Títol: **Interfície pel comandament teleoperat d'útils quirúrgics**

Volum: **I**

Alumne: **Marc Castellarnau Jordà**

Director/Ponent: **Alícia Casals**

Departament: **ESAII**

Data:13-01-2011

DADES DEL PROJECTE

Títol del Projecte: Interfície pel comandament teleoperat d'útils quirúrgics

Nom de l'estudiant: Marc Castellarnau Jordà
Titulació: Enginyeria Informàtica
Crèdits: 37.5
Director/Ponent: Alícia Casals Gelpí
Departament: ESAII

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Manel Frigola Bourlon

Vocal: Mario Martín Muñoz

Secretari: Alícia Casals Gelpí

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

AGRAÏMENTS

Primer de tot m'agradaria donar les gràcies als meus pares per preocupar-se de tot per tal que jo només hagués d'ocupar-me de realitzar el projecte de final de carrera.

Voldria donar les gràcies també a la directora del projecte, Alícia Casals, que sempre ha estat disponible i disposada a ajudar-me quan l'he necessitada. Agrair també l'ajuda proporcionada pel professor Josep Amat durant els inicis del projecte i durant l'estudi de les diferents classes de sensors.

Per últim desitjo també donar les gràcies a l'Ezio Cappellino per l'ajuda i els consells que m'ha donat a l'hora de realitzar el prototip i per haver estat disponible sempre que he tingut qualsevol problema.

CONTINGUT

1 MOTIVACIÓ

Actualment la robòtica és un camp que està evolucionant de forma molt significativa, ja que ofereix un gran ventall de possibilitats i de noves maneres d'afrontar tant problemes senzills, com d'altres de molt complexes. Per aquest motiu, el mercat de la robòtica ha augmentat molt en els darrers anys i fa que sigui un terreny amb grans sortides comercials i, per tant, també professionals.

Avui en dia, la majoria de robots que s'utilitzen es troben en el món de la indústria i realitzen tasques simples i repetitives, però cada vegada més està creixent la investigació en aquesta àrea amb l'objectiu d'assolir tasques més complexes. Aquest projecte n'és un exemple, ja que s'estudia l'ús de la robòtica per realitzar intervencions quirúrgiques que poden millorar-se gracies a la teleoperació assistida, per exemple l'ús del canvi d'escala en microcirurgia.

És important comentar que, una altra de les raons principals que han dut a realitzar aquest projecte de final de carrera, ha estat la curiositat personal en aquesta disciplina.

2 INTRODUCCIÓ

Existeix un model de comunicació, anomenat mestre/esclau, que es basa en que un dispositiu/procés, anomenat mestre, sempre tingui control sobre un altre dispositiu/procés, denominat l'esclau.

En intervencions quirúrgiques teleoperades (realitzades de forma remota) s'utilitza aquest model de comunicació. En elles, el mestre és un dispositiu capaç de captar les ordres de l'usuari per possibilitar la posterior transmissió d'aquestes a l'esclau (ex: braç robòtic) perquè les executi. Per això, cal un dispositiu per mesurar els moviments de l'operador de forma adequada.

El fet d'utilitzar aquesta configuració permet a la robòtica ajudar molt en alguns aspectes de les intervencions quirúrgiques al aprofitar la sinèrgia persona/robot. Tot i que les intervencions encara no es puguin automatitzar completament, en alguns aspectes l'ús de la robòtica pot representar un avantatge respecte la cirurgia convencional (ex: filtrar tremolors, escalar la senyal per realitzar moviments mil·limètrics,...).

Aquest projecte tracta doncs d'estudiar les limitacions (restriccions en els moviments, precisió,...) que posen els diferents dispositius que poden utilitzar-se com a mestre, i proposar un disseny segur, usable i eficient per utilitzar en intervencions quirúrgiques teleoperades.

2.1 Objectius

L'objectiu d'aquest projecte de final de carrera consisteix en el disseny d'una interfície capaç de sensoritzar els moviments d'un instrument quirúrgic durant una intervenció laparoscòpica. Aquest sistema permetrà comandar l'element terminal d'un robot en teleoperació.

Per poder aconseguir l'èxit en aquest projecte es pretén assolir els següents subobjectius:

- Realització del disseny de la interfície

    - Estudi dels antecedents d'ús de la robòtica amb especial èmfasi en el camp de la medicina.
    - Estudi dels requisits del sistema a dissenyar.
    - Estudi dels diferents tipus de sensors.
    - Estudi de les diferents possibilitats d'integració de cada tipus de sensor.
    - Avaluació dels avantatges i inconvenients de l'ús de les diferents classes de sensors per tal d'aconseguir dissenyar el sistema més adequat possible (eficient, usable, precís, segur, ...).

- Desenvolupament d'un prototip de l'estació de treball:

    - Disseny d'un controlador per a la lectura de la interfície.
    - Desenvolupament del programa de control del robot.
    - Avaluació de l'estació de treball experimental.

El pes del projecte recau en la part del disseny de la interfície, però s'ha decidit desenvolupar el prototip per acabar de completar-lo.

Com s'ha comentat en els objectius, existeixen diversos tipus de sensors que es poden utilitzar en el disseny del dispositiu mestre que haurà de sensoritzar els 4 graus de llibertat de l'instrument laparoscòpic i, que interessen per realitzar una intervenció quirúrgica d'aquesta classe (x, y, z, Θ). D'aquests tipus, en aquest treball, s'ha decidit utilitzar els sensors de flux magnètic per diferents motius, com són: la seva alta precisió i la possibilitat d'utilitzar-los sense el requisit d'utilitzar dispositius més aparatosos que condicionin o limitin els moviments del seu usuari. Aquest fet, que seria incòmode i poc pràctic a l'hora de realitzar un procediment tan delicat i precís com és una intervenció

quirúrgica, ha contribuït en la decisió d'integrar el sensor en un mànec permetent a l'usuari tractar-lo com si fos l'instrument laparoscòpic.

2.2 Estructura de la memòria

La memòria d'aquest projecte s'ha estructurat en diverses parts, tal com es descriu a continuació:

- En el primer apartat s'explica la motivació que ha fet que es portés a terme aquest treball.
- En el segon apartat es realitza una introducció al projecte, es descriuen els seus objectius principals i s'explica l'estructura de la seva memòria.
- En el tercer apartat, antecedents, s'explica el panorama general de la robòtica en l'actualitat. Es fa especial èmfasi a l'ús de la robòtica dins el camp de la medicina i s'exposen els principals problemes i les solucions que s'hi presenten.
- A l'apartat 4, plantejament del projecte, s'explica com es pretén abordar aquest treball de final de carrera.
- A l'apartat 5, interfície cirurgià sistema, s'expliquen les necessitats cinemàtiques requerides per aquesta. S'analitzen els graus de llibertat que es volen sensoritzar i els diversos tipus de sensors que es troben al mercat que permeten fer-ho. D'aquests últims es comenten els diferents avantatges i inconvenients que presenten.

  També s'estudien les diferents possibilitats d'integració de cada tipus de sensor per tal d'escollir aquell conjunt sensor/mètode d'integració que presenti un millor resultat a l'hora d'aconseguir les funcionalitats necessàries per complir els objectius proposats.
- A l'apartat 6, prototip, es tracten tots els temes relacionats amb el prototip desenvolupat. S'hi diferencien tres apartats, en el primer (hardware), es

mostra una visió dels components físics del sistema (electrònics, mecànics, electromecànics,...) i s'estudia la seva comunicació. En el segon apartat (software), s'expliquen les diferents aplicacions que formen part del prototip desenvolupat i els diferents paràmetres configurables que permeten treure un major ús del sistema, fent especial èmfasi en la seguretat. En l'últim apartat, avaluació de l'estació de treball experimental, s'estudia el prototip de laboratori construït i es valora el seu funcionament.

- A l'apartat 7 s'exposen les conclusions extretes de la realització del projecte.
- A l'apartat 8 es llista la bibliografia consultada per dur a terme aquest treball
- A l'últim apartat, annexos, s'hi amplia la informació proporcionada en aquesta memòria.

3 ANTECEDENTS

La paraula robòtica fa referència a la ciència o branca de la tecnologia que estudia el disseny i la construcció de màquines capaces de dur a terme tasques realitzades pel ser humà o que requereixen l'ús d'intel·ligència. Per tant, es pot definir la robòtica com la disciplina que aglutina el conjunt de coneixements teòrics i pràctics que permeten construir sistemes automatitzats, basats típicament en estructures mecàniques articulades, dotats de certa intel·ligència i destinats a substituir l'home en certes tasques.

A continuació s'esmenten diferents fets, al llarg de l'historia de la robòtica, per tal d'il·lustrar l'evolució d'aquesta branca de la tecnologia al llarg dels anys.

La primera referència al terme robot data de l'any 1921 quan l'escriptor txec Karel Capek va utilitzar aquest mot en la seva obra "Rossum's Universal Robots" per referir-se a un autòmat. Va extreure aquest terme a partir de la paraula txeca "robota" que significa servitud o treballs forçats. Posteriorment, Isaac Asimov, va definir la robòtica com la ciència que estudia els robots.

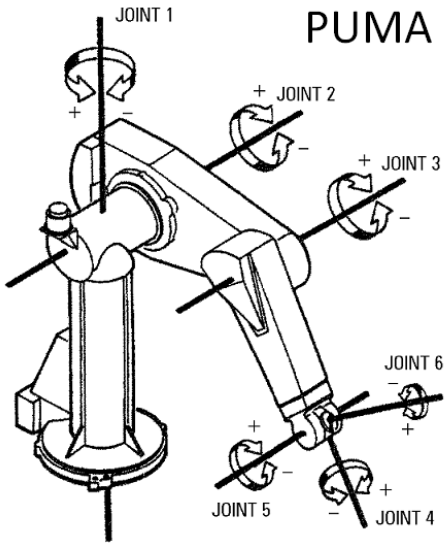
Des dels primers autòmats fins a les sondes espacials han passat diversos segles però si parlem d'intel·ligència només podem remuntar-nos uns trenta anys enrere.
Aquests darrers anys, conseqüentment al gran interès que ha despertat la robòtica pel gran ventall de possibilitats que ofereix, el desenvolupament d'aquesta disciplina ha estat enorme.

Al segle XX els automatismes comencen a ser utilitzats amb finalitats pràctiques. Durant la segona guerra mundial els científics inicien la construcció dels anomenats teleoperadors (manipuladors de control remot), que tot i lluny de ser robots, estaven basats en dispositius mecànics capaços de reproduir les accions realitzades per un operador situat a distància i s'utilitzaven per manipular elements radioactius.

A principis dels anys 60 es pot començar a parlar de robòtica quan l'empresa Unimation, fundada per George C. Devol i Joseph F. Engelberger (considerats l'"avi" i el "pare" de la

robòtica industrial), havia dissenyat i produït molta maquinària automàtica de tot tipus. Amb l'aparició dels microxips van poder resoldre els problemes amb els computadors utilitzats per controlar els robots i, l'empresa Unimation, es va convertir en una de les més rentables de tot el món.

Kawasaki Heavy Industries va realitzar una llicència amb Unimation l'any 1970 per desenvolupar el primer robot articulat proveït d'una càmera i controlat per un ordinador. Aquest era capaç d'apilar blocs de forma intel·ligent seguint una estratègia traçada prèviament. Va ser anomenat "braç Stanford".
El gran èxit que va tenir va afavorir el fet de considerar el control per computador i la sensorització com a parts fonamentals de la robòtica.



L'any 1973, l'institut d'investigació d'Stanford va desenvolupar el primer llenguatge de programació per a robots anomenat WAVE. Aquest va ser seguit, el 1974, pel llenguatge AL que, al ser fusionat amb el primer, va donar lloc al llenguatge comercial VAL.
El 1976 la NASA va fer ús del primer braç robòtic a l'espai.
L'any 1978 es va introduir el robot PUMA ("Programmable Universal Machine for Assembly") (figura 1) en tasques de muntatge.

Figura 1: Braç robòtic Puma (1978)

Al 1979 es desenvolupa el robot per al muntatge anomenat SCARA ("Selective Compliance Arm for Robòtic Assembly") (figura 2) a la universitat de Yamanashi al Japó.
Aquest mateix any es funda la Federació Internacional de Robòtica amb sèu a Estocolm.

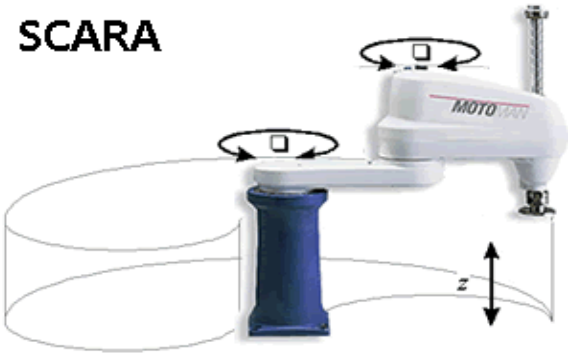


Figura 2: Braç robòtic Scara (1979)

Al 1982, IBM introdueix el robot RS-1 per a muntatge. Aquest està format per un braç constituït per tres dispositius de desplaçament ortogonal. El llenguatge que es va utilitzar per programar aquest robot va ser l'AML.

Com es pot observar els robots industrials van anar evolucionant molt i cada cop s'utilitzaven per més aplicacions, com per exemple: la realització de tasques perilloses (neteja de fuites de combustible en centrals nuclears, etc), en línies de muntatge automàtiques, per la pintura de peces, en tasques d'assemblatge, ...
Aquest gran desenvolupament de la robòtica va fer que es comencessin a desenvolupar robots humanoides (robots basats en l'aparença humana per permetre'ls interactuar amb eines destinades a ser utilitzades per persones).

Al 1985 el robot WASUBOT, construït per la universitat de Waseda (Tokio), podia tocar un instrument amb un teclat, després de llegir la partitura.

L'any 1996 Honda Motor Company implementa un robot humanoide, el P2, capaç de moure's de forma autònoma i de forma similar a la d'un ésser humà. El robot ASIMO (figura 3), creat l'any 2000, va ser un model evolucionat d'aquest.

Figura 3: Evolució dels robots humanoides

Al 1997 el robot Mars Pathfinder, desenvolupat per la NASA, explora i recull mostres de la superfície del planeta Mart.

L'any 1999 Sony Corporation construeix el AIBO ERS-110, considerat el primer robot d'entreteniment, ja que reprodueix el comportament d'un gos.

L'any 2000 una companyia de robòtica domèstica, Friendly Robotics, posa a la venda un tallagespa completament automàtic (Robomow RL500).

El Global Hawk, un vehicle no tripulat utilitzat en missions de vigilància per a les forces aèries i la marina d'Estats Units , va realitzar el seu primer vol autònom sense aturades a través de l'oceà Pacífic a l'abril de 2001. Va anar des de Califòrnia fins al sud d'Austràlia en 22 hores.

El mateix any, el Canadarm 2 va ser llançat i unit a l'estació espacial Internacional. Aquest és un braç robòtic utilitzat per a tasques de manteniment i assemblatge, com per exemple: moure provisions i equips, subjectar astronautes mentre treballen,...

Els dies 3 i 24 de gener de l'any 2004 els robots Spirit i Opportunity, respectivament, van aterrar sobre la superfície de Mart.
L'Spirit va funcionar 20 vegades més del que havien previst a la NASA, fins que l'any 2009 va quedar encallat en una zona on la terra era molt tova , però ha seguit realitzant investigació científica des d'aquell lloc. El 22 de març de l'any 2010 es va rebre l'última comunicació des de l'Spirit i segons el que suggereixen les projeccions de combustible es creu que ha entrat en un mode d'hibernació que pot durar mesos.
L'Opportunity ha realitzat proves sobre les roques i la superfície (cràters,...) de Mart de forma extensa. Ha superat, també, les expectatives quant a la seva durabilitat de forma excepcional, fins al punt de convertir-se, des del 19 de maig del 2010, en la missió més llarga duta a terme sobre la superfície del planeta.

Una versió nova de l'ASIMO, amb noves capacitats i comportaments, va ser anunciada per Honda l'any 2005.

Google va anunciar, al setembre del 2007, el seu "Lunar X Prize" que oferia 20 milions de dòlars a la primera companyia privada que aconseguís fer aterrar un robot a la lluna i que fos capaç de recórrer més de 500 metres i transmetre imatges i vídeo en alta definició.

Com es pot comprovar la robòtica fa algun temps que és present al món i, cada vegada, de manera més significativa. En els últims anys ha aparegut en nous camps on no s'havia utilitzat, cosa que fa augmentar les seves possibilitats d'aplicació de forma molt significativa.
Per aquest motiu, en la següent secció, es tractaran les diverses àrees on s'utilitza aquesta branca de la tecnologia.

3.1 Àrees d'investigació de la robòtica

Tot i existir moltes classes de robots, gran part dels que s'utilitzen avui en dia ho són per a finalitats industrials. En les tres últimes dècades, el 70% dels robots del món s'han concentrat en les industries de l'automòbil i de l'electrònica. La majoria d'ells realitzen tasques programables i fàcilment robotitzables.
L'evolució de les tecnologies per a la sensorització (sensors més precisos i complerts), la millora dels actuadors (més ràpids, flexibles i lleugers), els controls més robustos i adaptables i el disseny de noves interfícies està portant a l'augment dels robots que realitzen tasques molt més sofisticades.
Aquestes noves àrees d'aplicació de la robòtica s'han agrupat sota el nom de robòtica de serveis. Aquesta classe de robots es caracteritzen per acostar-se al ciutadà per realitzar tasques de tipus domèstic i professional, i apareixen en sectors com la medicina, neteja, manteniment, construcció...

A continuació s'expliquen les diferents àrees d'investigació de la robòtica tal i com s'han definit a la plataforma europea de la robòtica (EUROP):

- Robòtica industrial
- Robòtica de serveis
- Robots espacials i de seguretat

3.1.1 Robòtica industrial

La robòtica industrial és aquella part de la disciplina que es dedica a construir màquines destinades a realitzar treballs mecànics i repetitius de forma eficient i amb baix cost.

Com s'ha comentat anteriorment l'ús de la robòtica en la industria és una de les aplicacions, relativament més antigues, d'aquestes màquines. El primer robot (de Unimate) va ser instal·lat l'any 1961 per General Motors a les seves factories.
L'ús d'aquesta tecnologia ha tingut una gran influència en l'industria automobilística i electrònica, ja que gràcies a ella s'ha aconseguit una reducció de costos de producció d'un 40% en els últims 20 anys.

L'automatització industrial a les empreses és un factor rellevant per augmentar la productivitat i competitivitat dels seus productes enfront de nous mercats emergents que es basen en la mà d'obra barata. També cal remarcar que, l'ús de robots industrials, permet mantenir llocs de treball més qualificats on és requereix una major habilitat i competència.

En els pròxims anys la investigació de la robòtica, en aquest camp, haurà de buscar maximitzar la productivitat i la flexibilitat, i aconseguir una integració complerta en els processos de fabricació. Per tant alguns dels principals reptes que es plantegen són:

- Desenvolupar interfícies que possibilitin una programació i manipulació més simple dels robots (comunicació gestual, per veu,...).

- Establir sistemes de fabricació on els robots cooperin amb els treballadors. Això obliga a implementar mecanismes més segurs on els operaris i els robots puguin treballar de forma conjunta.

- Millorar els sistemes de sensors (augmentar la velocitat, la resolució, la precisió, incorporar intel·ligència, ús de sistemes de diagnòstic,...)

- Permetre una programació flexible en els robots basada en la informació dels sensors i no en seqüències preestablertes, cosa que augmentarà l'adaptabilitat a nous processos de fabricació.

- Desenvolupament de nous sistemes de manipulació i subjecció (braços més flexibles, ràpids i amb una relació força/pes 1:1 (les actuals són de 1:10)). Això obligarà a utilitzar materials més lleugers i resistents i motors més potents i reduïts.

- Millora dels sistemes on els robots cooperen, entre ells, en els processos de fabricació (modificació dels sistemes de transport, comunicació entre robots,...).

- Obtenir cicles de producció més curts i sistemes flexibles reconfigurables desenvolupant sistemes de planificació intel·ligent.

- Millorar els mètodes de detecció de fallades i de recuperació, la qual cosa augmentarà la robustesa dels sistemes.

- Desenvolupar sistemes d'autodiagnòstic intel·ligents, funcions de reparació autònoma o semi autònoma i mètodes per a la detecció preventiva de fallades.

3.1.2 Robòtica de serveis

La robòtica de serveis, explicada anteriorment, és la branca d'aquesta disciplina que desenvolupa robots capaços de proporcionar serveis de forma directe als membres de la societat.

L'ús d'aquesta tecnologia per aquests fins és més recent que la robòtica industrial i per això, és d'esperar que en els pròxims anys tingui una gran expansió. Amb el seu ús es poden solucionar alguns problemes com el difícil accés a certs llocs de treball, el perill que comporta la realització de certes tasques (ex: netejar els vidres exteriors d'un

gratacels, lluitar contra el foc, desactivar bombes,...), la necessitat de treballar en llocs desagradables per un ésser humà (clavegueres, abocadors, ...), etc

El terme robòtica de serveis no apareix fins a l'any 1989 quan Joseph Engelberger va publicar el llibre "Robotics in Service". Allà ja es definien diferents camps de l'aplicació de la robòtica de serveis:

- Agricultura
- Neteja
- Medicina (a l'apartat 4.2 s'entrarà en més detall en aquest camp, ja que és el que interessa en aquest projecte)
- Construcció
- Ajuda a persones grans o discapacitades
- Serveis militars
- Rehabilitació
- Vigilància
- Feines de la llar
- Mineria
- Transport (gasolineres,...)
- Manteniment
- ...

Posteriorment Schmierer ("Schraft and Schmierer" 1989) en va afegir alguns més com:

- Robots d'entreteniment
- Robots per a la lluita contra el foc
- Robots per al rescat
- Càtering i aplicacions de serveis en hotels i restaurants

3.1.3 Robòtica espacial i de seguretat

La robòtica espacial és aquella on les màquines han d'operar a l'espai exterior. Aquests robots poden ser controlats localment (ex: braç a l'exterior d'una nau) o des de grans distàncies (ex:robot que es mou per la superfície d'un planeta) i realitzen tasques de construcció, manteniment, assemblatge, exploració,...

Els principals motius d'utilitzar robots a l'espai és degut a que és un entorn hostil, perillós i llunyà. Això fa que moltes vegades el desplaçament humà fins a la destinació sigui inviable. Un altre motiu és el fet que viatjar a l'espai és molt car i en el cas d'enviar robots no cal fer que aquests retornin a la terra.

A l'hora del disseny d'aquesta classe de robots s'han de tenir en compte certs aspectes com per exemple:

- La comunicació remota amb un robot a l'espai comporta retards (ex: enviar un senyal fins a Mart tarda uns 30 minuts)

- El cost del transport dels robots fins a la seva destinació obliga a optimitzar al màxim el seu volum, pes i consum d'energia.

- Com que els llocs on s'envien els robots no són accessibles per als humans que els controlen, un petit error podria fer fracassar la missió. Per això, els robots espacials necessiten tenir sistemes molt robustos i l'autonomia necessària per realitzar auto diagnòstics i, fins i tot, auto reparacions.

- Les condicions extremes (radiacions, temperatura, pols, etc.) fan que aquesta classe de robots hagin d'estar construïts amb materials que no es vegin afectats per aquests fenòmens.

- Els robots que exploren altres planetes requereixen ser capaços de moure's per terrenys desconeguts.

3.2 La robòtica en la medicina

L'ús de la robòtica en medicina, com s'ha comentat en la secció anterior, forma part de l'anomenada robòtica de serveis. Tot i així, s'ha destinat un apartat separat per parlar-ne, ja que és on es centra l'objectiu d'aquest projecte i, per tant, s'ha considerat important realitzar-ne un estudi més exhaustiu.

La cirurgia assistida per ordinador ("CAS: Computer Assisted Surgery) representa un conjunt de mètodes que utilitzen la tecnologia informàtica per a la planificació necessària prèvia a la cirurgia, per a la seva guia i per a la seva realització. Algunes de les tècniques més importants del CAS són la realització de models del pacient i l'ús de robots quirúrgics que permetin dur a terme cirurgies mínimament invasives amb millors condicions de precisió i seguretat.
Els primers professionals que van començar a investigar aquestes tècniques van ser els professors Takeyoshi Dohi i Masakazu Tsuzuki, a la universitat de Tokio.

El CAS es recolza en els següents punts:

- Processament i visualització d'imatges mèdiques (raigs X, tomografies, ressonàncies magnètiques, ultrasons, exploracions internes, etc.). Mitjançant l'ús de diverses imatges mèdiques s'aconsegueixen construir models tridimensional que representen exactament els òrgans que s'han d'operar. Posteriorment aquestes imatges es poden processar i, utilitzant diversos contrasts, permeten diferenciar a través d'un ordinador els diferents teixits i estructures d'un ésser humà.

- Planificació d'operacions: gràcies a aquests models i a l'ús de software especialitzat en el processament d'imatges (ex: OsiriX) el cirurgià pot obtenir imatges, des de tots els angles i a qualsevol profunditat, cosa que li permet fer diagnòstics més acurats que l'ajudin a planificar la posterior operació.

- Simulació quirúrgica: amb els models obtinguts del pacient, els cirurgians, poden realitzar simulacions de la cirurgia que després poden programar en el robot que la realitzarà.

- Realització de la cirurgia mitjançant l'ús de la robòtica: els robots, generalment un braç mecànic, poden utilitzar-se per diferents funcions en una intervenció:

    - El robot pot realitzar, com s'ha comentat anteriorment, un conjunt d'accions que han estat preestablerts gràcies a la simulació quirúrgica.
    - El robot pot ser controlat de forma remota (teleoperació) i reproduir els moviments que realitzi el cirurgià (sistema mestre/esclau).
    - El robot i el cirurgià poden realitzar la cirurgia de forma conjunta. Mentre el cirurgià realitza el procediment, el robot realitza d'altres tasques com per exemple el posicionament i suport d'eines mèdiques (làsers quirúrgics, càmeres i altres), el subministrament d'anestèsia, etc

Des de l'any 1990 les intervencions assistides per robot s'han considerat de gran importància pel convenciment que l'ús d'aquesta tecnologia pot revolucionar la medicina. Alguns dels principals avantatges que presenta l'aplicació d'aquesta disciplina per a finalitats mèdiques són els següents:

- La precisió dels robots en el moviment és molt més exacta que la d'una persona.
- Els robots, a diferència del pols humà, no tenen vibracions.
- Els robots no es cansen, no es posen nerviosos ni tenen canvis d'humor.
- El comportament d'un robot és perfectament conegut abans de realitzar-se l'operació cosa que fa que es pugui estudiar i corregir.
- Els robots poden dirigir-se de forma remota.
- Els robots poden reaccionar molt més ràpidament que un metge.

- Diversos robots es poden coordinar per tal de realitzar tasques complexes amb una alta eficiència.

L'ús de la robòtica pot ser de gran utilitat en la medicina, ja que en certes situacions, com a l'hora de realitzar intervencions complexes o aquelles que es duen a terme en llocs on l'ull humà té difícil accés, pot proporcionar una alternativa millor a la cirurgia convencional.

Tot i que la principal finalitat que interessa de la robòtica en la medicina en aquest projecte és la del seu ús en cirurgia, a continuació s'ofereix una classificació dels diferents robots utilitzats en medicina segons la seva funcionalitat:

- Robots quirúrgics: aquells robots utilitzats per assistir a un cirurgià en una operació. Es poden classificar en diferents categories:

    - Robots autònoms: s'utilitzen en situacions on es requereix realitzar operacions de mecanitzats precises segons un model , generalment en tasques de foradar o tallar ossos. Després de subjectar la part sobre la que s'ha de treballar, es situa i s'orienta el robot a través d'un navegador (sovint basat en un sistema de visió) i posteriorment aquest realitza la feina.

    - Robots teleoperats: utilitzats en cirurgia de mínima invasió. Són robots controlats a distància a través d'un joystick, la veu o altres dispositius del tipus "mestre/esclau" i, per tant, no treballen de forma autònoma. Generalment s'utilitzen per moure la càmera endoscòpica i altre instrumental.

    - Robots manipuladors: robots que fan d'assistent a l'hora de carregar i subministrar eines mèdiques, es mouen per ordre de cirurgià en un espai restringit.

- Micro màquines: robots a escala micromètrica dedicats a tasques especifiques com la sutura automàtica, la unió de venes o biòpsies (extracció de teixit) automàtiques. Poden ser utilitzats per altres robots com a instruments i, generalment, es consideren més màquines que robots.

- Robots per a la distribució i l'emmagatzematge de medicines: són aquells robots utilitzats per automatitzar el control dels medicaments subministrats als pacients en els grans hospitals, per tal d'evitar errors humans.

- Robots per a la rehabilitació i pròtesis: moltes teràpies de recuperació es basen en la realització de moviments de forma repetitiva durant molt de temps. Això suposa un gran cost, ja que el fisioterapeuta també necessita realitzar un gran esforç. Amb l'ajuda dels robots, al fisioterapeuta només li cal realitzar un seguiment de l'evolució del pacient i dissenyar les teràpies més eficients.
  Un altre ús d'aquesta tecnologia és la creació d'extremitats robòtiques. Aquestes són cada cop són més utilitzades, ja que s'ha aconseguit que, a diferència de les utilitzades antigament que eren passives, les pròtesis mioelèctriques responen a la voluntat del pacient (estímuls nerviosos o musculars). Això ha permès incrementar, de forma significativa, la seva qualitat de vida. En un futur aquestes pròtesis funcionaran amb estímuls cerebrals.

Com s'ha comentat anteriorment, els robots que interessen per aquest projecte són els anomenats robots quirúrgics, en concret els teleoperats. Aquests poden ser manipulats a distància per els cirurgians seguint un model "mestre/esclau" donant lloc a la teleoperació o cirurgia assistida per computador.
Actualment aquests robots poden ser programats per realitzar moviments de forma molt precisa i permeten realitzar operacions a través d'una obertura molt petita (mil·límetres), gràcies a l'instrumental associat.

Les primeres aplicacions de la robòtica en cirurgia daten de l'any 1982, quan es va començar un programa que pretenia utilitzar un robot (PUMA) per realitzar operacions de neurocirurgia a Califòrnia.

En l'actualitat s'han desenvolupat els sistemes robòtics per ajudar a posicionar les càmeres en aquelles cirurgies on l'àrea d'intervenció no és accessible per l'ull humà degut a que els instruments entren al cos del pacient per un orifici mil·limètric. La primera operació d'aquest tipus, anomenada de mínima invasió o laparoscòpia, va tenir lloc el febrer de 1986, fruit de la recerca del departament d'ESAII de la UPC. Es va realitzar al Centre Quirúrgic Adrià, de Barcelona, on un robot de forma autònoma guiava la càmera a partir del seguiment dels instruments quirúrgics que incorporaven unes marques. El 1987, s'aconsegueix també a Alemanya, a partir de marques de color als instruments.

L'any 1992, l'enginyer Philippe Green va realitzar investigacions en manipulació remota a Palo Alto (Califòrnia) i va desenvolupar prototips de sensors. Seguidament el doctor Stephen Jacoben va desenvolupar braços i mans robòtiques que replicaven, per control inalàmbric, els moviments humans.

El 1994 la FDA ("Food and Drug Administration") va aprovar el primer robot per l'ús clínic en l'abdomen, l'AESOP. Aquest va ser fabricat per Computer Motion fundada amb una concessió per la investigació de la NASA amb la finalitat de desenvolupar un braç robòtic per el programa espacial d'Estats Units. Posteriorment aquest braç va ser modificat per tal de sostenir un laparoscopi i poder ser controlat per el cirurgia tant de forma manual com a distància.

Des de l'any 1994 fins el 2003 es van realitzar més de 300000 intervencions quirúrgiques d'aquest tipus a tot el món.

L'any 1995 es va formar l'empresa Intuitive Surgical Inc. que va adquirir els drets d'investigació de l' anteriorment mencionat Philippe Green.
En l'actualitat, un dels sistemes robòtics mèdics més famosos és l'anomena't da Vinci, construït per aquesta empresa.

Da Vinci

Aquests sistema robòtic va ser dissenyat per l'empresa Intuitive Surgical Inc. per tal de dur a terme cirurgies mínimament invasives (explicades en el següent apartat), facilitant així la realització de certes intervencions complexes.

El Robot Da Vinci (figura 4) està format per tres elements:

- La consola del cirurgià, a través de la qual es detecten els seus moviments i es traslladen electrònicament i en temps real al robot. Aquests moviments generalment s'escalen fins a ser micromoviments i s'hi filtren els tremolors. La consola consta de dos forats per als ulls, a través dels quals el cirurgià veu la imatge de la zona operada, transmesa per una càmera situada dins el pacient, i on també s'hi mostren els seus signes vitals. Per manipular els instruments situats en els braços del robot, el cirurgià utilitza dos pedals i dos controladors per a les mans, aquests últims presenten el desavantatge de no ser intuïtius ja que el cirurgià no realitza els mateixos moviments que duria a terme en una intervenció quirúrgica real.
  Cal esmentar les grans dimensions d'aquesta consola.

- Un robot amb 4 braços, situat amb el pacient, que controla el cirurgià. Els moviments de les articulacions d'aquests braços excedeixen el rang natural de mobilitat humana gracies als instruments quirúrgics utilitzats. En tres d'aquests braços robòtics s'hi munten diferents instruments quirúrgics articulats que són introduïts en el cos del pacient a través de trocars. En el quart braç s'hi munta una càmera endoscòpica.

- Un sistema de visió tridimensional d'alta resolució. La càmera endoscòpica proporciona al cirurgià una visió estereoscòpica a través de la consola.

El sistema da Vinci va ser dissenyat principalment per millorar les intervencions de cirurgia laparoscòpica convencional on el cirurgià opera dret i, amb les mans, aguanta els instruments que tenen una forma allargada i no disposen d'articulacions. A més a

més, el cirurgià ha de mirar la imatge proporcionada per una càmera aguantada per un ajudant, en un monitor de dues dimensions, per poder veure la zona on estava operant.

Figura 4: sistema robòtic per dur a terme cirurgies mínimament invasives amb el suport de la teleoperació Da Vinci

Gràcies al sistema robòtic teleoperat, el cirurgià pot manipular tots els instruments còmodament assegut, a través d'uns controls alineats amb els instruments i tenint una visió tridimensional de la zona operada. Tot i aquests avantatges, la posta apunt abans d'una intervenció és llarga i complexa degut a la pròpia estructura del robot amb 4 braços.

La primera intervenció realitzada utilitzant aquest sistema va ser duta a terme el 24 d'agost del 1999 per un equip de metges alemanys, en el departament de Cirurgia Cardiovascular i Toràcica de l'Hospital Johann Wolfang von Goethe.

El sistema da Vinci va ser utilitzat en 48000 procediments l'any 2006 i estava a la venda per aproximadament 1.2 milions de dollars.
L'any 2009 es va posar a la venda el nou sistema da Vinci HD per 1.75 milions de dòlars.

Cirurgia mínimament invasiva

El propòsit d'aquest projecte és aconseguir dissenyar una interfície de comandament teleoperat que permeti governar un útil quirúrgic a través d'un braç robòtic. Això permetrà realitzar intervencions de cirurgia mínimament invasives que tenen força avantatges en comparació amb la cirurgia oberta.

L'ús de la robòtica per a aquest fi encara té inconvenients, però alguns problemes com la poca precisió, haver de treballar en posicions incomodes o en zones de difícil accés i els possibles tremolors o la fatiga humana es poden solucionar amb l'ús d'aquestes tecnologies.

Des del punt de vista del pacient la cirurgia mínimament invasiva suposa un avantatge, ja que es redueix el seu patiment, el risc d'hemorràgies i el dany que es realitza als teixits. Tots aquests fets disminueixen el temps de recuperació i, per tant, l'estada a l'hospital. Un altre avantatge d'aquest tipus d'intervenció és estètic, ja que al reduir l'àrea de cirurgia la cicatriu resultant de l'operació és molt més petita.

D'altra banda, l'ús d'aquestes tècniques quirúrgiques que utilitzen robots, des del punt de vista del cirurgià, presenta alguns inconvenients com per exemple la necessitat que aquest senti el que està realitzant el robot com si ho fes ell i, en conseqüència, actuïn com un de sol. Per solucionar aquests problemes s'estan utilitzant sensors per detectar l'esforç que realitza l'útil quirúrgic. Aquest esforç és transmès posteriorment al cirurgià a través d'algun element hàptic.

Laparoscòpia

Tècnica endoscòpica de cirurgia mínimament invasiva on els instruments i el laparoscopi han de passar per una petita incisió (generalment entre 0.5 i 1 cm) cosa que fa que els graus de llibertat dels instruments quedin restringits a quatre.

Per tal de fer passar els instruments a través d'aquesta incisió, la tècnica més segura és afegir articulacions passives que fan que el mecanisme s'ajusti per ell mateix en cas que el pacient, durant la cirurgia, es mogui.

Figura 5: laparoscopi

Com s'ha comentat, en una intervenció basada en cirurgia laparoscòpica, apart dels instruments es fa servir un element molt important anomenat laparoscopi (figura 5). Aquest és un aparell òptic que s'utilitza per veure el contingut de la cavitat abdominal. N'existeixen dos tipus: un format per un sistema de lents telescòpiques que generalment està connectat a una càmera de vídeo; i un de digital on la càmera es situa al final del laparoscopi i s' eliminen així les lents utilitzant transmissió per fibra òptica.

El tipus de càmera que es fa servir s'anomena detector CCD (figura 6), detector de dispositiu d'acoblament de càrrega ("charge coupled device"). Aquestes càmeres compten fotons per tal de mesurar, en una matriu de punts, la quantitat de llum que els hi arriba a cada un d'ells i generar, gràcies a això, una imatge digitalitzada en color. Existeixen també les càmeres 3CCD on la llum es descomposta en vermell, verd i blau gràcies a un prisma i cada CCD s'encarrega de mesurar un d'aquests 3 colors. Aquestes càmeres ofereixen imatges de major qualitat i resolució.

Figura 6: detector CCD a l'extrem d'un laparoscopi

Al laparoscopi també si acobla un sistema de fibra òptica que està connectat a una font de llum freda (halògena o xenó) per tal d'il·luminar la zona operada.

La cirurgia laparoscòpica inclou les operacions entre l'abdomen i la pelvis. El procediment utilitzat consisteix en realitzar una petita incisió a la pell a través de la qual s'introdueix diòxid de carboni a la cavitat abdominal. Aquest gas és inofensiu i gràcies a

ell s'aconsegueix augmentar la zona de treball. Cal dir que els instruments s'introdueixen en aquest petit orifici a través de trocars quirúrgics (figura 7) que estan segellats per tal d'evitar les fugues d'aquest diòxid de carboni introduït en el pacient.

Figura 7: trocar

Per poder suturar dins del cos, els cirurgians necessiten realitzar moviments precisos i articulats, sobretot per realitzar les puntades, els nusos i per separar teixits. La majoria dels instruments laparoscòpics que es poden trobar al mercat són llargs i rígids i només permeten l'obertura i el tancament de la pinça a través d'un mànec semblant al d'unes tisores, o realitzen altres actuacions com la de bisturí electrònic. L'angle de rotació es pot ajustar a través del puny.

Figura 8: diferents d'instruments laparoscòpics mecànics

La investigació de les tècniques de laparoscòpia ha portat al desenvolupament de diferents instruments (figura 8), alguns purament mecànics i d'altres mecatrònics. Els primers són més barats i senzills de desenvolupar i, per tant, són els que predominen en el mercat. Tot i així, per sobrepassar les limitacions dels instruments mecànics, els mecatrònics i robòtics també han estat desenvolupats. En aquests, el moviment de graus de llibertat addicionals, és generat per actuadors.

4 PLANTEJAMENT DEL PROJECTE

En aquest apartat es tracten tots aquells aspectes relacionats amb el plantejament del projecte de final de carrera i la seva organització. En un projecte d'aquesta magnitud és molt important realitzar una bona planificació per tal de poder abordar-lo amb les màximes garanties d'èxit possibles.

El primer que cal fer és identificar les tasques principals en les què es divideix el projecte observant-ne els seus objectius. Un cop obtingudes aquestes tasques és necessari subdividir-les en altres més especifiques per tal de poder realitzar una millor planificació d'aquestes. Cal identificar aquelles parts del projecte que són més importants que d'altres i a les quals, per tant, s'haurà de dedicar més temps.

Una altra part molt important és identificar les dependències entre les diferents tasques, és a dir, mirar quines s'han de realitzar abans que d'altres ja que les segones requereixen tenir les primeres enllestides per tal de poder començar a fer-les.

Per últim, amb tota aquesta informació i sabent el temps del que es disposa per realitzar el projecte, s'ha d'assignar un ordre i un període de temps a cada tasca .

Cal tenir en compte que sempre és normal que hi hagi parts que portin menys temps del previst i d'altres que s'allarguin per possibles entrebancs o perquè és complicat realitzar una previsió del temps que s'haurà de dedicar a una tasca per enllestir-la. Tot i així s'ha d'intentar realitzar una planificació el màxim de realista possible.

A les planes següents es mostra un diagrama de Gantt (figures 9 i 10) on es pot observar la planificació, les dependències i els recursos assignats a cada una de les diferents tasques del projecte.

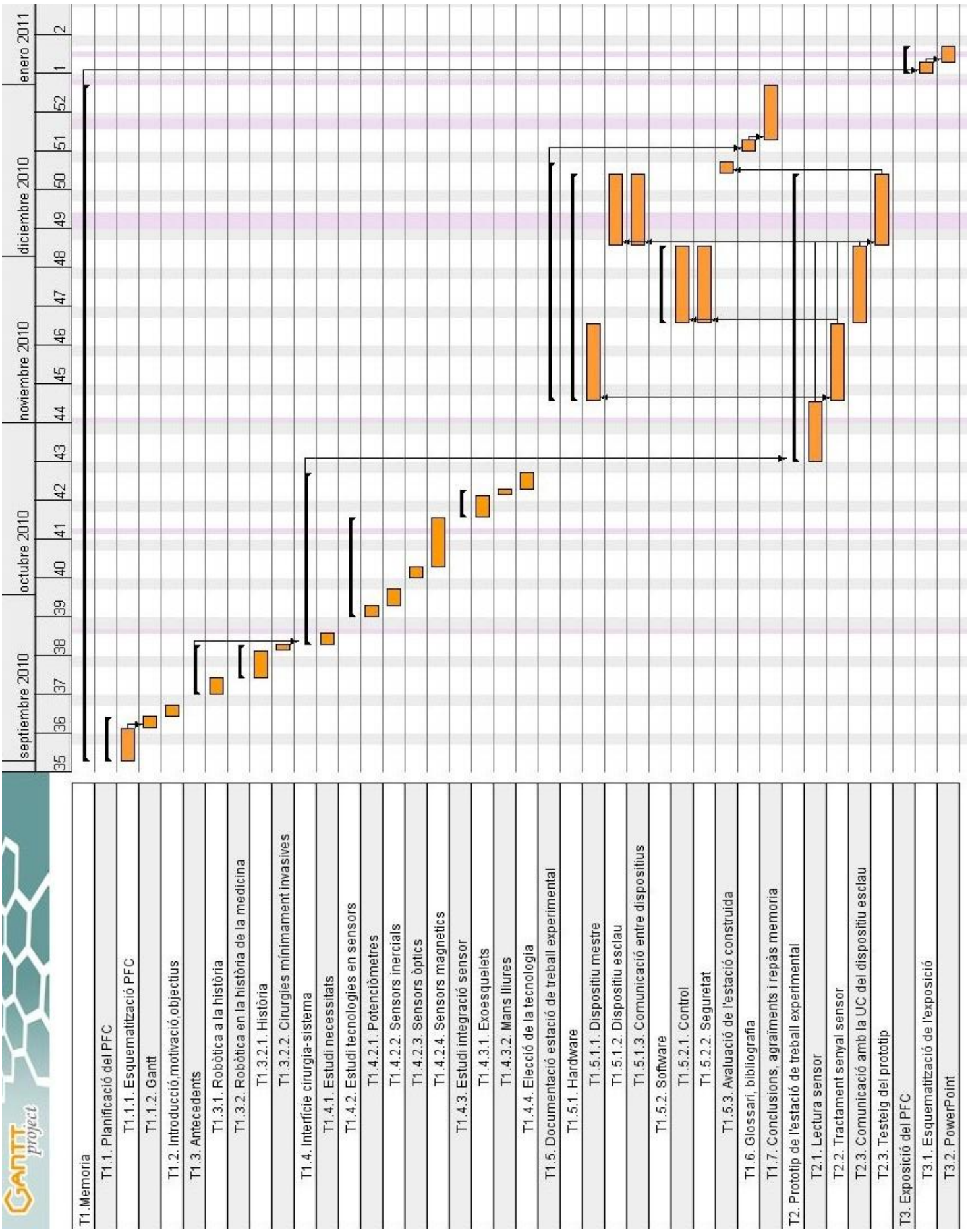


Figura 9: Planificació i dependències de cada tasca (diagrama Gantt)

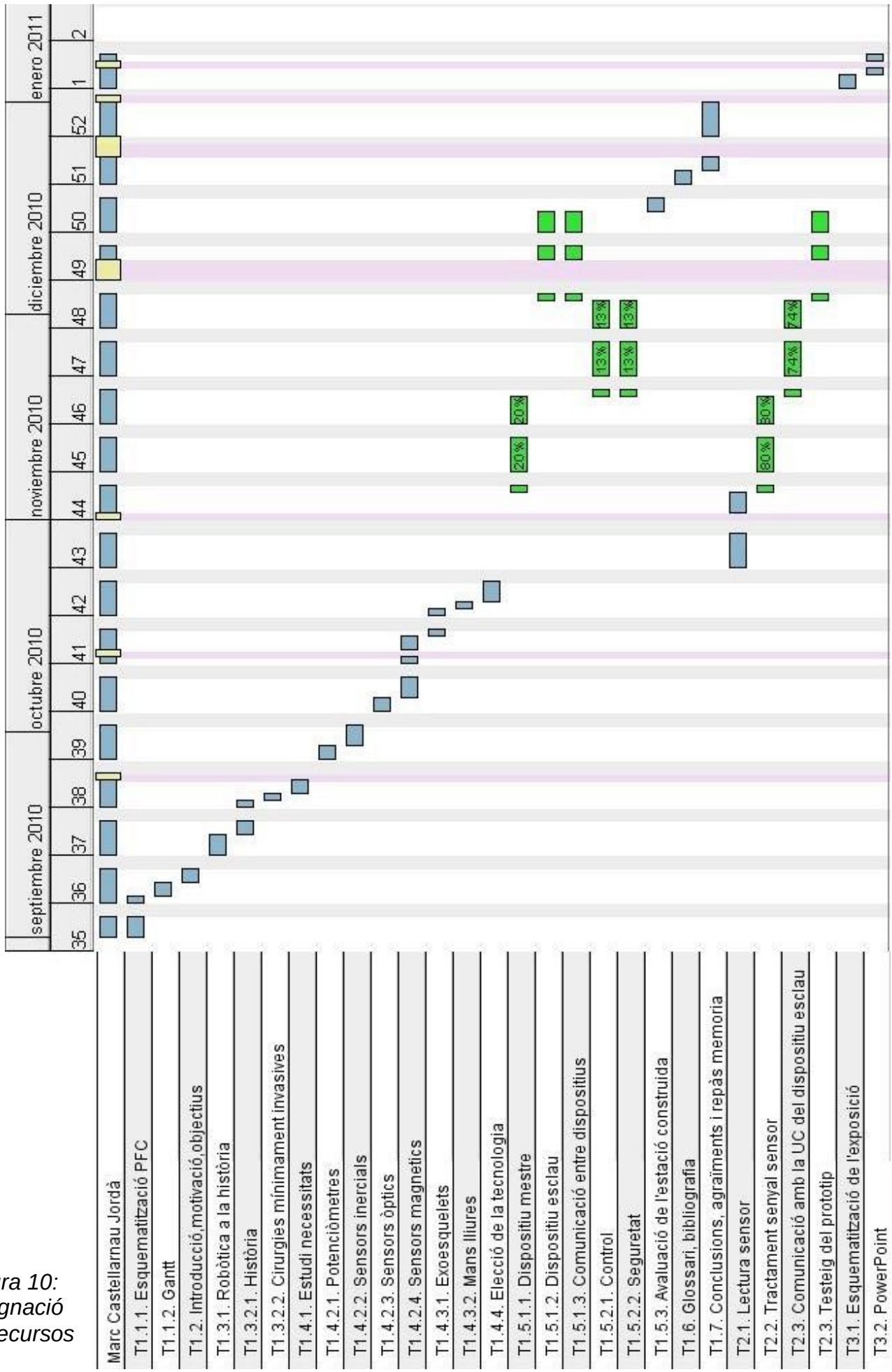


Figura 10: assignació de recursos

5 INTERFÍCIE CIRURGIÀ/SISTEMA

Com el nom del projecte indica, “Interfície pel comandament teleoperat d'útils quirúrgics”, el seu objectiu principal és el de dissenyar-ne una que permeti a un cirurgià comunicar-se amb una màquina per tal de poder realitzar una cirurgia mínimament invasiva de forma teleoperada.

Per tal de realitzar el disseny d'una bona interfície caldrà estudiar els requisits que aquesta haurà de complir i que es deriven de les funcionalitats del sistema:

- Fiabilitat: com que el sistema s'utilitzarà per realitzar intervencions a éssers humans caldrà que aquest ofereixi les màximes mesures de seguretat possibles per tal d'assegurar que no es produeixi cap dany al pacient.
- Precisió: per tal de realitzar una intervenció és necessari dur a terme moviments precisos. Per aquest motiu la interfície utilitzada per captar les ordres del cirurgià ha de ser capaç de fer-ho de forma precisa.
- Usabilitat: el sistema dissenyat ha de poder ser utilitzat per un cirurgià, fet que fa necessari que el mètode de comunicació entre aquest i la màquina sigui el màxim d'intuïtiu possible. Per aconseguir-ho s'ha decidit dissenyar una interfície que permeti llegir els moviments involucrats en la realització d'una operació quirúrgica. D'aquesta manera el cirurgià podrà realitzar els mateixos moviments que fa en una operació en la qual els útils quirúrgics els subjecta amb les seves pròpies mans.

  Aquest fet minimitzarà el temps d'aprenentatge requerit per l'ús del sistema dissenyat.
- Comoditat: per la delicadesa de la funcionalitat del sistema, es considera important dissenyar una interfície que possibiliti captar els moviments del cirurgià permetent que aquest treballi d'una forma còmode i natural.

5.1Necessitats cinemàtiques

Com s'ha explicat, es busca aconseguir sensoritzar certs moviments de l'instrument laparoscòpic per tal de poder transmetre'ls al braç robòtic que els reproduirà. Per tal de captar els moviments que interessen, els corresponents als 4 graus de llibertat involucrats en cirurgia mínimament invasiva (x, y, z, Θ), existeixen diversos tipus de sensors que ens poden ajudar.

A continuació s'expliquen les necessitats cinemàtiques del sistema que es vol desenvolupar, les principals classes de sensors que es troben al mercat i els seus avantatges i inconvenients per dur a terme l'objectiu desitjat. Aquesta és una part molt important ja que, com s'ha comentat anteriorment, per dur a terme intervencions de cirurgia mínimament invasiva de forma teleoperada s'han de tenir en compte aspectes com la fiabilitat, precisió, usabilitat, comoditat,... que tindran una repercussió molt important a l'hora de desenvolupar un sistema útil i eficaç.

5.1.1 Graus de llibertat

Un dels principals objectius d'aquest projecte és el de sensoritzar els moviments dels instruments laparoscòpics necessaris per dur a terme una cirurgia mínimament invasiva.

Aquests moviments que interessa captar són, perquè els instruments passen per un petit orifici, els corresponents a 4 graus de llibertat: x, y, z, Θ. Amb aquests es podran desplaçar els diferents utensilis involucrats en una microcirurgia en les tres dimensions i també es permetrà rotar en un dels tres eixos possibles. A continuació es mostra una imatge on es poden observar els moviments corresponents als 4 graus de llibertat esmentats a l'extrem de l'instrument laparoscòpic (figura 11):

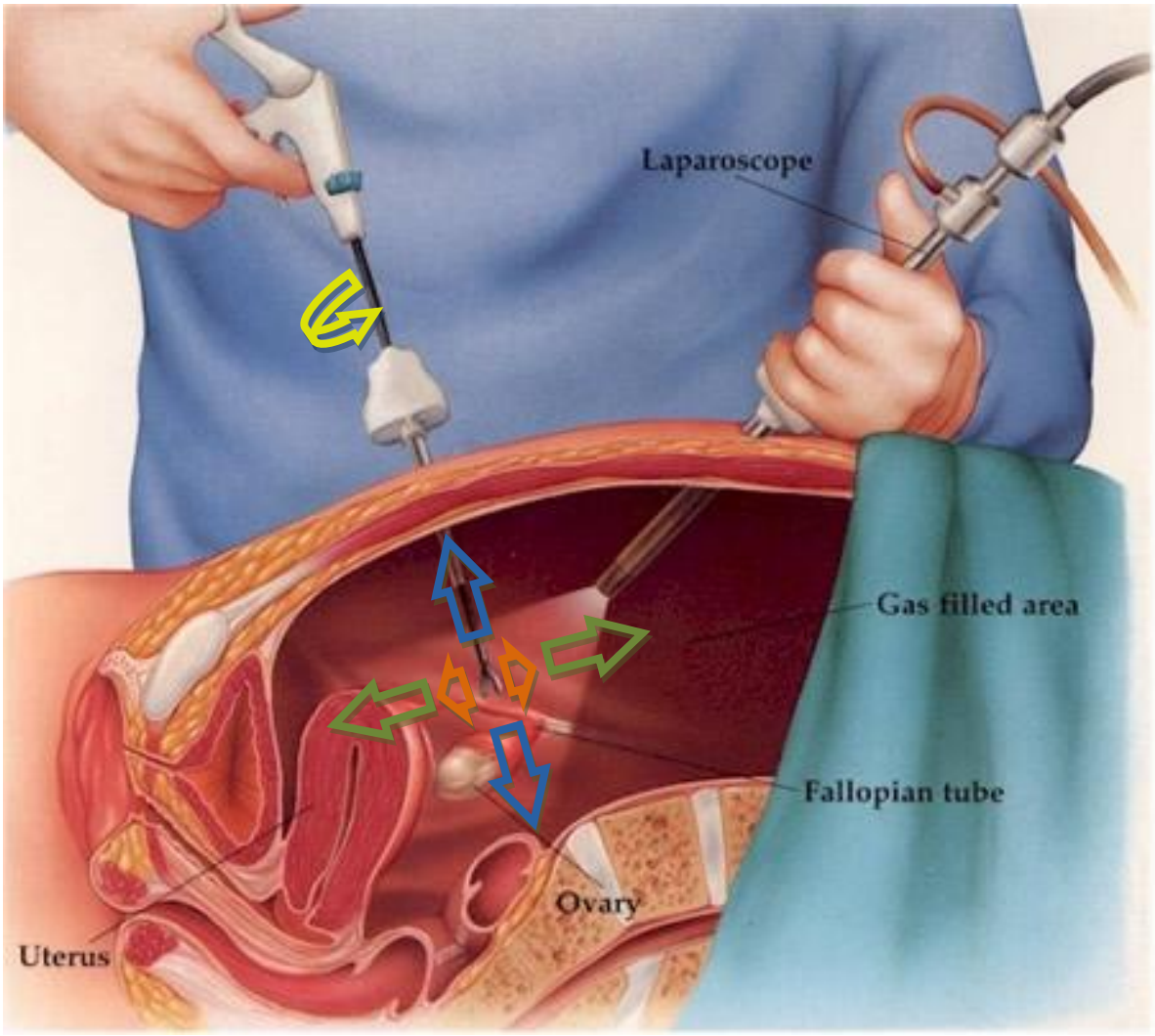


Figura 11: esquema dels 4 graus de llibertat involucrats en una intervenció mínimament invasiva.

5.1.2 Tipus de sensors

S'anomena sensor a aquell dispositiu que mesura magnituds físiques o químiques com per exemple acceleració, distància, temperatura, desplaçament, força, pressió, intensitat llumínica, ... Els sensors són capaços de transformar aquestes magnituds, anomenades variables d'instrumentació, en variables elèctriques com per exemple tensió (diferència de potencial), corrent elèctric (intensitat elèctrica), ...

Els sensors disposen de transductors, que són dispositius capaços de transformar una forma d'energia en una altra. D'aquesta manera s'aconsegueix transformar la magnitud

que es vol mesurar en una altra, la mesura de la qual és més simple. Aquesta transformació pot ser directa (ex: el mercuri d'un termòmetre) o es pot tenir el sensor connectat a un equip de mesura on es pugui llegir i processar el seu valor (ex: un conversor analògic-digital, un PC i un display,...).

Aquests dispositius tenen diferents característiques que són molt importants a l'hora d'utilitzar-los, ja que en funció d'aquestes oferiran diferents prestacions:

- Resolució: variació mínima en la variable d'instrumentació que el sensor és capaç de detectar.
- Repetitivitat: capacitat d'un instrument de donar el mateix resultat en mesures diferents de condicions iguals.
- Precisió: capacitat d'un instrument de mesurar una magnitud de forma que s'acosti el màxim a la magnitud real. Altrament es pot entendre com l'error màxim esperat entre la mesura i el valor real.
- Offset: valor de la variable de sortida quan la variable d'entrada és 0.
- Rang: conjunt de valors de la magnitud d'entrada (domini) en els què es pot utilitzar el sensor.
- Sensibilitat: relació entre la variació de la magnitud d'entrada i la de sortida.
- Temps de resposta: interval que es dóna entre la variació de la magnitud d'entrada i la sortida del sensor. Pot ser fix o dependre de la variació de la magnitud d'entrada.
- Linealitat: correlació que hi ha entre la magnitud d'entrada i la de sortida. Es considera que dues variables quantitatives estan correlacionades quan la variació d'una suposa la variació sistemàtica de l'altre. Aquesta correlació és lineal si un increment x de la variable A es tradueix en un increment y de la variable B independentment del valor de la variable A.

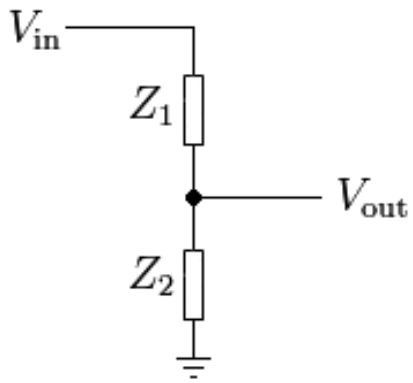
- Derives: desviació progressiva de la mesura d'una magnitud en el temps deguda a l'efecte d'una altre magnitud (temperatura, humitat,...) sobre el sensor.

Com s'ha comentat anteriorment, a continuació s'ofereix una visió de les diferents classes de sensors que es poden adquirir i s'estudien les seves característiques buscant el més adequat per tal d'assolir l'objectiu desitjat de la millor manera possible. El criteri que s'ha seguit per dur a terme aquesta classificació és la tecnologia utilitzada per els diferents sensors.

5.1.2.1 Potenciòmetres

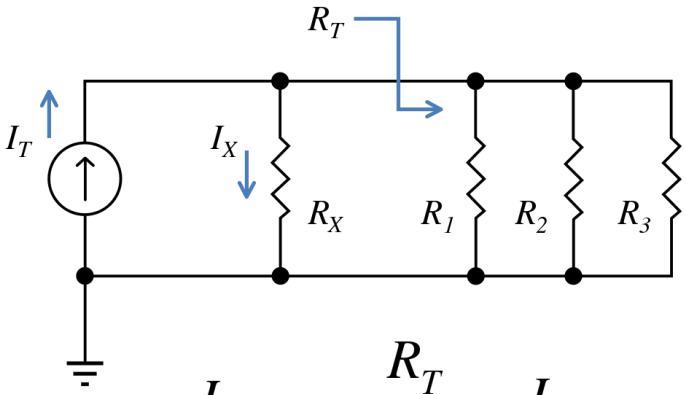
Els potenciòmetres són un component electrònic, un tipus de resistència variable.

L'ús més comú d'aquest component és com a divisor de tensió (figura 12) tot i que també pot ser utilitzat com a divisor de corrent, si les impedàncies es configuren en paral·lel enlloc de fer-ho en sèrie (figura 13):



$$V_{out} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{in}$$

Figura 12: Divisor de tensió



$$I_X = \frac{R_T}{R_X + R_T} I_T$$

Figura 13: Divisor de corrent

S'utilitzen generalment en circuits amb poc corrent, ja que en circuits amb corrents més elevades s'acostumen a utilitzar reòstats que poden dissipar majors potències.

Si el valor de la resistència variable es modifica en funció del moviment d'un component mecànic es poden utilitzar per mesurar aquest desplaçament.

Depenent de com varia la resistència, en funció del desplaçament és poden classificar en:

- Lineals: el valor de la resistència és proporcional al desplaçament.
- Logarítmics: la resistència varia de forma logarítmica en base al desplaçament.
- Sinusoïdals: la resistència és proporcional al sinus del desplaçament.

Segons el tipus de desplaçament que s'hagi de realitzar per modificar la resistència es poden classificar en:

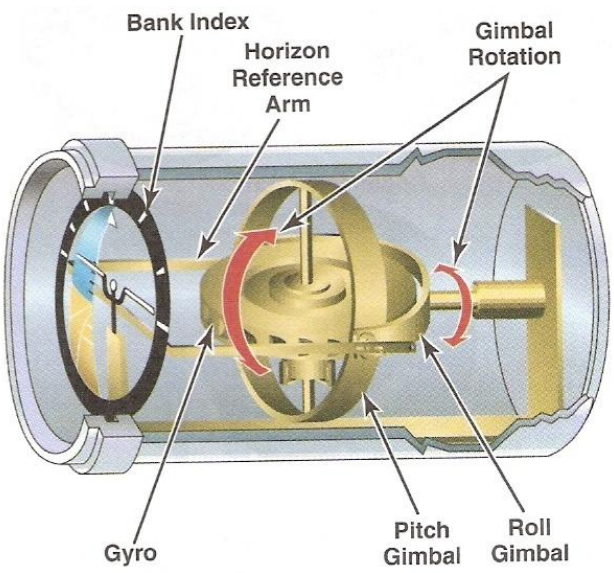
5.1.2.1.1 Potenciòmetres lliscants

Els potenciòmetres lliscants tenen la pista resistiva és recta, així que el desplaçament per modificar la resistència també ho és. Són més fràgils i ocupen més espai que els rotatius. Són utilitzats en equalitzadors, sintetitzadors, ...

5.1.2.1.2 Potenciòmetres rotatius

Els potenciòmetres rotatius mesuren la posició de gir del seu eix. Per fer-ho la seva pista resistiva és circular. Tenen una major duració i ocupen menys espai que els lliscants. Són utilitzats, per exemple, per controlar el volum de la ràdio.

5.1.2.2 Sensors inercials



Els sensors inercials (figura 14) mesuren l'acceleració i la velocitat angular utilitzant acceleròmetres, giroscopis i, també, magnetòmetres. Els primers proporcionen l'acceleració lineal (per mitjà de derivades permet obtenir la posició), els segons la velocitat angular (permeten obtenir l'orientació) i els últims ens proporcionen informació sobre el nord magnètic (correcció d'errors).

Figura 14: sensor inercial

Els sensors inercials són utilitzats en aplicacions de captura i anàlisi de moviment, ja que amb ells es pot estudiar el moviment en el pla o l'espai (depèn dels eixos que tinguin els seus sensors).

5.1.2.2.1 Giroscopis mecànics

Els giroscopis són dispositius que serveixen per mantenir o mesurar l'orientació. Basen el seu funcionament en el principi de conservació del moment angular (L) (figura 15):

$$\mathbf{L} = \mathbf{r} \times \mathbf{p} = \mathbf{r} \times m\mathbf{v}$$

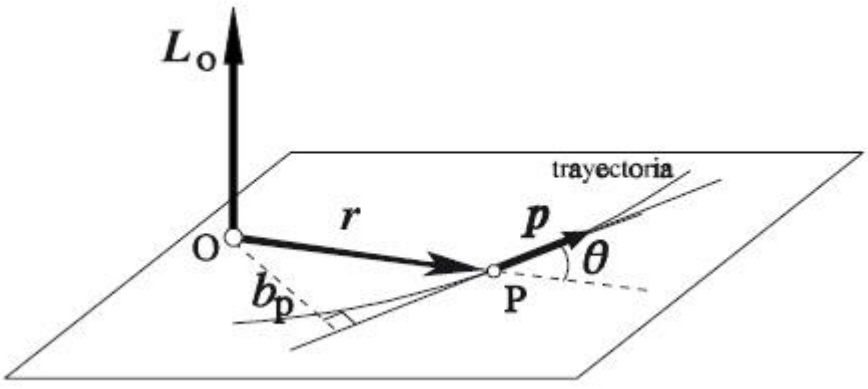


Figura 15: esquema moment angular

El moment angular d'una partícula respecte un punt O es calcula mitjançant el producte vectorial entre el radi (r) i la quantitat de moviment (p) sent aquesta última el producte entre la massa i la velocitat de la partícula.

El principi de conservació del moment angular diu que si en aquest sistema no hi intervenen forces externes L serà constant. Els giroscopis aprofiten aquest principi per calcular les forces externes que afecten el giroscopi i així poder calcular els canvis d'orientació soferts.

Els giroscopis mecànics necessiten motors per funcionar, cosa que fa que ocupin molt espai i que, per tant, no siguin pràctics per a la funcionalitat desitjada.

5.1.2.2.1.1 Giroscopis mecànics captius

Els giroscopis captius són aquells que estan subjectes a un suport. Acostumen a utilitzar un cardant (figura 16) com a suport, ja que, tot i que aquests no es mouen permeten que el giroscopi del seu interior pugui orientar els seus eixos de rotació en qualsevol direcció de l'espai.

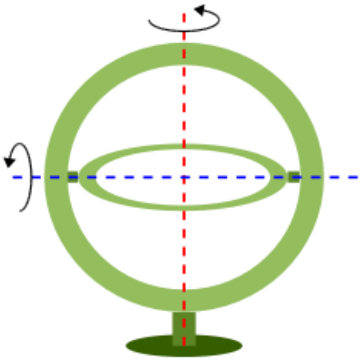


Figura 16: diagrama de moviment d'un cardant

Figura 17: giroscopi mecànic captiu en un cardant

Un giroscopi mecànic captiu (figura 17) consta bàsicament d'un disc que gira i que pot prendre qualsevol orientació. El gran moment angular associat al disc, originat per la seva alta velocitat de rotació, fa que els canvis d'orientació soferts per aquest deguts a moments externs siguin mínims. A més a més, el muntar aquest disc en un cardant fa que els moments externs que l'afecten es minimitzin i es pugui considerar que la seva orientació és gairebé fixa, encara que el seu suport es mogui. Un clar desavantatge d'aquesta classe de giroscopis és la seva necessitat de lubricació i la seva complicada alimentació.

Aquests tipus de sensors mesuren les forces externes a través de la velocitat de rotació dels eixos del cardant.

5.1.2.2.1.2 Giroscopis mecànics flotants

Els giroscopis mecànics flotants, com indica el seu nom, giren lliurement flotant a l'aire. Per fer-ho s'utilitzen dos potents imants. El primer es posa sobre la placa que fa de base i el segon es troba en el mateix giroscopi. Els dos imants són capaços de fer que el giroscopi es mantingui flotant a l'aire mentre que gira a una gran velocitat.

Aquests giroscopis són de molt difícil alimentació, perquè no estan subjectes a cap suport i floten a l'aire.

5.1.2.2.2 Giroscopis d'estat sòlid

Així com els giroscopis mecànics estan basats en el moviment d'alguna de les seves parts, els d'estat sòlid basen el seu funcionament en la vibració d'alguna d'aquestes.

El tipus més simple de sensor de rotació vibratori està format per una barra i el primer en observar i descriure el seu funcionament va ser Foucault. Aquest va lligar una barra en el mandril d'un torn i la va fer vibrar. Al fer girar el torn lentament va adonar-se'n que el pla de vibració de la barra mantenia la seva direcció.

D'aquesta manera, si es dissenya una caixa amb tres barres vibratòries apuntant en direccions perpendiculars, es pot inferir la rotació total en 3 dimensions.

Els giroscopis d'estat sòlid són menys precisos en la integració i donat que la precisió és una característica molt important en el nostre sistema, es descarta l'ús d'aquest tipus de sensors.

Giroscopis d'anell làser

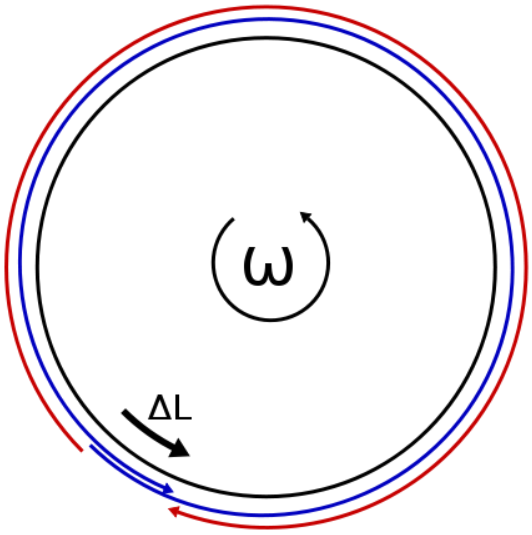


Figura 18: Configuració interferometria anell

Consisteixen en un anell làser en el qual dues ones iguals es propaguen en direccions oposades (figura 18). Es basa en l'efecte Sagnat que és un fenomen trobat en la interferometria originada per la rotació. Aquest efecte es manifesta en una configuració anomenada interferometria anell o interferòmetre Sagnat, on dos feixos de llum es disparen en la mateixa trajectòria però en direccions oposades. Per actuar com un anell aquesta trajectòria ha de tancar una àrea. Quan aquests feixos de llum tornen a arribar al punt d'entrada, gràcies a ser reflectits per miralls (figura 19), es permet que abandonin l'anell, de tal forma que es pot obtenir un patró d'interferència. La interferència entre aquests dos feixos reflexa els canvis en el patró d'ona estacionària (aquella formada per la interferència de dues ones d'igual amplitud i longitud d'ona que avancen en sentits oposats a través d'un mateix medi) i per tant, la rotació soferta per l'anell.

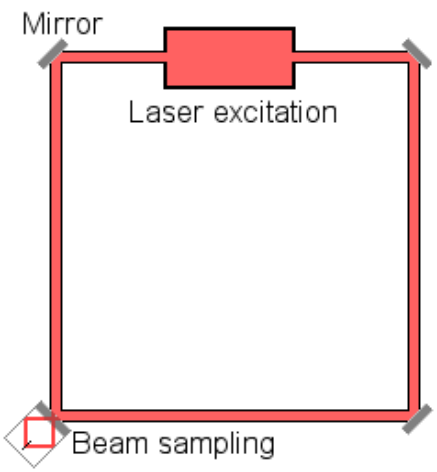


Figura 19: miralls per reflectir la llum

Els giroscopis basats en anells làser tenen l'avantatge, respecte als mecànics, de que al no tenir parts mòbils no tenen fricció i, per tant, no tenen termes de deriva inherents. Apart, també són molt més compactes, menys pesats i més exactes.

Aquesta classe de giroscopis pateixen un problema que es coneix com a "lock-in" a velocitats baixes de rotació. Quan l'anell gira a velocitats molt petites, les freqüències dels dos feixos són quasi idèntiques i la interferència ("crosstalk") que es produeix entre elles pot produir el que es coneix en anglès com "injection locking". Aquest fenomen es

refereix a l'efecte en la freqüència que es pot produir quan una oscil·lació harmònica és afectada per una segona oscil·lació amb una freqüència molt semblant. Quan la unió d'aquestes dues oscil·lacions és suficientment forta, el segon oscil·lador pot "capturar" el primer i provoca que aquest últim tingui la mateixa freqüència que el segon.

Aquest problema es pot solucionar amb el que es coneix com a "dithering" que consisteix en aplicar intencionadament una forma de soroll. La cavitat de l'anell làser és rotada de forma horària i anti-horària entorn als seus eixos, utilitzant un resort mecànic a la seva freqüència de ressonància. Això assegura que la velocitat angular del sistema estigui lluny del llindar de "lock-in". Aquesta solució encara té el problema que quan es canvia de direcció hi ha un petit interval de temps on el problema encara es pot donar.

Els giroscopis d'anell làser acostumen a ser utilitzats en vaixells i naus espacials.

5.1.2.3 Sensors òptics

En aquesta classe de sensors els senyals que es transmeten i es detecten són lluminosos.

5.1.2.3.1 Sensors de mesura de posició angular

Els sensors de mesura de posició angular mesuren l'angle de desplaçament d'un component mecànic. Per tal de saber el desplaçament angular d'aquest component es poden utilitzar codificadors rotatoris òptics que s'expliquen tot seguit.

Codificador rotatori òptic ("Optical rotary encoder")

Dispositiu òptic que serveix per transformar la posició angular d'un eix a un codi digital. S'utilitzen en la robòtica, en dispositius d'entrada d'un PC,...

N'existeixen de dos tipus:

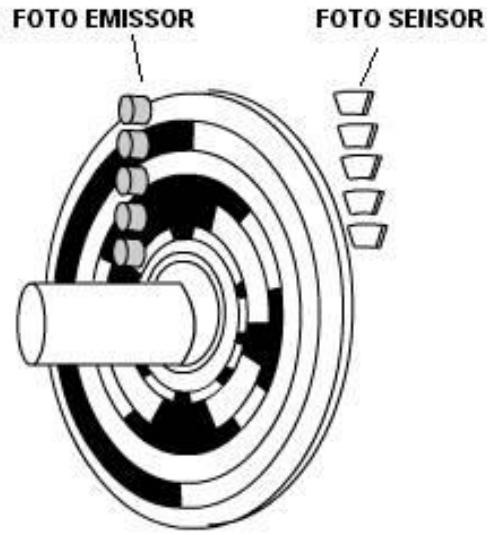


Figura 20: Codificador rotatori absolut òptic

Absoluts: aquests codificadors produeixen un codi digital únic per cada angle de rotació diferent de l'eix. Estan construïts a partir d'una fulla amb forma de disc que està fixa a l'eix del qual es vol mesurar el gir. En aquesta fulla s'hi talla un patró. Quan el disc gira, alguns foto emissors són detectats pels foto sensors i d'altres no, s'obté així un codi binari. En els codificadors rotatoris absoluts (figura 20) el patró gravat a la fulla es dissenya de forma que per cada posició possible de l'eix es creï un codi binari únic.

Cal comentar que existeix un tipus de codificadors absoluts electromecànics construïts a partir d'una fulla de metall que es posa en un disc aïllant que està fix a l'eix. Com en el cas anterior en aquesta fulla s'hi talla un patró i, sobre el radi del disc, s'hi posa una fila de contactes lliscants. Quan el disc gira, alguns contactes cauen en els forats que s'han tallat a la fulla de metall i d'altres fan contacte. Aquesta fulla està connectada al corrent elèctric i cada contacte està connectat amb un sensor elèctric per separat. D'aquesta manera s'obté un codi binari.

Incremental (relatiu): també utilitza un disc unit a l'eix, però més petit (els absoluts de dimensions reduïdes i gran resolució són difícils de construir). Aquest té marcades moltes línies (radis) (figura 21) i a través d'un interruptor òptic, quan una d'aquestes passa, es genera un pols elèctric. Aquests polsos es compten per saber quin angle ha girat l'eix.

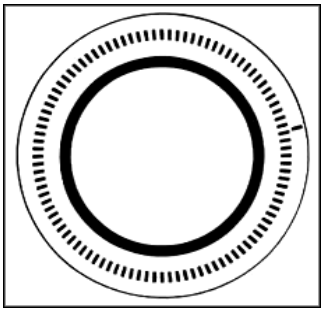


Figura 21: codificador rotatiu incremental òptic (amb detector de pas per 0).

Un problema d'aquest tipus de sensors és que, a diferència dels anteriors, només poden mesurar el canvi d'angle respecte la posició inicial de l'eix quan l'energia ha estat encesa. Això no és un problema, per exemple, per un ratolí però si per a altres aplicacions i es pot solucionar afegint un segon sensor òptic que detecti el pas per 0 (figura 21).

Un altre problema és el fet que no dóna informació sobre quina direcció està girant l'eix. Per solucionar-lo s'utilitzen el que s'anomena codificadors en quadratura, que utilitzen dues pistes codificades amb sectors de posició desfasats 90 graus. D'aquesta manera s'aconsegueix tant la posició com la direcció de rotació a través de dos canals de sortida del codificador de quadratura.

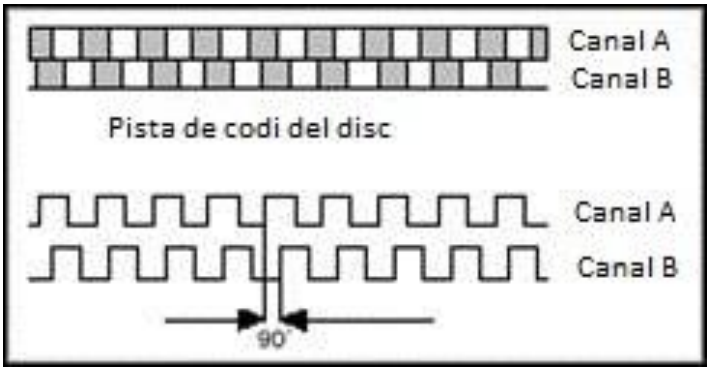


Figura 22: funcionament d'un codificador de quadratura.

Com es pot observar a la figura 22, si primer es detecta un pols en el canal A i a continuació en el B el disc està girant en sentit horari. Si al contrari, primer es detecta un pols en el canal B i després en el A el disc està girant en sentit anti-horari.

Per aconseguir les funcionalitats desitjades en tindríem prou amb un codificador incremental ja que només ens interessen els canvis de posició i no ens fa falta tenir coneixement de la posició absoluta del sensor.

5.1.2.3.2 Captura de moviments d'un cos

Antecedents

La tècnica de captura de moviments, explicada posteriorment, és comparable a l'antiga rotoscòpia la qual es considera la seva predecessora. La rotoscòpia permetia, mitjançant un dispositiu anomenat rotoscopi, utilitzar una referència filmada en viu per animar imatges dibuixant cada "frame" d'una animació sobre aquest film original. Aquest rotoscopi projectava, originalment, la filmació en viu sobre un vidre desllustrat sobre el qual el dibuixant traçava cada fotograma. Actualment, aquest equip ha estat substituït per computadores.

Tècnica

La captura de moviments és el procés de gravar moviment i traslladar-lo a un model digital.

Per tal de dur a terme la seva captura, els moviments d'un actor o d'alguna de les seves parts corporals són mostrejats moltes vegades per segon. Aquesta captura sempre és dels moviments, mai del seu aspecte visual.

Els models digitals obtinguts poden ser en 2 dimensions si s'utilitza una sola càmera o, si s'utilitzen dues càmeres, en 3 dimensions en la zona on les imatges d'aquestes es superposen.

Els sistemes òptics utilitzen les dades capturades per dues o més càmeres, calibrades per proporcionar projeccions superposades, per triangular la posició tridimensional de l'actor.

Generalment l'actor es posa marcadors, que poden ser leds o superfícies reflectants,a prop de cada articulació per identificar-ne la seva situació o l'angle que forma.

Per tal d'aconseguir rastrejar completament un intèrpret (des de tots els punts de vista) s'utilitzen un gran nombre de marcadors i, quan més grans siguin els espais, més càmeres són necessàries (generalment els sistemes tenen entre 6 i 24 càmeres però n'existeixen de fins a 300).

Aquests sistemes produeixen dades amb tres graus de llibertat per cada marcador i els altres tres, els corresponents a les rotacions, s'han d'inferir de la posició relativa de tres o més marcadors.

A continuació s'expliquen els diferents tipus de marcadors que s'utilitzen:

Marcadors passius

Aquests marcadors estan recoberts d'un material retroreflector, és a dir, que reflexa la llum de tornada cap a la font, en aquest cas prop de les lents de les càmeres, les quals

s'ajusten de forma que l'únic que es capturi sigui el reflex d'aquests marcadors ignorant així, els teixits i la pell.

A diferència dels sistemes de marcadors actius, els passius no requereixen que els actors portin cables o components electrònics. Només porten centenars de boles, generalment enganxades directament sobre la pell o sobre un vestit de licra, a través de velcro. Aquestes boles són de goma i estan recobertes de cinta reflectant que necessita ser canviada periòdicament..

Aquests sistemes poden capturar un gran nombre de marcadors a velocitats de fins a 2000 fotogrames per segon.

Els venedors d'aquests sistemes tenen software per tal de reduir el problema de solapament de marcadors, ja que tots ells tenen la mateixa aparença.

Marcadors actius

Els sistemes òptics actius (figura 23) triangulen posicions il·luminant un led cada vegada de forma molt ràpida o, il·luminant diversos leds i a través d'un software per identificar-los en funció de les seves posicions relatives (semblant a la navegació astronòmica).

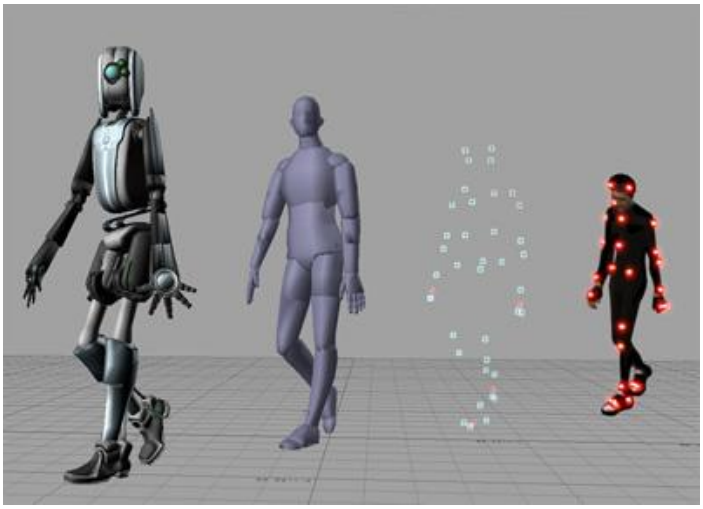


Figura 23: marcadors actius

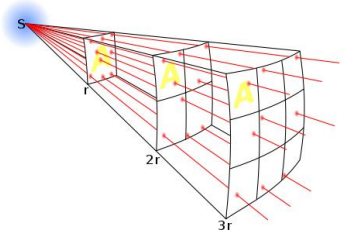


Figura 24: llei quadràtica inversa

Com que els leds no reflecteixen la llum que s'ha generat externament, els marcadors necessiten energia per emetre la seva pròpia llum. Per tal de dur a terme aquesta funcionalitat cal tenir en compte la llei quadràtica inversa (figura 24) que fa referència a alguns fenòmens

ondulatoris la intensitat dels quals disminueix amb el quadrat de la distància del centre on s'originen, entre ells la llum que és el que ens interessa. Gràcies a això es poden incrementar les distàncies i el volum per a la captura.

Es pot proporcionar la potència de cada marcador de forma seqüencial o amb patrons seqüencials en fase amb el sistema de captura. L'ús de patrons seqüencials consisteix en il·luminar un conjunt de leds de forma que aquests formin, per exemple, figures geomètriques, per tal de poder-los identificar. Aquestes tècniques proporcionen una identificació única per a cada marcador per a un determinat fotograma a costa del "frame rate" resultant (la tècnica de patrons seqüencials perjudica menys aquest paràmetre que la primera). L'habilitat d'identificar cada marcador d'aquesta manera és útil per a aplicacions en temps real com la que ens interessa en aquest projecte.

Una forma alternativa d'identificar cada marcador és fer-ho algorísmicament, tot i que això necessita més temps de procés.

Tecnologia sense marcadors

La recerca i aparició de noves tècniques en la visió per computador estan portant a un ràpid desenvolupament d'aproximacions a la captura de moviments que no requereixen l'ús de marcadors.

Aquests sistemes no requereixen que els intèrprets portin equipament especial per fer possible la captura dels seus moviments.

Algorismes especials s'estan dissenyant que permetin als sistemes analitzar múltiples fluxes òptics (aquestes tècniques, enlloc d'utilitzar diverses càmeres per tal d'obtenir una visió tridimensional, n'utilitzen una de sola però la canvien de posició ràpidament seguint un patró de forma que, dues imatges presses consecutivament, poden considerar-se del mateix instant permeten així extrapolar les 3 dimensions) i identificar formes humanes. Aquestes formes es separen en parts per tal de seguir-ne el seus moviments.

5.1.2.4 Sensors magnètics

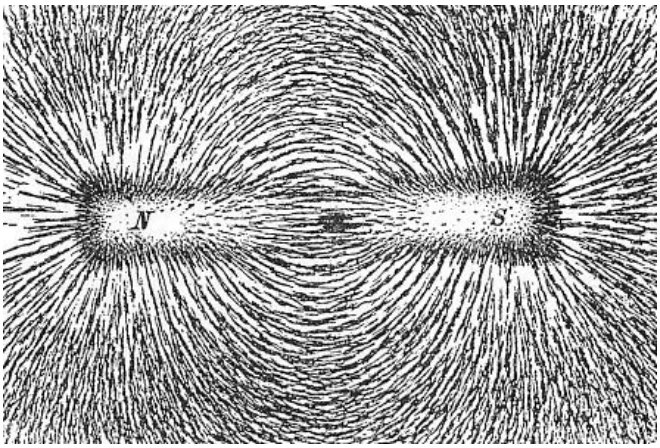
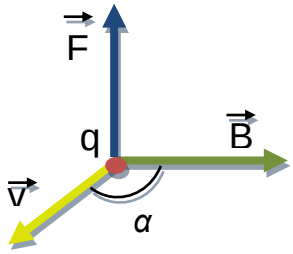


Figura 25: camp magnètic

Els sensors magnètics són aquells destinats a detectar la presència de camps magnètics (figura 25). Una simple aplicació d'exemple seria quan un robot mòbil necessita tenir coneixement del nord magnètic de la terra (brúixola) o quan certs components d'aquests requereixen detectar la presència de camps magnètics a l'entorn per evitar mal funcionaments.

Un camp magnètic és una entitat física, generada per la presència de càrregues elèctriques en moviment (ex:corrents elèctrics) o bé per la presència de partícules quàntiques amb espín, que es troba en una regió de l'espai en la qual una càrrega elèctrica (q) que és desplaça a certa velocitat (v) pateix l'efecte d'una força que és perpendicular i proporcional tant a la velocitat com al camp magnètic. El mòdul d'aquesta força es pot trobar mitjançant la següent equació (figura 26) i la seva direcció es pot obtenir amb la regla de la mà esquerra:



$$|\mathbf{F}| = q \cdot |\vec{\mathbf{v}}| \cdot |\vec{\mathbf{B}}| \cdot \sin\alpha$$

Figura 26: representació gràfica del càlcul de la força magnètica

En el Sistema Internacional d'unitats el camp magnètic es mesura en tesles (1 Weber/m^2 = 1 tesla. Weber és la unitat del SI pel flux magnètic).

Cal tenir certs factors en compte a l'hora de treballar amb camps magnètics de forma segura i precisa. Un dels més importants és que els camps magnètics i els elèctrics estan estretament relacionats i que els canvis en un dels dos poden induir canvis en l'altre (equacions de Maxwell). Un altre factor a tenir en compte és l'efecte que tenen els metalls en la propagació dels camps magnètics.

Els camps magnètics poden ser utilitzats per tal de calcular la posició i l'orientació en l'espai d'un objecte. En el sistema construït, aquest objecte serà l'útil quirúrgic utilitzat pel cirurgià.

El primer que cal estudiar per aconseguir-ho és com es propaguen els camps magnètics en l'espai. Per fer-ho és necessari observar les línies de camp magnètic (figura 27), denominades línies de força per Michael Faraday, ja que aquestes convergeixen allà on la força magnètica és major i es separen allà on aquesta és més dèbil. Aquestes línies són fonamentals a l'hora d'entendre com els electrons i els ions (partícules carregades negativament (anions, produïts pel guany d'electrons) i partícules carregades positivament (cations, produïts per la pèrdua d'electrons)) es mouen. Gràcies a aquest fet es poden utilitzar sensors magnètics per detectar aquestes línies de camp magnètic i obtenir una mesura d'orientació respecte a la font del camp magnètic.

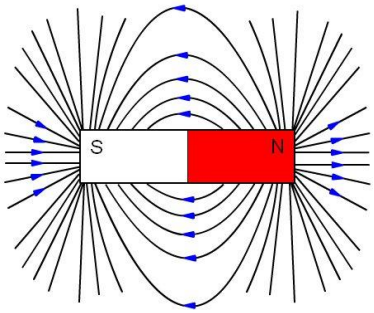


Figura 27: línies de força del camp magnètic (2D)

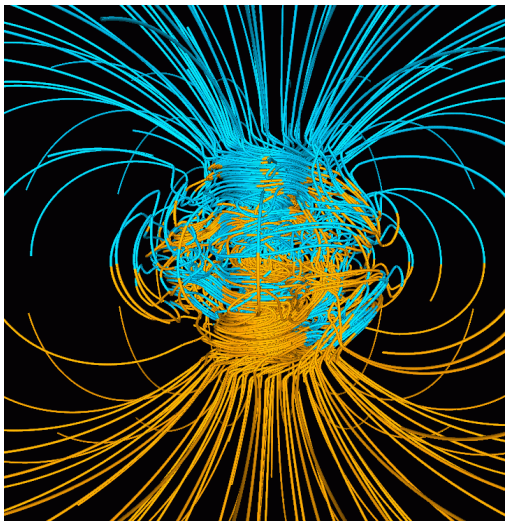


Figura 28: representació d'un model de radiació magnètica (3D)

Els camps magnètics s'expandeixen de forma esfèrica, és a dir, són ones tridimensionals. Aquest fet fa necessari aconseguir un model tridimensional de les línies de força del camp magnètic (figura 28) anomenat model de radiació.

Els models de radiació són únics per a cada camp magnètic i la seva realització és costosa, però un cop fets, permetran aconseguir la informació desitjada (posició i orientació) de forma ràpida i eficient.

A continuació es descriu el funcionament de diferents tipus de sensors magnètics buscant el més indicat per aconseguir l'objectiu desitjat.

5.1.2.4.1 Sensors de mesura de posició lineal

Els sensors de mesura de posició lineal són aquells que es basen en la mesura del desplaçament lineal d'un component mecànic.

5.1.2.4.1.1 Sensors magnètics inductius ("magnetic pickups")

Els sensors magnètics inductius estan basats en el canvi d'inductància com a conseqüència de la proximitat d'un element metàl·lic.

El sensor de proximitat de la figura 29 està format per una bobina situada sobre un imant en un encapsulat. En condicions estàtiques no hi ha flux en l'imant i per tant no s'indueix corrent a la bobina. Es produeix un canvi en el flux quan s'apropa o s'allunya un material ferromagnètic del nucli del sensor i aquest indueix un pols de corrent. La tensió d'aquest pols és proporcional a la velocitat del canvi de flux i a la distància entre el material i el sensor. Aquesta distància acostuma a ser petita (un mil·límetre) perquè el sensor sigui eficaç. La polaritat de la tensió induïda està en funció del material que s'allunya o s'apropa del nucli.

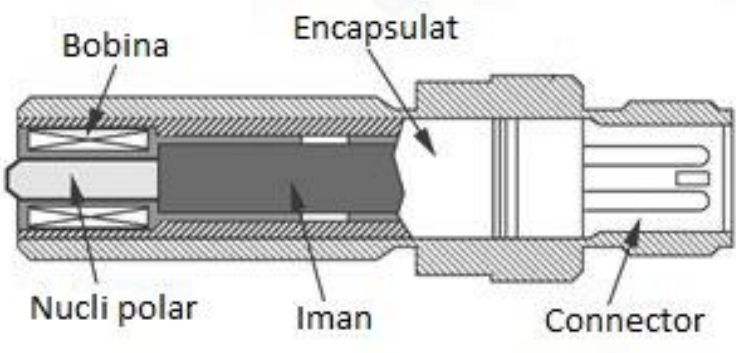


Figura 29: parts d'un sensor magnètic inductiu

Com que el que es pretén és que el cirurgià hagi de realitzar els mateixos moviments per utilitzar el sistema dissenyat que els que fa en una cirurgia convencional, l'ús d'aquests sensors és complicat. Això ve donat per la necessitat d'utilitzar-los muntats en una estructura de suport, aquest fet afectaria negativament la usabilitat i comoditat del sistema al condicionar i restringir els moviments del seu usuari.

5.1.2.4.1.2 Transformador diferencial de variació lineal ("LVDT")

Transformador utilitzat per mesurar desplaçaments lineals. Aquest transformador (figura 30) disposa de tres bobines solenoïdals disposades, extrem amb extrem, al voltant d'un tub (la bobina central s'anomena primària i les dels extrems secundàries). Lligat a l'objecte, la posició del qual es vol mesurar, es desplaça, segons l'eix del tub, un centre ferromagnètic de forma cilíndrica.

Quan un corrent altern circula a través del primari, causa un voltatge que és induït al secundari proporcionalment a la inductància (relació entre la intensitat de corrent elèctric i el flux magnètic que aquest corrent genera) mútua amb el primari.

Quan el centre ferromagnètic es mou la inductància mútua canvia i produeix una variació en el voltatge induït en el secundari. Les bobines estan connectades en sèrie, però invertides, així que el voltatge de sortida és la diferència entre els dos voltatges secundaris (d'aquí ve diferencial). Si el centre ferromagnètic està en la seva posició central les dues secundàries tenen el mateix voltatge i, per tant, el resultant és 0.

El centre es pot moure pràcticament de forma lliure (sense fricció), perquè no toca l'interior del tub, fet que fa que el LVDT sigui molt fiable.

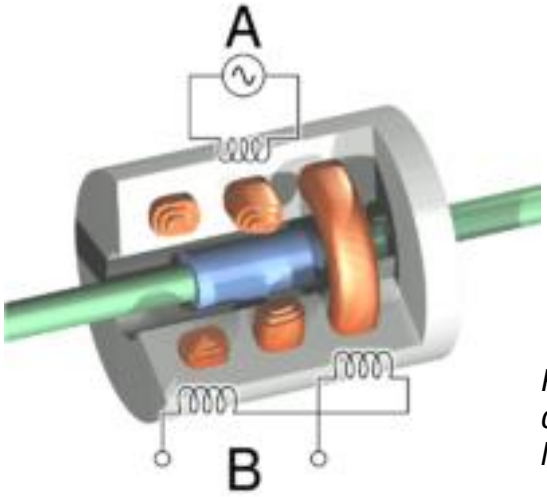


Figura 30: transformador diferencial de variació lineal

5.1.2.4.2 Sensors magnètics basats en l'efecte Hall

Aquests sensors basen el seu funcionament en l'efecte descobert pel físic nord americà Edwin Duntey Hall que consisteix en l'aparició d'un camp elèctric (anomenat camp Hall) en un conductor, quan aquest és travessat per un camp magnètic.

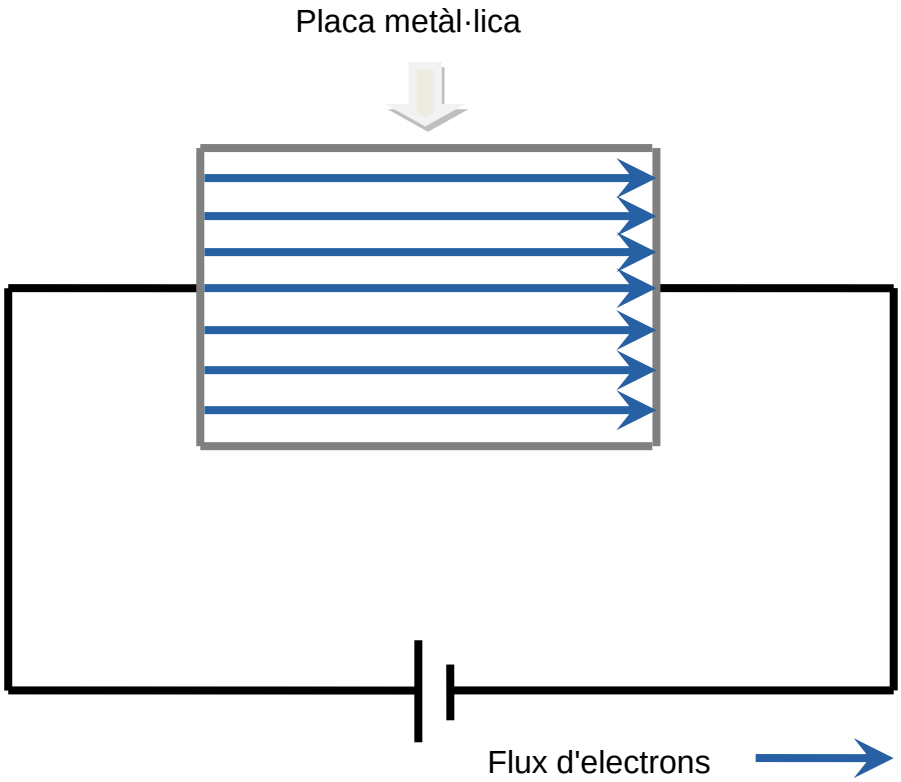


Figura 31: Comportament dels electrons en absència d'un camp magnètic

L'efecte Hall es produeix quan una placa metàl·lica, a través de la qual circula un flux de corrent elèctric, es veu influïda per un camp magnètic perpendicular al sentit en el qual es mou aquest flux. Com a resultat d'aquesta influència apareix, en la placa, un camp elèctric, perpendicular al camp magnètic i a l'elèctric generat per la bateria, anomenat camp de Hall. Aquest camp elèctric l'origina el desplaçament de les càrregues elèctriques negatives a un extrem de la placa que fa que a l'altre costat hi hagi una càrrega oposada, cosa que crea una diferència de potencial.

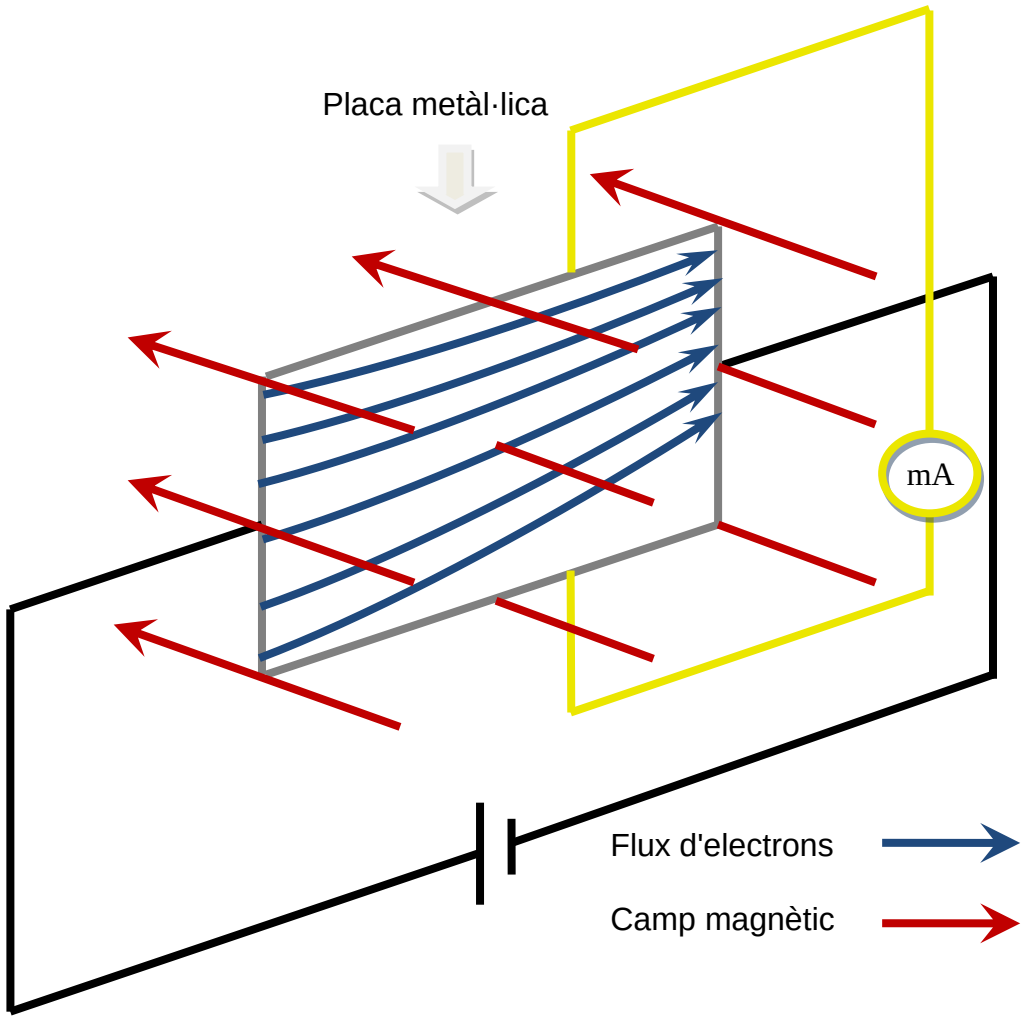


Figura 32: Comportament dels electrons en presència d'un camp magnètic

Els sensors basats en l'efecte Hall acostumen a estar formats d'un element conductor o semiconductor i un imant. Quan l'objecte ferromagnètic s'acosta al sensor es crea un camp elèctric que, al mesurar-lo, permet determinar la proximitat de l'objecte.

Els sensors d'efecte Hall han substituït els sensors inductius, explicats en l'apartat anterior, en diversos camps (ex: industria automobilística) perquè, a l'estar implementats amb un semiconductor, poden tenir electrònica integrada. Això fa que el senyal que produeixen tingui una utilització més fàcil, econòmica i directa, ja que en general ja està amplificat i condicionat.

5.1.2.4.3 Sensors de flux de camp magnètic

Una altra classe de sensors que permeten detectar la presència d'un camp magnètic són els anomenats sensors de flux magnètic.

El flux magnètic és una mesura del magnetisme i la seva unitat és el weber (volt·segon) (1 weber és equivalent al flux magnètic que al travessar una espira durant un segon produeix la mateixa força electromotriu que un volt).

Gràficament (figura 33) el flux magnètic és la quantitat de línies de camp magnètic (densitat de camp magnètic) que travessen l'espira i el seu valor es calcula a través de la fórmula següent:

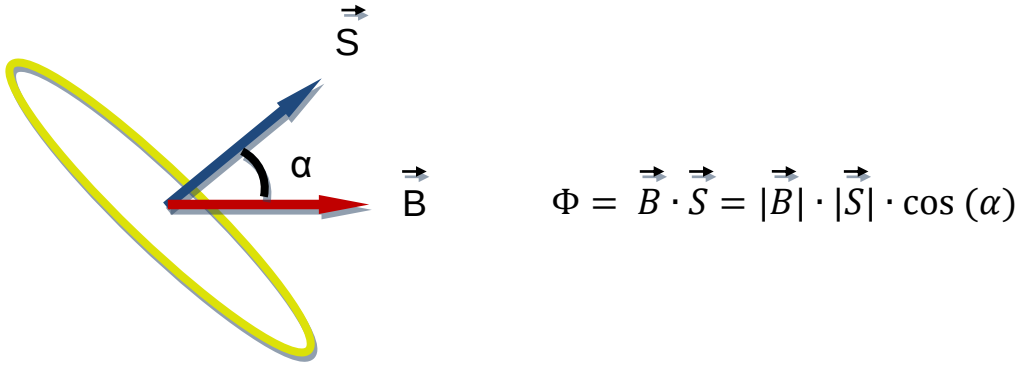


Figura 33: representació gràfica del flux magnètic a través d'una espira

És fàcil observar que quan l'angle entre els dos vectors sigui 90º o 270º el flux magnètic serà 0. D'altra banda, quan l'angle sigui 0º o 180º el flux serà màxim (el corrent induït a l'espira canviarà de sentit en funció de l'angle).

Amb el coneixement de la forma com els camps magnètics es propaguen i del funcionament dels sensors de flux magnètic, cal observar el problema que es vol resoldre per tal d'aconseguir trobar la manera d'utilitzar aquesta classe de sensors per assolir l'objectiu desitjat.

El que el sistema que s'està dissenyant requereix és la detecció de la posició i orientació d'un objecte en l'espai, és a dir, necessitem obtenir el coneixement de 6 variables en el temps (x,y,z,α,β,σ) (figura 34):

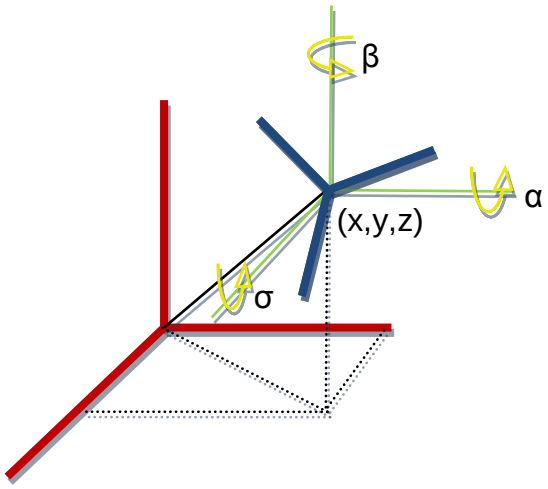


Figura 34: esquema dels eixos de coordenades del món i de l'objecte

El sistema dissenyat utilitza tres sensors de flux magnètic perpendiculars entre ells (figura 35):

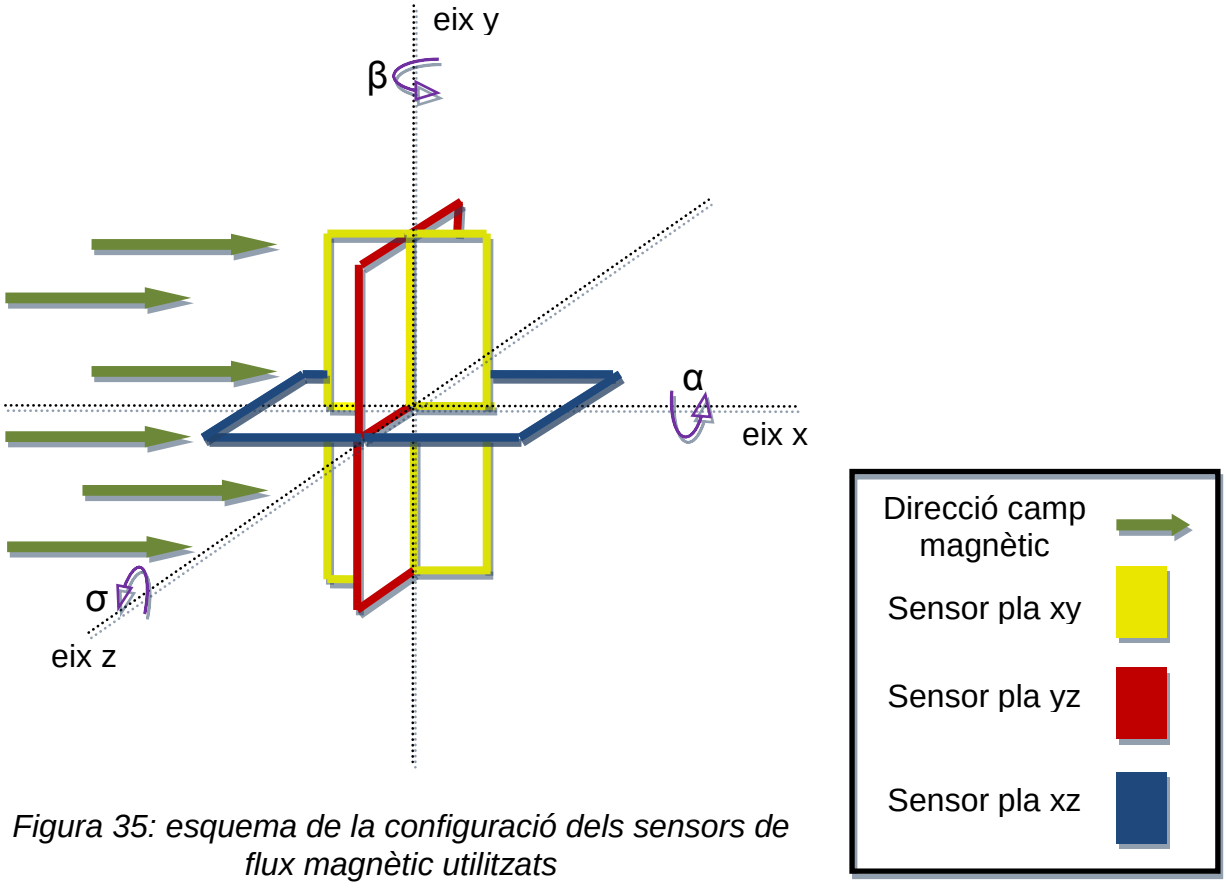


Figura 35: esquema de la configuració dels sensors de flux magnètic utilitzats

Si, per exemple, considerem un camp magnètic paral·lel a l'eix x (color verd a la figura del full anterior), és fàcil adonar-se que si rotem l'eix y en el temps els sensors xy i yz mostraran dos senyals sinusoïdals desfasats 90º (això és degut a la posició relativa entre els dos plans) (figura 36):

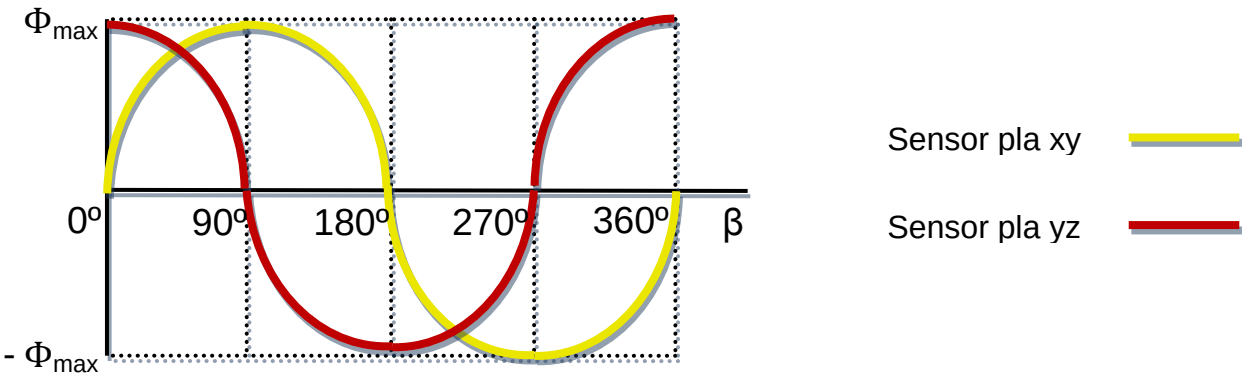


Figura 36: gràfic dels senyals captats per els sensors dels plans xy i yz en funció de l'angle β

Gràcies als valors dels dos senyals combinats es pot esbrinar el valor de β. Si només es disposa de la informació d'un dels dos sensors, per exemple el sensor del pla xy, no és pot diferenciar entre β=0º i β=180º ja que s'obté el mateix valor de flux magnètic ($\Phi = 0$).

En canvi, si es combinen les dades obtingudes a cada un dels sensors, es pot observar que:

- β=0º llavors Sensor pla $\Phi_{xy}=0$ i $\Phi_{yz}=\Phi_{max}$
- β=180º llavors Sensor pla $\Phi_{xy}=0$ i $\Phi_{yz}=-\Phi_{max}$

Conseqüentment a , com s'ha comentat, els dos sensors presenten el mateix senyal 90º desfasat es poden definir les equacions del flux de cada sensor de la següent forma:

$$\Phi_{xy} = \vec{B} \cdot \vec{S} = |\vec{B}| \cdot |\vec{S}| \cdot \sin(\beta)$$

$$\Phi_{yz} = \vec{B} \cdot \vec{S} = |\vec{B}| \cdot |\vec{S}| \cdot \cos(\beta)$$

Si ara es combinen les dues equacions anteriors es troba que:

$$\beta = arctg(\frac{\Phi \text{ xy}}{\Phi \text{ yz}})$$

Aquesta és una mesura radiogoniomètrica, ja que permet detectar la direcció de procedència d'un senyal.

Si utilitzem el mateix principi explicat amb les altres dues parelles de plans es poden esbrinar els altres angles de gir:

$$\alpha = arctg(\frac{\Phi \text{ xy}}{\Phi \text{ xz}})$$

$$\sigma = arctg(\frac{\Phi \text{ xz}}{\Phi \text{ yz}})$$

Degut a que el flux de cada pla pot valdre 0 les equacions de sobre tenen punts on són indeterminades. Per evitar aquest problema es poden utilitzar altres sistemes de coordenades com per exemple les polars.

D'aquesta manera es pot obtenir la direcció tangent a les línies de camp magnètic.

Com s'ha comentat anteriorment es busca obtenir la direcció i orientació d'un objecte en l'espai i com és obvi, no és possible trobar el valor de 6 variables amb 3 equacions.

Per tal de poder fer-ho és necessari obtenir noves equacions i per aconseguir-les el que es pot fer és utilitzar dos camps magnètics addicionals. D'aquesta forma s'utilitzaran 3 camps magnètics diferents amb la direcció entre els dos pols de cada un d'ells paral·lela a un dels 3 eixos de coordenades.

Això permet obtenir 3 fluxos magnètics diferents per a cada espira, cada un d'ells originat per un camp magnètic diferent, i s'obtenen, per tant, 9 equacions.

FILTRATGE DE SENYALS

El fet d'utilitzar 3 camps magnètics diferents fa que aparegui la necessitat de poder diferenciar-los entre ells. Això és degut a que a cada espira es detectarà el resultat dels 3 camps magnètics combinats.

Per tal de diferenciar aquests camps magnètics es poden utilitzar els anomenats filtres digitals. Aquests realitzen operacions matemàtiques en un senyal discretitzat per tal d'amplificar-ne o reduir-ne certs aspectes. En el cas tractat es diferenciaran els camps magnètics a través de les seves freqüències.

Molts filtres digitals basen el seu funcionament en la transformada ràpida de Fourier, un algorisme matemàtic que permet extreure l'espectre de freqüències d'un senyal. Mitjançant la seva manipulació es poden implementar filtres de pas de banda que permetran separar els senyals captats en funció de la seva freqüència.

La transformada ràpida de Fourier és un algorisme que permet calcular la transformada discreta de Fourier i la seva inversa en temps $O(n \log(n))$ enlloc de $O(n^2)$. La transformada discreta de Fourier descompon una seqüència de valors en components de diferents freqüències i la transformada ràpida de Fourier aconsegueix el mateix reduint el seu temps de càlcul gràcies a l'ús d'un algorisme de "divide and conquer".

Cal tenir en compte certes característiques de la transformada ràpida de Fourier per utilitzar-la, com és que el nombre de mostres del senyal que es vol transformar ha de ser una potència de 2.

La figura 37 mostra la resposta que es pot obtenir mitjançant l'ús d'un filtre digital de pas de banda:

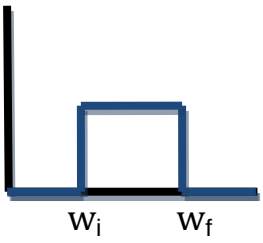


Figura 37: senyal d'un filtre digital que permet el pas de freqüències compreses entre w_i i w_f.

D'aquesta manera es pot aconseguir separar els senyals de cada un dels 3 camps magnètics.

Cal esmentar l'existència de filtres de pas de banda analògics però es descarta el seu ús, ja que presenten una menor estabilitat de funcionament en el temps. Això ve donat perquè els filtres analògics basen el seu funcionament en molts més components electrònics i cal tenir en compte que aquests donen les seves lectures dins un límit de tolerància, cosa que fa que els errors acumulatius siguin més grans.

Els filtres digitals permeten aconseguir dissenys molt més complexes (perquè els coeficients de la transformada ràpida de Fourier estan definits i guardats en una memòria) i per tant permeten definir transicions més ràpides, bandes de pas més reduïdes i un rebuig més eficient de les freqüències no desitjades.

També s'ha de tenir en compte que els filtres digitals tenen una major latència pel sistema que els analògics, ja que la dels últims es pot considerar negligible.

Com que l'exactitud i la seguretat són objectius molt importants del sistema dissenyat s'escull l'ús d'un filtre digital per tal d'aconseguir diferenciar els 3 camps magnètics utilitzats.

Un cop explicat com aconseguir filtrar els 3 camps magnètics, per tal de saber la posició i orientació en l'espai de l'objecte, s'ha de combinar la informació obtinguda de les 9 equacions trobades amb els models de radiació dels 3 camps magnètics.

Això ve donat perquè, com s'ha explicat anteriorment, per com els camps magnètics es propaguen amb la tangent respecte un únic camp magnètic en el punt on es troba l'objecte del qual seguim els moviments no en tenim prou per inferir una posició i orientació. En canvi, si combinem la informació de les tangents amb cada un dels tres camps magnètics en el punt on es troba l'objecte i els models de radiació, podem trobar la seva posició i orientació en l'espai.

Per fer-ho es realitzarà una cerca heurística sobre l'espai de solucions on, per cada punt, es té la tangent amb cada un dels tres camps magnètics (obtinguda dels models de radiació). Com a funció d'avaluació es poden comparar les tangents obtingudes gràcies als sensors de flux amb les tangents del punt estudiat de l'espai de solucions (ex: funció de desviació típica). Per generar els successors es poden agafar els punts més pròxims en l'espai de solucions i, com a funció per triar el successor, es pot fer servir la de "best neighbour" on s'escollirà aquell successor que tingui la millor avaluació.

5.2 Ús del sensor

Una part important a l'hora de dissenyar el sistema desitjat és la tria correcta de la manera mitjançant la qual el cirurgià utilitzarà el sensor escollit.

En funció del tipus de sensor que es decideixi utilitzar, és podrà fer servir d'una manera o d'una altra, ja que els requisits per a l'ús de cada un d'ells són diferents. Això vol dir que hi haurà sensors dels quals, tot i tenir característiques que requereix el sistema que s'està dissenyant, se n'haurà de descartar l'ús ja que, per la forma en que s'ha d'utilitzar, presenten desavantatges per aconseguir els objectius desitjats.

A continuació s'explicaran detalladament les diferents maneres gràcies a les quals es poden utilitzar els sensors descrits anteriorment. Per tal de poder maximitzar la usabilitat, la fiabilitat i la precisió del sistema dissenyat s'analitzaran els diferents avantatges i inconvenients que suposa l'ús de cada una d'aquestes tècniques.

5.2.1 Exosquelets

En els artròpodes (crustacis, aràcnids, insectes, ...) és l'esquelet extern que els recobreix. Proporciona el suport necessari per a l'eficàcia de l'aparell muscular i compleix una funció protectora i mecànica.
També s'anomena exosquelet a la base que secreten alguns coralls (freqüentment mineralitzada).

Exosquelets mecànics

També s'anomenen exosquelets robot i són esquelets externs (figura 38), generalment metàl·lics. Es pot entendre que són "robots que es vesteixen". Generalment ajuden a la persona que els porta a realitzar cert tipus d'activitats o moviments (per exemple córrer, aixecar pes,...).
Per poder dur a terme aquesta funcionalitat s'ajuden de sensors biomètrics que detecten els senyals nerviosos que el cervell envia als muscles del cos quan es desitja realitzar un moviment (senyals mioelèctrics). A partir d'aquests senyals el processador de l'exosquelet el fa moure ràpidament utilitzant motors situats a les seves articulacions Evidentment els exosquelets destinats a realitzar aquestes funcionalitats requereixen d'algun sistema energètic per funcionar.

Figura 38: exosquelet mecànic

Els exosquelets també poden ser utilitzats per llegir els moviments de la persona que els porta a través de l'ús de sensors a les seves articulacions, per detectar-ne els canvis de posició. Per tal de sensoritzar els moviments necessaris per dur a terme una cirurgia laparoscòpica es podrien, per tant, utilitzar sensors a les articulacions de l'exosquelet d'una mà.

Exosquelet d'una mà

Una de les possibilitats per a la construcció d'una interfície que possibiliti el comandament teleoperat d'un útil quirúrgic és mitjançant l'ús d'un exosquelet que permeti representar els moviments que realitza una mà per governar un instrument laparoscòpic.
Aquest exosquelet no haurà d'ajudar el seu usuari a realitzar cap moviment, així que no caldrà disposar de motors a les seves articulacions.
La finalitat d'aquest exosquelet serà la de permetre la lectura dels moviments del seu usuari per tal de poder reproduir-los amb el braç robòtic que controla l' instrument quirúrgic.

Per captar aquests moviments de les articulacions de l'exosquelet es poden utilitzar diversos tipus de sensors:

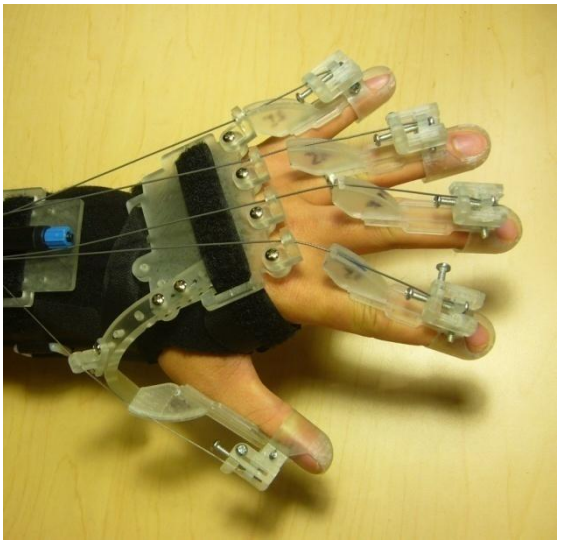


Figura 39: exosquelet d'una mà amb fils que passen per les seves articulacions

Els potenciòmetres es podrien utilitzar juntament amb uns fils lligats a diferents parts de l'exosquelet (figura 39). Tant en els lliscants com en els rotatius el desplaçament lineal dels fils es podria fer servir per modificar el valor de la resistència variable.

Els sensors magnètics de mesura de posició lineal també es podrien utilitzar amb aquesta configuració. En els transformadors diferencials de variació lineal, al moure la mà, aquests fils farien desplaçar els elements ferromagnètics variant així el voltatge induït en els secundaris de cada sensor. Això permetria detectar diferents moviments que el PC podria interpretar i transmetre al robot que controla l'útil quirúrgic.

Els sensors magnètics inductius es podrien utilitzar exactament de la mateixa manera. Al moure els fils es desplaçaria un element metàl·lic que estaria més proper o llunyà al nucli polar (imant) i que, per tant, produiria una variació en el corrent induït en la bobina.

Els sensors òptics de mesura de posició angular es podrien fer servir de forma que els fils desplacessin els codificadors rotatoris situats en les articulacions.

Aquests tipus de sensors ofereixen una alta resolució, però el fet de dependre d'una estructura mecànica fa que siguin poc segurs, per exemple, caldria tenir en compte certes mesures de seguretat en cas que es trenqués algun fil o alguna de les parts mecàniques de l'exosquelet.
A part, no es poden captar els moviments que ens interessen mitjançant aquest sistema, ja que el moviment de la mà en les tres dimensions no es podria obtenir amb càlcul del desplaçament de les articulacions de l'exosquelet.

Per aquest motiu s'ha descartat l'ús de tots els sensors que requereixen ser muntats en exosquelets.

5.2.2 Mans lliures

Una opció diferent a que el cirurgià porti els sensors necessaris és sense l'ús d'una estructura mecànica com és l'exosquelet. D'aquesta manera s'aconsegueix que el cirurgià pugui realitzar la seva feina d'una manera molt més còmode i natural, un dels objectius que té el sistema que s'està dissenyant.

Per altra banda, el fet d'utilitzar un exosquelet, també suposa desavantatges respecte aquestes tecnologies ja que, les seves parts mecàniques són molt més susceptibles a possibles averies degudes al desgast o a defectes de fabricació i el seu ús restringeix i dificulta els moviments de l'usuari.

A continuació s'expliquen els diferents mètodes de mans lliures que s'han considerat a l'hora de realitzar el disseny de la interfície.

5.2.2.1 Guants

Una de les possibilitats que eviten l'ús d'estructures mecàniques són els guants. Aquests són peces de diferents materials que cobreixen la mà i n'imiten la forma (generalment cobreixen cada dit per separat, tot i que també existeixen els guants del tipus manyopla). Els materials utilitzats en la seva confecció són diversos, per exemple, es pot utilitzar llana, cotó, làtex,...

Mitjançant el seu ús el cirurgià pot portar enganxats a la seva superfície els marcadors òptics, ja siguin passius o actius, que permetin el rastreig dels moviments de la seva mà. Cal tenir en compte que, en el cas d'utilitzar marcadors actius, es necessitaria algun tipus de sistema d'alimentació per subministrar-los energia. Aquest fet augmentaria la incomoditat en l'ús de la interfície, ja que segurament requeriria l'ús de cables que dificultarien la mobilitat de l'usuari del sistema.

Mitjançant aquesta tècnica es pot obtenir informació, tant dels desplaçaments en l'espai de la mà com dels girs involucrats en una cirurgia laparoscòpica.

5.2.2.2 Útils inalàmbrics

L'ús d'útils inalàmbrics on col·locar els sensors també permet el rastreig dels moviments involucrats en la realització d'una cirurgia mínimament invasiva. Els sensors que es podrien utilitzar serien els basats en la detecció de camps magnètics, els inercials i els òptics de captura de moviments.

A continuació s'expliquen els dos tipus d'útils inalàmbrics considerats en el disseny del sistema.

5.2.2.2.1 Útils específics

Un tipus d'útil específic on es podria col·locar el sensor seria el propi instrument laparoscòpic (figura 40). D'aquesta manera el cirurgià tindria les mateixes sensacions que quan realitza una cirurgia mínimament invasiva sense l'ajuda d'un braç robòtic.

Figura 40: exemple d'útil específic

També es podrien utilitzar altres útils, com per exemple un bolígraf. L'ús d'aquest no permetria que el cirurgià estès en una situació tant semblant a una cirurgia mínimament invasiva no teleoperada, però el seu suport i moviments són més àgils i còmodes.

5.2.2.2.2 Mànec sensoritzat

Un altre element en el qual és possible col·locar els sensors, és una peça que es subjecti amb la mà durant l'ús del sistema.

De la mateixa manera que el bolígraf en la secció anterior, aquest tipus d'útil no permet que el cirurgià estigui en una situació tant exacta a una cirurgia no teleoperada però presenta un suport i un moviment àgils i còmodes.

5.3 Elecció de la tecnologia

Amb l'objectiu de dissenyar una interfície que proporcioni al sistema unes millors prestacions, en aquest apartat es comentaran aquelles raons que han portat a escollir la tecnologia basada en els sensors de flux magnètic.

Un factor decisiu a l'hora de realitzar l'elecció de la tecnologia són les limitacions que comporta la necessitat d'integrar el sensor en un dispositiu de comandament manual d'un sistema mestre/esclau.

Com s'ha comentat en l'apartat 5.2, per diferents problemes es descarta l'ús d'un exosquelet on muntar el sensor. L'inconvenient més important que presenta és que no permet obtenir coneixement de la posició en l'espai, a més a més, el seu ús perjudica de forma important la comoditat, la seguretat i la durabilitat del sistema dissenyat. Això fa que els potenciòmetres (tant lliscants com rotatius), els sensors òptics de mesura de posició angular (codificadors de posició angular) i els sensors magnètics de posició lineal (tant els inductius com els transformadors diferencials de variació lineal) quedin descartats.

Un dels motius que fa que es descartin els sensors inercials és que deriven la posició de l'acceleració. Aquest fet perjudica la seva precisió, cosa que no es pot permetre en un sistema com el que s'està dissenyant. Un altre inconvenient que presenten és que els giroscopis utilitzats requereixen alimentació i aquesta és complexa i fa que augmenti el seu pes i dimensió.

Els giroscopis mecànics presenten apart altres inconvenients, com que tenen termes de deriva inherents que perjudiquen la seva precisió.

Els giroscopis d'estat sòlid, tot i no tenir termes de deriva inherents, ser més compactes i també menys pesats, són poc precisos en la integració.

Els sensors òptics utilitzats per la captura de moviments són pràctics per aplicacions on, per exemple, es necessita obtenir la posició de les articulacions, però no per l'aplicació que s'està dissenyant. Això és degut a que els hi falta precisió i per tant, fiabilitat, per un sistema que té per objectiu ser utilitzat en procediments tant delicats com són les intervencions quirúrgiques laparoscòpiques. Per això s'ha descartat el seu ús. Cal esmentar que els sistemes que utilitzen marcadors actius requereixen energia i el seu subministrament perjudicaria la comoditat del sistema.

Els sensors magnètics basats en l'efecte Hall no permeten obtenir tota la informació necessària, ja que amb ells no es pot obtenir coneixement de la posició en l'espai i per tant no són adequats per al sistema dissenyat.

Els sensors de flux magnètic són els més adequats per assolir els objectius del sistema. Això és degut a que no requereixen cap exosquelet per utilitzar-los i, per tant, permeten obtenir un grau elevat de seguretat i comoditat a l'hora de fer ús del sistema construït. Aquests sensors es poden muntar en un útil quirúrgic idèntic als utilitzats en un procediment convencional o en un mànec sensoritzat, de forma que la interfície sigui el màxim d'intuïtiva possible per a l'usuari. Aquest fet reduirà el temps d'aprenentatge requerit per utilitzar el sistema dissenyat ja que, permetrà aprofitar al màxim les habilitats que el cirurgià ja posseeix.

Un altre dels avantatges més rellevants que es deriven de l'ús d'aquesta tecnologia és que aquest tipus de sensors són molt precisos, requisit molt important del sistema que s'està construint.

Per tots aquests motius s'ha decidit dissenyar una interfície que utilitzi sensors de flux magnètic.

Cal no oblidar que l'ús d'aquesta tecnologia imposarà restriccions que s'hauran de tenir en compte a l'entorn on s'utilitzi la interfície. Algunes de les més importants són que s'hauran de tenir en compte els objectes metàl·lics i els camps magnètics rellevants que siguin presents en l'entorn del sistema, ja que aquests poden afectar als camps magnètics del sensor. És important recordar, també, que els corrents elèctrics generen camps magnètics.

A continuació s'ofereix una taula on es resumeixen els principals avantatges i inconvenients derivats de l'ús de cada una de les diferents tecnologies estudiades.

			AVANTATGES	INCONVENIENTS
POTENCIÒMETRES	Lliscants			- Muntatge en un exosquelet: no permeten obtenir posició, poca comoditat, poca fiabilitat i seguretat - Gran tamany - Fragilitat
	Rotatius		- Menor tamany que els lliscants - Menys fragilitat que els lliscants	- Muntatge en un exosquelet: no permeten obtenir posició, poca comoditat, poca fiabilitat i seguretat
INERCIALS	Giroscopis mecànics	Captius i flotants		- Requereixen motors: pesats, de gran tamany i de difícil alimentació - Poca precisió ja que la posició es deriva de l'acceleració i que tenen termes de deriva inherents (components mecànics)
	Giroscopis d'estat sòlid		- No tenen termes de deriva inherents - Més compactes i menys pesats que els mecànics	- Poc precisos en la integració i en que la posició es deriva de l'acceleració
ÒPTICS	Mesura de posició angular (codificadors incrementals)		- Alta resolució	- Muntatge en un exosquelet: no permeten obtenir posició, poca comoditat, poca fiabilitat i seguretat
	Captura de moviments	Marcadors actius		- Poc fiables - Requereixen alimentació (poca comoditat)
		Marcadors passius i sense marcadors	- Tamany i pes reduït - Alta comoditat	- Poc fiables
MAGNÈTICS	Posició lineal	Inductiu i "LVTD"	- Alta resolució	- Muntatge en un exosquelet: no permeten obtenir posició, poca comoditat, poca fiabilitat i seguretat
	Efecte Hall		- Tamany i pes reduït	- No permeten aconseguir la informació de posició
	Flux magnètic		- Alta seguretat, comoditat i precisió - Tamany i pes reduïts	- Poden veure's afectat per grans camps elèctrics i magnètics presents a l'entorn

6 IMPLEMENTACIÓ DE L'ESTACIÓ DE TREBALL EXPERIMENTAL

Un dels objectius d'aquest projecte consisteix en el desenvolupament d'un prototip de laboratori experimental d'un sistema de teleoperació assistida que permeti l'estudi de la realització d'intervencions de cirurgia mínimament invasiva.

La informació sobre aquesta estació es troba separada en tres seccions. En la primera secció, denominada hardware, es parla amb detall dels components físics que composen el sistema desenvolupat i de com es comuniquen entre ells. En la segona, software, s'expliquen les diferents aplicacions que utilitza el sistema construït i els paràmetres de control que aquestes implementen i que fan que el sistema sigui adaptable, útil i segur. En l'última secció d'aquest apartat s'estudia l'estació de treball construïda (figura 41) i s'avalua el seu funcionament.

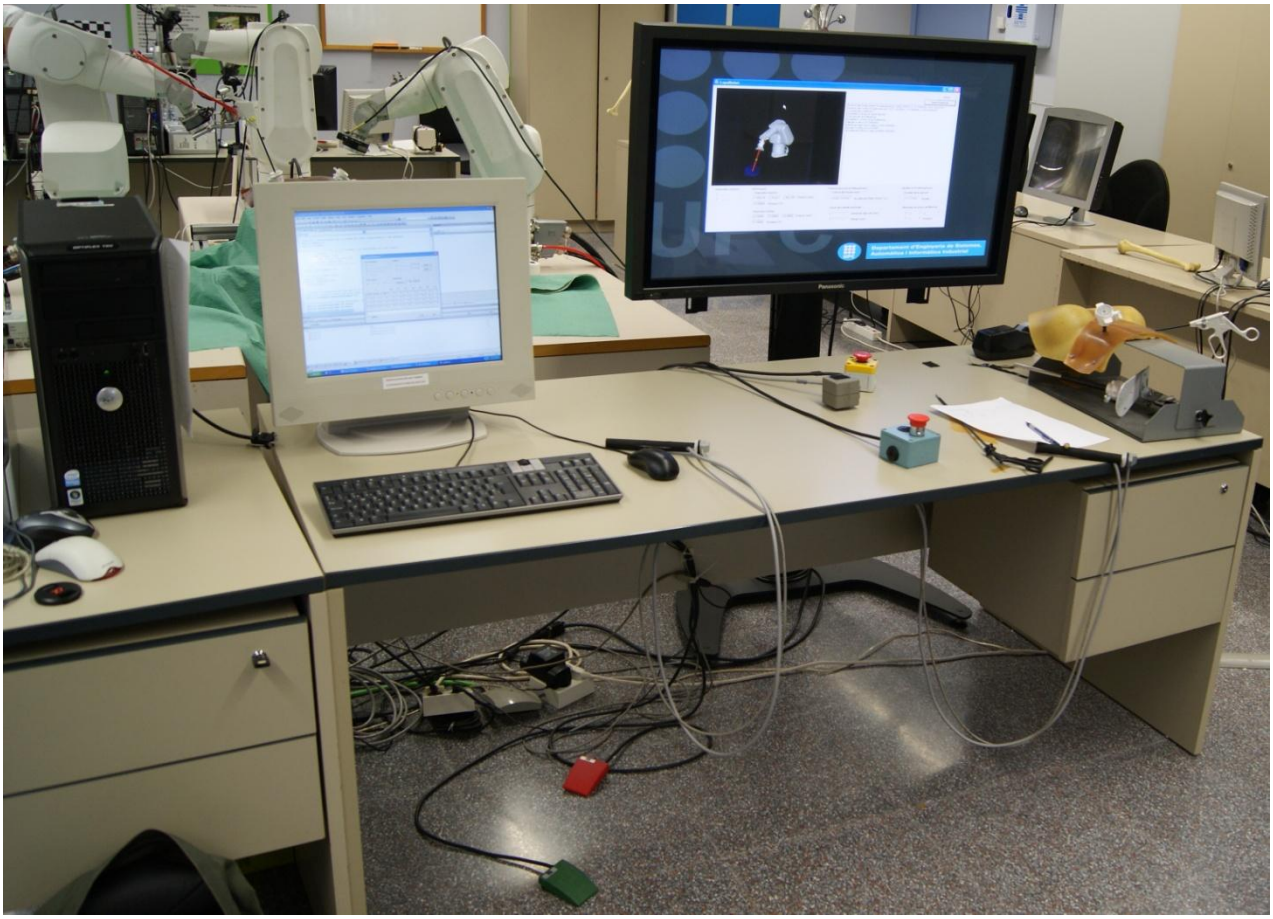


Figura 41: Estació de treball

6.1 Hardware

En aquesta secció s'expliquen els diferents components físics (elèctrics, electrònics, electromecànics,...) utilitzats per construir el prototip d'un sistema per a la teleoperació assistida d'aquest projecte. A la figura 42 se'n mostra un esquema general:

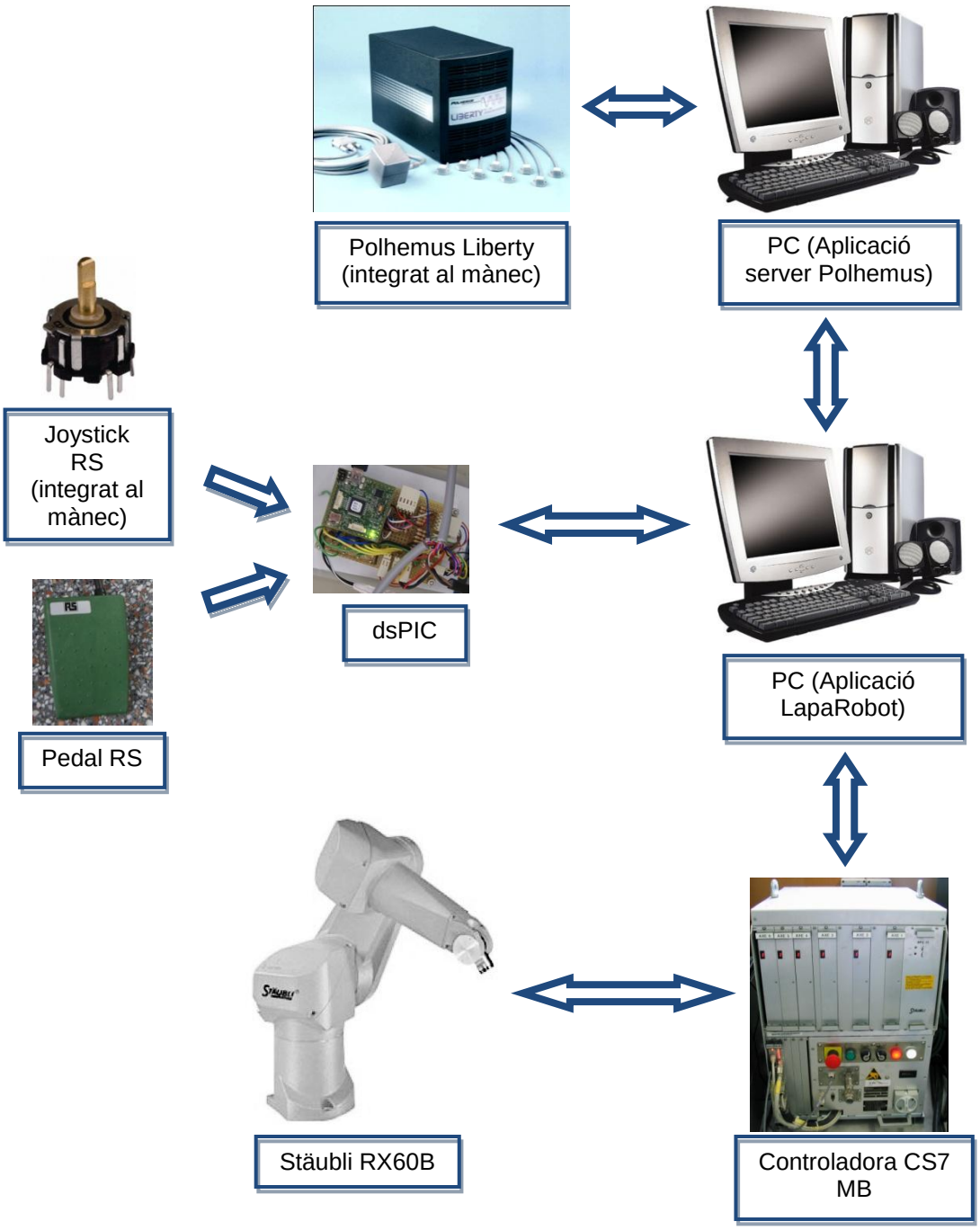


Figura 42: esquema dels diferents components del sistema.

Es diferencien tres apartats importants. En el primer es parla d'aquells elements utilitzats per la construcció del dispositiu mestre que capta els moviments que posteriorment es transmeten al braç robòtic, es a dir, la interfície d'usuari.

En el segon apartat es tracten tots els temes relacionats amb el braç robòtic utilitzat com a dispositiu esclau fent èmfasis a la seva estructura mecànica.

En l'últim apartat es parla d'aquells components necessaris per aconseguir la comunicació entre els dispositius mestre i esclau. Gràcies a aquesta comunicació poden transmetre's informació de l'un a l'altre gairebé en temps real (temps mínim: 16 milisegons).

6.1.1 Estació mestre

Per tal de captar els moviments que realitza un cirurgià a l'hora de dur a terme una intervenció quirúrgica mínimament invasiva i poder transmetre'ls al dispositiu esclau és necessari un dispositiu mestre el màxim d'adequat possible. Com s'ha explicat en l'apartat 5, s'ha decidit utilitzar un sensor basat en els sensors de flux magnètic integrat en un mànec que disposa d'un botó tipus joystick i, en la següent secció, es dóna una descripció dels components utilitzat en aquest prototip.

En la secció 6.1.1.2 es parla del pedal utilitzat per poder fixar l'offset desitjat respecte el dispositiu esclau.

6.1.1.1 Sensor magnètic i integració

El sensor utilitzat és el model Liberty 240/8 de la marca Polhemus (figura 43). Com que la tecnologia utilitzada pels sensors magnètics d'aquest fabricant està patentada, no es pot trobar informació sobre el seu funcionament intern. Per això en l'apartat 5.1.2.4.3 s'ha proporcionat una descripció aproximada d'un sistema que es capaç de dur a terme la mateixa funcionalitat.

Figura 43: Sensor Polhemus Liberty

El model Liberty és un dels sistemes de seguiment amb 6 graus de llibertat més ràpid i precís disponible actualment. Pot assolir una freqüència d'actualització de 240Hz, cosa que el fa molt adient per a l'ús en sistemes que funcionen en temps real, com el que s'està construint.

Els sensors utilitzats incorporen tecnologia en processadors de senyals digitals ("Digital signal processor (DSP)"). Aquests processadors són capaços d'executar algorismes de processament digital sobre un senyal d'entrada en temps real (generalment aquest senyal d'entrada és analògic i per tant es requereix un conversor analògic/digital). Gràcies a aquest factor es pot aconseguir una millor relació senyal/soroll, cosa que millora l'estabilitat, la resolució, la velocitat i l'abast del sistema.

Aquest model no es veu afectat per la xarxa elèctrica de la instal·lació on es fa servir i cada un dels sensors és capaç de detectar si existeix distorsió magnètica a l'entorn i alertar l'usuari. Aquests fets milloren de forma rellevant la seguretat del sistema dissenyat, una de les seves característiques primordials.

Un altre característica molt important del sistema de seguiment utilitzat és que sempre manté la freqüència d'actualització, fet molt important per poder assegurar la seva estabilitat.

Per últim, cal destacar la gran resolució i precisió que proporciona el model Liberty de Polhemus tant en la posició com en l'orientació.

A continuació és presenten algunes taules amb les especificacions tècniques més importants del model utilitzat:

Graus de llibertat	6
Nombre de sensors	1-8
Freqüència d'actualització	240 Hz (per sensor)
Precisió en una posició estàtica	0.0762 cm
Precisió en una orientació estàtica	0.15° (Root mean square)
Latència	3.5ms
Interfície	RS-232 or USB (both included)
Compatibilitat SO	GUI/SDK 2000/XP

Distància (cm)	Resolució de la posició (cm)	Resolució de la orientació (graus)
30.48	0.000127	0.0004º
60.96	0.000508	0.0014º
91.44	0.00254	0.0048º
121.92	0.0127	0.0117º
182.88	0.07874	0.060º
304.8	0.3683	0.280º

Per més informació consultar l'apèndix III on s'hi pot trobar la descripció i les especificacions del sistema o www.polhemus.com on en la secció de suport es pot trobar el document "LIBERTY Manual" on s'hi troba el manual d'ús d'aquest model.

Cal esmentar que aquest model del Polhemus, del qual es disposa al laboratori de robòtica de la UPC, no és inalàmbric, però que existeix el model Liberty Latus que ofereix les mateixes prestacions i si que ho és.

Finalment s'ha decidit integrar el Polhemus en un mànec (figura 44), aquest disposa d'un control addicional tipus joystick de 8 posicions (marca RS, codi=516-306) que s'ha utilitzat per implementar la rotació del dispositiu esclau. S'ha optat per aquesta solució ja que es considera que aquest mètode és més còmode que haver de realitzar la rotació

del sensor físicament. D'aquesta manera, quan l'usuari vulgui fer rotar l'instrument laparoscòpic sostingut per el dispositiu esclau, només haurà de prémer el joystick en la direcció corresponent.

Figura 44: Mànec sensoritzat (Sensor Polhemus: part esquerra del mànec. Joystick (color daurat): part dreta del mànec)

6.1.1.2 Ús del pedal i funció

Figura 45: Pedal RS

Per tal de poder fixar un offset del dispositiu mestre respecte el dispositiu esclau s'ha decidit utilitzar un pedal. La seva funcionalitat és la de tallar el flux d'informació entre els dos dispositius quan l'usuari no el premi amb el peu, cosa que permetrà realitzar un canvi de posició del mestre sense que afecti l'esclau. Això evitarà la necessitat del metge d'haver de treballar en posicions incòmodes durant la cirurgia i facilitarà la seva feina de forma significativa a l'augmentar la comoditat i la usabilitat del sistema. D'aquesta manera quant el practicant arribi al límit d'una de les seves articulacions podrà deixar anar el pedal, tornar a una posició natural, i continuar treballant respecte la seva posició anterior sense que el braç robòtic realitzi cap moviment.

Aquest dispositiu també augmentarà la seguretat del sistema ja que el dispositiu esclau no realitzarà moviments a no ser que l'usuari estigui prement el pedal.

A la figura 45 s'ofereix una imatge del pedal utilitzat, de la marca RS (codi RS: 325-2686)

6.1.2 Dispositiu esclau: braç robòtic

Figura 46: Stäubli RX60B

Per tal de realitzar una cirurgia teleoperada, a part del dispositiu mestre, es requereix el dispositiu esclau. Aquest rebrà les accions realitzades per el primer i les reproduirà.

En el sistema construït s'utilitzà un braç robòtic, model Stäubli RX60B (figura 46), per tal de reproduir les accions realitzades per el cirurgià que sustenta el mànec sensoritzat. Aquest model està compost per diversos elements connectats per articulacions, el moviment dels quals és generat per servomotors sense escombretes acoblats a resolvers (tipus de transformador elèctric rotatori utilitzat per mesurar l'angle de rotació). Els motors sense escombretes presenten avantatges respecte els que les porten, ja que aquestes, utilitzades per establir la connexió (transmetre corrent) entre una part mòbil i una de fixa, introdueixen fregament i això fa que es desgastin i per tant s'hagin de canviar periòdicament. Aquest model, a més a més, presenta un innovador sistema de recompte que permet saber totes les posicions absolutes de les articulacions a cada instant de temps, això evita la necessitat de realitzar maniobres de calibratge a cada arrancada.

El RX60B disposa de 6 graus de llibertat, tots ells rotacions. Amb aquests es poden representar perfectament els 4 que intervenen en el moviment de l'instrument laparoscòpic en un intervenció mínimament invasiva (explicats a la secció 5.1.1).

És un braç robòtic molt versàtil, ja que pot utilitzar-se per a moltes aplicacions (assemblatge. manipulació de càrregues, aplicacions de control, revisió i neteja de peces, ...) i també permet ser fixat al terra, a una paret o al sostre.

A l'annex IV se n'ofereix informació addicional (dimensions, volum de treball, capacitat de càrrega, graus de llibertat,...)

Cal dir que, el model de braç robòtic utilitzat a l'estació construïda, podria canviar-se sense necessitat de realitzar canvis a la resta d'elements del sistema.

Element terminal

El sistema de cirurgia assistida dissenyat permet que el braç robòtic (dispositiu mestre) utilitzi diferents elements terminals com són els instruments laparoscòpics (ref. pàgina 36). Aquests elements terminals permetran realitzar diferents funcions com per exemple tallar, subjectar, il·luminar, gravar, ...

S'utilitza una articulació passiva en aquest element terminal per evitar possibles danys ocasionats per moviments involuntaris del pacient i permetre que l'instrument laparoscòpic es posicioni lliurement en els eixos x i y. A la figura 47 és mostra un esquema d'aquest conjunt robot - articulació passiva - instrument:

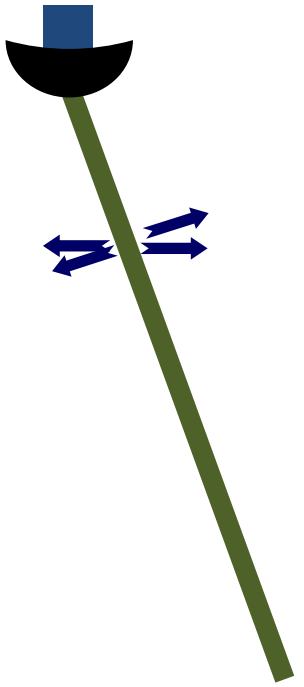


Figura 47: esquema robot - articulació passiva – instrument laparoscòpic

6.1.2.1 Controlador CS7 MB

El model CS7 MB és un dels armaris controladors compatibles amb el model de braç robòtic Stäubli i utilitza una unitat de control Adept MV5. Aquest model dirigeix el robot utilitzant amplificadors de potencia a cada eix del braç que subministren la tensió necessària per al seu funcionament.

La comunicació necessària per enviar les ordres al controlador es pot realitzar a través d'una connexió Ethernet o de ports RS-232 o RS-422/485. Aquestes ordres s'han de proporcionar amb el llenguatge de programació textual de alt nivell V+, el qual va ser desenvolupat l'any 1989 per Adept Technology i és una evolució del VALII (a l'apartat 4 s'ha fet referència al llenguatge VAL). Algunes de les característiques més rellevants d'aquest llenguatge són:

- Fiabilitat: molt pràctic per a sistemes que han de respondre a situacions imprevistes.
- Adaptabilitat: la modificació, millora i ampliació de programes requereix poc esforç.
- Llenguatge ràpid i que presenta un alt nivell d'interpretació en les aplicacions.
- Transportabilitat: els programes poden ser desenvolupats en un PC i introduïts posteriorment en el controlador.

Aquest llenguatge ofereix una manipulació en els espais cartesians o articulars molt senzilla i permet enviar al robot punts en l'espai cartesià allunyats del lloc on es troba l'element terminal, deixant que el controlador s'encarregui de generar la trajectòria i convertir-la a l'espai articular. Per tant el llenguatge V+ permet treballar en l'espai cartesià deixant que la cinemàtica inversa del robot (aquesta permet determinar la posició de les articulacions que fan que l'element terminal quedi en una posició determinada) es tracti de forma interna.

Una altre avantatge d'aquest llenguatge és que està pensat per a tasques de teleoperació i ofereix un sistema de sincronització per assegurar la consistència de les

instruccions executades. Aquest es basa en que quan es rep una instrucció de moviment es posa en una cua d'execució i quan el robot està apunt d'arribar a la posició desitjada reenvia un paquet amb la posició de l'element terminal. Aquest sistema de control, on la controladora espera respostes, fa que la trajectòria generada sigui una interpolació de punts propers ("continuous-path") passats a l'esclau com increments petits de la posició actual.

Per a l'enviament de comandes al robot s'ha utilitzat un intèrpret específic d'alt nivell desenvolupat pel grup GRINS de ESAII. Aquest s'executa en el bucle principal de la controladora i tradueix les instruccions enviades per Ethernet al llenguatge V+. Això permet al programador oblidar-se de la inicialització de paràmetres del robot i de la correcta execució de les instruccions de baix nivell.

6.1.3 Comunicació

Per tal que els dispositius mestre (mànec sensoritzat) i esclau (braç robòtic) es puguin transmetre informació, és necessària la comunicació entre diferents parts del sistema. Cal tenir en compte que, el sistema que es vol implementar, pretén suportar la teleoperació, es a dir, que el cirurgià que governa el dispositiu mestre pugui estar en una ubicació diferent a la del dispositiu esclau.

Aquesta secció s'ha dividit en dos apartats. En el primer apartat, models de comunicació, s'expliquen els diferents models de comunicació que existeixen per permetre que un dispositiu mestre i un dispositiu esclau mantinguin un flux d'informació. En el segon, components del sistema, es parla de tots aquells elements utilitzats per tal d'establir la comunicació entre tots els dispositius hardware que intervenen en el sistema construït.

6.1.3.1 Models de comunicació

A continuació es comenten els dos models de comunicació que es poden donar entre un dispositiu mestre i un dispositiu esclau.

6.1.3.1.1 Unilateral

Una comunicació es defineix com unilateral quan només hi ha un membre que transmeti informació.

En el sistema dissenyat la comunicació entre el mestre i l'esclau és unilateral, perquè l'únic dispositiu que transmet informació a l'altre és el mestre, és a dir, el flux d'informació és unidireccional.

6.1.3.1.2 Bilateral

En aquest tipus de comunicació, tant el mestre com l'esclau transmeten i reben informació.

Una possible ampliació del sistema dissenyat, utilitzant un sistema de comunicació bilateral entre els dispositius mestre i esclau, seria utilitzar sensors de força en l'element terminal per tal de captar la força que aquest fa i transmetre-la per mitjà d'algun element mecànic a la mà del cirurgià (hàptica). D'aquesta manera s'aconseguiria que el metge tingués una idea de la pressió que el braç robòtic està exercint sobre el punt de treball a través d'algun sistema de realimentació i així aconseguir una major similitud amb una cirurgia no teleoperada i, per tant, facilitaria la feina del cirurgià.

6.1.3.2 Components del sistema

En aquesta secció es descriuen els components necessaris per tal d'establir les comunicacions entre el sensor magnètic i el braç robòtic. A continuació s'ofereix un

diagrama (figura 48) de les diferents parts que formen el sistema dissenyat i que per tant s'han de comunicar:

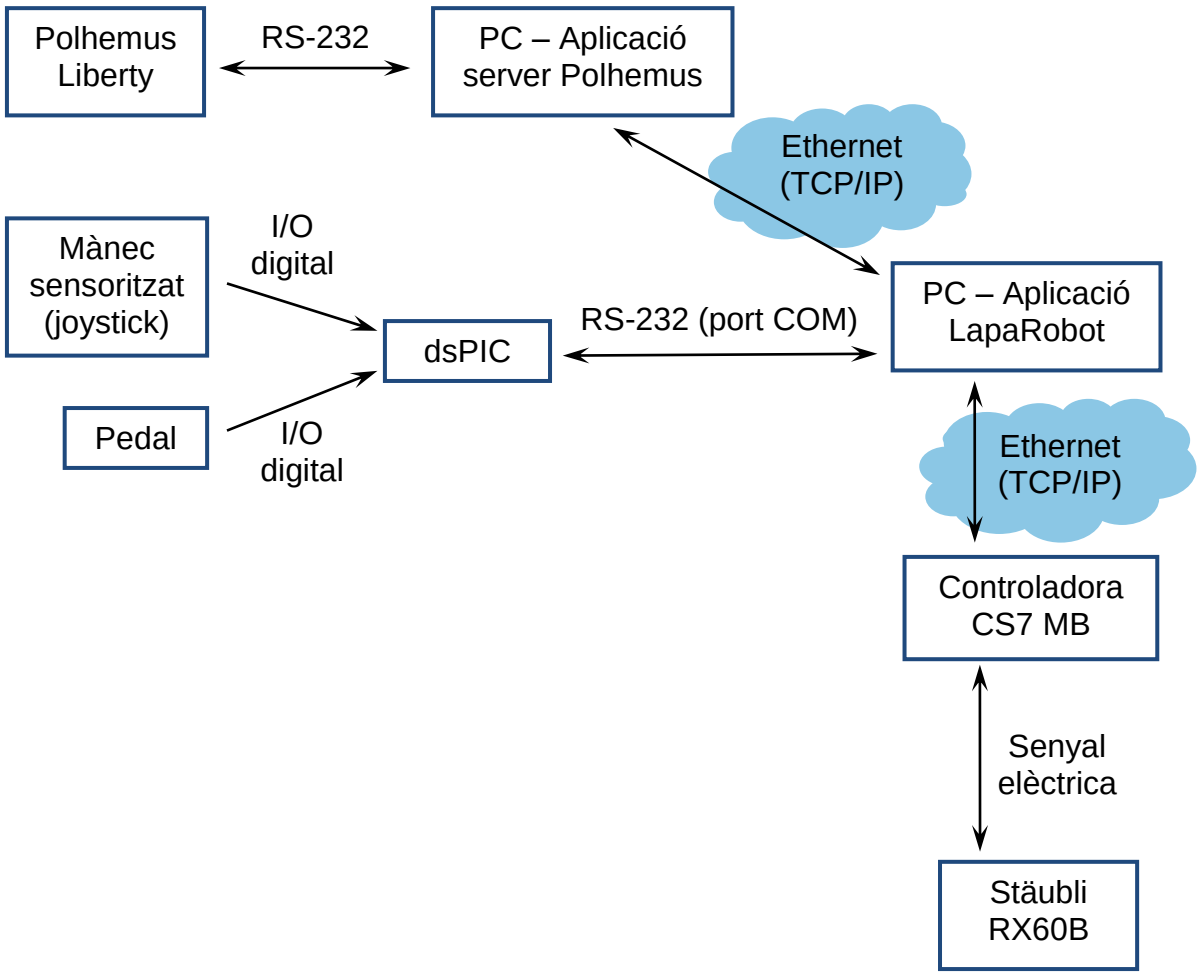


Figura 48: esquema de la comunicació dels diferents elements del sistema

En els següents apartats es comenten, per separat, algunes de les comunicacions entre els diferents elements del sistema.

Polhemus Liberty/PC (aplicació ServerPolhemus)

Com s'ha comentat en les especificacions tècniques el Polhemus Liberty pot connectar-se a un ordinador a través d'un port RS-232 o d'un USB. En el sistema construït s'utilitza

el port RS-232.

PC (aplicació ServerPolhemus) / PC (aplicació LapaRobot)

Per tal de comunicar les dues aplicacions que s'executen en computadores diferents ServerPolhemus i LapaRobot, com el nom de la primera indica, aquesta fa de servidor i la segona s'hi connecta a través d'un socket per demanar-li les dades que necessita de forma periòdica.

Aquesta connexió és realitza a través d'Ethernet utilitzant el protocol TCP/IP.

PC (aplicació LapaRobot)/Controladora CS7 MB

Per tal de comunicar l'aplicació LapaRobot amb la Controladora CS7 MB del robot, la primera crea un socket amb la segona. D'aquesta manera es crea una connexió a través d'Ethernet utilitzant el protocol TCP/IP.

Utilitzar aquest sistema de comunicació entre aquests dos elements fa possible que el sistema implementat suporti la teleoperació.

6.2 Software

En aquesta secció és parlarà de tots els temes relacionats amb el software utilitzat en l'estació de treball experimental construïda.

Per tal de justificar l'arquitectura software utilitzada és important conèixer les necessitats del sistema. Això permetrà entendre les decisions que han portat a implementar l'arquitectura que es descriu.

El requisit fonamental que s'ha hagut de tenir molt en compte duran el disseny de l'estructura software ha estat la necessitat de poder utilitzar el dispositiu esclau de forma teleoperada. Un altre fet que ha afectat al disseny software ha estat la necessitat de realitzar comprovacions i modificacions sobre la informació rebuda del dispositiu mestre abans de transmetre-la a l'escau.

Per aquests motius s'ha utilitzat una aplicació, "Server Polhemus", encarregada de llegir la posició del dispositiu mestre i fer de servidor per enviar-la a través de la xarxa. Una altre de les aplicacions utilitzades, anomenada "LapaRobot", llegeix la informació de posició del servidor i la de rotació del dsPIC, l'hi aplica totes les ajudes i restriccions de les que disposa el sistema i envia els increments de posició i orientació a la controladora del dispositiu esclau, on s'executa l'intèrpret que tradueix les instruccions al llenguatge V+.

A continuació s'ofereix un diagrama (figura 49) que mostra els diferents components software del sistema implementat:

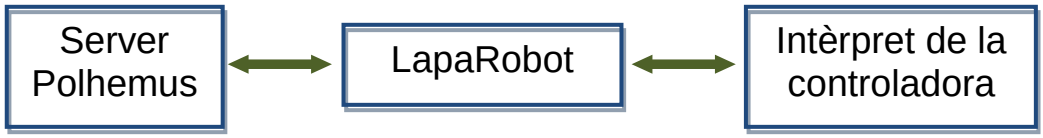


Figura 49: diagrama del software utilitzat per el sistema

Server Polhemus

Aplicació (figura 50) desenvolupada per el grup GRINS d'ESAII que permet configurar algunes característiques del Polhemus (format de les dades i nombre de sensors actius). També llegeix les dades del sensor, les mostra a la seva interfície i les envia a l'aplicació LapaRobot a través de la xarxa quan aquesta els hi demana. Permet mantenir comptadors de "frames" i de temps i mostra el "frame rate" de l'aplicació.

Per el correcte funcionament de l'aplicació LapaRobot cal activar els "checkboxes" de l'apartat "Data Acquisition" de "Position" i de "Direction Cosine Matrix" (els angles no són

necessaris ja que la rotació de l'element terminal del robot s'ha implementat en un control addicional tipus joystick que es troba en el mànec sensoritzat). També cal configurar els sensors actius activant només el primer.

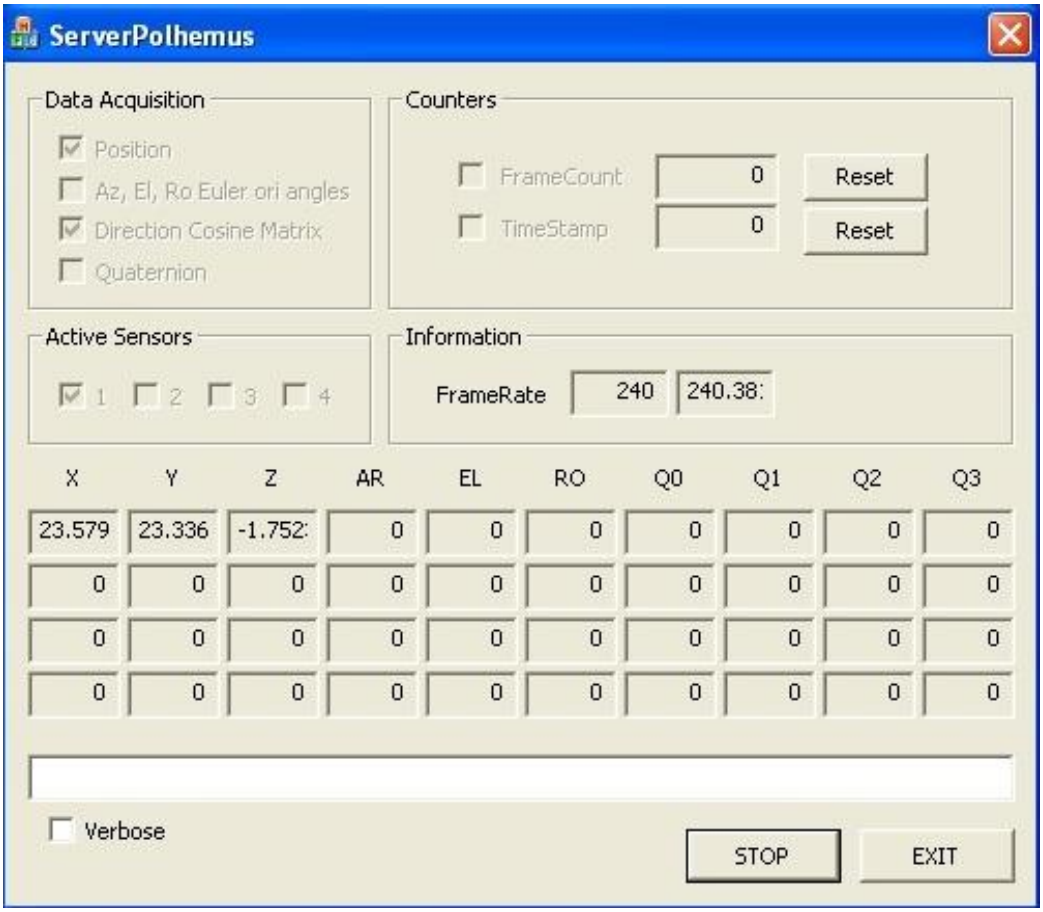


Figura 50: Aplicació ServerPolhemus

LapaRobot

Aplicació que es connecta amb l'aplicació ServerPolhemus a través d'un socket i li demana informació de forma periòdica utilitzant un temporitzador. Quan el servidor (aplicació ServerPolhemus) li envia una posició la llegeix i, després d'aplicar-li totes les ajudes i restriccions a la teleoperació explicades en l'apartat 6.2.1, envia l'increment de posició a la controladora del robot. Per enviar-la, utilitza un socket amb la controladora del robot. Amb l'ús dels dos sockets (el del dispositiu mestre i el de l'esclau) s'aconsegueix permetre que la intervenció quirúrgica es pugui realitzar de forma teleoperada ja que el cirurgià podrà estar en una ubicació i el braç robòtic en una altre.

És important esmentar que, aquesta aplicació, també realitza els càlculs necessaris per obtenir la posició que s'ha d'enviar al robot per tal que l'extrem de l'instrument laparoscòpic estigui en la posició rebuda del dispositiu mestre (ja que la posició que es llegeix del Polhemus correspon a la posició de l'extrem de l'instrument, es a dir, del punt de treball). Aquest càlcul és molt important ja que s'ha de tenir en compte que, en una intervenció mínimament invasiva, els instruments entren al cos del pacient a través d'un orifici sobre el qual no es poden realitzar esforços. Per això, per calcular la posició del robot que derivarà en que l'instrument laparoscòpic quedi a la posició desitjada, s'utilitza una translació de la posició de l'instrument seguint el vector entre l'extrem d'aquest i l'orifici d'entrada al cos del pacient (aquest últim té una posició fixa). A la figura 51 s'il·lustra aquesta translació:

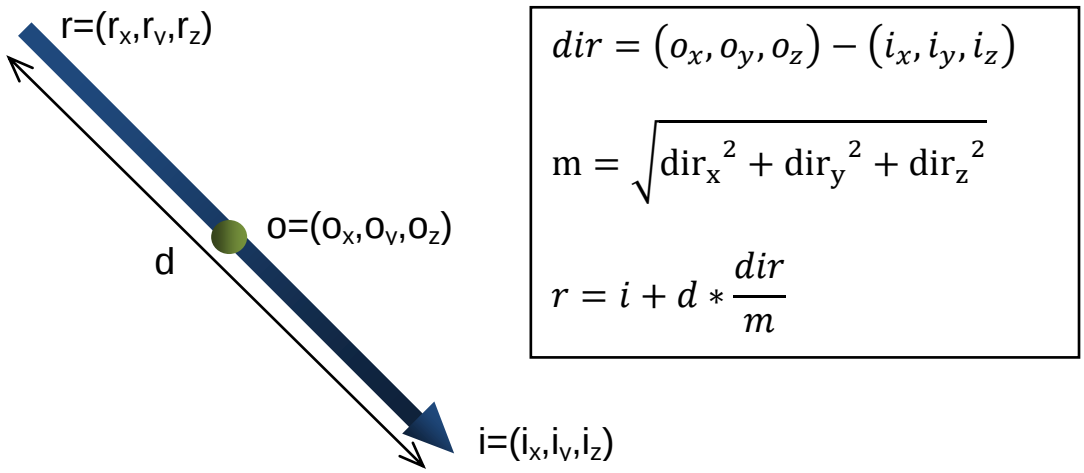


Figura 51: translació necessària per obtenir la posició del robot a partir de la posició de l'extrem de l'instrument laparoscòpic

L'aplicació, amb el mateix temporitzador utilitzat per demanar dades de posició al Polhemus, també llegeix d'una microcontroladora dsPIC (16bits), a través d'un port sèrie tipus COM, la posició del pedal i del joystick utilitzats. Aquesta microcontroladora envia un nombre de 16 bits a partir del qual es pot esbrinar si el pedal està o no polsat i si el joystick indica que l'instrument laparoscòpic s'ha de rotar en alguna direcció. Com s'ha explicat, si el pedal no està polsat es tallarà el flux d'informació entre el dispositiu mestre i el dispositiu esclau. Si l'element terminal s'ha de rotar s'enviarà l'increment de rotació corresponent a la controladora del robot a través del socket que s'ha comentat anteriorment.

LapaRobot també mostra una representació gràfica del braç robòtic utilitzat a través d'OpenGL gràcies a la qual es pot seguir la posició i rotació del dispositiu esclau en cada instant de temps.

A la figura 52 s'hi mostra l'aplicació i, a continuació, s'hi descriuen els seus elements i se n'explica el funcionament:

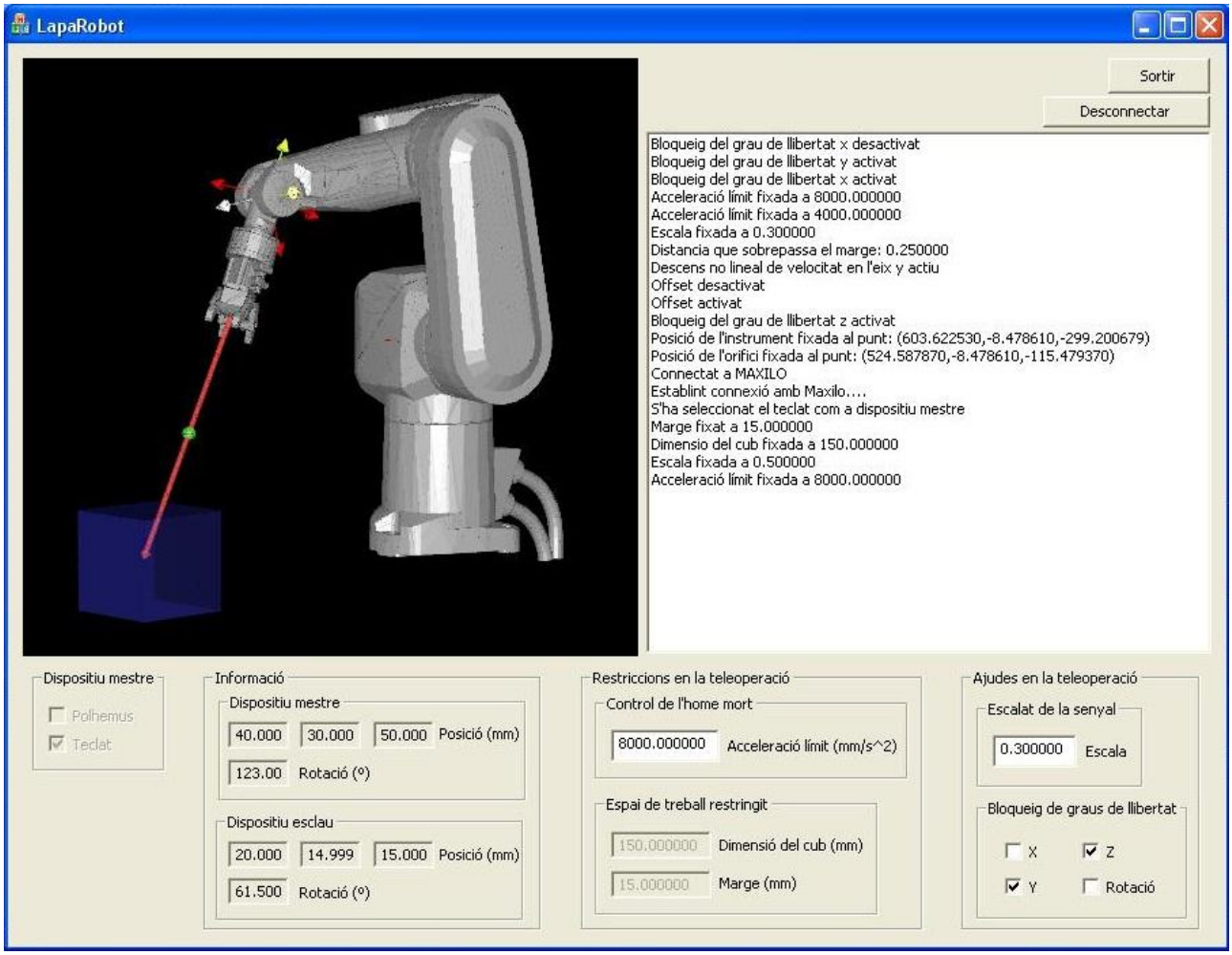


Figura 52: aplicació LapaRobot

Elements:

- Representació gràfica del braç robòtic: a la part superior esquerra de l'aplicació és mostra una representació amb OpenGL del braç robòtic per tal de poder seguir-ne els moviments. També s'hi mostra l'instrument laparoscòpic (color vermell), l'orifici

d'entrada al cos del pacient (esfera de color verd) i l'espai de treball restringit (cub de color blau).

- Botó sortir: permet tancar l'aplicació.
- Botó Connectar/Desconnectar: permet connectar-se mitjançant un socket amb la controladora del braç robòtic (anomenat MAXILO) i, s'hi el dispositiu mestre seleccionat és el Polhemus, amb el sensor magnètic.
- Llista de missatges: a sota del botó Connectar/Desconnectar s'hi troba una llista de missatges on el sistema dona informació a l'usuari dels paràmetres, tant d'ajudes com de restriccions a la teleoperació, fixats i de quan s'activen. També informa de la connexió o desconnexió amb els dispositius mestre i esclau i del dispositiu mestre escollit. Per últim també transmet a l'usuari aquells errors que es donen durant l'execució i de com hi respon el sistema.
- Secció dispositiu mestre: permet triar a l'usuari s'hi utilitzar el Polhemus o el teclat com a dispositiu mestre. Aquesta secció es deshabilita quan s'està connectat. A continuació és llisten els controls per cada un dels dos dispositius:
    - Polhemus:
        - Moviment en l'espai: el moviment del sensor és trasllada, després d'aplicar-hi les ajudes i restriccions en la teleoperació de les que disposa el sistema (explicades a l'apartat 6.2.1), a l'instrument laparoscòpic.
        - Rotació: el moviment del joystick del que disposa el mànec sensoritzat és trasllada, després d'aplicar les ajudes i restriccions en la teleoperació de les que disposa el sistema (explicades a l'apartat 6.2.1), a l'instrument laparoscòpic.
        - Offset: mentre el pedal roman polsat es desactiva l'offset permetent que els moviments en l'espai i les rotacions siguin transmeses al dispositiu esclau.
    - Teclat:
        - Moviment en l'espai: les fletxes avall i amunt permeten incrementar i decrementar respectivament la posició del dispositiu mestre en l'eix

x. Les fletxes dreta i esquerra permeten incrementar i decrementar respectivament la posició del dispositiu mestre en l'eix y. Els botons "Shift" i "Ctrl" permeten incrementar i decrementar respectivament la posició del dispositiu mestre en l'eix z. Aquests moviments és traslladen, després d'aplicar-hi les ajudes i restriccions en la teleoperació de les que disposa el sistema (explicades a l'apartat 6.2.1), a l'instrument laparoscòpic.

- Rotació: els botons "AvPág" i "Fin" permeten incrementar i decrementar respectivament la rotació del dispositiu mestre. Aquesta rotació es trasllada, després d'aplicar-hi les ajudes i restriccions en la teleoperació de les que disposa el sistema (explicades a l'apartat 6.2.1), a l'instrument laparoscòpic.
- Offset: el botó "Supr" permet activar i desactivar l'offset permeten que es transmetin o no els moviments del dispositiu mestre a l'esclau.

- Secció informació: en aquest apartat s'hi mostra, quan l'aplicació està connectada, la posició (en mil·límetres) i orientació (en graus) del dispositiu mestre i del dispositiu esclau respecte la seva posició inicial.
- Restriccions en la teleoperació:
  - Control de l'home mort: en aquesta secció es permet fixar l'acceleració límit (en mm/s^2) a partir de la qual s'activa el control de l'home mort i, després de realitzar el descens no lineal de la velocitat, s'atura el funcionament dels dispositius mestre i esclau.
  - Espai de treball restringit: en aquesta secció s'hi poden fixar les dimensions del cub (en mil·límetres) que representa l'espai de treball restringit i el marge (en mil·límetres), respecte els límits d'aquest cub, a partir del qual s'activa el descens no lineal de la velocitat. Cal esmentar que, tal i com s'ha implementat la restricció, mai es permet estar a un distància inferior a 0.5 mil·límetres de cap límit per assegurar que, per temes de resolució i precisió, el braç robòtic no els pugui sobrepassar. Aquesta secció es deshabilita quan s'està connectat.

- Ajudes en la teleoperació:
    - Escalat del senyal: en aquesta secció s'hi pot fixar el factor d'escala que s'aplicarà als increments de posició i rotació del dispositiu mestre abans de transmetre'ls al dispositiu esclau.
    - Bloqueig de graus de llibertat: en aquesta secció es permet activar i desactivar el bloqueig dels graus de llibertat dels tres eixos de moviment (x,y,z) i de la rotació. Quan algun d'aquests estigui activat els seus canvis en el dispositiu mestre no es transmetran al dispositiu esclau.

Intèrpret de la controladora

Com s'explica en l'apartat 6.1.2.1, s'utilitza un intèrpret específic d'alt nivell que rep les instruccions de l'aplicació LapaRobot i les converteix en instruccions del llenguatge V+ que són posteriorment interpretades i enviades al braç robòtic per la controladora CS7 MB.

A l'annex II s'adjunta una part del codi d'aquesta aplicació.

6.2.1 Paràmetres del sistema

Aquest apartat consta de dues seccions. En la primera, ajudes a la teleoperació, s'expliquen tots aquells aspectes configurables del sistema que modifiquen el seu comportament perquè l'usuari pugui obtenir aquella resposta que s'ajusti més a les seves necessitats, dotant així el sistema d'adaptabilitat.

En la segona secció, restriccions de seguretat, s'expliquen aquells paràmetres configurables del sistema destinats a fer el seu ús el màxim de segur possible. S'ha dedicat una secció apart a aquests paràmetres, i no s'han inclòs dins de les ajudes a la teleoperació, ja que per dur a terme la funcionalitat desitjada, realitzar intervencions quirúrgiques mínimament invasives de forma teleoperada, la seguretat serà un dels principals requisits per la delicadesa i precisió requerides per dur a terme una activitat

d'aquestes característiques. Si es proporciona al sistema una seguretat adaptable s'aconseguirà maximitzar-la, tot i que el sistema s'utilitzi per realitzar diferents procediments.

6.2.1.1 Ajudes a la teleoperació

Per tal d'augmentar l'adaptabilitat i per tant, la usabilitat del sistema, aquest permet a l'usuari la modificació de certs paràmetres. Amb el seu ajustament es fa possible adaptar el comportament del sistema a les necessitats del cirurgià per tal de maximitzar-ne la seva eficàcia.

A continuació s'expliquen amb detall els diferents paràmetres variables dels quals disposa l'estació de treball dissenyada.

6.2.1.1.1 Canvi d'escala

Un dels paràmetres més importants i pràctics d'un sistema de cirurgia teleoperada és la possibilitat de realitzar un canvi d'escala al senyal rebut del dispositiu mestre. Això farà possible realitzar moviments molt petits sense que els moviments realitzats pel cirurgià hagin de ser de la mateixa amplitud. Gràcies a aquesta funcionalitat es pot aconseguir la precisió necessària per poder realitzar intervencions de forma molt més segura.

El que s'aconsegueix amb un canvi d'escala és, per exemple, transformar un moviment d'un centímetre per part del cirurgià en un moviment d'un mil·límetre del braç robòtic (aplicar factor d'escala 1/10). A la figura 53 es mostren dues funcions, la primera és el desplaçament en l'eix x proporcionat pel sensor (color blau) i la segona la sortida després d'aplicar-l'hi un canvi d'escala (color verd):

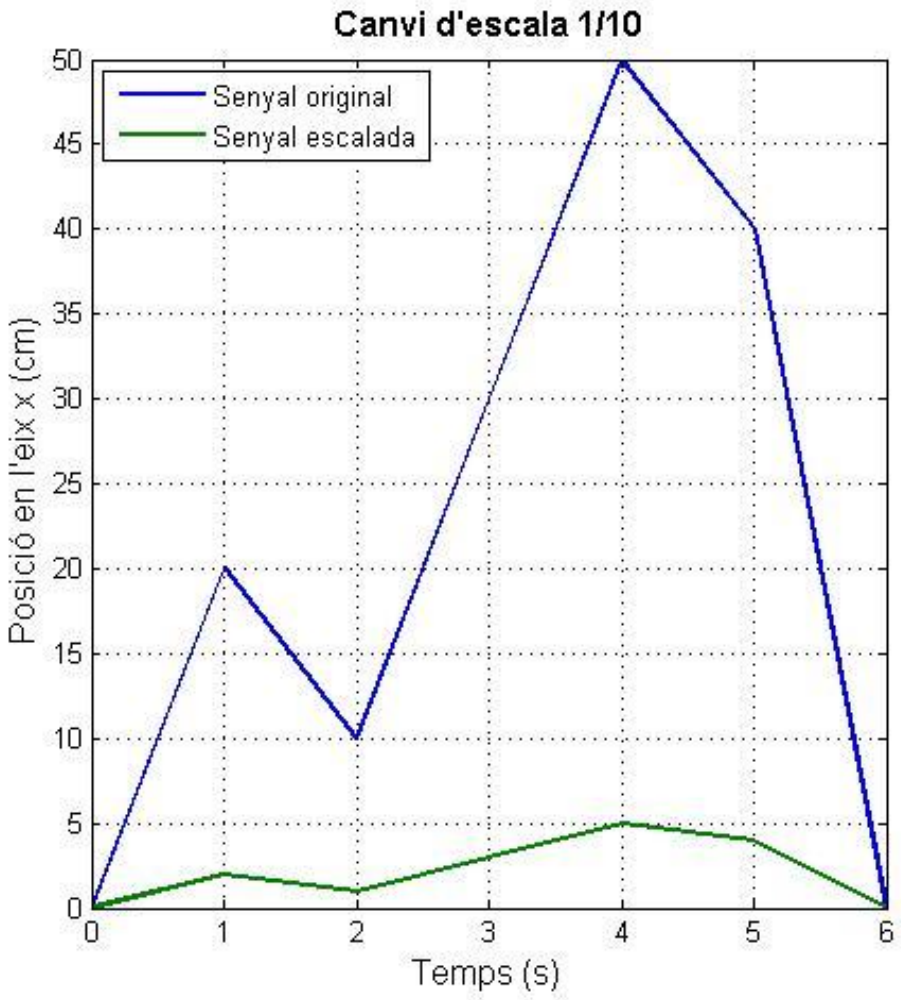


Figura 53: escalat d'un senyal (factor 1/10)

6.2.1.1.2 Offset

Una de les limitacions amb la que s'han d'enfrontar els cirurgians a l'hora de realitzar una operació és el moviment restringit de les articulacions humanes. Per tal de fer que el sistema dissenyat ofereixi una solució a aquest problema s'ha implementat un sistema d'offset. Al deixar de polsar el pedal es talla la comunicació entre el dispositiu mestre (sensor) i l'esclau (braç robòtic), permetent així al cirurgià tornar a una posició natural i continuar treballant com si no s'hagués mogut. El software dissenyat ignora els canvis que es produeixen en la situació del sensor durant el període de temps en que el pedal roman sense polsar i quan aquest es pressiona el cirurgià pot seguir treballant de forma relativa a la posició anterior. D'aquesta manera es pot aprofitar la superior mobilitat que

tenen les articulacions del braç robòtic respecte les humanes i proporcionar al cirurgià un nivell de comoditat alt per poder realitzar de la millor manera possible la intervenció quirúrgica.

La incorporació d'aquest element a l'estació de treball també augmenta la seguretat ja que, els moviments del dispositiu mestre no es transmetran al dispositiu esclau a no ser que s'estigui prement el pedal.

6.2.1.1.3 Bloqueig de graus de llibertat

Un altre avantatge rellevant que es pot aprofitar amb un sistema com el que s'ha dissenyat és la possibilitat d'ignorar certs graus de llibertat que proporciona el dispositiu mestre. D'aquesta manera es pot oferir a l'usuari una manera més segura de realitzar algunes maniobres fent que el software no tingui en compte certs moviments no útils (exemple: no realitza desplaçaments en l'eix x).

D'aquesta manera s'aconsegueix augmentar la comoditat i l'eficàcia del sistema, ja que el cirurgià podrà concentrar tota la seva atenció en realitzar, de forma òptima, els moviments necessaris per dur a terme l'operació sense haver de preocupar-se de realitzar algun moviment de forma involuntària que podria danyar els teixits del pacient.

6.2.1.2 Restriccions de seguretat

Com s'ha comentat anteriorment, en un sistema com el que s'ha dissenyat, la seguretat és un factor molt important per la seva utilització en cirurgia. Per això cal parar especial atenció en assegurar que aquest sistema sigui segur i, per tant, fiable. Dotar el sistema d'una seguretat adaptable permet maximitzar-la tot i que el sistema s'utilitzi per dur a terme diferents procediments.

Tot i que els tres tipus de ajudes a la teleoperació anteriors també augmenten la seguretat del sistema, a continuació s'expliquen algunes restriccions, el motiu de ser dels quals és únicament la seguretat.

6.2.1.2.1 Control de l'espai de treball

La primera restricció de seguretat que s'ha considerat important que el sistema tingui és un espai de treball restringible. D'aquesta manera l'usuari podrà definir l'espai de treball en el qual és mou el braç robòtic. Cal tenir en compte que aquest sempre haurà de ser més petit o igual que el volum de treball en el qual el braç robòtic pot situar el seu element terminal.

La finalitat de poder modificar aquest paràmetre és que d'aquesta manera es podran evitar danys sobre les zones on no s'està operant, ja siguin els teixits del voltant de la zona de treball o objectes de l'entorn on es realitzi el procediment.

A la figura 54 es mostra un gràfic d'aquest espai de treball, amb forma de cub, parametritzat:

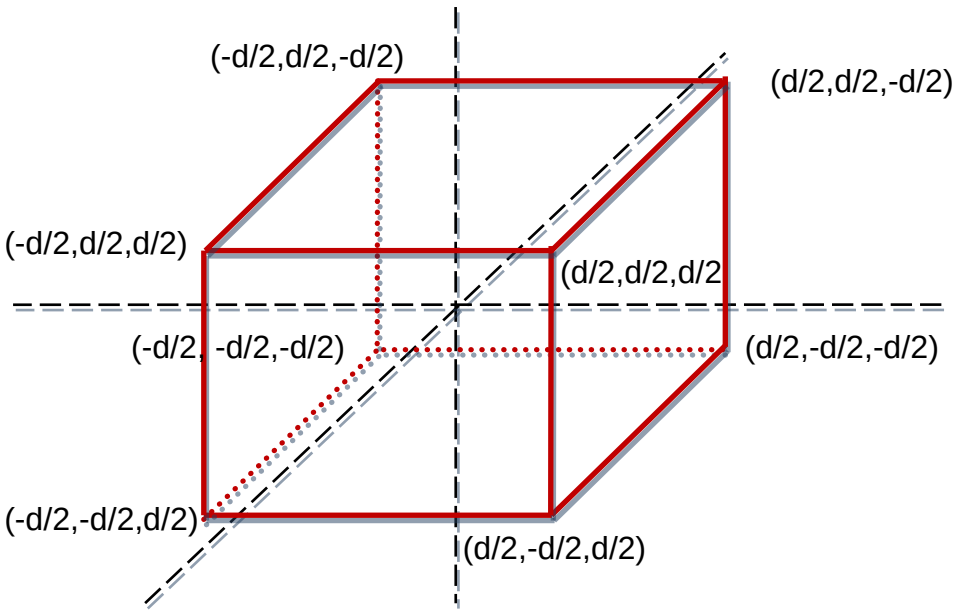


Figura 54: representació de l'espai de treball restringit (d és la dimensió del cub)

6.2.1.2.2 No linealitat

Aquesta restricció de seguretat permetrà evitar frenades brusques a l'arribar als límits de l'espai de treball assignat. Podria ser perillós per al pacient que, a l'arribar als límits de l'espai de treball, el braç robòtic efectués una frenada brusca, ja que aquesta podria produir danys sobre els teixits que l'element terminal té al voltant.

Per això s'ha decidit implementar aquesta restricció en el moviment per fer que el decrement de velocitat no sigui lineal (figura 55). D'aquesta manera, quan l'element terminal estigui a certa distància d'un límit de la caixa que conté l'espai de treball assignat, la velocitat del braç robòtic en aquella component anirà en decrement**.** Amb aquesta restricció s'aconsegueix que la velocitat no és redueixi dràsticament (gran desacceleració) a l'arribar els límits de l'espai de treball, evitant així els possibles danys que això pot significar (ex:al desaccelerar molt ràpid el braç robòtic pot vibrar i produir, així, laceracions en els teixits propers a l'element terminal).

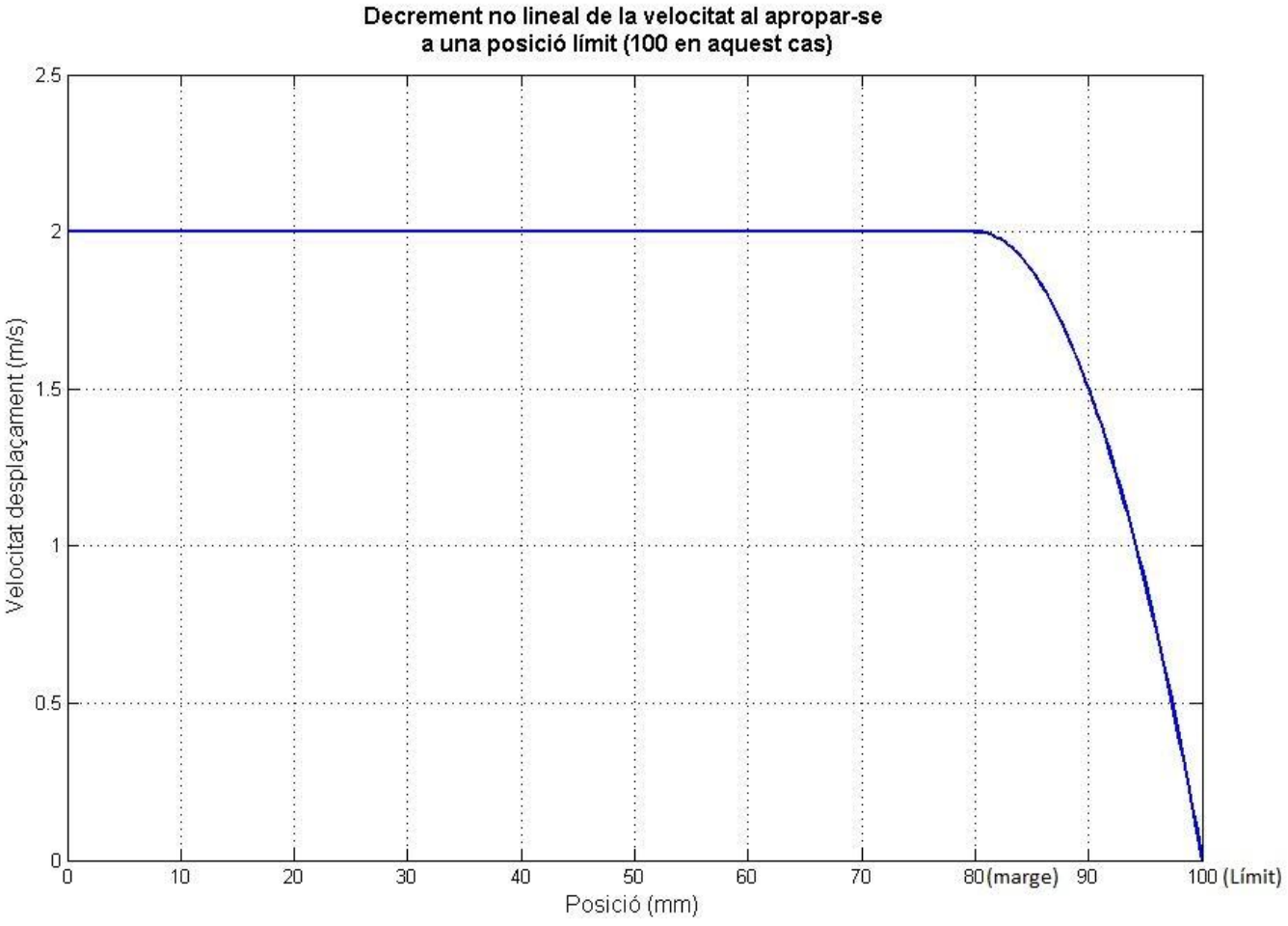


Figura 55: gràfica del decrement no lineal de la velocitat implementat

A la figura anterior es mostra la resposta d'aquest control a l'acostar-se a un límit (100mm) de l'espai de treball assignat a una certa velocitat (v_{marge}=2 m/s). Com es pot observar el control s'activa a l'arribar a certa distància del límit, anomenem a aquesta distància, marge (a la figura superior el marge=20mm).

Per decrementar la velocitat de forma no lineal s'ha utilitzat un model quadràtic inspirat en l'expressió de la força d'una molla (F=k*x):

$$r=k\cdot(x-(límit - marge)) \text{ (quan } x>(límit - marge), \text{ altrament } r=0)$$

$$v_{control\ robot} = v_{marge} - 1/2(r^2)$$

Es pot calcular el valor idoni de k per tal d'aconseguir aturar el moviment del sistema (amb un marge d'espai fixat, la velocitat quan s'arriba a aquest marge (v_{marge}) i el límit de l'espai de treball) a partir de les equacions següents:

$$v_{marge} = 1/2(r^2) \text{ (per aconseguir } v_{control}=0 \text{ quan } x=límit)$$

$$r=k\cdot marge \text{ (quan } x=límit)$$

Si combinem les equacions anteriors es pot obtenir el valor de k:

$$v_{marge} = 1/2(k \cdot marge)^2$$

$$2\cdot v_{marge} = k^2\cdot marge^2$$

$$k=sqrt(2\cdot v_{marge} / marge^2)$$

6.2.1.2.3 "Control de l'home mort"

Per últim, una restricció de seguretat molt important és l'anomena't control de l'home mort. La seva finalitat és evitar els danys ocasionats si el cirurgià deixés anar el dispositiu mestre per qualsevol circumstància.

Per tal d'assegurar aquesta situació, s'ha implementat un detector de moviments bruscs que fa que s'aturi el funcionament del dispositiu si l'acceleració és superior a un llindar (paràmetre variable). D'aquesta manera si per algun motiu es deixés anar el sensor magnètic s'evitarien els danys que això ocasionaria sobre el pacient.

Aquesta restricció permetrà, per tant, que quan l'acceleració superi aquest llindar s'aturi el moviment del braç robòtic, ja que es suposarà que s'ha deixat anar el sensor magnètic. Per reduir la velocitat s'utilitza el model no lineal explicat a l'apartat anterior.

6.3 Avaluació de l'estació de treball experimental

En aquesta secció s'estudiarà el funcionament de l'estació de treball experimental construïda per tal d'avaluar la seva resposta davant diferents situacions.

El primer aspecte que s'ha decidit observar és que les connexions amb la controladora del robot i, s'hi el dispositiu mestre és el Polhemus, amb el sensor, és realitzin correctament. També s'ha comprovat que en cas que no es pogués obrir un socket es notifiqués a l'usuari de l'error corresponent i es tornés a deixar el sistema en el seu estat inicial per mantenir-ne la integritat.

Els paràmetres corresponents a les direccions IP i els ports del Polhemus i la controladora del robot s'han definit en el codi amb la directiva de C++ #define, fet que els fa fàcilment modificables.

Seguidament s'ha observat el funcionament del sistema sense utilitzar cap de les ajudes n'hi restriccions a la teleoperació de les quals disposa. Primer de tot s'ha comprovat el

funcionament d'aquest observant com el dispositiu esclau respon al moviment del dispositiu mestre. Per fer-ho s'ha utilitzat la informació de posició que ofereix l'aplicació LapaRobot i s'ha comprovat que dels increments de posició en el mestre derivin moviments de la mateixa amplitud en l'esclau. Aquesta mateixa prova s'ha realitzat amb la rotació. Aquests experiments s'han realitzat amb els dos dispositius mestres oferts per l'aplicació. Cal esmentar que amb el Polhemus s'ha realitzat una comprovació addicional amb una cinta mètrica per tal d'assegurar que els moviments que aquest detecta siguin un reflex fidel dels moviments realitzats per l'usuari.

També s'ha comprovat que quan s'envia al robot una posició a la qual no pot arribar s'imprimeixi l'error corresponent indicant l'articulació que està pròxima al seu límit.

La següent comprovació que s'ha realitzat sobre el sistema és que les posicions inicials fixades del braç robòtic, el sensor magnètic, l'orifici d'entrada al cos del pacient i l'instrument laparoscòpic, siguin correctes. Per assegurar aquest fet tant important s'ha comprovat que la posició que s'envia a la controladora del braç robòtic al establir-hi la connexió (posició inicial del dispositiu esclau) es guardi en el sistema, després que el braç robòtic hi hagi arribat, com la posició inicial. Amb el dispositiu mestre s'ha comprovat que, sigui quina sigui la posició del sensor respecte la font al iniciar el sistema, aquesta sigui emmagatzemada com la posició inicial i es treballi amb ella com a "origen de coordenades".

L'aplicació actual defineix la posició inicial de l'instrument seguint la direcció de l'últim element del braç robòtic. L'orifici d'entrada al cos del pacient es defineix a la meitat d'aquest instrument. Els paràmetres corresponents a la llargada de l'instrument laparoscòpic i a la distància a la que es troba l'orifici respecte l'últim element del robot, s'han definit en el codi amb la directiva de C++ #define, fet que els fa fàcilment modificables.

Els aspectes del sistema dels quals s'ha comprovat el correcte funcionament seguidament, han estat les ajudes en la teleoperació: l'offset, l'escalat de la senyal i el bloqueig de graus de llibertat.

Per comprovar el correcte funcionament de l'offset s'ha observat que, tant amb el teclat (tecla "Supr") com amb el Polhemus (pedal) seleccionats com a dispositiu mestre, es pogués activar i desactivar el seu funcionament. També s'ha mirat que mentre l'offset roman actiu, tot i que la posició del dispositiu mestre canviï, la posició del dispositiu esclau no es modifiqui.

La següent ajuda a la teleoperació de la qual s'ha estudiat el funcionament és l'escalat del senyal. Per fer-ho s'ha observat que s'escalessin de forma correcte els increments de posició i rotació del dispositiu mestre abans de transmetre'ls al dispositiu esclau. S'ha comprovat que aquest factor d'escala no pogués fixar-se menor que 0.

Per últim s'ha observat que el bloqueig dels quatre graus de llibertats que permet l'aplicació LapaRobot, els involucrats en una cirurgia mínimament invasiva, tinguessin l'efecte desitjat en el moviment i rotació del dispositiu esclau. S'ha observat que tot i que hi haguessin increments al dispositiu mestre en un grau de llibertat bloquejat, aquests no es transmetien al braç robòtic.

Els últims paràmetres del sistema que s'han estudiat són les restriccions en la teleoperació: el control de l'home mort i l'espai de treball restringit.

Primerament s'ha comprovat el control de l'home mort. S'ha mirat que el càlcul de l'acceleració que realitza el sistema fos correcte i que quan es sobrepasses el límit fixat el control s'activés. També s'ha comprovat que aquesta acceleració límit no pogués fixar-se negativa. Seguidament s'ha estudiat la resposta del sistema amb el control actiu, s'ha observat que el descens no lineal de la velocitat seguís el model descrit a l'apartat 6.2.1.2.2 i que, quan s'aturés el moviment del braç robòtic, es desconnectes la connexió entre els dispositius mestre i esclau. Esmentar que s'ha definit un paràmetre en el codi, amb la directiva de C++ #define, amb el nombre d'iteracions que es realitzen quan s'activa el control abans d'aturar el moviment del dispositiu esclau, fet que fa que es pugui modificar fàcilment.

L'última restricció estudiada és l'espai de treball restringit. S'ha mirat que no es permeti que l'extrem de l'instrument laparoscòpic surti del cub que engloba aquest espai de treball restringit. Cal dir que per motius de seguretat, deguts a que la precisió i resolució del dispositiu esclau no és infinita, s'ha fet que l'instrument no pugui acostar-se a menys de 0.5 mil·límetres de les parets del cub. També s'ha estudiat que el descens no lineal

de la velocitat realitzat quan s'està a una distància inferior al marge fixat d'un dels límits del cub, sigui el descrit a l'apartat 6.2.1.2.2.

Cal dir que s'ha hagut de tenir en compte la situació que el factor d'escala sigui tan gran que, amb l'increment de posició realitzat, es surti dels límits del cub que engloba l'espai de treball restringit sense passar per la zona de descens no lineal de la velocitat. Quan es dona aquest fet s'ha decidit no transmetre aquest moviment al dispositiu esclau.

Per últim s'ha comprovat que no es pugui fixar un marge igual o superior a la meitat de les dimensions del cub, un marge negatiu, unes dimensions del cub negatives o unes dimensions del cub superiors a cert límit. Aquest límit per les dimensions del cub s'ha decidit fixar per tal d'augmentar la seguretat del sistema al assegurar que el robot pugui accedir a totes les posicions del seu interior i que, per tant, no es donin errors de posició del robot durant la cirurgia. Aquest límit s'ha definit en el codi amb la directiva de C++ #define per tal de fer-lo fàcilment modificable.

Realitzant una avaluació més general de l'estació construïda es pot observar que, tot i ser un prototip de laboratori experimental d'un sistema de teleoperació assistida i no poder ser utilitzat en una intervenció real, el sistema dissenyat és escalable i adaptable. Això és així ja que permet la modificació de la seva configuració per tal d'adaptar-se a situacions diferents. A més permetria canviar el dispositiu esclau sense necessitat de fer canvis en el dispositiu mestre o en les ajudes i restriccions en la teleoperació implementades.

Aquest sistema també es podria ampliar fàcilment per tal que utilitzes dos mànecs sensoritzats que permetessin al cirurgià utilitzar dos instruments laparoscòpics, per exemple un de subjecció i un d'incisió.

Un altre aspecte a destacar del sistema és la seguretat. El fet d'utilitzar mecanismes com l'offset, el control de l'home mort, l'espai de treball restringit, el descens no lineal de la velocitat i el bloqueig de graus de llibertat doten el sistema d'una seguretat important i adaptable, fet molt rellevant en intervencions quirúrgiques.

Cal destacar també que alguns d'aquests mecanismes també suposen una ajuda en la teleoperació. El fet de poder bloquejar certs graus de llibertat fa que l'usuari no s'hagi de preocupar de realitzar certs moviments involuntaris i, el disposar d'un espai de treball

restringit, permet no haver de vigilar en ocasionar danys als teixits propers a la zona d'intervenció. El sistema d'offset també elimina la necessitat de l'usuari de treballar en posicions incòmodes i permet aprofitar la superior mobilitat de les articulacions robòtiques.

Per últim, el poder aplicar un factor d'escala als moviments captats per el sensor magnètic permet a l'usuari realitzar moviments mil·limètrics amb el braç robòtic tot i que els seus moviments tinguin una amplitud diferent.

Per tal d'assegurar l'estabilitat i la fluïdesa del sistema, s'ha afegit un control del temps que transcorre entre dues lectures consecutives del sensor magnètic que fa que, en cas que aquest sigui relativament gran, es talli la connexió entre els dispositius mestre i esclau.

Un dels objectiu d'aquest projecte també era el d'aconseguir dissenyar una interfície cirurgià/sistema el màxim d'intuïtiva possible per tal que el cirurgià, pogués utilitzar el sistema d'una forma el màxim de semblant a una intervenció mínimament invasiva real. El fet d'utilitzar un sensor magnètic capaç de detectar els moviments en l'espai de l'instrument laparoscòpic fa que, el temps d'aprenentatge per utilitzar el sistema, es minimitzi al permetre que el cirurgià aprofiti de forma molt significativa les habilitats que ja posseeix.

Tot i aquests punts a favor del sistema cal puntualitzar que aquest prototip no podria utilitzar-se en una intervenció real. Per fer-ho caldria millorar i afegir certs elements.

Alguns d'aquests són:

- Mecanitzar alguns instruments laparoscòpics per permetre controlar l'obertura i el tancament de les seves pinces. Per això podria utilitzar-se un altre pedal que funciones amb la mateixa implementació que el de l'offset.
- Implementar una articulació passiva per incorporar al braç robòtic.
- Millora de la sincronització entre el dispositiu mestre i esclau.

- Integrar un sistema de visió per tal de poder veure la zona operada. Preferiblement en tres dimensions.
- Afegir filtres per ignorar les vibracions de la mà de l'usuari.
- Afegir els components necessaris per tal de fer que el sistema pogués preparar-se per una intervenció de forma senzilla i ràpida.
- Estudi i testeig molt més exhaustiu del sistema davant de situacions imprevistes que es poden donar en una intervenció quirúrgica.
- ....

Un altre millora que, tot i no ser imprescindible permetria a l'usuari rebre una retroalimentació del dispositiu esclau, seria utilitzar sensors de força i algun tipus de tecnologia hàptica per proporcionar una sensació de l'esforç que està realitzant el braç robòtic sobre el punt de treball.

7 CONCLUSIONS

Un cop finalitzat aquest projecte de final de carrera és important dedicar uns moments per reflexionar i extreure'n les conclusions finals més importants.

Primer de tot cal observar els objectius inicialment proposats i veure s'hi s'han complert i en quin nivell ja que, hi ha punts als quals es podia haver dedicat més temps i tractat amb més profunditat. Tot i així mai s'ha de perdre de vista el fet que per la realització d'aquest projecte es disposava d'un temps (quatre mesos) i uns recursos (una persona) limitats i, que per tant, s'han hagut de prioritzar aquelles parts més rellevants per aconseguir els objectius principals.

L'objectiu més important d'aquest projecte era el d'estudiar el disseny d'una interfície que permetés sensoritzar els moviments d'un instrument laparoscòpic involucrats en una intervenció quirúrgica.

Per fer-ho es va començar estudiant els antecedents de la robòtica repassant certs fets al llarg de la seva història. Es va decidir entrar en més profunditat en l'ús de la robòtica en la medicina i, en concret, en les intervencions quirúrgiques assistides per ordinador ja que, era on es centrava l'objectiu d'aquest treball.

Un cop es va realitzar aquest punt, es va poder dur a terme un estudi dels requisits del sistema per facilitar la tria posterior del sensor més adequat que permetés detectar els moviments necessaris per realitzar una intervenció quirúrgica laparoscòpica.

Tot seguit doncs, es va procedir a l'estudi dels diferents sensors que existeixen al mercat que permeten sensoritzar aquests moviments. En alguns d'ells, degut a que ràpidament es va poder veure que no servirien per l'objectiu desitjat, no es va entrar en massa profunditat ja que, es va preferir dedicar el temps del es disposava en estudiar de forma més exhaustiva aquells que oferien unes millors característiques.

Un cop acabat aquest punt es van analitzar les diferents formes en que es podia integrar cada sensor i, ràpidament, es van poder veure aquells mètodes que perjudicaven més el sistema dissenyat i aquells que oferien unes millors característiques.

Amb tota aquesta informació es va procedir a escollir el conjunt sensor/mètode d'integració que oferia unes garanties majors d'obtenir un sistema segur, útil, eficient, pràctic i intuïtiu.

Per tal de completar aquest projecte es va voler dissenyar un prototip de laboratori experimental per l'estudi de la realització de cirurgies mínimament invasives teleoperades i assistides per ordinador.

L'estació de treball construïda, com s'ha comentat a l'apartat 6.3 (avaluació de l'estació de treball experimental), no pot ser utilitzada en una intervenció real però si que permet veure els grans avantatges que suposa l'ús d'un sensor que permet detectar els moviments en l'espai de l'instrument laparoscòpic. Aquest fet fa que la interfície sigui molt intuïtiva i facilita molt la seva utilització per part d'un cirurgià.

En aquest prototip, a part de la rotació i els moviments necessaris per dur a terme laparoscòpies, també s'han implementat algunes funcionalitats extres. S'ha acabat de completar l'estació dotant a l'aplicació LapaRobot d'ajudes i restriccions en la teleoperació com l'escalat de la senyal, l'offset, l'espai de treball restringit, el control de l'home mort i la possibilitat de bloquejar els diferents graus de llibertat. També s'ofereix una representació en OpenGL dels elements principals que intervenen en la intervenció (braç robòtic, orifici d'entrada al cos del pacient, instrument laparoscòpic i espai de treball restringit).

Per tal de fer possible la feina fora del laboratori de la universitat, s'ha implementat l'opció de controlar el dispositiu esclau amb el teclat i s'ha fet servir l'aplicació "Robot Controller Emulator" desenvolupada per el grup GRINS de la UPC (a l'annex V s'adjunta la documentació d'aquesta aplicació). Aquesta permet que les aplicacions que interactuen amb el robot Stäubli puguin fer-ho sense la necessitat d'utilitzar el robot real ja que, emula la seva controladora.

En quant a les meves conclusions personals, estic molt satisfet de la realització d'aquest projecte.

Dur a terme aquest treball m'ha permès adquirir coneixements en quan a la organització i realització d'un projecte d'aquesta envergadura, cosa molt important de cara a estar preparat per reptes futurs.

He pogut aprendre moltes coses noves en el camp de la robòtica, que personalment m'interessava, i durant l'estudi dels sensors també he pogut enriquir els meus coneixements en alguns camps de la física, una disciplina que sempre m'ha cridat l'atenció i per la qual sempre he sentit curiositat. També he adquirit alguns coneixements bàsics sobre les intervencions quirúrgiques mínimament invasives, més en concret les laparoscòpiques.

Respecte la part pràctica m'he divertit en la realització del prototip, exceptuant aquells moments, presents sempre a l'hora de programar, en que t'encalles durant una bona estona per culpa d'un petit error. La realització d'una aplicació que ofereix un resultat tant físic com poder moure un mànec sensoritzat i veure com en deriva el moviment d'un braç robòtic, és molt gratificant.

També he pogut ampliar els meus coneixements en el llenguatge C++ i utilitzar per primera vegada les llibreries MFC (“Microsoft Foundation Classes”). Aquestes últimes faciliten el desenvolupament d’aplicacions al proporcionar un accés senzill a les API (“application programming interface”) de Windows.

La realització d’aquest prototip també m’ha permès repassar alguns conceptes d’OpenGL i de la comunicació utilitzant sockets.

8 BIBLIOGRAFIA

- Robòtica (Apunts Robòtica FIB)

- Lossy compression Fourier transform (Apunts Compressió de dades i imatges FIB)

- Búsqueda heurística (Apunts Intel·ligència artificial FIB)

- Cinemàtica inversa (Apunts Física per la modelització i l'animació realista FIB)

- www.polhemus.com

- www.wordreference.com

- www.wikipedia.org

- http://robotica.es

- www.roboticaeducativa.com

- www.leia.es/E-ROBOT/fundamentos.html

- www.roboticspot.com/index.php

- www.monografias.com/trabajos6/larobo/larobo.shtml

- www.iearobotics.com

- www.robotics-platform.eu/cms/index.php

- www.roboticadeservicios.com/

- http://www-pagines.fib.upc.es/~rob/protegit/treballs/Q2_03-04/aplic_medicas/index.html

- http://computerassistedsurgeryblog.com/

- www.laparoscopy.com

- www.laparoscopy.net

- www.laparoscopyhospital.com

- http://scielo.isciii.es/pdf/aue/v30n1/v30n1a01.pdf

- http://jama.ama-assn.org/cgi/data/287/3/402/DC1/1

- http://www.umm.edu/esp_ency/article/007016.htm

- http://robots-argentina.com.ar/Sensores_general.htm

- http://isa.uniovi.es/~idiaz/SA/Teoria/04-05/SA.Sensores.pdf

- www.sensorsmag.com/

- www.superrobotica.com/Sensores.htm

- www.worldlingo.com/ma/enwiki/es/Potentiometer

- www.unicrom.com/Tut_resistenciavariable.asp

- www.artechhouse.com/GetBlob.aspx?strName=Groves_255_CH4.pdf

- www.starter-project.com/Presentazioni/Rocchi.pdf

- www.rchelisite.com/how_do_gyros_work.php

- www.gyroscopes.org/

- www.freepatentsonline.com/7197928.html

- http://zone.ni.com/devzone/cda/tut/p/id/7349

- www.worldlingo.com/ma/enwiki/es/Rotary_encoder

- http://zone.ni.com/devzone/cda/ph/p/id/132

- www.motioncapture.com/

- www.phasespace.com/motion-capture-faq.html#1

- www.dcs.shef.ac.uk/intranet/research/resmes/CS0111.pdf

- www-istp.gsfc.nasa.gov/Education/Mmfield.html

- www-istp.gsfc.nasa.gov/Education/Mmap.html

- www-istp.gsfc.nasa.gov/Education/Imagnet.html

- www.physics.sjsu.edu/becker/physics51/mag_field.htm

- www.freepatentsonline.com/5418454.html

- www.woodward.com/pdf/ic/02010.pdf

- www.engineersedge.com/instrumentation/lvdt_linear_voltage_displacment_tranducer.htm

- www.brighthub.com/engineering/mechanical/articles/62615.aspx

- www.lvdt.co.uk

- http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/hall.html

- http://scienceworld.wolfram.com/physics/

- http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/fluxmg.html

- http://books.google.cat/books?id=R8VAjMitH1QC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

- www.tsc.uvigo.es/DAF/Investigacion/PDFs/transpac-4.pdf

- http://books.google.cat/books?id=O8F8XSuz8D4C&printsec=frontcover&dq=digital+filters&source=bl&ots=F4ZRE2rcar&sig=6xcbi_T1510Wn4m1fZ0Be3azJOA&hl=es&ei=MH7lTKqWN9C7hAeWm5iIDQ&sa=X&oi=book_result&ct=result&resnum=2&ved=0CB8Q6AEwATgK#v=onepage&q&f=false

- http://bdml.stanford.edu/twiki/pub/Haptics/PhotonicRobots/MVControllerUsersGuide.pdf

- www.etitudela.com/profesores/rpm/rpm/downloads/manualv.pdf

- Física II, editorial edebé

- http://glprogramming.com/

- www.opengl.org/

- www.salle.url.edu/~oscarg/resources/openGLTutorialSpanish.pdf

- http://bdml.stanford.edu/twiki/pub/Haptics/PhotonicRobots/MVControllerUsersGuide.pdf

- www.scribd.com/doc/5562249/BrazoRobotTFC

- www.etitudela.com/profesores/rpm/rpm/downloads/manualv.pdf

- www.cplusplus.com/

- www.cppreference.com/wiki/start

- http://msdn.microsoft.com/es-es/

- www.codeproject.com

9 ANNEXOS

9.1 ANNEX I. Glossari

- Graus de llibertat: en mecànica, els graus de llibertat són el conjunt independent de desplaçaments i rotacions que especifiquen, completament, la posició i orientació d'un cos en l'espai. Un cos rígid en n dimensions té n(n+1)/2 graus de llibertat:

- n translacions
- n(n-1)/2 rotacions

- Roll, pitch i yaw:

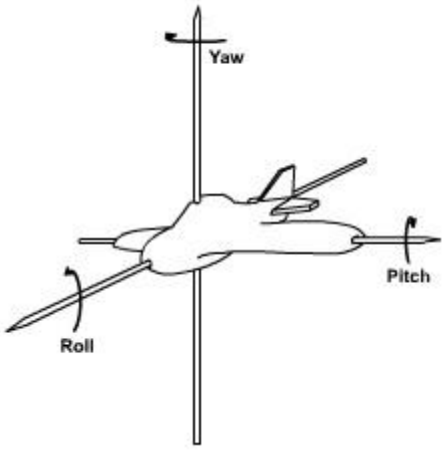


Figura 56: Roll, pitch i yaw

- Comunicació mestre/esclau: model de comunicació on un dispositiu (mestre) té control unidireccional sobre un altre (esclau)

- Trocar: instrument mèdic amb forma de cilindre utilitzat en cirurgies laparoscòpiques per facilitar l'entrada i la sortida de càmeres i instruments laparoscòpics a l'abdomen del pacient.

- Endoscòpia: tècnica utilitzada en medicina que consisteix en introduir els instruments i una càmera dins el pacient, per un orifici (natural o quirúrgic), amb l'objectiu de treballar sobre algun òrgan o cavitat corporal.

- *Visió estereoscòpica:* capacitat d'integrar dues imatges que es sobreposen (preses des d'angles diferents) en una de sola i produir, així, la sensació de les tres dimensions.

- *Mecatrònica:* combinació sinèrgica de diferents branques de l'enginyeria: mecànica, electrònica, informàtica i control.

- *Radiogoniometria:* tècnica que permet esbrinar l'orientació (gonio) de la qual prové un senyal de ràdio.

- *Interferometria:* tècnica per al diagnòstic de les propietats de dues o més ones, a través de l'estudi del patró d'interferència creat per la seva superposició.

- *Biometria:* estudi d'aquells mètodes que permeten identificar, de forma única, un ésser humà gràcies a trets físics o de comportament.

- *Termes de deriva inherents:* s'entén com a deriva la desviació progressiva d'una mesura respecte el valor de la magnitud real. Aquest fet fa que es vagi acumulant error.

- *Espín:* propietat física de les partícules subatòmiques per la qual tota partícula elemental té un moment angular intrínsec de valor fix.

- *Autòmat:* màquina que imita la figura i els moviments d'un ser animat.

9.2 ANNEX II. Codi LapaRobot

En aquest annex s'hi adjunta el codi de la classe més important de l'aplicació LapaRobot.

```
// VirtualRobotDlg.cpp : implementation file
//

#include "stdafx.h"
#include "VirtualRobot.h"
#include "VirtualRobotDlg.h"
#include "DlgProxy.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

//Definir l'identificador per el socket del robot (controladora) i del polhemus
#define RMAXILO   0
#define POLHEMUS 1

//Definir l'IP i el port per el Polhemus i la controladora del braç robòtic
#define IP_POLHEMUS "127.0.0.1"
#define PORT_POLHEMUS 11001

//IP i port per l'emulador de la controladora del robot
#define IP_ROBOT_MAXILO "127.0.0.1"
#define PORT_ROBOT_MAXILO 1005

/*
//IP i port per la controladora del robot real
#define IP_ROBOT_MAXILO "192.168.1.53"
#define PORT_ROBOT_MAXILO 1004*/


#define DATO_POLHEMUS 100

//Codis per emparellar els paquets de dades amb els increments que s'envien a
la controladora del robot amb els que aquesta respon quan està apunt d'arribar
a la posició desitjada
#define TELEOP 1
#define TELEOPROTATION 2
#define TOOLOP 3


//Definir les dimensions del cub que representa la caixa conenidora del volum
de treball del braç robot en una cirurgia laparoscòpica (en mm)
//No és el volum de treball del braç robòtic
#define DIMCUBESPAITREBALLRESTRINGIT 150

//Posició inicial de l'última articulació del robot->
(x,y,z)=(386.46414499999997,48.999994000000001,264.26352500000002)
#define LASTARTICULACIONX 386.464145
```

```
#define LASTARTICULACIONY 48.999994
#define LASTARTICULACIONZ 264.263525

//Distancies relatives entre l'última articulació del robot i la punta de la
pinça
#define DISTLASARTICULATIONGRASPX 74.53585
#define DISTLASARTICULATIONGRASPY 0
#define DISTLASARTICULATIONGRASPZ -173.263525

//Definir la llargada de l'instrument laparoscòpic
#define LLARGADALAPAROSCOPICINSTRUMENT 400

//Definir posició relativa del tool
#define TOOLX 0.0
#define TOOLY 0
#define TOOLZ 173.263525+LLARGADALAPAROSCOPICINSTRUMENT


//Definir la distància respecte la punta de la pinça del robot a la que es
troba l'orifici
#define DISTANCIAGRASPORIFICI 200

//Nombre d'iteracions per fer els increments amb el teclat de forma incremental
#define ITINCREMENT 1

//Definir el temps cada quan es crida el timer (en milisegons)
#define TEMPSTIMER 15

//Definir el nombre d'iteracions que s'apliquen al control de l'home mort per
aturar el moviment del robot
#define ITCONTROLHOMEMORT 10

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	enum { IDD = IDD_ABOUTBOX };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

// Implementation
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}
```

```
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)

END_MESSAGE_MAP()


// CVirtualRobotDlg dialog


IMPLEMENT_DYNAMIC(CVirtualRobotDlg, CDialog);

CVirtualRobotDlg::CVirtualRobotDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CVirtualRobotDlg::IDD, pParent)
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
	m_pAutoProxy = NULL;
}

CVirtualRobotDlg::~CVirtualRobotDlg()
{
	// If there is an automation proxy for this dialog, set
	//  its back pointer to this dialog to NULL, so it knows
	//  the dialog has been deleted.
	OpenGLDeleteDSPList();
	delete RPQScenario;
	delete m_pDC;

	if (m_pAutoProxy != NULL)
		m_pAutoProxy->m_pDialog = NULL;
}

void CVirtualRobotDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);

	//Establiment de les comunicacions entre les variables de control i els
butons per conectar cada un dels dos robots i el list box per mostrar els
missatges
	DDX_Control(pDX, IDC_BUTTON_SOCKET0, m_CButton_Socket);
	DDX_Control(pDX, IDC_LIST_MSG, m_CListMsg);

	//Establiment de les comunicacions entre les variables de control de
l'informació dels dispositius mestre i esclau i els seus edit boxes
	//Mestre
	DDX_Control(pDX, IDC_MestrePosX, m_MestrePosX);
	DDX_Control(pDX, IDC_MestrePosY, m_MestrePosY);
	DDX_Control(pDX, IDC_MestrePosZ, m_MestrePosZ);
	DDX_Control(pDX, IDC_MestreRot, m_MestreRot);
	//Esclau
	DDX_Control(pDX, IDC_EsclauPosX, m_EsclauPosX);
	DDX_Control(pDX, IDC_EsclauPosY, m_EsclauPosY);
	DDX_Control(pDX, IDC_EsclauPosZ, m_EsclauPosZ);
	DDX_Control(pDX, IDC_EsclauRot, m_EsclauRot);

	//Establiment de les comunicacions entre les variables de control dels
controls implementats i els seus edit boxes o checkboxes
	//Control de l'home mort
	DDX_Control(pDX, IDC_chmLim, m_chmLim);

	//Escalat de la senyal
	DDX_Control(pDX, IDC_escala, m_escala);
```

```
	//Port serie tipus COM per rebre les dades del pedal i el joystick quan
el dispositiu mestre és el Polhemus
	DDX_Control(pDX, IDC_MSCOMM1, serie);

	//Espai de treball restringit
	DDX_Control(pDX, IDC_dimCub, m_dimCub);
	DDX_Control(pDX, IDC_marge, m_marge);

	//Bloqueix de graus de llibertat
	DDX_Check(pDX, IDC_bx, Cbx);
	DDX_Check(pDX, IDC_bx, Cbx);
	DDX_Check(pDX, IDC_bz, Cbz);
	DDX_Check(pDX, IDC_br, Cbr);
	//FI establiment de les comunicacions entre les variables de control dels
controls implementats i els seus edit boxes o checkboxes

	//Establiment de les comunicacions entre les variables de control del
dispositiu mestre seleccionat i els seus checkboxes
	DDX_Check(pDX, IDC_mestrePolhemus, CmestrePolhemus);
	DDX_Check(pDX, IDC_mestreTeclat, CmestreTeclat);
}

BEGIN_MESSAGE_MAP(CVirtualRobotDlg, CDialog)
	ON_WM_SYSCOMMAND()
	ON_WM_CLOSE()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	//}}AFX_MSG_MAP
	ON_BN_CLICKED(IDC_BUTTON_SOCKET0,
&CVirtualRobotDlg::OnBnClickedButtonSocket)
	ON_WM_DRAWITEM()
	ON_WM_LBUTTONDOWN()
	ON_WM_MOUSEMOVE()
	ON_WM_MOUSEWHEEL()
	ON_WM_RBUTTONDOWN()
	ON_WM_MBUTTONDOWN()
	ON_EN_CHANGE(IDC_chmLim, &CVirtualRobotDlg::OnEnChangechmlim)
	ON_EN_CHANGE(IDC_escala, &CVirtualRobotDlg::OnEnChangeescala)
	ON_EN_CHANGE(IDC_marge, &CVirtualRobotDlg::OnEnChangemarge)
	ON_BN_CLICKED(IDC_bx, &CVirtualRobotDlg::OnBnClickedbx)
	ON_BN_CLICKED(IDC_by, &CVirtualRobotDlg::OnBnClickedby)
	ON_BN_CLICKED(IDC_bz, &CVirtualRobotDlg::OnBnClickedbz)
	ON_BN_CLICKED(IDC_br, &CVirtualRobotDlg::OnBnClickedbr)
	ON_WM_TIMER()
	ON_EN_CHANGE(IDC_dimCub, &CVirtualRobotDlg::OnEnChangedimcub)
	ON_BN_CLICKED(IDC_mestrePolhemus,
&CVirtualRobotDlg::OnBnClickedmestrepolhemus)
	ON_BN_CLICKED(IDC_mestreTeclat,
&CVirtualRobotDlg::OnBnClickedmestreteclat)
END_MESSAGE_MAP()


// CVirtualRobotDlg message handlers

BOOL CVirtualRobotDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.
```

```
	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	//Generates RPQScenario
	try{
		RPQScenario = new
CRPQScenario(".\\DataFiles\\Scenario1RobsPinca.sce");
	}
	catch(...)
	{
		MessageBox( "Error", NULL, MB_OK );
		exit(-1);
	}
	//Initialization of RPQScenario
	InitScenario();

	//Inicialització de les variables d'estat de connexió pels dispositius
mestre i esclau
	ConnPolhemus = false; //Polhemus NO connectat via socket
	ConnRobot = false;	//Robot (controladora) NO connectat via socket

	//Inicialització de les variables de control
	RobSel = -1;	//Indica que no hi ha cap robot seleccionat

	OpenGLInit();		//Inicialitzar les estructures OpenGL
	OpenGLInitDspList();//Inicialitzar les DspList

	GetDlgItem(IDC_VIEW)->Invalidate(FALSE);

	////////////////////////////////////////////////////////////////////////
////////////////////////////////
	//////////////INICIALITZACIÓ VARIABLES PER ELS CONTROLS (GENERALS I DE
SEGURETAT) IMPLEMENTATS//////////////

	st=TEMPSTIMER; //Fixar el temps entre mostres del dispositiu mestre
	st=st/1000;
	sr=0.0;
	iPosicio=Vec_3D(0,0,0);
	iRotacio=0;
	//Inicialitzar les posicions i rotacions del mestre i l'esclau que es
mostren a l'interficie
```

```
	CMestrePosX.Format("%f",0.0);
	m_MestrePosX.SetWindowText(CMestrePosX);
	CMestrePosY.Format("%f",0.0);
	m_MestrePosY.SetWindowText(CMestrePosY);
	CMestrePosZ.Format("%f",0.0);
	m_MestrePosZ.SetWindowText(CMestrePosZ);
	CMestreRot.Format("%f",0.0);
	m_MestreRot.SetWindowText(CMestreRot);
	CEsclauPosX.Format("%f",0.0);
	m_EsclauPosX.SetWindowText(CEsclauPosX);
	CEsclauPosY.Format("%f",0.0);
	m_EsclauPosY.SetWindowText(CEsclauPosY);
	CEsclauPosZ.Format("%f",0.0);
	m_EsclauPosZ.SetWindowText(CEsclauPosZ);
	CEsclauRot.Format("%f",0.0);
	m_EsclauRot.SetWindowText(CEsclauRot);

	//Variables de laparoscopia
	instrumentPos=Vec_3D(0,0,0); //Inicialitzar la posició inicial de
l'instrument
	instrumentRot=0.0;
	iPosicioInstrument=Vec_3D(0,0,0);
	iRotacioInstrument=0.0;
	distGraspOrifici=DISTANCIAGRASPORIFICI;//Guardar el punt on es troba
l'orifici a través del qual els instruments laparoscòpis entren al cos del
pacient
	llargadaLaparoscopicInstrument=LLARGADALAPAROSCOPICINSTRUMENT;
	//Guardar la llargada de l'instrument laparoscòpic
	orifici=Vec_3D(0,0,0);//Inicialitzar la posició del orifici
	initOriInst=false;//Indicar que les posicions de l'orifici i l'instrument
encara no s'han inicialitzat (s'inicialitzen quan es connecta amb el robot i
aquest envia la posició inicial)

	//Variables de configuració i control
	//Control de l'home  mort
	chmLim=8000; //Acceleració a partir de la qual s'aplica el control de
l'home mort (8m/s^2)
	CchmLim.Format("%f",chmLim);
	m_chmLim.SetWindowText(CchmLim); //Mostrar l'acceleració límit a
l'interficie

	chmAct=false; //Control de l'home mort desactivat
	chmCont=0; //Contador intern a 0 (no s'ha activat el control encara)
	chmIt=ITCONTROLHOMEMORT; //Nombre d'iteracions que passen desde que
s'activa el control fins que s'atura el moviment del robot
	chmSvm=0; //Mòdul de la velocitat de la lectura anterior on s'ha activat
el control
	chmK=0; //Factor per el descens no lineal de la velocitat
	chmD=Vec_3D(0,0,0); //Direcció de desplaçament utilitzada en el descens
no lineal de velocitat del control de l'home mort(x,y,z)

	//Escalat de la senyal
	escala=1; //Factor d'escala que s'aplica a la senyal del sensor a 1 (no
s'escala la senyal)
	Cescala.Format("%f",escala);
	m_escala.SetWindowText(Cescala); //Mostrar l'escala a l'interficie

	//Espai de treball restringit
	dimCub=DIMCUBESPAITREBALLRESTRINGIT; //Dimensió del cub de l'espai de
treball restringit la màxima del volum de treball del robot(x)
```

```
	CdimCub.Format("%f",dimCub);
	m_dimCub.SetWindowText(CdimCub); //Mostrar la dimensió del cub a
l'interficie
	marge=DIMCUBESPAITREBALLRESTRINGIT/10; //Proximitat respecte els límits
de la caixa contenidora a partir del qual s'activa el control
	Cmarge.Format("%f",marge);
	m_marge.SetWindowText(Cmarge); //Mostrar el marge a l'interficie

	//Bloqueix de graus de llibertat
	bx=false; //Bloqueig del grau de llibertat x desactivat
	by=false; //Bloqueig del grau de llibertat y desactivat
	bz=false; //Bloqueig del grau de llibertat z desactivat
	br=false; //Bloqueig del grau de llibertat de la rotació desactivat

	actOffset=false;//Offset desactivat
	//FI variables de configuració i control

	//Algunes variables s'inicien en el mètode onRecive()

	//////////////FI INICIALITZACIÓ VARIABLES PER ELS CONTROLS (GENERALS I DE
SEGURETAT) IMPLEMENTATS//////////////
	////////////////////////////////////////////////////////////////////////
////////////////////////////////////

	//Indicar quin és el dispositiu mestre seleccionat (per defecte serà el
polhemus)
	CButton* pCheck = (CButton*)GetDlgItem(IDC_mestrePolhemus);
	pCheck->SetCheck(1);
	mestrePolhemus=true;
	mestreTeclat=false;

	//Inicialitzar les variables dels increments del teclat de posició i
rotació
	iPosicioTeclat=Vec_3D(0,0,0);
	iRotacioTeclat=0.0;

	//Inicialitzar el contador i el nombre d'iteracions per fer els
increments amb el teclat
	contIncrement=Vec_3D(0,0,0);
	itIncrement=ITINCREMENT;

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CVirtualRobotDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.
```

```
void CVirtualRobotDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this function to obtain the cursor to display while the
user drags
//  the minimized window.
HCURSOR CVirtualRobotDlg::OnQueryDragIcon()
{
	return static_cast<HCURSOR>(m_hIcon);
}

// Automation servers should not exit when a user closes the UI
//  if a controller still holds on to one of its objects.  These
//  message handlers make sure that if the proxy is still in use,
//  then the UI is hidden but the dialog remains around if it
//  is dismissed.

void CVirtualRobotDlg::OnClose()
{
	if (CanExit())
		CDialog::OnClose();
}

void CVirtualRobotDlg::OnOK()
{
	if (CanExit())
		CDialog::OnOK();
}

void CVirtualRobotDlg::OnCancel()
{
	if (CanExit())
		CDialog::OnCancel();
}

BOOL CVirtualRobotDlg::CanExit()
```

```
{
	// If the proxy object is still around, then the automation
	//  controller is still holding on to this application.  Leave
	//  the dialog around, but hide its UI.
	if (m_pAutoProxy != NULL)
	{
		ShowWindow(SW_HIDE);
		return FALSE;
	}

	return TRUE;
}

void CVirtualRobotDlg::OnRecieve(int id, int nErrorCode){

	//Si és un paquet de dades del Polhemus
	if(id==POLHEMUS){

		//Càlcul dem temps que ha passat entre l'anterior vegada que s'ha
rebut un paquet del Polhemus i s'ha entrat aquí i l'actual
		//Fem servir aquest càlcul per comparar-lo amb el timer i comprovar
que l'execució sigui fluida
		QueryPerformanceCounter((LARGE_INTEGER*)&tFinal); //Guardar el
temps actual
		QueryPerformanceFrequency((LARGE_INTEGER*)&frecuency); //Guardar la
freqüència de l'ordinador
		tCicle=((tFinal-tInicial)/(float)frecuency)*1000.0; //Calcular el
temps que ha passat (en milisegons)
		QueryPerformanceCounter((LARGE_INTEGER*)&tInicial); //Guardem el
temps actual com l'inicial per el següent càlcul

		//Si han passat més de 1 segon desde l'última dada tractada del
Polhemus avisar a l'usuari
		if((tCicle>1000) && !resetmestre){
			CString string;
			string.Format("%f",(float)tCicle);
			m_CListMsg.InsertString(0,"Ha passat massa temps desde
l'última dada del Polhemus tractada("+string+" milisegons)");//Missatge en el
box
			//TancaConnexions();
		}

		CString str;
		ClientsockPolhemus->Receive(&str);//Guardar el missatge del
Polhemus en un string
		CPolhemusManager::PolhemusSensorInfo psi; //Estructura de dades on
es el Polhemus manager ens parsejarà l'informació del sensor
		pm.GetPolhemusSensorInfo(str,psi);//Coloca les dades de l'string
(info del Polhemus) a l'estructura psi

		//Si és la primera lectura del sensor guardem la posició actual com
l'anterior (per evitar salts bruscs)
		if(resetmestre && !mestreTeclat){
			sp=Vec_3D(psi.x*10,psi.y*10,-psi.z*10);
			sr=0;
			sv=Vec_3D(0,0,0);
			resetmestre=false;
			//Inicialitzar les posicions i rotacio del mestre que es
mostren a l'interficie
			CMestrePosX.Format("%f",0.0);
```

```
                m_MestrePosX.SetWindowText(CMestrePosX);
                CMestrePosY.Format("%f",0.0);
                m_MestrePosY.SetWindowText(CMestrePosY);
                CMestrePosZ.Format("%f",0.0);
                m_MestrePosZ.SetWindowText(CMestrePosZ);
                CMestreRot.Format("%f",0.0);
                m_MestreRot.SetWindowText(CMestreRot);
                CEsclauPosX.Format("%f",0.0);
          }

          //Variable  per enviar nova posició del sensor al mètode que
calcula la nova posició de l'instrument després d'aplicar els controls (passar
a mil.límetres)
          Vec_3D rPosicio=Vec_3D(psi.x*10,-psi.y*10,-psi.z*10);

          //Mostrar la nova posició del mestre a l'interficie
          Vec_3D aux=rPosicio-sp;
          float auxX= atof(CMestrePosX);
          float auxY= atof(CMestrePosY);
          float auxZ= atof(CMestrePosZ);
          CMestrePosX.Format("%f",auxX+aux[0]);
          m_MestrePosX.SetWindowText(CMestrePosX);
          CMestrePosY.Format("%f",auxY+aux[1]);
          m_MestrePosY.SetWindowText(CMestrePosY);
          CMestrePosZ.Format("%f",auxZ+aux[2]);
          m_MestrePosZ.SetWindowText(CMestrePosZ);

          //Si l'offset està desactivat calcular la nova posicio del
l'instrument despres d'aplicar els controls (deixa els increments a la variable
global iPosicioInstrument)
          if(!actOffset) calcularPosicioInstrument(rPosicio);
          else{
                sv=(rPosicio-sp)/st; //guardar la velocitat
                sp=rPosicio;//Guardar la nova posició del sensor
                iPosicioInstrument=Vec_3D(0,0,0); //No mourem el dispositiu
esclau
                chmNum=0; //Possar el contador de mostres consecutives que fa
que es supera el control de l'home mort per el Polhemus a 0
          }
     }
     //Si és un paquet que ens envia la controladora (resposta: casi ha
arribat a una posició)
     else if(id==RMAXILO){

          //Si encara em de reiniciar el dispositiu mestre i aquest és el
teclat inicialitzem les variables
          if(resetmestre && mestreTeclat){
                sp=Vec_3D(0,0,0);
                sr=0;
                sv=Vec_3D(0,0,0);
                resetmestre=false;
                //Inicialitzar les posicions i rotacio del mestre que es
mostren a l'interficie
                CMestrePosX.Format("%f",0.0);
                m_MestrePosX.SetWindowText(CMestrePosX);
                CMestrePosY.Format("%f",0.0);
                m_MestrePosY.SetWindowText(CMestrePosY);
                CMestrePosZ.Format("%f",0.0);
                m_MestrePosZ.SetWindowText(CMestrePosZ);
                CMestreRot.Format("%f",0.0);
```

```
                m_MestreRot.SetWindowText(CMestreRot);
                CEsclauPosX.Format("%f",0.0);
            }

            double rposeMAXILO[6];//Variable per guardar la posició on es troba
l'element terminal del robot (el tool si està configurat o sinó l'ancatge de
l'element terminal)
            double rjoints[6];//Variable per guardar les posicions de les
articulacions rebudes de la controladora
            int code;//Variable per guardar el codi d'operació enviat per la
controladora del robot
            int error;//Variable per guardar, si existeix, el codi de l'error
que envia la controladora
            CString m_strController; //Variable per guardar, si existeix,
l'string que indica l'error que envia la controladora

            CString str;
            ClientsockRobot->Receive(&str);//Guardar el missatge de la
controladora en un string

            //Aquest mètode de l'interpret utilitzat parseja l'string i guarda
l'informació extreta a cada variable
            Interpret::capture_data(str, rposeMAXILO, rjoints, &error,
&m_strController, &code);

            //Guardar la posició que retorna el robot
            posMaxilo[0]=rposeMAXILO[0];
            posMaxilo[1]=rposeMAXILO[1];
            posMaxilo[2]=rposeMAXILO[2];

            //Guardar la rotació que retorna el robot
            rotMaxilo=rjoints[5];

            //Si arriba un missatge amb aquest codi vol dir que el robot ja ha
definit el tool, harà se li ha d'enviar la posició inicial.
            if(code==TOOLOP){
                //Enviar posició inicial
                Vec_DP Joints(6);
                Joints[0] = 0.0;   Joints[1] = -68.0; Joints[2] = 170.0;
                Joints[3] = 0.0;   Joints[4] = 55.0;    Joints[5] = 0.0;

      str=Interpret::absolute_joints(Joints[0],Joints[1],Joints[2],Joints[3],Jo
ints[4],Joints[5],TELEOP);
                ClientsockRobot->Send(&str); //Enviar la posició inicial del
robot a la controladora
                return;
            }

            //Si és la primera dada que rebem de la controladora robot la
guardem com a posició inicial
            if(resetrobot){

                //Inicialitzar les posicions i rotació de l'esclau que es
mostren a l'interficie
                CEsclauPosX.Format("%f",0.0);
                m_EsclauPosX.SetWindowText(CEsclauPosX);
                CEsclauPosY.Format("%f",0.0);
                m_EsclauPosY.SetWindowText(CEsclauPosY);
                CEsclauPosZ.Format("%f",0.0);
                m_EsclauPosZ.SetWindowText(CEsclauPosZ);
```

```
                    CEsclauRot.Format("%f",0.0);
                    m_EsclauRot.SetWindowText(CEsclauRot);

                    Vec_3D posAux= Vec_3D(0,0,0);
                    posAux[0]=posMaxilo[0]+DISTLASARTICULATIONGRASPX;
                    posAux[1]=posMaxilo[1]+DISTLASARTICULATIONGRASPY;
                    posAux[2]=posMaxilo[2]+DISTLASARTICULATIONGRASPZ;

                    instrumentRot=rjoints[5];//Guardar la rotació inicial del
robot

                    //Inicialitzem la variable de la velocitat del sensor a 0
                    sv=Vec_3D(0,0,0);

                    //Calcular la distancia entre l'última articulació i el grasp
                    Vec_3D aux= posAux-posMaxilo;

      distLastArticulationGrasp=sqrt(pow(aux[0],2)+pow(aux[1],2)+pow(aux[2],2))
;

                    //Calcular el vector unitari amb la direcció cap on estarà
l'orifici i cap on anirà l'instrument
                    aux= aux/distLastArticulationGrasp;
                    //Calcular la posició de l'orifici per on passen els
instruments
                    orifici=posAux+aux*distGraspOrifici;
                    //Calcular la posició de l'element terminal del braç robòtic
                    instrumentPos= posAux+aux*llargadaLaparoscopicInstrument;
                    //Inicialitzem la posició del sensor al mateix lloc on hi ha
l'instrument
                    sp=instrumentPos;

                    //Indicar que les posicions de l'orifici i l'instrument s'han
incialitzat (per pintar-los en el OpenGLDraw)
                    initOriInst=true;

                    CString str0;
                    CString str1;
                    CString str2;
                    str0.Format("%f",orifici[0]);
                    str1.Format("%f",orifici[1]);
                    str2.Format("%f",orifici[2]);
                    m_CListMsg.InsertString(0,"Posició de l'orifici fixada al
punt: ("+str0+","+str1+","+str2+")");//Missatge en el box
                    str0.Format("%f",instrumentPos[0]);
                    str1.Format("%f",instrumentPos[1]);
                    str2.Format("%f",instrumentPos[2]);
                    m_CListMsg.InsertString(0,"Posició de l'instrument fixada al
punt: ("+str0+","+str1+","+str2+")");//Missatge en el box

                    //Guardar el centre de l'espai de treball restringit
                    centreEspaiTreballRestringit=
Vec_3D(instrumentPos[0],instrumentPos[1],instrumentPos[2]);

                    //Posarar el reset a fals
                    resetrobot=false;

          }
```

```
            //Sinó modifiquem la informació de la posició del robot a
l'interficie (centreEspaiTreballRestringit manté la posició inicial de
l'instrument respecte els eixos del món)
            else{
                  //Mostrar la nova posició del mestre a l'interficie
                  CEsclauPosX.Format("%f",instrumentPos[0]-
centreEspaiTreballRestringit[0]);
                  m_EsclauPosX.SetWindowText(CEsclauPosX);
                  CEsclauPosY.Format("%f",instrumentPos[1]-
centreEspaiTreballRestringit[1]);
                  m_EsclauPosY.SetWindowText(CEsclauPosY);
                  CEsclauPosZ.Format("%f",instrumentPos[2]-
centreEspaiTreballRestringit[2]);
                  m_EsclauPosZ.SetWindowText(CEsclauPosZ);
            }

            //Si el braç robòtic no ha tingut errors i el codi del missatge és
de moviment
            if(error==0  && (code==TELEOP)){

                  //Variable per indicar si hi ha algun increment del teclat
pendent d'enviar
                  bool enviarIncrementTeclat=false;

                  //Si el robot es controla amb el teclat i hi ha algun
increment per fer es necessari calcular els increments que s'han de passar en
aquest mètode. Si el control de l'home mort està actiu també és necessari
                  if(mestreTeclat && (iPosicioTeclat[0]!=0 ||
iPosicioTeclat[1]!=0 || iPosicioTeclat[2]!=0 || chmAct)){
                        //Calcular la nova posició del
sensor(anterior+increment)
                        Vec_3D rPosicio=sp+iPosicioTeclat;

                        //Mostrar la nova posició del mestre a l'interficie
                        float auxX= atof(CMestrePosX);
                        float auxY= atof(CMestrePosY);
                        float auxZ= atof(CMestrePosZ);
                        CMestrePosX.Format("%f",auxX+iPosicioTeclat[0]);
                        m_MestrePosX.SetWindowText(CMestrePosX);
                        CMestrePosY.Format("%f",auxY+iPosicioTeclat[1]);
                        m_MestrePosY.SetWindowText(CMestrePosY);
                        CMestrePosZ.Format("%f",auxZ+iPosicioTeclat[2]);
                        m_MestrePosZ.SetWindowText(CMestrePosZ);

                        //Si ja hem fet totes les iteracions posem els
increments a 0 (moviment acabat)
                        if(contIncrement[0]==itIncrement){
                              iPosicioTeclat[0]=0;
                        }
                        else{
                              //Incrementar el contador corresponent
                              contIncrement[0]+=1;
                        }
                        //Si ja hem fet totes les iteracions posem els
increments a 0 (moviment acabat)
                        if(contIncrement[1]==itIncrement){
                              iPosicioTeclat[1]=0;
                        }
                        else{
                              //Incrementar el contador corresponent
```

```
                        contIncrement[1]+=1;
                    }
                    //Si ja hem fet totes les iteracions posem els
increments a 0 (moviment acabat)
                    if(contIncrement[2]==itIncrement){
                        iPosicioTeclat[2]=0;
                    }
                    else{
                        //Incrementar el contador corresponent
                        contIncrement[2]+=1;
                    }

                    //Indicar que hi ha un increment del teclat pendent
d'enviar al robot
                    enviarIncrementTeclat=true;

                    //Si l'offset està desactivat calcular la nova posicio
del l'instrument despres d'aplicar els controls (deixa els increments a la
variable global iPosicioInstrument)
                    if(!actOffset) calcularPosicioInstrument(rPosicio);
                    else{
                        sv=(rPosicio-sp)/st; //guardar la velocitat el
mestre
                        sp=rPosicio;//Guardar la nova posició del mestre
                        iPosicioInstrument=Vec_3D(0,0,0); //No mourem el
dispositiu esclau
                    }

                    //Calcular la nova posicio de l'instrument
                    instrumentPos=instrumentPos+iPosicioInstrument;

                    //Càlcul de la posició que es correspon a l'element
terminal del robot per tal de realitzar laparoscòpies
                    //Calcular el vector direcció entre el punt que marca
el sensor (punt de treball de l'instrument laparoscòpic) i l'orifici d'entrada
dels instruments
                    Vec_3D dir= orifici-instrumentPos;
                    //Calcular el mòdul del vector direcció (per fer-lo
unitari)
                    float modul=
sqrt(pow(dir[0],2)+pow(dir[1],2)+pow(dir[2],2));
                    //Calcular el vector de direcció unitari
                    Vec_3D dirU= dir/modul;
                    //Calcular la posició del grasp del braç robòtic
                    rPosicio=
instrumentPos+(dirU*llargadaLaparoscopicInstrument);
                    //Calcular la posició de l'última articulacio del braç
robòtic (la que se li ha de passar al robot)
                    rPosicio=rPosicio-
Vec_3D(DISTLASARTICULATIONGRASPX,DISTLASARTICULATIONGRASPY,DISTLASARTICULATIONG
RASPZ);
                    //Fi càlcul de la posició que es correspon a l'element
terminal del robot per tal de realitzar laparoscòpies

                    //Calcular l'increment que s'ha de passar al robot
                    iPosicio=rPosicio-posMaxilo;

                    //Moviment directe sense laparoscòpia
                    //iPosicio=iPosicioTeclat;
                }
```

```
                  //Mestre Polhemus calcular l'increment
                  else if(!mestreTeclat){

                        //Calcular la nova posicio de l'instrument
                        instrumentPos=instrumentPos+iPosicioInstrument;

                        iPosicioInstrument=Vec_3D(0,0,0);//Deixem l'increment a
0

                        //Càlcul de la posició que es correspon a l'element
terminal del robot per tal de realitzar laparoscòpies
                        //Calcular el vector direcció entre el punt que marca
el sensor (punt de treball de l'instrument laparoscòpic) i l'orifici d'entrada
dels instruments
                        Vec_3D dir= orifici-instrumentPos;
                        //Calcular el mòdul del vector direcció (per fer-lo
unitari)
                        float modul=
sqrt(pow(dir[0],2)+pow(dir[1],2)+pow(dir[2],2));
                        //Calcular el vector de direcció unitari
                        Vec_3D dirU= dir/modul;
                        //Calcular la posició del grasp del braç robòtic
                        Vec_3D rPosicio=
instrumentPos+(dirU*llargadaLaparoscopicInstrument);
                        //Calcular la posició de l'última articulacio del braç
robòtic (la que se li ha de passar al robot)
                        rPosicio=rPosicio-
Vec_3D(DISTLASARTICULATIONGRASPX,DISTLASARTICULATIONGRASPY,DISTLASARTICULATIONG
RASPZ);
                        //Fi càlcul de la posició que es correspon a l'element
terminal del robot per tal de realitzar laparoscòpies

                        //Calcular l'increment que s'ha de passar al robot
                        iPosicio=rPosicio-posMaxilo;
                  }

                  //Si el robot virtual de l'interficie no dona errors durant
la cinemàtica inversa enviar la nova posició a la controladora
                  if (RPQScenario->L_PQPRob[RMAXILO]-
>SetBaseToToolZYiZii(rposeMAXILO[0],rposeMAXILO[1],rposeMAXILO[2],rposeMAXILO[3
]*PI/180,rposeMAXILO[4]*PI/180,rposeMAXILO[5]*PI/180))
                  {
                        //Si controlem el robot amb el Polhemus enviem sempre
un nou increment
                        if(!mestreTeclat){
                              //Construir l'string que s'ha d'enviar a la
controladora del robot per tal de que es mogui a la nova posició del Polhemus
                              //El valor TELEOP servirà per indicar que el
missatge que s'envia a la controladora és de moviment (quan la controladora
contesti a aquest missatge per dir que casi ha arribat a la posició desitjada
també utilitzara aquest codi)
                              str =
Interpret::incremental_cartesian(iPosicio[0],iPosicio[1],iPosicio[2],0,0,0,TELE
OP);
                              //Si el robot esta connectat enviar l'increment
                              if(ConnRobot) ClientsockRobot->Send(&str);
//Enviar la nova posició al robot
                        }
```

```
                        //Si es controla amb el teclat només s'envia una nova
posició si algun increment és diferent de 0 o el control de l'home mort està
actiu i l'offset no ho està
                        else{
                              if((enviarIncrementTeclat || chmAct) &&
!actOffset){

                                    //Construir l'string que s'ha d'enviar a la
controladora del robot per tal de que es mogui a la nova posició del Polhemus
                                    //El valor TELEOP ens servirà per indicar
que el missatge que s'envia a la controladora és de moviment (quan la
controladora contesti a aquest missatge per dir que casi ha arribat a la
posició desitjada també utilitzara aquest codi)
                                    str =
Interpret::incremental_cartesian(iPosicio[0],iPosicio[1],iPosicio[2],0,0,0,TELE
OP);

                                    //Si el robot esta connectat se li envia
l'increment
                                    if(ConnRobot) ClientsockRobot->Send(&str);

                              }
                        }
                  }
                  //Si el robot virtual de l'interficie dona errors durant la
cinemàtica inversa
                  else{
                        m_CListMsg.InsertString(0,"El robot virtual de
l'interficie ha donat errors en la cinemàtica inversa");//Missatge en el box

                        //Construir l'string que s'ha d'enviar a la
controladora del robot per tal de que es mogui a la nova posició del Polhemus
                        str =
Interpret::incremental_cartesian(iPosicio[0],iPosicio[1],iPosicio[2],iRotacio,0
,0,TELEOP);
                        //Si el robot esta connectat se li envia l'increment
                        if(ConnRobot) ClientsockRobot->Send(&str);
                  }
            }
            //Si hi ha hagut algun error en el moviment ho indiquem a
l'interficie i no movem el robot
            else if (error!=0 && (code==TELEOP)){
                  m_CListMsg.InsertString(0,"Error en el moviment del braç
robòtic: "+m_strController);//Missatge en el box

                  //Enviar increments 0 per mantenir el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                  str = Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOP);
                  //Si el robot esta connectat enviar l'increment
                  if(ConnRobot) ClientsockRobot->Send(&str);
            }
            else if(error==0 && (code==TELEOPROTATION)){

                  float iRotacioAux=iRotacioTeclat;

                  //Mostrar la nova rotació del dispositiu mestre a
l'interficie
                  float aux= atof(CMestreRot);
                  CMestreRot.Format("%f",aux+iRotacioTeclat*(-1));
```

```
                m_MestreRot.SetWindowText(CMestreRot);

                iRotacioTeclat=0;

                //Si l'offset o el control de l'home mort estan actius no es
fara cas de la rotació
                if(actOffset || chmAct){
                      iRotacioInstrument=0.0;
                }
                else{
                      sr=sr+iRotacioAux;//Guardar la nova rotació
                      iRotacioAux=iRotacioAux*escala; //Escalar l'increment
rotació
                      iRotacioInstrument=iRotacioAux;//Calcular l'increment
d'angle que se li passarà al robot
                }
                if(br)iRotacioInstrument=0;

                instrumentRot=instrumentRot+iRotacioInstrument;//Calcular la
nova rotació de l'instrument
                iRotacio=iRotacioInstrument;

                //Mostrar la nova rotació del dispositiu esclau a
l'interficie
                CEsclauRot.Format("%f",rotMaxilo*(-1));
                m_EsclauRot.SetWindowText(CEsclauRot);

                //Pintar el robot de l'interficie i enviar el nou increment
d'orientació
                if(RPQScenario->L_PQPRob[RMAXILO]-
>SetBaseToToolZYiZii(rposeMAXILO[0],rposeMAXILO[1],rposeMAXILO[2],rposeMAXILO[3
]*PI/180,rposeMAXILO[4]*PI/180,rposeMAXILO[5]*PI/180)){
                      //Només enviem la rotació és diferent de 0
                      if(iRotacio!=0){
                            //Enviar increments 0 per mantenir el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                            str=
Interpret::incremental_joints(0,0,0,0,0,iRotacio,TELEOPROTATION);
                            //Si el robot esta connectat li enviem
l'increment
                            if(ConnRobot) ClientsockRobot->Send(&str);
//Enviar la nova posició al robot
                      }
                }
                iRotacio=0;
          }
          else if(error!=0 && (code==TELEOPROTATION)){
                m_CListMsg.InsertString(0,"Error en el moviment del braç
robòtic: "+m_strController);//Missatge en el box

                //Enviar increments 0 per mantenir el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                str=
Interpret::incremental_joints(0,0,0,0,0,0,TELEOPROTATION);
                //Si el robot esta connectat li enviem l'increment
                if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
          }
```

```
	}
}

void CVirtualRobotDlg::calcularPosicioInstrument(Vec_3D p){

	CString str0;

	//CONTROL DE L'HOME MORT
	float vcontrol;

	//Si control de l'home mort s'ha activat en una lectura anterior i no hem
fet totes les iteracions (robot no aturat)
	if(chmAct && (chmIt>chmCont)){

		//Incrementem el contador d'iteracions
		chmCont++;

		//Calcular la nova velocitat de control
		vcontrol= chmSvm-(0.5*pow(chmK*chmCont,2));

		//Calcular la nova posició tenint en conte el control
		p= sp+(chmD*(vcontrol/chmSvm));

		//Guardar el vector de velocitat de desplaçament nou (no serà real)
		sv=chmD*(vcontrol/chmSvm);

		//Assegurar que a l'última iteració la velocitat sigui 0
		if(chmCont==chmIt) sv=Vec_3D(0,0,0);

		str0.Format("%d",chmCont);
		CString str1;
		str1.Format("%f",vcontrol/chmSvm);
		m_CListMsg.InsertString(0,"Home mort actiu (it:"+str0+") velocitat
reduida a un "+str1+"% del total");//Missatge en el box

	}
	//Si el control s'ha activa i ja hem fet totes les iteracions (robot
aturat)
	else if(chmAct && (chmCont=chmIt)){

		//Control de seguretat adicional per no sobrepassar els limits del
cub de l'espai de treball restringit
		if(instrumentPos[0]>(centreEspaiTreballRestringit[0]+(dimCub/2))) {
			instrumentPos[0]=centreEspaiTreballRestringit[0]+(dimCub/2);
		}
		else if(instrumentPos[0]<(centreEspaiTreballRestringit[0]-
(dimCub/2))) {
			instrumentPos[0]=centreEspaiTreballRestringit[0]-(dimCub/2);
		}
		if(instrumentPos[1]>(centreEspaiTreballRestringit[1]+(dimCub/2))) {
			instrumentPos[1]=centreEspaiTreballRestringit[1]+(dimCub/2);
		}
		else if(instrumentPos[1]<(centreEspaiTreballRestringit[1]-
(dimCub/2))) {
			instrumentPos[1]=centreEspaiTreballRestringit[1]-(dimCub/2);
		}
		if(instrumentPos[2]>(centreEspaiTreballRestringit[2]+(dimCub/2))) {
			instrumentPos[2]=centreEspaiTreballRestringit[2]+(dimCub/2);
		}
```

```
            else if(instrumentPos[2]<(centreEspaiTreballRestringit[2]-
(dimCub/2))) {
                  instrumentPos[2]=centreEspaiTreballRestringit[2]-(dimCub/2);
            }

            m_CListMsg.InsertString(0,"Control de l'home mort actiu totes les
iteracions realizades (robot aturat)");//Missatge en el box
            chmAct=false;

            TancaConnexions(); //Tancar les connexions si s'està connectat al
Polhemus que sempre envia posicions

            //Deixar el robot en la mateixa posició que estava
            return;
      }
      //Si el control de l'home mort no s'ha activat en una lectura anterior
      else{
            //Calcular el increment de posició entre la lectura actual i
l'anterior .
            Vec_3D i= p - sp;

            //Calcular el vector 3d de la velocitat de desplaçament en l'últim
interval de temps
            Vec_3D v= i/st;

            //Calcular el vector 3d de l'acceleració en l'últim interval de
temps
            Vec_3D a= (v-sv)/st;

            //Calcular el mòdul de l'acceleració
            float ac= sqrt(pow(a[0],2)+pow(a[1],2)+pow(a[2],2));

            //Per activar el control de l'home mort s'ha de superar
l'acceleració mínima durant 6 de posició del dispositiu mestre
            if(((ac-5000)>chmLim) && mestrePolhemus){
                  chmNum++;
                  p=sp;
            }
            else{
                  chmNum=0;
            }

            //Si l'acceleració sobrepassa el límit del control de l'home mort
            if(((ac>chmLim) && mestreTeclat) || ((chmNum>5)&& ((ac-
5000)>chmLim) && mestrePolhemus)){

                  str0.Format("%f",ac);
                  m_CListMsg.InsertString(0,"S'ha activat el control de l'home
mort. Acceleracio de: "+str0);//Missatge en el box

                  chmAct=true; //Activar el control

                  //Si el robot estava parat a la lectura anterior ja no es mou
(no cal fer el descens no lineal de la velocitat)
                  if(sv[0]==0 && sv[1]==0 && sv[2]==0){
                        chmCont=chmIt;
                        m_CListMsg.InsertString(0,"Control de l'home mort actiu
totes les iteracions realizades (robot aturat)");//Missatge en el box
                        //TancaConnexions(); //Tancar les connexions
                        return;
```

```
                  }
                  //Si el robot no estava parat a la lectura anterior del
sensor
                  else{
                        //Posar el contador d'iteracions des que s'ha activat
el control a 1
                        chmCont=1;

                        //Calcular el mòdul de la velocitat de la lectura
anterior
                        chmSvm= sqrt(pow(sv[0],2)+pow(sv[1],2)+pow(sv[2],2));

                        //Calcular el coeficient k per el descens       no
lineal de la velocitat
                        chmK= sqrt((2*chmSvm)/(chmIt*chmIt));

                        //Calcular la velocitat de control per tal de fer el
descens no lineal
                        vcontrol= chmSvm - (0.5*pow(chmK*chmCont,2));

                        //Calcular el vector de desplaçament de la lectura
anterior per utilitzar-lo en el control després de fer-lo unitari
(velocitat*temps)
                        chmD= sv*st;

                        //Calcular la nova posició tenint en conte el control
                        p= sp+(chmD*(vcontrol/chmSvm));

                        //Guardar el vector de velocitat de desplaçament
                        sv= chmD*(vcontrol/chmSvm);
                  }
            }
            //Si no s'activa el control de l'home mort
            else{
                  //Guardar el vector de velocitat de desplaçament
                  sv=v;
            }
      }

      //FI CONTROL DE L'HOME MORT

      //Control de seguretat adicional per no sobrepassar els limits del cub de
l'espai de treball restringit
      if(instrumentPos[0]>(centreEspaiTreballRestringit[0]+(dimCub/2))) {
            instrumentPos[0]=centreEspaiTreballRestringit[0]+(dimCub/2);
      }
      else if(instrumentPos[0]<(centreEspaiTreballRestringit[0]-(dimCub/2))) {
            instrumentPos[0]=centreEspaiTreballRestringit[0]-(dimCub/2);
      }
      if(instrumentPos[1]>(centreEspaiTreballRestringit[1]+(dimCub/2))) {
            instrumentPos[1]=centreEspaiTreballRestringit[1]+(dimCub/2);
      }
      else if(instrumentPos[1]<(centreEspaiTreballRestringit[1]-(dimCub/2))) {
            instrumentPos[1]=centreEspaiTreballRestringit[1]-(dimCub/2);
      }
      if(instrumentPos[2]>(centreEspaiTreballRestringit[2]+(dimCub/2))) {
            instrumentPos[2]=centreEspaiTreballRestringit[2]+(dimCub/2);
      }
      else if(instrumentPos[2]<(centreEspaiTreballRestringit[2]-(dimCub/2))) {
            instrumentPos[2]=centreEspaiTreballRestringit[2]-(dimCub/2);
```

```
	}
	//FI control de seguretat adicional per no sobrepassar els limits del cub
de l'espai de treball restringit

	//Guardar l'increment que s'ha donat al sensor
	Vec_3D incrementSensor=p-sp;

	//Guardar la nova posició del sensor(amb el control de l'home mort actiu
no seran reals)
	sp=p;

	//ESCALAT DE LA SENYAL
	//Calcular la nova posició de l'instrument aplicant el factor d'escala.
	Vec_3D pInstr=instrumentPos+(incrementSensor*escala);
	//FI ESCALAT DE LA SENYAL

	//CONTROL ESPAI DE TREBALL RESTRINGIT
	//Inicialitzar el vector temporal per guardar les distàncies que
sobrepassen el marge per cada eix
	Vec_3D cetDist= Vec_3D(0,0,0);

	//Inicialitzar el vector auxiliar que indica si s'ha d'activar el control
de l'espai restringit per cada eix (0->desactivat, 1->activat)
	Vec_3D cetAct= Vec_3D(0,0,0);

	//Inicialitzar el vector auxiliar que indica el factor utilitzat per el
descens no lineal de la velocitat de cada eix
	Vec_3D cetK= Vec_3D(0,0,0);

	//Calcular la velocitat del l'instrument derivada de la lectura actual
	Vec_3D rv= (pInstr-instrumentPos)/st;

	//Inicialitzar el vector amb les velocitats de control per cada eix
	Vec_3D rvCtr=rv;

	//Vector auxiliar per indicar que no mourem el robot en cada un dels
eixos (0->es moura, 1->no es moura)
	Vec_3D auxLim= Vec_3D(0,0,0);

	//Comprovar que l'increment de posició no sigui tant gran que sortim de
la caixa que conté l'espai de treball restringit en tots els eixos
	if((pInstr[0]>(centreEspaiTreballRestringit[0]+(dimCub/2))) ||
(pInstr[0]<(centreEspaiTreballRestringit[0]-(dimCub/2)))){
		auxLim[0]=1;//Indicar que per aquest eix no mourem el robot
		m_CListMsg.InsertString(0,"Amb l'escala actual es sobrepassa el
limit de l'eix x saltanse el marge");
		m_CListMsg.InsertString(0,"No es realitzarà el moviment");
	}
	if((pInstr[1]>(centreEspaiTreballRestringit[1]+(dimCub/2))) ||
(pInstr[1]<(centreEspaiTreballRestringit[1]-(dimCub/2)))){
		auxLim[1]=1;//Indicar que per aquest eix no mourem el robot
		m_CListMsg.InsertString(0,"Amb l'escala actual es sobrepassa el
limit de l'eix y saltanse el marge");
		m_CListMsg.InsertString(0,"No es realitzarà el moviment");
	}
	if((pInstr[2]>(centreEspaiTreballRestringit[2]+(dimCub/2))) ||
(pInstr[2]<(centreEspaiTreballRestringit[2]-(dimCub/2)))){
		auxLim[2]=1;//Indicar que per aquest eix no mourem el robot
		m_CListMsg.InsertString(0,"Amb l'escala actual es sobrepassa el
limit de l'eix z saltanse el marge");
```

```
            m_CListMsg.InsertString(0,"No es realitzarà el moviment");
      }

      bool davant=pInstr[0]>(centreEspaiTreballRestringit[0]+((dimCub/2)-
marge));
      bool darrere=pInstr[0]<(centreEspaiTreballRestringit[0]-((dimCub/2)-
marge));

      //Si la posició en l'eix x del robot sobrepassa el marge establert per
activar el control en algun dels dos extrems i l'increment de posició no es
tant gran que sortim de la caixa que conté l'espai de treball restringit
      if((davant || darrere) && (auxLim[0]==0)){

            //Guardar la distància que sobrepassa el marge de seguretat
            //Si el passa per la part positiva del centre del cub i la
velocitat és positiva (no ens allunyem del marge)
            if(davant && (rv[0]>0)){
                  //Activar el control per aquest eix
                  cetAct[0]=1;
                  cetDist[0]= pInstr[0]-
(centreEspaiTreballRestringit[0]+((dimCub/2)-marge));
            }
            //Si el passa per la part negativa del centre del cub i la
velocitat és negativa (no ens estem allunyant del marge)
            else if(darrere && (rv[0]<0)){
                  //Activar el control per aquest eix
                  cetAct[0]=1;
                  cetDist[0]= abs(pInstr[0]-(centreEspaiTreballRestringit[0]-
((dimCub/2)-marge)));
            }
            if(cetAct[0]){
                  str0.Format("%f",cetDist[0]);
                  m_CListMsg.InsertString(0,"Descens no lineal de velocitat en
l'eix x actiu");
                  m_CListMsg.InsertString(0,"Distancia que sobrepassa el marge:
"+str0);//Missatge en el box
            }
      }

      bool dreta=pInstr[1]>(centreEspaiTreballRestringit[1]+((dimCub/2)-
marge));
      bool esquerra=pInstr[1]<(centreEspaiTreballRestringit[1]-((dimCub/2)-
marge));

      //Si la posició en l'eix y del robot sobrepassa el marge establert per
activar el control en algun dels dos extrems i l'increment de posició no es
tant gran que sortim de la caixa que conté l'espai de treball restringit
      if((dreta || esquerra) && (auxLim[1]==0)){

            //Guardar la distància que sobrepassa el marge de seguretat
            //Si el passa per la part positiva del centre del cub i la
velocitat és positiva (no ens allunyem del marge)
            if(dreta && (rv[1]>0)){
                  //Activar el control per aquest eix
                  cetAct[1]=1;
                  cetDist[1]= pInstr[1]-
(centreEspaiTreballRestringit[1]+((dimCub/2)-marge));
            }
            //Si el passa per la part negativa del centre del cub i la
velocitat és negativa (no ens estem allunyant del marge)
```

```
            else if(esquerra && (rv[1]<0)){
                  //Activar el control per aquest eix
                  cetAct[1]=1;
                  cetDist[1]= abs(pInstr[1]-(centreEspaiTreballRestringit[1]-
((dimCub/2)-marge)));
            }
            if(cetAct[1]){
                  str0.Format("%f",cetDist[1]);
                  m_CListMsg.InsertString(0,"Descens no lineal de velocitat en
l'eix y actiu");
                  m_CListMsg.InsertString(0,"Distancia que sobrepassa el marge:
"+str0);//Missatge en el box
            }
      }

      bool up=pInstr[2]>(centreEspaiTreballRestringit[2]+((dimCub/2)-marge));
      bool down=pInstr[2]<(centreEspaiTreballRestringit[2]-((dimCub/2)-marge));

      //Si la posició en l'eix z del robot sobrepassa el marge establert per
activar el control en algun dels dos extrems i l'increment de posició no es
tant gran que sortim de la caixa que conté l'espai de treball restringit
      if((up || down) && (auxLim[2]==0)){

            //Guardar la distància que sobrepassa el marge de seguretat
            //Si el passa per la part positiva del centre del cub i la
velocitat és positiva (no ens allunyem del marge)
            if(up && (rv[2]>0)){
                  //Activar el control per aquest eix
                  cetAct[2]=1;
                  cetDist[2]= pInstr[2]-
(centreEspaiTreballRestringit[2]+((dimCub/2)-marge));
            }
            //Si el passa per la part negativa del centre del cub i la
velocitat és negativa (no ens estem allunyant del marge)
            else if(down && (rv[2]<0)){
                  //Activar el control per aquest eix
                  cetAct[2]=1;
                  cetDist[2]= abs(pInstr[2]-(centreEspaiTreballRestringit[2]-
((dimCub/2)-marge)));
            }
            if(cetAct[2]){
                  str0.Format("%f",cetDist[2]);
                  m_CListMsg.InsertString(0,"Descens no lineal de velocitat en
l'eix z actiu");
                  m_CListMsg.InsertString(0,"Distancia que sobrepassa el marge:
"+str0);//Missatge en el box
            }
      }

      //Si s'ha activat el control de l'espai de treball restringit per l'eix x
      if(cetAct[0]==1){

            //Calcular el factor k per el descens no lineal de la velocitat per
aquest eix
            cetK[0]= sqrt((2*abs(rv[0]))/pow(marge,2));

            if(rv[0]>0){
                  //Calcular la velocitat de control del robot per aquest eix
                  rvCtr[0]= rv[0]- 0.5*pow((cetK[0]*cetDist[0]),2);
            }
```

```
            else{
                  //Calcular la velocitat de control del robot per aquest eix
                  rvCtr[0]= rv[0]+ 0.5*pow((cetK[0]*cetDist[0]),2);
            }
      }

      //Si s'ha activat el control de l'espai de treball restringit per l'eix y
      if(cetAct[1]==1){

            //Calcular el factor k per el descens no lineal de la velocitat per
aquest eix
            cetK[1]= sqrt((2*abs(rv[1]))/pow(marge,2));

            if(rv[1]>0){
                  //Calcular la velocitat de control del robot per aquest eix
                  rvCtr[1]= rv[1]- 0.5*pow((cetK[1]*cetDist[1]),2);
            }
            else{
                  //Calcular la velocitat de control del robot per aquest eix
                  rvCtr[1]= rv[1]+ 0.5*pow((cetK[1]*cetDist[1]),2);
            }
      }

      //Si s'ha activat el control de l'espai de treball restringit per l'eix z
      if(cetAct[2]==1){

            //Calcular el factor k per el descens no lineal de la velocitat per
aquest eix
            cetK[2]= sqrt((2*abs(rv[2]))/pow(marge,2));

            if(rv[2]>0){
                  //Calcular la velocitat de control del robot per aquest eix
                  rvCtr[2]= rv[2]- 0.5*pow((cetK[2]*cetDist[2]),2);
            }
            else{
                  //Calcular la velocitat de control del robot per aquest eix
                  rvCtr[2]= rv[2]+ 0.5*pow((cetK[2]*cetDist[2]),2);
            }
      }

      //Si el robot no està aturat en l'eix x (evitar divisió entre 0) i
l'increment de posició no es tant gran que sortim de la caixa que conté l'espai
de treball restringit
      if(rv[0]!=0 && (auxLim[0]==0)){
            //Calcular la nova posició del robot aplicant el control en aquesta
component
            pInstr[0]=instrumentPos[0]+ ((rvCtr[0]/rv[0]) * (pInstr[0]-
instrumentPos[0]));
      }
      //Si el robot està aturat en aquest eix la posició és la mateixa que
l'anterior
      else{
            pInstr[0]=instrumentPos[0];
      }

      //Si el robot no està aturat en l'eix y (evitar divisió entre 0) i
l'increment de posició no es tant gran que sortim de la caixa que conté l'espai
de treball restringit
      if(rv[1]!=0 && (auxLim[1]==0)){
```

```
            //Calcular la nova posició del robot aplicant el control en aquesta
component
            pInstr[1]=instrumentPos[1]+ ((rvCtr[1]/rv[1]) * (pInstr[1]-
instrumentPos[1]));
      }
      //Si el robot està aturat en aquest eix la posició és la mateixa que
l'anterior
      else{
            pInstr[1]=instrumentPos[1];
      }

      //Si el robot no està aturat en l'eix z (evitar divisió entre 0) i
l'increment de posició no es tant gran que sortim de la caixa que conté l'espai
de treball restringit
      if(rv[2]!=0 && (auxLim[2]==0)){

            //Calcular la nova posició del robot aplicant el control en aquesta
component
            pInstr[2]=instrumentPos[2]+ ((rvCtr[2]/rv[2]) * (pInstr[2]-
instrumentPos[2]));
      }
      //Si el robot està aturat en aquest eix la posició és la mateixa que
l'anterior
      else{
            pInstr[2]=instrumentPos[2];
      }

      if(mestrePolhemus){
            iPosicioInstrument=iPosicioInstrument+(pInstr-instrumentPos);

      }
      else iPosicioInstrument=pInstr-instrumentPos;

      //Per els graus de llibertats bloquejats el increment serà 0
      if(bx){
            iPosicioInstrument[0]=0;
      }
      if(by){
            iPosicioInstrument[1]=0;
      }
      if(bz){
            iPosicioInstrument[2]=0;
      }
}

void CVirtualRobotDlg::OnClose(int id, int nErrorCode)
{

}

void CVirtualRobotDlg::OnBnClickedButtonSocket(){

      //Si es vol controlar el robot amb el Polhemus obrir una connexió amb ell
      if(!mestreTeclat){
            if (!ConnPolhemus) //Si el Polhemus NO està connectat via socket
(si es clica el botó es per connectar)
            {
                  //Deshabilitar els edit controls per canviar les dimensions
de l'espai restringit i el marge
                  m_dimCub.EnableWindow(false);
```

```
                  m_marge.EnableWindow(false);
                  //Deshabilitar els checkboxes per canviar de dispositiu
mestre
                  CButton* pCheck = (CButton*)GetDlgItem(IDC_mestreTeclat);
                  pCheck->EnableWindow(false);
                  pCheck = (CButton*)GetDlgItem(IDC_mestrePolhemus);
                  pCheck->EnableWindow(false);

                  //CREAR SOCKET PER EL POLHEMUS
                  ClientsockPolhemus = new COurSocket(this, POLHEMUS);//Crear
un socket nou amb l'id del polhemus
                  int error=ClientsockPolhemus-
>CreateAndConnect(PORT_POLHEMUS,_T(IP_POLHEMUS));//Conectar amb el Polhemus
(falta el port i la IP)
                  m_CListMsg.InsertString(0,"Establint connexió el Polhemus
...");//Informació en el box de missatges
                  //FI CREAR SOCKET POLHEMUS

                  //Port COM
                  serie.SetPortOpen(true); //Obrir el port per rebre les dades
del pedal i del joystick del mànec sensoritzat
            }
            else //Si el Polhemus està connectat via socket (si es clica el
botó es per desconnectar)
            {
                  TancaConnexions();
            }
      }
      //Si es vol controlar el robot amb el teclat obrir la connexió amb el
robot directament
      else{
            if(!ConnRobot){
                  //Desabilitar els edit controls per canviar les dimensions de
l'espai restringit i el marge
                  m_dimCub.EnableWindow(false);
                  m_marge.EnableWindow(false);
                  //Deshabilitar els checkboxes per canviar de dispositiu
mestre
                  CButton* pCheck = (CButton*)GetDlgItem(IDC_mestreTeclat);
                  pCheck->EnableWindow(false);
                  pCheck = (CButton*)GetDlgItem(IDC_mestrePolhemus);
                  pCheck->EnableWindow(false);
                  m_CButton_Socket.SetWindowText("Desconnectar"); //Canviar el
text del boto per connectar/desconnectar el Polhemus i el robot (s'hi està
connectat)

                  //CREAR SOCKET PER EL ROBOT (controladora)
                  ClientsockRobot = new COurSocket(this, RMAXILO); //Crear un
socket nou amb l'id del robot
                  int error=ClientsockRobot-
>CreateAndConnect(PORT_ROBOT_MAXILO,_T(IP_ROBOT_MAXILO));//Conectar amb el
RobotControllerEmulator?¿
                  m_CListMsg.InsertString(0,"Establint connexió amb
Maxilo....");//Informació en el box de missatges
                  //FI CREAR SOCKET ROBOT (controladora)
            }
            else TancaConnexions();
      }
```

```
	UpdateData(false);
}

//Mètode que es crida per desconectar el socket del Polhemus i el de la
controladora (si està conectada)
void CVirtualRobotDlg::TancaConnexions(){

		//Desactivar el timer que demana dades al Polhemus i pinta el robot
virtual
		KillTimer(DATO_POLHEMUS);

		//Si el robot (controladora) també connectat via socket el
desconectem
		if(ConnRobot){
			//TANCAMENT DE CONNEXIÓ AMB EL ROBOT (controladora)
			CString str;
			str = Interpret::close_connexion(RMAXILO);//Guardar a str
l'string que s'ha d'enviar a la controladora per tancar la connexió
			ClientsockRobot->Send(&str);//Enviar l'string
			Sleep(1000);//Esperar un temps perquè arribi la trama que
envia la controladora per finalitzar correctament la connexió
			ConnRobot = false;	//Indicar que el robot NO està
connectat via socket
			ClientsockRobot->Close(); //Tancar el socket
			delete ClientsockRobot;
			m_CListMsg.InsertString(0,"Connexio tancada amb
MAXILO");//Missatge en el box
			//FI TANCAMENT DE CONNEXIÓ AMB EL ROBOT
		}

		//Si el control per teclat està desactivat TANCAMENT DE CONNEXIÓ
AMB EL POLHEMUS
		if(!mestreTeclat && ConnPolhemus){
			ConnPolhemus = false;	//Indicar que Polhemus NO està
connectat via socket
			ClientsockPolhemus->Close(); //Tancar el socket del Polhemus
			delete ClientsockPolhemus;
			m_CListMsg.InsertString(0,"Connexio tancada amb
Polhemus");//Missatge en el box
		}

		//Habilitar els edit controls per canviar les dimensions de l'espai
restringit i el marge
		m_dimCub.EnableWindow(true);
		m_marge.EnableWindow(true);
		//Habilitar els checkboxes per canviar de dispositiu mestre
		CButton* pCheck = (CButton*)GetDlgItem(IDC_mestrePolhemus);
		pCheck->EnableWindow(true);

		//Si el mestre és el Polhemus marquem el checkbox
		if(mestrePolhemus){
			pCheck->SetCheck(1);
		}

		pCheck = (CButton*)GetDlgItem(IDC_mestreTeclat);
		pCheck->EnableWindow(true);

		//Si el mestre és el teclat marquem el checkbox
		if(mestreTeclat){
			pCheck->SetCheck(1);
```

```
			}
			UpdateData(true);

			m_CButton_Socket.SetWindowText("Connectar"); //Canviar el text del
botó indicant que es per connectar el Polhemus i el robot
}

//Mètode que es crida automàticament quan la connexió amb un socket ha estat
creada correctament
void CVirtualRobotDlg::OnConnect(int id, int nErrorCode)
{
	//Si el socket connectat és el del Polhemus
	if (id == POLHEMUS)
	{
			if(nErrorCode==0)//Si no hi han hagut errors
			{
				m_CListMsg.InsertString(0,"Connectat al Polhemus");
				ConnPolhemus = true; //Indicar que el Polhemus està connectat
via socket
				m_CButton_Socket.SetWindowText("Desconnectar"); //Canviar el
text del boto per connectar/desconnectar el Polhemus i el robot (s'hi està
connectat)

				//CREAR SOCKET PER EL ROBOT (controladora)
				ClientsockRobot = new COurSocket(this, RMAXILO); //Crear un
socket nou amb l'id del robot
				int error=ClientsockRobot-
>CreateAndConnect(PORT_ROBOT_MAXILO,_T(IP_ROBOT_MAXILO));//Conectar amb el
RobotControllerEmulator?¿
				m_CListMsg.InsertString(0,"Establint connexió amb
Maxilo....");//Informació en el box de missatges
				//FI CREAR SOCKET ROBOT (controladora)
			}
			else //Si hi ha hagut errors
			{
				m_CListMsg.InsertString(0,"No s'ha pogut establir la conexió
amb el Polhemus.");//Missatge al box
				TancaConnexions();
			}
	}
	//Si el socket creat és el del robot (controladora)
	else if (id == RMAXILO)
	{
			if(nErrorCode==0)//Si no hi han hagut errors
			{
				m_CListMsg.InsertString(0,"Connectat a MAXILO"); //Missatge
al box
				ConnRobot = true; //Indicar que el robot (controladora) està
connectat via socket

				//Inici de la cadena de moviments: S'envia el Tool al robot
				//Configurant un tool al robot s'aconsegueix que, quan se li
passa una posició on ha d'anar, l'element que hi ha d'arribar serà aquesta eina
que serà l'element terminal(enlloc del punt on s'ancla l'element terminal)
				//En el cas interesarà configurar el tool que correspongui al
tipus d'instrument laparoscòpic utilitzat
				HomogTransf Tool =  RPQScenario->L_PQPRob[RMAXILO]-
>GetTool();
				EulerZYiZii ToolRotation;
				ToolRotation = EulerZYiZii(Tool.getRotationMatrix());
```

```
                    CString str;

                    //Guardar l'string que s'ha d'enviar a la controladora per
enviar el tool
                    //Atencio amb els -65.0 en EixZ!!!
                    str = Interpret::tool(Tool[0][3],Tool[1][3],Tool[2][3]-
65.0,ToolRotation.getAlfa()*180/PI,ToolRotation.getBeta()*180/PI,ToolRotation.g
etGamma()*180/PI,TOOLOP);
                    ClientsockRobot->Send(&str); //Enviar el tool a la
controladora per configurarlo


      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////
                    //////////////INICIALITZACIÓ VARIABLES PER ELS CONTROLS
(GENERALS I DE SEGURETAT) IMPLEMENTATS//////////////

                    //Variables de configuració i control
                    //Control de l'home  mort
                    chmAct=false; //Control de l'home mort desactivat
                    chmCont=0; //Contador intern a 0 (no s'ha activat el control
encara)
                    chmSvm=0; //Mòdul de la velocitat de la lectura anterior on
s'ha activat el control
                    chmK=0; //Factor per el descens no lineal de la velocitat
                    chmD=Vec_3D(0,0,0); //Direcció de desplaçament utilitzada en
el descens no lineal de velocitat del control de l'home mort(x,y,z)
                    chmNum=0; //Guardar el nombre de mostres de dades del
dispositiu mestre consecutives que han superat el control de l'home mort fins
el moment

                    //////////////FI INICIALITZACIÓ VARIABLES PER ELS CONTROLS
(GENERALS I DE SEGURETAT) IMPLEMENTATS//////////////

      ////////////////////////////////////////////////////////////////////////
////////////////////////////////////

                    //Posar la variable resetmestre a cert per indicar que la
primera dada que es rebi s'utilitzarà com a posició inicial absoluta
                    resetmestre=true;
                    //Posem la variable resetrobot a cert per indicar que la
primera dada que es rebi del robot s'ha d'utilitzar com a posició inicial
absoluta
                    resetrobot=true;

                    //Activar el timer que demana dades al Polhemus i pinta el
robot virtual
                    SetTimer(DATO_POLHEMUS,TEMPSTIMER,0);

             }
             else //Si hi ha hagut errors
             {
                    m_CListMsg.InsertString(0,"No s'ha pogut establir la conexió
amb la controladora del robot");//Missatge al box
                    TancaConnexions();
             }
      }
      UpdateData(false);
}
```

```
void CVirtualRobotDlg::InitScenario()
{
	//Position of the robots and objects in the workspace  (x,y,z,rx,ry,rz)
	//Robots
	for (int indxRob = 0; indxRob < RPQScenario->GetNumRob(); indxRob++)
	{
		RPQScenario->L_PQPRob[indxRob]->SetHomogMatrixXYZ(0 , 0.0  , 0.0 ,
0.0 , 0.0 , 0.0 , 0.0);


		Vec_DP Joints(6);
		Joints[0] = 0.0;   Joints[1] = -68.0*PI/180.0; Joints[2] =
170.0*PI/180.0;
		Joints[3] = 0.0;   Joints[4] = 55.0*PI/180;    Joints[5] = 0.0;

		//RPQScenario->L_PQPRob[indxRob]->SetJoints(Joints);

		RPQScenario->L_PQPRob[indxRob]->SetToolXYZ(0,0,0,0,0,0);
		//RPQScenario->L_PQPRob[indxRob]-
>SetToolXYZ(TOOLX,TOOLY,TOOLZ,0,0,0);


		RPQScenario->L_PQPRob[indxRob]->GenRepLev1();


	}
	//Objects
	for (int indxObj = 0; indxObj < RPQScenario->GetNumPQPObj(); indxObj++)
	{
		RPQScenario->L_PQPObj[indxObj]->SetHomogMatrixXYZ(0 , 240.0 ,0.0 ,
95.0 , 0.0 , PI/2 ,0.0 );
		RPQScenario->L_PQPObj[indxObj]->GenRepLev1();
	}

	//Setting maximum increment of each axis
	for (int indxRob = 0; indxRob < RPQScenario->GetNumRob(); indxRob++)
	{
		for (int indxJoint = 0; indxJoint < RPQScenario->L_PQPRob[indxRob]-
>GetNumSubObj()-1; indxJoint++)
		{
			RPQScenario->L_PQPRob[indxRob]->SetMaxInc(indxJoint,10);
		}
	}

	//Fixar la configuració inicial dels robots
	Vec_DP SecPOSE = Vec_DP(6);
	SecPOSE[0] = 0.0;
	SecPOSE[1] = 0.0;
	SecPOSE[2] = 0.0;
	SecPOSE[3] = 0.0;
	SecPOSE[4] = 0.0;
	SecPOSE[5] = 0.0;

	for (int indxRob = 0; indxRob < RPQScenario->GetNumRob(); indxRob++)
	{
		RPQScenario->L_PQPRob[indxRob]->SetJoints(SecPOSE);   //Sets the
joints of the Robot0
	}

	//Define Grasp POSE for the object
```

```
	//Object: 80x80 box
//	Grasp.SetGraspHT(HomogTransf (RotationMatrix(EulerXYZ(0.0,	0.0 ,
0.0)), Vec_3D(	0.0, 40.0, 0.0 ))); //0
}



void CVirtualRobotDlg::OpenGLInit()
{
	//PIXELFORMATDESCRIPTOR pfd;
	int		n;
	HGLRC		hrc;

	m_pDC = new CClientDC(GetDlgItem(IDC_VIEW));

	ASSERT(m_pDC != NULL);

	PIXELFORMATDESCRIPTOR pfd = {
		sizeof(PIXELFORMATDESCRIPTOR),	// size of this pfd
		1,				// version number
		PFD_DRAW_TO_WINDOW |		// support window
		  PFD_SUPPORT_OPENGL |		// support OpenGL
		  PFD_DOUBLEBUFFER,		// double buffered
		PFD_TYPE_RGBA,			// RGBA type
		24,				// 24-bit color depth
		0, 0, 0, 0, 0, 0,		// color bits ignored
		0,				// no alpha buffer
		0,				// shift bit ignored
		0,				// no accumulation buffer
		0, 0, 0, 0,			// accum bits ignored
		16,				// 16-bit z-buffer
		0,				// no stencil buffer
		0,				// no auxiliary buffer
		PFD_MAIN_PLANE,			// main layer
		0,				// reserved
		0, 0, 0				// layer masks ignored
	};


	int pixelformat;
	if ( (pixelformat = ChoosePixelFormat(m_pDC->GetSafeHdc(), &pfd)) == 0 ){
		MessageBox("ChoosePixelFormat failed");
	}
	if (SetPixelFormat(m_pDC->GetSafeHdc(), pixelformat, &pfd) == FALSE){
		MessageBox("SetPixelFormat failed");
	}
	CRect m_oldRect;
	CWnd * pWnd = GetDlgItem(IDC_VIEW);
	pWnd->GetClientRect(&m_oldRect);

	n = ::GetPixelFormat(m_pDC->GetSafeHdc());
	::DescribePixelFormat(m_pDC->GetSafeHdc(), n, sizeof(pfd), &pfd);

	hrc = wglCreateContext(m_pDC->GetSafeHdc());

	wglMakeCurrent(m_pDC->GetSafeHdc(), hrc);

	glViewport(0, 0, m_oldRect.right, m_oldRect.bottom);

	glClearDepth(1.0);
```

```
	//Hidden surface removal
	glEnable(GL_DEPTH_TEST);
	glDepthFunc(GL_LEQUAL);

	//Turn on antialiasing and best visual representation
	glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
	glEnable(GL_BLEND);
	glEnable(GL_LINE_SMOOTH);
	glHint(GL_LINE_SMOOTH_HINT,GL_NICEST);
	glEnable(GL_POLYGON_SMOOTH);
	glHint(GL_POLYGON_SMOOTH_HINT,GL_NICEST);
	////////////////////////////////////////////////////

	glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
	glShadeModel(GL_SMOOTH);	// Definir el model d'ombres per els
objectes
	glPolygonMode(GL_FRONT, GL_FILL);

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(10,((double)m_oldRect.right/m_oldRect.bottom)
,1.0f,100000.0f);

	//LIGTHS ////////////////////////////////////////////////////
	GLfloat ambientlight[]  =   {0.5f ,0.5f ,0.5f ,1.0f};
	GLfloat difuselight[]   =   {1.0f ,1.0f ,1.0f ,1.0f};
	GLfloat specularlight[] =   {1.0f ,1.0f ,1.0f ,1.0f};
	GLfloat lightposition[] =   {0.0f,100.0f,100.0f,1.0f};
	glLightfv(GL_LIGHT0,GL_AMBIENT,ambientlight);
	glLightfv(GL_LIGHT0,GL_DIFFUSE,difuselight);
	glLightfv(GL_LIGHT0,GL_SPECULAR,specularlight);
	glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
	glEnable(GL_LIGHTING);			// Activar l'iluminació
	glEnable(GL_LIGHT0);			// Activar el llum definit
anteriorment

	OpenGLInitMaterials();
	OpenGLInitPOSEView();
	//OpenGLInitNormals();

}

void CVirtualRobotDlg::OpenGLInitMaterials()
{
	cromoShi = 32.0;
	cromoAmb[0]=0.4;  cromoAmb[1]=0.4;  cromoAmb[2]=0.4;  cromoAmb[3]=1.0;
	cromoDif[0]=0.4;  cromoDif[1]=0.4;  cromoDif[2]=0.4;  cromoDif[3]=1.0;
	cromoSpe[0]=0.35; cromoSpe[1]=0.35; cromoSpe[2]=0.35; cromoSpe[3]=1.0;

	acromoAmb[0]=0.4; acromoAmb[1]=0.4; acromoAmb[2]=0.4; acromoAmb[3]=0.4;
	acromoDif[0]=0.4; acromoDif[1]=0.4; acromoDif[2]=0.4; acromoDif[3]=0.4;
	acromoSpe[0]=0.35;	acromoSpe[1]=0.35;	acromoSpe[2]=0.35;
	acromoSpe[3]=0.4;

	cromoRedAmb[0]=0.9;	cromoRedAmb[1]=0.2;	cromoRedAmb[2]=0.2;
	cromoRedAmb[3]=0.9;
	cromoRedDif[0]=0.9;	cromoRedDif[1]=0.2;	cromoRedDif[2]=0.2;
	cromoRedDif[3]=0.9;
	cromoRedSpe[0]=0.9; cromoRedSpe[1]=0.2; cromoRedSpe[2]=0.2;
cromoRedSpe[3]=0.9;
```

```
	cromoGreenAmb[0]=0.2;	cromoGreenAmb[1]=0.9;	cromoGreenAmb[2]=0.2;
	cromoGreenAmb[3]=0.9;
	cromoGreenDif[0]=0.2;	cromoGreenDif[1]=0.9;	cromoGreenDif[2]=0.2;
	cromoGreenDif[3]=0.9;
	cromoGreenSpe[0]=0.2;	cromoGreenSpe[1]=0.9;	cromoGreenSpe[2]=0.2;
cromoGreenSpe[3]=0.9;

	cromoBlueAmb[0]=0.2;	cromoBlueAmb[1]=0.2;	cromoBlueAmb[2]=0.9;
	cromoBlueAmb[3]=0.9;
	cromoBlueDif[0]=0.2;	cromoBlueDif[1]=0.2;	cromoBlueDif[2]=0.9;
	cromoBlueDif[3]=0.9;
	cromoBlueSpe[0]=0.2;	cromoBlueSpe[1]=0.2;	cromoBlueSpe[2]=0.9;
cromoBlueSpe[3]=0.9;

	mat_spec[0]=0.90; mat_spec[1]=10;		mat_spec[2]=0.10;
	mat_spec[3]=1.0;
	mat_blanc[0]=1.0; mat_blanc[1]=1.0; mat_blanc[2]=1.0; mat_blanc[3]=1.0;
	mat_vermell[0]=1.0;	mat_vermell[1]=0.0;	mat_vermell[2]=0.0;
	mat_vermell[3]=1.0;
	mat_verd[0]=0.0;  mat_verd[1]=1.0;  mat_verd[2]=0.0;  mat_verd[3]=1.0;

}

void CVirtualRobotDlg::OpenGLInitPOSEView()
{
	OpenGL_fita=(double)PI/3;
	OpenGL_psi=(double)PI/9;
	OpenGL_R_CAMARA = 10000.0;
	OpenGL_Pos_x=OpenGL_R_CAMARA*sin(OpenGL_fita)*cos(OpenGL_psi);
	OpenGL_Pos_y=OpenGL_R_CAMARA*cos(OpenGL_fita)*cos(OpenGL_psi);
	OpenGL_Pos_z=OpenGL_R_CAMARA*sin(OpenGL_psi);
	OpenGL_Cen_x=0.0;
	OpenGL_Cen_y=0.5;
	OpenGL_Cen_z=0.0;
}

void CVirtualRobotDlg::OpenGLDrawAxis(int model)
{
	GLUquadricObj *cylinder[7];

	double baseRad1;
	double topRad1;
	double heigh1;
	double alcada;
	double baseRad2;
	double topRad2;
	double heigh2;

	if (model == 1)
	{
		//Paràmetres per dibuixar eix Z,Y i X del mon
		baseRad1 = 1; topRad1 = 1; heigh1 = 100;
		alcada = 70;
		baseRad2 = 5; topRad2 = 0; heigh2 = 50;
	}
	else
	{
		//Paràmetres per dibuixar eix Z,Y i X de cada una de les
articulacions
```

```
		baseRad1 = 2; topRad1 = 1; heigh1 = 80;
		alcada = 75;
		baseRad2 = 10;topRad2 = 0; heigh2 = 20;
	}

		//Eix: Z
		glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_blanc);
		glMaterialfv(GL_FRONT, GL_AMBIENT, mat_blanc);
		glMaterialfv(GL_FRONT, GL_SHININESS ,mat_blanc);
		glMaterialfv(GL_FRONT, GL_SPECULAR ,mat_blanc);
		cylinder[0] = gluNewQuadric();
		cylinder[1] = gluNewQuadric();
		gluQuadricOrientation(cylinder[0],GLU_OUTSIDE);
		gluCylinder(cylinder[0],baseRad1,topRad1, heigh1,10,1);
		glTranslated(0,0,alcada);
			gluCylinder(cylinder[1], baseRad2,topRad2, heigh2, 10, 1);
		glTranslated(0,0,-alcada);
		//Eix: Y
		glRotated(90,-1,0,0);
			glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_spec);
			glMaterialfv(GL_FRONT, GL_AMBIENT, mat_spec);
			glMaterialfv(GL_FRONT, GL_SHININESS ,mat_spec);
			glMaterialfv(GL_FRONT, GL_SPECULAR ,mat_spec);
			gluCylinder(cylinder[0],baseRad1,topRad1, heigh1,10,1);
			glTranslated(0,0,alcada);
				gluCylinder(cylinder[1], baseRad2,topRad2, heigh2, 10,
1);
			glTranslated(0,0,-alcada);
		glRotated(90,1,0,0);
		//Eix: X
		glRotated(90,0,1,0);
			glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_vermell);
			glMaterialfv(GL_FRONT, GL_AMBIENT, mat_vermell);
			glMaterialfv(GL_FRONT, GL_SHININESS ,mat_vermell);
			glMaterialfv(GL_FRONT, GL_SPECULAR ,mat_vermell);
			gluCylinder(cylinder[0],baseRad1,topRad1, heigh1,10,1);
			glTranslated(0,0,alcada);
				gluCylinder(cylinder[1], baseRad2,topRad2, heigh2, 10,
1);
			glTranslated(0,0,-alcada);
		glRotated(90,0,-1,0);
}

//Mètode per pintar el tool
void CVirtualRobotDlg::OpenGLDrawTool(void)
{
	GLUquadricObj *cylinder[7];

	double baseRad1;
	double topRad1;
	double heigh1;
	double alcada;
	double baseRad2;
	double topRad2;
	double heigh2;

	//Paràmetres per dibuixar eix Z,Y i X de cada una de les articulacions
	baseRad1 = 4; topRad1 = 4;
	alcada = TOOLZ;
	//baseRad2 = 10;topRad2 = 0; heigh2 = 20;
```

```
	//Eix: Z
	glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_blanc);
	glMaterialfv(GL_FRONT, GL_AMBIENT, mat_blanc);
	glMaterialfv(GL_FRONT, GL_SHININESS ,mat_blanc);
	glMaterialfv(GL_FRONT, GL_SPECULAR ,mat_blanc);
	cylinder[0] = gluNewQuadric();
	cylinder[1] = gluNewQuadric();
	gluQuadricOrientation(cylinder[0],GLU_OUTSIDE);
	glTranslated(0,0,-alcada);
	gluCylinder(cylinder[0],baseRad1,topRad1, alcada,10,1);
}


void CVirtualRobotDlg::OpenGLInitDspList()
{
	int i;
	int indx;
	dsplist = new int**[RPQScenario->GetNumElements()];
	dspedgeslist = new int**[RPQScenario->GetNumElements()];

	GLdouble dNormalX;
	GLdouble dNormalY;
	GLdouble dNormalZ;

	//1: Declaració de les dsplists per els robots
	//Surface lists
	for (indx = 0; indx < RPQScenario->GetNumRob(); indx++)
	{
		dsplist[indx] = new int *[3];
		dsplist[indx][0] = new int [1];
		dsplist[indx][1] = new int [RPQScenario->L_PQPRob[indx]-
>GetNumSubObj()];
		dsplist[indx][2] = new int [RPQScenario->L_PQPRob[indx]-
>GetNumSubObj()];
	}
	//Edges list
	for (indx = 0; indx < RPQScenario->GetNumRob(); indx++)
	{
		dspedgeslist[indx] = new int *[3];
		dspedgeslist[indx][0] = new int [1];
		dspedgeslist[indx][1] = new int [RPQScenario->L_PQPRob[indx]-
>GetNumSubObj()];
		dspedgeslist[indx][2] = new int [RPQScenario->L_PQPRob[indx]-
>GetNumSubObj()];
	}
	//2: Declaració de les dsplists per els objectes
	//Surface lists
	for (indx = RPQScenario->GetNumRob(); indx < RPQScenario-
>GetNumElements(); indx++)
	{
		dsplist[indx] = new int *[3];
		dsplist[indx][0] = new int [1];
		dsplist[indx][1] = new int [RPQScenario->L_PQPObj[indx-RPQScenario-
>GetNumRob()]->GetNumSubObj()];
		dsplist[indx][2] = new int [RPQScenario->L_PQPObj[indx-RPQScenario-
>GetNumRob()]->GetNumSubObj()];
	}
	//Edges list
	for (indx = RPQScenario->GetNumRob(); indx < RPQScenario-
>GetNumElements(); indx++)
```

```
	{
		dspedgeslist[indx] = new int *[3];
		dspedgeslist[indx][0] = new int [1];
		dspedgeslist[indx][1] = new int [RPQScenario->L_PQPObj[indx-
RPQScenario->GetNumRob()]->GetNumSubObj()];
		dspedgeslist[indx][2] = new int [RPQScenario->L_PQPObj[indx-
RPQScenario->GetNumRob()]->GetNumSubObj()];
	}

	//Generar les dsplists per els robots
	int idObj,idSubObj;
	for (idObj = 0; idObj < RPQScenario->GetNumRob() ; idObj++)
	{
		//1rst rep level - Surface list
		dsplist[idObj][0][0] = glGenLists(1);
		glNewList(dsplist[idObj][0][0],GL_COMPILE);
		//Generates 1rst rep level:
		RPQScenario->L_PQPRob[idObj]->GenRepLev1();
		for (i = 0; i < 12; i++)
		{
			glBegin(GL_LINE_STRIP);
			glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[0][0].tris[i].p1);
			glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[0][0].tris[i].p2);
			glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[0][0].tris[i].p3);
			glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[0][0].tris[i].p1);
			glEnd();
		}
		glEndList();
		//1rst rep level - Edges list
		dspedgeslist[idObj][0][0] = glGenLists(1);
		glNewList(dspedgeslist[idObj][0][0],GL_COMPILE);
		for(i=0; i<12; i++)
		{
			glBegin(GL_LINE_STRIP);
			glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[0][0].tris[i].p1);
			glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[0][0].tris[i].p2);
			glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[0][0].tris[i].p3);
			glEnd();
		}
		glEndList();

		for (idSubObj = 0; idSubObj < RPQScenario->L_PQPRob[idObj]-
>GetNumSubObj(); idSubObj++)
		{
			//2nd rep level - Surface list
			dsplist[idObj][1][idSubObj] = glGenLists(1);
			glNewList(dsplist[idObj][1][idSubObj],GL_COMPILE);
			for (i = 0; i < 12; i++)
			{
				glBegin(GL_LINE_STRIP);
				glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[1][idSubObj].tris[i].p1);
```

```
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[1][idSubObj].tris[i].p2);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[1][idSubObj].tris[i].p3);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[1][idSubObj].tris[i].p1);
                        glEnd();
                  }
                  glEndList();

                  //2nd rep level - Edges list
                  dspedgeslist[idObj][1][idSubObj] = glGenLists(1);
                  glNewList(dspedgeslist[idObj][1][idSubObj],GL_COMPILE);
                  for (i = 0; i < 12; i++)
                  {
                        glBegin(GL_LINE_STRIP);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[1][idSubObj].tris[i].p1);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[1][idSubObj].tris[i].p2);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[1][idSubObj].tris[i].p3);
                        glEnd();
                  }
                  glEndList();

                  //3rd rep level - Surface list
                  dsplist[idObj][2][idSubObj] = glGenLists(1);
                  glNewList(dsplist[idObj][2][idSubObj],GL_COMPILE);
                  glBegin(GL_TRIANGLES);
                  for (i = 0; i < RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].num_tris; i++)
                  {
                        // Calculate the vector normal coming out of the 3D
polygon.
                        OpenGLCalcNormal(RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p3,
                                                RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p2,
                                                RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p1,
                                                &dNormalX,&dNormalY,
&dNormalZ);
                        // Set the normal vector for the polygon
                        glNormal3d(dNormalX, dNormalY, dNormalZ);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p1);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p2);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p3);
                  }
                  glEnd();
                  glEndList();

                  //3rd rep level - Edges list
                  dspedgeslist[idObj][2][idSubObj] = glGenLists(1);
                  glNewList(dspedgeslist[idObj][2][idSubObj],GL_COMPILE);
                  for (i = 0; i < RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].num_tris; i++)
```

```
                  {
                        glBegin(GL_LINE_STRIP);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p1);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p2);
                        glVertex3dv(RPQScenario->L_PQPRob[idObj]-
>TriModel[2][idSubObj].tris[i].p3);
                        glEnd();
                  }
                  glEndList();
            }
      }

      //Generar les dsplists per els objectes
      for (idObj=RPQScenario->GetNumRob(); idObj < RPQScenario-
>GetNumElements(); idObj++)
      {
            //1rst rep level - Surface list
            dsplist[idObj][0][0] = glGenLists(1);
            glNewList(dsplist[idObj][0][0],GL_COMPILE);
            //Generates 1rst rep level:
            RPQScenario->L_PQPObj[idObj-RPQScenario->GetNumRob()]-
>GenRepLev1();
            for (i = 0; i < 12; i++)
            {
                  glBegin(GL_LINE_STRIP);
                  glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[0][0].tris[i].p1);
                  glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[0][0].tris[i].p2);
                  glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[0][0].tris[i].p3);
                  glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[0][0].tris[i].p1);
                  glEnd();
            }
            glEndList();

            //1rst rep level - Edges list
            dspedgeslist[idObj][0][0] = glGenLists(1);
            glNewList(dspedgeslist[idObj][0][0],GL_COMPILE);
            for(i=0; i<12; i++)
            {
                  glBegin(GL_LINE_STRIP);
                  glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[0][0].tris[i].p1);
                  glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[0][0].tris[i].p2);
                  glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[0][0].tris[i].p3);
                  glEnd();
            }
            glEndList();

            for (idSubObj = 0; idSubObj < RPQScenario->L_PQPObj[idObj-
RPQScenario->GetNumRob()]->GetNumSubObj(); idSubObj++)
            {
                  //2nd rep level - Surface list
                  dsplist[idObj][1][idSubObj] = glGenLists(1);
```

```
                    glNewList(dsplist[idObj][1][idSubObj],GL_COMPILE);
                    for (i = 0; i < 12; i++)
                    {
                          glBegin(GL_LINE_STRIP);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[1][idSubObj].tris[i].p1);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[1][idSubObj].tris[i].p2);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[1][idSubObj].tris[i].p3);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[1][idSubObj].tris[i].p1);
                          glEnd();
                    }
                    glEndList();

                    //2nd rep level - Edges list
                    dspedgeslist[idObj][1][idSubObj] = glGenLists(1);
                    glNewList(dspedgeslist[idObj][1][idSubObj],GL_COMPILE);
                    for (i = 0; i < 12; i++)
                    {
                          glBegin(GL_LINE_STRIP);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[1][idSubObj].tris[i].p1);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[1][idSubObj].tris[i].p2);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[1][idSubObj].tris[i].p3);
                          glEnd();
                    }
                    glEndList();

                    //3rd Level - Surface list
                    dsplist[idObj][2][idSubObj] = glGenLists(1);
                    glNewList(dsplist[idObj][2][idSubObj],GL_COMPILE);
                    glBegin(GL_TRIANGLES);
                    for (i = 0; i < RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[2][idSubObj].num_tris; i++)
                    {
                          // Calculate the vector normal coming out of the 3D
polygon.
                          OpenGLCalcNormal(RPQScenario->L_PQPObj[idObj-
RPQScenario->GetNumRob()]->TriModel[2][idSubObj].tris[i].p3,
                                                RPQScenario->L_PQPObj[idObj-
RPQScenario->GetNumRob()]->TriModel[2][idSubObj].tris[i].p2,
                                                RPQScenario->L_PQPObj[idObj-
RPQScenario->GetNumRob()]->TriModel[2][idSubObj].tris[i].p1,
                                                &dNormalX,&dNormalY,
&dNormalZ);
                          // Set the normal vector for the polygon
                          glNormal3d(dNormalX, dNormalY, dNormalZ);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[2][idSubObj].tris[i].p1);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[2][idSubObj].tris[i].p2);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[2][idSubObj].tris[i].p3);
                    }
                    glEnd();
                    glEndList();
```

```
                    //3rd Level - Edges list
                    dspedgeslist[idObj][2][idSubObj] = glGenLists(1);
                    glNewList(dspedgeslist[idObj][2][idSubObj],GL_COMPILE);
                    for (i = 0; i < RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[2][idSubObj].num_tris; i++)
                    {
                          glBegin(GL_LINE_STRIP);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[2][idSubObj].tris[i].p1);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[2][idSubObj].tris[i].p2);
                          glVertex3dv(RPQScenario->L_PQPObj[idObj-RPQScenario-
>GetNumRob()]->TriModel[2][idSubObj].tris[i].p3);
                          glEnd();
                    }
                    glEndList();
              }
        }
}


void CVirtualRobotDlg::OpenGLDeleteDSPList()
{
      int indx;
      for (indx = 0; indx < RPQScenario->GetNumElements(); indx++)
      //numero obj creats
      {
            delete dsplist[indx][0];
            delete dsplist[indx][1];
            delete dsplist[indx][2];
            delete dsplist[indx];
      }
      delete dsplist;

      for (indx = 0; indx < RPQScenario->GetNumElements(); indx++)
      //numero obj creats
      {
            delete dspedgeslist[indx][0];
            delete dspedgeslist[indx][1];
            delete dspedgeslist[indx][2];
            delete dspedgeslist[indx];
      }
      delete dspedgeslist;
      dspedgeslist = NULL;

      //Activar quan es faci servir la funció OpenGLCalcNormal()
      /*for (indx = 0; indx < RPQScenario->GetNumElements(); indx++)
      //numero obj creats
      {
            delete normlist[indx][0];
            delete normlist[indx][1];
            delete normlist[indx][2];
            delete normlist[indx];
      }
      delete normlist;
      normlist = NULL;*/


}
```

```
void CVirtualRobotDlg::OpenGLInitNormals()
{
	GLdouble dNormalX;
	GLdouble dNormalY;
	GLdouble dNormalZ;
	int indx;
	int i;
	int idRob;
	int idObj;
	int idSubObj;

	/*normlist[NumElements][SubObj][Coord]
		[NumElements] = Number of Robots + number of Objects (1rst robots,
2nd objects)
		[SubObj] = Number of subObjects
		[Coord] = 3 coordinates of the normal vector: x,y,z
	*/
	normlist = new double**[RPQScenario->GetNumElements()];

	//1rst: Declaration of normlists for the Robots
	for (indx = 0; indx < RPQScenario->GetNumRob(); indx++)
	{

		normlist[indx] = new double *[RPQScenario->L_PQPRob[indx]-
>GetNumSubObj()];
		for (i = 0; i < RPQScenario->L_PQPRob[indx]->GetNumSubObj(); i++)
		{
			normlist[indx][i] = new double [3];
		}
	}
	//2nd: Declaration of normlists for the Objects
	for (indx = RPQScenario->GetNumRob(); indx < RPQScenario-
>GetNumElements(); indx++)
	{
		normlist[indx] = new double *[RPQScenario->L_PQPObj[indx-
RPQScenario->GetNumRob()]->GetNumSubObj()];
		for (i = 0; i < RPQScenario->L_PQPObj[indx-RPQScenario-
>GetNumRob()]->GetNumSubObj(); i++)
		{
			normlist[indx][i] = new double [3];
		}
	}

	//Robot Normals
	for (idRob = 0; idRob < RPQScenario->GetNumRob(); idRob++)
	{
		for (idSubObj = 0; idSubObj < RPQScenario->L_PQPRob[idRob]-
>GetNumSubObj(); idSubObj++)
		{
			//3rd rep level
			for (i = 0; i < RPQScenario->L_PQPRob[idRob]-
>TriModel[2][idSubObj].num_tris; i++)
			{
				glVertex3dv(RPQScenario->L_PQPRob[idRob]-
>TriModel[2][idSubObj].tris[i].p1);
				glVertex3dv(RPQScenario->L_PQPRob[idRob]-
>TriModel[2][idSubObj].tris[i].p2);
				glVertex3dv(RPQScenario->L_PQPRob[idRob]-
>TriModel[2][idSubObj].tris[i].p3);
```

```
                        // Calculate the vector normal coming out of the 3D
polygon.
                        OpenGLCalcNormal(RPQScenario->L_PQPRob[idRob]-
>TriModel[2][idSubObj].tris[i].p3,
                                         RPQScenario->L_PQPRob[idRob]-
>TriModel[2][idSubObj].tris[i].p2,
                                         RPQScenario->L_PQPRob[idRob]-
>TriModel[2][idSubObj].tris[i].p1,
                                         &dNormalX,&dNormalY, &dNormalZ);
                        // Store the normal vector for the polygon
                        normlist[idRob][idSubObj][0] = dNormalX;
                        normlist[idRob][idSubObj][1] = dNormalY;
                        normlist[idRob][idSubObj][2] = dNormalZ;
                  }
            }
      }

      //Object Normals
      for (idObj = 0; idObj < RPQScenario->GetNumPQPObj(); idObj++)
      {
            for (idSubObj = 0; idSubObj < RPQScenario->L_PQPRob[idObj]-
>GetNumSubObj(); idSubObj++)
            {
                  //3rd rep level
                  for (i = 0; i < RPQScenario->L_PQPRob[idRob]-
>TriModel[2][idSubObj].num_tris; i++)
                  {
                        glVertex3dv(RPQScenario->L_PQPObj[idObj]-
>TriModel[2][idSubObj].tris[i].p1);
                        glVertex3dv(RPQScenario->L_PQPObj[idObj]-
>TriModel[2][idSubObj].tris[i].p2);
                        glVertex3dv(RPQScenario->L_PQPObj[idObj]-
>TriModel[2][idSubObj].tris[i].p3);
                        // Calculate the vector normal coming out of the 3D
polygon.
                        OpenGLCalcNormal(RPQScenario->L_PQPObj[idObj]-
>TriModel[2][idSubObj].tris[i].p3,
                                         RPQScenario->L_PQPObj[idObj]-
>TriModel[2][idSubObj].tris[i].p2,
                                         RPQScenario->L_PQPObj[idObj]-
>TriModel[2][idSubObj].tris[i].p1,
                                         &dNormalX,&dNormalY, &dNormalZ);
                        // Store the normal vector for the polygon
                        normlist[RPQScenario->GetNumRob()+idObj][idSubObj][0] =
dNormalX;
                        normlist[RPQScenario->GetNumRob()+idObj][idSubObj][1] =
dNormalY;
                        normlist[RPQScenario->GetNumRob()+idObj][idSubObj][2] =
dNormalZ;
                  }
            }
      }
}

void CVirtualRobotDlg::OpenGLCalcNormal(GLdouble fVert1[], GLdouble
fVert2[],GLdouble fVert3[],
                                                GLdouble *fNormalX, GLdouble
*fNormalY, GLdouble *fNormalZ)
{
      GLdouble Qx, Qy, Qz, Px, Py, Pz;
```

```
	Qx = fVert2[0]-fVert1[0];
	Qy = fVert2[1]-fVert1[1];
	Qz = fVert2[2]-fVert1[2];
	Px = fVert3[0]-fVert1[0];
	Py = fVert3[1]-fVert1[1];
	Pz = fVert3[2]-fVert1[2];

	*fNormalX = Py*Qz - Pz*Qy;
	*fNormalY = Pz*Qx - Px*Qz;
	*fNormalZ = Px*Qy - Py*Qx;

	GLdouble len = (GLdouble)(sqrt((*fNormalX)*(*fNormalX) +
(*fNormalY)*(*fNormalY) + (*fNormalZ)*(*fNormalZ)));
   if (len)
   {
      *fNormalX = *fNormalX / len;
      *fNormalY = *fNormalY / len;
      *fNormalZ = *fNormalZ / len;
   }

}

void CVirtualRobotDlg::OpenGLDraw()
{
	// Clear the framebuffer and depth buffe
	glEnable(GL_DEPTH_TEST);
	glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT|GL_ACCUM_BUFFER_BIT);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(OpenGL_Pos_x, OpenGL_Pos_y, OpenGL_Pos_z,
			OpenGL_Cen_x, OpenGL_Cen_y, OpenGL_Cen_z,
			0.0f,0.0f,1.0f );
	double matrix[4][4];

	//INI: Dibuixa eix X, Y i Z del mon
	OpenGLDrawAxis(1);
	glMaterialfv(GL_FRONT, GL_AMBIENT, cromoAmb);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, cromoDif);
	glMaterialfv(GL_FRONT, GL_SPECULAR, cromoSpe);
	glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);

	glPushMatrix();

	int idRepLev;
	int idSubObj;
	int parell = 0;

	//Pintar els robots
////////////////////////////////////////////////////////////////
	for (int idObj = 0; idObj < RPQScenario->GetNumRob() ; idObj++)
	{
		glPushMatrix();
		for (idRepLev = 2; idRepLev < 3; idRepLev++)
		{
			glPushMatrix();
			if (RobSel != idObj)
			{
				glMaterialfv(GL_FRONT, GL_AMBIENT, acromoAmb);
```

```
                        glMaterialfv(GL_FRONT, GL_DIFFUSE, acromoDif);
                        glMaterialfv(GL_FRONT, GL_SPECULAR, acromoSpe);
                        glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);
                  }
                  for (idSubObj = 0; idSubObj < RPQScenario->L_PQPRob[idObj]-
>GetNumSubObj(); idSubObj++)
                  {
                        RPQScenario->L_PQPRob[idObj]-
>GetTransposedHomogMatrix(idSubObj,matrix);
                        glMultMatrixd(&matrix[0][0]);
                        OpenGLDrawAxis(0);
                        glMaterialfv(GL_FRONT, GL_AMBIENT, cromoAmb);
                        glMaterialfv(GL_FRONT, GL_DIFFUSE, cromoDif);
                        glMaterialfv(GL_FRONT, GL_SPECULAR, cromoSpe);
                        glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);
                        glCallList(dsplist[idObj][idRepLev][idSubObj]);
                  }
                  //Per pintar el tool (no fa les rotacions de l'articulació
passiva)
                  /*RPQScenario->L_PQPRob[idObj]-
>GetTransposedHomogMatrix(idSubObj,matrix);
                  glMultMatrixd(&matrix[0][0]);
                  OpenGLDrawTool();*/

                  glMaterialfv(GL_FRONT, GL_AMBIENT, cromoAmb);
                  glMaterialfv(GL_FRONT, GL_DIFFUSE, cromoDif);
                  glMaterialfv(GL_FRONT, GL_SPECULAR, cromoSpe);
                  glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);
                  glPopMatrix();
            }
            glPopMatrix();
      }


      //Pintar els objectes
////////////////////////////////////////////////////////////
      for (int ii = 0; ii < RPQScenario->GetNumPQPObj();ii++)
      {
            glPushMatrix();
            for (idRepLev = 2; idRepLev < 3; idRepLev++)
            {
                  glPushMatrix();
                  for (idSubObj = 0; idSubObj < RPQScenario->L_PQPObj[ii]-
>GetNumSubObj(); idSubObj++)
                  {
                        RPQScenario->L_PQPObj[ii]-
>GetTransposedHomogMatrix(idSubObj,matrix);
                        glMultMatrixd(&matrix[0][0]);
                        if (ii == 0)      //Real Object
                        {
                              glMaterialfv(GL_FRONT, GL_AMBIENT, cromoRedAmb);
                              glMaterialfv(GL_FRONT, GL_DIFFUSE, cromoRedDif);
                              glMaterialfv(GL_FRONT, GL_SPECULAR, cromoRedSpe);
                              glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);
                        }
                        else
                        {
                              glMaterialfv(GL_FRONT, GL_AMBIENT,
cromoGreenAmb);
```

```
                                    glMaterialfv(GL_FRONT, GL_DIFFUSE,
cromoGreenDif);
                                    glMaterialfv(GL_FRONT, GL_SPECULAR,
cromoGreenSpe);
                                    glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);
                              }
                              //No hi ha objectes a l'escenari
                              //glCallList(dsplist[ii+RPQScenario-
>GetNumRob()][idRepLev][idSubObj]);
                              OpenGLDrawAxis(0);
                              glMaterialfv(GL_FRONT, GL_AMBIENT, cromoAmb);
                              glMaterialfv(GL_FRONT, GL_DIFFUSE, cromoDif);
                              glMaterialfv(GL_FRONT, GL_SPECULAR, cromoSpe);
                              glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);

                        }
                        glPopMatrix();
                  }
                  glPopMatrix();
            }
            glPopMatrix();

            //Si s'han incialitzat les posicions de l'orifici i l'instrument es
pinten
            if(initOriInst){

                  //Pintar la posició retornada per el robot
                  /*glPushMatrix();
                  glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_spec);
                  glMaterialfv(GL_FRONT, GL_AMBIENT, mat_spec);
                  glMaterialfv(GL_FRONT, GL_SHININESS ,mat_spec);
                  glMaterialfv(GL_FRONT, GL_SPECULAR ,mat_spec);
                  glTranslated(posMaxilo[0],posMaxilo[1],posMaxilo[2]);
                  glutSolidSphere(10,10,10);
                  glPopMatrix();*/

                  //Pintar l'instrument
                  GLUquadricObj *cylinder;
                  glPushMatrix();
                  glMaterialfv(GL_FRONT, GL_AMBIENT, cromoRedAmb);
                  glMaterialfv(GL_FRONT, GL_DIFFUSE, cromoRedDif);
                  glMaterialfv(GL_FRONT, GL_SPECULAR, cromoRedSpe);
                  glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);
                  cylinder = gluNewQuadric();

                  //Calcul de les rotacions que s'han d'aplicar a l'instrument per
cada eix
                  //Calcular el vector direcció entre el punt que marca el sensor
(punt de treball de l'instrument laparoscòpic) i l'orifici d'entrada dels
instruments
                  Vec_3D dir= orifici-instrumentPos;

                  //Calcular el centre de la posició de l'instrument
                  float modul=sqrt(pow(dir[0],2)+pow(dir[1],2)+pow(dir[2],2));
                  Vec_3D dirU=dir/modul;
                  Vec_3D
transPos=instrumentPos+dirU*(llargadaLaparoscopicInstrument/2);
                  glTranslated(transPos[0],transPos[1],transPos[2]);
```

```
            //Amb la fórmula del producte escalar calcular l'angle entre el
vector (x,z)=(0,-1) i el vector (x,z)=(dir[0],dir[2]) del qual s'esbrinarà la
rotació que s'ha de fer de l'eix y
            float roty=acos(-dir[2]/sqrt(pow(dir[0],2)+pow(dir[2],2)));
            roty= roty*180/PI; //Passar a graus
            if(dir[0]>0) roty=roty*(-1);

            //Amb la fórmula del producte escalar calcular l'angle entre el
vector (y,z)=(0,1) i el vector (y,z)=(dir[1],dir[2]) del qual s'esbrinarà la
rotació que s'ha de fer de l'eix x
            float rotx=acos(dir[2]/sqrt(pow(dir[1],2)+pow(dir[2],2)));
            rotx= rotx*180/PI; //Passar a graus
            if(dir[1]>0) rotx=rotx*(-1);

            //Rotar l'eix x
            glRotated(rotx,1,0,0);
            //Rotar l'eix y
            glRotated(roty,0,1,0);

            //Centrar el cilindre que representa l'instrument a l'origen de
coordenades
            glTranslated(0,0,-llargadaLaparoscopicInstrument/2);

            gluCylinder(cylinder,4,4,llargadaLaparoscopicInstrument,10,1);
            glPopMatrix();
            //Fi pintar instrument

            //Pintar orifici per on entren els instruments
            glPushMatrix();
            glMaterialfv(GL_FRONT, GL_AMBIENT, cromoGreenAmb);
            glMaterialfv(GL_FRONT, GL_DIFFUSE, cromoGreenDif);
            glMaterialfv(GL_FRONT, GL_SPECULAR, cromoGreenSpe);
            glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);
            glTranslated(orifici[0],orifici[1],orifici[2]);
            glutSolidSphere(10,10,10);

            //GLUquadricObj *cylinder;
            cylinder = gluNewQuadric();
            //gluCylinder(cylinder,10,10,10,10,1);

            glPopMatrix();

            //Pintar la posició de l'extrem de l'instrument
            glPushMatrix();
            glMaterialfv(GL_FRONT, GL_AMBIENT, cromoRedAmb);
            glMaterialfv(GL_FRONT, GL_DIFFUSE, cromoRedDif);
            glMaterialfv(GL_FRONT, GL_SPECULAR, cromoRedSpe);
            glMaterialf(GL_FRONT, GL_SHININESS, cromoShi);
            glTranslated(instrumentPos[0],instrumentPos[1],instrumentPos[2]);
            //glutSolidSphere(7,10,10);

            //Rotar l'eix x
            glRotated(rotx,1,0,0);
            //Rotar l'eix y
            glRotated(roty,0,1,0);
            cylinder = gluNewQuadric();
            gluCylinder(cylinder,8,1,10,10,1);

            glPopMatrix();
```

```
            //Pintar el cub que representa l'espai de treball restringit
            glPushMatrix();

            //Definir el color el cub
            GLfloat blauTransparentDif[4];
            GLfloat blauTransparentAmb[4];
            GLfloat blauTransparentSpe[4];
            blauTransparentDif[0]=cromoBlueDif[0];
            blauTransparentDif[1]=cromoBlueDif[1];
            blauTransparentDif[2]=cromoBlueDif[2];
            blauTransparentDif[3]=cromoBlueDif[3]-0.65;
            blauTransparentAmb[0]=cromoBlueAmb[0];
            blauTransparentAmb[1]=cromoBlueAmb[1];
            blauTransparentAmb[2]=cromoBlueAmb[2];
            blauTransparentAmb[3]=cromoBlueAmb[3]-0.65;
            blauTransparentSpe[0]=cromoBlueSpe[0];
            blauTransparentSpe[1]=cromoBlueSpe[1];
            blauTransparentSpe[2]=cromoBlueSpe[2];
            blauTransparentSpe[3]=cromoBlueSpe[3]-0.65;

            glMaterialfv(GL_FRONT, GL_DIFFUSE, blauTransparentAmb);
            glMaterialfv(GL_FRONT, GL_AMBIENT, blauTransparentDif);
            glMaterialf(GL_FRONT, GL_SHININESS ,cromoShi);
            glMaterialfv(GL_FRONT, GL_SPECULAR ,blauTransparentSpe);

      glTranslated(centreEspaiTreballRestringit[0],centreEspaiTreballRestringit
[1],centreEspaiTreballRestringit[2]);
            glutSolidCube(dimCub);
            glPopMatrix();
      }
      SwapBuffers(wglGetCurrentDC()); // imatge traspassar a pantalla
}


void CVirtualRobotDlg::OnDrawItem(int nIDCtl, LPDRAWITEMSTRUCT
lpDrawItemStruct)
{
      if (nIDCtl == IDC_VIEW)
      {
            OpenGLDraw();
      }
      CDialog::OnDrawItem(nIDCtl, lpDrawItemStruct);
}

void CVirtualRobotDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
      OpenGL_pos_mouse=point;

      CDialog::OnLButtonDown(nFlags, point);
}

void CVirtualRobotDlg::OnRButtonDown(UINT nFlags, CPoint point)
{
      OpenGL_pos_mouse=point;

      CDialog::OnRButtonDown(nFlags, point);
}

void CVirtualRobotDlg::OnMouseMove(UINT nFlags, CPoint point)
{
```

```
	if(nFlags==MK_RBUTTON)
	{
		CRect rb,rd;
		CWnd * pBoto = GetDlgItem(IDC_VIEW);
		CWnd * pDialeg = GetDlgItem(IDD_VIRTUALROBOT_DIALOG);
		pBoto->GetWindowRect(&rb);
		this->GetWindowRect(&rd);

		if(rb.PtInRect(CPoint(point.x+3+rd.left,point.y+29+rd.top))){
			OpenGL_fita+=0.004*(double)(point.x-OpenGL_pos_mouse.x);
			OpenGL_psi+=0.004*(double)(point.y-OpenGL_pos_mouse.y);
			if (OpenGL_psi>PI*0.5) OpenGL_psi=PI*0.49999;
			if (OpenGL_psi<PI*(-1)*0.5) OpenGL_psi=(-1)*PI*0.49999;

			OpenGL_pos_mouse=point;

	OpenGL_Pos_x=OpenGL_Cen_x+OpenGL_R_CAMARA*sin(OpenGL_fita)*cos(OpenGL_psi
);

	OpenGL_Pos_y=OpenGL_Cen_y+OpenGL_R_CAMARA*cos(OpenGL_fita)*cos(OpenGL_psi
);
			OpenGL_Pos_z=OpenGL_Cen_z+OpenGL_R_CAMARA*sin(OpenGL_psi);
			GetDlgItem (IDC_VIEW)->Invalidate(FALSE);
		}
	}
	if(nFlags==MK_LBUTTON)
	{
		CRect rb,rd;
		CWnd * pBoto = GetDlgItem(IDC_VIEW);
		CWnd * pDialeg = GetDlgItem(IDD_VIRTUALROBOT_DIALOG);
		pBoto->GetWindowRect(&rb);
		this->GetWindowRect(&rd);

		if(rb.PtInRect(CPoint(point.x+3+rd.left,point.y+29+rd.top))){

			double px=OpenGL_Pos_x-OpenGL_Cen_x;
			double py=OpenGL_Pos_y-OpenGL_Cen_y;
			double pz=OpenGL_Pos_z-OpenGL_Cen_z;
			double sqxy=sqrt(px*px+py*py);
			double
sqxyz=sqrt(px*pz*px*pz+py*py*pz*pz+(px*px*px*px+py*py*py*py+2*px*px*py*py));
			OpenGL_Cen_x+=OpenGL_R_CAMARA*0.00067*(double)(point.x-
OpenGL_pos_mouse.x)*py/sqxy;
			OpenGL_Cen_y-=OpenGL_R_CAMARA*0.00067*(double)(point.x-
OpenGL_pos_mouse.x)*px/sqxy;

			OpenGL_Cen_x-=OpenGL_R_CAMARA*0.00067*(double)(point.y-
OpenGL_pos_mouse.y)*pz*px/sqxyz;
			OpenGL_Cen_y-=OpenGL_R_CAMARA*0.00067*(double)(point.y-
OpenGL_pos_mouse.y)*pz*py/sqxyz;
			OpenGL_Cen_z+=OpenGL_R_CAMARA*0.00067*(double)(point.y-
OpenGL_pos_mouse.y)*(px*px+py*py)/sqxyz;

			OpenGL_pos_mouse=point;

	OpenGL_Pos_x=OpenGL_Cen_x+OpenGL_R_CAMARA*sin(OpenGL_fita)*cos(OpenGL_psi
);

	OpenGL_Pos_y=OpenGL_Cen_y+OpenGL_R_CAMARA*cos(OpenGL_fita)*cos(OpenGL_psi
);
```

```
                    OpenGL_Pos_z=OpenGL_Cen_z+OpenGL_R_CAMARA*sin(OpenGL_psi);
                    GetDlgItem (IDC_VIEW)->Invalidate(FALSE);
              }
        }
        CDialog::OnMouseMove(nFlags, point);
}

BOOL CVirtualRobotDlg::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
{
        //Applies Zoom
        OpenGL_R_CAMARA-=(double)(zDelta);
        if(OpenGL_R_CAMARA<50) OpenGL_R_CAMARA = 50;
        OpenGL_Pos_x=OpenGL_Cen_x+OpenGL_R_CAMARA*sin(OpenGL_fita)*cos(OpenGL_psi
);
        OpenGL_Pos_y=OpenGL_Cen_y+OpenGL_R_CAMARA*cos(OpenGL_fita)*cos(OpenGL_psi
);
        OpenGL_Pos_z=OpenGL_Cen_z+OpenGL_R_CAMARA*sin(OpenGL_psi);
        GetDlgItem (IDC_VIEW)->Invalidate(FALSE);
        return CDialog::OnMouseWheel(nFlags, zDelta, pt);
}

void CVirtualRobotDlg::OnMButtonDown(UINT nFlags, CPoint point)
{
        //Reset to initial View parameters
        OpenGLInitPOSEView();
        GetDlgItem (IDC_VIEW)->Invalidate(FALSE);
        CDialog::OnMButtonDown(nFlags, point);
}

//Comprova que l'string passat sigui un nombre (accepta negatius i decimals)
bool CVirtualRobotDlg::formatNombre(CString str)
{
        int a=str.GetLength(); //Guardem longitud string
        bool err=false;
        bool dec=false;

        for(int i=0;i<a;i++){//Per cada dígit
              char c=str.GetAt(i);
              if((i!=0) && (c==45)) err=true; //no és el primer caràcter i és un
signe - (char[-]=150)
              else if(i!=0 &&(c==46)){      //Si és un . (indica decimals)
(char[.]=46) i no és el primer digit
                    if(dec==false) dec=true;
                    else err=true; //Si ja hi havia un punt anteriorment error
              }
              else if((c<48) || (c>57)){ //Sino es un numero
                    if(c!=45) err=true; //Sino es un signe menys al principi
(altrament entra al primer if)
              }
        }

        return err;
}

//Mètode que es crida al canviar l'acceleració límit
void CVirtualRobotDlg::OnEnChangechmlim()
{
        CString aux;
        m_chmLim.GetWindowText(aux); //Guardar la nova acceleració a l'string
auxiliar
```

```
	bool err=formatNombre(aux); //Comprovar que el format sigui correcte
	if(!err)m_chmLim.GetWindowText(CchmLim);
	else m_chmLim.SetWindowText(CchmLim);

	if(err) m_CListMsg.InsertString(0,"Introduit un format incorrecte en el
camp de l'acceleració");//Informació en el box de missatges
	else if(atof(CchmLim)>=0){//Si l'acceleració no és menor que 0
		chmLim= atof(CchmLim); //Guardar la nova acceleració a la variable
		m_CListMsg.InsertString(0,"Acceleració límit fixada a
"+CchmLim);//Informació en el box de missatges
	}
	//Si l'acceleració és més petita que 0
	else{
		CchmLim.Format("%f",chmLim);
		m_chmLim.SetWindowText(CchmLim);
		m_CListMsg.InsertString(0,"L'acceleració límit no pot ser més
petita que 0");//Informació en el box de missatges
	}
}

//Mètode que es crida al canviar l'escala
void CVirtualRobotDlg::OnEnChangeescala()
{
	CString aux;
	m_escala.GetWindowText(aux); //Guardar la nova escala a l'string auxiliar
	bool err=formatNombre(aux); //Comprovar que el format sigui correcte
	if(!err)m_escala.GetWindowText(Cescala);//Guardar la nova escala a
l'string corresponent
	else m_escala.SetWindowText(Cescala); //Deixar l'escala que hi havia

	if(err) m_CListMsg.InsertString(0,"Introduit un format incorrecte en el
camp de l'escala");//Informació en el box de missatges
	//Si l'escala no és menor que 0
	else if(atof(Cescala)>=0){
		escala= atof(Cescala); //Guardar la nova escala a la variable
		m_CListMsg.InsertString(0,"Escala fixada a "+Cescala);//Informació
en el box de missatges
	}
	//Si l'escala és més petita que 0
	else{
		Cescala.Format("%f",escala);
		m_escala.SetWindowText(Cescala);
		m_CListMsg.InsertString(0,"L'escala no pot ser més petita que
0");//Informació en el box de missatges
	}
}

//Mètode que es crida quan es canvia la dimensio del cub de l'espai de treball
restringit
void CVirtualRobotDlg::OnEnChangedimcub()
{
	CString aux;
	m_dimCub.GetWindowText(aux); //Guardar la nova dimensió a l'string
auxiliar
	bool err=formatNombre(aux); //Comprovar que el format sigui correcte
	if(!err)m_dimCub.GetWindowText(CdimCub);//Guardar la dimensió a l'string
corresponent
	else m_dimCub.SetWindowText(CdimCub); //Deixar la dimensió que hi havia
```

```
	float dimCubAux=atof(CdimCub);//Guardar la nova dimensió del cub a la
variable auxiliar

	if(err) m_CListMsg.InsertString(0,"Introduit un format incorrecte en el
camp de l'escala");//Informació en el box de missatges
	//Comprovar que el limit que es vol fixar no sobrepassi el limit
corresponenet del robot en una cirurgia laparoscòpica
	else if(dimCubAux>DIMCUBESPAITREBALLRESTRINGIT){
		dimCub=DIMCUBESPAITREBALLRESTRINGIT;//Fixar el limit del robot en
una cirurgia laparoscòpica
		CdimCub.Format("%f",dimCub);
		m_dimCub.SetWindowText(CdimCub); //Mostrar la dimensió del cub a
l'interficie
		m_CListMsg.InsertString(0,"La dimensio del cub que es volia fixar
sobrepassava els límits possibles");//Informació en el box de missatges
	}
	//Si la dimensió del cub que es vol fixar no és més gran que 0
	else if(dimCubAux<=0){
		CdimCub.Format("%f",dimCub);
		m_dimCub.SetWindowText(CdimCub); //Mostrar la dimensió del cub a
l'interficie
		m_CListMsg.InsertString(0,"La dimensio del cub que es volia fixar
no era major que 0");//Informació en el box de missatges
	}
	else{
		dimCub= dimCubAux;
		m_CListMsg.InsertString(0,"Dimensio del cub fixada a
"+CdimCub);//Informació en el box de missatges
	}
	//Si la nova dimensió del cub fixada és més petita que el marge es
modifica aquest últim
	if((dimCub/2)<marge){
		marge=(dimCub/2)/10; //Fixar el marge a un 10% la dimensió del cub
		Cmarge.Format("%f",marge);
		m_marge.SetWindowText(Cmarge); //Mostrar el marge a l'interficie
		m_CListMsg.InsertString(0,"El marge era més gran que la dimensió
del cub entre 2");//Informació en el box de missatges
		m_CListMsg.InsertString(0,"S'ha fixat un nou marge a un 10% de la
dimensió del cub entre 2");//Informació en el box de missatges
	}
}

//Mètode que es crida al canviar el marge
void CVirtualRobotDlg::OnEnChangemarge()
{
	CString auxiliar;
	m_marge.GetWindowText(auxiliar); //Guardar el nou marge a l'string
auxiliar
	bool err=formatNombre(auxiliar); //Comprovar que el format sigui correcte
	if(!err)m_marge.GetWindowText(Cmarge);//Guardar el marge a l'string
corresponent
	else m_marge.SetWindowText(Cmarge); //Deixar el marge que hi havia

	float aux=marge; //Guardem el marge actual en una variable auxiliar
	marge= atof(Cmarge); //Guardar el nou marge a la variable

	if(err) m_CListMsg.InsertString(0,"Introduit un format incorrecte en el
camp del marge");//Informació en el box de missatges
	//Mirar que el marge no sigui més gran que els limits de l'espai de
treball (si ho és no es canviarà el marge) o sigui més petit que 0
```

```
	else if((marge>(dimCub/2)) || marge<0){
		if(marge>(dimCub/2)) m_CListMsg.InsertString(0,"El marge que es
volia fixar no era més petit que els límits de l'espai de
treball");//Informació en el box de missatges
		else if(marge<0) m_CListMsg.InsertString(0,"El marge no pot ser
menor que 0");//Informació en el box de missatges

		marge=aux; //Deixar el marge com estava
		Cmarge.Format("%f",marge);
		m_marge.SetWindowText(Cmarge); //Mostrar el marge a l'interficie
	}
	else m_CListMsg.InsertString(0,"Marge fixat a "+Cmarge);//Informació en
el box de missatges
}

//Mètode que es crida si és modifica el checkbox de bloqueig del grau de
llibertat x
void CVirtualRobotDlg::OnBnClickedbx()
{
	bx=!bx;

	//Informació en el box de missatges
	if(bx)m_CListMsg.InsertString(0,"Bloqueig del grau de llibertat x
activat");
	else m_CListMsg.InsertString(0,"Bloqueig del grau de llibertat x
desactivat");
}

//Mètode que es crida si és modifica el checkbox de bloqueig del grau de
llibertat y
void CVirtualRobotDlg::OnBnClickedby()
{
	by=!by;

	//Informació en el box de missatges
	if(by)m_CListMsg.InsertString(0,"Bloqueig del grau de llibertat y
activat");
	else m_CListMsg.InsertString(0,"Bloqueig del grau de llibertat y
desactivat");
}


void CVirtualRobotDlg::OnBnClickedbz()
{
	bz=!bz;

	//Informació en el box de missatges
	if(bz)m_CListMsg.InsertString(0,"Bloqueig del grau de llibertat z
activat");
	else m_CListMsg.InsertString(0,"Bloqueig del grau de llibertat z
desactivat");
}

//Mètode que es crida si és modifica el checkbox de bloqueig del grau de
llibertat de la rotació
void CVirtualRobotDlg::OnBnClickedbr()
{
	br=!br;

	//Informació en el box de missatges
```

```
	if(br)m_CListMsg.InsertString(0,"Bloqueig del grau de llibertat de la
rotació activat");
	else m_CListMsg.InsertString(0,"Bloqueig del grau de llibertat de la
rotació desactivat");
}

//Timer que es fa servir per demanar dades al Polhemus i al dsPIC (pedal i
joystick) i pintar el robot virtual de l'interficie
void CVirtualRobotDlg::OnTimer(UINT_PTR nIDEvent)
{
	CString str;
	// TODO: Add your message handler code here and/or call default
	switch(nIDEvent)
	{
	case DATO_POLHEMUS:
		{
			//Si el robot es controla per el Polhemus li demanem una dada
(posició)
			if(!mestreTeclat){
				//Construir l'string per demanar una dada al Polhemus
				str=pm.GetMessagePolhemus(1,PM_COSINE,false,false);
				//Enviar l'string al Polhemus per demanar-li una dada
				ClientsockPolhemus->Send(&str);

				//Demanar una dada al dsPic a través del port serie
(COM)
				serie.SetOutput(COleVariant(CString('C')));
			}

			//Pintar el robot virtual
			OpenGLDraw();

			break;
		}
	default: break;
	}
	CDialog::OnTimer(nIDEvent);
}

//Mètode que rep events del teclat (per moure el robot amb el teclat quan no es
disposa del Polhemus)
BOOL CVirtualRobotDlg::PreTranslateMessage( MSG * pMsg ){
	if(pMsg->message==WM_KEYDOWN){
		int key= pMsg->wParam;
		CString str;
		str.Format("%d",key);
		//m_CListMsg.InsertString(0,"Key: "+str);

		//Si DOWN (incrementar la x)
		if(key==40){
			iPosicioTeclat[0]=0.5;
			//Inicialitzar el contador d'iteracions per els increments a
1
			contIncrement[0]=1;
			//Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
			str = Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOP);
			//Si el robot esta connectat li enviem l'increment
```

```
                  if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
            }
            //Si UP (decrementar la x)
            if(key==38){
                  iPosicioTeclat[0]=-0.5;
                  //Inicialitzar el contador d'iteracions per els increments a
1
                  contIncrement[0]=1;
                  //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                  str = Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOP);
                  //Si el robot esta connectat li enviem l'increment
                  if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
            }
            //Si RIGHT(incrementar la y)
            if(key==39){
                  iPosicioTeclat[1]=0.5;
                  //Inicialitzar el contador d'iteracions per els increments a
1
                  contIncrement[1]=1;
                  //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                  str = Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOP);
                  //Si el robot esta connectat li enviem l'increment
                  if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
            }
            //Si LEFT (decrementar la y)
            if(key==37){
                  iPosicioTeclat[1]=-0.5;
                  //Inicialitzar el contador d'iteracions per els increments a
1
                  contIncrement[1]=1;
                  //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                  str = Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOP);
                  //Si el robot esta connectat li enviem l'increment
                  if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
            }
            //Si RIGHT SHIFT (incrementar la z)
            if(key==16){
                  iPosicioTeclat[2]=0.5;
                  //Inicialitzar el contador d'iteracions per els increments a
1
                  contIncrement[2]=1;
                  //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                  str = Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOP);
                  //Si el robot esta connectat li enviem l'increment
                  if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
            }
            //Si RIGHT CONTROL (decrementar la z)
```

```
            if(key==17){
                  iPosicioTeclat[2]=-0.5;
                  //Inicialitzar el contador d'iteracions per els increments a
1
                  contIncrement[2]=1;
                  //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                  str = Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOP);
                  //Si el robot esta connectat li enviem l'increment
                  if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
            }
            //Si SUPR (activar/desactivar offset)
            if(key==46){
                  actOffset=!actOffset;
                  if(actOffset) m_CListMsg.InsertString(0,"Offset activat");
                  else m_CListMsg.InsertString(0,"Offset desactivat");
            }
            //Si FIN (rotar a l'esquerra)
            if(key==35){
                  iRotacioTeclat=0.5;
                  //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                  str =
Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOPROTATION);
                  //Si el robot esta connectat li enviem l'increment
                  if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
            }
            //Si AVPAG (rotar a la dreta)
            if(key==34){
                  iRotacioTeclat=-0.5;
                  //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                  str =
Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOPROTATION);
                  //Si el robot esta connectat li enviem l'increment
                  if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
            }
    }
    return CDialog::PreTranslateMessage(pMsg);
}

//Mètode que es crida quan es canvia el dispositiu mestre a través de
l'interficie
void CVirtualRobotDlg::OnBnClickedmestrepolhemus(){

      //Canviar el dispositiu mestre
      mestrePolhemus=!mestrePolhemus;
      mestreTeclat=!mestreTeclat;

      //Informació en el box de missatges
      if(mestrePolhemus){
            m_CListMsg.InsertString(0,"S'ha seleccionat el Polhemus com a
dispositiu mestre");
            CButton* pCheck = (CButton*)GetDlgItem(IDC_mestreTeclat);
```

```
			pCheck->SetCheck(0);
		}
		else{
			m_CListMsg.InsertString(0,"S'ha seleccionat el teclat com a
dispositiu mestre");
			CButton* pCheck = (CButton*)GetDlgItem(IDC_mestreTeclat);
			pCheck->SetCheck(1);
		}
		UpdateData(true);
}

//Mètode que es crida quan es canvia el dispositiu mestre a través de
l'interficie
void CVirtualRobotDlg::OnBnClickedmestreteclat(){
		mestrePolhemus=!mestrePolhemus;
		mestreTeclat=!mestreTeclat;

		//Informació en el box de missatges
		if(mestreTeclat){
			m_CListMsg.InsertString(0,"S'ha seleccionat el teclat com a
dispositiu mestre");
			CButton* pCheck = (CButton*)GetDlgItem(IDC_mestrePolhemus);
			pCheck->SetCheck(0);
		}
		else{
			m_CListMsg.InsertString(0,"S'ha seleccionat el Polhemus com a
dispositiu mestre");
			CButton* pCheck = (CButton*)GetDlgItem(IDC_mestrePolhemus);
			pCheck->SetCheck(1);
		}
		UpdateData(true);
}

BEGIN_EVENTSINK_MAP(CVirtualRobotDlg, CDialog)
		ON_EVENT(CVirtualRobotDlg, IDC_MSCOMM1, 1,
CVirtualRobotDlg::OnCommMscomm1, VTS_NONE)
END_EVENTSINK_MAP()

//Mètode que es crida quan es rep una dada per el port serie COM
void CVirtualRobotDlg::OnCommMscomm1(){

		COleVariant Input;
		Input=serie.GetInput(); //Guardar el missatge rebut
		int status_actual;
		CString strIn(Input.bstrVal); //Guardar el missatge en un string
		sscanf(strIn,"%d",&status_actual);//Guardar la tira de 16 bits rebuda en
un double

		//PEDAL
		if((status_actual & 0x01)==0x01){//si el pedal està apretat
			actOffset=false;
		}
		if((status_actual & 0x01)==0x00){//si el pedal no està apretat
			actOffset=true;
		}

		//JOYSTICK
		if((status_actual & 0x40)==0x40){//si s'apreta el joystick a la dreta
				iRotacioTeclat=1;
```

```
                    //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                    CString str =
Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOPROTATION);
                    //Si el robot esta connectat li enviem l'increment
                    if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
      }
      if((status_actual & 0x10)==0x10){//si s'apreta el joystick a l'esquerra
                    iRotacioTeclat=-1;
                    //Enviar increments 0 iniciar el cicle de
(enviarPosicioControladora->rebreRespostaControladora-
>enviarPosicioControladora)
                    CString str =
Interpret::incremental_cartesian(0,0,0,0,0,0,TELEOPROTATION);
                    //Si el robot esta connectat li enviem l'increment
                    if(ConnRobot) ClientsockRobot->Send(&str); //Enviar la nova
posició al robot
      }
}
```

9.3 ANNEX III. Polhemus Liberty

POLHEMUS
INNOVATION IN MOTION™

LIBERTY

THE FORERUNNER IN ELECTROMAGNETIC TRACKING TECHNOLOGY

THE ABSOLUTE BEST

The fastest, most accurate, scalable electromagnetic tracker available, LIBERTY™ represents a quantum leap in new technology. State-of-the-art Digital Signal Processor (DSP) electronics make it the perfect real-time solution for 6 Degree-of-Freedom (6DOF) needs. LIBERTY has speed, ease-of-use via an intuitive Graphical User Interface (GUI), scalability, distortion sensing, and improved signal-to-noise ratios which increase stability and resolution while providing consistent high quality data.

FEATURES

Update Rate

LIBERTY tracks objects at a speed of 240 updates per second, all sensors simultaneously. Latency is less than 4ms.

Scalable

LIBERTY 240/8
Four sensor channels are available on the base product. The addition of a single circuit board inside the same chassis allows for a total of eight sensor channels.
LIBERTY 240/16
Four sensor channels are available on the base product. The system is then upgradeable to 8, 12, or 16 sensor channels within the same chassis by having additional circuit board(s) installed.

Communications Interface

LIBERTY operates via both an RS-232 serial interface and USB.

Distortion Sensing

Each sensor with LIBERTY is able to independently detect distortion within the environment, alerting the user to make appropriate changes if necessary.

Multiple User Definable Profiles

The GUI allows four independent user-definable profiles for setting system parameters such as filtering, output formats, coordinate rotations and many more.

Multiple Output Formats

User may select position in Cartesian coordinates (English or metric); orientation in direction cosines, Euler angles or quaternions.

Angular Coverage

All-attitude

Drift-Free

Solid state electronics

THE PROFESSIONAL'S CHOICE

The Forerunner in Tracking Technology

LIBERTY incorporates an unprecedented speed of 240 updates per second per sensor. The system comes complete with distortion detection while providing the most accurate and consistent data. With the addition of the Polhemus stylus, LIBERTY becomes a highly accurate 3D digitizer.

Easy, Intuitive User Interface

LIBERTY comes standard with Windows® 2000/XP/Vista® GUI and a comprehensive, easy to use Software Developers Kit (SDK). The GUI allows four independent user definable profiles for setting system parameters such as filtering, output formats, coordinate rotations and many more. This is a valuable feature for multiple applications or users. For visualization, an integrated motion box provides navigable points of view and can include text data. Additional features include a data record/playback component, plus the ability to quickly export data via Microsoft® "Named Pipe."

A/C Magnetics: Increases Stability, Resolution, Speed and Range

Incorporating state of the art Digital Signal Processor (DSP) electronics in concert with A/C magnetics provides the user with improved signal-to-noise ratios which increase range, stability, resolution and speed. The system is essentially unaffected by facility power grids, and update rates are always maintained, allowing it to keep up with the voluminous rate of tracking data produced.

APPLICATIONS

Limited only by your imagination!

LIBERTY

COMPONENTS

LIBERTY includes a System Electronics Unit (SEU), one sensor and one source.

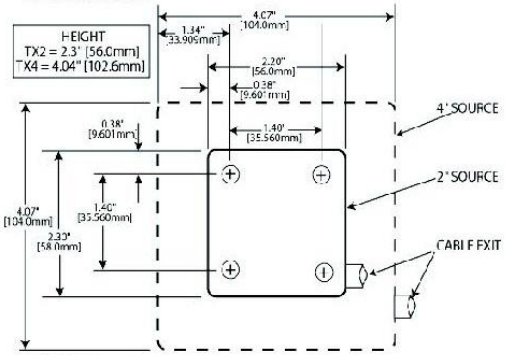
Optional accessories include a longer range source, stylus and a variety of cable lengths for each sensor, stylus or source.

System Electronics Unit

Contains the hardware and software necessary to generate and sense the magnetic fields, compute position and orientation, and interface with the host computer via RS-232 or USB.
12.2 in. (31 cm) L x 7 in. (17.8 cm) W x 8.5 in. (21.6 cm) H; weight 9 lbs. (4.1 kg)
240/12 and 240/16:
12.2 in. (31 cm) L x 7 in. (17.8 cm) W x 11 in. (27.94 cm) H; weight 11 lbs. (5 kg)

Source

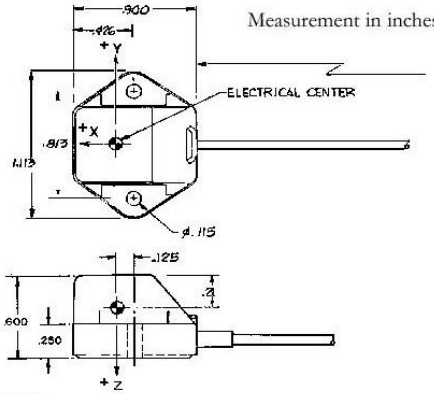
The source is the system's reference frame for sensor measurements.



Weight
TX2: 8.8 oz. (250 gm) **Thread size** 1/4" x 20
TX4: 1.60 lbs. (726 gm) **Thread size** 1/4" x 20

Sensor

A lightweight, small cube, the sensor's position and orientation is precisely mea ured as it is moved.



Weight
0.32 oz. (9.1 gm)

POLHEMUS
INNOVATION IN MOTION™

The systems are not certified for medical or bio-medical use. Any reference to medical or bio-medical use are examples of what medical companies have done with the systems after obtaining all necessary or appropriate medical certifications. The end user/OEM must comply with all pertinent FDS/CE and all other regulatory requirements.

SPECIFICATIONS

Update Rate
240 Hz per sensor, simultaneous samples

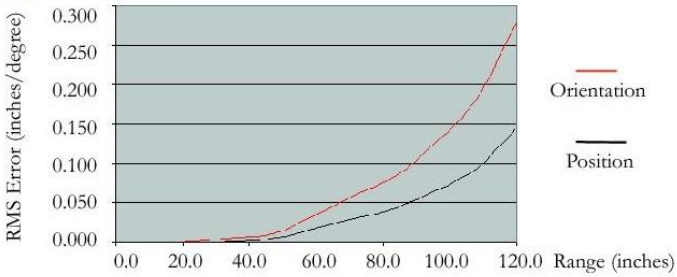
Latency
3.5 milliseconds

Number of Sensors
240/8 has 1 to 8 sensors, 240/16 has 1 to 16

Static Accuracy
0.03 in. RMS for X, Y or Z position; 0.15° RMS for sensor orientation

Interface
USB; RS232 to 115,200 Baud rate, both standard

Range vs. Resolution



Range (inches)	Position Resolution (inches)	Orientation Resolution (degrees)
12.0	0.00005	0.0004
24.0	0.0002	0.0014
36.0	0.001	0.0048
48.0	0.005	0.0117
72.0	0.031	0.060
120.0	0.145	0.280

Multiple Systems
Provision available to operate two separate systems in same environment

Data format
Operator selectable ASCII or IEEE 754 binary; English/Metric Units

External Event Marker
User input flag and output marker

Output Sync Pulse
TTL frame sync output

Software Tools
GUI and SDK included
USB drivers for Windows® XP/Vista® included
Linux®- contact factory

Operating Temperature
0°C to 50°C at a relative humidity of 10% to 95%, noncondensing

Power Requirements
100-240 VAC, 50 – 60 Hz, single phase, 50 W

Regulatory
FCC Part 15, class A
CE: EN61326-1: 1997/A1:1998/A2:2001/A3:2003 emission
EN61326-1: 1997/A1:1998/A2:2001/A3:2003 Immunity

*Large metallic objects, such as desks or cabinets, located near the source or sensor, may adversely affect the performance of the system.

www.polhemus.com

40 Hercules Drive • PO Box 560 • Colchester, Vermont 05446-0560
US and Canada 800.357.4777 • 802.655.3159 • fax 802.655.1439

ISO 9001 REGISTERED

9.4ANNEX IV. Robot Stäubli RX60B

A continuació es descriuen les diferents característiques del braç robòtic utilitzat, la majoria derivades de la seva estructura mecànica.

Dimensions

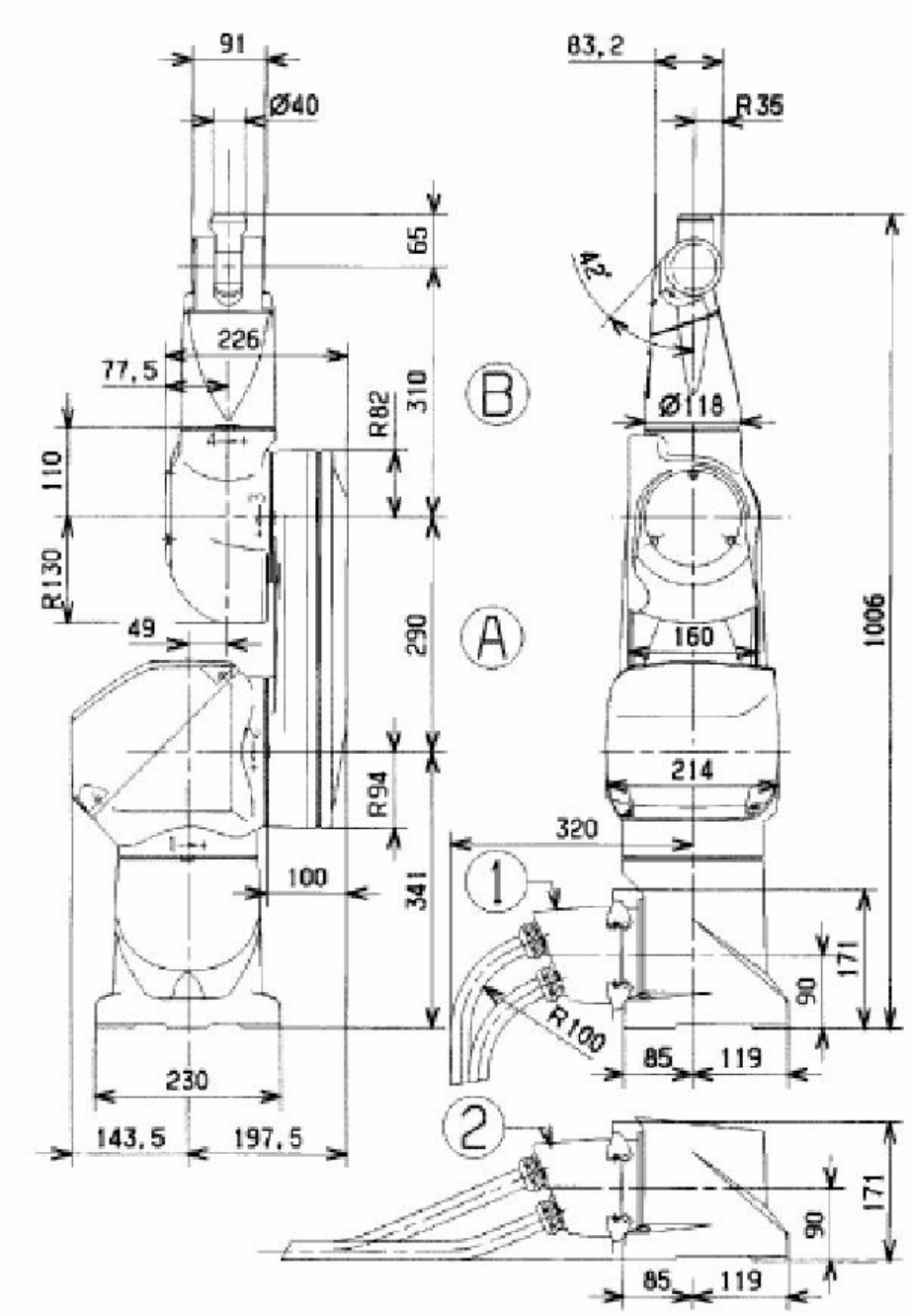


Figura 57: esquema de les dimensions del model Stäubli RX60B

Graus de llibertat

El braç robòtic utilitzat consta de 6 graus de llibertat, tots ells rotacions.

A la figura 58 es pot observar l'estructura mecànica del model Stäubli RX60B i els seus 6 graus de llibertat numerats:

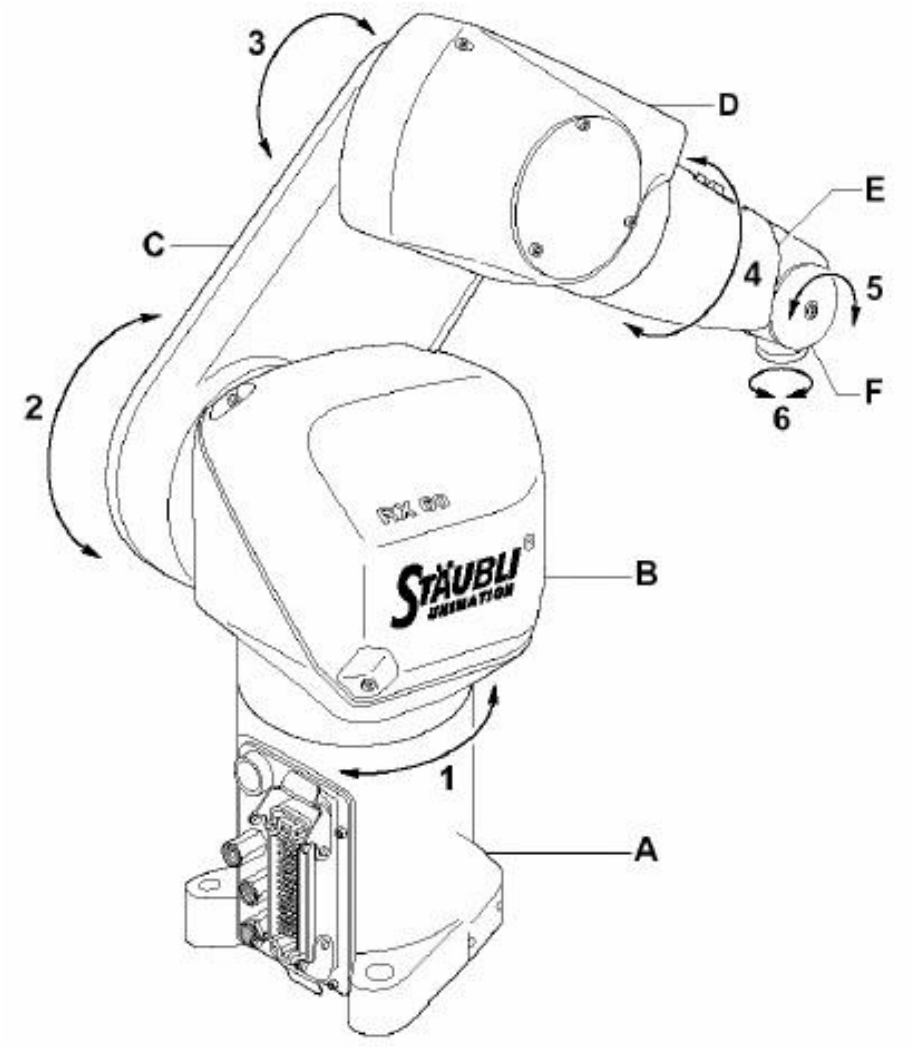


Figura 58: Elements del braç robòtic (la base (A), l'espatlla (B), el braç (C), el colze (D), l'avantbraç (E) i el canell (F)) i graus de llibertat (numerats).

Volum de treball

Es defineix el volum de treball (figura 59) com el conjunt de posicions abastables per l'element terminal.

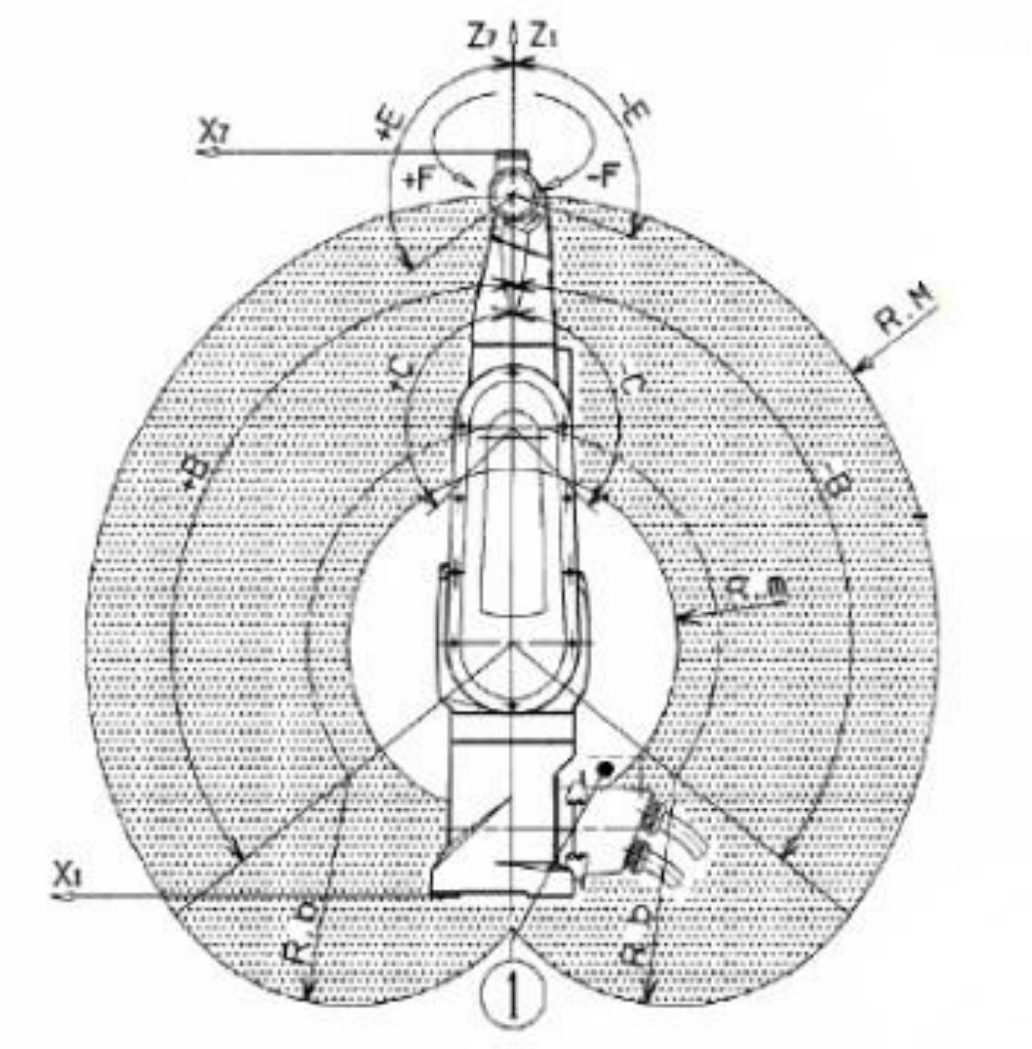


Figura 59: volum de treball del robot Stäubli RX60B

	Punt del robot	Articulacions	Dimensió
Radi màxim (R.M)	Centre del canell	Articulacions entre 2 i 5.	600 mm
Radi mínim (R.m)	Centre del canell	Articulacions entre 2 i 5.	233 mm
Radi suplementari (R.b)	Centre del canell	Articulacions entre 3 i 5.	310 mm

Característiques tècniques

Articulació	A	B	C	D	E	F
Límit d'amplitud (º)	320	255	269	540	230	540
Distribució del rang de treball (º)	+/-160	+/-127.5	+/-134.5	+/- 270	+120.5 / -109.5	+/- 270
Velocitat nominal (º/s)	287	287	319	410	320	700
Resolució angular ($º \cdot 10^{-3}$)	0.724	0.724	0.806	1.177	0.879	2.747

* L'amplitud de les articulacions pot ser limitada a través de software

Condicions de treball:

- Temperatura: de 5ºC fins a 40ºC
- Humitat: de 30% fins a 95%, sense condensació
- Altitud màxima: 2000m

Pes:

- Braç estàndard (sense element terminal): 44kg

Capacitat de càrrega:

- A velocitat nominal: 2.5kg
- A velocitat reduïda: 4.5kg

9.5 ANNEX V. Documentació Robot Controller Emulator

App. version: 1.5
Doc. version: 1.0
Autor: Albert Hernansanz, GRINS-UPC (albert.hernansanz@upc.edu)

1. Descripció

Robot Controller Emulator és un programa que emula les controladores dels robots Staübli models CS7 i CS8 (Maxilo, Certap i CS8), permetent que les aplicacions que interactuen amb els robots puguin fer-ho sense haver d'usar els robots reals. L'emulació és totalment transparent per l'usuari. L'aplicació crea un servidor que es comunica mitjançant un socket, tal i com ho fan les controladores reals.

L'emulació es basa en dos aspectes:

- Capa de comunicacions idèntica: D'aquesta manera, tant les comunicacions sortints com entrants són idèntiques. Es pot utilitzar la classe *interpret* per enviar i rebre comandes.
- Cinemàtica: La cinemàtica dels robots reals està contemplada en l'emulador, així com el conjunt de missatges d'error que generen els robots reals: sobreeiximent dels límits absoluts angulars, dels increments angulars, ...

La controladora virtual està configurada segons els següents límits (Fig.1):

	Min Abs	Forat		Max Abs	
J0	-160			160	Graus
	-2,79	0	0	2,79	Radiants
J1	-217,5			94.8	Graus
	-3,79	0	0	1.65	Radiants
J2	-44,5	85	95	224,5	Graus
	-0,78	1,48	1,66	3,92	Radiants
J3	-175			175	Graus
	-3,05	0	0	3,05	Radiants
J4	-112	-5	5	112	Graus
	-1.9548	-0,09	0,09	2,01	Radiants
J5	-175			175	Graus
	-3,05	0	0	3,05	Radiants

Fig1: Les caselles en blau representen els límits reals, corresponent a la configuració escollida pels robots (Esquerra, colze amunt, Canell avall)

Les limitacions respecte als increments màxims són (en graus):

- *incremental_cartesian* < 20.1
- *incremental_joints* < 20.1
- *incremental_tool* < 20.1

Si es desitja modificar els límits, s'ha d'editar el fitxer de configruació del robot: *StaubliLimitsCtrNoModel.rob*, que es troba dins el directori */DataFiles*. Per més informació sobre aquest fitxer, consultar l'Apèndix 1.

2. Requeriments

Fitxer de descripció del robot a emular: *.\DataFiles\StaubliLimitsCtrNoModel.rob*. Atenció que aquest s'ha de trobar dins el subdirectori *.\DataFiles*. Si es vol modificar els límits, es recomana fer una còpia de l'original amb un altre nom i guardar els canvis en el fitxer amb el nom original.
El port a utilitzar per part de l'aplicació ha d'estar lliure i el tallafocs ha de permetre la comunicació tant entrant com sortint.

3. Funcionament

El servidor es crea automàticament a l'iniciar l'aplicació. Si es pot crear satisfactòriament, en la pantalla "Controller Status" dona el següent missatge:

Server @=PC-ALBERT[169.254.2.2:1005] || *Status: Listening for incoming connections*

On:

- PC-ALBERT: Alias del pc on s'està executant l'aplicació.
- *169.254.2.2:1005:* Adreça ip del pc, així com el port utilitzat.
- *Status: Listening for incoming connections:* Informa que l'aplicació està llesta per rebre comandes des d'un client.

Si per algun motiu (port no disponible, tallafocs, etc...) no es pot iniciar el servidor, en Controller Status es mostra el següent missatge: "*Imposible to listen to incoming connections*"

En la següent imatge es pot veure la interfície de l'aplicació d'emulació (fig.2) i que servirà de guia per descriure la funcionalitat de *Robot Controller Emulator*.

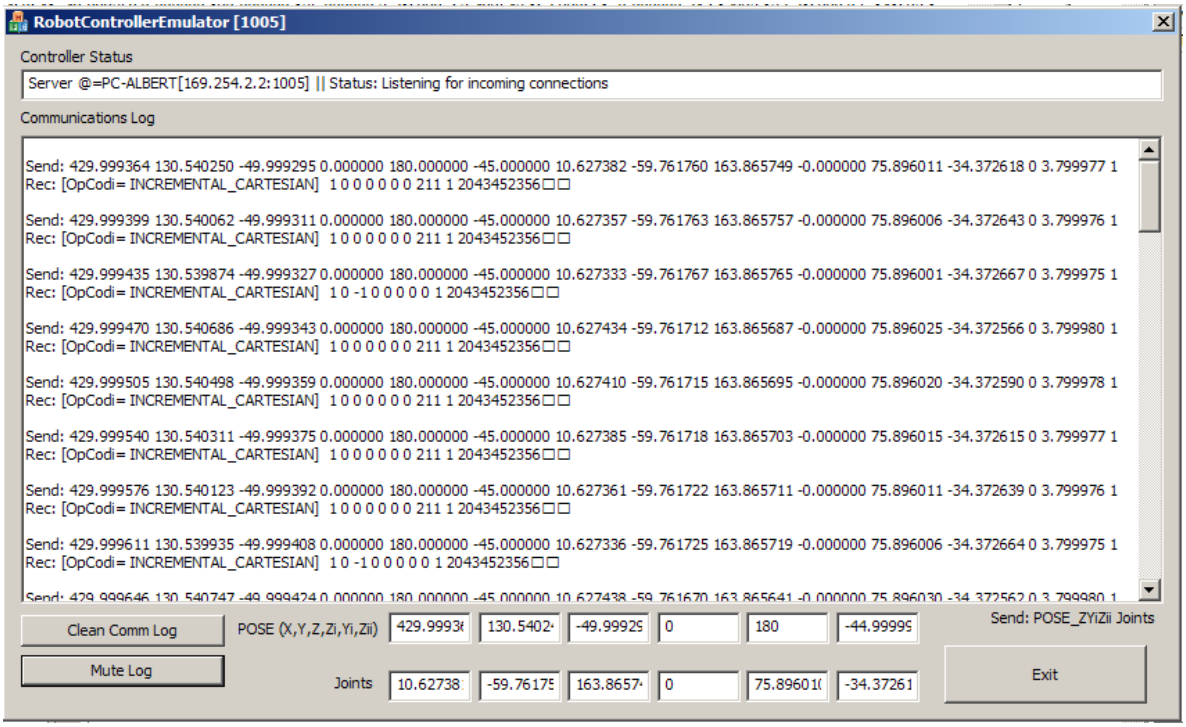


Fig. 2: Interfície de Robot Controller Emulator

a. *Controller Status*:
Mostra les comandes d'estat de l'emulador, així com si es produeix alguna situació en la que succeeix una excepció en el robot (límits angulars, ...)

b. Communications Log:
Mostra les comunicacions entrants i sortints generades segons el protocol establert en la classe *Interpret*.
- Send: Missatge generat per l'aplicació d'emulació.
- Rec: [Codi operació] + missatge rebut.

c. *Clear Comm Log:*
Esborra el contingut de les pantalles *Controller Status* i *Communications Log*.

d. *Mute Log*:
Permet activar o desactivar la pantalla de *Controller Status*. Inicialment està en mode "*mute*". Al canviar d'estat, el contingut de la pantalla *Communications Log* no s'esborra.

e. POSE(X,Y,Z,rZ,rYi,rZii):
POSE de l'element terminal del robot. La posició s'expressa en mm, mentre que l'orientació segueix la notació d'Euler Z, Y', Z''.

f. Joints:
Posició de cadascuna de les articulacions del robot, expressada en graus.

g. *Exit:*
Eixir de l'aplicació, destruint el servidor i alliberant el port.

4. Apèndix 1: Robot File Descriptor

```
/*Robot File Descriptor v2.0
Author/s: Xavier Giralt & Albert Hernansanz      [GRINS-ESAII.UPC]
Contact: xavier.giralt@upc.edu, albert.hernansanz@upc.edu
*/
//Version Robot File Descriptor
2.0
//Robot Id
StaubliRx60B
//Num SubObjects
7
//Limits of each joint [rad,mm] Limits of real controller.
-2.792527 2.792526
-3.796091 1.654498
-0.7767 3.9182
-3.0543 3.0543
-1.9548 1.9548
-3.0543 3.0543
//D-H description (Tipe of(R=1 v P=0), Alpha, a,d, Theta)
1 0.0          0.0          0.0          0.0
1 -1.570796 0.0             0.0          0.0
1 0.0          290.0  49.0   0.0
1 1.570796   0.0     310.0  0.0
1 -1.570796 0.0             0.0          0.0
1 1.570796   0.0             0.0          0.0
//End effector Transformation Matrix parameters        NOTE: AxisZ: +65mm to arribe to
the end of the Staubli robot
0.0     0.0     195.0 0.0 0.0 0.0
//Seed Link files
.\\DataFiles\\Rob2Art
//Filetypes
-1
```