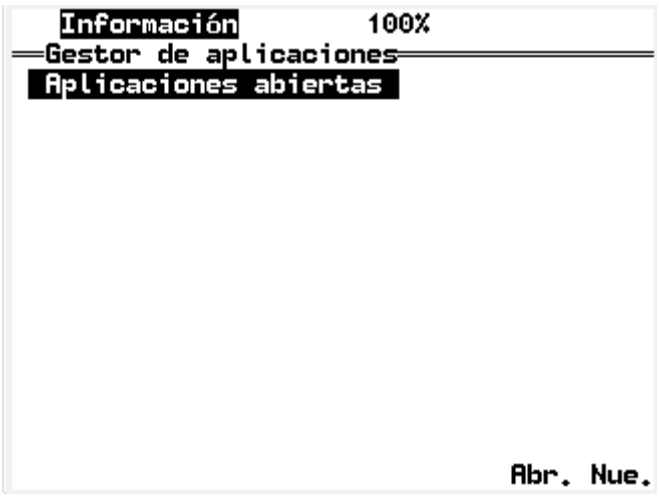


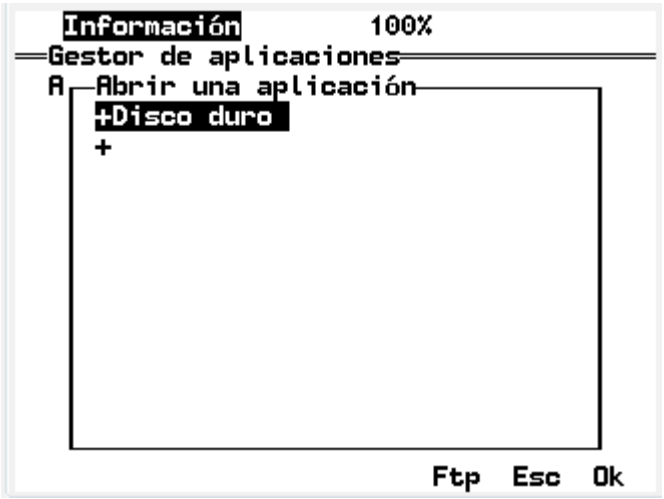
14. **Clonar el programa del git**

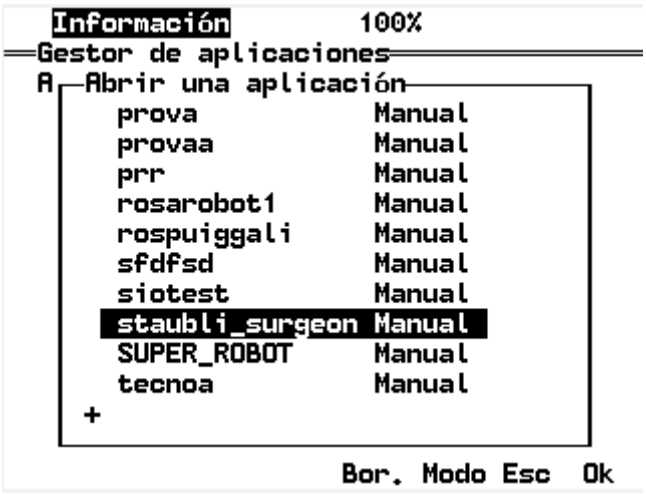
15. **Amb les fletxes amunt i avall ens mourem per el menu, i ens col·locarem sobre "Gestor de aplicaciones"**

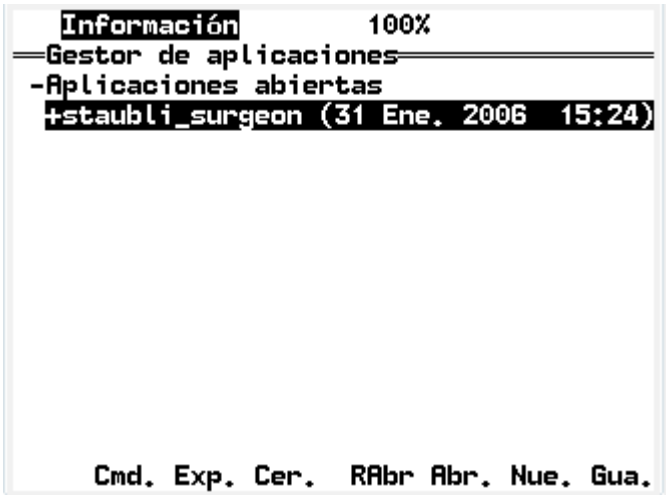


16. **Un cop dins el menú (entrem amb la fletxa dreta), obrim l'aplicació staubli-surgeon realitzant les següents tasques**

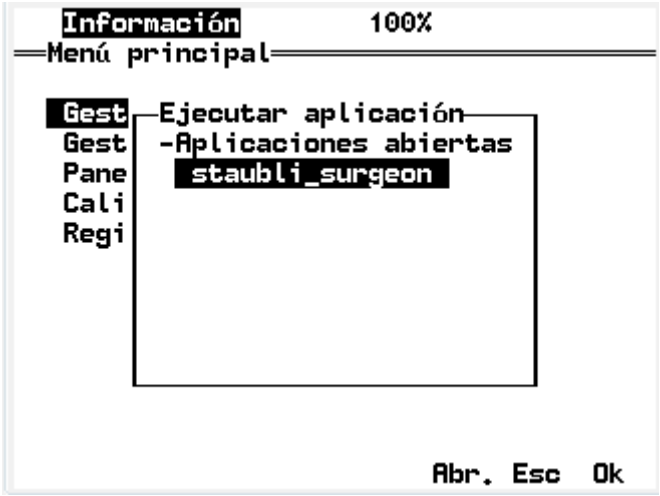


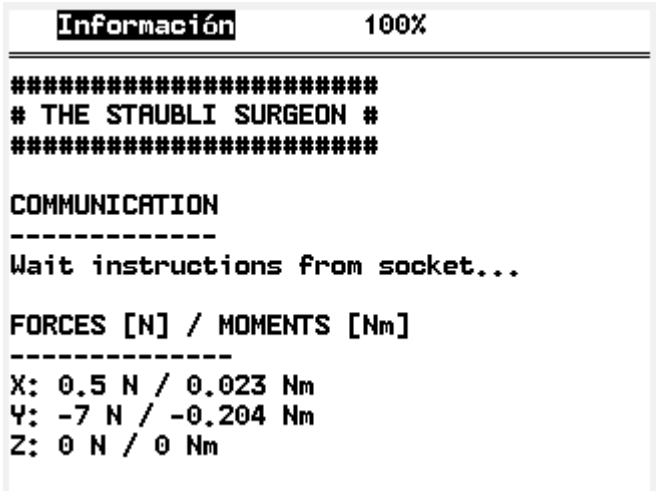




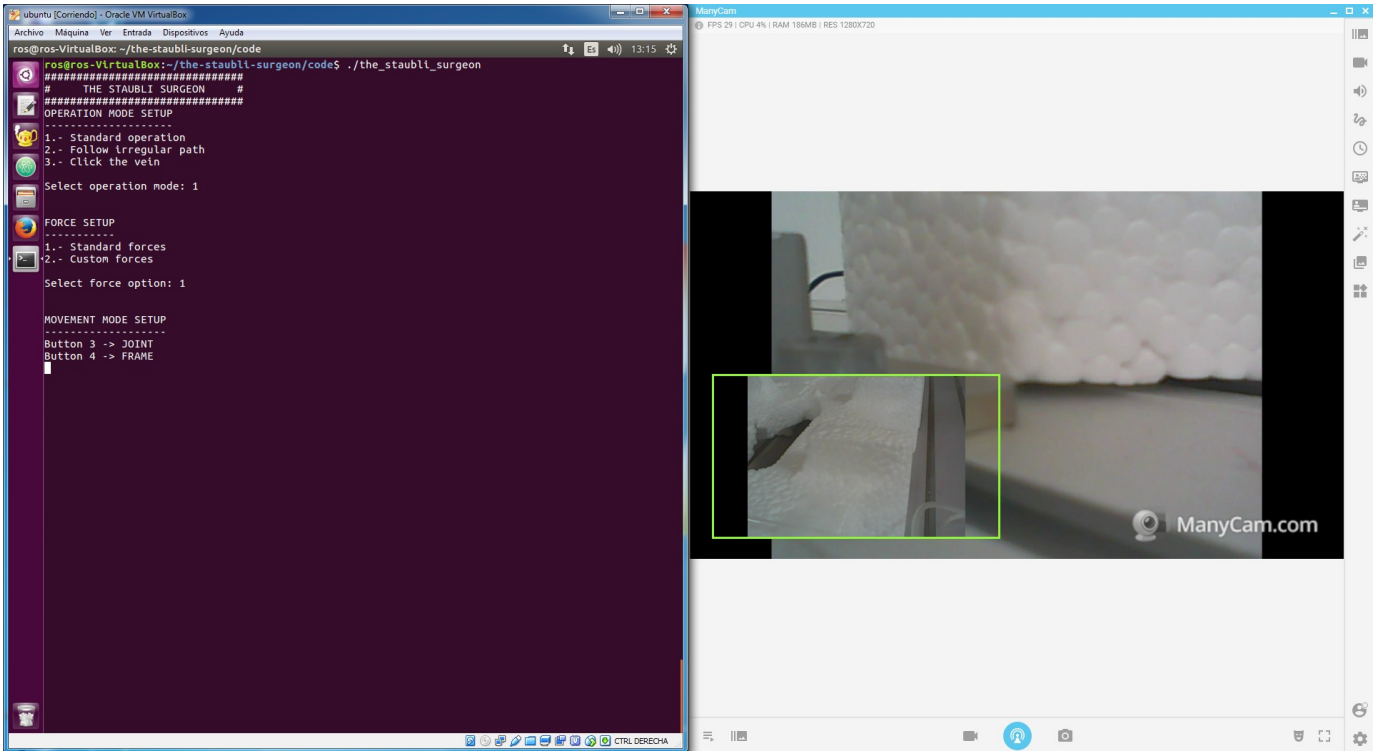


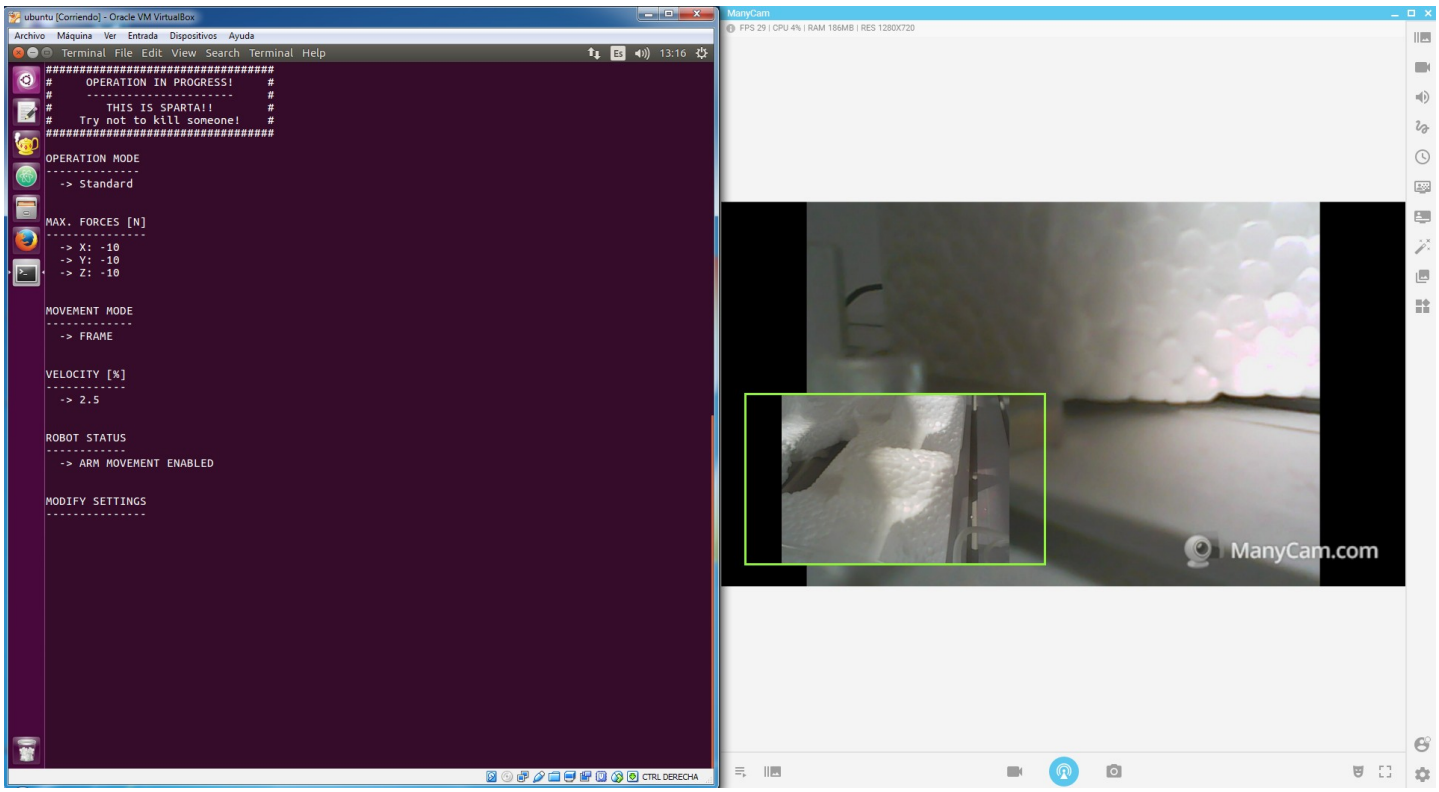
17. **Prémer el boto ver de Run.**
    - Un cop s'engega el progama, la consola ens mostrara la informació de les forces per pantalla.





18. **A ubuntu, obrir una terminal i executem el programa que inicia el sistema anomenat "the-staubli-surgeon"**
    - Arribats a aquest punt el sistema queda iniciat i cal seguir les indicacions del programa i de la pràctica





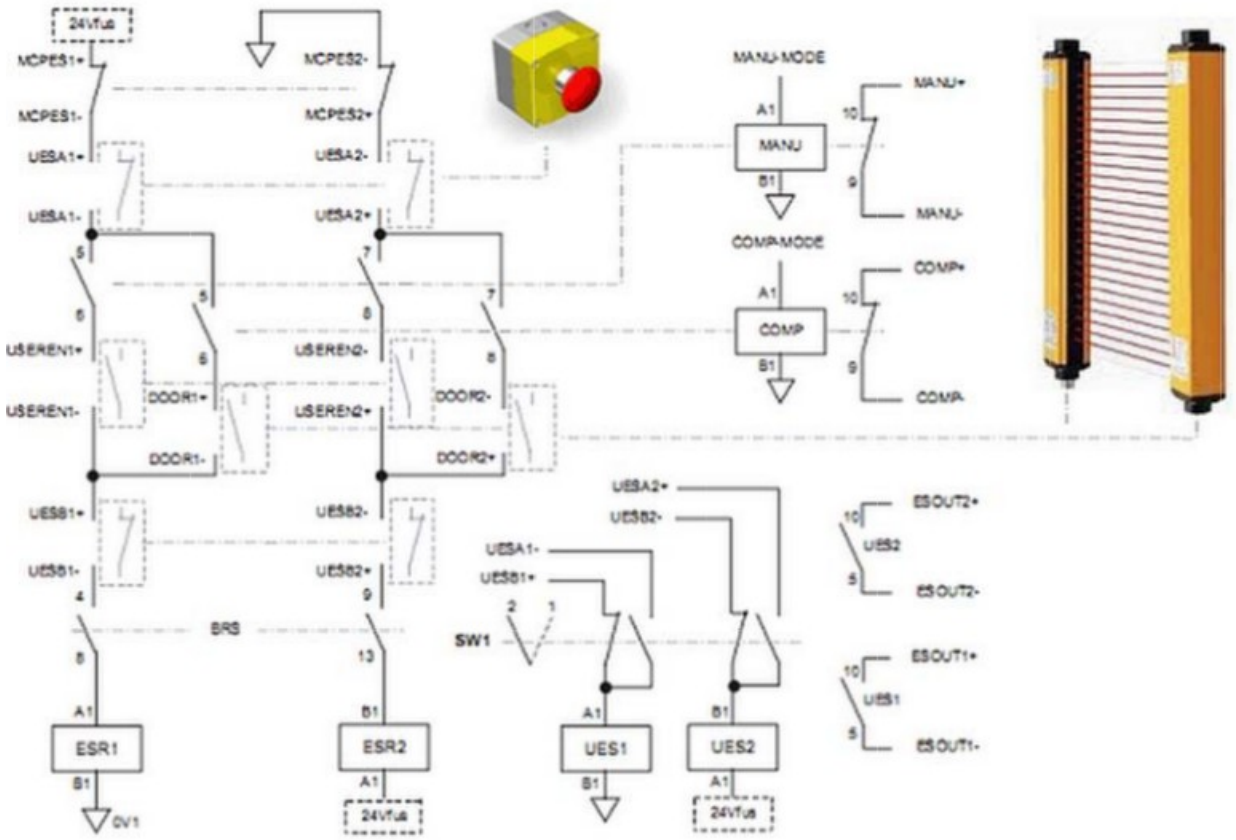
13.2.- ANNEX [B]: Mesures de seguretat

ATENCIÓ!!

El robot és una maquina de moviments rapids. Aquests moviments poden ésser perillosos. Abans de moure el robot, comproveu sempre que el seu entorn est+a en l'estat previst i assegureu-vos que cap persona no està dins el seu espai de treball.

Sistema de seguretat del robot

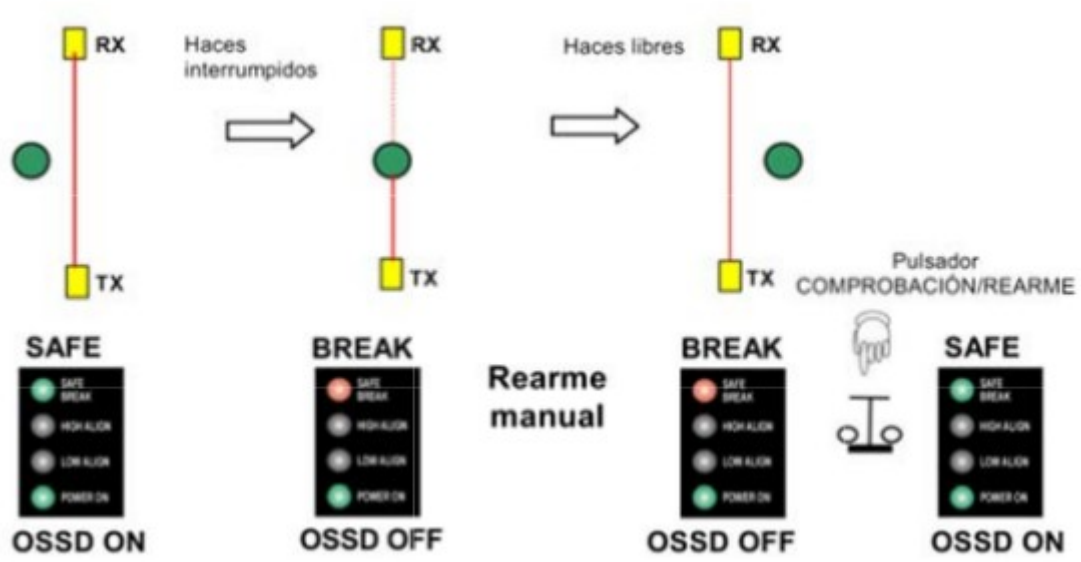
La i instal·lació del robot Stäubli en el laboratori disposa d'un interruptor de parada d'emergència i dues barreres de presencia.



L'interruptor de parada d'emergència fa el mateix efecte que l'interruptor d'emergència del comandament MCP. És a dir, és un interruptor de seguretat que només s'ha de prémer en cas de perill per una persona o per algun aparell. **No s'ha d'utilitzar per a parar el robot de manera habitual.**

Les barreres de presencia també actuen de manera similar, en detectar la presencia d'un cos entre cada parella de barreres emissora i receptora, provoquen una parada d'emergència. **Aquestes barreres només són operatives quan el robot està en mode automàtic,** és a dir, en mode manual podrem interrompre les barreres sense que aquestes generin una parada d'emergència.

Quan un objecte interromp els rajos infrarojos de les barreres, els contactes de seguretat s'obren (estat "BREAK"). En aquest estat, el robot no es podrà moure de manera automàtics i, l'entorn del robot queda sense alimentació per motius de seguretat. Per poder executar un programa en mode automàtic, les dues barreres han d'estar armades (estat "SAFE"). Per armar les barreres, s'ha de mantenir polsats els polsadors de rearmament durant al menys 0.5 segons.



13.3.- ANNEX [C]: Llenguatge VAL3

El llenguatge VAL3 és un llenguatge de programació d'alt nivell, flexible i interpretat (el codi font del programa passa a llenguatge màquina abans d'executar-se). Aquest llenguatge està expressament dissenyat per la programació de robots industrials de manipulació i d'assemblatge. El llenguatge VAL3 ha conservat els aspectes fonamentals de les possibilitats d'un llenguatge informàtic en temps real estàndard, integrant-se les funcionalitats específiques pel comandament d'un robot en una cèl·lula industrial. En les següents pàgines es farà un petit repàs a les principals prestacions que ofereix el VAL3.

- **Aplicacions**
  Una aplicació VAL3 és un software autònom de programació del robot i de les entrades/sortides vinculades a la controladora CS8C. Una aplicació VAL3 està constituïda per un conjunt de programes (instruccions VAL3 que s'han d'executar), per un conjunt de variables locals (dades de l'aplicació), per un conjunt de biblioteques (instruccions i dades externes per l'aplicació) i, quan una aplicació s'està executant, per un conjunt de tasques (programes en curs d'execució).

  Una aplicació VAL3 sempre conté els programes start() i stop(), un sistema de referència "world" (tipus frame) i una eina "flange" (tipus tool). En el moment de crear-se també conté les instruccions i les dades del model escollit.

  Les instruccions VAL3 no permeten manipular aplicacions. La càrrega, descàrrega, posada en marxa i aturada de les aplicacions es fa únicament a través de la interfície d'usuari del CS8C.

- **Programes**
  Un programa és una seqüència d'instruccions VAL3 per executar. Un programa està constituït per una seqüència d'instruccions (instruccions que s'han d'executar, per un conjunt de variables locals (dades internes del programa) i per un conjunt de paràmetres (dades facilitades al programa quan es crida). Els programes permeten agrupar seqüències d'instruccions susceptibles de ser utilitzades en diversos llocs en una aplicació i són reentrants, això vol dir que un programa es pot cridar a si mateix de manera recursiva (instrucció call), o ser cridat simultàniament per diverses tasques. Cada cop que es crida un programa té variables locals i paràmetres propis.

  El programa start() és el programa que es crida quan es posa en marxa una aplicació VAL3 i no pot tenir paràmetres. El programa stop() és el programa que es crida quan s'atura una aplicació VAL3 i no pot tampoc tenir paràmetres.

  Una biblioteca VAL3 és un software reutilitzable per una aplicació o altre biblioteca VAL3. Una biblioteca està constituïda per un conjunt de programes (instruccions VAL3 que s'han d'executar), per un conjunt de variables globals (dades de la biblioteca), per un conjunt de biblioteques (instruccions i dades externes utilitzades per la biblioteca) i, quan està executant-se, per un conjunt de tasques (programes propis de la biblioteca que s'està executant).

Totes les constants i variables VAL3 tenen un tipus, això permet un control inicial pel sistema en el moment d'editar un programa i, per tant, la detecció immediata de certs errors de programació. El VAL3 suporta els següents tipus senzills:

- tipus bool: per els valors booleans (cert/fals).
- tipus num: per els valors numèrics.
- tipus string: per les cadenes de caràcters.
- tipus dio: per les entrades/sortides tot o res.
- tipus aio: per les entrades/sortides numèriques (analògiques/digitals).
- tipus sio: per les entrades/sortides en connexió sèrie i socket Ethernet.

Un tipus estructurat és un tipus que reuneix diverses dades separades en camps. Els camps de tipus estructurats són accessibles individualment pel seu nom. El VAL3 suporta els següents tipus estructurats:

- tipus trsf: per les transformacions geomètriques cartesianes.
- tipus frame: per els plans geomètrics cartesians.
- tipus tool: per les eines ajustades en un robot.
- tipus point: per les posicions cartesianes d'una eina.
- tipus joint: per les posicions articulars del robot.
- tipus config: per les configuracions del robot.
- tipus mdesc: per els paràmetres de desplaçament del robot.

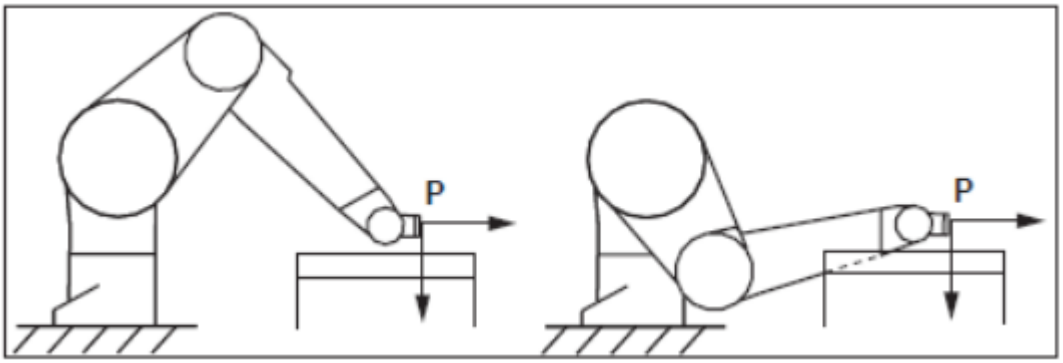
Una transformació (tipus trsf) descriu la posició i la orientació d'un sistema de referència cartesià respecte a una altre sistema de referència, els camps del qual presenten un ordre dins l'estructura trsf. Aquests camps s'expressen en mil·límetres o polsades (x, y, z) i en graus (rx, ry, rz). Les coordenades (x, y, z) són les coordenades cartesianes de l'origen del pla respecte al sistema que serveix de referència. Quan rx, ry i rz són nuls, ambdós plans tenen la mateixa orientació.

Un frame (tipus frame) defineix la posició de sistemes de referència a la cèl·lula. El sistema de referència d'una variable de tipus frame es defineix en el moment de la seva inicialització. El sistema de referència "world" està sempre definit en una aplicació VAL3 i tot pla està directament, o a través d'altres sistemes de referència, vinculat al sistema "world".

Un punt (tipus point) permet definir la posició i la orientació de l'eina del robot a la cèl·lula. Un tipus point és un tipus estructurat en camps que contenen la posició del punt en el seu sistema de referència (trsf) i la configuració del braç per arribar a la posició (config). El sistema de referència d'un punt és una variable tipus frame que es defineix al iniciar-la.

Un punt articular (tipus joint) defineix la posició angular de cada eix del robot, els camps del qual presenten un odre dins l'estructura joint. Aquests camps s'expressen en graus pels eixos de rotació i en mil·límetres o polsades pels eixos, lineals.

En general un robot té diverses possibilitats per arribar a una determinada, posició cartesiana, aquestes diferents possibilitats s'anomenen configuracions, (tipus config). És important especificar, entre totes les configuracions possibles, les, que són vàlides i les que es volen prohibir, ja que el robot podria col·lisionar amb, l'entorn. Per resoldre aquest problema, el tipus point permet especificar les, configuracions admeses per robot gràcies al seu camp de tipus config. A la figura següent, es representen 2 configuracions diferents per un mateix punt.



- **Constants i variables**
  Una constant és una dada definida directament en un programa VAL3, sense declaració prèvia. Una constant té un tipus que ve determinat implícitament pel sistema.

  Una variable és una dada a la qual es fa referència en un programa pel seu nom. L'abast d'una variable pot ser global (tots els programes de l'aplicació poden utilitzar la variable) o local (la variable és accessible únicament des del programa en el que està declarada).

  Quan es passa un paràmetre per valor, el sistema crea una variable local i la inicialitza amb el valor de la variable o de l'expressió subministrada pel programa sol·licitant. Quan es passa un paràmetre per referència, el programa deixa de treballar sobre una còpia de la dada passada pel sol·licitant, per fer-ho directament sobre la pròpia dada, a la que senzillament es canvia de nom localment.

- **Tasques**
  Una tasca és un programa que s'està executant. En una aplicació es trobarà, una tasca pels desplaçaments del braç, una tasca autòmat, una tasca per la, interfície d'usuari, una tasca pel seguiment dels senyals de seguretat, tasques de, comunicació, etc. Una tasca es caracteritza per un nom (identificador únic de tasca, dins la biblioteca o aplicació), per una prioritat (paràmetre per la seqüenciació de, les tasques), per un programa (punt d'entrada i punt de sortida de la tasca), per un estat (actiu o aturat) i per la pròpia instrucció a executar.

- **Instruccions**
  Les instruccions del llenguatge VAL3 es poden dividir en quatre grups. Les instruccions de control de seqüència, les instruccions de moviment, les instruccions de control del robot i les instruccions de la pàgina de l'usuari. A continuació farem un petit resum de les instruccions utilitzades en aquest projecte.

  - **void userPage ():** fa que es presenti la pàgina de l'usuari a la pantalla de la MCP.

  - **void cls ():** esborra la pàgina d'usuari i col·loca el cursor a (0, 0).

  - **void sioLink(<sio &variable>, <sio origen>):** connecta variable a l'entrada/sortida del sistema a l'origen del qual està enllaçada. Es genera un error d'execució si origen és una entrada/sortida protegida en el sistema.

  - **num sioSet(<sio sortida>, <num valor>):** assigna valor a sortida. Es genera un error d'execució si sortida no està enllaçada a una entrada/sortida del sistema.

  - **num sioGet(<sio entrada>):** retorna el valor numèric d'entrada. Es genera un error d'execució si entrada no està enllaçada a una entrada/sortida del sistema.

  - **void movej(<joint joint> o <point punt>, <tool eina>, <mdesc desc>):** registra una ordre de moviment articular fins a la posició joint o punt, amb l'eina i els paràmetres de moviment desc. Es genera un error d'execució si desc té valors no vàlids, si posició està fora de l'abast del software, si punt és inaccessible, o si una comanda de moviment anterior registrat no ha pogut ser executada (destí fora d'abast).

  - **void movel(<point punt>, <tool eina>, <mdesc desc>):** registra una ordre de moviment lineal fins la posició punt, amb l'eina i els paràmetres de moviment desc. Es genera un error d'execució si desc té valors no vàlids, si punt és, inaccessible, si el moviment cap al punt és impossible en línia recta, o si una, comanda de moviment anterior registrat no ha pogut ser executada (destí, fora d'abast).

  - **point here(<tool eina>, <frame marca de referència>):** retorna la posició, actual de l'eina a marca de referència (posició ordenada i no posició, mesurada). El sistema de referència del punt retornat és marca de referència., La configuració del punt retornat és la configuració en curs del braç.

  - **joint herej():** retorna la posició actual del camp braç. El valor retornat, correspon a la posició enviada als amplificadors per la controladora, i no a la, posició llegida en els codificadors de l'eix. La posició del camp de la, controladora es reactualitza cada 4 ms.

- **bool pointToJoint(<tool eina>, <joint inicial>, <point posició>, <joint &coordenades>):** calcula les coordenades articulars que corresponen a la posició especificada. Retorna cert si s'han trobat unes coordenades articulars, fals si no existeix una solució. La posició articular cercada respecta la configuració de posició. Els camps de valor "free" no imposen la configuració. Pels eixos que poden fer més d'una volta hi ha diverses solucions articulars que tenen exactament la mateixa configuració, en aquest cas s'adopta la solució més propera a la inicial. No hi pot haver una solució si posició està fora de l'abast o fora dels marges del software. Si posició especifica una configuració, pot estar fora dels marges per aquesta configuració, però dins dels marges per una altra configuració. Es genera un error d'execució si posició no té sistema de referència definit.

- **bool isInRange(<joint jposició>):** prova si la posició d'un camp s'ajusta als, límits de software de camp de braç. Quan el braç està fora dels límits del software de camp (després d'una operació de moviment), el braç ja no pot maniobrar-se per mitja de l'aplicació VAL3, només són possibles les maniobres manuals (amb direccions de moviment limitades).

- **void resetMotion():** atura el braç a la trajectòria i anul·la totes les ordres de moviment registrades. L'autorització de moviment programat es pot restaurar si havia estat suspesa per la instrucció stopMove(). Si no s'especifica cap posició articular de sortida, la propera ordre de moviment s'efectuarà a partir de la posició en curs del braç, fos la que fos.

- **void waitEndMove():** anul·la l'allisat de l'última ordre de moviment registrada i espera a que aquesta s'hagi executat. Aquesta instrucció no espera que el robot s'estabilitzi a la seva posició final, únicament que la consigna de la posició enviada als variadors correspongui amb la posició final desitjada. Quan l'espera d'estabilització completa del moviment és necessària, s'ha d'utilitzar la instrucció isSettled(). Es genera un error d'execució si una ordre de moviment anteriorment registrada no pot ser executada (destí fora d'abast).

13.4.- ANNEX [D]: Codi font

Els arxius que conformen el codi font d'aquest projecte els podem separar en dos, els arxius desenvolupats en C++ i els arxius desenvolupats en VAL3. Tot el codi desenvolupat en aquest projecte es troba en el CD adjunt a la memòria.

Aquest document no incorpora el codi font del sistema desenvolupat, el motiu principal recau en l'extensió d'aquest. No obstant, podem trobar el projecte de VAL3, el codi C++ i altres documents al repositori GitHhub: https://github.com/BrunoAltadill/staubli-surgeon