*UNIVERSITAT POLITÈCNICA DE CATALUNYA*

# CONTROL LOCAL I REMOT DE PROCESSOS MITJANÇANT JAVA SOBRE UNA PLATAFORMA WINDOWS

*PROJETE FINAL DE CARRERA*


Document: **MEMÒRIA DEL PROJECTE**

Alumne: Daniel Castillo Hand.

Director: Josep Cugueró Escofet.


Data: Juny de 2004

Titulació: Enginyeria Tècnica Industrial.
Especialitat: Electrònica Industrial.


Escola: Escola Universitària d'Enginyeria Tècnica Industrial de Terrassa

Departament: Enginyeria de Sistemes, Automàtica i Informàtica Industrial

# ÍNDEX

# 1. INTRODUCCIÓ.

El present projecte, tal i com indica el títol tracta sobre el Control local i remot de processos mitjançant Java sobre una plataforma Windows.

Pel que fa al control, s'entén com a tal el conjunt d'operacions orientades a supervisar l'estat d'un sistema, tal com un vehicle, màquina o procés industrial, per tal de reduir-ne o anul·lar-ne la desviació respecte al comportament desitjat del mateix. Control local es refereix al cas en el qual la gestió de les esmentades operacions son realitzades per un sistema electrònic (digital o analògic), mecànic, hidràulic,...ubicat en un espai adjunt a l'indret on es troba ubicat el procés, amb una connexió amb aquest per la qual no es transmet altra informació que la pròpia de la comunicació destinada a aquesta gestió. Control remot fa referència al cas en el qual la gestió de les operacions destinades al control son realitzades per un sistema , generalment electrònic digital o informàtic, ubicat en un indret remot respecte a la localització del procés. En aquest cas la comunicació entre el gestor i el procés es realitza a través d'un canal compartit amb altres paquets d'informació aliena a la pròpia de la comunicació destinada a aquesta gestió. En el present cas, el gestor de les operacions de control serà un ordinador i el canal a través del qual es realitzarà la comunicació en el cas de control remot serà Internet.

Java fa referència a l'entorn de programació del mateix nom format per un llenguatge de programació i un conjunt d'eines i utilitats destinades a ésser utilitzades de forma accessòria en la programació amb ell.

Quan es parla de Plataforma Windows es fa referència a l'entorn que envolta aquest sistema operatiu, sobre el qual se sustenten les diverses aplicacions a nivell d'usuari. D'aquesta forma l'ordinador al que es feia referència anteriorment serà un d'arquitectura tipus PC.

La present memòria s'estructura en capítols, el contingut dels quals se sintetitza a continuació:

El capítol 1, el present, és l'introductori; en ell s'introdueixen els continguts del projecte, s'explica l'estructura de la Memòria així com certs requisits de coneixements previs per al major aprofitament en el seguiment de la mateixa.

El capítol 2 enuncia els objectius del projecte, així com els condicionants tècnics i els imposats a l'enunciat i als objectius del projecte a tenir en compte per al desenvolupament del mateix.

El capítol 3 descriu els elements de l'entorn en el qual es desenvoluparà el projecte, alguns d'ells amb gran detall per així comprendre millor certes decisions i camins que es prenguin per tal de resoldre problemes apareguts durant el desenvolupament.

El capítol 4 descriu l'antecedent principal d'aquest projecte, element a partir del qual es partirà en el camí per a la consecució dels objectius.

El capítol 5 enuncia els diversos problemes a resoldre i el 6 estudia les diverses alternatives per solucionar-los.

El capítol 7 desenvolupa les solucions adoptades en el capítol anterior amb la finalitat de l'obtenció de resultats, tècnicament més tangibles i el 8 descriu amb detall els resultats obtinguts.

El capítol 9 representa la càrrega temporal de les diverses tasques realitzades durant el desenvolupament del projecte, el 10 es fa una estimació del cost econòmic del desenvolupament del projecte, l'11 expressa les conclusions i el 12 cita les referències externes emprades en aquest projecte.

Finalment l'Apèndix conté el conjunt d'arxius font de diversos codis emprats en el desenvolupament, que per la seva naturalesa i extensió no s'han inclòs en el cos de la memòria. L'apèndix també conté certs càlculs detallats efectuats per tal de resoldre problemes numèrics concrets el resultats dels quals si son citats en el cos de la memòria.

Cal mencionar, que per al correcte seguiment de la present memòria, així com per a la màxima comprensió dels seus continguts és bastant recomanable la possessió de coneixements previs en el camp de la programació orientada a objectes en llenguatge C++ així com en Java, del control automàtic continu així com discret de processos i de l'entorn de desenvolupament, programació i aplicació al control de Matlab.

# 2. OBJECTIUS DEL PROJECTE.

## 2.1. DESCRIPCIÓ DELS OBJECTIUS.

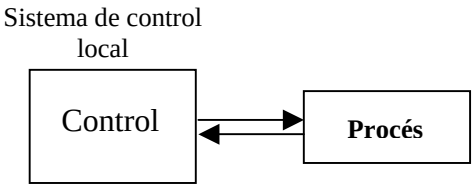
### 2.1.1. ESTUDI DE VIABILITAT DE CONTROL LOCAL.

El primer objectiu consisteix en l'estudi de la viabilitat de la realització de control local mitjançant l'entorn del llenguatge de programació Java, a través del PC, amb el procés de la maqueta seleccionada per controlar. Per aquesta tasca es procurarà aprofitar al màxim el material del projecte descrit al capítol d'antecedents amb les millores i ampliacions que per la necessitat de la situació concreta siguin necessàries.

### 2.1.2. APLICACIÓ DE CONTROL LOCAL.

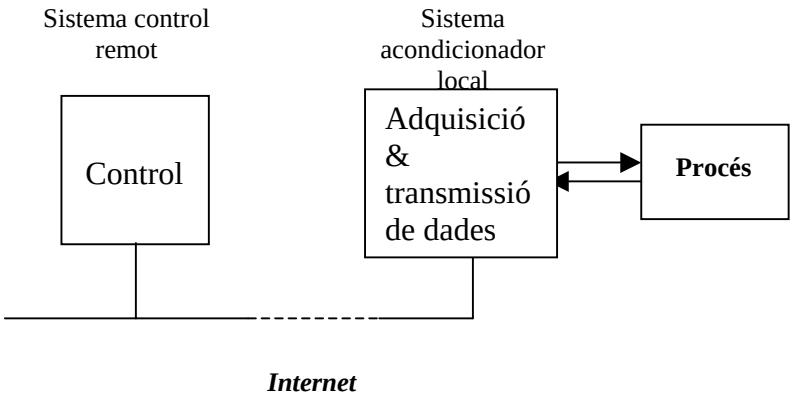
Amb posterioritat a la consecució de l'objectiu descrit a l'apartat anterior i en cas de ser el resultat prou satisfactori, el següent objectiu seria la realització d'una interfície per al control local del procés seleccionat. Per tal d'il·lustrar-ne el funcionament, és objectiu el disseny d'una aplicació en llenguatge Java per controlar una de les variables del procés seleccionat.

En la consecució d'aquest objectiu, com a pas previ al disseny de l'aplicació, serà necessària el modelat del procés a controlar, així com el disseny del corresponent controlador digital.



### 2.1.3. APLICACIÓ DE CONTROL REMOT.

Un altre objectiu important consisteix en l'estudi de la viabilitat de controlar remotament, interactuant al nivell més baix possible, el procés a través d' Internet. En cas de cas de ser el resultat prou satisfactori el subsegüent objectiu seria el disseny de les classes necessàries per tal de realitzar el control del procés de forma remota, a partir de la classe dissenyada anteriorment pel control local.




Finalment i en cas satisfactori s'hauria d'implementar una aplicació per tal d'il·lustrar el funcionament del control remot del procés seleccionat.

## 2.2. CONDICIONANTS.

### 2.2.1. ÚS DEL LLENGUATGE DE PROGRAMACIÓ JAVA.

El principal condicionant per la realització del present projecte és el de l'ús del llenguatge de programació Java per a la interacció, al nivell més baix possible com a mínim a nivell d'usuari, amb la maqueta del procés a controlar. És a dir, la interacció amb la maqueta ha d'ésser a través d'aquest llenguatge tant a nivell local com remot i al nivell més baix possible, tot emprant, quan aquesta interacció a baix nivell no sigui del tot possible, les eines disponibles per part del propi llenguatge Java.

### 2.2.2. DESENVOLUPAMENT SOBRE UNA PLATAFORMA WINDOWS.

Aquest condicionant, especificat al títol del projecte i descrit al capítol introductori, quedaria cobert en l'acompliment del condicionant anterior, ja que Java, com es descriurà en el següent capítol, proporciona aplicacions que poden funcionar sobre qualsevol plataforma, entre elles Windows, sempre que es compleixin uns requisits de Software instal·lat sobre aquesta.

### 2.2.3. UTILITZACIÓ DE TECNOLOGIA IDL PER A LES COMUNICACIONS.

Donat que el tema de les Comunicacions Industrials en general i en concret el control remot de processos ja han sigut objecte de diversos treballs anteriors, i en ells s'han desenvolupat diversos sistemes de control remot en llenguatge Java com ara Sockets, Servlets, Applets, i Java RMI, és també condicionant del projecte l'ús en el desenvolupament per a l'acompliment de l'objectiu del control remot, la investigació i a ser possible la utilització de la tecnologia IDL i l'arquitectura CORBA com a eines per a les comunicacions industrials. IDL, de la mateixa manera que RMI, son tecnologies que proporcionen al llenguatge Java eines pel desenvolupament d'aplicacions distribuïdes en diferents màquines.

La tecnologia IDL es diferencia respecte RMI en que la segona treballa íntegrament en Java, mentre la primera, que està basat en CORBA ("Common Object Request Broker Architecture"), permet treballar a més a més amb uns altres llenguatges de programació.

### 2.2.4. DESENVOLUPAMENT EN L'ENTORN DE LA MAQUETA DEL PONT-GRUA.

L'altre condicionant important per al desenvolupament del projecte consisteix en el seu desenvolupament per tal d'interactuar amb la maqueta del pont-grua, la descripció detallada de la qual figura al següent capítol.

### 2.2.5. CONDICIONANT TEMPORAL.

Finalment, un altre condicionant a tenir en compte, encara que més implícit que els anteriors, és el referit al límit temporal. El projecte s'ha de desenvolupar en un període d'uns quatre mesos naturals, aproximadament, i la seva càrrega temporal de treball està prevista ser d'unes 450 hores reals.

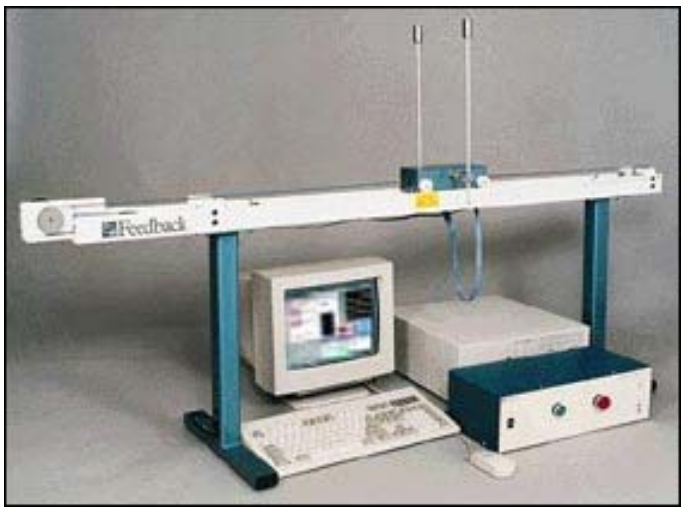
# 3. ENTORN DE DESENVOLUPAMENT.

## 3.1. MAQUETA DEL PONT-GRUA.

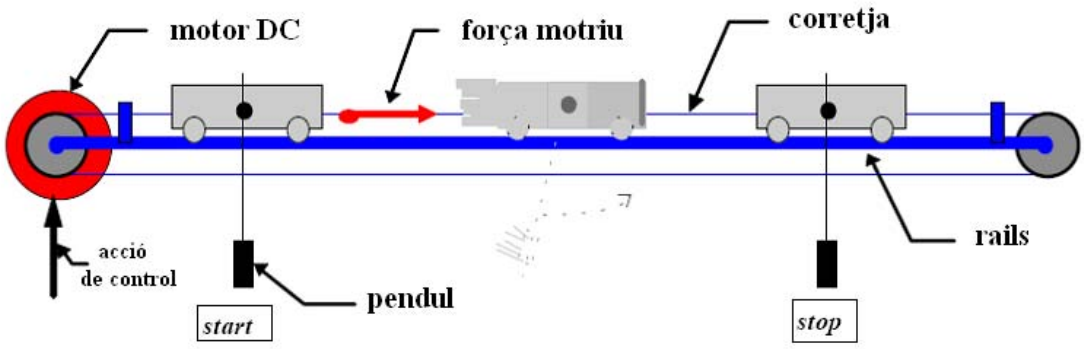
### 3.1.1. DESCRIPCIÓ DE LA MAQUETA.

La maqueta sobre la qual es desenvoluparà el projecte és la maqueta ***DIGITAL PENDULUM CONTROL SYSTEM 33-005*** de la casa ***Feedback***, a la qual es farà referència a partir d'ara com a **Maqueta del pont-grua**. L'esmentada maqueta es troba ubicada al laboratori de control avançat (aula 2.05) de l'edifici TR11 del Campus de Terrassa de la Universitat Politècnica de Catalunya a l'any acadèmic 2003-2004.

La maqueta del pont-grua consisteix en un carro recolzat sobre dos rails d'alumini elevats amb dos pènduls "bessons" suspesos del mateix. El carro es troba subjecte per mitjà d'una corretja encaixada a dos engranatges situats als dos extrems dels rails. Un motor de corrent continu exerceix tracció sobre un d'aquests engranatges, mentre l'altre gira lliurement.



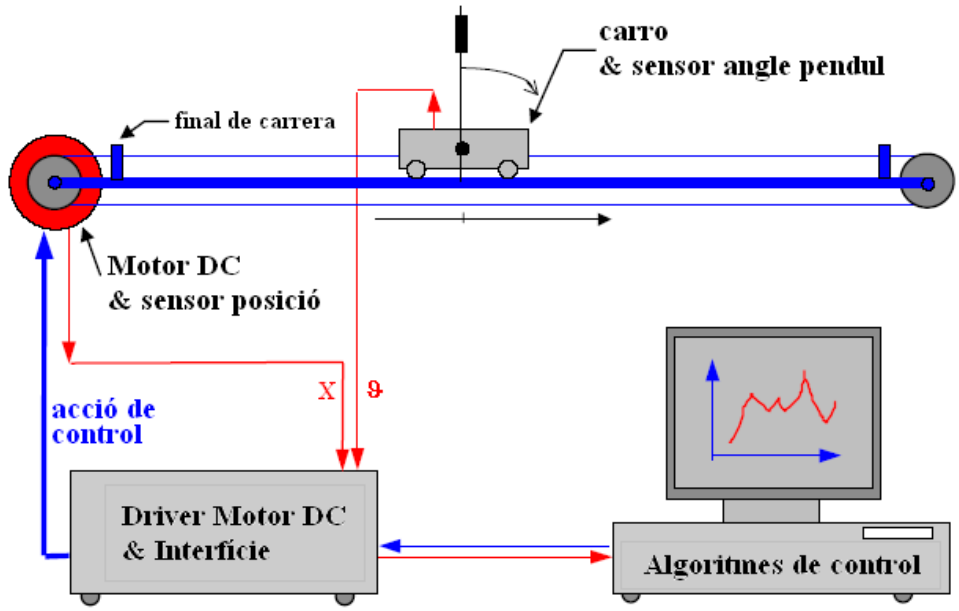

El pèndol pivota lliurement, mentre dos "encoders" situats al carro i a l'engranatge motriu respectivament, proporcionen la posició angular del pèndul i la posició lineal del carro.




*Maqueta del pèndol-grua*

### 3.1.2. INTERFÍCIE DE COMUNICACIÓ.

La comunicació entre l'usuari i la maqueta està prevista per a ésser realitzada a través del programa *MATLAB* en una versió igual o superior a la 5.3.




ϑ : Posició angular del pendul

X : Posició lineal del carro

Juntament amb la maqueta, el fabricant subministra un seguit de "software" consistent en:

- Un RTK (*Real-Time Kernel*).
- Conjunt de funcions de comunicació.
- Un conjunt d'algoritmes d'exemples de control ("toolbox").

El RTK realitza la funció de supervisió del conjunt de tasques en temps real, tot creant un llaç de control. Les funcions de comunicació entre l'entorn de *MATLAB* i el RTK es troben en una llibreria d'enllaç dinàmic (*.DLL), anomenada ***pd_call.dll*** implementada per a interactuar amb ella a través de *MATLAB*. Aquesta llibreria ha estat escrita de forma tancada, cosa que impossibilita la seva manipulació, ja que el RTK conté totes les funcions necessàries per realitzar el control així com la recol·lecció de dades en temps real. Per poder utilitzar la llibreria cal que estigui ubicada en un directori de treball de l'entorn *MATLAB* des d'on es pretengui interactuar amb la maqueta.

Totes les funcions responsables de la comunicació entre el RTK i l'entorn de *MATLAB* tenen el següent protocol de crida des de *MATLAB:*

```
valorRetorn = pd_call( 'NomFuncio', [ Arguments ] )
```

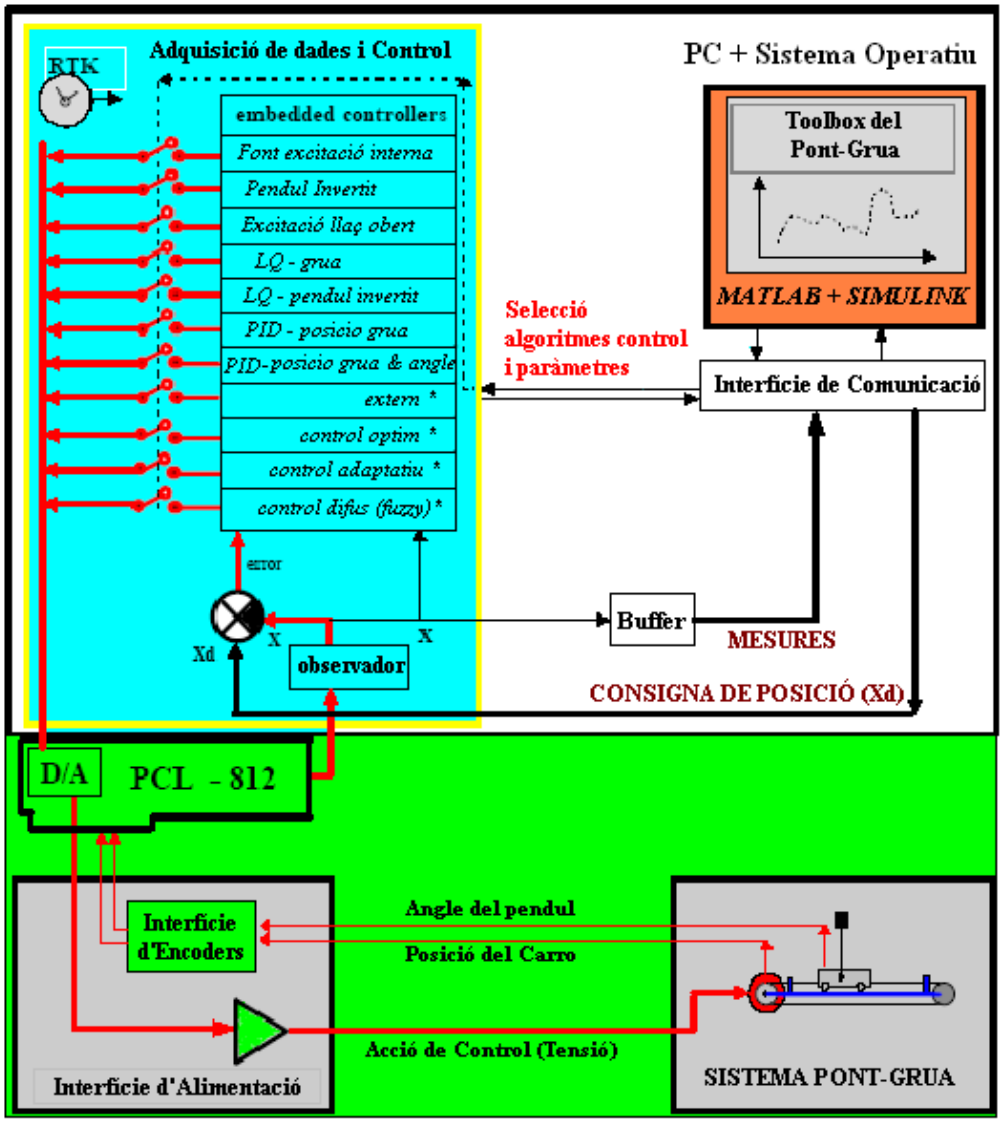
on:

*ValorRetorn* → Valor retornat per la funció corresponent en forma de variable de Matlab. (vector o matriu).

*Pd_call* → Nom de la llibreria de comunicació.

*'NomFuncio'* → Nom de la funció per cridar, en format "String".

*Arguments* → Arguments de la funció en forma de vector o matriu de Matlab. (opcional segons la funció)




* opcional segons versió

Les diverses funcions disponibles de la llibreria de comunicació son les que es descriuen a continuació:

| **NOM FUNCIÓ** | **TASCA** | **Valor retorn** | **Arguments** |
|---|---|---|---|
| GetAlgNo | Retorna el numero corresponent a l'algoritme de control actiu al RTK. | Numero de l'algoritme de control. (*3) | *cap* |
| GetBaseAddress | Retorna l'adreça base de la targeta de comunicació. | Numero de l'adreça base de la targeta. | *cap* |
| GetCartPosFilt | Retorna els paràmetres del filtre digital de la senyal de Posició del carro. | Matriu amb els paràmetres del filtre. (*1) | *cap* |
| GetCartSpeedFilt | Retorna els paràmetres del filtre digital de la senyal de Velocitat del carro. | Matriu amb els paràmetres del filtre. (*1) | *cap* |
| GetDesPosition | Retorna la consigna de posició del carro. | Valor de la consigna. | *cap* |
| GetDivider | Retorna el divisor del rellotge auxiliar del RTK | Valor del divisor. (*7) | *cap* |
| GetHistory | Retorna el contingut del Buffer en forma de matriu. | Matriu amb els valors del Buffer. (*2) | *cap* |
| GetNoOfSamples | Retorna el nombre de mostres del Buffer. | Numero de mostres. | *cap* |
| GetP | Retorna els paràmetres de control (diferents segons l'algoritme) | Vector amb els valors dels paràmetres. (*3) | *cap* |
| GetPendPosFilt | Retorna els paràmetres del filtre digital de la senyal de Posició del pèndul. | Matriu amb els paràmetres del filtre. (*1) | *cap* |
| GetPendSpeedFilt | Retorna els paràmetres del filtre digital de la senyal de Velocitat del pèndul. | Matriu amb els paràmetres del filtre. (*1) | *cap* |
| GetPW | Retorna els paràmetres de l'algoritme d'excitació en llaç obert. | Vector amb els valors dels paràmetres. (*4) | *cap* |
| GetSampleTime | Retorna el període de mostreig. | Valor del període en segons. | *cap* |
| LoadLibrary | Carrega la llibreria del RTK. | Comptador del mòdul. (*6) | *cap* |

| **NOM FUNCIÓ** | **TASCA** | **Valor retorn** | **Arguments** |
|---|---|---|---|
| ResetEncoder | Posa els comptadors de Posició i Angle a 0 | Valor d'èxit. (*5) | *cap* |
| ResetTime | Posa el comptador de temps a 0. | Valor d'èxit. (*5) | *cap* |
| SetAlgNo | Fixa l'algoritme de control actiu al RTK. | Valor d'èxit. (*5) | Numero de l'algoritme de control. (*3) |
| SetBaseAddress | Fixa l'adreça base de la targeta de comunicació. | Valor d'èxit. (*5) | Numero de l'adreça base de la targeta. |
| SetCartPosFilt | Fixa els paràmetres del filtre digital de la senyal de Posició del carro. | Valor d'èxit. (*5) | Matriu amb els paràmetres del filtre. (*1) |
| SetCartSpeedFilt | Fixa els paràmetres del filtre digital de la senyal de Velocitat del carro. | Valor d'èxit. (*5) | Matriu amb els paràmetres del filtre. (*1) |
| SetDesPosition | Fixa la consigna de posició del carro. | Valor d'èxit. (*5) | Valor de la consigna. |
| SetDivider | Fixa el divisor del rellotge auxiliar del RTK | Valor d'èxit. (*5) | Valor del divisor. (*7) |
| SetP | Fixa els paràmetres de control (diferents segons l'algoritme) | Valor d'èxit. (*5) | Vector amb els valors dels paràmetres. (*3) |
| SetPendPosFilt | Fixa els paràmetres del filtre digital de la senyal de Posició del pèndul. | Valor d'èxit. (*5) | Matriu amb els paràmetres del filtre. (*1) |
| SetPendSpeedFilt | Fixa els paràmetres del filtre digital de la senyal de Velocitat del pèndul. | Valor d'èxit. (*5) | Matriu amb els paràmetres del filtre. (*1) |
| SetPW | Fixa els paràmetres de l'algoritme d'excitació en llaç obert. | Valor d'èxit. (*5) | Vector amb els valors dels paràmetres. (*4) |
| SetSampleTime | Fixa el període de mostreig. | Valor d'èxit. (*5) | Valor del període en segons. |
| StartAcq | Esborra el contingut del Buffer | Valor d'èxit. (*5) | *cap* |
| StopPractical | Atura el carro (Acció de Control = 0) | Valor d'èxit. (*5) | *cap* |
| UnloadLibrary | Descarrega la llibreria del RTK. | Comptador del mòdul.(*6) | *cap* |

| NOM FUNCIÓ | TASCA | Valor retorn | Arguments |
|---|---|---|---|
| LoadExtAlg | Carrega el controlador extern implementat a la llibreria dinàmica indicada. | Valor d'èxit. (*5) | Nom de la llibreria (*.dll) del controlador amb la ruta del directori, en format "String". (*8) |

*Nota:*

- En Matlab, els valors numèrics son en format real d'alta precisió (equivalent al "double" de C o Java).
- En Matlab, els vectors o matrius son taules dels respectius valors.
- En Matlab, els "String" son vectors de caràcters en format ASCII.

A continuació es detallen els formats dels Arguments i valors de retorn de certes funcions.

**(*1). Funcions referents als filtres digitals de senyals:**

Les funcions *GetCartPosFilt, GetCartSpeedFilt, SetCartPosFilt, SetCartSpeedFilt, GetPendPosFilt, GetPendSpeedFilt, SetPendPosFilt* i *SetPendSpeedFilt* fan referència als filtres digitals implementats al RTK per filtrar diverses senyals de mesura de paràmetres amb la finalitat d'eliminar-ne el soroll.

Les diverses funcions fixen o retornen els paràmetres dels filtres de les senyals de Velocitat i Posició del Carro i del pèndul, respectivament. Els 4 filtres digitals del RTK representats en forma d'equació en diferències, tenen l'estructura:

$$y(t) = a_0 x(t) + \sum_{i=1}^{8} a_i x(t - iT_s) + \sum_{i=1}^{8} b_i y(t - iT_s)$$

on:

y(t) → Senyal de sortida del filtre.

x(t) → Senyal d'entrada al filtre.

$T_s$ → Període de mostreig.

$a_0..a_8$ , $b_1..b_8$ → Paràmetres del filtre.

Les matriu argument de les funcions que fixen els paràmetres dels filtres i les matriu de retorn de les que retornen els esmentats paràmetres, tenen el format següent:

$$\begin{matrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\ 0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \end{matrix}$$

**(*2). Funció *GetHistory*:**

Aquesta funció retorna una matriu que conté les mostres del Buffer del RTK des de la darrera vegada que s'hagi esborrat el contingut del mateix o s'hagi posat el comptador de temps a 0 i posteriorment n'esborra aquest contingut. El Buffer té una capacitat màxima de 1000 mostres de cada paràmetre. L'estructura de la matriu retornada és la següent:

| **Fila de la matriu** | **Paràmetre** | **Unitats** |
|---|---|---|
| 1 | Temps transcorregut | Segons |
| 2 | Angle del pèndul | Radiants |
| 3 | Velocitat angular del pèndul | Radiants/segon |
| 4 | Posició lineal del carro | Metres |
| 5 | Velocitat lineal del carro. | Metres/segon |
| 6 | Acció de Control. | Valor relatiu ($\pm 1$) |
| 7 | Consigna (posició del carro) | Metres |

**(*3). Funcions *SetAlgNo, GetAlgNo, SetP* i *GetP* :**

Les funcions *SetAlgNo* i *GetAlgNo* requereixen com a argument la primera i retorna la segona un numero corresponent a l'algoritme de control. La correspondència entre aquest numero i l'algoritme de control es pot veure a la següent taula:

| **Numero d'algoritme** | **Descripció** |
|---|---|
| 0 | CAP (equivalent a fixar acció de control a 0) |
| 1 | Excitació en llaç obert |
| 2 | LQ – Pèndul invertit |
| 3 | LQ – Mode grua |
| 4 | Control Fuzzy (lògica difusa) |
| 5 | PID |
| 6 # | Control òptim |
| 7 # | Control adaptatiu |
| 99 | Controlador extern (veure *8) |

# només disponible en algunes versions

Les funcions *SetP* i *GetP* fixen i retornen els paràmetres de l'algoritme de control en forma de vector. Els elements del vector tenen diferent significat segons el tipus d'algoritme tal i com es detalla a continuació:

| | **ALGORITME CONTROL** | | | | |
|---|---|---|---|---|---|
| **Element vector** | 0 | 1 | 2 | 3 | 5 |
| 1 | - | $\lvert x_{màx} \rvert$ | $\lvert U_{\min} \rvert$ | $\lvert U_{\min} \rvert$ | $\lvert U_{\min} \rvert$ |
| 2 | - | - | K1 | K1 | Kp |
| 3 | - | - | K2 | K2 | Ki |
| 4 | - | - | K3 | K3 | Kd |
| 5 | - | - | K4 | K4 | $\lvert x_{màx} \rvert$ |
| 6 | - | - | $\lvert x_{màx} \rvert$ | $\lvert x_{màx} \rvert$ | $\lvert U_{màx} \rvert$ |
| 7 | - | - | $\lvert U_{màx} \rvert$ | $\lvert U_{màx} \rvert$ | Saturació Integrador |
| 8 | - | - | Coeficient W | - | - |
| 9 | - | - | Valor R | - | - |
| 10 | - | - | Zona estabilització | - | - |

On:

$|x_{màx}|$ → Rang de desplaçament màxim del carro.

$|U_{\min}|$ → Valor mínim de l'Acció de Control (en valor absolut) per vèncer la fricció estàtica del carro.

$|U_{màx}|$ → Valor absolut màxim de l'acció de control.

Per l'algoritme 1 (Excitació en llaç obert) la resta dels paràmetres s'especifiquen amb la funció *SetPW* i s'obtenen amb *GetPW*.

**(*4). Funció *SetPW* i *GetPW*:**

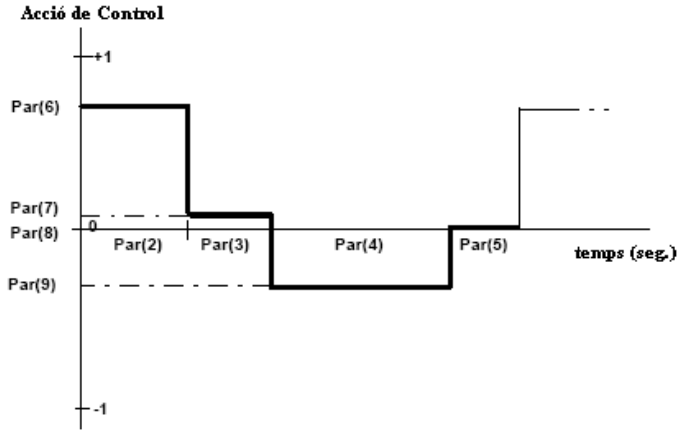
Aquestes funcions fixen i retornen els paràmetres de la font d'excitació interna, de forma anàloga a les funcions anteriors per al cas particular de l'algoritme d'excitació en llaç obert. El primer element del vector indica el tipus de senyal amb la qual es vol excitar el sistema.

| **1er element vector** | **Tipus de senyal** |
|---|---|
| 0 | Constant. |
| 1 | Senyal quadrada. |
| 2 | Senyal triangular. |
| 3 | Senyal sinusoïdal |
| 4 | Senyal periòdic amb valors aleatoris |

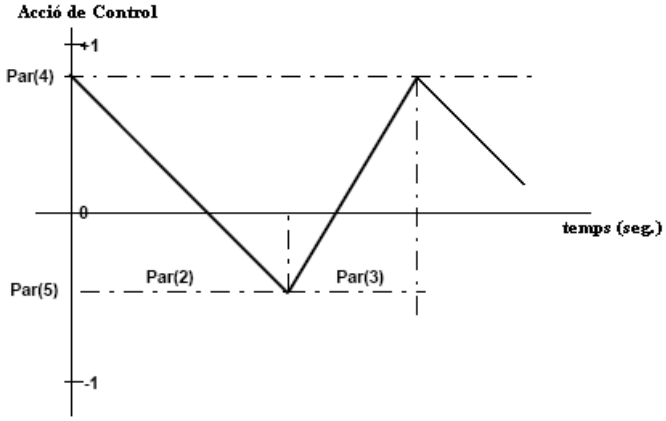
En el cas de senyal constant l'únic altre valor necessari és el valor de l'acció de control, que s'especifica al segon element del vector.

En els casos de la resta de tipus de senyals els diferents elements dels vectors tenen els següents significats:

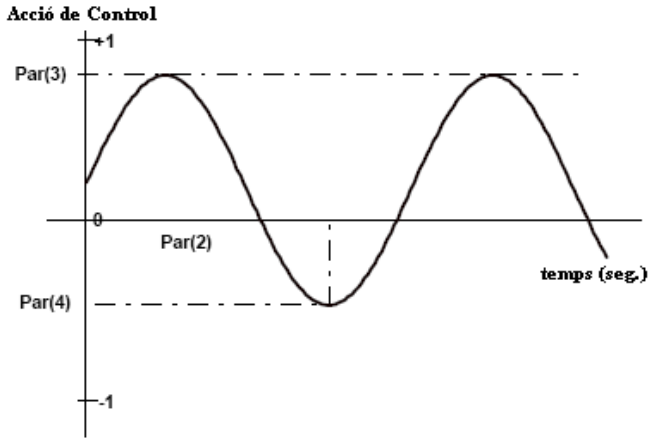
*Senyal quadrada:*



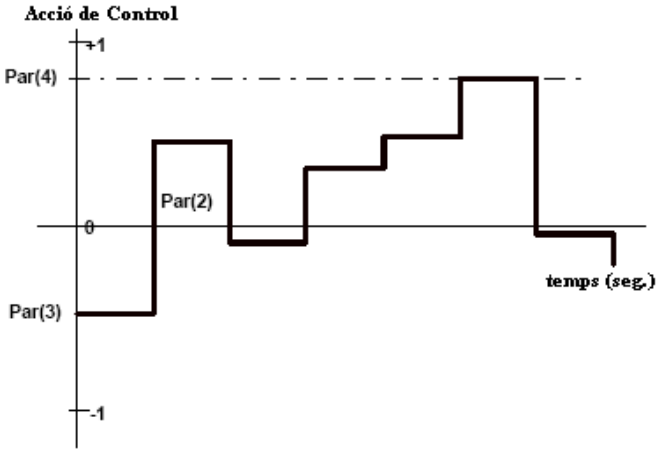
*Senyal triangular:*




*Senyal sinusoïdal:*




*Senyal periòdic amb valors aleatoris:*




on *Par(n)* representa l'element n del vector corresponent.

**(*5). Funcions amb valor d'èxit de retorn:**

Les funcions encarregades de fixar paràmetres del RTK així com algunes altres d'accions diverses retornen un numero per indicar si l'operació que havien de realitzar s'ha realitzat correctament. En cas afirmatiu retornen 1, i qualsevol altre valor en cas contrari.

**(*6). Funcions *LoadLibrary* i *UnloadLibrary* :**

Aquestes funcions carreguen i descarreguen el mòdul ***pd_dll.dll*** a la memòria. Aquesta acció és realitzada de totes maneres quan es crida per primer cop una funció de la llibreria ***pd_call.dll*** . El valor de retorn és el valor del comptador de referència del mòdul; és a dir, si el mòdul es carrega per primera vegada, el comptador val 1, el segon cop que es crida no es torna a carregar perquè ja ho està de carregat i aleshores el comptador passa a valer 2. De forma anàloga succeeix amb les descarregues.

**(*7). Funcions *SetDivider* i *GetDivider* :**

Aquestes funcions fixen i retornen el valor del rellotge auxiliar del RTK. Si aquest és igual a $k>1$, això vol dir que les mesures es prenen $k$ vegades més ràpidament que la freqüència de mostreig.

**(*8). Funció *LoadExtAlg* :**

Aquesta funció carrega a la memòria del RTK l'algoritme de Control Extern de la llibreria indicada. Per utilitzar-lo s'ha de seleccionar l'algoritme de control nº 99 i fixar els paràmetres com en els altres casos (la interpretació dels paràmetres depèn del disseny d'aquest algoritme). L'argument requerit és la ruta i el nom de la llibreria de l'algoritme de control extern en format "String", en la forma:

```
pd_call('LoadExtAlg','C:\directori\elMeuControladorExtern.dll');
```

### 3.1.3. MODEL I PARÀMETRES DEL SISTEMA.

El sistema té una entrada, l'acció de control **u**, i 4 variables de sortida mesurables que poden interessar controlar segons el cas:

- Posició del carro. (**$x_1$**)
- Velocitat del carro. (**$x_3$**)
- Posició angular del pèndul. (**$x_2$**)
- Velocitat angular del pèndul. (**$x_4$**)

Les equacions que relaciones aquestes 4 variables i l'entrada del sistema son les següents:

$$(m_c + m_p) * (x_1 - l * \sin(x_2))'' = F - T_c$$

$$(m_c + m_p) * (l * \cos(x_2))'' = V - (m_c + m_p) * g$$

$$J * x_2'' = (F - T_c) * l * \cos(x_2) + V * l * \sin(x_2) - D_p$$

on les sigles representen:

| Sigla | Paràmetre | Unitat |
|---|---|---|
| $m_p$ | Massa del pèndul | kg |
| $m_c$ | Massa del carro | kg |
| $l$ | Distància entre l'eix i el centre de gravetat del pèndul | m |
| $F$ | Força de control aplicada al carro | N |
| $T_c$ | Fricció motriu del carro | N |
| $V$ | Força de reacció que actua verticalment al carro | N |
| $J$ | Moment d'inèrcia del sistema respecte al centre de gravetat | kg * $m^2$ |
| $D_p$ | Fricció motriu del pèndul | N |
| $g$ | Acceleració de la gravetat | m / $s^2$ |

(.)' representa la primera derivada respecte al temps

(.)'' representa la segona derivada respecte al temps

La fricció del pèndul és proporcional a la velocitat angular: $D_p = f_p * x_4$

La fricció del carro es pot aproximar com: $T_c = f_c * x_3$

La força de control F depèn de l'acció de control: F = u * M , on M és la força (en valor absolut) màxima aplicable al carro.

A partir de les equacions i després d'eliminar la variable V s'obtenen les equacions d'estat:

$$x_1' = x_3$$

$$x_3' = \frac{a*(F - f_c*x_3 - \mu*x_4{}^2*\sin(x_2)) + l*\cos(x_2)*(\mu*g*\sin(x_2) - f_p*x_4)}{J + \mu*l*\sin^2(x_2)}$$

$$x_2' = x_4$$

$$x_4' = \frac{l*\cos(x_2)*(F - f_c*x_3 - \mu*x_4{}^2*\sin(x_2)) + \mu*g*\sin(x_2) - f_p*x_4}{J + \mu*l*\sin^2(x_2)}$$

on:

$$a = l^2 + \frac{J}{m_c + m_p} \quad , \quad \mu = (m_c + m_p)*l$$

Els valors dels paràmetres son subministrats pel fabricant, tot i que molts d'ells estan subjectes a ésser obtinguts per experimentació amb el sistema.

| **Paràmetre** | **Valor teòric** |
|---|---|
| Desplaçament límit $\mathbf{X_{1\ màx.}}$ | $\pm$ 0.5 m. |
| Acceleració gravetat **g** | 9.81 m/s$^2$. |
| Distància entre l'eix i el centre de gravetat del pèndul **l** | 0.017 m. |
| Massa del pèndul $\mathbf{m_p}$ | 0.11 kg. |
| Massa del carro $\mathbf{m_c}$ | 1.12 kg. |
| Moment d'inèrcia del sistema respecte al centre de gravetat **J** | 0.0136 kg*m$^2$. |
| Força de control màxima **M** | 17.0 N. |
| Coeficient fricció del carro $\mathbf{f_c}$ | 0.05 N*s/m. |
| Coeficient fricció del pèndul $\mathbf{f_p}$ | *Negligible* |

## 3.2. TARGETA PCL-812PG.

### 3.2.1. DESCRIPCIÓ.

La targeta d'adquisició PC-Labcard 812PG és una targeta per PC/XT/AT i compatibles. Permet capturar qualsevol tipus de senyal analògica d'entrada (amb les limitacions dels rangs màxims permesos), així com l'emissió d'un senyal de sortida i la lectura i escriptura de senyals digitals.

En aquest projecte la targeta PCL-812PG és la que s'utilitza per comunicar el PC amb el procés del Pèndol - Grua.

### 3.2.2. CARACTERISTIQUES TÈCNIQUES.

Les característiques tècniques de la targeta son les següents:

- 16 canals d'entrada analògics ("single-ended").
- Un convertidor standard industrial de 12 bits que realitza la conversió dels senyals analògics d'entrada per aproximacions successives. La freqüència màxima de mostreig és de 30Mhz en mode DMA.
- Rangs d'entrada analògics programables per software (Bipolar) :
  - +/- 12.5V
  - +/- 5V
  - +/- 2.5V
  - +/- 0.625V
  - +/- 0.3125V.
- Tres entrades analògiques amb els següents modes Trigger:
  - Software Trigger.
  - Trigger de Nivell (programable).
  - Trigger de Pols (programable).
- Possibilitat de transferència de dades A/D convertides per interrupció o DMA.
- Un temporitzador /comptador INTEL 8253-5.

- Dos canals de sortida D/A amb una resolució de 12 bits. Amb rang de sortida:

  - de 0 a +5V
  - de 0 a +10V.

- 16 entrades digitals compatibles TTL/DTL.
- 16 sortides digitals compatibles TTL/DTL.

### 3.2.3. ESPECIFICACIONS DEL PRODUCTE.

Les especificacions generals del producte son les següents:

- Entrades Analògiques (convertidor A/D )

  - Canals : 16 “single-ended”
  - Resolució: 12 bits
  - Rangs d’entrada: Bipolar :

    - +/- 12.5V
    - +/- 5V
    - +/- 2.5V
    - +/- 0.625V
    - +/- 0.3125V.

  - Sobre Voltatge: Continu +/- 30V màx.
  - Convertidor : HADC574Z.
  - Velocitat de Conversió : 30 Mhz. màx.
  - Precisió : 0.015 % llegint +/-1 bit
  - Lineal en : +/- 1 bit
  - Modes en Trigger : per software, per placa o extern.
  - Transferència de dades : Control per programa, per interrupció o DMA.

- Sortides Analògiques (Convertidor D/A)

  - Canals : 2
  - Resolució : 12 bits
  - Rang de sortida :

    - 0 a +5V
    - 0 a +10V.

  - Dispositius Analògics : AD7541AKN o equivalents.
  - Lineal en : +/-12 bits
  - Temps d'estabilització: 30 micro-segons.

- Entrades Digitals

  - Canal : 16 bits
  - Nivell : TTL compatible
  - Voltatge d'entrada :

    - Nivell Baix → 0.8V màx.
    - Nivell Alt → 2.0V mín.

- Sortides Digitals

  - Canal : 16 bits.
  - Nivell : TTL compatible.
  - Voltatge de sortida:

    - Nivell Baix → 8mA a 0.5V màx.
    - Nivell Alt → 0.05mA a 2.7V màx.

- Temporitzador/Comptador programable

  - Dispositiu : INTEL 8253
  - Comptadors : 3 canals de 16 bits.
  - Entrada, porta : TTL/DTL/CMOS
  - Base de Temps : 2MHz.

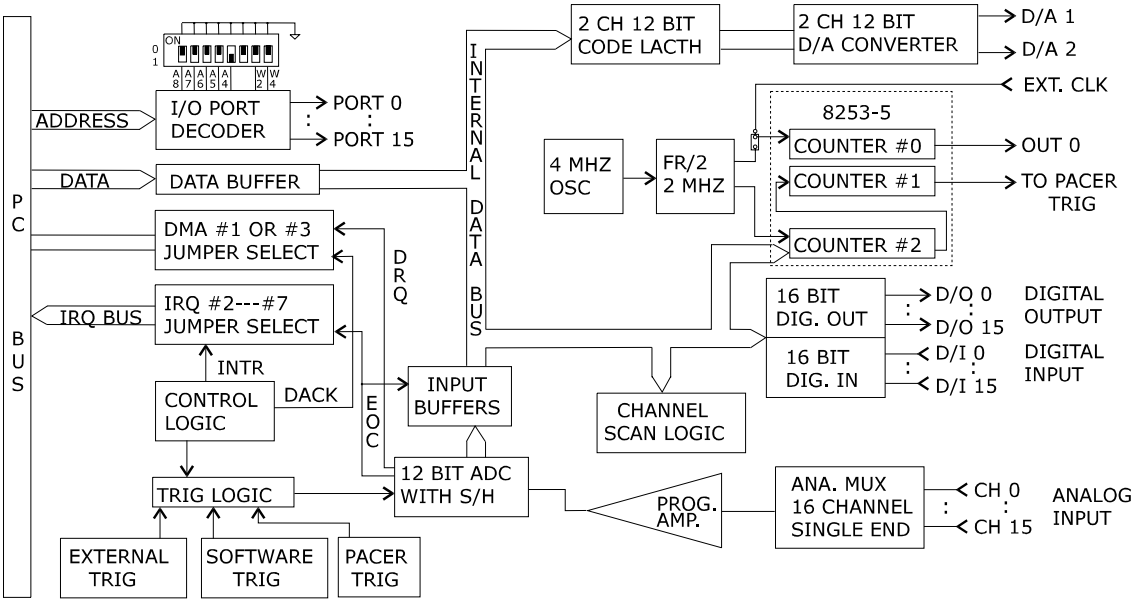
- Canal d'Interrupció

  - Nivell : IRQ 2 a 7, seleccionable mitjançant jumpers en placa.
  - Permès : Via S0, S1 y S2 del registre de CONTROL.

- Canal DMA

  - Nivell : 1 o 3, seleccionable mitjançant jumpers en placa
  - Permès : Via S0, S1 y S2 del registro de CONTROL.

## 3.2.4. DIAGRAMA DE BLOCS DE LA TARGETA PCL-812PG.



## 3.3. LLENGUATGE DE PROGRAMACIÓ JAVA.

No és pas objecte del present capítol explicar detalladament el llenguatge de programació Java, però donat que és condicionant per al desenvolupament del projecte l'ús en darrera instància d'aquest llenguatge, a continuació es fa una breu descripció de les característiques principals del llenguatge de programació i una descripció més detallada de les capacitats del llenguatge més rellevants en el desenvolupament del present projecte.

### 3.3.1. CARACTERÍSTIQUES GENERALS.

Java és un llenguatge de programació d'alt nivell, compilat i interpretat, que s'orienta a objectes, i que és multi - tasca, portable i segur. El llenguatge va ser dissenyat per l'empresa Sun MicroSystems l'any 1991 i des de llavors s'han anat ampliant les seves capacitats.

Les aplicacions implementades en el llenguatge Java es desenvolupen en el marc d'un entorn anomenat La Plataforma Java (the Java Platform). Aquest entorn, que té dos components, la màquina virtual i l'API (Application Programming Interface), garanteix la completa independència dels programes Java, tant de l'arquitectura del maquinari com del sistema operatiu de la màquina en la qual s'executin.

Java és un llenguatge que deriva directament de C i de C++, llenguatge aquest darrer del qual hereta la majoria de les paraules reservades, les estructures de control i la sintaxi, però a diferència d'aquests, que són compilats, o de l'antic BASIC, que era interpretat, Java és compilat i interpretat simultàniament.

Quan una persona desenvolupa una aplicació en un llenguatge compilat com ara C o C++, l'arxiu binari generat pel compilador i que conte el codi que implementa l'esmentada aplicació, només pot ésser executat sobre la plataforma sobre la qual s'ha desenvolupat, degut a que el codi es específic per aquesta plataforma.

La plataforma Java es troba per sobre d'altres plataformes. El codi que generen els seus compiladors no es específic d'una màquina física en particular, si no d'una màquina virtual, la Màquina Virtual Java (JVM). Encara quan existeixen múltiples implantacions de la JVM, cadascuna específica de la

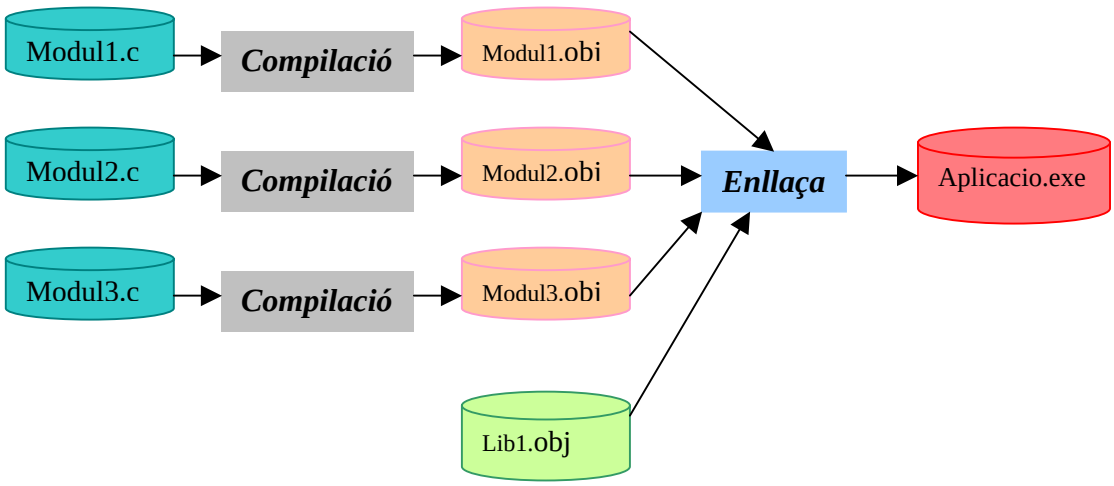
plataforma sobre la qual es sustenta, existeix una única especificació de la JVM, que proporciona una vista independent del hardware y del sistema operatiu sobre el que s' estigui treballant. D'aquesta manera un programador en Java escriu el seu programa una vegada, el compila i l'executa a qualsevol màquina.

Es precisament la JVM la clau de la independència dels programes escrits en Java, sobre el sistema operatiu i el hardware en el que s'executen, ja que es la encarregada de proporcionar la vista d'un nivell d'abstracció superior, on a més a més de la independència de la plataforma anteriorment mencionada, presenta un llenguatge de programació simple, orientat a objectes, amb verificació estricta de tipus de dades, múltiples fils, amb enllaçat dinàmic i amb recol·lecció automàtica d'escombriaires.

### 3.3.2. DISSENY D'APLICACIONS EN JAVA.

El disseny d'una aplicació en llenguatge Java difereix lleugerament del procés que es segueix en el cas de llenguatges de programació compilats com C,C++, FORTRAN,... Per tal d'il·lustrar-ho compararem el procés seguit amb el llenguatge C i el seguit amb Java.

Un procés de disseny d'una aplicació en C o C++ segueix el següent esquema de seqüències:

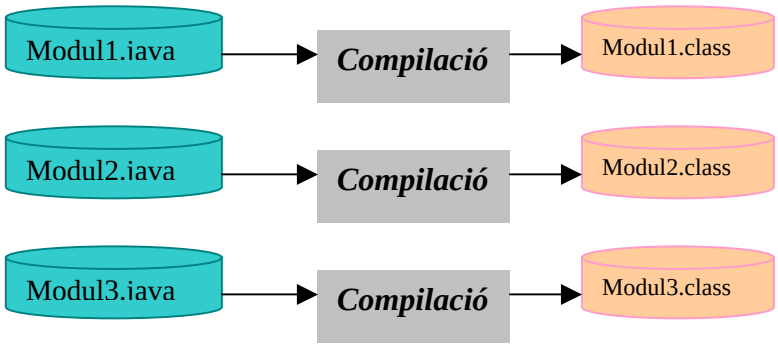


on:

- *Mòdul***n.***c* → son els codis font de les classes i aplicacions implementades.
- *Mòdul***n.***obj* → son els codis compilats dels respectius codis font.
- *Prog.exe* → és el codi executable.
- *Lib***n.***h* → son les llibreries amb implementació de classes i/o mètodes emprats directa o indirectament per l'aplicació principal.

El programa de Compilació del respectiu llenguatge (C en aquest cas), transforma el codi font en codi compilat. Posteriorment els codis compilats de l'aplicació principal i de les classes utilitzades per aquesta son enllaçats per generar el codi executable per la màquina. Un cop realitzat el procés l'únic element necessari per l'execució de l'aplicació és el fitxer *Prog.exe*. Però aquest codi només podrà ésser executat sobre una plataforma (Sistema Operatiu) igual a la que sobre la qual s'ha realitzat el procés anterior.

En canvi, un procés de disseny d'una aplicació en Java segueix el següent esquema de seqüències:




on:

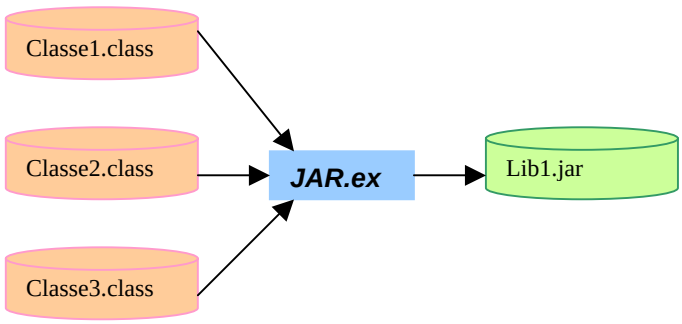
- *Mòdul***n.***java* → son els codis font de les classes i aplicacions implementades.
- *Mòdul***n.***class* → son els codis compilats dels respectius codis font.

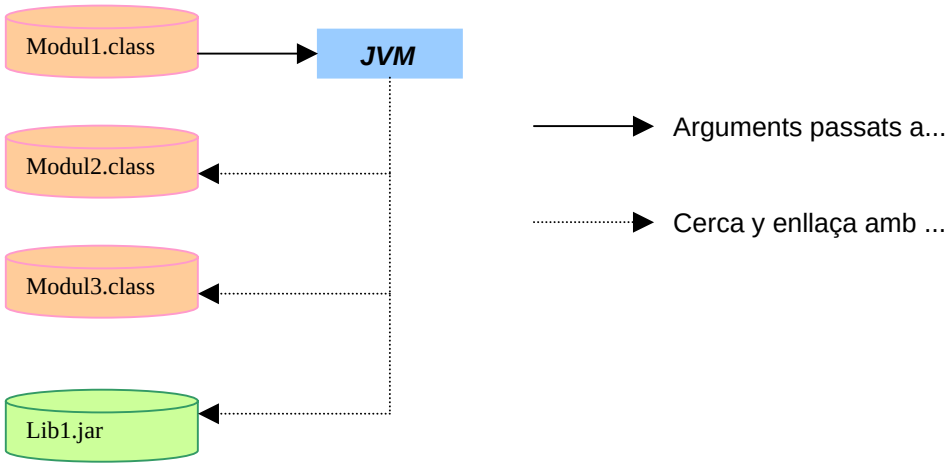
El procés comença com a qualsevol llenguatge: Es comença per escriure els fitxers font (que han de tenir l'extensió *.java) amb el codi desitjat. A continuació, es compilen aquests fitxers (el compilador es diu **javac.exe**) i s'obtenen els fitxers compilats amb l'extensió *.class, els quals ja poden ser distribuïts. No hi ha, doncs, enllaçat (linkage). Ara bé, aquests fitxers *.class, resultat de la compilació, no són pas codi màquina ni poden ser entesos per cap sistema operatiu; contenen "bytecodes": una mena de codi de baix nivell que serà interpretat per la JVM (**java.exe**); aquesta haurà de ser instal·lada a la màquina "real" en la qual es vulgui executar l'aplicació continguda als fitxers *.class.

Això fa que el codi de "bytecodes" a distribuir sigui independent del maquinari i del sistema operatiu de la màquina en la qual calgui executar-lo. Evidentment, qui executa el codi (la Màquina Virtual de Java), sí que ho és de dependent.

Aleshores per executar l'aplicació compilada, només cal cridar la JVM tot passant-li com a argument el fitxer compilat (*.class) que conté l'aplicació principal. La JVM ja s'encarrega de cercar al directori de treball i als directoris de classes (directoris configurats per a que la JVM hi cerqui les classes que s'utilitzaran ) les classes compilades que requereix l'aplicació i realitzar una mena de procés d'enllaçat en temps d'execució de l'aplicació.

Sovint, si el nombre de fitxers *.class és gran, es comprimeixen tots en un únic fitxer (*.jar) i es distribueixen així. La JVM és capaç de llegir directament d'aquests fitxers *.jar.






Per tal de fer funcionar una aplicació en Java només cal que en la màquina en la qual es vulgui executar hi hagi instal·lat el ***Java Runtime Enviroment*** (JRE), que consisteix bàsicament en la JVM, classes compilades que Sun MicroSystems posa a disposició dels programadors per la realització de les aplicacions, i algunes aplicacions addicionals necessàries a l'hora d'executar aplicacions més complexes; En la distribució de les aplicacions s'haurà de tenir en compte que la versió del JRE necessària per executar l'aplicació ha d'ésser igual o superior a la del JDK (conjunt de classes i aplicacions emprats en la implementació i compilació, com el compilador **javac.exe**) emprat per la seva implementació i compilació; per al desenvolupament del present projecte s'utilitzarà la versió 1.3.1 del JDK. Cal destacar que el JRE no posseeix cap compilador ni eina de desenvolupament; aquesta tasca ja ha estat realitzada.

### 3.3.3. EXCEPCIÓNS.

Java disposa d'un element particular, present a gairebé tots els llenguatges més moderns: les excepcions. Com tos els elements de Java, una excepció és un objecte que realitza la tasca d'esperar que es produeixin errors o condicions estranyes en el programa. Si passa alguna eventualitat l'excepció avisa i ajuda a reconduir el problema tot intentant recuperar el funcionament normal de l'aplicació. Tot objecte excepció deriva de la classe *Exception* del paquet de classes *java.lang* . El funcionament és força senzill: el programador protegeix un bloc de codi amb la declaració d'una excepció. Si en l'execució

d'aquell tros de programa es produeix un error, l'objecte excepció el captura i fa fluir el programa cap un fragment de codi on s'intentarà corregir l'esmentat error. Un cop corregit, intentarem retornar on estàvem i continuar amb normalitat.

En el cas que no es protegeixi el codi amb declaracions d'excepcions, el programa sempre intentarà trobar un gestor d'excepcions i si no el troba al lloc on s'ha produït, el buscarà al lloc des d'on hem cridat el mètode o la classe i així anirà pujant fins arribar a la Màquina Virtual de Java, la qual es limitarà a tancar el programa. Davant d'una excepció es pot actuar bàsicament de dues formes:

- Atacar immediatament el problema i solucionar-lo quan es presenta, utilitzant un bloc *try-catch*.

- Passar el problema al mètode que crida al codi que falla, tot indicant-li que s'ha produït una excepció. En aquest cas, serà el mètode qui crida a qui ha de solucionar els problemes. Si es segueix aquesta segona estratègia s'ha de fer una d'aquestes dues coses:

    - Llençar l'excepció mitjançant la clàusula *throw*.

    - No preveure res i definir el mètode com a lloc potencial d'errors de diferents categories, tot posposant a la definició del mètode, la clàusula *throws <Un_Tipus_d_Excepcio>* .

El programador també, pot crear excepcions pròpies; en aquest cas només s'ha d'estendre la classe *java.lang.Exception*.

En general, les excepcions de Java porten associats uns "String", amb els missatges descriptius del tipus d'error produït així com les possibles causes, de tal forma que qui gestioni l'excepció pugui actuar en conseqüència.

## 3.3.4. JAVA NATIVE INTERFACE (JNI).

### 3.3.4.1. *Introducció.*

Tal i com s'ha mencionat anteriorment, Java és un llenguatge de programació independent de la plataforma sobre la que s'executen les aplicacions en ell implementades. Aquesta independència de plataforma és deguda, en part, a que Java només contempla els aspectes estàndards de les diferents plataformes, la qual cosa impossibilita la implementació d'aplicacions on sigui necessària la utilització de característiques dependents de la plataforma, com per exemple l'accés a Hardware a baix nivell o la utilització de recursos compartits, com llibreries d'enllaç dinàmic (*.DLL, en el cas de Windows) .

Per solucionar aquest problema Sun ofereix la Interfície Nativa de Java o *Java Native Interface* (JNI), que s'inclou a les diverses versions del JDK.

JNI és la interfície de programació del llenguatge Java per executar codi natiu, es a dir codi compilat al llenguatge binari (llenguatge màquina) propi d'una plataforma o sistema. A les diverses versions del JDK s'inclouen les eines necessàries per la seva utilització. El JNI permet al codi escrit en Java que s'executa a través de la JVM interactuar amb aplicacions i llibreries escrites en altres llenguatges, com ara C o C++ o fins i tot llenguatge Ensamblador, tot i que les eines subministrades per les diferents versions del JDK estan orientats a que l'esmentada escriptura de codi es realitzi en el llenguatge C o C++.

JNI també incorpora les eines per executar codi escrit en Java des d'aplicacions escrites en altres llenguatges.

### 3.3.4.2. *Implementació de codi per mètodes natius.*

Per tal d'implementar codi font per mètodes natius d'una classe en Java s'ha de seguir següent procediment.

A l'hora d'editar la classe que declara els mètodes natius, aquests s'han de declarar com a natius. Aquesta declaració es realitza afegint a la classe de l'objecte retornat per la funció el modificador *native*. Quan, en temps d'execució, s'invoqui un mètode natiu, la JVM "cedirà" el control del

programa al sistema operatiu de la plataforma sobre la qual s'estigui executant i aquest invocarà el mètode, el codi executable del qual ha d'estar carregat en memòria. Per tal que el codi executable dels mètodes natius d'una determinada classe estigui carregat en memòria mentre aquesta classe estigui en servei, la classe ha d'invocar en el seu inicialitzador de classe o estàtic el mètode estàtic *loadLibrary* de la classe *System*, que pertany al paquet *java.lang*, passant com a argument el nom de la Llibreria (fitxer *.dll en el cas de Windows) on es troba el codi dels mètodes natius. Aquesta llibreria ha d'estar ubicada al directori de treball o als directoris en els quals el sistema operatiu cerca els executables (**path**). En el moment de l'edició de la classe és de suposar que aquesta llibreria no existeix encara, i per això és important que el nom de la llibreria es tingui en compte a l'hora de generar-la en el procés que es descriu a continuació.

En els mètodes natius, els paràmetres que són de tipus primitius es passen per copia, i la resta de tipus per referència. Tots els objectes passats a un mètode natiu són tractats com locals de forma que s'alliberen al sortir del mètode.

Després de l'edició de la classe, es procedeix a la seva compilació, com si d'una classe habitual de Java es tractés. Per fer aquesta operació s'executa **java.exe**, amb el fitxer font de la classe com a argument, amb la qual cosa s'obté un fitxer amb extensió *.*class*.

Posteriorment es genera un fitxer de capçalera en llenguatge C amb l'eina de desenvolupament **javah.exe** indicant l'opció **–jni** i passant com a argument el fitxer de classe compilat (*.*class*). El resultat és un fitxer de capçalera en llenguatge C (fitxer *.*h*), amb els prototipus dels mètodes natius amb l'estil JNI.

Una vegada que s'obté el fitxer que conté els prototipus, ja es pot implementar el codi dels mètodes natius en llenguatge C i fer la compilació per tal d'obtenir la llibreria d'enllaç dinàmic. Aquest últim pas es pot fer amb un entorn de desenvolupament de C com per exemple **Visual C++.** Per últim s'ha de tenir en compte que la llibreria generada ha de tenir el nom especificat com a argument al mètode de càrrega de la llibreria invocat a l'inicialitzador de la classe de Java.

#### 3.3.4.3. *Correspondència entre tipus Java i tipus C.*

Els diferents tipus primitius de variables i classes Java, tenen la següent correspondència en codi C.

| Tipus Java | Tipus C | Signatura |
|---|---|---|
| void | void | - |
| boolean | jboolean | Z |
| byte | jbyte | B |
| char | jchar | C |
| short | jshort | S |
| int | jint | I |
| long | jlong | J |
| float | jfloat | F |
| double | jdouble | D |
| String | jstring | - |
| Exception | jthrowable | - |
| Object | jobject | - |
| Class | jclass | - |

Els Arrays d'elements estan simbolitzats per <*tipus*>*Array* , on *tipus* es refereix al tipus dels elements de l'Array, com per exemple *jdoubleArray* .

Les signatures serveixen per indicar el tipus de variable, amb els tipus primitius, quan es pretén consultar o modificar valors des de mètodes natius d'atributs d'objectes Java.

#### 3.3.4.4. *Exemple: Classe ProvaJNI.*

Com a exemple de tot això es detalla la següent classe.

La classe anomenada *ProvaJNI* , té un atribut privat de tipus enter , 1 mètode públic que retorna el valor de l'atribut i 4 mètodes natius, també públics endemés del constructor. El funcionament és molt simple; els mètodes natius sumen, resten, calculen el producte i el quocient de dos enters passats com a arguments, assignen el resultat a l'atribut privat de la classe i retornen el resultat. En el cas del càlcul de quocient, es llença una

excepció anomenada *ExcepcioJNI* en cas de ser el segon argument del mètode (denominador) 0. Aquestes funcions poden ser realitzades des de Java directament, però la seva trivialitat il·lustra molt bé l'estructura de disseny d'una classe amb mètodes natius. El codi font Java de la classe és el següent:

```
class ProvaJNI
{
   private int resultat;

   public ProvaJNI()
   {
         resultat=0;
   }
   public native int suma(int a,int b);
   public native int resta(int a,int b);
   public native int producte(int a,int b);
   public native int quocient(int a,int b)throws ExcepcioJNI;

   public int getResultat()
   {
         return resultat;
   }

   static
   {
         System.loadLibrary("ProvaJNI");
   }
}
```

Com es pot observar els mètodes natius son declarats amb l'antecedent *native* i l'inicialitzador de la classe carrega la llibreria *ProvaJNI* , que en el cas de sistema operatiu Windows aquesta tindrà l'extensió *ProvaJNI.dll*.

Abans d'implementar el codi font dels mètodes natius es procedeix a compilar la classe; primer gravem l'arxiu com a *ProvaJNI.java* i posteriorment teclegem des del directori on estigui ubicat el fitxer font:

```
javac ProvaJNI.java
```

S'observa que la classe es compila correctament tot i no haver-se implementat el codi d'alguns mètodes. Per que la classe funcioni

correctament el següent pas consisteix en generar l'arxiu amb les capçaleres dels mètodes natius amb l'aplicació **javah.exe** ; Des de la línia de comandes teclegem:

```
javah –jni ProvaJNI
```

Al mateix directori s'haurà creat un fitxer de capçalera del llenguatge C , anomenat *ProvaJNI.h* . Aquest fitxer conté el següent codi:

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class ProvaJNI */

#ifndef _Included_ProvaJNI
#define _Included_ProvaJNI
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     ProvaJNI
 * Method:    suma
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_ProvaJNI_suma
  (JNIEnv *, jobject, jint, jint);

/*
 * Class:     ProvaJNI
 * Method:    resta
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_ProvaJNI_resta
  (JNIEnv *, jobject, jint, jint);

/*
 * Class:     ProvaJNI
 * Method:    producte
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_ProvaJNI_producte
  (JNIEnv *, jobject, jint, jint);

/*
 * Class:     ProvaJNI
 * Method:    quocient
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_ProvaJNI_quocient
  (JNIEnv *, jobject, jint, jint);

#ifdef __cplusplus
}
#endif
#endif
```

Dins d'aquest fitxer es troben les capçaleres en codi C dels mètodes natius, com per exemple del mètode *suma* :

```
JNIEXPORT jint JNICALL Java_ProvaJNI_suma
  (JNIEnv *, jobject, jint, jint);
```

Els descriptors `JNIEXPORT` i `JNICALL` son necessaris perquè la llibreria dinàmica que es creí sàpiga quines propietats tenen les funcions en el marc d'una DLL. En aquest cas son funcions exportables de JNI (exportable vol dir que poden ser cridades externament a la DLL) i que es cridaran a través de JNI. Entre aquests dos descriptors hi figura el tipus de variable retornada per la funció d'acord amb les equivalències de tipus entre Java i JNI en C. Posteriorment apareix el nom del mètode en C amb el format de JNI; aquest és `Java_`*`nomDeLaClasseJava_nomDelMetode`* . Finalment, entre parèntesis i separats per comes hi ha els tipus dels arguments del mètode en el mateix ordre que en Java, precedits per dos arguments addicionals que son objectes de les classes `JNIEnv*` i `jobject` . `JNIEnv*` és un punter a l'estructura JNINativeInterface que conté els punters a les funcions del JNI; aquest argument sempre figura i serveix per comunicar-se entre codi natiu i codi Java. `jobject` és una referència a l'objecte que es passa com a paràmetre implícit; en cas de ser el mètode natiu un mètode de classe (estàtic), aquesta referència seria a un objecte del tipus `jclass`.

El penúltim pas és l'implementació del codi dels mètodes. Per realitzar aquesta tasca, en cas de fer servir **Visual C++** , o algun altre entorn de programació de C o C++ s'han de seleccionar les opcions que correspongui per tal de generar una llibreria d'enllaç dinàmic per Win32. Segons l'opció seleccionada el programa ja crea alguns dels fitxers necessaris per generar-la, però en cas contrari s'ha de generar un fitxer que contingui la funció *DLLMain* o bé incloure-la en el fitxer on s'implementen els mètodes propis. El fitxer amb l'implementació dels mètodes (*ProvaJNI.cpp* en aquest cas) tindrà el següent aspecte:

```
#include "ProvaJNI.h"

JNIEXPORT jint JNICALL Java_ProvaJNI_suma
  (JNIEnv *env, jobject obj, jint par1, jint par2){

    jint resultat=par1+par2;
    jclass classe;
    jfieldID ID_atribut_resultat;
    classe=(*env).GetObjectClass(obj);
ID_atribut_resultat=(*env).GetFieldID(classe,"resultat","I");
    (*env).SetIntField(obj,ID_atribut_resultat,resultat);

     return resultat;
}

JNIEXPORT jint JNICALL Java_ProvaJNI_resta
  (JNIEnv *env, jobject obj, jint par1, jint par2){

    jint resultat=par1-par2;
    jclass classe;
    jfieldID ID_atribut_resultat;
    classe=(*env).GetObjectClass(obj);
ID_atribut_resultat=(*env).GetFieldID(classe,"resultat","I");
    (*env).SetIntField(obj,ID_atribut_resultat,resultat);

    return resultat;
}

JNIEXPORT jint JNICALL Java_ProvaJNI_producte
  (JNIEnv *env, jobject obj, jint par1, jint par2){

    jint resultat=par1*par2;
    jclass classe;
    jfieldID ID_atribut_resultat;
    classe=(*env).GetObjectClass(obj);
ID_atribut_resultat=(*env).GetFieldID(classe,"resultat","I");
    (*env).SetIntField(obj,ID_atribut_resultat,resultat);

    return resultat;
}

JNIEXPORT jint JNICALL Java_ProvaJNI_quocient
  (JNIEnv *env, jobject obj, jint num, jint den){

    if(den==0)
    {
     jclass classeExcepcio=(*env).FindClass("ExcepcioJNI");
        char                                  missatgeError[80]=
   "S'ha intentat dividir per zero!!";
        (*env).ThrowNew(classeExcepcio,missatgeError);
        return 0;
    }
    jint resultat=num/den;
    jclass classe;
    jfieldID ID_atribut_resultat;
    classe=(*env).GetObjectClass(obj);
ID_atribut_resultat=(*env).GetFieldID(classe,"resultat","I");
    (*env).SetIntField(obj,ID_atribut_resultat,resultat);

    return resultat;
}
```

En aquest codi, a més a més d'assignar noms als arguments del mètode, hi ha les instruccions que realitzen les operacions, calculen els resultats i els retornen d'acord amb els tipus corresponents. Endemés s'assigna a l'atribut privat de l'objecte Java el resultat amb les instruccions següents:

```
    jclass classe;
    jfieldID ID_atribut_resultat;
    classe=(*env).GetObjectClass(obj);

ID_atribut_resultat=(*env).GetFieldID(classe,"resultat","I");
    (*env).SetIntField(obj,ID_atribut_resultat,resultat);
```

La primera d'elles crea una instància a un objecte de la classe `jclass` , que representa una classe de Java (en aquest cas la classe de l'objecte sobre el que cridem el mètode, però pot ser qualsevol altra classe Java). La segona crea una instància a un objecte de la classe `jfieldID` , que consisteix en l'identificador en JNI dels atributs (variables) dels objectes Java. Les següents instruccions

```
classe=(*env).GetObjectClass(obj);
ID_atribut_resultat=(*env).GetFieldID(classe,"resultat","I");
```

creen els objectes corresponents a través de mètodes de la classe *JNIenv*, passant com a argument implícit l'objecte JNIenv cap al que apunta el primer argument del mètode natiu. En el cas de la classe de l'objecte l'únic argument explícit és el propi objecte mentre que per obtenir l'identificador de l'atribut cal la classe de Java, el nom de l'atribut (en format "String") i la firma corresponent al tipus Java de l'atribut. Finalment la instrucció:

```
(*env).SetIntField(obj,ID_atribut_resultat,resultat);
```

assigna a l'atribut de l'objecte el valor designat.

En el cas del mètode *quocient* es comprova que el denominador sigui diferent de 0 i en cas contrari es llença una excepció amb les instruccions:

```
jclass classeExcepcio=(*env).FindClass("ExcepcioJNI");
        char                                  missatgeError[80]=
   "S'ha intentat dividir per zero!!";
(*env).ThrowNew(classeExcepcio,missatgeError);
```

La primera d'elles cerca la classe Java de l'excepció; aquesta cerca la realitza entre les classes disponibles per la classe de la funció nativa (és a dir que estigui en el mateix Paquet o en un Paquet importat). La segona crea l' "String" amb el missatge d'error i la darrera crea i llença l'excepció.

Finalment ja només manca compilar i generar la llibreria dinàmica. Abans de fer-ho s'ha de recordar d'incloure el fitxer de capçalera dels mètodes natius de la classe al fitxer amb l'implementació del codi, incloure en aquest el fitxer de capçalera *jni.h* i configurar el compilador de C perquè inclogui en els directoris de cerca de fitxers de capçalera els directoris del JDK on hi ha ubicats els arxius de capçalera de codi natiu com *jni.h*. També s'ha de recordar de donar a la llibreria generada el nom especificat com a argument al mètode de càrrega de la llibreria invocat a l'inicialitzador de la classe de Java, en aquest cas *ProvaJNI.dll*.

Per realitzar aquesta tasca des de l'entorn **visual C++** el procediment és el següent. En primer lloc s'escull l'opció *Win32 Dynamic-Link Library*, a la qual es pot accedir amb el desplegament de l'opció *New* dins el menú *File*. Després s'obté una finestra on s'ha d'indicar el nom del projecte i la seva localització. A continuació s'escull l'opció *Basic DLL project* , amb la qual cosa es creen un seguit de fitxers necessaris per la construcció de la DLL. Al projecte creat s'hi ha d'afegir els arxius de codi font i de capçalera C que hem implementat i procedir a la compilació i enllaç.

Un cop realitzades totes les tasques s'obté l'arxiu *ProvaJNI.dll* . L'únic requisit perquè la classe Java implementada funcioni és que al directori de treball o en un dels directoris de cerca d'executables del sistema operatiu (***path***) es trobi ubicada la llibreria dinàmica, amb precaució per casos de polimorfisme.

# 3.4. TECNOLOGIA IDL I ARQUITECTURA CORBA.

## 3.4.1. INTRODUCCIÓ.

IDL és una tecnologia per objectes distribuïts, és a dir, objectes interactuant sobre diferents plataformes a través d'una xarxa. La tecnologia IDL permet interactuar als diversos objectes independentment del seu llenguatge d'implementació. Això és possible perquè IDL està basada en l'arquitectura CORBA (Common Object Request Brokerage Architecture), un model industrial estàndard per objectes distribuïts.

Les sigles IDL també fan referència al llenguatge de definició d'interfície (Interface Definition Language). Aquest llenguatge és un llenguatge de sintaxi semblant a ANSI C, amb el qual es defineixen les interfícies remotes, és a dir els mètodes d'un objecte que podran ésser invocats remotament. Tots els llenguatges d'alt nivell que suporten CORBA tenen la seva pròpia equivalència entre IDL i el respectiu llenguatge. Aquestes equivalències i el conjunt del disseny de l'arquitectura CORBA son tasques d'un consorci industrial anomenat OMG (Object Management Group) .

## 3.4.2. IDL I JAVA.

El llenguatge Java és un dels que suporta IDL; l'empresa dissenyadora del llenguatge, Sun MicroSystems, és membre del consorci OMG.

El llenguatge IDL és un llenguatge de tipus declaratiu i a diferència dels llenguatges d'alt nivell habituals, no és un llenguatge ni interpretat ni compilat, sinó "mapejat". Això vol dir que el llenguatge simplement defineix (declara) una interfície (conjunt de mètodes d'una o diverses classes), els mètodes de la qual poden ser invocats en una màquina diferent a la màquina on resideix l'objecte de la classe. A partir d'aquesta definició d'interfície remota, el "mapeig" o traducció entre la interfície definida en llenguatge IDL i les corresponents classes Java és realitzada per una aplicació anomenada Compilador IDL a Java (**idlj.exe**), inclosa al JDK 1.3.1. Les equivalències de sintaxi més significatives entre els dos llenguatges es llisten tot seguit:

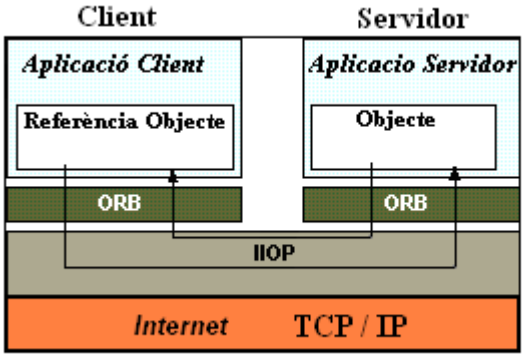
| **IDL** | **Java** |
|---|---|
| module | package |
| boolean | boolean |
| char, wchar | char |
| octet | byte |
| string, wstring | java.lang.String |
| short, unsigned short | short |
| long, unsigned long | int |
| long long, unsigned long long | long |
| float | float |
| double | double |
| fixed | java.math.BigDecimal |
| enum, struct, union | class |
| sequence, array | array |
| exception | exception |
| const | final |
| interface | interface |

En el cas dels Arrays de diversos elements d'un tipus la sintaxi és *sequence<Tipus_elements>*.

Per tal de suportar la interacció entre objectes de programes diferents, Sun MicroSystems proporciona l'anomenat ORB (Object Request Broker), que consisteix en una llibreria de classes que permet una comunicació de baix nivell entre aplicacions implementades en Java i aplicacions implementades en algun altre llenguatge compatible amb CORBA o en el propi Java.

### 3.4.3. LA COMUNICACIÓ AMB CORBA.

En l'arquitectura CORBA, tant la invocació de mètodes d'objectes remots com la recepció d'objectes de retorn dels mateixos es processa mitjançant el protocol compartit IIOP (Internet Inter-ORB Protocol), que està basat en el protocol estàndard d'Internet (TCP/IP). La següent figura mostra com un objecte distribuït és compartit entre un client CORBA i un servidor.




Aquest protocol és totalment transparent per a l'implementador tant de l'objecte distribuït com de les aplicacions que en requereixen servei.

### 3.4.4. DISSENY D'APLICACIONS AMB OBJECTES DISTRIBUÏTS MITJANÇANT IDL EN JAVA.

Per al disseny d'aplicació amb objectes distribuïts s'han de distingir dos possibles tasques:

- Disseny i implementació de l'objecte distribuït.
- Disseny de l'aplicació que usa objectes distribuïts prèviament implementats.

*Disseny i implementació de l'objecte distribuït (servidor).*

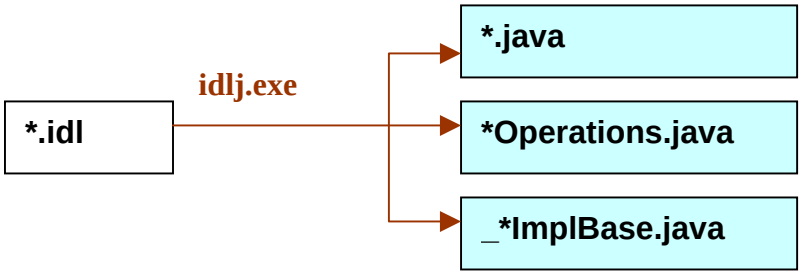
En aquest cas la primera tasca a realitzar és la definició de la interfície remota, és a dir, els mètodes que estaran disponibles per a ésser invocats remotament. Aquesta definició es fa en llenguatge IDL. En cas de no tenir cap entorn de desenvolupament integrat d' IDL es pot realitzar amb un editor de text ASCII, i es genera un arxiu de text amb extensió *.idl.

Posteriorment aquest codi IDL es "mapeja" amb el compilador **idlj** , disponible al JDK 1.3.1. Es crida l'aplicació des del directori on interessi que es generin els arxius Java resultants del "mapeig", passant com a arguments un "String" amb les opcions de compilació i el fitxer *.idl. Aquestes opcions son les següents:

- "-fall" → Quan es vol implementar les dues bandes.
- "-fclient" → Quan es vol implementar la banda del client.
- "-fserver" → Quan es vol implementar la banda del servidor.

En aquest cas es teclejaria **idlj –fserver <nomFitxer>.idl**

El compilador **idlj** genera la versió Java de l'interfície definida, així com fitxers font de classes Java i l'esquelet que permeten a les aplicacions comunicar-se amb l'ORB.



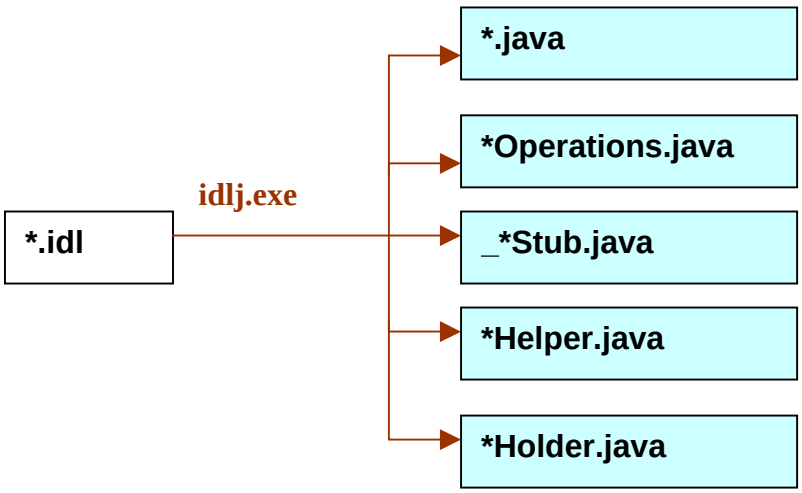

Com es veu e la figura els fitxers generats son 3:

- _*ImplBase.java* : Aquesta classe abstracta és l'esquelet del servidor. Implementa la interfície *.java* . D'aquesta classe ha de ser descendent la classe que representi l'objecte remot que es vol implementar.
- **Operations.java* : Aquesta interfície conté els mètodes per ser invocats remotament que s'han declarat al fitxer *.idl.
- **.java* : Aquesta interfície és la definida inicialment en versió Java. Descendeix de **Operations.java*, així com de *org.omg.CORBA.Object* i *org.omg.CORBA.portable.IDLEntity* . La referència remota de l'objecte el veurà com un objecte titular d'aquesta interfície.

Ara només manca realitzar l'implementació de l'objecte a nivell de servidor. Per fer-ho es crea la classe corresponent com a classe derivada de la classe _**ImplBase.java* i s'implementen els mètodes de la interfície remota. Un cop implementada la classe, que generalment se sol anomenar amb el substantiu de la interfície seguida pel qualificatiu *servent* , ja només cal que l'aplicació servidor que gestioni l'objecte el declari com a objecte d'aquesta classe i realitzi les tasques de connexió del ORB i registre de l'objecte actiu amb el nom a la xarxa.

*Disseny de l'aplicació que usa objectes distribuïts (client).*

Per al disseny d'aplicacions d'us remot d'objectes a través de IDL en llenguatge Java, es segueix el següent procediment. Partint del codi IDL de definició d'interfície, el qual s'ha de distribuir en cas que l'implementador del servidor no sigui el del client, aquest es "mapeja" amb el compilador **idlj** , amb l'opció "–fclient". El compilador **idlj** genera la versió Java de l'interfície definida al costat client o usuari remot.



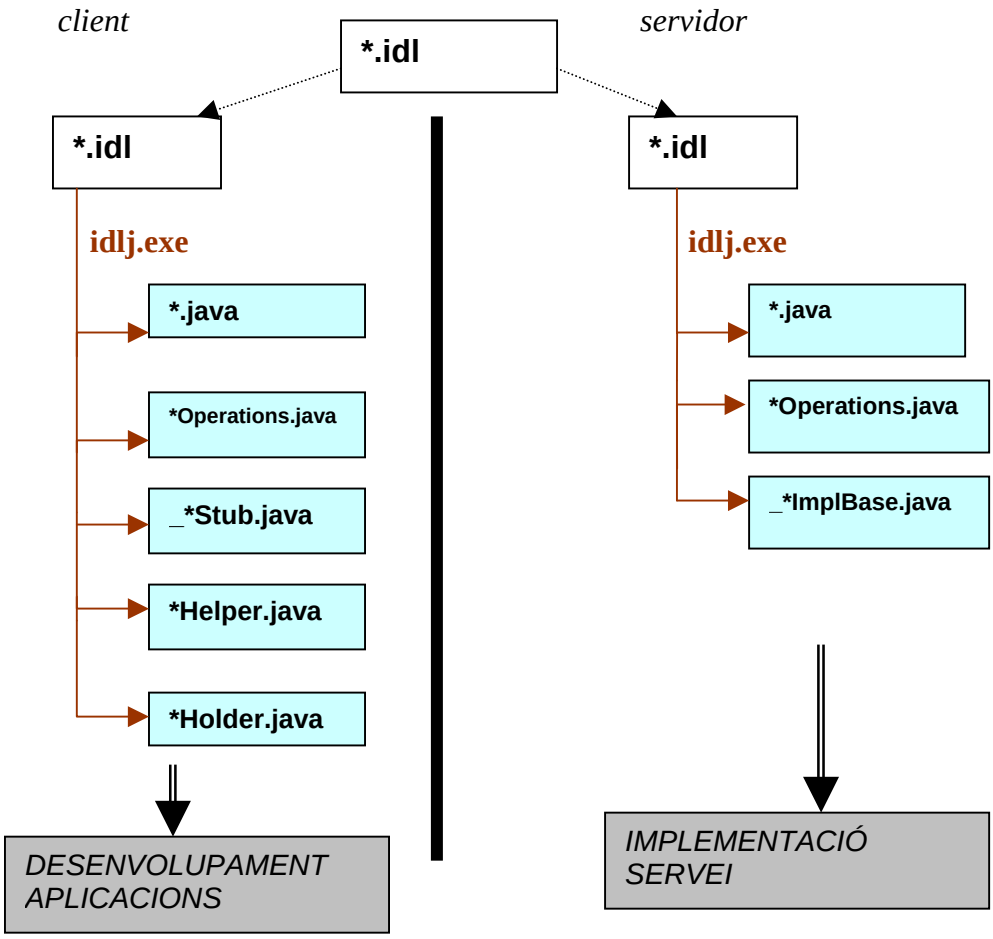

En aquest cas els fitxers generats son 5:

- **Operations.java* : % explicat anteriorment %
- **.java* : % explicat anteriorment %
- _**Stub.java* : Aquesta classe és la que proporciona funcionalitat CORBA al client. Implementa la interfície **.java* .

- **Helper.java* : Aquesta classe final proporciona funcionalitats auxiliars, entre elles el mètode *narrow* que converteix les referències dels objectes CORBA als seus propis tipus.
- **Holder.java* : Aquesta classe final conte un exemplar públic del tipus **.java* . Proporciona operacions per arguments d'entrada i sortida, que CORBA posseeix, però que no son fàcils de convertir a la semàntica de Java.

A partir d'aquestes classes i interfícies ja només cal que quan en una aplicació es desitgi declarar una referència remota a un objecte d'aquesta classe, es realitzin les tasques de connexió del ORB i cerca de l'objecte actiu amb el nom corresponent a la xarxa.

Aquests conceptes es poden il·lustrar al següent esquema:



En ell s'aprecia com a partir de la definició de la interfície remota tant servidor (únic) com clients (un o més), obtenen llurs classes en Java per poder implementar aplicacions. En aquest punt és on s'aprecia la característica principal que distingeix IDL d'altres tecnologies d'objectes remots, ja que a partir de la definició de la interfície remota es poden obtenir les classes necessàries per implementar respectivament client i servidor en diferents llenguatges de programació. És a dir, el servidor podria estar implementat en Java i el client en FORTRAN o C++, per citar dos llenguatges notablement implantats en el sector. Malgrat això, en el present projecte es treballarà en llenguatge Java en la implementació dels dos costats (client i servidor).

# 4. ANTECEDENTS.

## 4.1. DESENVOLUPAMENT D'APLICACIONS DE CONTROL REMOT EN JAVA.

El principal antecedent del present projecte és un projecte realitzat anteriorment en el que es dissenya i s'implementa una classe Java per interactuar amb la targeta de comunicació PCL812-PG. Tal i com s'especificava als objectius, el punt de partida per la interacció amb la maqueta del pont-grua des de Java partirà de l'ús d'aquesta classe, ja que aquesta maqueta es comunica amb el PC amb una targeta de comunicació d'aquest model.

Com un dels resultats del projecte, a nivell d'usuari s'obtenia la classe *PCL812PG.class* , que permetia interactuar des de Java amb la placa. L'esmentada classe posseïa la següent interfície pública:

**obrir**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode recupera els paràmetres de la targeta de l'arxiu de configuració, i els emmagatzema en la memòria per tenir una referència ràpida. Necessita ser cridat abans que qualsevol altre mètode. |
| **PARÀMETRES:** | DeviceNum, de tipus *long*, que és el número d'ordre assignat a la targeta que es vol utilitzar, dins de la relació de les targetes instal·lades. (Veure *L'aplicació DeviceInstallation* en l'annex) |
| **RETORN:** | Cap. |
| **EXCEPCIONS:** | • ConfigDataLost, si no es poden recuperar les dades de configuració de la targeta.<br>• CreateFileFailed, si el driver de baix nivell no es pot obrir.<br>• MemoryAllocateFailed, si no es pot assignar memòria. |

**tancar**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode es necessita per alliberar la memòria assignada amb el mètode *obrir* quan ja no s'utilitzen més funcions d'aquesta llibreria. |
| **PARÀMETRES:** | Cap. |
| **RETORN:** | Cap. |
| **EXCEPCIONS:** | InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat. |

**lecturaCanalAnalogic**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet llegir qualsevol dels setze canals analògics d'entrada. |
| **PARÀMETRES:** | • canal, de tipus *short*, que és el número de canal i pot ser un valor entre 0 i 15 (canal 0 al canal 15).<br>• guany, de tipus *short*, que és el guany i que admet com a possibles valors 0, 1, 2 i 3, que són els codis corresponents als guanys 1, 2, 4 i 8. |
| **RETORN:** | El valor de la lectura de la dada com un tipus float. |
| **EXCEPCIONS:** | • AICorversionFailed, hi ha un problema a la conversió A/D.<br>• AIScaleFailed, si no es pot escalar el valor binari a volts.<br>• BoardIDNotSupported, si el Board ID no és el correcte.<br>• DataNotReady, si TrigMode=1 i la dada no està preparada.<br>• FunctionNotSupported, si aquest mètode no es pot utilitzar amb la targeta seleccionada.<br>• InvalidChannel, si el paràmetre canal no es troba entre 0 i 15.<br>• InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat.<br>• InvalidGain, si el paràmetre guany no és 0, 1, 2 o 3. |

**escripturaCanalAnalogic**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet escriure a qualsevol dels dos canals analògics de sortida. |
| **PARÀMETRES:** | • canal, de tipus *short*, que és el número de canal i el seu valor pot ser 0 o 1 (canal 0 o canal 1).<br>• valor_sortida, de tipus float, que és el valor de sortida que es vol escriure. |
| **RETORN:** | Cap. |
| **EXCEPCIONS:** | • BoardIDNotSupported, si el Board ID no és el correcte.<br>• FunctionNotSupported, si aquest mètode no es pot utilitzar amb la targeta seleccionada.<br>• InvalidAnalogOutValue, si el paràmetre valor de sortida no es troba entre 0 i el valor màxim de sortida (5 o 10).<br>• InvalidChannel, si el paràmetre canal no és 0 o 1.<br>• InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat. |

**lecturaByte**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet llegir qualsevol dels dos ports digitals d'entrada. |
| **PARÀMETRES:** | num_port, de tipus *short*, que és el número del port i el seu valor pot ser 0 o 1 (port 0 o port 1). |
| **RETORN:** | El valor de la lectura de la dada com un tipus *short*. |
| **EXCEPCIONS:** | • BoardIDNotSupported, si el Board ID no és el correcte.<br>• FunctionNotSupported, si aquest mètode no es pot utilitzar amb la targeta seleccionada.<br>• InvalidChannel, si el paràmetre num_port no és 0 o 1.<br>• InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat. |

**escripturaByte**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet escriure a qualsevol dels dos ports digitals de sortida. |
| **PARÀMETRES:** | • num_port, de tipus *short*, que és el número del port i el seu valor pot ser 0 o 1 (port 0 o port 1).<br>• mask, que és un valor de tipus *short* per indicar els bits als que es vol canviar l'estat. Si han de canviar tots el seu valor en hexadecimal seria FF.<br>• valor_sortida, de tipus *short*, que és el valor de sortida que es vol escriure. |
| **RETORN:** | Cap. |
| **EXCEPCIONS:** | • BoardIDNotSupported, si el Board ID no és el correcte.<br>• FunctionNotSupported, si aquest mètode no es pot utilitzar amb la targeta seleccionada.<br>• InvalidChannel, si el paràmetre num_port no és 0 o 1.<br>• InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat. |

**lecturaBit**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet llegir qualsevol dels vuit bits dels dos ports digitals d'entrada. |
| **PARÀMETRES:** | • num_port, de tipus *short*, que és el número del port i el seu valor pot ser 0 o 1 (port 0 o port 1).<br>• num_bit, de tipus *short*, que és el número de bit i pot ser un valor entre 0 i 7. |
| **RETORN:** | El valor de la lectura del bit com un tipus *short*. |
| **EXCEPCIONS:** | • BoardIDNotSupported, si el Board ID no és el correcte.<br>• FunctionNotSupported, si aquest mètode no es pot utilitzar amb la targeta seleccionada.<br>• InvalidChannel, si el paràmetre num_port no és 0 o 1.<br>• InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat.<br>• InvalidInputParam, si el paràmetre num_bit no es troba entre 0 i 7. |

**escripturaBit**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet escriure qualsevol dels vuit bits dels dos ports digitals de sortida. |
| **PARÀMETRES:** | • num_port, de tipus *short*, que és el número del port i el seu valor pot ser 0 o 1 (port 0 o port 1).<br>• num_bit, de tipus *short*, que és el número de bit i pot ser un valor entre 0 i 7.<br>• estat, que és un valor de tipus *short* que pot ser 0 o 1. |
| **RETORN:** | Cap. |
| **EXCEPCIONS:** | • BoardIDNotSupported, si el Board ID no és el correcte.<br>• FunctionNotSupported, si aquest mètode no es pot utilitzar amb la targeta seleccionada.<br>• InvalidChannel, si el paràmetre num_port no és 0 o 1.<br>• InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat.<br>• InvalidInputParam, si el paràmetre num_bit no es troba entre 0 i 7. |

**lecturaByteSortidaActual**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet llegir qualsevol dels dos ports digitals de sortida. |
| **PARÀMETRES:** | num_port, de tipus *short*, que és el número del port i el seu valor que pot ser 0 o 1 (port 0 o port 1). |
| **RETORN:** | El valor de la lectura de la dada com un tipus *short*. |
| **EXCEPCIONS:** | • BoardIDNotSupported, si el Board ID no és el correcte.<br>• FunctionNotSupported, si aquest mètode no es pot utilitzar amb la targeta seleccionada.<br>• InvalidChannel, si el paràmetre num_port no és 0 o 1.<br>• InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat. |

**lecturaBitSortidaActual**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet llegir qualsevol dels vuit bits dels dos ports digitals de sortida. |
| **PARÀMETRES:** | • num_port, de tipus *short*, que és el número del port i el seu valor que pot ser 0 o 1 (port 0 o port 1).<br>• num_bit, de tipus *short*, que és el número de bit i pot ser un valor entre 0 i 7. |
| **RETORN:** | El valor de la lectura del bit com un tipus *short*. |
| **EXCEPCIONS:** | • BoardIDNotSupported, si el Board ID no és el correcte.<br>• FunctionNotSupported, si aquest mètode no es pot utilitzar amb la targeta seleccionada.<br>• InvalidChannel, si el paràmetre num_port no és 0 o 1.<br>• InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat.<br>• InvalidInputParam, si el paràmetre num_bit no es troba entre 0 i 7. |

**lecturaBytePort**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet llegir qualsevol dels dos ports digitals d'entrada. |
| **PARÀMETRES:** | • adr_port, de tipus *short*, que és l'adreça del port digital d'entrada. Aquesta adreça depen de l'adreça base del port I/O per la targeta PC-Labcard 812PG, i per tant els valors possibles són l'adreça base + 6, pel port 0, i l'adreça base +7, pel port 1. |
| **RETORN:** | El valor de la lectura del byte com un tipus *short*. |
| **EXCEPCIONS:** | • InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat.<br>• KeInvalidHandleValue, si el driver en mode Kernel no pot ser obert.<br>• KeFileNotFound, si s'intenta obrir el driver en mode Kernel mentre el driver no s'està executant.<br>• KeTooManyCmds, si es crea un aparent bucle sense fi per al driver en mode Kernel.<br>• KeInvalidHandle, si el handle per al driver en mode Kernel no és vàlid.<br>• KeInvalidParameter, si el paràmetre passat al driver en mode Kernel és incorrecte.<br>• KeNoAccess, si s'intenta accedir a l'adreça del port o de memòria que no s'ha definit al Registre per aquesta targeta. |

**escripturaBytePort**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet escriure a qualsevol dels dos ports digitals de sortida. |
| **PARÀMETRES:** | • adr_port, de tipus *short*, que és l'adreça del port digital de sortida. Aquesta adreça depen de l'adreça base del port I/O per la targeta PC-Labcard 812PG, i per tant els valors possibles són l'adreça base + 13, pel port 0, i l'adreça base +14, pel port 1.<br>• ByteData, de tipus *short*, que és valor de sortida que es vol escriure. |
| **RETORN:** | Cap. |
| **EXCEPCIONS:** | • InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat.<br>• KeInvalidHandleValue, si el driver en mode Kernel no pot ser obert.<br>• KeFileNotFound, si s'intenta obrir el driver en mode Kernel mentre el driver no s'està executant.<br>• KeTooManyCmds, si es crea un aparent bucle sense fi per al driver en mode Kernel.<br>• KeInvalidHandle, si el handle per al driver en mode Kernel no és vàlid.<br>• KeInvalidParameter, si el paràmetre passat al driver en mode Kernel és incorrecte.<br>• KeNoAccess, si s'intenta accedir a l'adreça del port o de memòria que no s'ha definit al Registre per aquesta targeta. |

**lecturaWordPort**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode permet llegir els dos ports digitals d'entrada, per obtenir un valor de tipus word. |
| **PARÀMETRES:** | • adr_port, de tipus *short*, que és l'adreça del port digital d'entrada, concretament la corresponent al port 0. Aquesta adreça depen de l'adreça base del port I/O per la targeta PC-Labcard 812PG, i per tant l'únic valor possible és l'adreça base + 6. |
| **RETORN:** | El valor de la lectura del word com un tipus *short*. |
| **EXCEPCIONS:** | • InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat.<br>• KeInvalidHandleValue, si el driver en mode Kernel no pot ser obert.<br>• KeFileNotFound, si s'intenta obrir el driver en mode Kernel mentre el driver no s'està executant.<br>• KeTooManyCmds, si es crea un aparent bucle sense fi per al driver en mode Kernel.<br>• KeInvalidHandle, si el handle per al driver en mode Kernel no és vàlid.<br>• KeInvalidParameter, si el paràmetre passat al driver en mode Kernel és incorrecte.<br>• KeNoAccess, si s'intenta accedir a l'adreça del port o de memòria que no s'ha definit al Registre per aquesta targeta. |

**escripturaWordPort**

| | |
|---|---|
| **FUNCIÓ:** | Aquest mètode utilitza els dos ports digitals de sortida per escriure una dada de tipus word. |
| **PARÀMETRES:** | • adr_port, de tipus *short*, que és l'adreça del port digital de sortida, concretament la corresponent al port 0. Aquesta adreça depèn de l'adreça base del port I/O per la targeta PC-Labcard 812PG, i per tant l'únic valor possible és l'adreça base + 13.<br>• WordData, de tipus *short*, que és el valor de tipus word de sortida |
| **RETORN:** | Cap. |
| **EXCEPCIONS:** | • InvalidDriverHandle, si DriverHandle és NULL o no s'ha creat.<br>• KeInvalidHandleValue, si el driver en mode Kernel no pot ser obert.<br>• KeFileNotFound, si s'intenta obrir el driver en mode Kernel mentre el driver no s'està executant.<br>• KeTooManyCmds, si es crea un aparent bucle sense fi per al driver en mode Kernel.<br>• KeInvalidHandle, si el handle per al driver en mode Kernel no és vàlid.<br>• KeInvalidParameter, si el paràmetre passat al driver en mode Kernel és incorrecte.<br>• KeNoAccess, si s'intenta accedir a l'adreça del port o de memòria que no s'ha definit al Registre per aquesta targeta. |

Per l'ús d'aquesta classe s'ha de tenir en compte que a més a més del fitxer de classe compilat *PCL812PG.class* i els corresponents a les diverses excepcions definides, en el directori de treball, on un directori de cerca d'executables del sistema operatiu (**path**), hi ha de figurar la llibreria **PCL812PG.dll**, que és la que conté la implementació dels mètodes.

De cara a la utilització adequada de la classe, prèviament a la invocació dels diversos mètodes s'ha d'invocar el mètode *obrir*, i per finalitzar-ne l'ús el mètode *tancar*.

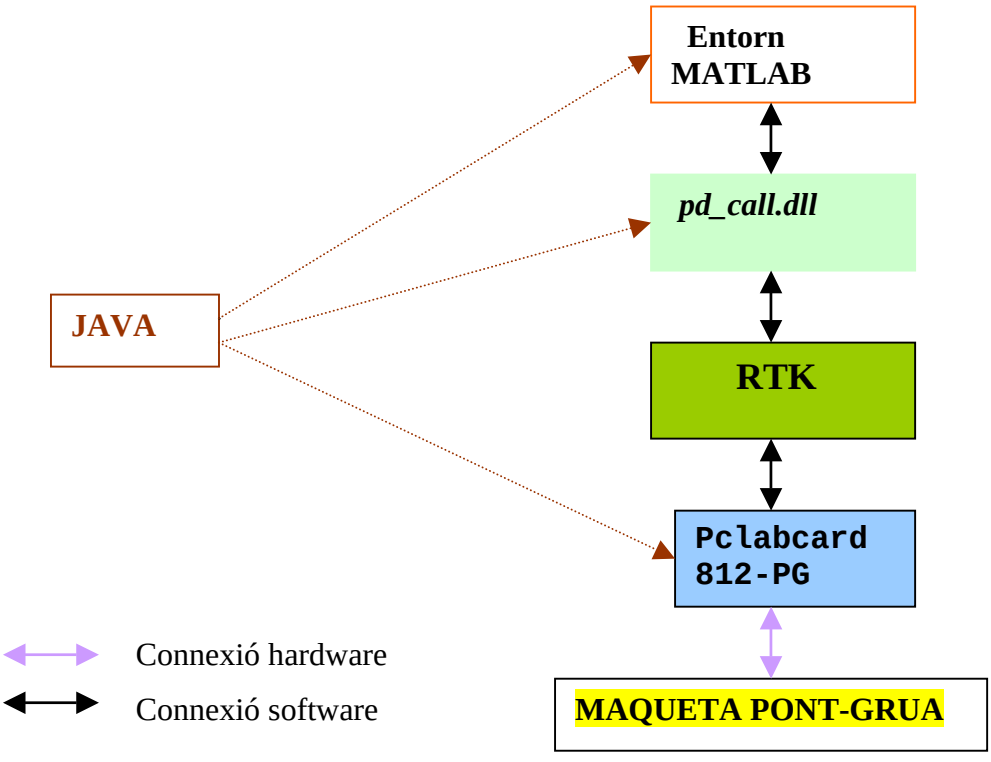
# 5. RELACIÓ DE PROBLEMES A RESOLDRE.

## 5.1. INTERACCIÓ AMB LA MAQUETA DES DE JAVA.

Tal com ja s'ha mencionat, el principal condicionant per la realització del present projecte és el de l'ús del llenguatge de programació Java per a la interacció amb la maqueta del procés a controlar. D'aquesta manera el principal problema a resoldre és la interacció entre l'entorn Java i la maqueta al nivell més baix possible.

La interfície d'usuari que proporciona el fabricant de la maqueta està dissenyada per ser utilitzada des de l'entorn de Matlab. Hi ha una alternativa que consisteix en que l'usuari dissenyi el seu propi algoritme de control i substitueixi els diversos algoritmes disponibles al RTK per aquest. En aquest cas la implementació de l'esmentat algoritme es realitza en llenguatge C i es compila amb l'entorn de compilació de llenguatge C que proporciona Matlab (*The Matlab Compiler*). Però fins i tot en aquest cas el control de flux i la interacció amb l'usuari s'ha de realitzar a través de l'entorn de Matlab.

D'aquesta manera, queda establert que l'objectiu consisteix en poder interactuar a través de Java (enllaçant amb C++ a través de JNI si calgués), amb el procés al nivell més baix possible, tot mantenint el correcte funcionament del mateix.



## 5.2. COMUNICACIÓ ENTRE MAQUETA I USUARI EN EL CAS DE CONTROL REMOT.

Un altre problema per resoldre posteriorment a la resolució de l'anterior és la comunicació entre la maqueta i l'usuari remot. Aquest problema es procurarà resoldre emprant la tecnologia IDL en tant sigui possible, tot mantenint unes mínimes prestacions.

## 5.3. MODELAT I CONTROL DEL PROCÉS.

Finalment ens trobem amb el problema de modelat del sistema. Com a pas previ per al seu control. En aquest projecte l'objectiu serà controlar la posició lineal del carro. En el procés de modelat s'haurà de tenir en compte que es tracta d'un procés no lineal, i procurar aconseguir que els models aproximats que s'emprin no s'allunyin massa del sistema real, a efectes de poder obtenir les prestacions desitjades. També s'haurà de tenir en compte que el procés es controlarà discretament, mesurant entrades actualitzant sortides en instants de temps concrets, ja que el PC és un controlador discret.

# 6. ESTUDI DE LES DIFERENTS ALTERNATIVES.

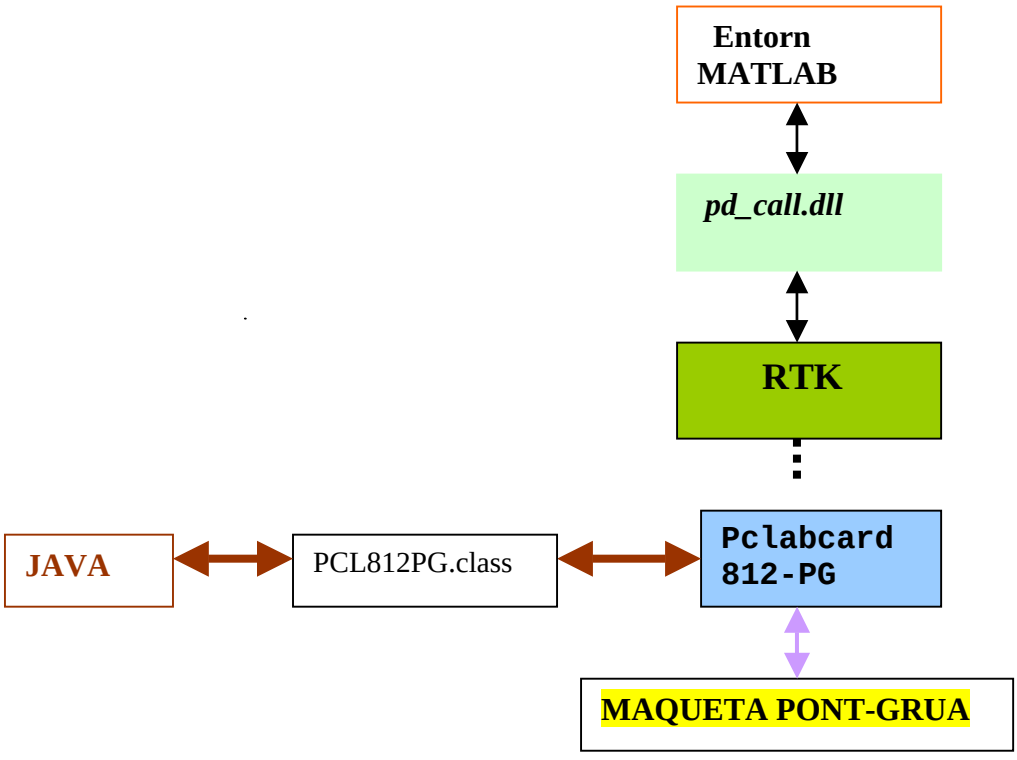
## 6.1. INTERACCIÓ AMB LA MAQUETA DES DE JAVA.

Per solucionar aquest problema estudiarem les diverses alternatives partint de la base de que interessa interactuar amb la maqueta al nivell més baix possible. De forma que es començarà estudiant aquelles alternatives que permetin interactuar al nivell més baix i en cas de ser inviable la interacció, es passaria a estudiar alternatives que permetin la interacció a un nivell més alt.

### 6.1.1. INTERACCIÓ A TRAVÉS DE LA TARGÈTA DE COMUNICACIÓ PCL812-PG.

Aquesta alternativa consisteix en utilitzar la classe mencionada al capítol 4 per interactuar des d'aplicacions escrites en llenguatge Java amb la targeta de comunicació PCL812-PG que és la targeta amb la qual es comunica el RTK de Matlab amb la maqueta del Pont – Grua.

En aquest cas l'esquema del nivell d'interacció seria el següent:



Aquesta alternativa seria la més interessant, ja que d'aquesta manera es prendria el control al nivell més baix possible i l'aplicació o classe Java s'aïllaria de l'entorn de Matlab i de les llibreries del RTK.

Per prendre aquest camí, es parteix de la classe *PCL812PG* i es crea una classe que inclou un objecte d'aquesta, el qual és la via d'interacció amb la targeta de comunicació amb la maqueta.

Se sap que la maqueta disposa d'una entrada (Acció de control), que és la tensió d'alimentació del Driver del motor, i de dues variables de sortida (Posició del carro i Angle del pèndul). Donat que la documentació proporcionada pel fabricant de la maqueta no especifica dades de comunicació a baix nivell, apareix el problema de desconeixement dels protocols de comunicació emprats. Mercès a una aplicació de MS-DOS subministrada pel fabricant amb la qual es pot observar els nivells de tensió a l'entrada i els valors numèrics dels encoders de Posició i Angle respectivament en el marc d'experiments senzills de reacció del sistema, es dedueix el següent:

- Rang Tensió Driver motor → 0,0::5,0 (volts)
- Resolució encoder Posició Carro → 16 bits. (0 :: 65535)
- Resolució encoder Angle Pèndul → 16 bits. (0 :: 65535)

També es dedueix que la relació entre nivell de tensió i velocitat i sentit del carro és la següent:

- 0,0 volts → Màxima velocitat a l'esquerra.
- 2,5 volts → ATURAT.
- 5,0 volts → Màxima velocitat a la dreta.

La relació entre el valor dels encoders i les magnituds mesurables (Posició i Angle) depèn del punt de referència, que es fixa amb la funció *ResetEncoder* del RTK, també invocable des de la present aplicació.

Després de diverses accions experimentals, es descobreix que el Driver del motor de la maqueta (l'Acció de Control) es comunica amb la targeta a través del canal 0 analògic d'aquesta. Per comprovar experimentalment sobre el terreny i obtenir coneixement sobre l'arquitectura de la comunicació a través de la targeta es realitza una simple aplicació *AplicacioControlCarro* . Aquesta aplicació, el codi detallat de la qual es troba a l'Apèndix 1, crea una finestra amb dos botons (DRETA i ESQUERRA) les etiquetes dels quals son prou

descriptives de la seva funció. Prenent DRETA o ESQUERRA es fixa la tensió del canal 0 de la targeta a 3,0 i 2,0 volts respectivament, de forma que s'aconsegueix que el carro es mogui en un sentit o en un altre. Si no es prem cap botó es deixa anar es fixa la tensió a 2,5 v. de forma que s'atura el carro. En cas de tancar-se la finestra principal, també es fixa la tensió a 2,5 prèviament a tancar el programa.

Aquesta experimentació dona un resultat positiu. Posteriorment el que interessa és conèixer la forma de transmissió de dades de les variables de sortida (Posició i Angle), transmeses a través dels respectius Encoders. Per obtenir resultats experimentals es desenvolupen unes aplicacions en Java, anomenades *EntradaDigital* i *EntradaAnalogica* (Apèndixs 2 i 3), que llegeixen els valors de les diverses entrades analògiques i digitals de la targeta durant un temps amb un determinat període de mostreig. Posteriorment aquestes dades son gravades en un arxiu *.dat, que posteriorment des de Matlab es visualitzaran en una gràfica per tal d'observar la seva evolució de valors durant el període de temps que ha durat l'experiment. Aquest procés es repeteix per les diverses entrades analògiques i digitals de la placa. Durant els experiments es provoquen moviments manualment del Carro, així com del Pèndul per tal de descobrir per quin o quins canals es transmet cada dada.

Amb els diversos experiments realitzats s'observa en les gràfiques evolucions gents coherents amb els moviments provocats manualment al carro i al pèndul. L'experimentació ens deixa amb la conclusió que la maqueta i la targeta es comuniquen, pel que fa a lectures de senyals de Posició i Angle, a través d'un protocol desconegut i no pas deduïble d'una forma raonable.

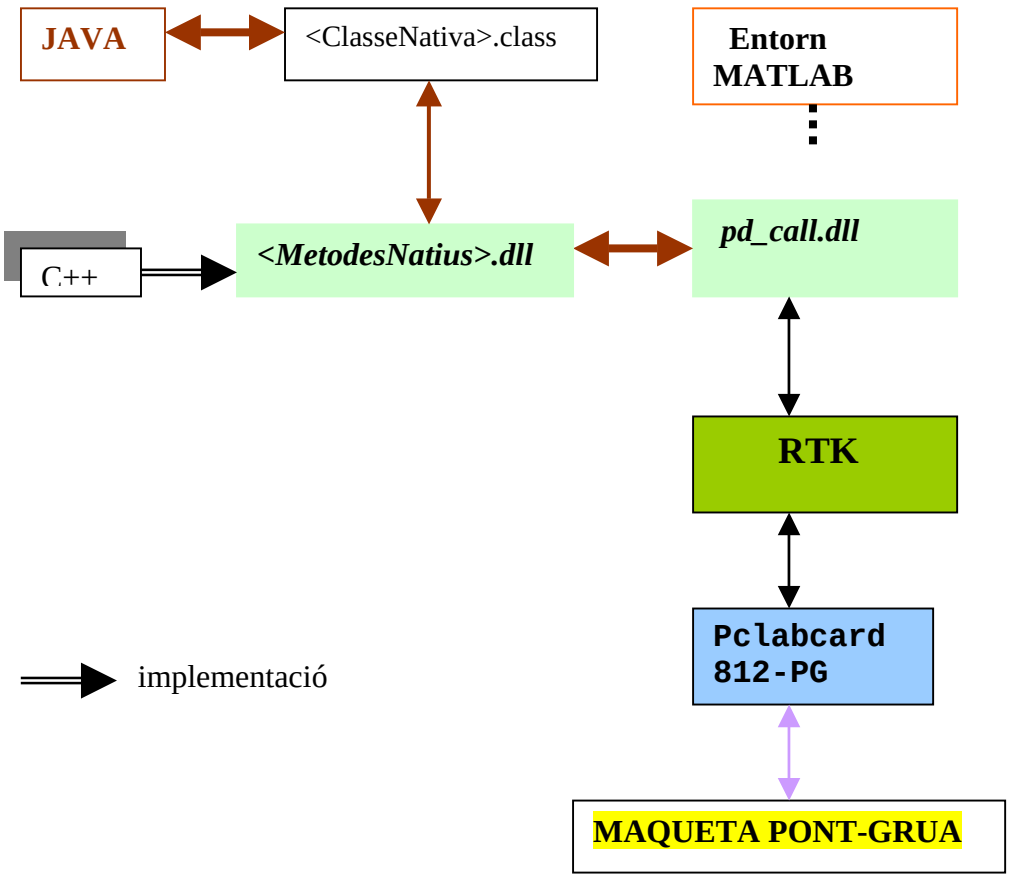
Un cop arribat a aquest nivell, queda demostrat que la maqueta del pont-grua no permet que amb la classe *PCL812PG* del projecte del que es pretenia partir inicialment, es pugui interactuar per tal de realitzar control de la mateixa.

D'aquesta manera queda exclosa la possibilitat d'interacció a aquest nivell, cosa que obliga a passar a un nivell superior.

### 6.1.2. INTERACCIÓ A TRAVÉS DE LA LLIBRERIA DE COMUNICACIÓ DEL RTK.

Aquesta alternativa consisteix en utilitzar la llibreria dinàmica ***pd_call.dll*** utilitzada des de Matlab per interactuar amb el RTK per comunicar amb la maqueta. Donat que no és possible interactuar d'una forma directa i immediata amb una DLL des de Java, aquesta opció requereix codi natiu (JNI).

En aquest cas l'esquema del nivell d'interacció seria el següent:




Per interactuar a través de la llibreria dinàmica de comunicació hi ha una sèrie de conceptes referits a les DLLs que convé citar.

Una DLL és un arxiu executable que conté funcions exportables que poden ser utilitzades per altres programes que no tenen la necessitat de contenir-les en el seu codi executable.

Perquè des d'un programa, escrit en codi C per exemple, hi ha dues formes d'enllaçar codi compilat d'una aplicació que invoca funcions d'una DLL amb la DLL:

1. **Enllaç estàtic :** Consisteix en utilitzar una llibreria estàtica d'importació, generada a partir de la DLL que conté referències a les funcions exportables d'aquesta. Per aquesta opció es necessita la llibreria estàtica d'importació o en el seu defecte l'arxiu de definició de recursos exportables d'aquesta *.def.
2. **Enllaç dinàmic :** Consisteix en utilitzar la DLL en temps d'execució de l'aplicació que n'invoqui funcions. En aquest cas no es necessita res més que la DLL, però és necessari per part de l'usuari de la mateixa un coneixement a fons dels recursos o funcions exportables de la llibreria, ja que un error en la seva crida, pas d'arguments, etc. No provocarà un error de compilació o enllaçat sinó un error en temps d'execució.

Donat que en el present cas no es disposa de res més que la pròpia DLL l'única possible alternativa és la segona.

A partir del protocol de crida de les funcions de la DLL

**`valorRetorn = pd_call( 'NomFuncio', [ Arguments ] )`**

i després d'un intens procés de coneixement de l'entorn de Matlab, es pot deduir que la llibreria dinàmica *pd_call.dll* és un arxiu *mex* (executable de Matlab). Aquests arxius, malgrat estan implementats en C o altres llenguatges d'alt nivell com Fortran, estan específicament dissenyats per a ésser utilitzats des de Matlab. La llibreria conté únicament una funció exportable anomenada *mexFunction* , que respon a la sintaxi des de C de

```
void mexFunction(int nlhs ,mxArray** plhs,
                        int nrhs,mxArray** prhs);
```

Com es pot observar la funció té 4 arguments i no retorna cap valor. Els arguments son els següents:

`nlhs` : Enter que indica el nombre d'arguments de sortida.

`** plhs` : Vector de punters a estructures de tipus MxArray; son els elements de sortida de la funció.

`nrhs` : Enter que indica el nombre d'arguments d'entrada.

`** prhs` : Vector de punters a estructures de tipus MxArray; son els elements d'entrada de la funció.

Les estructures de tipus MxArray son objectes que representen les variables de Matlab. Un MxArray pot ser un vector de números reals, de números complexes, una matriu, un Array tridimensional, un "String",...i és per això que el seu tamany és variable, depenent dels elements que el formen. En cas de ser valors numèrics, aquests en Matlab corresponen al tipus double de C. Quan des de la finestra de comandes de Matlab es crida a una funció, aquesta pot ser una *mexFunction* o no. En cas de ser-ho els arguments de la crida des de Matlab son passats com a MxArrays a la funció i els valors de retorn com uns punters als MxArrays de sortida amb llurs valors que n'indiquen la quantitat.

Per treballar amb MxArrays des de C , Matlab disposa d'unes llibreries en C amb mètodes de creació, destrucció, manipulació i transformació de MxArrays. Les capçaleres d'aquests mètodes es poden trobar als arxius *matrix.h* i *mex.h* entre d'altres.

El codi executable d'aquests mètodes es troba també en llibreries dinàmiques i Matlab no posa a disposició de l'usuari les llibreries estàtiques d'importació de les mateixes, de manera que s'han de crear a partir dels arxius de definició de recursos exportables *.def. Això és degut a que els diversos compiladors de C i C++ no segueixen la mateixa arquitectura en aquest camp i aleshores les llibreries d'importació difereixen lleugerament entre elles. La creació de les llibreries d'importació es realitza amb l'executable **lib.exe** disponible al **Microsoft Visual Studio** juntament amb Visual C++. Per agilitzar el procés s'ha procedit a la creació de l'arxiu de comandes *Matlab_C_libs.bat* que realitza les comandes necessàries per crear les llibreries d'importació a partir dels arxius de definició de recursos exportables.

A partir de tot això s'intenta interactuar amb la DLL de comunicació (***pd_call.dll***) a través de C++ inicialment, en cas de resultar en èxit es passaria a Java a través de JNI. Per fer-ho s'escriu un petit programa en C++, *ProvaMex.cpp* el qual simplement carrega la DLL en memòria, i crida a la funció amb l'argument `'StopPractical'` , que és la crida a la sub-funció que fa aturar el carro.

Això s'aconsegueix amb el suport d'unes funcions de C++ disponibles a la llibreria *windows.h*. Aquestes funcions son les següents:

| **Prototipus de crida a la funció** | **Retorn** | **Acció que realitza** |
|---|---|---|
| `LoadLibrary(const char*);` | `HINSTANCE` | Carrega en memòria la DLL el nom de la qual s'especifica a l'argument. Retorna un punter a la DLL o NULL si no l'ha pogut carregar. |
| `GetProcAddress (HINSTANCE,const char*);` | `PUNTER_FUNCIO` | Obté el punter a la funció de la DLL a on apunta el primer argument, amb el nom que s'especifica al segon. (veure **) |
| `FreeLibrary(HINSTANCE);` | `BOOL` | Descarrega de la memòria la DLL a on apunta l'argument. Retorna TRUE en cas d'èxit i FALSE en cas contrari. |

(**) → El punter retornat per la segona funció ho és, en cas d'haver-se trobat, d'una funció determinada. El tipus del valor retornat, per tant, ha d'ésser prèviament definit com un punter que apunta a una funció amb un determinat tipus de valor de retorn i uns determinats tipus d'arguments. Per exemple, partim del cas en el que es té una DLL on es troba una funció que retorna un *int* i pren com a arguments un *float* i un *int* . A l'hora de definir el tipus del valor retornat per la funció `GetProcAddress` en cas de sol·licitar la recerca d'aquesta funció seria de la forma:

```
typedef int (*PUNTER_FUNCIO)(float,int);
```

El resultat de l'execució de *ProvaMex.cpp* ens indica que el programa ha carregat la llibreria, ha trobat la funció *mexFunction* , però s'ha produït un error desconegut en temps d'execució i el programa ha finalitzat abruptament.

Després d'això s'intenta investigar per veure si es va pel bon camí i es realitza un simulacre de creació d'arxiu *mex*. Des de Matlab s'escriu un fitxer *Prova_call.m* que realitza unes accions sense gaire transcendència, però que té en comú amb la funció que ens interessa el seu protocol de crida. A partir de *Prova_call.m* es genera amb l'entorn de compilació C de Matlab, enllaçat amb **Visual C++** , l'arxiu mex *Prova_call.dll* i un conjunt d'arxius font en

llenguatge C , utilitzats per generar l'arxiu mex. Aquests arxius son *Prova_call.h , Prova_call.cpp, Prova_call_mex.cpp* .

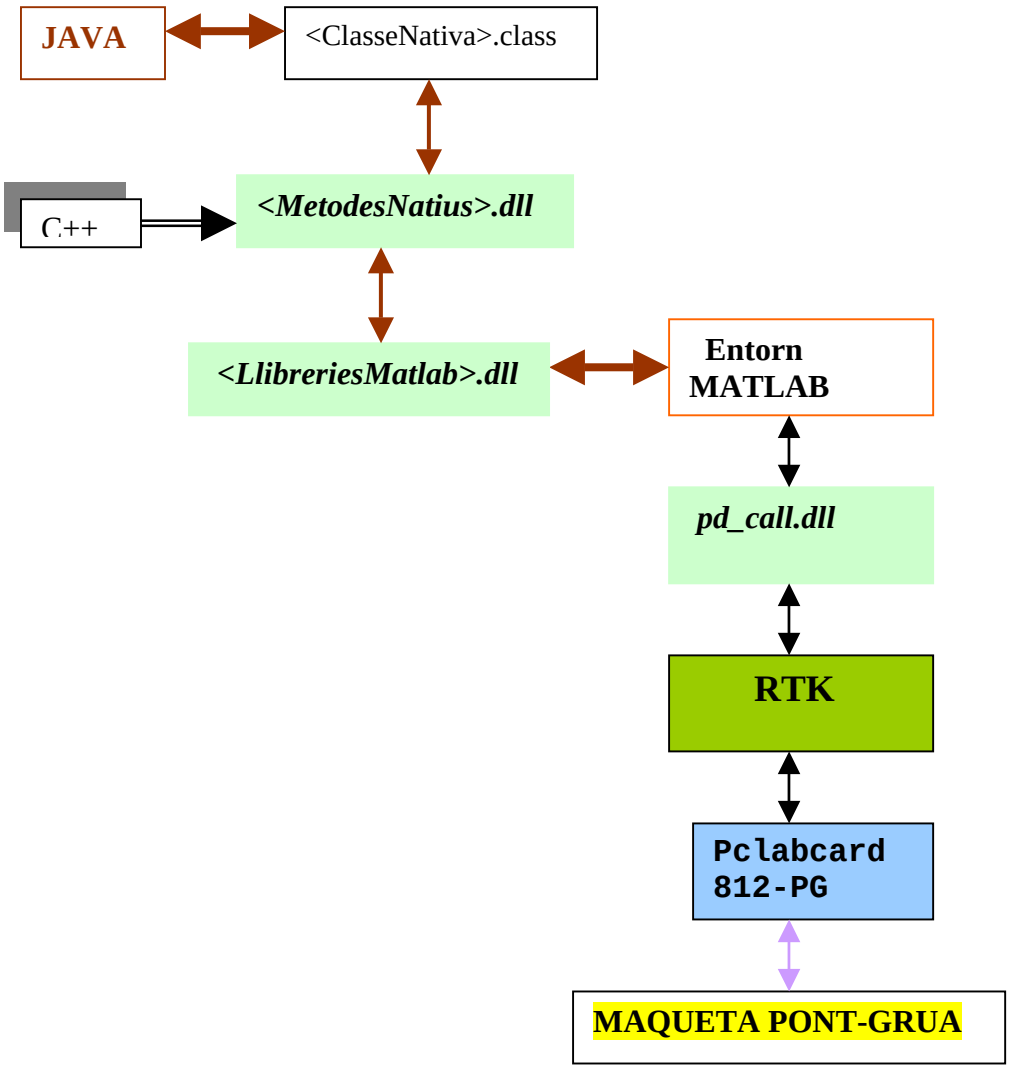
A partir d'aquests arxius els indicis apunten a que anem pel bon camí i aleshores provem de realitzar la tasca anteriorment realitzada amb *pd_call.dll* ara amb *Prova_call.dll* ;cridar una funció implementada per ser invocada des de Matlab des de C++. El resultat és satisfactori.

D'aquesta forma i després de diverses proves alternatives de crida a la funció de *pd_call.dll* queda evidenciat que aquesta funció no rep i transmet la totalitat de la informació que processa a través dels seus arguments d'entrada i sortida, ja que els arxius *mex* es comuniquen a voltes amb l'entorn de Matlab d'altres formes. Això explicaria els errors en temps d'execució de *ProvaMex.cpp* i també explicaria el fet que l'aplicació de prova *Prova_call.dll* si funcionés ja que en el seu codi no es comunicava amb l'entorn de Matlab des de dins de la funció.

Aquesta situació no ens deixa més remei que considerar també inviable la interacció amb la maqueta a aquest nivell i passar a un nivell superior.

### 6.1.3. INTERACCIÓ A TRAVÉS DE L'ENTORN DE MATLAB

Aquesta alternativa consisteix en intentar interactuar des de Java , a través de JNI amb l'entorn de Matlab per així poder comunicar amb el RTK i a través d'ell amb la maqueta. Aquesta interacció és la de més alt nivell possible i en cas de no ser satisfactòria deixaria molt poques possibilitats per explorar. En aquest cas l'esquema del nivell d'interacció seria el següent:




Per interactuar des de codi C amb l'entorn de Matlab, es disposa d'una llibreria dinàmica (*.dll) de mètodes, les capçaleres dels quals es troben a l'arxiu *engine.h* proporcionat amb Matlab, juntament amb les llibreries de les funcions per la gestió d'MxArrays. Aquesta llibreria proporciona recursos que permeten interactuar amb l'entorn de Matlab, a través d'uns punters a objectes de tipus *Engine* . Un punter d'aquest tipus és una referència d'un entorn de Matlab obert i carregat al sistema operatiu. Amb altres mètodes es poden realitzar des de C crides a comandes de l'entorn de Matlab d'una forma anàloga a la que es faria servir si es realitzés des de la finestra de comandes de Matlab. Entre els mètodes que proporciona la llibreria cal destacar els següents:

| **Prototipus de crida a la funció** | **Retorn** | **Acció que realitza** |
|---|---|---|
| `engOpen(const char*);` | `Engine*` | Obra un entorn de treball de Matlab i en retorna la referència. Com a argument requereix un "String" amb el format "\0" |
| `engClose(Engine*);` | `-` | Tanca l'entorn de Matlab apuntat per l'argument. En cas d'haver-hi altres punters Engine apuntant-hi el deixa obert a mercès dels altres punters. |
| `engEvalString(Engine*, cons char*);` | `-` | Executa la comanda escrita a través del "String" del segon argument a l'entorn apuntat pel primer. El contingut de l'"String" equival a teclejar la comanda des de la finestra de Matlab. |
| `engPutArray(Engine*, mxArray*);` | `-` | Insereix l'mxArray del segon argument a l'entorn apuntat pel primer. |
| `engGetArray(Engine*, const char*);` | `mxArray*` | Retorna un punter a mxArray creat com a imatge de la variable de l'entorn de Matlab apuntada pel primer argument, de nom el contingut del "String" del segon argument. |

Amb aquests instruments es procedeix a realitzar una experimentació per comprovar la viabilitat d'interactuar amb Matlab, des de C++ amb aquestes eines.

S'implementa en codi Matlab l'arxiu *LlacObert.m* que excita el sistema en llaç obert, invocant els corresponents mètodes de la llibreria ***pd_call.dll*** des de Matlab, i s'observen els resultats.

Posteriorment s'implementa l'equivalent de *LlacObert.m* en C++, d'acord amb les eines anteriorment esmentades, i s'observen els resultats.

Es comprova que tot i una certa lentitud al arrancar respecte de la velocitat de resposta de l'entorn de Matlab al ésser invocat l'algoritme escrit en *LlacObert.m*, s'aprecia la correspondència entre els comportaments, la qual cosa obre les portes a interactuar des de Java (passant per C++ a través de JNI) amb la maqueta del pont-grua.

## 6.2. COMUNICACIÓ ENTRE MAQUETA I USUARI EN EL CAS DE CONTROL REMOT.

Per la resolució d'aquest problema s'ha de procurar resoldre emprant la tecnologia IDL en tant sigui possible, tot mantenint unes mínimes prestacions. En aquest cas les prestacions serien tenir una interfície amb la maqueta amb la mateixa capacitat d'interactuar que la interfície per al control local.

Donat que amb la resolució del problema anterior, queda establert que serà necessària la implementació de part d'aquesta interfície per al control local amb codi natiu (JNI), l'únic possible entrebanc pot ser la incompatibilitat entre aquestes dues prestacions de l'entorn Java. Per tal de sortir de dubtes, s'implementa una interfície remota a partir de l'exemple de l'apartat 3.4.4 del capítol 3 *ProvaJNI* , la definició de la qual s'implementa a l'arxiu *ProvaJNI.idl*. El desenvolupament i, al igual que en la resta de casos, els codis dels arxius involucrats en el seu desenvolupament es poden consultar als Apèndixs 11 al 23.

Amb aquesta experimentació comprovem com, en principi, la funcionalitat és correcta, amb la qual cosa quedaria definitivament establert el camí de l'ús de la tecnologia IDL en el cas del control remot.

## 6.3. MODELAT I CONTROL DEL PROCÉS.

### 6.3.1. MODELAT DEL PROCÉS.

El procés de modelat del procés de la maqueta del pont-grua, s'ha de realitzar tenint en compte l'objectiu del seu control. Per fer-ho es proposa com a objectiu el control de la posició del carro en el rail. Per realitzar la tasca de modelat es parteix en primer lloc de les equacions d'estat subministrades en el manual del fabricant, tot tenint en compte que certs valors de certs paràmetres que en elles figuren no son del tot exactes i poden variar fins i tot d'un exemplar de maqueta a un altre.

Les equacions d'estat ens relaciones les variables mesurables del procés amb les seves derivades. En el procés aquestes segueixen relacions de no linealitat. Una tècnica per tal d'eliminar aquestes no linealitats ,consisteix en aproximar el model a un de lineal entorn d'un punt de treball de la o les variables que les provoquen. D'aquesta manera es podran aplicar les tècniques de control lineal, per una zona d'operació al voltant del punt de treball especificat.

En aquest cas la variable que interessa "fixar" entorn d'un punt de treball, és la posició angular del pèndul $(\theta)$. Davant de desplaçaments lineals del carro i sense canvis bruscs de sentit, $(\theta)$ està al voltant del punt $\theta = \pi.\text{rad}$.

A partir d'aquestes premisses i aplicant el desenvolupament de Taylor, s'obté a partir de les equacions d'estat del procés, que la funció de transferència (FT) que relaciona l'acció de control Ua(s) amb la posició del carro X(s) és la següent (veure desenvolupament detallat a l'Apèndix 24):

$$\frac{X(s)}{U(s)} = \frac{17 * \left[(aJ)s^2 + (af_p - l^2 f_p)s + (a\mu g - l^2 \mu g)\right]}{s\left[J^2 s^3 + (f_p J + f_c aJ)s^2 + (J\mu g + f_c af_p - f_c l^2 f_p)s + (f_c a\mu g - f_c l^2 \mu g)\right]}$$

Aquesta FT ens indica que el sistema s'aproxima entorn del punt de treball a un sistema de $4^t$ ordre i que poseeix un integrador pur.

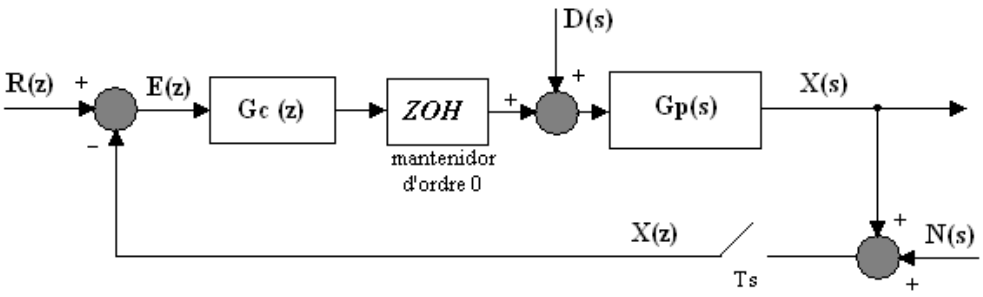
Davant de la complexitat de resoldre sistemes de control amb FT d'aquests ordres, i tenint en compte que el modelat d'un procés simplement és una eina per a l'obtenció d'unes prestacions en el control, es procedeix al següent.

Ja que la peça clau de la FT és el fet que el sistema posseeix un integrador, considerarem inicialment que el sistema es podria aproximar a un de $1^{er}$ ordre amb un integrador, amb la qual cosa simplifiquem bastant els càlculs. Si posteriorment i al disseny del controlador el sistema dona uns resultats molt allunyats dels esperats es procediria a un modelat més acurat.

## 6.3.2. CONTROL DEL PROCÉS.

Per al control de posició del carro hi hauria un gran nombre de tècniques possibles per dissenyar el controlador corresponent.

Si considerem el clàssic llaç de control discret d'un sistema continu:




on:

R(z) → És la consigna de la variable que es pretén controlar.

E(z) → És l'error del procés a controlar.

D(s) → És la senyal de pertorbació, que es desitja que afecti el mínim possible al procés de seguiment de la consigna per part de la variable que es pretén controlar.

X(s) → És la senyal de la variable que es pretén controlar.

N(s) → És el soroll que afecta la lectura del valor de la senyal X(s).

X (z) → És la senyal X(s) mostrejada.

Ts → És el període de mostreig.

Entre les tècniques disponibles n'analitzarem dues:

- Síntesi directa: Amb aquesta tècnica s'obté un controlador, en el que tant l'estructura com els paràmetres venen determinats per la resolució dels càlculs.
- Assignació de pols: Amb aquesta tècnica es parteix d'una estructura del controlador determinada i es cerquen els valors dels paràmetres per un control en el que s'exigeixen unes prestacions. Aquestes es modelen numèricament a través de l'especificació de pols que es desitja que tingui la relació entre la senyal de consigna i la de la variable a controlar.

En aquest cas i com que ens veiem obligats a interactuar amb la maqueta a través de la llibreria del RTK prenem algun dels algoritmes de control disponibles en ella. En aquest cas la tècnica de Síntesi directa no és viable donat que amb ella no es sintonitzen paràmetres d'un controlador, sinó que es determina l'estructura del mateix.

Per tant, prenem l'algoritme de control PID i en determinem els paràmetres, en el domini discret, amb la tècnica de l'assignació de pols, posteriorment a haver obtingut un model aproximat del procés en l'esmentat domini.

# 7. DESENVOLUPAMENT DE LA SOLUCIÓ ADOPTADA.

## 7.1. INTERACCIÓ AMB LA MAQUETA DES DE JAVA.

### 7.1.1. DISSENY DE LA CLASSE.

Per la interacció amb la maqueta es procedeix al disseny d'una classe en Java anomenada *FBK33200local.class* . La nomenclatura s'ha fixat d'acord amb els següents criteris:

- FBK son les sigles del fabricant (Feedback).
- 33-200 és el codi de referència de la maqueta.
- "local" fa referència al caràcter d'interacció a nivell local de la classe en contraposició a l'ús remot.

La classe *FBK33200local.class* haurà de disposar de mètodes, que d'acord amb el que s'ha establert al capítol anterior hauran d'ésser implementats en codi natiu, que permetin des de Java una interacció anàloga a la que permet l'entorn de Matlab.

D'aquesta manera la relació de mètodes públics serà paral·lela, en quan a funcionalitat, a la de les funcions, la disponibilitat de les quals proporciona la llibreria ***pd_call.dll*** .

S'implementaran dos mètodes constructors de les instàncies de la classe, i es tornarà a implementar el mètode finalitzador de la classe, heretat de la classe *java.lang.Object*, de la qual descendeix la present com totes les classes Java en que no s'especifica l'herència d'una classe concreta.

La diferència entre els dos constructors serà en que un d'ells no requereix arguments i l'altre requereix un argument per indicar l'adreça base de la targeta de comunicació PCL812-PG. En cas de ser cridat el primer aquesta adreça serà la que especifica la configuració per defecte del RTK.

La classe disposarà, a més a més, de dos mètodes privats consistents en realitzar les tasques de càrrega de l'entorn de Matlab al sistema operatiu i descàrrega del mateix, respectivament. Aquests mètodes seran invocats pels constructors i pel finalitzador, respectivament.

A més a més, i pel cas dels mètodes de fixació i obtenció de l'algoritme de control actiu al RTK s'implementaran una sèrie de variables enteres finals de classe (*public static final int*), equivalents a constants, que equiparin les sigles dels algoritmes de control al numero a ells associats.

En el cas de la funció que retorna el contingut del BUFFER del RTK en forma de matriu, aquesta en Java s'ha descompost en 7 funcions que retornen vectors d'elements corresponents a les 7 files de la matriu segons la variable l'historial de la qual es vulgui obtenir.

### 7.1.2. EXCEPCIONS.

També s'ha de tenir en compte que pel fet de estar manipulant variables de Matlab de forma externa i convertint a aquest tipus objectes de tipus reconeguts en Java I viceversa , hi ha la possibilitat de que es produeixin errors. Per aquest fet es crea una classe d'excepció anomenada *ExcepcioFBK33200* que els diversos mètodes de la classe *FBK33200local* llençaran en cas que es produeixi algun error o en cas que Matlab no respongui. Aquestes excepcions es llençaran amb uns missatges que informin del tipus d'error que ha provocat el llançament.

| **Missatge de l'excepció** | **Motiu de llançament** |
|---|---|
| "No s'ha pogut carregar Matlab" | No haver-se pogut carregar l'entorn de Matlab amb la invocació del mètode cridat inicialment. |
| "S'ha cridat una funció de Matlab sense haver-se carregat prèviament" | Haver cridat un mètode sense haver-se carregat l'entorn de Matlab prèviament. |
| "S'ha produït un error amb el BUFFER de l'historial de mesures" | Error produït quan els mètodes que retornen les mostres de les diverses variables emmagatzemades al BUFFER del RTK, no retornen uns valors coherents. |
| "No s'ha pogut carregar el Controlador Extern" | Error produït al no poder carregar l'algoritme de control extern, per no haver-se trobat la llibreria *.dll corresponent. |
| "Error desconegut amb la llibreria PD_CALL.DLL" | No haver retornat 0 les funcions que ho retornen per indicar èxit en l'operació sol·licitada. |
| "Numero de paràmetres de la funció setParFiltreVel/PosCarro/Pèndul incorrecte" | No haver passat com a argument a aquestes funcions un Array de 18 elements, encara que alguns valguin 0. |

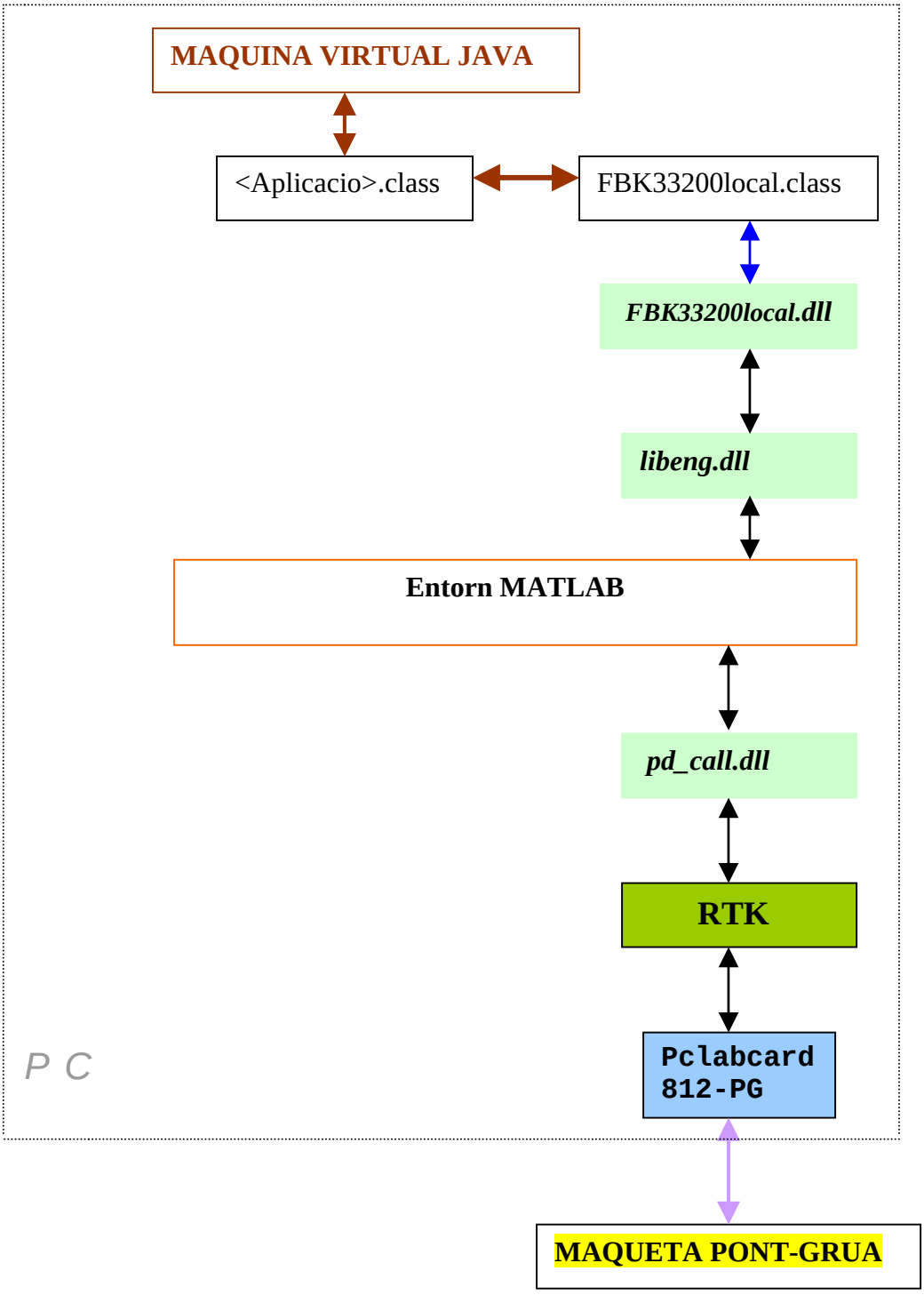
| **Missatge de l'excepció** | **Motiu de llançament** |
|---|---|
| "Filtre digital de Posició/velocitat del Carro/Pèndul NO CAUSAL!" | Haver passat com a argument a les funcions anteriors un Array amb el segon element $\neq 0$. (el filtre digital resultant seria no causal.) |
| "Mes de 20 paràmetres de la funció setParametresControl" | Haver passat com a argument a la funció un Array de més de 20 elements. |
| "Numero de paràmetres de la funció setPW incorrecte" | Haver passat a aquesta funció, que fixa paràmetres de la font d'excitació interna per al cas de llaç obert, relació de paràmetres en format incorrecte. |
| "Valor amplitud senyal d'excitació interna fora de límits" | Haver passat a la funció anterior algun dels paràmetres de fixació del valor de l'acció de control per damunt dels límits: $-1 \leq M \leq 1$ |

La codificació en Java i en codi natiu C dels mètodes de la classe i de les excepcions es troba detallada als Apèndixs 25 al 28. Els arxius involucrats son els següents:

- *FBK33200local.java* → codi font Java de la classe.
- *ExcepcioFBK33200* → codi font Java de l'excepció creada.
- *FBK33200local.h* → arxiu de capçaleres en C dels mètodes natius.
- *FBK33200local.cpp* → arxiu amb la implementació dels mètodes natius i del mètode *DllMain*, necessari a totes les DLLs.

### 7.1.3. ESQUEMA D'INTERACCIÓ.

L'esquema de la següent pàgina il·lustra el funcionament de la classe des d'un punt de vista de capes de nivells d'interacció i control.


MAQUINA VIRTUAL JAVA
<Aplicacio>.class
FBK33200local.class
FBK33200local.dll
libeng.dll
Entorn MATLAB
pd_call.dll
RTK
Pclabcard 812-PG
P C
MAQUETA PONT-GRUA

## 7.2. COMUNICACIÓ ENTRE MAQUETA I USUARI EN EL CAS DE CONTROL REMOT.

### 7.2.1. DISSENY DE LA INTERFÍCIE REMOTA.

En el cas del control remot es defineix la interfície remota com una interfície IDL on els mètodes son d'idèntica sintaxi als de control local. Aquesta interfície en Java pren el nom de interfície *FBK33200remot.idl* . Els mètodes disponibles per a l'usuari remot son la totalitat dels que disposa l'usuari local amb les següents excepcions:

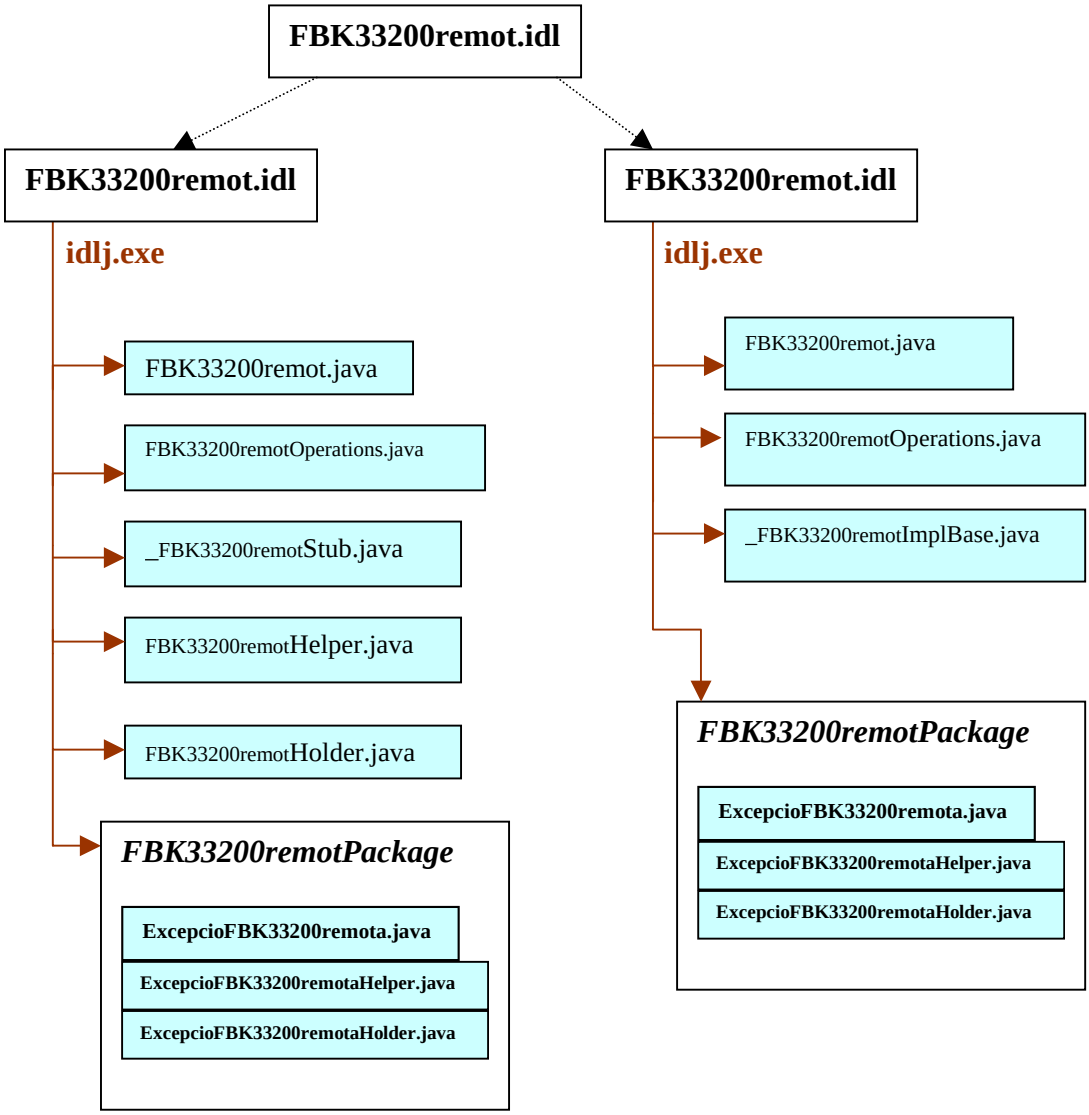
- El mètode `getAdrPCL812PG();` Ja que no té sentit que l'usuari remot li pugui interessar l'adreça base de la placa de comunicació de la màquina local.
- El mètode `setAdrPCL812PG(int);` Perquè l'usuari remot no ha de poder modificar allò que no depèn d'ell i només del local.
- El mètode `comptadorsAzero();` Perquè no té sentit que l'usuari remot pugui controlar el punt de referència dels comptadors, és a dir els punts que es consideraran 0 a partir d'un moment donat en l'execució del diversos algoritmes de control. Aquesta és una tasca que ha de ser competència exclusiva de l'usuari local.

A més a més i donat que en els diversos mètodes de la classe hi intervenen Arrays de números reals (float), s'haurà de definir aquest tipus a la implementació de l'arxiu *.idl.

S'haurà de crear una classe d'excepcions amb el procediment de la tecnologia IDL, que es llençaran en el cas que l'objecte local en llenci una del tipus *ExcepcioFBK33200* . Aquesta classe d'excepcions s'anomenarà *ExcepcioFBK33200remota* i el "mapeig" de IDL a Java la ubicarà en un paquet anomenat *FBK33200remotPackage* .

També s'implementaran, com en el cas de la classe local, les variables enteres finals de classe, que equiparen les sigles dels algoritmes de control al numero a ells associats.

L'esquema dels arxius generats a partir de l'arxiu de definició d'interfície remota serà el següent:



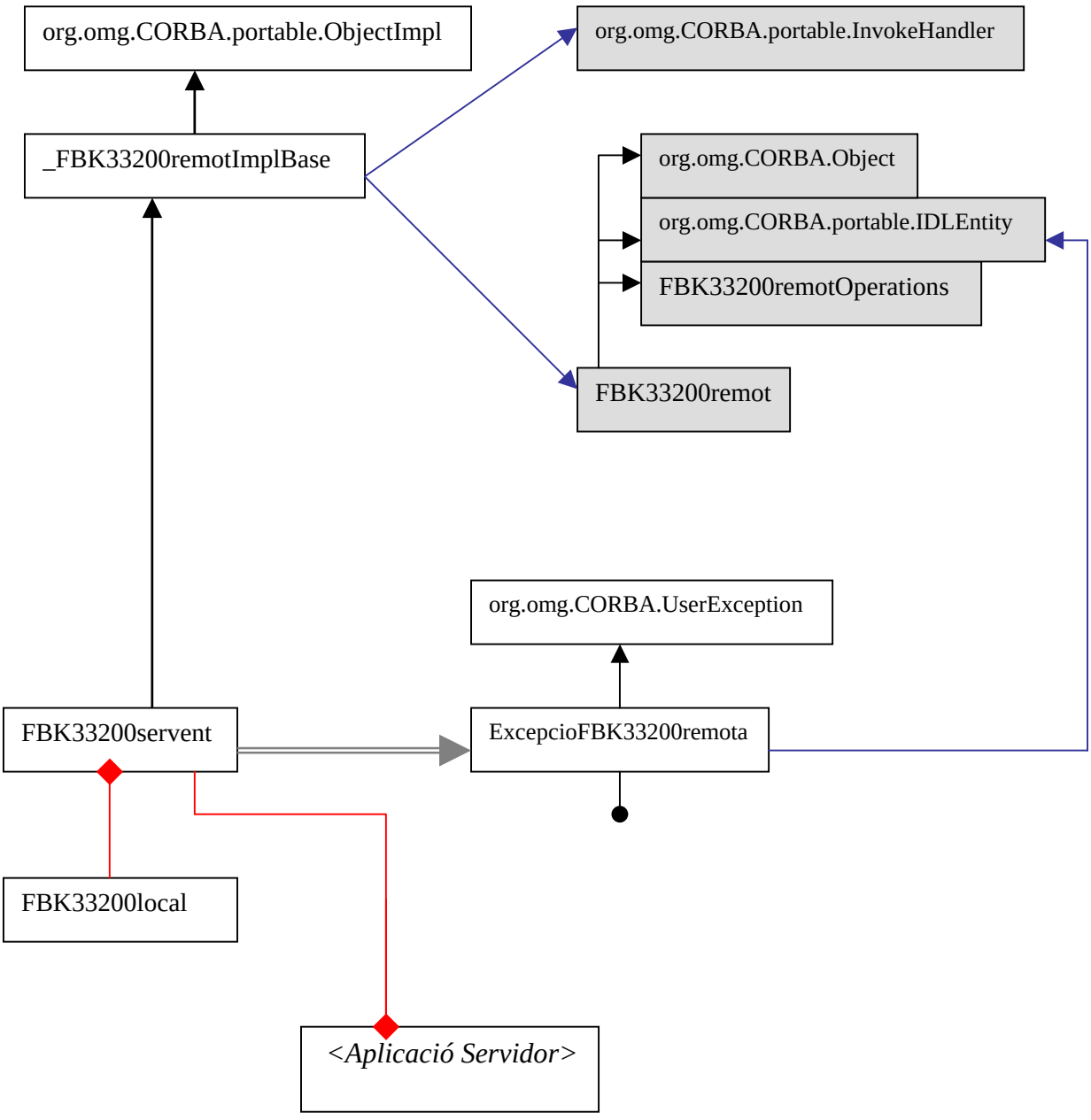

A partir de les respectives classes compilades el que manca és implementar la classe servent. Declarem aquesta classe, en aquest cas, *FBK33200servent* ,estenent la classe *_FBK33200remotImplBase.java* , és a dir declarant-la com a classe filla d'aquesta. Aquesta classe tindrà un atribut privat, que serà una instància de la classe *FBK33200local* . Els diversos mètodes invocaran els seus homòlegs del cas local sobre l'objecte de l'atribut privat. Es capturaran les possibles excepcions llançades durant aquesta invocació i en cas d'haver-se produït el llançament es procedirà al llançament de l'excepció remota. Tenint en compte que els objectes remots poden rebre peticions de diverses referències de forma simultània, es protegeix el codi amb la inclusió d'aquest entre les claus del bloc:

```
synchronized(this) {
      //codi....
}
```

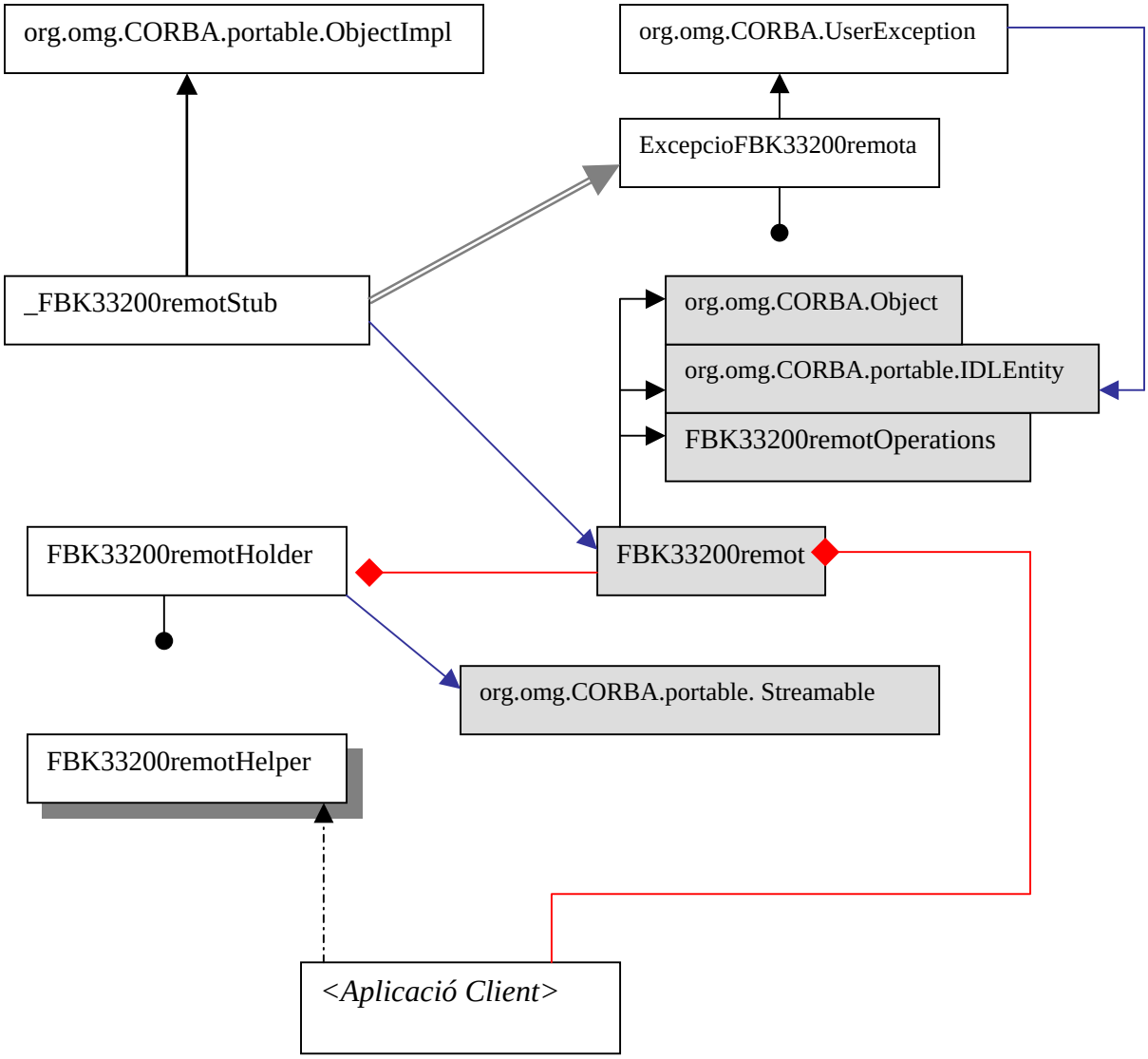
A partir d'aquí ja es pot implementar el servidor i el client de l'aplicació.

El següent diagrama de classes il·lustra l'estructura d'aquestes i la relació entre elles.

*Servidor*:



*Client*:




On:

Representa continència d'un objecte de la classe en els objectes de l'altra

Representa herència de classes i interfícies (les segones poden tenir herència múltiple)

Representa implementació d'interfície

Representa excepcions que llencen algun dels mètodes de la classe.

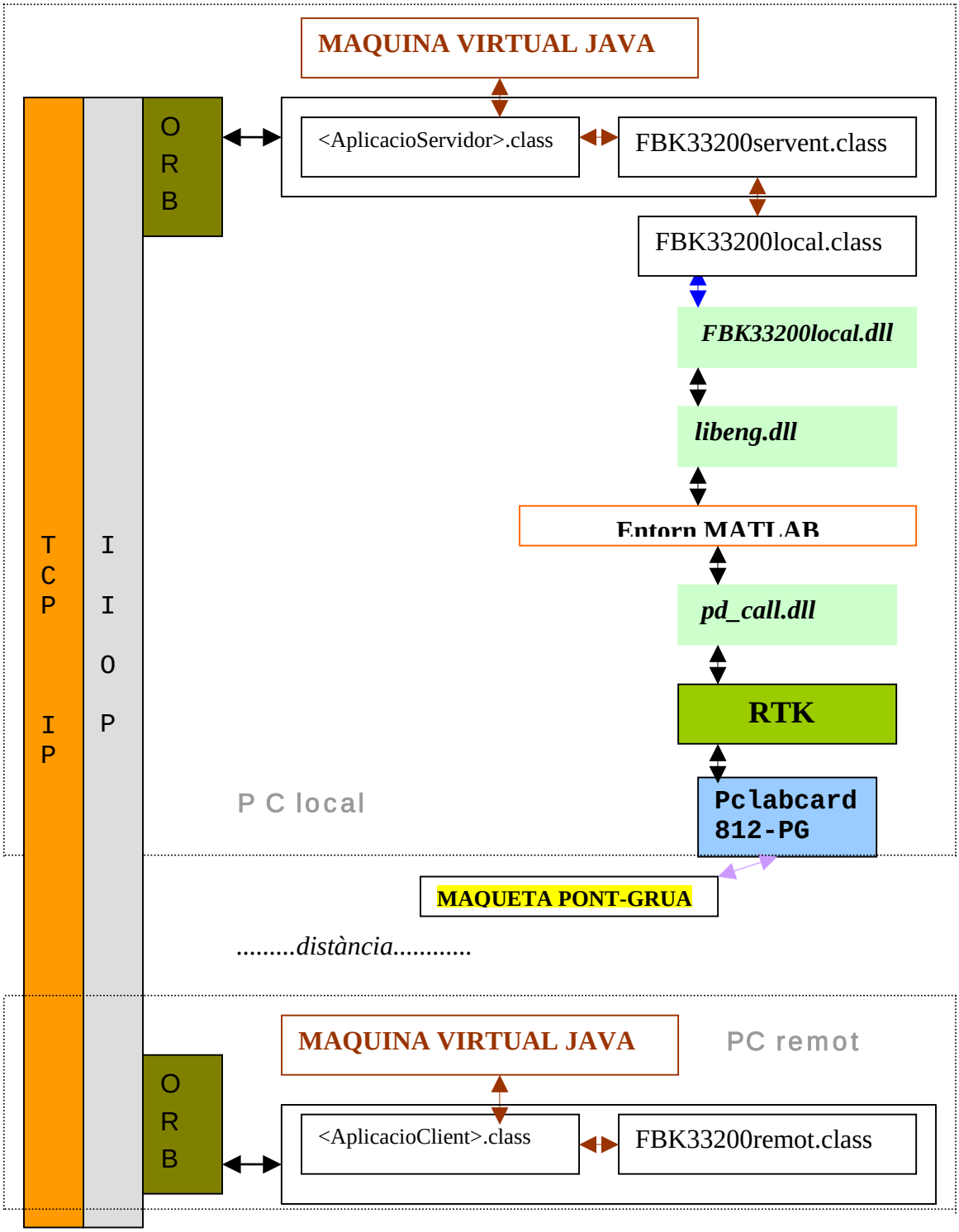
Representa que la classe és final. (no es pot estendre)

Representa que es criden mètodes de la classe des de l'aplicació.

Els rectangles grisos representen interfícies, els blancs classes i els ombrejats classes abstractes.

### 7.2.2. ESQUEMA D'INTERACCIÓ.

El següent esquema il·lustra el funcionament de la interfície de control remot des d'un punt de vista de capes de nivells d'interacció i control:



# 7.3. MODELAT I CONTROL DEL PROCÉS.

## 7.3.1. MODELAT DEL PROCÉS.

Si es considera el procés com un sistema discret format per un procés de 1er ordre amb un integrador en la forma

$$Gp = \frac{K_p}{s(\tau * s + 1)},$$

un mostrejador, que actua amb un període de 0.01 seg. i un mantenidor d'ordre 0, s'obté el model equivalent discret del procés continu com un amb una FT discreta de la forma:

$$Gp(z) = \frac{a * z + b}{(z + c) * (z - 1)}$$

Desenvolupant el polinomi del denominador s'obté una FT en la forma:

$$Gp(z) = \frac{a * z + b}{z^2 + c * z + d} \quad \text{on } Gp(z) = \frac{X(z)}{M(z)}$$

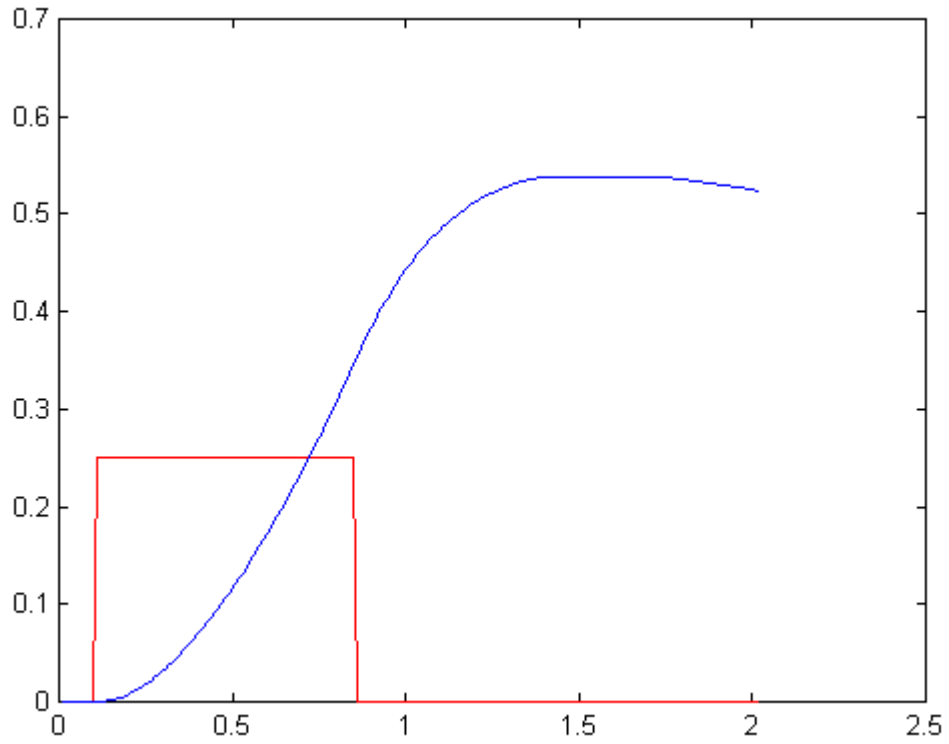
sent M(z) l'acció de control i X(z) la sortida, en aquest cas la posició del carro.

En el format d'equació en diferències seria:

x[k] = a*m[k-1] + b*m[k-2] – c*x[k-1] – d*x[k-2]

En el procés de modelat es parteix de la realització d'un experiment en llaç obert amb el procés. En aquest experiment, de duració 2 seg., s'excita el procés amb una senyal impuls d'amplitud 0.25 que comença a t = 0.1 seg. i acaba a t = 0.85 seg. L'arxiu de l'algoritme de l'experiment és *LlacObert.m* .

L'evolució temporal de la sortida i l'entrada en funció del temps es pot apreciar al següent gràfic:




A partir de les mostres obtingudes en els diversos instants, el problema de l'obtenció dels paràmetres a, b, c i d de la funció de transferència discreta resulta en la resolució d'un sistema d'equacions, on aquestes son:

x[k] = a*m[k-1] + b*m[k-2] – c*x[k-1] – d*x[k-2]

per k=0 fins k=nm , on nm son el numero de mostres que es tinguin.

Donat que el numero d'incògnites és de 4, amb aquesta quantitat d'equacions n'hi hauria prou. Però tenint en compte que les mostres de la sortida poden estar brutes de soroll i que el model no deixa de ser aproximat, la millor resolució és aquella que contempli totes les equacions i calculi els valors de les incògnites amb algun criteri de minimitzi el possible error.

Un sistema possible és el dels mínims quadrats, que calcula els valors de les incògnites perquè es compleixin les relacions en el màxim nombre d'equacions i en aquelles en que no, el quadrat de l'error sigui el més petit possible.

La formula que, amb aquest criteri, permet obtenir els valors de les incògnites a partir de la taula de mostres és:

$$\left[(\phi^T * \phi)^{-1} * \phi^T\right] * \hat{V} = \hat{C}$$

on $\hat{V}$ és el vector amb les mostres dels diversos instants de temps, els termes independents des de l'òptica de sistemes d'equacions lineals ,

$\hat{C}$ és el vector amb els coeficients que interessa trobar (les incògnites) i $\left[(\phi^T * \phi)^{-1} * \phi^T\right]$ és la matriu pseudo-inversa de $\phi$ , essent aquesta la que conté els valors dels coeficients de les incògnites; en aquest cas els valors de les mostres anteriors al de mesura de la sortida, d'acord amb la formula.

Amb aquest procediment obtenim els següents valors dels paràmetres:

a = -0.001

b = 0.0021

c = -1.9642

d = 0.9642

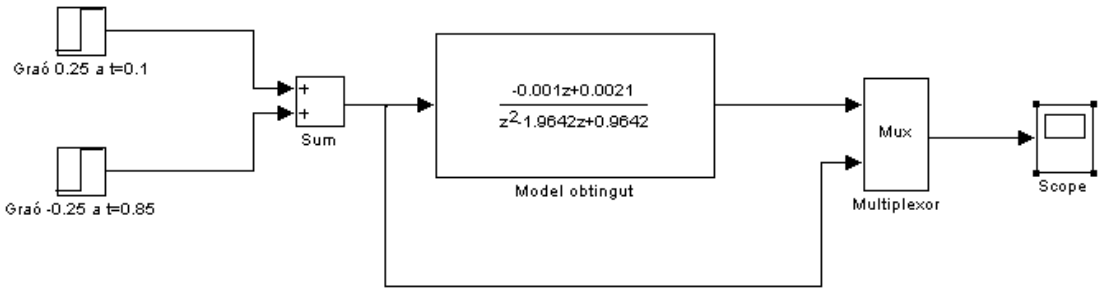
De manera que la FT discreta seria:

$$Gp(z) = \frac{-0.001 * z + 0.0021}{z^2 - 1.9642 * z + 0.9642} = \frac{-0.001 * z + 0.0021}{(z - 0.9642) * (z - 1)}$$

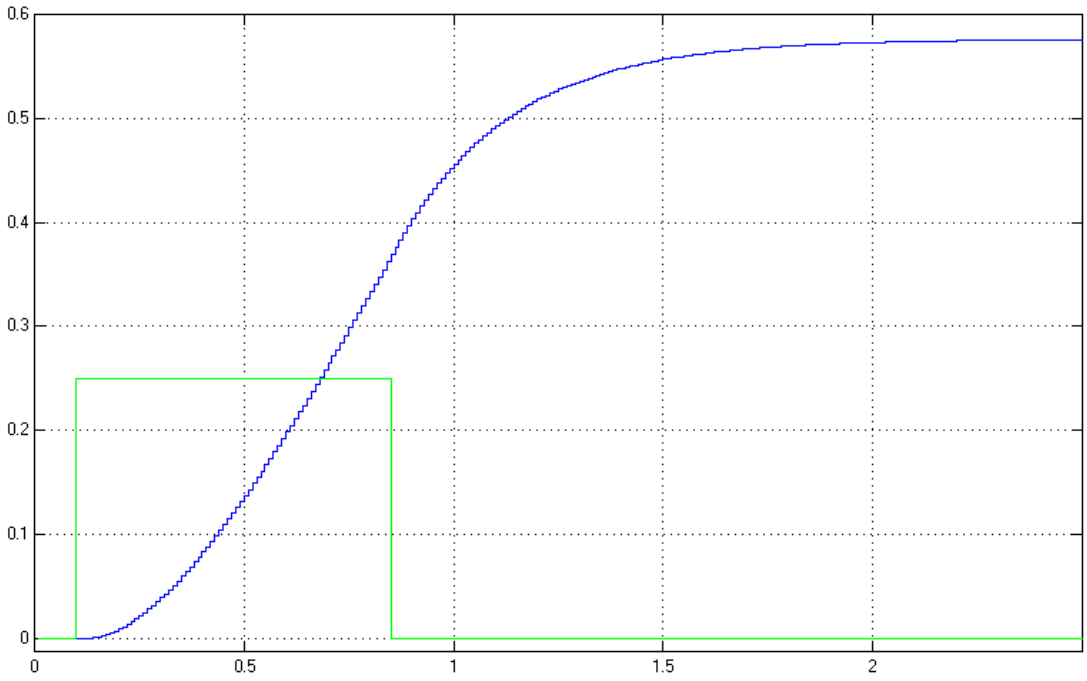
Abans de considerar vàlid aquest model realitzem la simulació del mateix, davant d'una entrada igual a la de l'experiment sobre el procés real i observem la sortida.

Per fer-ho recorrem a **Simulink**.

La figura del model de simulació és:




Després de realitzar la simulació, obtenim l'evolució temporal de la sortida i l'entrada en funció del temps.




S'observa que el resultat és prou semblant a l'obtingut amb el procés real, cosa que indica que aquest model és prou adequat.

### 7.3.2. CONTROL DEL PROCÉS.

De cara al disseny del controlador, és a dir, a l'obtenció dels paràmetres de l'algoritme PID, prèviament a imposar una especificació amb l'assignació de pols, s'ha de determinar la rapidesa natural del procés. S'entén com a tal el temps que triga la sortida a assolir el 63.2 % del seu valor en règim permanent.

Si es pren com a sortida la posició, aquest valor pot ser no tant directe d'obtenir, ja que al posseir un integrador el valor en règim permanent és $\infty$ . De manera que es prendrà com a sortida la velocitat lineal del carro. La FT que relaciona l'acció de control i la velocitat s'obté derivant la FT que relaciona acció de control amb posició del carro.

$$Gv(z) = \frac{-0.001*z+0.0021}{(z-0.9642)*(z-1)} * \frac{(z-1)}{T_S * z} = \frac{-0.1*z+0.21}{z*(z-0.9642)}$$

on Gv(z) és la FT que relaciona acció de control i velocitat del carro.

Per obtenir el valor en règim permanent per una entrada d'un graó d'amplitud 0.25, ja que aquesta és l'amplitud de la senyal amb la que hem excitat el sistema en l'experiment, apliquem el teorema del valor final per al cas discret:

$$\lim_{Z \to 1} (Gv(z) * 0.25) = 0.7682$$

Amb els valors obtinguts en l'experiment observem l'instant en que la velocitat del carro arriba al 63.2% de 0.7682, el valor 0.4855 . La velocitat triga a assolir aquest valor al voltant d'uns 0.36 segons.

De manera que les especificacions que s'hagin d'imposar en el disseny del controlador hauran d'especificar una rapidesa igual o inferior a 0.36 segons.

A l'hora de determinar el tipus de regulador a utilitzar d'entre els que permet l'algoritme PID (P, I, PI, PD o PID) s'opta per un PD, perquè l'acció integral té la funció d'eliminar l'error en règim permanent y els sistemes amb un integrador, com el present, ja ho asseguren per si sols en llaç tancat.

Tot y lo exposat, el fet de no posseir acció integral al regulador pot fer impossible el rebuig, en règim permanent, d'una pertorbació de tipus graó. No obstant una pertorbació de tipus graó seria el cas, en que algú exercís una força en sentit oposat al moviment del carro de forma permanent, a partir d'un instant. I inclòs, havent-hi acció integral al regulador, aquesta pertorbació no seria rebutjada si en amplitud superés la força màxima que pot exercir el motor del carro (17.5 N). Mentre que una pertorbació de tipus pols de temps finit, si seria rebutjada en règim permanent.

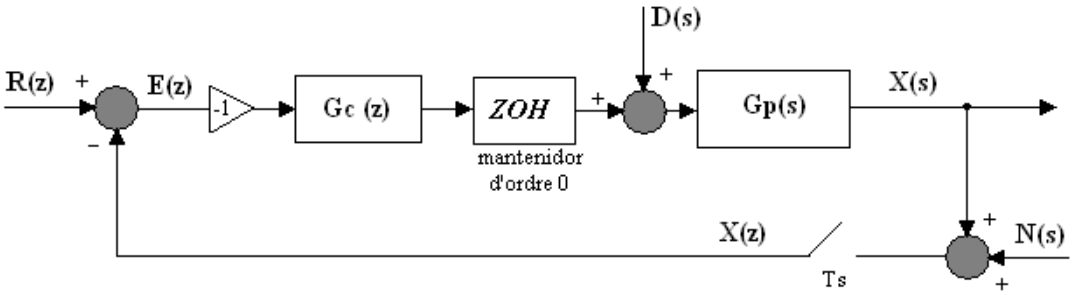
Un cop determinat el tipus de regulador, es procedeix a modelar matemàticament. El model del regulador PD seria de la forma:

$$Gc = Kp + Kd * (\frac{z-1}{z})$$

És en aquest punt quan es detecten dos errors notables en la redacció del manual del fabricant:

1. Segons el manual la formula de l'algoritme PID de la llibreria del RTK de la maqueta és $Gc_{PID} = Kp + Ki * (\frac{1}{z-1}) + Kd * (\frac{z-1}{z})$. És a dir que l'acció integral y derivativa son accions acumulativa y diferenciadora respectivament, entre dos instants independentment del període de mostreig. Això no és cert ja que tant l'acció integral com derivativa tenen en compte el període de mostreig y per tant això s'ha de tenir en compte a l'hora de modelar el regulador.

2. A l'entrada del regulador, en el cas de l'algoritme PID, s'inverteix el valor de la senyal, de forma que es produeix l'efecte que es produiria si hi hagués un guany de –1 entre el comparador i el regulador. D'aquesta manera els valors obtinguts en el procés de disseny de Kp i Kd, hauran d'invertir llurs signes per tal que el control resulti l'esperat.




D'acord amb tot això y oblidant-se del signe fins al final, el model del regulador PD seria de la forma:

$$Gc = Kp + Kd * (\frac{z-1}{T_S * z})$$

Es procedeix a la resolució matemàtica del llaç de control per obtenir el polinomi característic, p(z) que és el denominador de X(z)/R(z) i de E(z)/D(z)

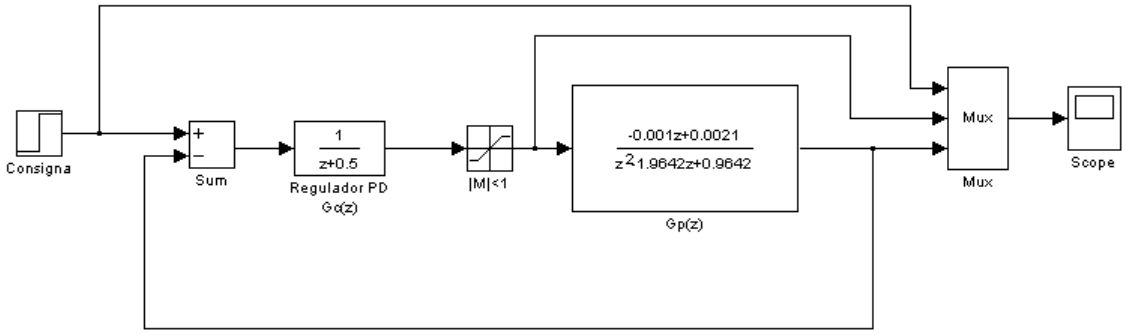
$$p(z) = Np * Nc + Dp * Dc$$

on Nc és el numerador de la FT del procés ,
Nc és el numerador de la FT del controlador ,
Dp és el denominador de la FT del procés i
Dc és el denominador de la FT del controlador.

$$p(z) = z^3 - (0.001Kp + 0.1Kd + 1.9642)z^2 + (0.0021Kp + 0.31Kd + 0.9642)z - 0.21Kd$$

**Simulació prèvia:**

Posteriorment a l'implementació sobre el procés real del controlador calculat es procedirà a la simulació del resultat amb el model calculat, però tenint en compte la saturació de l'acció de control, que en el cas del pont-grua és de 1 en valor absolut. Amb **Simulink** la figura del model de simulació és:




on la FT discreta que aquí apareix com a 1/z-0.5 se substituirà per la FT d'un regulador PD discret amb els valors calculats de Kp i Kd.

**Especificació:**

Comencem especificant un seguiment del mateix ordre de rapidesa que el procés. Especifiquem $\tau_e = 0.4seg$ .

Els pols que s'haurien d'especificar en el cas del domini continu serien:

$p_c = -(1/0.4) = -2.5$ .

En el domini discret aquests pols seran:

$p_d = \ell^{(pc*Ts)} = 0.975$

Per tal de especificar matemàticament el polinomi característic a partir d'aquest pol tenint en compte que p(z) és de tercer ordre, i tenint en compte que amb l'algoritme PD tenim 2 paràmetres lliures (Kp i Kd) s'haurà d'especificar el pol doblat i especificar un pol paràsit $p_p$ , que és necessari perquè es compleixi l'equació i que posteriorment comprovarem si afecta al disseny. El polinomi característic especificat és doncs:

$$(z-0.975)^2 * (z-p_p)$$

Igualem els polinomis:

$$(z-0.975)^2 * (z-p_p) = p(z)$$

i obtenim els valors:

$p_p = 0.02696$

$Kp = 0.5529$

$Kd = 0.122$

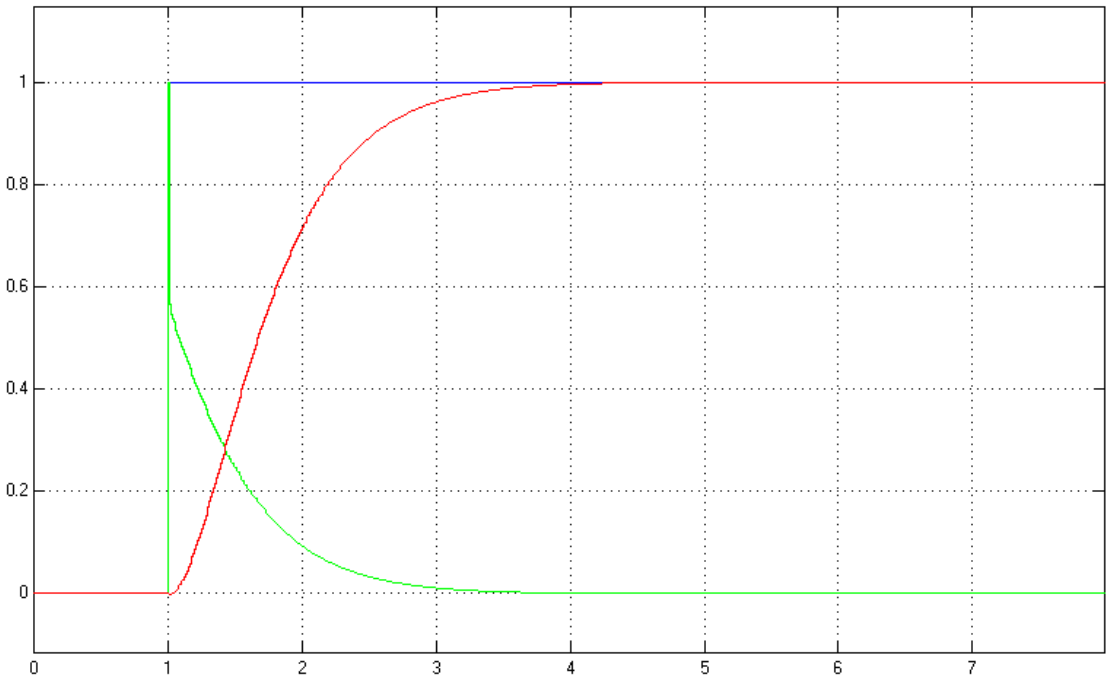
Comprovem el pol paràsit

$$p_c = \frac{\ln(p_d)}{T_S} = \frac{\ln(0.02696)}{0.01} = -361.34$$

$$|-361.34| \geq |8*(-2.5)|$$

de manera que respecte als pols especificats és totalment despreciable i no afecta l'estabilitat

Procedim a la simulació, i obtenim la resposta temporal:




*la línia blava és la consigna, la verda l'acció de control i la vermella la sortida.*

Observem com l'acció de control es satura a l'inici, per tant relaxem l'especificació. Especifiquem $\tau_e = 0.5seg$ .

Els pols que s'haurien d'especificar en el cas del domini continu serien:

$p_c$ = -(1/0.5) =-2 .

En el domini discret aquests pols seran:

$$p_d = \ell^{(pc*Ts)} = 0.9802$$

El polinomi característic especificat és doncs:

$$(z-0.9802)^2 * (z-p_p)$$

Igualem els polinomis:

$$(z-0.9802)^2 * (z-p_p) = p(z)$$

i obtenim els valors:

$p_p$ = 0.0077

Kp = 0.352

Kd = 0.035

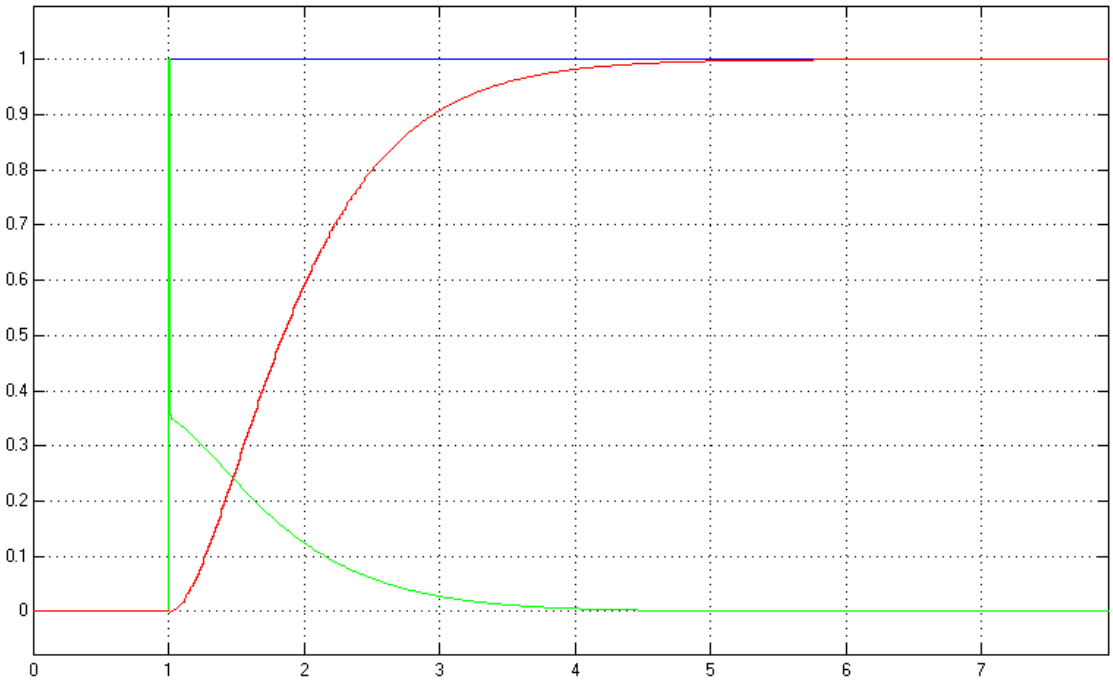
Comprovem el pol paràsit

$$p_c = \frac{\ln(p_d)}{T_S} = \frac{\ln(0.0077)}{0.01} = -486.65$$

$$|-486.65| \geq |8*(-2)|$$

de manera que respecte als pols especificats és totalment despreciable i no afecta l'estabilitat

Procedim a la simulació, i obtenim la resposta temporal:




Observem com l'acció de control se segueix saturant a l'inici, per tant relaxem encara més l'especificació. Especifiquem $\tau_e = 0.55seg$ .

Els pols que s'haurien d'especificar en el cas del domini continu serien:

$p_c$ = -(1/0.55) =-0.8182 .

En el domini discret aquests pols seran:

$$p_d = \ell^{(pc*Ts)} = 0.98198$$

El polinomi característic especificat és doncs:

$$(z-0.98198)^2 * (z-p_p)$$

Igualem els polinomis:

$$(z-0.98198)^2 * (z-p_p) = p(z)$$

i obtenim els valors:

$p_p = 0.000998$

Kp = 0.2997

Kd = 0.0046

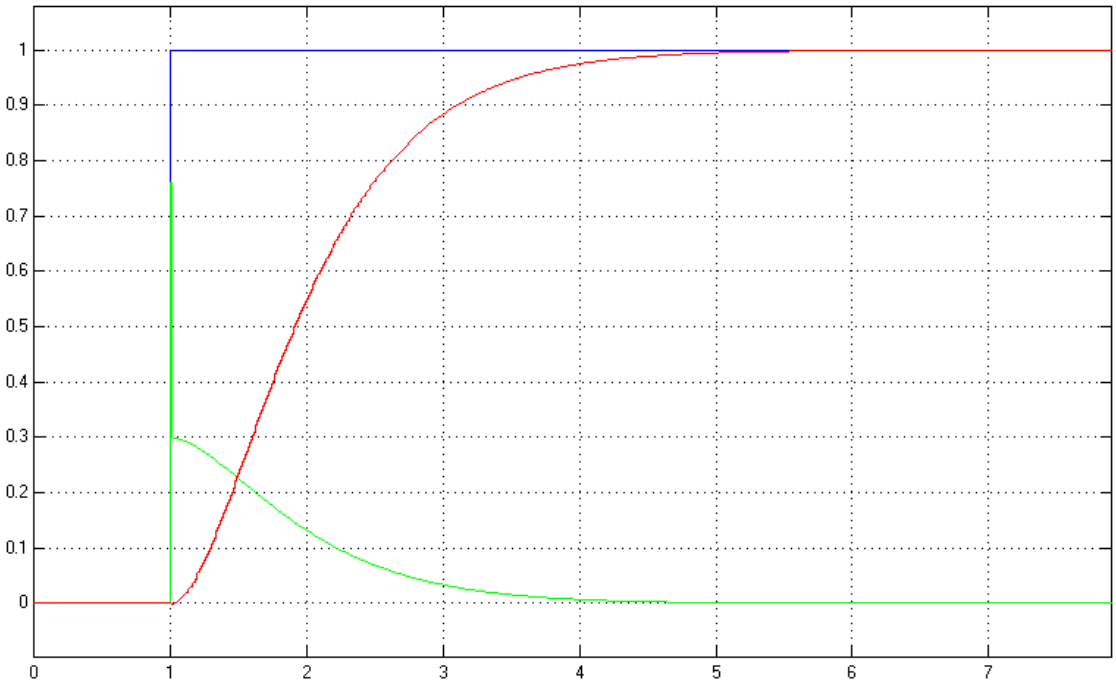
Comprovem el pol paràsit

$$p_c = \frac{\ln(p_d)}{T_S} = \frac{\ln(0.000998)}{0.01} = -690.98$$

$$|-690.98| \geq |8*(-0.8182)|$$

de manera que respecte als pols especificats és totalment despreciable i no afecta l'estabilitat

Procedim a la simulació, i obtenim la resposta temporal:

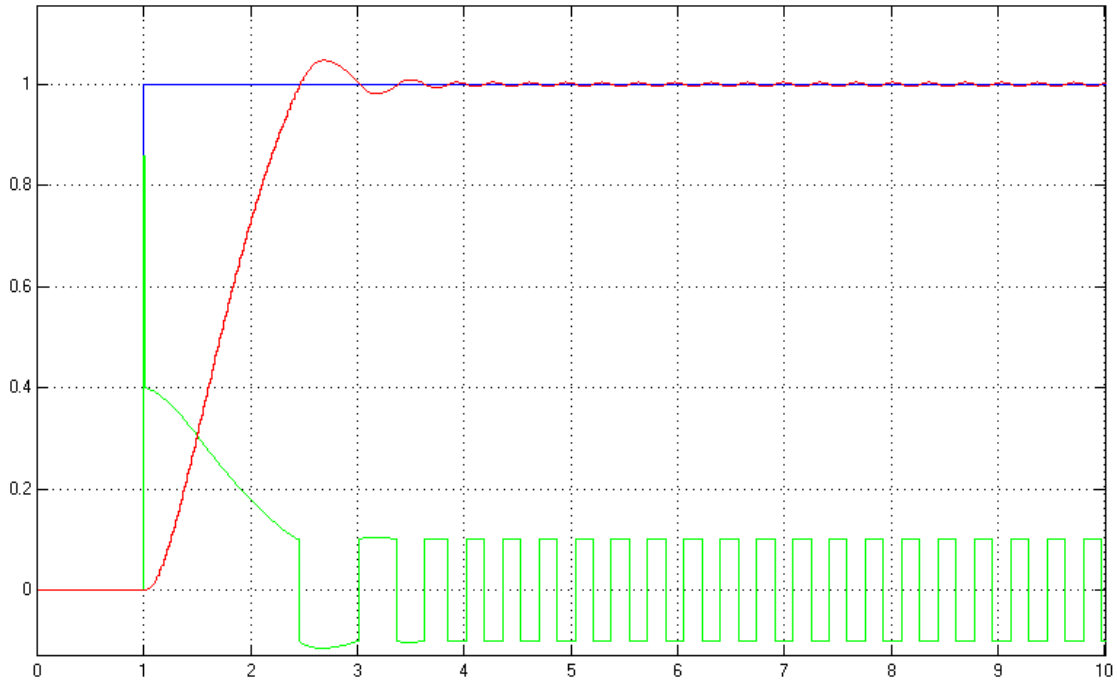


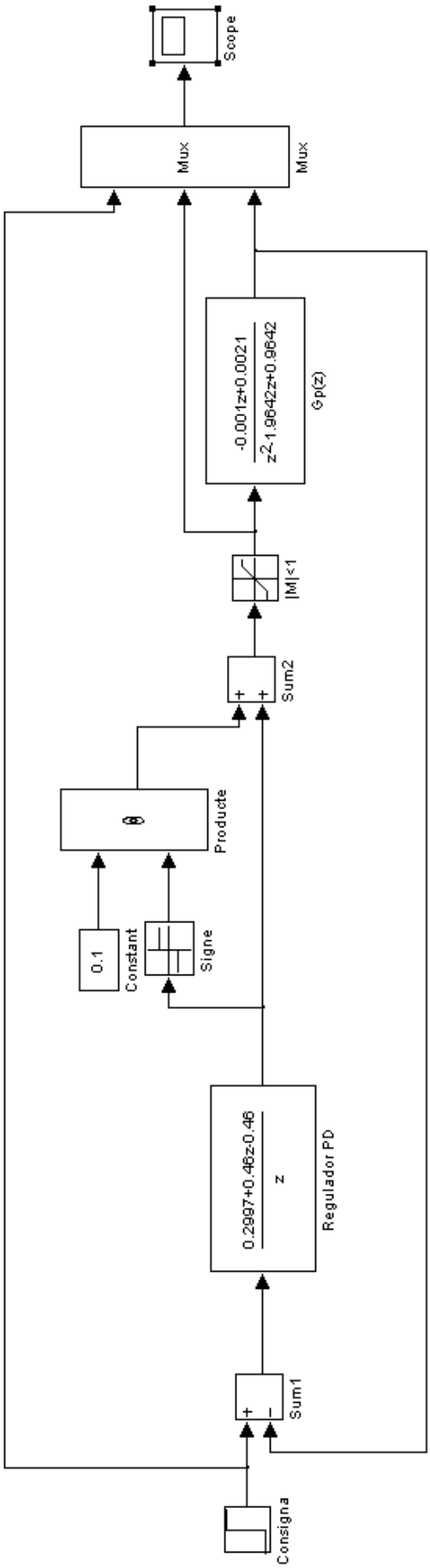

Observem com ara l'acció de control no se satura pas i el temps de resposta és de l'ordre del doble de l'especificat al pol ja que el pol estava doblat a l'especificació. Per tant considerem aquests paràmetres vàlids, i passem a comprovar el resultat sobre el procés real.

S'observa com la resposta té una evolució semblant a la de la simulació, però hi ha un error en règim permanent. A més a més aquest error es fa més

palès com més petita és la consigna, arribant fins i tot a no arrancar el carro si aquesta és del ordre de 0.1. Si tenim en compte que el rang de consignes possibles és de –0.5 a 0.5 i que per tant una resolució superior a 0.1 no és recomanable de cara a obtenir mínimes prestacions, optem per afegir a l'acció de control un "offset" que ens permet l'algoritme PID de la llibreria del RTK. Aquest valor, que es defineix com a valor mínim de l'Acció de Control (en valor absolut) per vèncer la fricció estàtica del carro ($|M_{min}|$), és el primer argument de la funció que assigna els valors dels paràmetres del controlar. El que fa és sumar a l'acció de control calculada el valor indicat amb el seu signe. De manera que si $M_{calc} = 0.55 \rightarrow M=0.55 + |M_{min}|$ , o si $M_{calc} = -0.55 \rightarrow M=0.55 - |M_{min}|$. D'acord amb el manual de la maqueta aquest valor està al voltant de 0.1.

Realitzem prèviament, la simulació incloent-hi aquesta variació. Amb **Simulink** la figura del model de simulació és la que s'il·lustra a la pàgina següent i la resposta a una consigna graó unitari, la que s'il·lustra a continuació:

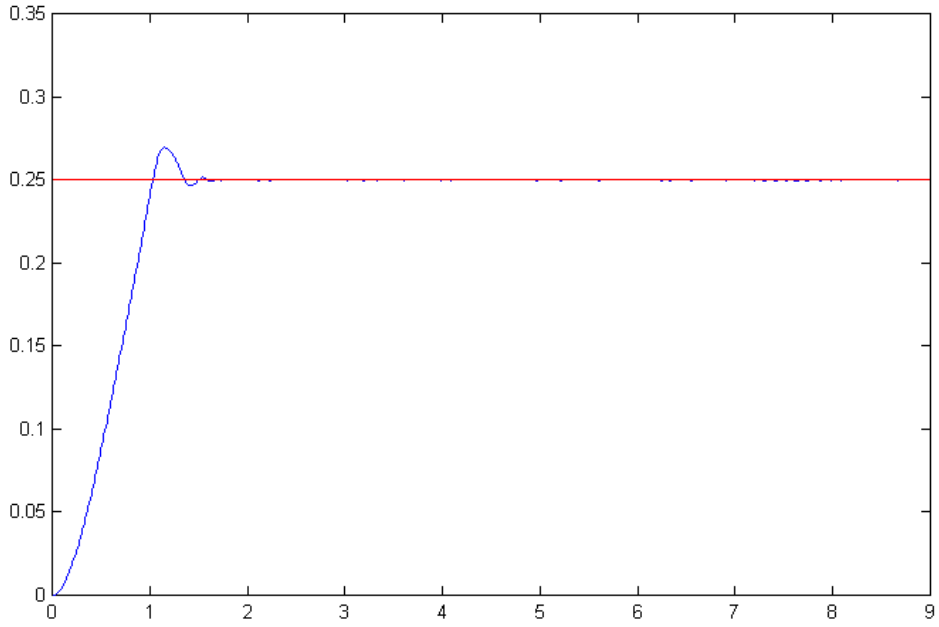



Scope
Mux
Mux
-0.001z+0.0021
z^2-1.9642z+0.9642
Gp(z)
|M|<1
Sum2
Producte
0.1
Constant
Signe
0.2997+0.46z-0.46
z
Regulador PD
Sum1
Consigna

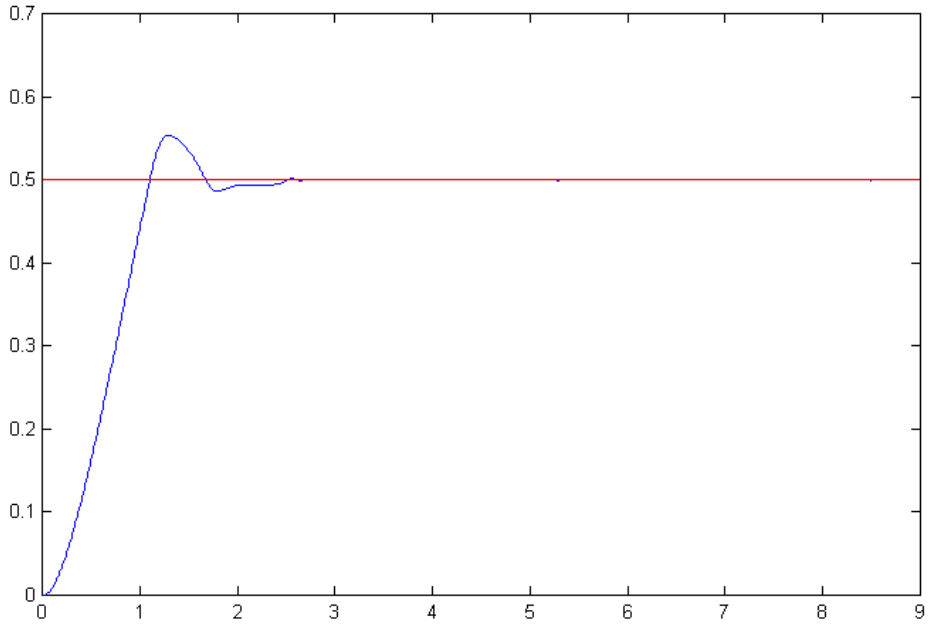
Observem en la resposta temporal com hi ha un petit sobre-pic, malgrat que l'acció de control no se satura per consignes unitàries, que son les d'amplitud més gran amb que es pot trobar el sistema ja que el rail té una longitud d'un metre de llarg. Considerem aleshores, el sobre-pic com una petita reducció de les prestacions inicialment anhelades, amb la compensació d'aconseguir error nul en règim permanent i garantir el venciment de la inèrcia per part del carro per a petites consignes.

Passem doncs a comprovar el resultat sobre el procés real i obtenim la següent resposta temporal.

*Resposta davant una entrada graó d'amplitud 0.25:*



*Resposta davant una entrada graó d'amplitud 0.25:*




Observem, doncs, com el sobre-pic és del mateix ordre que en el cas de la simulació, de l'ordre d'entre un 8% i un 10% i que s'anul·la l'error en règim permanent.

Al ésser el sobre-pic d'aquest ordre s'haurà de tenir en compte de no fixar consignes massa a prop del valor de posició límit (+/- 0.5), ja que aquest sobre-pic provocaria que el carro sortís dels límits i els finals de carrera desconnectessin l'actuador. El límit pràctic de consignes estaria al voltant de les posicions +/- 0.4, prenent el centre del rail com a punt 0.

# 8. RESULTATS.

## 8.1. CLASSE *FBK33200local.*

La classe resultant del disseny explicat al primer apartat del capítol anterior, la classe *FBK33200local* , és una classe que dona uns serveis anàlegs als que li poden ser subministrats a l'usuari treballant des de l'entorn de Matlab.

Els mètodes de la classe, llancen excepcions de la classe *ExcepcioFBK33200* en cas d'haver algun error, amb el missatge d'error descriptiu del mateix d'acord amb la relació de possibles errors descrita al capítol 7.1.2.

Per a ésser utilitzada la classe s'ha de disposar del fitxer de classe compilat *FBK33200local.class* , el de la classe d'excepció llançada *ExcepcioFBK33200.class* , com tots els casos en Java i a més a més el de la llibreria dinàmica amb la implementació dels mètodes natius de la classe *FBK33200local.dll*, que s'ha de trobar al directori de treball o en un directori inclòs a la ruta de directoris de cerca d'executables (**path**).

Per tal d'il·lustrar els serveis disponibles per aquesta classe, a continuació es llista la interfície publica de la mateixa amb una breu descripció de les tasques realitzades per cada mètode, els seus arguments i valors retornats.

| ***CONSTRUCTOR per defecte*** | public FBK33200local()throws ExcepcioFBK33200 |
|---|---|
| **Tasca realitzada** | Crea una instància de la classe amb l'adreça de la targeta PCL812-PG configurada prèviament al RTK. A més, carrega l'entorn de Matlab, fixa l'acció de control a 0, esborra el contingut del Buffer i fixa la posició lineal i angular en aquell instant com a 0 i $\pi$ rad. Respectivament. |
| **Arguments** | - |

| ***CONSTRUCTOR*** | public FBK33200local(int adrecaPCL812PG) throws ExcepcioFBK33200 |
|---|---|
| **Tasca realitzada** | Crea una instància de la classe amb l'adreça de la targeta PCL812-PG de l'argument. A més, carrega l'entorn de Matlab, fixa l'acció de control a 0, esborra el contingut del Buffer i fixa la posició lineal i angular en aquell instant com a 0 i $\pi$ rad. Respectivament. |
| **Arguments** | Adreça de la targeta en format **int**. |

| ***FINALITZADOR*** | public void finalize() |
|---|---|
| **Tasca realitzada** | Descarrega l'entorn de Matlab, si aquest ha estat prèviament carregat. |
| **Arguments** | - |
| **Retorn** | - |

| **PROTOTIPUS** | public native int getTipusControl()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna el numero de l'algoritme de control actiu al RTK, d'acord amb les equivalències indicades a les taules de variables finals de la classe. |
| **Arguments** | - |
| **Retorn** | **int** que representa l'algoritme de control actiu al RTK. |

| **PROTOTIPUS** | public native int getAdrPCL812PG()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna l'adreça base de la targeta de comunicació PCL812-PG. |
| **Arguments** | - |
| **Retorn** | **int** amb l'adreça. |

| **PROTOTIPUS** | public native float[] getParFiltrePosCarro()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna els paràmetres del filtre del RTK de la senyal de posició del carro. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors dels paràmetres, d'acord amb el format anàleg a l'emprat en Matlab. |

| **PROTOTIPUS** | public native float[] getParFiltreVelCarro()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna els paràmetres del filtre del RTK de la senyal de velocitat del carro. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors dels paràmetres, d'acord amb el format anàleg a l'emprat en Matlab. |

| **PROTOTIPUS** | public native float getConsigna()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna la consigna de posició del carro vigent al RTK. |
| **Arguments** | - |
| **Retorn** | Valor de la consigna en format **float** . |

| **PROTOTIPUS** | public native int getDivisorRTK()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna el valor del divisor del rellotge auxiliar del RTK. |
| **Arguments** | - |
| **Retorn** | Valor del divisor en format **int** . |

| **PROTOTIPUS** | public native float[] getHistorialTemps()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb l'historial d'instants de temps (en segons) continguts al Buffer del RTK. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native float[] getHistorialAnglePendul()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb l'historial de valors de l'angle del pèndul continguts al Buffer del RTK. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native float[] getHistorialVelPendul()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb l'historial de valors de velocitat angular del pèndul continguts al Buffer del RTK. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native float[] getHistorialPosCarro()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb l'historial de valors de la posició lineal del carro continguts al Buffer del RTK. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native float[] getHistorialVelCarro()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb l'historial de valors de velocitat lineal del carro continguts al Buffer del RTK. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native float[] getHistorialAccioControlRelativa() throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb l'historial de valors de l'acció de control relativa (-1::+1) continguts al Buffer del RTK. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native float[] getHistorialConsigna() throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb l'historial de valors de la consigna de posició lineal del carro continguts al Buffer del RTK. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native int getNumMostresBufer() throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna el numero de mostres del Buffer del RTK (la capacitat màxima d'aquest és de 1000). |
| **Arguments** | - |
| **Retorn** | Valor en format **int** . |

| **PROTOTIPUS** | public native float[] getParametresControl() throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb els paràmetres de l'algoritme de control. El significat dels elements és anàleg al cas de Matlab. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native float[] getParFiltrePosPendul() throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna els paràmetres del filtre del RTK de la senyal de posició angular del pèndul. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors dels paràmetres, d'acord amb el format anàleg a l'emprat en Matlab. |

| **PROTOTIPUS** | public native float[] getParFiltreVelPendul() throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna els paràmetres del filtre del RTK de la senyal de velocitat angular del pèndul. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors dels paràmetres, d'acord amb el format anàleg a l'emprat en Matlab. |

| **PROTOTIPUS** | public native float[] getPW()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna un vector amb els paràmetres de l'algoritme d'excitació en llaç obert. El significat dels elements és anàleg al cas de Matlab. |
| **Arguments** | - |
| **Retorn** | Vector de **float** amb els valors. |

| **PROTOTIPUS** | public native float getPeriodeMostreig()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Retorna el període de mostreig a partir del qual treballen els algoritmes del RTK. |
| **Arguments** | - |
| **Retorn** | Període en segons i format **float** . |

| **PROTOTIPUS** | public native int carregaLlibreria()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Carrega la llibreria del RTK en memòria i retorna el valor del comptador del mòdul. |
| **Arguments** | - |
| **Retorn** | Valor del comptador en format **int**. |

| **PROTOTIPUS** | public native void carregaControladorExtern<br>(String fitxer)throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Carrega en memòria l'algoritme de control extern implementat a la llibreria, el nom i ruta de la qual s'especifica a l'argument. |
| **Arguments** | Nom i ruta de la llibreria en format **String** . |
| **Retorn** | - |

| **PROTOTIPUS** | public native void comptadorsAzero()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa el valor dels comptadors de posició i angle a 0 a l'instant en què s'invoca. |
| **Arguments** | - |
| **Retorn** | - |

| **PROTOTIPUS** | public native void tempsAzero()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa el valor del comptador de temps a 0 a l'instant en què s'invoca. |
| **Arguments** | - |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setTipusControl(int tipus)<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa l'algoritme de control actiu al RTK. |
| **Arguments** | **int** que representa l'algoritme de control, d'acord amb les equivalències indicades a les taules de variables finals de la classe. |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setAdrPCL812PG(int adreca)<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa l'adreça base de la targeta de comunicació PCL812-PG. |
| **Arguments** | L'adreça en format **int** . |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setParFiltrePosCarro<br>(float[] parametres)throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa els paràmetres del filtre del RTK de la senyal de posició del carro. |
| **Arguments** | Vector de **float** amb els valors dels paràmetres, d'acord amb el format anàleg a l'emprat en Matlab. |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setParFiltreVelCarro<br>(float[] parametres)throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa els paràmetres del filtre del RTK de la senyal de velocitat lineal del carro. |
| **Arguments** | Vector de **float** amb els valors dels paràmetres, d'acord amb el format anàleg a l'emprat en Matlab. |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setConsigna(float r)<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa la consigna de la posició lineal del carro a l'algoritme del RTK actiu en l'instant en què s'invoca. |
| **Arguments** | Valor de la consigna en format **float** . |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setDivisorRTK(int divisor)<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa el valor del divisor del rellotge auxiliar del RTK. |
| **Arguments** | Valor del divisor en format **int** . |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setParametresControl (float[] parametres)throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa els paràmetres de l'algoritme de control. |
| **Arguments** | Vector de **float** amb els valors. El significat dels elements és anàleg al cas de Matlab. |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setParFiltrePosPendul (float[] parametres)throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa els paràmetres del filtre del RTK de la senyal de posició angular del pèndul. |
| **Arguments** | Vector de **float** amb els valors dels paràmetres, d'acord amb el format anàleg a l'emprat en Matlab. |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setParFiltreVelPendul (float[] parametres)throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa els paràmetres del filtre del RTK de la senyal de velocitat angular del pèndul. |
| **Arguments** | Vector de **float** amb els valors dels paràmetres, d'acord amb el format anàleg a l'emprat en Matlab. |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setPW(float[] parametres) throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa els paràmetres de l'algoritme d'excitació en llaç obert.. |
| **Arguments** | Vector de **float** amb els valors. El significat dels elements és anàleg al cas de Matlab. |
| **Retorn** | - |

| **PROTOTIPUS** | public native void setPeriodeMostreig(float ts) throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa el període de mostreig a partir del qual treballaran els algoritmes del RTK. |
| **Arguments** | Període en segons i format **float** . |
| **Retorn** | - |

| **PROTOTIPUS** | public native void iniciarAdquisicioDades() throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Esborra el contingut del Buffer del RTK i considera temps 0 a partir de l'instant en què s'invoca. |
| **Arguments** | - |
| **Retorn** | - |

| **PROTOTIPUS** | public native void paraCarro()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Fixa l'acció de control a 0. |
| **Arguments** | - |
| **Retorn** | - |

| **PROTOTIPUS** | public native int descarregaLlibreria()<br>throws ExcepcioFBK33200; |
|---|---|
| **Tasca realitzada** | Descarrega la llibreria del RTK en memòria i retorna el valor del comptador del mòdul. |
| **Arguments** | - |
| **Retorn** | Valor del comptador en format **int**. |

S'ha de tenir en compte que a l'hora de passar com a arguments o obtenir com a retorn vectors amb valors que a les funcions homologues de Matlab eren en format de matriu, la distribució dels elements és la següent.

Si la matriu seria:

$a11 \quad a21 \quad a31 \quad a41$

$a12 \quad a22 \quad a32 \quad a42$

Llavors el vector serà:

$a11 \quad a12 \quad a21 \quad a22 \quad a31 \quad a32 \quad a41 \quad a42$

on $an$ son els elements.

A més la classe disposa d'una sèrie de variables *int* finals (constants) de classe (estàtiques), que simbolitzen els diversos algoritmes de control i els diferents tipus de senyals de la font d'excitació interna en el cas d'excitació en llaç obert.

| Variable final | valor | Mètodes subjectes a utilització |
|---|---|---|
| CAP | 0 | get/setTipusControl |
| LLAC_OBERT | 1 | get/setTipusControl |
| LQ_INVERTIT | 2 | get/setTipusControl |
| LQ_GRUA | 3 | get/setTipusControl |
| FUZZY | 4 | get/setTipusControl |
| PID | 5 | get/setTipusControl |
| CONTROL_OPTIM | 6 | get/setTipusControl |
| CONTROL_ADAPTATIU | 7 | get/setTipusControl |
| EXTERN | 99 | get/setTipusControl |
| CONSTANT | 0 | get/setPW |
| SENYAL_QUADRADA | 1 | get/setPW |
| SENYAL_TRIANGULAR | 2 | get/setPW |
| SENYAL_SINUSOIDAL | 3 | get/setPW |
| SENYAL_PERIODIC_ALEATORI | 4 | get/setPW |

## 8.2. APLICACIÓ DE CONTROL LOCAL.

De cara a il·lustrar el funcionament de la classe, es procedeix a l'implementació d'una aplicació anomenada *AplicacioFBK33200local* , el codi font de la qual es troba a l'arxiu *AplicacioFBK33200local.java*. Aquesta aplicació assigna com a paràmetres de l'algoritme de control PID els valors obtinguts amb la sintonia realitzada i comprova el funcionament de l'aplicació davant de diverses consignes.

Els valors de l'algoritme de control son:

Kp =0.2997

Ki = 0
Kd = 0.0046
$|U_{min}|$ = 0.1

## 8.3. INTERFÍCIE *FBK33200remot.*

Aquesta interfície disposa dels mateixos mètodes públics que la classe *FBK33200local* , amb les següents excepcions:

**Mètodes no disponibles:**

| NOM MÈTODE | Motiu |
|---|---|
| getAdrPCL812PG | Aquest mètode retorna l'adreça base de la targeta de comunicació PCL812-PG a través de la qual es comunica el PC local amb la maqueta. No té sentit que l'usuari remot li pugui ser d'interès aquesta dada. |
| setAdrPCL812PG | Aquest mètode fixa l'adreça base de la targeta de comunicació PCL812-PG a través de la qual es comunica el PC local amb la maqueta. No té sentit que l'usuari remot pugui tenir capacitat de modificar aquest paràmetre. |
| comptadorsAzero | Aquest mètode fixa un punt de referència dels comptadors de posició (lineal i angular), la determinació del qual ha d'ésser responsabilitat exclusiva de l'usuari local. |

**Mètodes addicionals:**

| NOM MÈTODE | Motiu |
|---|---|
| AssignarReferenciaMatlab | Aquest mètode crea un no objecte *FBK33200local* al servidor, que és sobre el qual s'invocaran els mètodes de la interfície remota invocats pel client. En cas d'haver diversos objectes d'aquesta classe el RTK i l'entorn de Matlab referenciats per ell serien el mateix. Aquest mètode ha d'ésser el primer en ser invocat per part del client. |
| alliberarReferenciaMatlab | Allibera l'objecte creat amb el mètode anterior. |

La descripció d'aquests mètodes es detalla a continuació:

| **PROTOTIPUS** | public void assignarReferenciaMatlab ()<br>throws FBK33200remotPackage.ExcepcioFBK33200remota |
|---|---|
| **Arguments** | - |
| **Retorn** | - |

| **PROTOTIPUS** | public void alliberarReferenciaMatlab () |
|---|---|
| **Arguments** | - |
| **Retorn** | - |

Els mètodes d'aquesta interfície llencen excepcions del tipus *ExcepcioFBK33200remota* en lloc de les seves homologues per al cas local *ExcepcioFBK33200*. Aquesta classe d'excepcions es troben al paquet *FBK33200remotPackage*, i per tant això s'ha de tenir en compte a l'hora de gestionar-les.

**Substitució del constructor:**

En aquesta interfície remota, al igual que en totes les basades en CORBA, la tasca del constructor és realitzada pel mètode de classe *narrow* de la classe abstracta *FBK33200remotHelper*. Aquest mètode retorna la referència a l'objecte remot, resident al servidor, tot passant-li com a argument un objecte de la classe *org.omg.CORBA.Object*, la creació i inicialització del qual es realitza en funció de la ubicació del servidor i el nom de la interfície remota. El procediment detallat es pot apreciar en la implementació de l'aplicació client i servidor de la present interfície.

| **PROTOTIPUS** | public static FBK33200remot narrow<br>(org.omg.CORBA.Object obj) |
|---|---|
| **Arguments** | org.omg.CORBA.Object obj |
| **Retorn** | FBK33200remot |

## 8.4. APLICACIÓ DE CONTROL REMOT.

De cara a il·lustrar el funcionament de la interfície remota, es procedeix a l'implementació d'una aplicació client anomenada *ClientFBK33200.class* , el codi font de la qual es troba a l'arxiu *ClientFBK33200.java* i una aplicació servidor anomenada *ServidorFBK33200.class* , el codi font de la qual es troba a l'arxiu *ServidorFBK33200.java* .

L'aplicació servidor crea un objecte *FBK33200local* , realitza les inicialitzacions pertinents i posteriorment crea un objecte de la classe *FBK33200servent*, registre el mateix amb l'ORB al servidor de noms, obté l'arrel del context de noms, bateja la referència de l'objecte a la xarxa i espera invocacions per part de possibles clients.

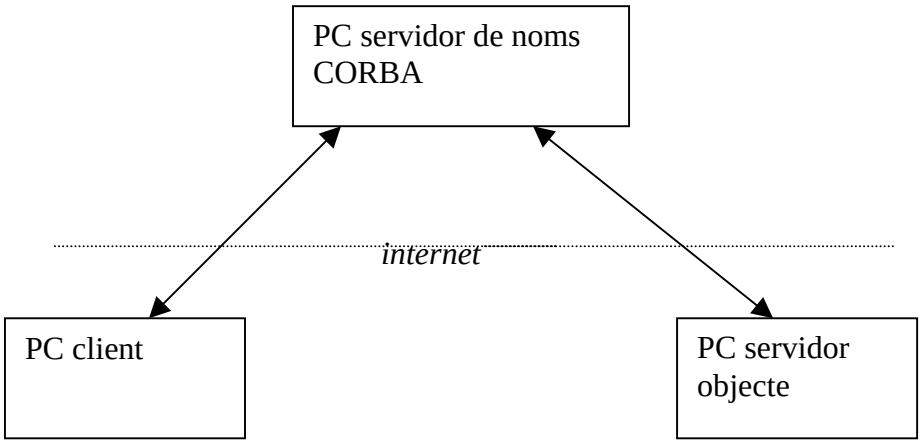
L'aplicació client crea una referència a un objecte que implementí la interfície *FBK33200remot* , crea un objecte de la classe *org.omg.CORBA.Object* a partir de la ubicació a la xarxa del servidor de noms, obté la referència de l'objecte ubicat al servidor amb el mètode *narrow* de la classe *FBK33200remotHelper* i posteriorment realitza tasques anàlogues a les realitzades per l'aplicació local amb l'objecte *FBK33200local*.

Per tal de executar les aplicacions hi ha un concepte, mencionat en aquests darrers paràgrafs que convé explicar: el Servidor de Noms.

En una aplicació client-servidor CORBA, hi intervenen tres subjectes.

- El servidor de l'objecte remot o servidor de l'aplicació.
- El client.
- El servidor de noms.

Els dos primers tenen una funció ja coneguda, mentre que el Servidor de Noms és una aplicació que gestiona el "trànsit" a la xarxa de diversos objectes amb interfícies remotes. Tant el servidor de l'aplicació com el client, quan es connecten a Internet han de cercar el Servidor de Noms, és a dir que s'ha de conèixer la seva ubicació a la xarxa (adreça IP).




Un servidor de noms pot ser únic per cada servidor d'aplicació o pot donar servei a diversos d'ells. En aquest cas el servidor de noms donarà servei, únicament al servidor de l'aplicació i es trobarà ubicat al mateix PC , mentre que el client es trobarà ubicat en un altre PC.

L'aplicació del servidor de noms CORBA rep el nom de **tnameserv.exe** , i es troba disponible tant al JDK 1.3.1 com al JRE 1.3 o superiors versions.

Per tal d'executar les aplicacions a la línia de comandes s'ha d'indicar el següent.

Per executar el servidor de noms:

```
tnameserv -ORBInitialPort <portSN>
```

on `portSN` és el port de l'aplicació del servidor de noms
(si no s'indica, s'assigna el port 900).

Per executar l'aplicació del servidor de l'objecte:

```
java ServidorFBK33200 -ORBInitialPort <portSN> -ORBInitialHost
<hostSN>
```

on `hostSN` és l'adreça IP d'on s'ubica el servidor de noms
(no s'indica si es troba al mateix ordinador)

Per executar l'aplicació del client:

```
java ClientFBK33200 -ORBInitialPort <portSN> -ORBInitialHost
<hostSN>
```

Com om que en aquest cas el Servidor de noms i el de l'objecte resideixen al mateix PC, que correspon al PC `barreja.upc.es` amb l'adreça IP `147.83.107.24`, i que el port 900 està lliure per utilitzar-lo, es procedeix a l'execució.

L'execució de les aplicacions comporta les següents dependències:

| **Per executar...** | **Ha d'estar en funcionament...** |
|---|---|
| <Servidor_Objecte> | <Servidor_de_Noms><br>→ `tnameserv.exe` |
| <Client> | <Servidor_Objecte> |

A partir d'això prosseguim. Iniciem el servidor de noms des d'una consola de comandes:

```
tnameserv
```

Des d'una altra consola de comandes executem el servidor de l'objecte:

```
java ServidorFBK33200
```

Posteriorment des del PC del client teclegem:

```
java ClientFBK33200 -ORBInitialHost 147.83.107.24
```

Hem de recordar tancar tant el servidor de l'objecte com el de noms quan ja no s'hagin d'utilitzar; en el nostre cas posteriorment a la realització de l'execució de l'aplicació client.

## 8.5. REQUISITS PER LA UTILITZACIÓ DE LES CLASSES JAVA.

De cara a la utilització de les classes, tant local com remotament hi ha una sèrie de requisits que s'han de tenir en compte pel seu correcte funcionament.

### 8.5.1. CLASSE *FBK33200local*.

- La llibreria dinàmica *FBK33200local.dll* ha de trobar-se en el directori de treball, o en un dels directoris de cerca d'executables del SO (**path**).
- En el PC ha d'estar instal·lat Matlab 5.3 o superior.
- El software del pont-grua ha d'estar correctament instal·lat al PC.
- El directori d'executables de Matlab ha d'estar inclòs a la ruta de cerca d'executables del SO (**path**). Generalment aquest directori és <directoriMareMatlab>/bin .
- En el PC ha d'estar instal·lat el JRE (*Java Runtime enviroment*) versió 3.1 o superior.
- Si es desitja cridar les classes tal i com s'ha descrit a la memòria, el directori d'executables del JRE o del JDK ha d'estar inclòs a la ruta de cerca d'executables del SO (**path**) i els fitxers comprimits de classes (*.jar o *.zip) han d'estar inclosos a la ruta de cerca de classes (**classpath**).

### 8.5.2. INTERFÍCIE *FBK33200remot.*

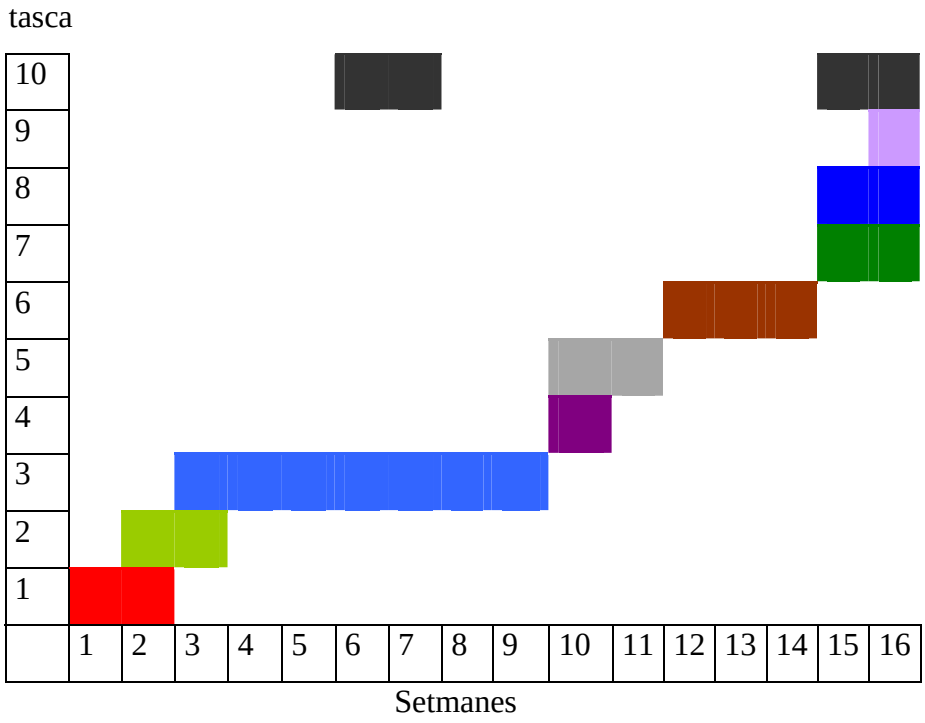
- *Les mateixes que el cas local*.
- S'ha de conèixer la IP de la màquina on s'ubica el servidor de noms i el port per on treballa.
- En cas d'utilitzar-se el mateix PC pels dos servidors, tenir en compte la disponibilitat del port per on treballarà el Servidor de Noms; en cas de ser possible és recomanable el port per defecte, 900.

# 9. PLÀ D'ACCIÓ.

En aquest capítol es detalla la distribució temporal de tasques entre les 16 setmanes que ha ocupat el desenvolupament del projecte. Les tasques realitzades es poden enumerar en les següents:

1. Recopilació d'informació de *JAVA IDL* i *CORBA*.
2. Coneixement de l'entorn de treball.
    - Llenguatge de programació Java.
    - Targeta PCL812-PG.
    - Llibreria PCL812-PG.
    - Maqueta del Pont-grua.
3. Interacció a nivell local.
4. Disseny d'una interfície en Java que permeti el control local.
5. Disseny d'una interfície en Java IDL basada en CORBA per poder utilitzar el controlador de la llibreria dissenyada a la tasca 4 en una màquina remota al procés a controlar.
6. Modelat del procés.
7. Disseny d'una aplicació en Java per il·lustrar el funcionament de la interfície de la tasca 4.
8. Disseny d'una aplicació en Java per il·lustrar el funcionament de la interfície de la tasca 5.
9. Avaluació de resultats i conclusions.
10. Redacció de la Memòria. En aquest cas la tasca s'ha desenvolupat, bàsicament en dos períodes de temps separats.

La distribució temporal en setmanes de las tasques realitzades s'il·lustra en el següent Diagrama de Gantt:




La càrrega global, quantificada en hores de treball, ha estat d'unes 475 hores.

| SETMANA | data inici | data final |
|---|---|---|
| 1 | 16-02-04 | 22-02-04 |
| 2 | 23-02-04 | 29-02-04 |
| 3 | 01-03-04 | 07-03-04 |
| 4 | 08-03-04 | 14-03-04 |
| 5 | 15-03-04 | 21-03-04 |
| 6 | 22-03-04 | 28-03-04 |
| 7 | 29-03-04 | 04-04-04 |
| 8 | 05-04-04 | 11-04-04 |
| 9 | 12-04-04 | 18-04-04 |
| 10 | 19-04-04 | 25-04-04 |
| 11 | 26-04-04 | 02-05-04 |
| 12 | 03-05-04 | 09-05-04 |
| 13 | 10-05-04 | 16-05-04 |
| 14 | 17-05-04 | 23-05-04 |
| 15 | 24-05-04 | 30-05-04 |
| 16 | 31-05-04 | 06-06-04 |

# 10. ESTIMACIÓ DE COST ECONÒMIC.

A continuació es detalla, a títol orientatiu, la valoració econòmica de la tasca realitzada.

| CAPITOL | DENOMINACIÓ | Quantitat | PREU UNITARI | PREU PARCIAL | PREU CAPITOL |
|---|---|---|---|---|---|
| 1 | ***Material informàtic adicional per subministar amb el producte final*** | | | | **0,00 €** |
| 1.1 | Java Runtime Enviroment (JRE) | 1 | 0,00 € | 0,00 € | |
| 2 | ***Eines de desenvolupament*** | | | | **0,00 €** |
| 2.1 | Eina de programació en Java (JDK versió 1.3.1) | 1 | 0,00 € | 0,00 € | |
| 2.2 | Entorn de programació Visual C++ ***subministrat per l'escola*** | 1 | - | - | |
| 2.3 | Matlab 5.3 ***subministrat per l'escola*** | 1 | - | - | |
| 2.4 | Software de la maqueta 33-005 de feedback ***subministrat per l'escola*** | 1 | - | - | |
| 3 | ***Disseny*** | | | | **7.500,00 €** |
| 3.1 | Hores de familiarització amb l'entorn de treball y les eines necessaries pel desenvolupament del projecte | 50 | 25,00 € | 1.250,00 € | |
| 3.2 | Hores realització del disseny dels algoritmes i implementació dels mateixos | 250 | 25,00 € | 6.250,00 € | |
| 4 | ***Documentació*** | | | | **1.500,00 €** |
| 4.1 | Hores redacció i revisió de la documentació adjunta al projecte | 75 | 20,00 € | 1.500,00 € | |
| 5 | ***Proves*** | | | | **2.500,00 €** |
| 5.1 | Hores de realització proves del correcte funcionament del sistema | 100 | 25,00 € | 2.500,00 € | |
| **TOTAL** | | | | | **11.500,00 €** |
| **TOTAL (amb I.V.A.)** | | | | | **13.340,00 €** |

# 11. CONCLUSIONS.

## 11.1. VALORACIÓ DELS RESULTATS OBTINGUTS.

La valoració dels resultats obtinguts és positiva ja que s'han assolit el conjunt dels objectius proposats.

L'objectiu de major complexitat i que ha ocupat major càrrega temporal ha sigut el primer d'ells, la interacció des de l'entorn Java amb la maqueta seleccionada, la maqueta del pont-grua. En relació amb aquest aspecte és interessant destacar que aquest alt grau de complexitat en la interacció des d'un entorn diferent al que està inicialment previst pel fabricant, és degut a l'absència d'informació subministrada per part d'aquest sobre la interacció a baix nivell entre el PC i la maqueta. En el suposat cas que el fabricant hagués subministrat juntament amb la documentació de la maqueta els protocols de comunicació, a nivell de capa d'enllaç (segons el model OSI) entre el PC i l'acondicionador de senyals d'entrada i sortida de la maqueta, s'hauria pogut desenvolupar el projecte aprofitant la classe dissenyada en el projecte antecedent d'aquest. En aquest cas la interacció hauria sigut a un nivell més baix del que ha resultat ser finalment i l'alliberament de càrrega temporal hauria permès un major desenvolupament de la part relacionada amb el control del procés pròpiament dit.

Malgrat tot la valoració de la tasca realitzada és positiva, ja que s'ha desenvolupat una interacció amb un procés d'una determinada forma no prevista per part del fabricant. Això obre portes i crea fonaments per a futurs treballs que vagin encaminats a realitzar tasques anàlogues amb altres maquetes on la interacció estigui prevista d'una forma tancada. Una possible exemple d'això seria la maqueta "germana" de la del pont-grua, la maqueta de l'helicòpter, ubicada al mateix laboratori ,del mateix fabricant i amb una interfície bastant anàloga a la de la maqueta treballada.

Amb el desenvolupament del projecte s'han adquirit coneixements sobre llibreries d'enllaç dinàmic, interacció amb l'entorn de Matlab des de codi C i sobretot el funcionament i les capacitats de la tecnologia IDL en el camp de les Comunicacions Industrials.

Altres dades a destacar també son certs errors en la documentació proporcionada pel fabricant, que han sigut detectats durant el desenvolupament del projecte, i que per tant poden ser tinguts en compte per tal d'evitar que puguin provocar en un futur errors en processos d'experimentació amb la maqueta.

## 11.2. PROPOSTES D'AMPLIACIÓ.

De cara a possibles ampliacions del present projecte es podria proposar la continuació en el camp del control del procés. Seria interessant plantejar l'objectiu del control del pèndul en caiguda lliure, i posteriorment invertit, a nivell local i la fixació de la consigna de posició del carro a nivell remot, de forma que s'apreciés la utilitat de combinar els dos.

# 12. REFERÈNCIES.

## 12.1. BIBLIOGRAFIA.

[1]. *Títol:* MANUAL DE LA MAQUETA ”Digital Pendulum Control System”
*Autor:* Feedback Instruments LTD.

[2]. *Títol:* Aprenda Java como si estuviera en primero.
*Autor: (diversos)*
*Editat per:* Escuela Superior de Ingenieros Industriales (Universidad de Navarra)

[3]. *Títol:* Ingeniería de Control Moderna.
*Autor:* Katsuhiko Ogata
*Editorial:* Prentice Hall

[4]. *Títol:* Sistemas Controlados por Computador.
*Autor:* Karl J.Aström , Bjorn Wittenmark
*Editorial:* Paraninfo

[5]. *Títol:* The Java Native Interface
*Autor:* Sheng Liang
*Editorial:* Addison-Wesley

## 12.2. FONTS DE RECURSOS D’INTERNET.

[6]. http://java.sun.com
Web dels inventors de Java

[7]. http://www1.ceit.es/asignaturas/Informat1/AyudaInf/
Secció informàtica web Universitat Navarra

[8]. http://www.xtec.es/formacio/curstele/d110/

Curs de Programació en Java.

[9]. http://www.programacion.com/java/tutorial/idl/

Tutorial de IDL i CORBA

[10]. http://www.zator.com/Cpp/E1_4_4b.htm

Tutorial de les llibreries dinàmiques (*.dll)

[11]. http://www.mathworks.com/

Web dels fabricants de Matlab

## 12.3. TREBALLS ANTERIORS.

[12]. *Títol:* Desenvolupament d'aplicacions de control remot en Java

*Tipologia:* Projecte Final de Carrera.

*Autor:* Julián Fernández Barba

*Defensat:* gener de 2004

*UNIVERSITAT POLITÈCNICA DE CATALUNYA*

# CONTROL LOCAL I REMOT DE PROCESSOS MITJANÇANT JAVA SOBRE UNA PLATAFORMA WINDOWS

*PROJETE FINAL DE CARRERA*


Document: **APÈNDIXS**

Alumne: Daniel Castillo Hand.

Director: Josep Cugueró Escofet.

Data: Juny de 2004

Titulació: Enginyeria Tècnica Industrial.
Especialitat: Electrònica Industrial.

Escola: Escola Universitària d'Enginyeria Tècnica Industrial de Terrassa

Departament: Enginyeria de Sistemes, Automàtica i Informàtica Industrial

## 13.1. ARXIU *AplicacioControlCarro.java*

```
import java.awt.*;
import java.awt.event.*;

class AplicacioControlCarro
{
   public static void main(String args[])
   {
         try{
               FinestraControlCarro instFcc=
                                          new FinestraControlCarro();
         }catch(Exception ex){
               System.out.println(ex.getMessage());
         }
   }
}

class FinestraControlCarro extends Frame implements WindowListener
{
   private PCL812PG targeta;
   private Label estat;

   public FinestraControlCarro() throws PCL812PGException
   {
         targeta=new PCL812PG();
         targeta.obrir(0);
         targeta.escripturaCanalAnalogic((short)0,(float)2.5);

         super.setTitle("Control del carro");

         estat=new Label("PARAT");
         BotoDreta botoDreta=new BotoDreta(estat,targeta);
         BotoEsquerra botoEsquerra=new BotoEsquerra(estat,targeta);

         super.add(botoDreta);
         super.add(botoEsquerra);

         super.setLayout(new GridLayout(3,3,200,20));
         super.pack();
         super.setVisible(true);

         this.addWindowListener(this);
   }

   public void windowClosing(WindowEvent we)
   {
         try{
               targeta.escripturaCanalAnalogic((short)0,(float)2.5);
         }catch(PCL812PGException ex){}

         System.exit(0);
   }
   public void windowOpened(WindowEvent we){}
   public void windowClosed(WindowEvent we){}
   public void windowIconified(WindowEvent we){}
   public void windowDeiconified(WindowEvent we){}
   public void windowActivated(WindowEvent we){}
   public void windowDeactivated(WindowEvent we){}

}
```

```
class Boto extends Button implements MouseListener
{
   protected Label etiqueta;
   protected PCL812PG targeta;

   public Boto(String nom,Label etiqueta,PCL812PG targeta)
   {
         super(nom);
         this.etiqueta=etiqueta;
         this.targeta=targeta;
         super.addMouseListener(this);
   }
   public void mouseClicked(MouseEvent me){}
   public void mouseEntered(MouseEvent me){}
   public void mouseExited(MouseEvent me) {}
   public void mousePressed(MouseEvent me){}

   public void mouseReleased(MouseEvent me)
   {
         try{
               targeta.escripturaCanalAnalogic((short)0,(float)2.5);
         }catch(PCL812PGException ex){}
         System.exit(0);
   }
}

class BotoDreta extends Boto
{
   public BotoDreta(Label etiqueta, PCL812PG targeta)
   {
         super("DRETA",etiqueta,targeta);
   }

   public void mousePressed(MouseEvent me)
   {
         try{
               super.targeta.escripturaCanalAnalogic
                                          ((short)0,(float)2.8);
         }catch(PCL812PGException ex){}

         super.etiqueta.setText("DRETA");
   }
}

class BotoEsquerra extends Boto
{
   public BotoEsquerra(Label etiqueta, PCL812PG targeta)
   {
         super("ESQUERRA",etiqueta,targeta);
   }

   public void mousePressed(MouseEvent me)
   {
         try{

   super.targeta.escripturaCanalAnalogic((short)0,(float)2.2);
         }catch(PCL812PGException ex){}

         super.etiqueta.setText("ESQUERRA");
   }
}
```

## 13.2. ARXIU *EntradaDigital.java*

```
import java.io.*;

class EntradaDigital
{

public static void main (String arg[]) throws java.io.IOException
{
   try{

         PCL812PG targeta= new PCL812PG();
         targeta.obrir(0);
         targeta.escripturaCanalAnalogic((short)0,(float)2.5);

         long tempsActual;

         int i=0;
         float []temps = new float [51];
         float []r = new float [51];
         short []y = new short [51];

         for(i=0; i<51; i++)
         {
               temps[i]=(float)(0.1*i);
               if(i<10 || i>30) r[i]=(float)2.5;
               else r[i]=(float)3;
         }

         for(i=0; i<51; i++)
         {
               tempsActual = System.currentTimeMillis();
               targeta.escripturaCanalAnalogic((short)0,r[i]);

               y[i] = targeta.lecturaWordPort((short)((0x220) + 6));

               try{
                     Thread.sleep
            ((long)(100-(System.currentTimeMillis()-tempsActual)));

               }catch(InterruptedException ie){}
         }
         targeta.tancar();

         String rutaFitxer="c:\\matlab\\work\\word.dat";

         FileWriter fw = new FileWriter (rutaFitxer);
         BufferedWriter bw = new BufferedWriter(fw);
         PrintWriter valors = new PrintWriter(bw);
         for(i=0;i<51;i++)
         {
               bw = new BufferedWriter
                              (new FileWriter (rutaFitxer, true));
               valors = new PrintWriter(bw);
               valors.println(temps[i]+" "+r[i]+" "+y[i]);
               valors.close();
         }

   }catch(PCL812PGException e){}
}
}
```

## 13.3. ARXIU *EntradaAnalogica.java*

```
import java.io.*;

class EntradaAnalogica
{

public static void main (String arg[]) throws java.io.IOException
{

   try{

         if (arg.length !=1)
         {
         System.err.println("Indica el numero d'entrada 0 al 15");
         System.exit(1);
         }

         short canal;
         canal = Short.valueOf(arg[0]).shortValue();

         if (canal<0 || canal>15)
         {
         System.err.println("Indica el numero d'entrada 0 al 15");
         System.exit(1);
         }

         PCL812PG targeta= new PCL812PG();
         targeta.obrir(0);
         targeta.escripturaCanalAnalogic((short)0,(float)2.5);

         long tempsActual;

         int i=0;
         float []temps = new float [51];
         float []r = new float [51];
         float []y = new float [51];

         for(i=0; i<51; i++)
         {
               temps[i]=(float)(0.1*i);
               if(i<10 || i>30) r[i]=(float)2.5;
               else r[i]=(float)3;
         }

         for(i=0; i<51; i++)
         {
               tempsActual = System.currentTimeMillis();
               targeta.escripturaCanalAnalogic((short)0,r[i]);

               y[i] = targeta.lecturaCanalAnalogic(canal,(short)0);

               try{
               Thread.sleep
            ((long)(100-(System.currentTimeMillis()-tempsActual)));
               }catch(InterruptedException ie){}
         }
         targeta.tancar();
```

```
            String rutaFitxer="c:\\matlab\\work\\prova.dat";
            FileWriter fw = new FileWriter (rutaFitxer);
            BufferedWriter bw = new BufferedWriter(fw);
            PrintWriter valors = new PrintWriter(bw);

            for(i=0;i<51;i++)
            {
            bw = new BufferedWriter(new FileWriter (rutaFitxer, true));
                  valors = new PrintWriter(bw);
                  valors.println(temps[i]+" "+r[i]+" "+y[i]);
                  valors.close();
            }

      }catch(PCL812PGException te){}
}
}
```

## 13.4. ARXIU *Matlab_C_libs.bat*

```
path=C:\archiv~1\micros~3\VC98\Bin;%path%

lib /def:"C:\MATLABR11\extern\include\libmatlb.def"
/machine:ix86 /OUT:libmatlb.lib

lib /def:"C:\MATLABR11\extern\include\libmmfile.def"
/machine:ix86 /OUT:libmmfile.lib

lib /def:"C:\MATLABR11\extern\include\libmx.def"
/machine:ix86 /OUT:libmx.lib

lib /def:"C:\MATLABR11\extern\include\matlab.def"
/machine:ix86 /OUT:matlab.lib

lib /def:"C:\MATLABR11\extern\include\feng.def"
/machine:ix86 /OUT:feng.lib

lib /def:"C:\MATLABR11\extern\include\libeng.def"
/machine:ix86 /OUT:libeng.lib


rem  ************************************************************
rem  Comentari:
rem  El path de la primera linia es el que correspon a on s'ubiquen
rem  els executables del compilador de c++
rem  Cada dues linies son una linia de comandes
rem  S'ha d'executar al directori on hi hagin els arxius *.def
rem  i el directori dels executables de Matlab ha d'estar inclos
rem  al path del SO
rem  ************************************************************
```

## 13.5. ARXIU *ProvaMex.cpp*

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <mex.h>
#include <conio.h>

typedef void (*FUNCIO_MEX)(int,mxArray*[],int,mxArray*[]);

int main(int argc, char *argv[])
{
  HINSTANCE instLlib;
  FUNCIO_MEX funcioMex;

  mxArray **instMxArray1;
  mxArray **instMxArray2;

  instLlib=LoadLibrary("Pd_call.dll");
  if(instLlib!=NULL)
  {
    funcioMex=(FUNCIO_MEX)GetProcAddress(instLlib,"mexFunction");
    if(funcioMex!=NULL)
    {
        instMxArray1[0] = mxCreateString("StopPractical");

        (*funcioMex)(1,instMxArray2,1,instMxArray1);

        mxDestroyArray(instMxArray1[0]);
        mxDestroyArray(instMxArray2[0]);

    }
    else printf("No s'ha trobat la funcio !!\n");

    FreeLibrary(instLlib);
  }
  else printf("No s'ha pogut carregar la llibreria\n");

  getch();
  return 0;
}
```

## 13.6. ARXIU *Prova_call.m*

```
function y=Prova_call(a,b,c)

if(strcmp(a,'suma'))
   y=b(1)+b(2);
elseif(strcmp(a,'sumaProducte'))
   y=(b(1)+b(2))*c(1);
elseif(strcmp(a,'salut'))
   disp('Hola a tothom!!!');
   y='Hola a tothom!!!';
elseif(strcmp(a,'diset'))
   y=17;
end
```

## 13.7. ARXIU *Prova_call.h*

```
/*
 * MATLAB Compiler: 2.0.1
 * Date: Thu Apr 15 12:07:45 2004
 * Arguments: "-x" "Prova_call"
 */
#ifndef MLF_V2
#define MLF_V2 1
#endif

#ifndef __Prova_call_h
#define __Prova_call_h 1

#include "matlab.h"

extern mxArray * mlfProva_call(mxArray * a, mxArray * b, mxArray * c);
extern void mlxProva_call(int nlhs,
                          mxArray * plhs[],
                          int nrhs,
                          mxArray * prhs[]);

#endif
```

## 13.8. ARXIU *Prova_call.c*

```
/*
 * MATLAB Compiler: 2.0.1
 * Date: Thu Apr 15 12:07:45 2004
 * Arguments: "-x" "Prova_call"
 */
#include "Prova_call.h"

/* The function "MProva_call" is the implementation version of the "Prova_call"
 * M-function from file "C:\MATLABR11\work\Prova_call.m" (lines 1-13). It
 * contains the actual compiled code for that M-function. It is a static
 * function and must only be called from one of the interface functions,
 * appearing below.
 *
 * function y=Prova_call(a,b,c)
 */
static mxArray * MProva_call(int nargout_,
                             mxArray * a,
                             mxArray * b,
                             mxArray * c) {
    mxArray * y = mclGetUninitializedArray();
    mclValidateInputs("Prova_call", 3, &a, &b, &c);

    /* if(strcmp(a,'suma'))
     */
    if (mlfTobool(mlfStrcmp(a, mxCreateString("suma")))) {
        /* y=b(1)+b(2);
         */
        mlfAssign(
          &y,
          mlfPlus(
            mlfIndexRef(b, "(?)", mlfScalar(1.0)),
            mlfIndexRef(b, "(?)", mlfScalar(2.0))));
    /*
```

```
     * elseif(strcmp(a,'sumaProducte'))
     */
    } else if (mlfTobool(mlfStrcmp(a,
                                   mxCreateString("sumaProducte")))) {
        /* y=(b(1)+b(2))*c(1);
         */
        mlfAssign(
          &y,
          mlfMtimes(
            mlfPlus(
              mlfIndexRef(b, "(?)", mlfScalar(1.0)),
              mlfIndexRef(b, "(?)", mlfScalar(2.0))),
            mlfIndexRef(c, "(?)", mlfScalar(1.0))));
    /* elseif(strcmp(a,'salut'))
     */
    } else if (mlfTobool(mlfStrcmp(a, mxCreateString("salut")))) {
        /* disp('Hola a tothom!!!');
         */
        mlfDisp(mxCreateString("Hola a tothom!!!"));
        /* y='Hola a tothom!!!';
         */
        mlfAssign(&y, mxCreateString("Hola a tothom!!!"));
    /* elseif(strcmp(a,'diset'))
     */
    } else if (mlfTobool(mlfStrcmp(a, mxCreateString("diset")))) {
         /* y=17;
          */
        mlfAssign(&y, mlfScalar(17.0));
    /* end
     */
    }
    mclValidateOutputs("Prova_call", 1, nargout_, &y);
    return y;
}


/*
 * The function "mlfProva_call" contains the normal interface for the
 *  "Prova_call"  M-function  from  file  "C:\MATLABR11\work\Prova_call.m"
(lines
 * 1-13). This function processes any input arguments and passes them to
the
 * implementation version of the function, appearing above.
 */
mxArray * mlfProva_call(mxArray * a, mxArray * b, mxArray * c) {
    int nargout = 1;
    mxArray * y = mclGetUninitializedArray();
    mlfEnterNewContext(0, 3, a, b, c);
    y = MProva_call(nargout, a, b, c);
    mlfRestorePreviousContext(0, 3, a, b, c);
    return mlfReturnValue(y);
}
```

```
/*
 * The function "mlxProva_call" contains the feval interface for the
 * "Prova_call" M-function from file "C:\MATLABR11\work\Prova_call.m"
(lines
 * 1-13). The feval function calls the implementation version of
Prova_call
 * through this function. This function processes any input arguments
and
 * passes them to the implementation version of the function, appearing
above.
 */
void mlxProva_call(int nlhs, mxArray * plhs[], int nrhs, mxArray *
prhs[]) {
    mxArray * mprhs[3];
    mxArray * mplhs[1];
    int i;
    if (nlhs > 1) {
        mlfError(
          mxCreateString(
            "Run-time Error: File: Prova_call Line: 1 Column:"
            " 0 The function \"Prova_call\" was called with m"
            "ore than the declared number of outputs (1)"));
    }
    if (nrhs > 3) {
        mlfError(
          mxCreateString(
            "Run-time Error: File: Prova_call Line: 1 Column"
            ": 0 The function \"Prova_call\" was called with"
            " more than the declared number of inputs (3)"));
    }
    for (i = 0; i < 1; ++i) {
        mplhs[i] = NULL;
    }
    for (i = 0; i < 3 && i < nrhs; ++i) {
        mprhs[i] = prhs[i];
    }
    for (; i < 3; ++i) {
        mprhs[i] = NULL;
    }
    mlfEnterNewContext(0, 3, mprhs[0], mprhs[1], mprhs[2]);
    mplhs[0] = MProva_call(nlhs, mprhs[0], mprhs[1], mprhs[2]);
    mlfRestorePreviousContext(0, 3, mprhs[0], mprhs[1], mprhs[2]);
    plhs[0] = mplhs[0];
}
```

## 13.9. ARXIU *Prova_call_mex.c*

```
/*
 * MATLAB Compiler: 2.0.1
 * Date: Thu Apr 15 12:07:45 2004
 * Arguments: "-x" "Prova_call"
 */

#ifndef MLF_V2
#define MLF_V2 1
#endif

#include "matlab.h"
#include "Prova_call.h"

static mlfFunctionTableEntry function_table[1]
  = { { "Prova_call", mlxProva_call, 3, 1 } };

/*
 * The function "mexFunction" is a Compiler-generated mex wrapper,
suitable for
 * building a MEX-function. It initializes any persistent variables as
well as
 * a function table for use by the feval function. It then calls the
function
 * "mlxProva_call". Finally, it clears the feval table and exits.
 */
void mexFunction(int nlhs, mxArray * * plhs, int nrhs, mxArray * * prhs)
{
    mlfTry {
        mlfFunctionTableSetup(1, function_table);
        mclImportGlobal(0, NULL);
        mlxProva_call(nlhs, plhs, nrhs, prhs);
        mlfFunctionTableTakedown(1, function_table);
    } mlfCatch {
        mlfFunctionTableTakedown(1, function_table);
        mclMexError();
    } mlfEndCatch
}
```

## 13.10. ARXIU *LlacObert.m*

```
pd_call('SetAlgNo',0);

disp('situa el pendol al centre i prem return');
pause;
pd_call('ResetEncoder');

pd_call('SetSampleTime',0.01);

pd_call('SetPW',[0 0.25]);
pd_call('ResetTime');

while(pd_call('GetNoOfSamples')<10);
end;

pd_call('SetAlgNo',1);

while(pd_call('GetNoOfSamples')<85);
end;

pd_call('SetAlgNo',0);

while(pd_call('GetNoOfSamples')<200);
end;

resultat=pd_call('GetHistory');

plot(resultat(1,:),resultat(6,:),'r-',resultat(1,:),resultat(4,:),'b-');
```

## 13.11. ARXIU *ProvaJNI.idl*

```
interface ProvaJNIremot
{
   exception ExcepcioJNI{};

   long suma(in long a, in long b);
   long resta(in long a, in long b);
   long producte(in long a, in long b);
   long quocient(in long a, in long b) raises (ExcepcioJNI);

   long getResultat();
};
```

## 13.12. ARXIU *ProvaJNIremot.java*

```
/**
* ProvaJNIremot.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

public interface ProvaJNIremot extends ProvaJNIremotOperations,
org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity
{
} // interface ProvaJNIremot
```

## 13.13. ARXIU *ProvaJNIremotOperations.java*

```
/**
* ProvaJNIremotOperations.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

public interface ProvaJNIremotOperations
{
  int suma (int a, int b);
  int resta (int a, int b);
  int producte (int a, int b);
  int quocient (int a, int b) throws ProvaJNIremotPackage.ExcepcioJNI;
  int getResultat ();
} // interface ProvaJNIremotOperations
```

## 13.14. ARXIU *_ProvaJNIremotImplBase.java*

```
/**
* _ProvaJNIremotImplBase.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

public abstract class _ProvaJNIremotImplBase extends
org.omg.CORBA.portable.ObjectImpl implements ProvaJNIremot,
org.omg.CORBA.portable.InvokeHandler
{
  // Constructors
  public _ProvaJNIremotImplBase (){}

  private static java.util.Hashtable _methods = new java.util.Hashtable ();
  static
  {
    _methods.put ("suma", new java.lang.Integer (0));
    _methods.put ("resta", new java.lang.Integer (1));
    _methods.put ("producte", new java.lang.Integer (2));
    _methods.put ("quocient", new java.lang.Integer (3));
    _methods.put ("getResultat", new java.lang.Integer (4));
  }

  public org.omg.CORBA.portable.OutputStream _invoke
      (String method, org.omg.CORBA.portable.InputStream in,
               org.omg.CORBA.portable.ResponseHandler rh)
  {
    org.omg.CORBA.portable.OutputStream out = null;
    java.lang.Integer __method =
                          (java.lang.Integer)_methods.get(method);
    if (__method == null)
      throw new org.omg.CORBA.BAD_OPERATION
               (0, org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);

    switch (__method.intValue ())
    {
       case 0:  // ProvaJNIremot/suma
       {
         int a = in.read_long ();
```

```
          int b = in.read_long ();
          int __result = (int)0;
          __result = this.suma (a, b);
          out = rh.createReply();
          out.write_long (__result);
          break;
        }

        case 1:  // ProvaJNIremot/resta
        {
          int a = in.read_long ();
          int b = in.read_long ();
          int __result = (int)0;
          __result = this.resta (a, b);
          out = rh.createReply();
          out.write_long (__result);
          break;
        }

        case 2:  // ProvaJNIremot/producte
        {
          int a = in.read_long ();
          int b = in.read_long ();
          int __result = (int)0;
          __result = this.producte (a, b);
          out = rh.createReply();
          out.write_long (__result);
          break;
        }

        case 3:  // ProvaJNIremot/quocient
        {
          try {
            int a = in.read_long ();
            int b = in.read_long ();
            int __result = (int)0;
            __result = this.quocient (a, b);
            out = rh.createReply();
            out.write_long (__result);
          }catch (ProvaJNIremotPackage.ExcepcioJNI __ex) {
            out = rh.createExceptionReply ();
            ProvaJNIremotPackage.ExcepcioJNIHelper.write (out, __ex);
          }
          break;
        }

        case 4:  // ProvaJNIremot/getResultat
        {
          int __result = (int)0;
          __result = this.getResultat ();
          out = rh.createReply();
          out.write_long (__result);
          break;
        }
       default:
          throw new org.omg.CORBA.BAD_OPERATION
                (0, org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);
    }

    return out;
  }
```

```
  // _invoke
  // Type-specific CORBA::Object operations
  private static String[] __ids = {
    "IDL:ProvaJNIremot:1.0"};

  public String[] _ids ()
  {
    return __ids;
  }

} // class _ProvaJNIremotImplBase
```

## 13.15. ARXIU *_ProvaJNIremotStub.java*

```
/**
* _ProvaJNIremotStub.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

public class _ProvaJNIremotStub extends
org.omg.CORBA.portable.ObjectImpl implements ProvaJNIremot
{
  // Constructors
  // NOTE:  If the default constructor is used, the
  //        object is useless until _set_delegate (...)
  //        is called.
  public _ProvaJNIremotStub ()
  {
    super ();
  }
  public _ProvaJNIremotStub
                    (org.omg.CORBA.portable.Delegate delegate)
  {
    super ();
    _set_delegate (delegate);
  }
  public int suma (int a, int b)
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out =
                                          _request ("suma", true);
       _out.write_long (a);
       _out.write_long (b);
       _in = _invoke (_out);
       int __result = _in.read_long ();
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return suma (a, b);
    } finally {
        _releaseReply (_in);
    }
  } // suma
```

```
public int resta (int a, int b)
{
  org.omg.CORBA.portable.InputStream _in = null;
  try {
     org.omg.CORBA.portable.OutputStream _out =
                                        _request ("resta", true);
     _out.write_long (a);
     _out.write_long (b);
     _in = _invoke (_out);
     int __result = _in.read_long ();
     return __result;
  } catch (org.omg.CORBA.portable.ApplicationException _ex) {
     _in = _ex.getInputStream ();
     String _id = _ex.getId ();
     throw new org.omg.CORBA.MARSHAL (_id);
  } catch (org.omg.CORBA.portable.RemarshalException _rm) {
     return resta (a, b);
  } finally {
      _releaseReply (_in);
  }
} // resta

public int producte (int a, int b)
{
  org.omg.CORBA.portable.InputStream _in = null;
  try {
     org.omg.CORBA.portable.OutputStream _out =
                                        _request ("producte", true);
     _out.write_long (a);
     _out.write_long (b);
     _in = _invoke (_out);
     int __result = _in.read_long ();
     return __result;
  } catch (org.omg.CORBA.portable.ApplicationException _ex) {
     _in = _ex.getInputStream ();
     String _id = _ex.getId ();
     throw new org.omg.CORBA.MARSHAL (_id);
  } catch (org.omg.CORBA.portable.RemarshalException _rm) {
     return producte (a, b);
  } finally {
      _releaseReply (_in);
  }
} // producte

public int quocient (int a, int b)throws ProvaJNIremotPackage.ExcepcioJNI
{
  org.omg.CORBA.portable.InputStream _in = null;
  try {
     org.omg.CORBA.portable.OutputStream _out =
                                _request ("quocient", true);
     _out.write_long (a);
     _out.write_long (b);
     _in = _invoke (_out);
     int __result = _in.read_long ();
     return __result;
  } catch (org.omg.CORBA.portable.ApplicationException _ex) {
     _in = _ex.getInputStream ();
     String _id = _ex.getId ();
     if (_id.equals ("IDL:ProvaJNIremot/ExcepcioJNI:1.0"))
        throw ProvaJNIremotPackage.ExcepcioJNIHelper.read (_in);
```

```
        else    throw new org.omg.CORBA.MARSHAL (_id);
      } catch (org.omg.CORBA.portable.RemarshalException _rm) {
         return quocient (a, b);
      } finally {
          _releaseReply (_in);
      }
    } // quocient

    public int getResultat ()
    {
      org.omg.CORBA.portable.InputStream _in = null;
      try {
         org.omg.CORBA.portable.OutputStream _out =
                                       _request ("getResultat", true);
         _in = _invoke (_out);
         int __result = _in.read_long ();
         return __result;
      } catch (org.omg.CORBA.portable.ApplicationException _ex) {
         _in = _ex.getInputStream ();
         String _id = _ex.getId ();
         throw new org.omg.CORBA.MARSHAL (_id);
      } catch (org.omg.CORBA.portable.RemarshalException _rm) {
         return getResultat ();
      } finally {
          _releaseReply (_in);
      }
    } // getResultat

    // Type-specific CORBA::Object operations
    private static String[] __ids = {"IDL:ProvaJNIremot:1.0"};

    public String[] _ids ()
    {
      return (String[])__ids.clone ();
    }

    private void readObject (java.io.ObjectInputStream s)
    {
       try
       {
         String str = s.readUTF ();
         org.omg.CORBA.Object obj =
                    org.omg.CORBA.ORB.init ().string_to_object (str);
         org.omg.CORBA.portable.Delegate delegate =
                ((org.omg.CORBA.portable.ObjectImpl) obj)._get_delegate ();
         _set_delegate (delegate);
       } catch (java.io.IOException e) {}
    }

    private void writeObject (java.io.ObjectOutputStream s)
    {
       try
       {
         String str = org.omg.CORBA.ORB.init ().object_to_string (this);
         s.writeUTF (str);
       } catch (java.io.IOException e) {}
    }
  } // class _ProvaJNIremotStub
```

## 13.16. ARXIU *ProvaJNIremotHelper.java*

```
/**
* ProvaJNIremotHelper.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

abstract public class ProvaJNIremotHelper
{
  private static String  _id = "IDL:ProvaJNIremot:1.0";

  public static void insert (org.omg.CORBA.Any a, ProvaJNIremot that)
  {
    org.omg.CORBA.portable.OutputStream out =
                                          a.create_output_stream ();
    a.type (type ());
    write (out, that);
    a.read_value (out.create_input_stream (), type ());
  }

  public static ProvaJNIremot extract (org.omg.CORBA.Any a)
  {
    return read (a.create_input_stream ());
  }

  private static org.omg.CORBA.TypeCode __typeCode = null;
  synchronized public static org.omg.CORBA.TypeCode type ()
  {
    if (__typeCode == null)
    {
      __typeCode = org.omg.CORBA.ORB.init ().create_interface_tc
                        (ProvaJNIremotHelper.id (), "ProvaJNIremot");
    }
    return __typeCode;
  }

  public static String id ()
  {
    return _id;
  }

  public static ProvaJNIremot read
                       (org.omg.CORBA.portable.InputStream istream)
  {
    return narrow (istream.read_Object (_ProvaJNIremotStub.class));
  }

  public static void write
     (org.omg.CORBA.portable.OutputStream ostream, ProvaJNIremot value)
  {
    ostream.write_Object ((org.omg.CORBA.Object) value);
  }

  public static ProvaJNIremot narrow (org.omg.CORBA.Object obj)
  {
    if (obj == null) return null;
    else if (obj instanceof ProvaJNIremot) return (ProvaJNIremot)obj;
    else if (!obj._is_a (id ()))throw new org.omg.CORBA.BAD_PARAM ();
```

```
    else
    {
      org.omg.CORBA.portable.Delegate delegate =
              ((org.omg.CORBA.portable.ObjectImpl)obj)._get_delegate ();
      return new _ProvaJNIremotStub (delegate);
    }
  }
}
```

## 13.17. ARXIU *ProvaJNIremotHolder.java*

```
/**
* ProvaJNIremotHolder.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

public final class ProvaJNIremotHolder implements
org.omg.CORBA.portable.Streamable
{
  public ProvaJNIremot value = null;

  public ProvaJNIremotHolder ()
  {
  }

  public ProvaJNIremotHolder (ProvaJNIremot initialValue)
  {
    value = initialValue;
  }

  public void _read (org.omg.CORBA.portable.InputStream i)
  {
    value = ProvaJNIremotHelper.read (i);
  }

  public void _write (org.omg.CORBA.portable.OutputStream o)
  {
    ProvaJNIremotHelper.write (o, value);
  }

  public org.omg.CORBA.TypeCode _type ()
  {
    return ProvaJNIremotHelper.type ();
  }

}
```

## 13.18. ARXIU *ExcepcioJNI.java*

```
package ProvaJNIremotPackage;

/**
* ProvaJNIremotPackage/ExcepcioJNI.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

public final class ExcepcioJNI extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity
{

  public ExcepcioJNI ()
  {
  } // ctor

} // class ExcepcioJNI
```

## 13.19. ARXIU *ExcepcioJNIHelper.java*

```
package ProvaJNIremotPackage;

/**
* ProvaJNIremotPackage/ExcepcioJNIHelper.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

abstract public class ExcepcioJNIHelper
{
  private static String  _id = "IDL:ProvaJNIremot/ExcepcioJNI:1.0";

  public static void insert
         (org.omg.CORBA.Any a, ProvaJNIremotPackage.ExcepcioJNI that)
  {
    org.omg.CORBA.portable.OutputStream out =
                                          a.create_output_stream ();
    a.type (type ());
    write (out, that);
    a.read_value (out.create_input_stream (), type ());
  }

  public static ProvaJNIremotPackage.ExcepcioJNI extract
                                             (org.omg.CORBA.Any a)
  {
    return read (a.create_input_stream ());
  }

  private static org.omg.CORBA.TypeCode __typeCode = null;
  private static boolean __active = false;
```

```
  synchronized public static org.omg.CORBA.TypeCode type ()
  {
    if (__typeCode == null)
    {
      synchronized (org.omg.CORBA.TypeCode.class)
      {
        if (__typeCode == null)
        {
          if (__active)
          {
            return org.omg.CORBA.ORB.init().create_recursive_tc
                                                              ( _id );
          }
          __active = true;
          org.omg.CORBA.StructMember[] _members0 =
                                      new org.omg.CORBA.StructMember [0];
          org.omg.CORBA.TypeCode _tcOf_members0 = null;
          __typeCode = org.omg.CORBA.ORB.init ().create_struct_tc
(ProvaJNIremotPackage.ExcepcioJNIHelper.id (),"ExcepcioJNI", _members0);
          __active = false;
        }
      }
    }
    return __typeCode;
  }

  public static String id ()
  {
    return _id;
  }

  public static ProvaJNIremotPackage.ExcepcioJNI read
                           (org.omg.CORBA.portable.InputStream istream)
  {
    ProvaJNIremotPackage.ExcepcioJNI value =
                                  new ProvaJNIremotPackage.ExcepcioJNI ();
    // read and discard the repository ID
    istream.read_string ();
    return value;
  }

  public static void write (org.omg.CORBA.portable.OutputStream ostream,
                                       ProvaJNIremotPackage.ExcepcioJNI value)
  {
    // write the repository ID
    ostream.write_string (id ());
  }

}
```

## 13.20. ARXIU *ExcepcioJNIHolder.java*

```
package ProvaJNIremotPackage;

/**
* ProvaJNIremotPackage/ExcepcioJNIHolder.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from ProvaJNI.idl
*/

public final class ExcepcioJNIHolder implements
org.omg.CORBA.portable.Streamable
{
  public ProvaJNIremotPackage.ExcepcioJNI value = null;

  public ExcepcioJNIHolder (){}

  public ExcepcioJNIHolder
                        (ProvaJNIremotPackage.ExcepcioJNI initialValue)
  {
    value = initialValue;
  }

  public void _read (org.omg.CORBA.portable.InputStream i)
  {
    value = ProvaJNIremotPackage.ExcepcioJNIHelper.read (i);
  }

  public void _write (org.omg.CORBA.portable.OutputStream o)
  {
    ProvaJNIremotPackage.ExcepcioJNIHelper.write (o, value);
  }

  public org.omg.CORBA.TypeCode _type ()
  {
    return ProvaJNIremotPackage.ExcepcioJNIHelper.type ();
  }

}
```

## 13.21. ARXIU *ProvaJNIservent.java*

```
class ProvaJNIservent extends _ProvaJNIremotImplBase
{
   private ProvaJNI pjni;

   public ProvaJNIservent()
   {
         synchronized(this)
         {
               pjni=new ProvaJNI();
               pjni.suma(0,0);
         }
   }

   public int suma(int a,int b)
   {
         synchronized(this)
         {
               return pjni.suma(a,b);
         }
   }

   public int resta(int a,int b)
   {
         synchronized(this)
         {
               return pjni.resta(a,b);
         }
   }

   public int producte(int a,int b)
   {
         return pjni.producte(a,b);
   }

   public int quocient(int a,int b)throws ProvaJNIremotPackage.ExcepcioJNI
   {
         int resultat;
         try{
               synchronized(this)
               {

                     resultat=pjni.quocient(a,b);
               }
         }catch(Exception e){
               String m=e.getMessage();
               throw new ProvaJNIremotPackage.ExcepcioJNI();
         }

         return resultat;
   }

   public int getResultat()
   {
         return pjni.getResultat();
   }
}
```

## 13.22. ARXIU *ProvaJNIservidor.java*

```
//paquets necessaris pel funcionament d'aplicaions servidor CORBA
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;

public class ProvaJNIservidor
{
  public static void main(String args[])
  {
    try{

      /* Creem i inicialitzem l'objecte de tipus ORB
      amb els parametres passats al metode main:
      port i adreça Servidor de Noms */
      ORB orb = ORB.init(args, null);

     // creacio de l'objecte servent i registre del mateix amb l'ORB
      ProvaJNIservent instServidor = new ProvaJNIservent();
      orb.connect(instServidor);

      // Obtenim el context de noms de l'arrel
      org.omg.CORBA.Object objRef =
                          orb.resolve_initial_references("NameService");
      NamingContext ncRef = NamingContextHelper.narrow(objRef);

     // bateig de la referencia de l'objecte a la xarxa
      NameComponent nc = new NameComponent("ProvaJNIremot", "");
      NameComponent path[] = {nc};
      ncRef.rebind(path, instServidor);

      // Esperar invocacions per part de clients
      java.lang.Object sync = new java.lang.Object();

      synchronized(sync)
      {
        sync.wait();
      }

    } catch(Exception e) {
        System.err.println("ERROR: " + e);
        e.printStackTrace(System.out);
      }
  }
}
```

## 13.23. ARXIU *ProvaJNIclient.java*

```
//paquets necessaris pel funcionament d'aplicaions client CORBA
import org.omg.CosNaming.*;
import org.omg.CORBA.*;

public class ProvaJNIclient
{
  public static void main(String args[])
  {
    try{

      /* Creem i inicialitzem l'objecte de tipus ORB
      amb els parametres passats al metode main:
      port i adreça Servidor de Noms */
      ORB orb = ORB.init(args, null);

      // Obtenim el context de noms de l'arrel
      org.omg.CORBA.Object objRef =
                          orb.resolve_initial_references("NameService");
      NamingContext ncRef = NamingContextHelper.narrow(objRef);

      //Obtenim la referencia de l'objecte remot
      NameComponent nc = new NameComponent("ProvaJNIremot", "");
      NameComponent path[] = {nc};
      ProvaJNIremot inst = ProvaJNIremotHelper.narrow(ncRef.resolve(path));

      //Cridem els metodes remotament
      //i mostrem els resultats

     int resultatNatiu, resultatJava;

     inst.suma(70,19);
     inst.resta(70,19);

     resultatNatiu=inst.producte(70,5);
     resultatJava=inst.getResultat();
     System.out.println("La funcio nativa retorna "+resultatNatiu);
     System.out.println
               ("i la variable d'objecte resultat val: "+resultatJava);

    } catch(Exception e) {
        System.out.println("ERROR : " + e);
        e.printStackTrace(System.out);
      }
  }
}
```

# 13.24. LINEALITZACIÓ DEL MODEL DEL PROCÉS ENTORN D'UN PUNT DE TREBALL.

Per tal d'obtenir una funció de transferència que relacioni la posició del carro respecte l'acció de control, partint de les equacions d'estat del sistema, es procedeix a la linealització entorn d'un punt de treball de les variables responsables d'aquesta no linealitat. Les equacions d'estat son:

[1] $x_1' = x_3$

[2] $$x_3' = \frac{a*(F - f_c*x_3 - \mu*x_4^{\,2}*\sin(x_2)) + l*\cos(x_2)*(\mu*g*\sin(x_2) - f_p*x_4)}{J + \mu*l*\sin^2(x_2)}$$

[3] $x_2' = x_4$

[4] $$x_4' = \frac{l*\cos(x_2)*(F - f_c*x_3 - \mu*x_4^{\,2}*\sin(x_2)) + \mu*g*\sin(x_2) - f_p*x_4}{J + \mu*l*\sin^2(x_2)}$$

on:

$$a = l^2 + \frac{J}{m_c + m_p} \quad , \quad \mu = (m_c + m_p)*l$$

essent:

| | |
|---|---|
| $X_1$ | Posició del carro. |
| $X_3$ | Velocitat del carro. |
| $X_2$ | Posició angular del pèndul. |
| $X_4$ | Velocitat angular del pèndul. |

El significat de les sigles dels diversos paràmetres és l'especificada a l'apartat 1.3 del capítol 3 de la memòria.

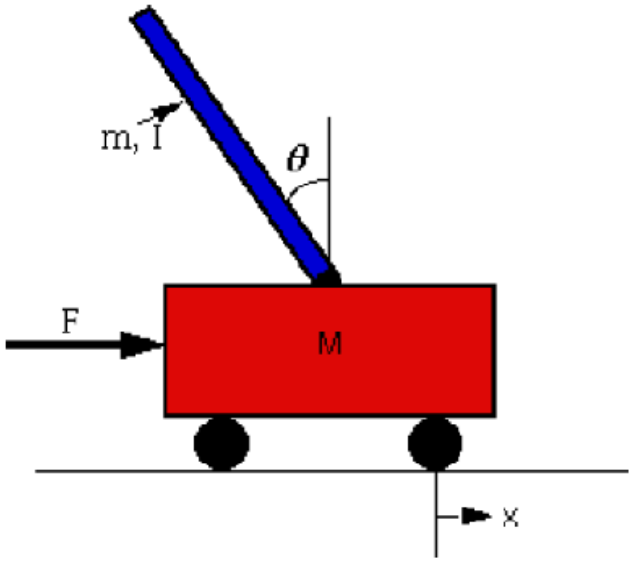
Oblidant-nos temporalment de la primera equació, i substituint la velocitat angular del pèndul ($x_4$) per la derivada de la posició angular ($x_2$'), el sistema queda reduït a dues equacions:

[2] $$x_3' = \frac{a(F - f_c x_3 - \mu(x_2')^2 \sin(x_2)) + l\cos(x_2)(\mu g \sin(x_2) - f_p x_2')}{J + \mu l \sin^2(x_2)}$$

[4] $$x_2'' = \frac{l\cos(x_2)(F - f_c x_3 - \mu(x_2')^2 \sin(x_2)) + \mu g \sin(x_2) - f_p x_2'}{J + \mu l \sin^2(x_2)}$$

A partir d'aquestes dues equacions obtindrem la funció de transferència que relaciona la velocitat del carro respecte l'acció de control. Posteriorment la posició del carro s'obtindrà integrant la velocitat.

La variable que provoca la no linealitat és l'angle del pèndul ($x_2$), de manera que aproximarem les equacions entorn del punt de treball al voltant d'on es procurarà que treballi el pèndol. Com que l'objectiu és el control de posició del carro, si el moviment del mateix no és massa brusc, el pèndul estarà al voltant del punt $\theta = \pi rad.$ , on $\theta$ és l'angle del pèndul. (a les equacions d'estat $\theta \rightarrow x_2$)




Ens trobem amb 4 termes no lineals a les equacions:

$\sin(x_2)$

$\cos(x_2)$

$\sin^2(x_2)$

$(x_2')^2$

D'acord amb les aproximacions de Taylor, prenent els dos primers termes, els de grau 0 i 1, l'aproximació seria:

$f(x) = f(x_0) + \Delta x * f'(x_0)$ on $\Delta x = x - x_0$ i $x_0$ és el punt de treball (en aquest cas $\pi$).

De forma que els 4 termes no lineals anteriors es podrien aproximar com a :

Si $x_2$ està al voltant de $\pi$ rad., llavors:

$$\sin(x_2) \approx \sin(\pi) + \Delta x_2 * \cos(\pi) = -\Delta x_2$$

$$\cos(x_2) \approx \cos(\pi) + \Delta x_2 * (-1) * \sin(\pi) = -1$$

$$\sin^2(x_2) \approx \sin^2(\pi) + \Delta x_2 * (2 * \sin(\pi) * \cos(\pi)) = 0^2 + \Delta x_2 * 2 * 0 * (-1) = 0$$

$x_2$' és la velocitat del pèndul; si el pèndol pràcticament no es mourà, llavors

$$x_2' \cong 0 \Rightarrow (x_2')^2 \cong 0$$

Les equacions queden doncs:

[2] $$x_3' = \frac{a(F - f_c x_3) - l(\mu g(-\Delta x_2) - f_p x_2')}{J}$$

[4] $$x_2'' = \frac{-l(F - f_c x_3) + \mu g(-\Delta x_2) - f_p x_2'}{J}$$

Tenint en compte que la derivada d'una funció amb valors al voltant d'un punt, té relació amb l'increment i no pas amb el valor del punt, podem afirmar que $x_2' = (\Delta x_2)'$

D'aquesta manera les equacions quedarien amb dues variables ($x_3$) i ($\Delta x_2$):

[2] $$x_3' = \frac{a(F - f_c x_3) - l(\mu g(-\Delta x_2) - f_p (\Delta x_2)')}{J}$$

[4] $$(\Delta x_2)'' = \frac{-l(F - f_c x_3) + \mu g(-\Delta x_2) - f_p (\Delta x_2)'}{J}$$

Com que aquestes equacions diferencials son lineals, podem aplicar la transformada de Laplace:

[2] $$x_3(s)s = \frac{a(F - f_c x_3(s)) - l(\mu g(-\Delta x_2(s)) - f_p \Delta x_2(s)s)}{J}$$

[4] $$\Delta x_2(s)s^2 = \frac{-l(F - f_c x_3(s)) + \mu g(-\Delta x_2(s)) - f_p \Delta x_2(s)s}{J}$$

Aïllant $\Delta x_2$ a l'equació [4] i substituint a l'equació [2], obtenim la relació $\frac{x_3(s)}{F(s)}$

$$\frac{X_3(s)}{F(s)} = \frac{(aJ)s^2 + (af_p - l^2 f_p)s + (a\mu g - l^2 \mu g)}{\left[J^2 s^3 + (f_p J + f_c a J)s^2 + (J\mu g + f_c a f_p - f_c l^2 f_p)s + (f_c a \mu g - f_c l^2 \mu g)\right]}$$

Donat que la força que exerceix l'actuador és proporcional a l'acció de control en la forma $F(s) = 17N * U(s)$ on $|U| \leq 1$ , l'equació queda en la forma:

$$\frac{X_3(s)}{U(s)} = \frac{17*\left[(aJ)s^2 + (af_p - l^2 f_p)s + (a\mu g - l^2 \mu g)\right]}{\left[J^2 s^3 + (f_p J + f_c aJ)s^2 + (J\mu g + f_c af_p - f_c l^2 f_p)s + (f_c a\mu g - f_c l^2 \mu g)\right]}$$

Finalment, com que $x_3$=$x_1$' que en Laplace s'expressaria: $X_3(s) = X_1(s) * s$

La relació $x_1$ , que anotarem com a **x**, i **u** serà:

$$\frac{X(s)}{U(s)} = \frac{17*\left[(aJ)s^2 + (af_p - l^2 f_p)s + (a\mu g - l^2 \mu g)\right]}{s\left[J^2 s^3 + (f_p J + f_c aJ)s^2 + (J\mu g + f_c af_p - f_c l^2 f_p)s + (f_c a\mu g - f_c l^2 \mu g)\right]}$$

## 13.25. ARXIU *FBK33200local.java*

```
public class FBK33200local
{
   public static final int CAP=0;
   public static final int LLAC_OBERT=1;
   public static final int LQ_INVERTIT=2;
   public static final int LQ_GRUA=3;
   public static final int FUZZY=4;
   public static final int PID=5;
   public static final int CONTROL_OPTIM=6;
   public static final int CONTROL_ADAPTATIU=7;
   public static final int EXTERN=99;

   public static final int CONSTANT=0;
   public static final int SENYAL_QUADRADA=1;
   public static final int SENYAL_TRIANGULAR=2;
   public static final int SENYAL_SINUSOIDAL=3;
   public static final int SENYAL_PERIODIC_ALEATORI=4;


   private native void carregaMatlab()throws ExcepcioFBK33200;
   private native void descarregaMatlab();

   public FBK33200local()throws ExcepcioFBK33200
   {
         this.carregaMatlab();
         this.paraCarro();
         this.comptadorsAzero();
         this.tempsAzero();
   }

   public FBK33200local(int adrecaPCL812PG)throws ExcepcioFBK33200
   {
         this.carregaMatlab();
         this.setAdrPCL812PG(adrecaPCL812PG);
         this.paraCarro();
         this.comptadorsAzero();
         this.tempsAzero();
   }

   public void finalize()
   {
         descarregaMatlab();
   }

   public native int getTipusControl()throws ExcepcioFBK33200;
   public native int getAdrPCL812PG()throws ExcepcioFBK33200;
   public native float[] getParFiltrePosCarro()throws ExcepcioFBK33200;
   public native float[] getParFiltreVelCarro()throws ExcepcioFBK33200;

   public native float getConsigna()throws ExcepcioFBK33200;
   public native int getDivisorRTK()throws ExcepcioFBK33200;

   public native float[] getHistorialTemps()throws ExcepcioFBK33200;
   public native float[] getHistorialAnglePendul()throws ExcepcioFBK33200;
   public native float[] getHistorialVelPendul()throws ExcepcioFBK33200;
   public native float[] getHistorialPosCarro()throws ExcepcioFBK33200;
   public native float[] getHistorialVelCarro()throws ExcepcioFBK33200;
   public native float[] getHistorialAccioControlRelativa()
```

```
                                                    throws ExcepcioFBK33200;
   public native float[] getHistorialConsigna()throws ExcepcioFBK33200;

   public native int getNumMostresBufer()throws ExcepcioFBK33200;
   public native float[] getParametresControl()throws ExcepcioFBK33200;

   public native float[] getParFiltrePosPendul()throws ExcepcioFBK33200;
   public native float[] getParFiltreVelPendul()throws ExcepcioFBK33200;

   public native float[] getPW()throws ExcepcioFBK33200;
   public native float getPeriodeMostreig()throws ExcepcioFBK33200;
   public native int carregaLlibreria()throws ExcepcioFBK33200;
   public native void carregaControladorExtern(String fitxer)
                                                    throws ExcepcioFBK33200;
   public native void comptadorsAzero()throws ExcepcioFBK33200;
   public native void tempsAzero()throws ExcepcioFBK33200;
   public native void setTipusControl(int tipus)throws ExcepcioFBK33200;
   public native void setAdrPCL812PG(int adreca)throws ExcepcioFBK33200;

   public native void setParFiltrePosCarro(float[] parametres)
                                                    throws ExcepcioFBK33200;
   public native void setParFiltreVelCarro(float[] parametres)
                                                    throws ExcepcioFBK33200;

   public native void setConsigna(float r)throws ExcepcioFBK33200;
   public native void setDivisorRTK(int divisor)throws ExcepcioFBK33200;
   public native void setParametresControl(float[] parametres)
                                                    throws ExcepcioFBK33200;

   public native void setParFiltrePosPendul(float[] parametres)
                                                    throws ExcepcioFBK33200;
   public native void setParFiltreVelPendul(float[] parametres)
                                                    throws ExcepcioFBK33200;

   public native void setPW(float[] parametres)throws ExcepcioFBK33200;
   public native void setPeriodeMostreig(float ts)throws ExcepcioFBK33200;
   public native void iniciarAdquisicioDades()throws ExcepcioFBK33200;
   public native void paraCarro()throws ExcepcioFBK33200;
   public native int descarregaLlibreria()throws ExcepcioFBK33200;

   static
   {
         System.loadLibrary("FBK33200local");
   }
}
```

## 13.26. ARXIU *ExcepcioFBK33200.java*

```
public class ExcepcioFBK33200 extends Exception
{
   public ExcepcioFBK33200()
   {
         super();
   }
   public ExcepcioFBK33200(String missatgeError)
   {
         super(missatgeError);
   }
}
```

## 13.27. ARXIU *FBK33200local.h*

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class FBK33200local */

#ifndef _Included_FBK33200local
#define _Included_FBK33200local
#ifdef __cplusplus
extern "C" {
#endif
#undef FBK33200local_CAP
#define FBK33200local_CAP 0L
#undef FBK33200local_LLAC_OBERT
#define FBK33200local_LLAC_OBERT 1L
#undef FBK33200local_LQ_INVERTIT
#define FBK33200local_LQ_INVERTIT 2L
#undef FBK33200local_LQ_GRUA
#define FBK33200local_LQ_GRUA 3L
#undef FBK33200local_FUZZY
#define FBK33200local_FUZZY 4L
#undef FBK33200local_PID
#define FBK33200local_PID 5L
#undef FBK33200local_CONTROL_OPTIM
#define FBK33200local_CONTROL_OPTIM 6L
#undef FBK33200local_CONTROL_ADAPTATIU
#define FBK33200local_CONTROL_ADAPTATIU 7L
#undef FBK33200local_EXTERN
#define FBK33200local_EXTERN 99L
/*
 * Class:     FBK33200local
 * Method:    carregaMatlab
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_FBK33200local_carregaMatlab
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    descarregaMatlab
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_FBK33200local_descarregaMatlab
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getTipusControl
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_FBK33200local_getTipusControl
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getAdrPCL812PG
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_FBK33200local_getAdrPCL812PG
  (JNIEnv *, jobject);
```

```
/*
 * Class:     FBK33200local
 * Method:    getParFiltrePosCarro
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParFiltrePosCarro
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getParFiltreVelCarro
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParFiltreVelCarro
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getConsigna
 * Signature: ()F
 */
JNIEXPORT jfloat JNICALL Java_FBK33200local_getConsigna
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getDivisorRTK
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_FBK33200local_getDivisorRTK
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getHistorialTemps
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialTemps
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getHistorialAnglePendul
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialAnglePendul
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getHistorialVelPendul
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialVelPendul
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getHistorialPosCarro
 * Signature: ()[F
 */
```

```
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialPosCarro
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getHistorialVelCarro
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialVelCarro
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getHistorialAccioControlRelativa
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL
Java_FBK33200local_getHistorialAccioControlRelativa
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getHistorialConsigna
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialConsigna
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getNumMostresBufer
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_FBK33200local_getNumMostresBufer
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getParametresControl
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParametresControl
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getParFiltrePosPendul
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParFiltrePosPendul
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getParFiltreVelPendul
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParFiltreVelPendul
  (JNIEnv *, jobject);
```

```
/*
 * Class:     FBK33200local
 * Method:    getPW
 * Signature: ()[F
 */
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getPW
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    getPeriodeMostreig
 * Signature: ()F
 */
JNIEXPORT jfloat JNICALL Java_FBK33200local_getPeriodeMostreig
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    carregaLlibreria
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_FBK33200local_carregaLlibreria
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    carregaControladorExtern
 * Signature: (Ljava/lang/String;)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_carregaControladorExtern
  (JNIEnv *, jobject, jstring);

/*
 * Class:     FBK33200local
 * Method:    comptadorsAzero
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_FBK33200local_comptadorsAzero
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    tempsAzero
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_FBK33200local_tempsAzero
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    setTipusControl
 * Signature: (I)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setTipusControl
  (JNIEnv *, jobject, jint);

/*
 * Class:     FBK33200local
 * Method:    setAdrPCL812PG
 * Signature: (I)V
 */
```

```
JNIEXPORT void JNICALL Java_FBK33200local_setAdrPCL812PG
  (JNIEnv *, jobject, jint);

/*
 * Class:     FBK33200local
 * Method:    setParFiltrePosCarro
 * Signature: ([F)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setParFiltrePosCarro
  (JNIEnv *, jobject, jfloatArray);

/*
 * Class:     FBK33200local
 * Method:    setParFiltreVelCarro
 * Signature: ([F)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setParFiltreVelCarro
  (JNIEnv *, jobject, jfloatArray);

/*
 * Class:     FBK33200local
 * Method:    setConsigna
 * Signature: (F)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setConsigna
  (JNIEnv *, jobject, jfloat);

/*
 * Class:     FBK33200local
 * Method:    setDivisorRTK
 * Signature: (I)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setDivisorRTK
  (JNIEnv *, jobject, jint);

/*
 * Class:     FBK33200local
 * Method:    setParametresControl
 * Signature: ([F)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setParametresControl
  (JNIEnv *, jobject, jfloatArray);

/*
 * Class:     FBK33200local
 * Method:    setParFiltrePosPendul
 * Signature: ([F)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setParFiltrePosPendul
  (JNIEnv *, jobject, jfloatArray);

/*
 * Class:     FBK33200local
 * Method:    setParFiltreVelPendul
 * Signature: ([F)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setParFiltreVelPendul
  (JNIEnv *, jobject, jfloatArray);
```

```
/*
 * Class:     FBK33200local
 * Method:    setPW
 * Signature: ([F)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setPW
  (JNIEnv *, jobject, jfloatArray);

/*
 * Class:     FBK33200local
 * Method:    setPeriodeMostreig
 * Signature: (F)V
 */
JNIEXPORT void JNICALL Java_FBK33200local_setPeriodeMostreig
  (JNIEnv *, jobject, jfloat);

/*
 * Class:     FBK33200local
 * Method:    iniciarAdquisicioDades
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_FBK33200local_iniciarAdquisicioDades
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    paraCarro
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_FBK33200local_paraCarro
  (JNIEnv *, jobject);

/*
 * Class:     FBK33200local
 * Method:    descarregaLlibreria
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_FBK33200local_descarregaLlibreria
  (JNIEnv *, jobject);

#ifdef __cplusplus
}
#endif
#endif
```

## 13.28. ARXIU *FBK33200local.cpp*

```
#include "FBK33200local.h"
#include <string.h>
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include "engine.h"
#include "mex.h"

Engine *matlab;
bool matlabCarregat=false;


BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                            )
{
    return TRUE;
}


JNIEXPORT void JNICALL Java_FBK33200local_carregaMatlab
  (JNIEnv *env, jobject obj)
{
    if(!(matlab = engOpen("\0")))
    {
        char missatgeError[80]="No s'ha pogut carregar Matlab";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }
    else
    {
        matlabCarregat=true;
    }
}

JNIEXPORT void JNICALL Java_FBK33200local_descarregaMatlab
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==true)
    {
        engClose(matlab);
        matlabCarregat=false;
    }
}

JNIEXPORT jint JNICALL Java_FBK33200local_getTipusControl
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");
```

```
        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        return 999;
    }

    jint resultat;
    mxArray *mxArray_tc=NULL;
    engEvalString(matlab,"tc=pd_call('GetAlgNo');");
    mxArray_tc=engGetArray(matlab,"tc");
    resultat=(jint)mxGetScalar(mxArray_tc);

    return resultat;
}

JNIEXPORT jint JNICALL Java_FBK33200local_getAdrPCL812PG
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return 999;
    }

    jint resultat;
    mxArray *mxArray_adr=NULL;
    engEvalString(matlab,"adr=pd_call('GetBaseAddress');");
    mxArray_adr=engGetArray(matlab,"adr");
    resultat=(jint)mxGetScalar(mxArray_adr);
    return resultat;
}

JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParFiltrePosCarro
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_parametres=NULL;

    engEvalString(matlab,"parametres=pd_call('GetCartPosFilt');");
    mxArray_parametres=engGetArray(matlab,"parametres");
    partReal=(double *)mxGetPr(mxArray_parametres);
```

```
    int tamany_parametres;
    tamany_parametres =
mxGetM(mxArray_parametres)*mxGetN(mxArray_parametres);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}

JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParFiltreVelCarro
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_parametres=NULL;

    engEvalString(matlab,"parametres=pd_call('GetCartSpeedFilt');");
    mxArray_parametres=engGetArray(matlab,"parametres");
    partReal=(double *)mxGetPr(mxArray_parametres);

    int tamany_parametres;
    tamany_parametres =
mxGetM(mxArray_parametres)*mxGetN(mxArray_parametres);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jfloat JNICALL Java_FBK33200local_getConsigna
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return 999;
    }

    jfloat resultat;
    mxArray *mxArray_r=NULL;
    engEvalString(matlab,"r=pd_call('GetDesPosition');");
    mxArray_r=engGetArray(matlab,"r");
    resultat=(jfloat)mxGetScalar(mxArray_r);
    return resultat;
}

JNIEXPORT jint JNICALL Java_FBK33200local_getDivisorRTK
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return 999;
    }

    jint resultat;
    mxArray *mxArray_divisor=NULL;
    engEvalString(matlab,"div=pd_call('GetDivider');");
    mxArray_divisor=engGetArray(matlab,"div");
    resultat=(jint)mxGetScalar(mxArray_divisor);
    return resultat;
}


#define FILES_HISTORIAL 7

#define FILA_TEMPS 0
#define FILA_ANGLE_PENDUL 1
#define FILA_VEL_PENDUL 2
#define FILA_POS_CARRO 3
#define FILA_VEL_CARRO 4
#define FILA_ACCIO_CONTROL 5
#define FILA_CONSIGNA 6
```

```
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialTemps
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_historial=NULL;

    engEvalString(matlab,"hist=pd_call('GetHistory');");
    mxArray_historial=engGetArray(matlab,"hist");
    partReal=(double *)mxGetPr(mxArray_historial);

    if(mxGetM(mxArray_historial)!=FILES_HISTORIAL)
    {
        char missatgeError[80]="S'ha produit un error amb el BUFFER de
l'historial de mesures";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    int tamany_parametres;
    tamany_parametres = mxGetN(mxArray_historial);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i*FILES_HISTORIAL +
FILA_TEMPS];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialAnglePendul
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_historial=NULL;

    engEvalString(matlab,"hist=pd_call('GetHistory');");
    mxArray_historial=engGetArray(matlab,"hist");
    partReal=(double *)mxGetPr(mxArray_historial);

    if(mxGetM(mxArray_historial)!=FILES_HISTORIAL)
    {
        char missatgeError[80]="S'ha produit un error amb el BUFFER de
l'historial de mesures";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    int tamany_parametres;
    tamany_parametres = mxGetN(mxArray_historial);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i*FILES_HISTORIAL +
FILA_ANGLE_PENDUL];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialVelPendul
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_historial=NULL;

    engEvalString(matlab,"hist=pd_call('GetHistory');");
    mxArray_historial=engGetArray(matlab,"hist");
    partReal=(double *)mxGetPr(mxArray_historial);

    if(mxGetM(mxArray_historial)!=FILES_HISTORIAL)
    {
        char missatgeError[80]="S'ha produit un error amb el BUFFER de
l'historial de mesures";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    int tamany_parametres;
    tamany_parametres = mxGetN(mxArray_historial);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i*FILES_HISTORIAL +
FILA_VEL_PENDUL];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialPosCarro
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_historial=NULL;

    engEvalString(matlab,"hist=pd_call('GetHistory');");
    mxArray_historial=engGetArray(matlab,"hist");
    partReal=(double *)mxGetPr(mxArray_historial);

    if(mxGetM(mxArray_historial)!=FILES_HISTORIAL)
    {
        char missatgeError[80]="S'ha produit un error amb el BUFFER de
l'historial de mesures";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    int tamany_parametres;
    tamany_parametres = mxGetN(mxArray_historial);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i*FILES_HISTORIAL +
FILA_POS_CARRO];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialVelCarro
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_historial=NULL;

    engEvalString(matlab,"hist=pd_call('GetHistory');");
    mxArray_historial=engGetArray(matlab,"hist");
    partReal=(double *)mxGetPr(mxArray_historial);

    if(mxGetM(mxArray_historial)!=FILES_HISTORIAL)
    {
        char missatgeError[80]="S'ha produit un error amb el BUFFER de
l'historial de mesures";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    int tamany_parametres;
    tamany_parametres = mxGetN(mxArray_historial);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i*FILES_HISTORIAL +
FILA_VEL_CARRO];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jfloatArray JNICALL
Java_FBK33200local_getHistorialAccioControlRelativa
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_historial=NULL;

    engEvalString(matlab,"hist=pd_call('GetHistory');");
    mxArray_historial=engGetArray(matlab,"hist");
    partReal=(double *)mxGetPr(mxArray_historial);

    if(mxGetM(mxArray_historial)!=FILES_HISTORIAL)
    {
        char missatgeError[80]="S'ha produit un error amb el BUFFER de
l'historial de mesures";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    int tamany_parametres;
    tamany_parametres = mxGetN(mxArray_historial);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i*FILES_HISTORIAL +
FILA_ACCIO_CONTROL];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getHistorialConsigna
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_historial=NULL;

    engEvalString(matlab,"hist=pd_call('GetHistory');");
    mxArray_historial=engGetArray(matlab,"hist");
    partReal=(double *)mxGetPr(mxArray_historial);

    if(mxGetM(mxArray_historial)!=FILES_HISTORIAL)
    {
        char missatgeError[80]="S'ha produit un error amb el BUFFER de
l'historial de mesures";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    int tamany_parametres;
    tamany_parametres = mxGetN(mxArray_historial);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i*FILES_HISTORIAL +
FILA_CONSIGNA];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jint JNICALL Java_FBK33200local_getNumMostresBufer
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return 999;
    }

    jint resultat;
    mxArray *mxArray_mostres=NULL;
    engEvalString(matlab,"mostres=pd_call('GetNoOfSamples');");
    mxArray_mostres=engGetArray(matlab,"mostres");
    resultat=(jint)mxGetScalar(mxArray_mostres);
    return resultat;
}

JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParametresControl
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }
    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_parametres=NULL;

    engEvalString(matlab,"parametres=pd_call('GetP');");
    mxArray_parametres=engGetArray(matlab,"parametres");
    partReal=(double *)mxGetPr(mxArray_parametres);

    int tamany_parametres;
    tamany_parametres =
mxGetM(mxArray_parametres)*mxGetN(mxArray_parametres);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for(int i=0; i<tamany_parametres; i++)
    {
         partReal_Jfloat[i]=(jfloat)partReal[i];
    }
    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}
```

```
JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParFiltrePosPendul
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_parametres=NULL;

    engEvalString(matlab,"parametres=pd_call('GetPendPosFilt');");
    mxArray_parametres=engGetArray(matlab,"parametres");
    partReal=(double *)mxGetPr(mxArray_parametres);

    int tamany_parametres;
    tamany_parametres =
mxGetM(mxArray_parametres)*mxGetN(mxArray_parametres);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}

JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getParFiltreVelPendul
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
```

```
    jfloat *partReal_Jfloat;
    mxArray *mxArray_parametres=NULL;

    engEvalString(matlab,"parametres=pd_call('GetPendSpeedFilt');");
    mxArray_parametres=engGetArray(matlab,"parametres");
    partReal=(double *)mxGetPr(mxArray_parametres);

    int tamany_parametres;
    tamany_parametres =
mxGetM(mxArray_parametres)*mxGetN(mxArray_parametres);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
    delete[] partReal_Jfloat;
    return resultat;
}

JNIEXPORT jfloatArray JNICALL Java_FBK33200local_getPW
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        jfloatArray resultatDefectuos;
        resultatDefectuos=(*env).NewFloatArray(1);
        return resultatDefectuos;
    }

    jfloatArray resultat;
    double *partReal;
    jfloat *partReal_Jfloat;
    mxArray *mxArray_parametres=NULL;

    engEvalString(matlab,"parametres=pd_call('GetPW');");
    mxArray_parametres=engGetArray(matlab,"parametres");
    partReal=(double *)mxGetPr(mxArray_parametres);

    int tamany_parametres;
    tamany_parametres =
mxGetM(mxArray_parametres)*mxGetN(mxArray_parametres);
    partReal_Jfloat=new jfloat[tamany_parametres];
    for (int i=0; i<tamany_parametres; i++)
    {
        partReal_Jfloat[i]=(jfloat)partReal[i];
    }

    resultat=(*env).NewFloatArray(tamany_parametres);
    (*env).SetFloatArrayRegion(resultat, 0,
tamany_parametres,partReal_Jfloat);
```

```
    delete[] partReal_Jfloat;
    return resultat;
}

JNIEXPORT jfloat JNICALL Java_FBK33200local_getPeriodeMostreig
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return 999;
    }

    jfloat resultat;
    mxArray *mxArray_ts=NULL;
    engEvalString(matlab,"ts=pd_call('GetSampleTime');");
    mxArray_ts=engGetArray(matlab,"ts");
    resultat=(jfloat)mxGetScalar(mxArray_ts);
    return resultat;
}

JNIEXPORT jint JNICALL Java_FBK33200local_carregaLlibreria
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return 999;
    }

    jint resultat;
    mxArray *mxArray_contCarregues=NULL;
    engEvalString(matlab,"cont=pd_call('LoadLibrary');");
    mxArray_contCarregues=engGetArray(matlab,"cont");
    resultat=(jint)mxGetScalar(mxArray_contCarregues);
    return resultat;
}

JNIEXPORT void JNICALL Java_FBK33200local_carregaControladorExtern
  (JNIEnv *env, jobject obj, jstring fitxer)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }
```

```
    mxArray *mxArray_fitxer;
    const char *fitxerASCI = (*env).GetStringUTFChars(fitxer, 0);
    mxArray_fitxer = mxCreateString(fitxerASCI);
    mxSetName(mxArray_fitxer, "nomFitxer");
    engPutArray(matlab,mxArray_fitxer);
    engEvalString(matlab,"pd_call('LoadExtAlg',nomFitxer);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="No s'ha pogut carregar el Controlador
Extern";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    mxDestroyArray(mxArray_fitxer);
}

JNIEXPORT void JNICALL Java_FBK33200local_comptadorsAzero
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    engEvalString(matlab,"pd_call('ResetEncoder');");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }
}

JNIEXPORT void JNICALL Java_FBK33200local_tempsAzero
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");
```

```
        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    engEvalString(matlab,"pd_call('ResetTime');");
}

JNIEXPORT void JNICALL Java_FBK33200local_setTipusControl
  (JNIEnv *env, jobject obj, jint tipus)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    double valorsReals[]={(double)tipus};
    mxArray *mxArray_tc=NULL;
    double  *partReal;
    mxArray_tc=mxCreateDoubleMatrix(1,1,mxREAL);
    mxSetName(mxArray_tc,"tc");
    partReal = (double *)mxGetPr(mxArray_tc);
    memcpy(partReal, valorsReals, 1 * sizeof(double) );
    engPutArray(matlab,mxArray_tc);
    engEvalString(matlab,"pd_call('SetAlgNo',tc);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_tc);
}

JNIEXPORT void JNICALL Java_FBK33200local_setAdrPCL812PG
  (JNIEnv *env, jobject obj, jint adreca)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }
```

```
    double valorsReals[]={(double)adreca};
    mxArray *mxArray_adr=NULL;
    double  *partReal;
    mxArray_adr=mxCreateDoubleMatrix(1,1,mxREAL);
    mxSetName(mxArray_adr,"adr");
    partReal = (double *)mxGetPr(mxArray_adr);
    memcpy(partReal, valorsReals, 1 * sizeof(double) );
    engPutArray(matlab,mxArray_adr);
    engEvalString(matlab,"pd_call('SetBaseAddress',adr);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_adr);
}

JNIEXPORT void JNICALL Java_FBK33200local_setParFiltrePosCarro
  (JNIEnv *env, jobject obj, jfloatArray parametres)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    double valorsReals[18];
    int numParametres;
    jfloat *parametres_Jfloat;

    numParametres=(*env).GetArrayLength(parametres);
    if(numParametres!=18)
    {
        char missatgeError[80]="Numero de parametres de la funcio
setParFiltrePosCarro incorrecte";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        return;
    }

    parametres_Jfloat=(*env).GetFloatArrayElements(parametres,0);

    int i;
    for(i=0; i<18; i++)
    {
```

```
        if(i==1 && parametres_Jfloat[i]!=0)
        {
                char missatgeError[80]="Filtre digital de Posicio del
Carro NO CAUSAL!";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
                return;
        }
        valorsReals[i]=(double)parametres_Jfloat[i];
    }

    mxArray *mxArray_parametres=NULL;
    double  *partReal;
    mxArray_parametres=mxCreateDoubleMatrix(2,9,mxREAL);
    mxSetName(mxArray_parametres,"parametres");
    partReal = (double *)mxGetPr(mxArray_parametres);
    memcpy(partReal, valorsReals, 2*9*sizeof(double) );
    engPutArray(matlab,mxArray_parametres);
    engEvalString(matlab,"pd_call('SetCartPosFilt',parametres);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_parametres);
}

JNIEXPORT void JNICALL Java_FBK33200local_setParFiltreVelCarro
  (JNIEnv *env, jobject obj, jfloatArray parametres)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    double valorsReals[18];
    int numParametres;
    jfloat *parametres_Jfloat;

    numParametres=(*env).GetArrayLength(parametres);
    if(numParametres!=18)
    {
        char missatgeError[80]="Numero de parametres de la funcio
setParFiltreVelCarro incorrecte";
        jclass ExcepcioFBK33200;
```

```
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        return;
    }

    parametres_Jfloat=(*env).GetFloatArrayElements(parametres,0);

    int i;
    for(i=0; i<18; i++)
    {
        if(i==1 && parametres_Jfloat[i]!=0)
        {
                char missatgeError[80]="Filtre digital de Velocitat del
Carro NO CAUSAL!";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
                return;
        }
        valorsReals[i]=(double)parametres_Jfloat[i];
    }

    mxArray *mxArray_parametres=NULL;
    double  *partReal;
    mxArray_parametres=mxCreateDoubleMatrix(2,9,mxREAL);
    mxSetName(mxArray_parametres,"parametres");
    partReal = (double *)mxGetPr(mxArray_parametres);
    memcpy(partReal, valorsReals, 2*9*sizeof(double) );
    engPutArray(matlab,mxArray_parametres);
    engEvalString(matlab,"pd_call('SetCartSpeedFilt',parametres);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_parametres);
}

JNIEXPORT void JNICALL Java_FBK33200local_setConsigna
  (JNIEnv *env, jobject obj, jfloat r)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
```

```
    }

    double valorsReals[]={(double)r};
    mxArray *mxArray_consigna=NULL;
    double  *partReal;
    mxArray_consigna=mxCreateDoubleMatrix(1,1,mxREAL);
    mxSetName(mxArray_consigna,"r");
    partReal = (double *)mxGetPr(mxArray_consigna);
    memcpy(partReal, valorsReals, 1 * sizeof(double) );
    engPutArray(matlab,mxArray_consigna);
    engEvalString(matlab,"pd_call('SetDesPosition',r);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_consigna);
}

JNIEXPORT void JNICALL Java_FBK33200local_setDivisorRTK
  (JNIEnv *env, jobject obj, jint divisor)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    double valorsReals[]={(double)divisor};
    mxArray *mxArray_div=NULL;
    double  *partReal;
    mxArray_div=mxCreateDoubleMatrix(1,1,mxREAL);
    mxSetName(mxArray_div,"div");
    partReal = (double *)mxGetPr(mxArray_div);
    memcpy(partReal, valorsReals, 1 * sizeof(double) );
    engPutArray(matlab,mxArray_div);
    engEvalString(matlab,"pd_call('SetDivider',div);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
```

```
    }

    mxDestroyArray(mxArray_div);
}

JNIEXPORT void JNICALL Java_FBK33200local_setParametresControl
  (JNIEnv *env, jobject obj, jfloatArray parametres)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    double valorsReals[20];
    int numParametres;
    jfloat *parametres_Jfloat;

    numParametres=(*env).GetArrayLength(parametres);
    parametres_Jfloat=(*env).GetFloatArrayElements(parametres,0);

    if(numParametres > 20)
    {
        char missatgeError[80]="Mes de 20 parametres de la funcio
setParametresControl";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        return;
    }

    int i;
    for(i=0; i<numParametres; i++)
    {
        valorsReals[i]=(double)parametres_Jfloat[i];
    }
    for(i=numParametres; i<20; i++)
    {
        valorsReals[i]=0;
    }

    mxArray *mxArray_parametres=NULL;
    double  *partReal;
    mxArray_parametres=mxCreateDoubleMatrix(1,20,mxREAL);
    mxSetName(mxArray_parametres,"parametres");
    partReal = (double *)mxGetPr(mxArray_parametres);
    memcpy(partReal, valorsReals, 1*20*sizeof(double) );
    engPutArray(matlab,mxArray_parametres);
    engEvalString(matlab,"pd_call('SetP',parametres);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
```

```
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_parametres);
}

JNIEXPORT void JNICALL Java_FBK33200local_setParFiltrePosPendul
  (JNIEnv *env, jobject obj, jfloatArray parametres)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    double valorsReals[18];
    int numParametres;
    jfloat *parametres_Jfloat;

    numParametres=(*env).GetArrayLength(parametres);
    if(numParametres!=18)
    {
        char missatgeError[80]="Numero de parametres de la funcio
setParFiltrePosPendul incorrecte";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        return;
    }

    parametres_Jfloat=(*env).GetFloatArrayElements(parametres,0);

    int i;
    for(i=0; i<18; i++)
    {
        if(i==1 && parametres_Jfloat[i]!=0)
        {
                char missatgeError[80]="Filtre digital de Posicio del
Pendul NO CAUSAL!";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
                return;
        }
        valorsReals[i]=(double)parametres_Jfloat[i];
    }

    mxArray *mxArray_parametres=NULL;
```

```
    double  *partReal;
    mxArray_parametres=mxCreateDoubleMatrix(2,9,mxREAL);
    mxSetName(mxArray_parametres,"parametres");
    partReal = (double *)mxGetPr(mxArray_parametres);
    memcpy(partReal, valorsReals, 2*9*sizeof(double) );
    engPutArray(matlab,mxArray_parametres);
    engEvalString(matlab,"pd_call('SetPendPosFilt',parametres);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_parametres);
}

JNIEXPORT void JNICALL Java_FBK33200local_setParFiltreVelPendul
  (JNIEnv *env, jobject obj, jfloatArray parametres)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    double valorsReals[18];
    int numParametres;
    jfloat *parametres_Jfloat;

    numParametres=(*env).GetArrayLength(parametres);
    if(numParametres!=18)
    {
        char missatgeError[80]="Numero de parametres de la funcio
setParFiltreVelPendul incorrecte";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        return;
    }

    parametres_Jfloat=(*env).GetFloatArrayElements(parametres,0);

    int i;
    for(i=0; i<18; i++)
    {
        if(i==1 && parametres_Jfloat[i]!=0)
        {
```

```
                char missatgeError[80]="Filtre digital de Velocitat del
Pendul NO CAUSAL!";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
                return;
            }
            valorsReals[i]=(double)parametres_Jfloat[i];
        }

        mxArray *mxArray_parametres=NULL;
        double  *partReal;
        mxArray_parametres=mxCreateDoubleMatrix(2,9,mxREAL);
        mxSetName(mxArray_parametres,"parametres");
        partReal = (double *)mxGetPr(mxArray_parametres);
        memcpy(partReal, valorsReals, 2*9*sizeof(double) );
        engPutArray(matlab,mxArray_parametres);
        engEvalString(matlab,"pd_call('SetPendSpeedFilt',parametres);");

        mxArray *mxArray_ans;
        mxArray_ans=engGetArray(matlab,"ans");
        if(mxGetScalar(mxArray_ans)!=0)
        {
            char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
            jclass ExcepcioFBK33200;
            ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

            (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        }

        mxDestroyArray(mxArray_parametres);
    }

    JNIEXPORT void JNICALL Java_FBK33200local_setPW
      (JNIEnv *env, jobject obj, jfloatArray parametres)
    {
        if(matlabCarregat==false)
        {
            char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
            jclass ExcepcioFBK33200;
            ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

            (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
            return;
        }

        double valorsReals[20];
        int numParametres;
        jfloat *parametres_Jfloat;

        numParametres=(*env).GetArrayLength(parametres);
        if(numParametres==0)
        {
            char missatgeError[80]="Numero de parametres de la funcio setPW
incorrecte";
            jclass ExcepcioFBK33200;
            ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");
```

```
        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

        return;
    }

    parametres_Jfloat=(*env).GetFloatArrayElements(parametres,0);
    int k;

    switch((int)parametres_Jfloat[0])
    {
        case 0:
        if(numParametres!=2)
        {
                char missatgeError[80]="Numero de parametres de la
funcio setPW incorrecte";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        if(parametres_Jfloat[1]<-1 || parametres_Jfloat[1]>1)
        {
                char missatgeError[80]="Valor amplitud senyal
d'excitacio interna fora de limits";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        k=2;
        break;

        case 1:
        if(numParametres!=9)
        {
                char missatgeError[80]="Numero de parametres de la
funcio setPW incorrecte";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        if(parametres_Jfloat[5]<-1 || parametres_Jfloat[5]>1
                || parametres_Jfloat[6]<-1 || parametres_Jfloat[6]>1
                || parametres_Jfloat[7]<-1 || parametres_Jfloat[7]>1
                || parametres_Jfloat[8]<-1 || parametres_Jfloat[8]>1)
        {
                char missatgeError[80]="Valor amplitud senyal
d'excitacio interna fora de limits";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
```

```
        }
        k=9;
        break;

        case 2:
        if(numParametres!=5)
        {
                char missatgeError[80]="Numero de parametres de la
funcio setPW incorrecte";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        if(parametres_Jfloat[3]<-1 || parametres_Jfloat[3]>1
                || parametres_Jfloat[4]<-1 || parametres_Jfloat[4]>1)
        {
                char missatgeError[80]="Valor amplitud senyal
d'excitacio interna fora de limits";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        k=5;
        break;

        case 3:
        if(numParametres!=4)
        {
                char missatgeError[80]="Numero de parametres de la
funcio setPW incorrecte";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        if(parametres_Jfloat[2]<-1 || parametres_Jfloat[2]>1
                || parametres_Jfloat[3]<-1 || parametres_Jfloat[3]>1)
        {
                char missatgeError[80]="Valor amplitud senyal
d'excitacio interna fora de limits";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        k=4;
        break;

        case 4:
        if(numParametres!=4)
        {
```

```
                char missatgeError[80]="Numero de parametres de la
funcio setPW incorrecte";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        if(parametres_Jfloat[2]<-1 || parametres_Jfloat[2]>1
                || parametres_Jfloat[3]<-1 || parametres_Jfloat[3]>1)
        {
                char missatgeError[80]="Valor amplitud senyal
d'excitacio interna fora de limits";
                jclass ExcepcioFBK33200;
                ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

                (*env).ThrowNew(ExcepcioFBK33200,missatgeError);

                return;
        }
        k=4;
        break;

        default:
        char missatgeError[80]="Primer parametres de la funcio setPW no
esta en l'interval 0:4";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");
        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;

    }

    int i;
    for(i=0; i<k; i++)
    {
        valorsReals[i]=(double)parametres_Jfloat[i];
    }
    for(i=k; i<20; i++)
    {
        valorsReals[i]=0;
    }

    mxArray *mxArray_parametres=NULL;
    double  *partReal;
    mxArray_parametres=mxCreateDoubleMatrix(1,20,mxREAL);
    mxSetName(mxArray_parametres,"parametres");
    partReal = (double *)mxGetPr(mxArray_parametres);
    memcpy(partReal, valorsReals, 1*20*sizeof(double) );
    engPutArray(matlab,mxArray_parametres);
    engEvalString(matlab,"pd_call('SetPW',parametres);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");
```

```
        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_parametres);
}

JNIEXPORT void JNICALL Java_FBK33200local_setPeriodeMostreig
  (JNIEnv *env, jobject obj, jfloat ts)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    double valorsReals[]={(double)ts};
    mxArray *mxArray_periodeMostreig=NULL;
    double  *partReal;
    mxArray_periodeMostreig=mxCreateDoubleMatrix(1,1,mxREAL);
    mxSetName(mxArray_periodeMostreig,"ts");
    partReal = (double *)mxGetPr(mxArray_periodeMostreig);
    memcpy(partReal, valorsReals, 1 * sizeof(double) );
    engPutArray(matlab,mxArray_periodeMostreig);
    engEvalString(matlab,"pd_call('SetSampleTime',ts);");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }

    mxDestroyArray(mxArray_periodeMostreig);
}

JNIEXPORT void JNICALL Java_FBK33200local_iniciarAdquisicioDades
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    engEvalString(matlab,"pd_call('StartAcq');");
```

```
}

JNIEXPORT void JNICALL Java_FBK33200local_paraCarro
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return;
    }

    engEvalString(matlab,"pd_call('StopPractical');");

    mxArray *mxArray_ans;
    mxArray_ans=engGetArray(matlab,"ans");
    if(mxGetScalar(mxArray_ans)!=0)
    {
        char missatgeError[80]="Error desconegut amb la llibreria
PD_CALL.DLL";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
    }
}

JNIEXPORT jint JNICALL Java_FBK33200local_descarregaLlibreria
  (JNIEnv *env, jobject obj)
{
    if(matlabCarregat==false)
    {
        char missatgeError[80]="S'ha cridat una funcio de Matlab sense
haver-se carregat previament";
        jclass ExcepcioFBK33200;
        ExcepcioFBK33200=(*env).FindClass("ExcepcioFBK33200");

        (*env).ThrowNew(ExcepcioFBK33200,missatgeError);
        return 999;
    }

    jint resultat;
    mxArray *mxArray_contCarregues=NULL;
    engEvalString(matlab,"cont=pd_call('UnloadLibrary');");
    mxArray_contCarregues=engGetArray(matlab,"cont");
    resultat=(jint)mxGetScalar(mxArray_contCarregues);
    return resultat;
}
```

## 13.29. ARXIU *AplicacioFBK33200local.java*

```
import java.io.*;

public class AplicacioFBK33200local
{
   public static void main (String args[])throws Exception
   {
         FBK33200local instFBK33200=new FBK33200local();

         System.out.println("S'ha carregat el MATLAB");

         float[] parPID=new float[7];
         parPID[0]=(float)0.1;
         parPID[1]=(float)-0.294;
         parPID[2]=(float)0;
         parPID[3]=(float)-0.0042;
         parPID[4]=(float)0.5;
         parPID[5]=(float)1;
         parPID[6]=(float)0.5;

         instFBK33200.setParametresControl(parPID);
         instFBK33200.setConsigna((float)0);
         instFBK33200.setPeriodeMostreig((float)0.01);
         instFBK33200.setTipusControl(FBK33200local.PID);


         InputStreamReader teclat=new InputStreamReader(System.in);
         BufferedReader liniaTeclat = new BufferedReader(teclat);
         System.out.println("Escriu la posicio desitjada de -0.5 a +0.5");
         System.out.println("Tecleja SORTIR per acabar el programa");
         System.out.println
                     ("Tecleja ACTUAL per obtenir la posicio del carro");

         String linia;
         float novaConsigna=0;
         float posicioActual=0;
         float[] historialPosicions;
         int mostresRebudes=0;

         while(true)
         {
               linia=liniaTeclat.readLine();

               if(linia.equalsIgnoreCase("sortir")) break;

               if(linia.equalsIgnoreCase("actual"))
               {
                     do{
                        mostresRebudes=instFBK33200.getNumMostresBufer();
                     }while(mostresRebudes==0);


                     historialPosicions=instFBK33200.getHistorialPosCarro();
                     posicioActual=historialPosicions[mostresRebudes-1];
                     instFBK33200.tempsAzero();

                     System.out.println
                                 ("POSICIO ACTUAL = "+posicioActual+" ");
```

```
                        continue;
                }

                try{

                        novaConsigna=Float.parseFloat(linia);

                        if(novaConsigna>0.5 || novaConsigna<-0.5)
                        {
                                System.out.println
                                        ("posicio especificada fora de rang!!");
                                System.out.println
                                ("Escriu la posicio desitjada de -0.5 a +0.5");
                        }
                        else
                        {
                                instFBK33200.setConsigna(novaConsigna);
                        }

                }catch(NumberFormatException excepcioNumerica){

                        System.out.println("Format incorrecte!!!");
                        System.out.println
                                ("Escriu la posicio desitjada de -0.5 a +0.5");
                        System.out.println
                                ("Tecleja SORTIR per acabar el programa");
                        System.out.println
                           ("Tecleja ACTUAL per obtenir la posicio del carro");

                }

        }

        instFBK33200.paraCarro();
        instFBK33200.finalize();

        System.out.println("S'ha acabat!!");
   }
}
```

## 13.30. ARXIU *FBK33200remot.idl*

```
interface FBK33200remot
{
   typedef sequence<float> Vector_Float;

   exception ExcepcioFBK33200remota{};

   //constants enteres:

   const long CAP=0;
   const long LLAC_OBERT=1;
   const long LQ_INVERTIT=2;
   const long LQ_GRUA=3;
   const long FUZZY=4;
   const long PID=5;
   const long CONTROL_OPTIM=6;
   const long CONTROL_ADAPTATIU=7;
   const long EXTERN=99;

   const long CONSTANT=0;
   const long SENYAL_QUADRADA=1;
   const long SENYAL_TRIANGULAR=2;
   const long SENYAL_SINUSOIDAL=3;
   const long SENYAL_PERIODIC_ALEATORI=4;


   //els metodes per ser invocats remotament:

   void assignarReferenciaMatlab()raises (ExcepcioFBK33200remota);
   void alliberarReferenciaMatlab();

   long getTipusControl()raises (ExcepcioFBK33200remota);
// long getAdrPCL812PG()raises (ExcepcioFBK33200remota);
   Vector_Float getParFiltrePosCarro()raises (ExcepcioFBK33200remota);
   Vector_Float getParFiltreVelCarro()raises (ExcepcioFBK33200remota);

   float getConsigna()raises (ExcepcioFBK33200remota);
   long getDivisorRTK()raises (ExcepcioFBK33200remota);

   Vector_Float getHistorialTemps()raises (ExcepcioFBK33200remota);
   Vector_Float getHistorialAnglePendul()raises (ExcepcioFBK33200remota);
   Vector_Float getHistorialVelPendul()raises (ExcepcioFBK33200remota);
   Vector_Float getHistorialPosCarro()raises (ExcepcioFBK33200remota);
   Vector_Float getHistorialVelCarro()raises (ExcepcioFBK33200remota);
   Vector_Float getHistorialAccioControlRelativa()
                                       raises (ExcepcioFBK33200remota);
   Vector_Float getHistorialConsigna()raises (ExcepcioFBK33200remota);

   long getNumMostresBufer()raises (ExcepcioFBK33200remota);
   Vector_Float getParametresControl()raises (ExcepcioFBK33200remota);

   Vector_Float getParFiltrePosPendul()raises (ExcepcioFBK33200remota);
   Vector_Float getParFiltreVelPendul()raises (ExcepcioFBK33200remota);

   Vector_Float getPW()raises (ExcepcioFBK33200remota);
   float getPeriodeMostreig()raises (ExcepcioFBK33200remota);
   long carregaLlibreria()raises (ExcepcioFBK33200remota);
   void carregaControladorExtern(in string fitxer)
                                       raises (ExcepcioFBK33200remota);
```

```
// void comptadorsAzero()raises (ExcepcioFBK33200remota);
   void tempsAzero()raises (ExcepcioFBK33200remota);
   void setTipusControl(in long tipus)raises (ExcepcioFBK33200remota);
// void setAdrPCL812PG(in long adreca)raises (ExcepcioFBK33200remota);

   void setParFiltrePosCarro(in Vector_Float parametres)
                                        raises (ExcepcioFBK33200remota);
   void setParFiltreVelCarro(in Vector_Float parametres)
                                        raises (ExcepcioFBK33200remota);

   void setConsigna(in float r)raises (ExcepcioFBK33200remota);
   void setDivisorRTK(in long divisor)raises (ExcepcioFBK33200remota);
   void setParametresControl(in Vector_Float parametres)
                                        raises (ExcepcioFBK33200remota);

   void setParFiltrePosPendul(in Vector_Float parametres)
                                        raises (ExcepcioFBK33200remota);
   void setParFiltreVelPendul(in Vector_Float parametres)
                                        raises (ExcepcioFBK33200remota);

   void setPW(in Vector_Float parametres)raises (ExcepcioFBK33200remota);
   void setPeriodeMostreig(in float ts)raises (ExcepcioFBK33200remota);
   void iniciarAdquisicioDades()raises (ExcepcioFBK33200remota);
   void paraCarro()raises (ExcepcioFBK33200remota);
   long descarregaLlibreria()raises (ExcepcioFBK33200remota);
};
```

## 13.31. ARXIU *FBK33200remot.java*

```
/**
* FBK33200remot.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
*/

public interface FBK33200remot extends FBK33200remotOperations,
org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity
{

  //constants enteres:
  public static final int CAP = (int)(0);
  public static final int LLAC_OBERT = (int)(1);
  public static final int LQ_INVERTIT = (int)(2);
  public static final int LQ_GRUA = (int)(3);
  public static final int FUZZY = (int)(4);
  public static final int PID = (int)(5);
  public static final int CONTROL_OPTIM = (int)(6);
  public static final int CONTROL_ADAPTATIU = (int)(7);
  public static final int EXTERN = (int)(99);
  public static final int CONSTANT = (int)(0);
  public static final int SENYAL_QUADRADA = (int)(1);
  public static final int SENYAL_TRIANGULAR = (int)(2);
  public static final int SENYAL_SINUSOIDAL = (int)(3);
  public static final int SENYAL_PERIODIC_ALEATORI = (int)(4);
} // interface FBK33200remot
```

## 13.32. ARXIU *FBK33200remotOperations.java*

```
/**
* FBK33200remotOperations.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
*/

public interface FBK33200remotOperations
{

  //els metodes per ser invocats remotament:
  void assignarReferenciaMatlab ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void alliberarReferenciaMatlab ();
  int getTipusControl ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;

  //     long getAdrPCL812PG()raises (ExcepcioFBK33200remota);
  float[] getParFiltrePosCarro ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getParFiltreVelCarro ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float getConsigna () throws FBK33200remotPackage.ExcepcioFBK33200remota;
  int getDivisorRTK () throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getHistorialTemps ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getHistorialAnglePendul ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getHistorialVelPendul ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getHistorialPosCarro ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getHistorialVelCarro ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getHistorialAccioControlRelativa ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getHistorialConsigna ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  int getNumMostresBufer ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getParametresControl ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getParFiltrePosPendul ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getParFiltreVelPendul ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float[] getPW () throws FBK33200remotPackage.ExcepcioFBK33200remota;
  float getPeriodeMostreig ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  int carregaLlibreria ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void carregaControladorExtern (String fitxer)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;

  //     void comptadorsAzero()raises (ExcepcioFBK33200remota);
  void tempsAzero () throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setTipusControl (int tipus)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
```

```
  //     void setAdrPCL812PG(in long adreca)raises (ExcepcioFBK33200remota);
  void setParFiltrePosCarro (float[] parametres)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setParFiltreVelCarro (float[] parametres)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setConsigna (float r)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setDivisorRTK (int divisor)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setParametresControl (float[] parametres)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setParFiltrePosPendul (float[] parametres)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setParFiltreVelPendul (float[] parametres)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setPW (float[] parametres)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void setPeriodeMostreig (float ts)
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void iniciarAdquisicioDades ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
  void paraCarro () throws FBK33200remotPackage.ExcepcioFBK33200remota;
  int descarregaLlibreria ()
                    throws FBK33200remotPackage.ExcepcioFBK33200remota;
} // interface FBK33200remotOperations
```

## 13.33. ARXIU *_FBK33200remotImplBase.java*

```
/**
* _FBK33200remotImplBase.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
* miércoles 2 de junio de 2004 16H46' CEST
*/

public abstract class _FBK33200remotImplBase extends
org.omg.CORBA.portable.ObjectImpl implements FBK33200remot,
org.omg.CORBA.portable.InvokeHandler
{

  // Constructors
  public _FBK33200remotImplBase ()
  {
  }

  private static java.util.Hashtable _methods = new java.util.Hashtable ();
  static
  {
    _methods.put ("assignarReferenciaMatlab", new java.lang.Integer (0));
    _methods.put ("alliberarReferenciaMatlab", new java.lang.Integer (1));
    _methods.put ("getTipusControl", new java.lang.Integer (2));
    _methods.put ("getParFiltrePosCarro", new java.lang.Integer (3));
    _methods.put ("getParFiltreVelCarro", new java.lang.Integer (4));
    _methods.put ("getConsigna", new java.lang.Integer (5));
    _methods.put ("getDivisorRTK", new java.lang.Integer (6));
    _methods.put ("getHistorialTemps", new java.lang.Integer (7));
    _methods.put ("getHistorialAnglePendul", new java.lang.Integer (8));
    _methods.put ("getHistorialVelPendul", new java.lang.Integer (9));
    _methods.put ("getHistorialPosCarro", new java.lang.Integer (10));
    _methods.put ("getHistorialVelCarro", new java.lang.Integer (11));
    _methods.put
         ("getHistorialAccioControlRelativa", new java.lang.Integer (12));
    _methods.put ("getHistorialConsigna", new java.lang.Integer (13));
    _methods.put ("getNumMostresBufer", new java.lang.Integer (14));
    _methods.put ("getParametresControl", new java.lang.Integer (15));
    _methods.put ("getParFiltrePosPendul", new java.lang.Integer (16));
    _methods.put ("getParFiltreVelPendul", new java.lang.Integer (17));
    _methods.put ("getPW", new java.lang.Integer (18));
    _methods.put ("getPeriodeMostreig", new java.lang.Integer (19));
    _methods.put ("carregaLlibreria", new java.lang.Integer (20));
    _methods.put ("carregaControladorExtern", new java.lang.Integer (21));
    _methods.put ("tempsAzero", new java.lang.Integer (22));
    _methods.put ("setTipusControl", new java.lang.Integer (23));
    _methods.put ("setParFiltrePosCarro", new java.lang.Integer (24));
    _methods.put ("setParFiltreVelCarro", new java.lang.Integer (25));
    _methods.put ("setConsigna", new java.lang.Integer (26));
    _methods.put ("setDivisorRTK", new java.lang.Integer (27));
    _methods.put ("setParametresControl", new java.lang.Integer (28));
    _methods.put ("setParFiltrePosPendul", new java.lang.Integer (29));
    _methods.put ("setParFiltreVelPendul", new java.lang.Integer (30));
    _methods.put ("setPW", new java.lang.Integer (31));
    _methods.put ("setPeriodeMostreig", new java.lang.Integer (32));
    _methods.put ("iniciarAdquisicioDades", new java.lang.Integer (33));
    _methods.put ("paraCarro", new java.lang.Integer (34));
    _methods.put ("descarregaLlibreria", new java.lang.Integer (35));
  }
```

```
  public org.omg.CORBA.portable.OutputStream _invoke (String method,
                                org.omg.CORBA.portable.InputStream in,
                                org.omg.CORBA.portable.ResponseHandler rh)
  {
    org.omg.CORBA.portable.OutputStream out = null;
    java.lang.Integer __method = (java.lang.Integer)_methods.get
(method);
    if (__method == null)
      throw new org.omg.CORBA.BAD_OPERATION (0,
org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);

    switch (__method.intValue ())
    {

  //els metodes per ser invocats remotament:
       case 0:  // FBK33200remot/assignarReferenciaMatlab
       {
         try {
           this.assignarReferenciaMatlab ();
           out = rh.createReply();
         } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
           out = rh.createExceptionReply ();
           FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
         }
         break;
       }

       case 1:  // FBK33200remot/alliberarReferenciaMatlab
       {
         this.alliberarReferenciaMatlab ();
         out = rh.createReply();
         break;
       }

       case 2:  // FBK33200remot/getTipusControl
       {
         try {
           int __result = (int)0;
           __result = this.getTipusControl ();
           out = rh.createReply();
           out.write_long (__result);
         } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
           out = rh.createExceptionReply ();
           FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
         }
         break;
       }


  //     long getAdrPCL812PG()raises (ExcepcioFBK33200remota);
       case 3:  // FBK33200remot/getParFiltrePosCarro
       {
         try {
           float __result[] = null;
           __result = this.getParFiltrePosCarro ();
           out = rh.createReply();
           FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
         } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
```

```
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 4:  // FBK33200remot/getParFiltreVelCarro
        {
          try {
            float __result[] = null;
            __result = this.getParFiltreVelCarro ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 5:  // FBK33200remot/getConsigna
        {
          try {
            float __result = (float)0;
            __result = this.getConsigna ();
            out = rh.createReply();
            out.write_float (__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 6:  // FBK33200remot/getDivisorRTK
        {
          try {
            int __result = (int)0;
            __result = this.getDivisorRTK ();
            out = rh.createReply();
            out.write_long (__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 7:  // FBK33200remot/getHistorialTemps
        {
          try {
            float __result[] = null;
            __result = this.getHistorialTemps ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
```

```
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 8:  // FBK33200remot/getHistorialAnglePendul
        {
          try {
            float __result[] = null;
            __result = this.getHistorialAnglePendul ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 9:  // FBK33200remot/getHistorialVelPendul
        {
          try {
            float __result[] = null;
            __result = this.getHistorialVelPendul ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 10:  // FBK33200remot/getHistorialPosCarro
        {
          try {
            float __result[] = null;
            __result = this.getHistorialPosCarro ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 11:  // FBK33200remot/getHistorialVelCarro
        {
          try {
            float __result[] = null;
            __result = this.getHistorialVelCarro ();
```

```
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 12:  // FBK33200remot/getHistorialAccioControlRelativa
        {
          try {
            float __result[] = null;
            __result = this.getHistorialAccioControlRelativa ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 13:  // FBK33200remot/getHistorialConsigna
        {
          try {
            float __result[] = null;
            __result = this.getHistorialConsigna ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 14:  // FBK33200remot/getNumMostresBufer
        {
          try {
            int __result = (int)0;
            __result = this.getNumMostresBufer ();
            out = rh.createReply();
            out.write_long (__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 15:  // FBK33200remot/getParametresControl
        {
          try {
```

```
            float __result[] = null;
            __result = this.getParametresControl ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 16:  // FBK33200remot/getParFiltrePosPendul
        {
          try {
            float __result[] = null;
            __result = this.getParFiltrePosPendul ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 17:  // FBK33200remot/getParFiltreVelPendul
        {
          try {
            float __result[] = null;
            __result = this.getParFiltreVelPendul ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 18:  // FBK33200remot/getPW
        {
          try {
            float __result[] = null;
            __result = this.getPW ();
            out = rh.createReply();
            FBK33200remotPackage.Vector_FloatHelper.write (out,
__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }
```

```
        case 19:  // FBK33200remot/getPeriodeMostreig
        {
          try {
            float __result = (float)0;
            __result = this.getPeriodeMostreig ();
            out = rh.createReply();
            out.write_float (__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 20:  // FBK33200remot/carregaLlibreria
        {
          try {
            int __result = (int)0;
            __result = this.carregaLlibreria ();
            out = rh.createReply();
            out.write_long (__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 21:  // FBK33200remot/carregaControladorExtern
        {
          try {
            String fitxer = in.read_string ();
            this.carregaControladorExtern (fitxer);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }


  //     void comptadorsAzero()raises (ExcepcioFBK33200remota);
        case 22:  // FBK33200remot/tempsAzero
        {
          try {
            this.tempsAzero ();
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 23:  // FBK33200remot/setTipusControl
        {
```

```
          try {
            int tipus = in.read_long ();
            this.setTipusControl (tipus);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }


  //     void setAdrPCL812PG(in long adreca)raises
(ExcepcioFBK33200remota);
        case 24:  // FBK33200remot/setParFiltrePosCarro
        {
          try {
            float parametres[] =
FBK33200remotPackage.Vector_FloatHelper.read (in);
            this.setParFiltrePosCarro (parametres);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 25:  // FBK33200remot/setParFiltreVelCarro
        {
          try {
            float parametres[] =
FBK33200remotPackage.Vector_FloatHelper.read (in);
            this.setParFiltreVelCarro (parametres);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 26:  // FBK33200remot/setConsigna
        {
          try {
            float r = in.read_float ();
            this.setConsigna (r);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 27:  // FBK33200remot/setDivisorRTK
        {
```

```
          try {
            int divisor = in.read_long ();
            this.setDivisorRTK (divisor);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 28:  // FBK33200remot/setParametresControl
        {
          try {
            float parametres[] =
FBK33200remotPackage.Vector_FloatHelper.read (in);
            this.setParametresControl (parametres);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 29:  // FBK33200remot/setParFiltrePosPendul
        {
          try {
            float parametres[] =
FBK33200remotPackage.Vector_FloatHelper.read (in);
            this.setParFiltrePosPendul (parametres);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 30:  // FBK33200remot/setParFiltreVelPendul
        {
          try {
            float parametres[] =
FBK33200remotPackage.Vector_FloatHelper.read (in);
            this.setParFiltreVelPendul (parametres);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 31:  // FBK33200remot/setPW
        {
          try {
```

```
            float parametres[] =
FBK33200remotPackage.Vector_FloatHelper.read (in);
            this.setPW (parametres);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 32:  // FBK33200remot/setPeriodeMostreig
        {
          try {
            float ts = in.read_float ();
            this.setPeriodeMostreig (ts);
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 33:  // FBK33200remot/iniciarAdquisicioDades
        {
          try {
            this.iniciarAdquisicioDades ();
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 34:  // FBK33200remot/paraCarro
        {
          try {
            this.paraCarro ();
            out = rh.createReply();
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        case 35:  // FBK33200remot/descarregaLlibreria
        {
          try {
            int __result = (int)0;
            __result = this.descarregaLlibreria ();
            out = rh.createReply();
            out.write_long (__result);
          } catch (FBK33200remotPackage.ExcepcioFBK33200remota __ex) {
            out = rh.createExceptionReply ();
```

```
            FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (out,
__ex);
          }
          break;
        }

        default:
          throw new org.omg.CORBA.BAD_OPERATION (0,
org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);
      }

      return out;
    } // _invoke

    // Type-specific CORBA::Object operations
    private static String[] __ids = {
      "IDL:FBK33200remot:1.0"};

    public String[] _ids ()
    {
      return __ids;
    }


} // class _FBK33200remotImplBase
```

## 13.34. ARXIU *_FBK33200remotStub.java*

```
/**
* _FBK33200remotStub.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
*/

public class _FBK33200remotStub extends
org.omg.CORBA.portable.ObjectImpl implements FBK33200remot
{
  // Constructors
  // NOTE:  If the default constructor is used, the
  //        object is useless until _set_delegate (...)
  //        is called.
  public _FBK33200remotStub ()
  {
    super ();
  }

  public _FBK33200remotStub (org.omg.CORBA.portable.Delegate delegate)
  {
    super ();
    _set_delegate (delegate);
  }


  //els metodes per ser invocats remotament:
  public void assignarReferenciaMatlab () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("assignarReferenciaMatlab", true);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       assignarReferenciaMatlab ();
    } finally {
        _releaseReply (_in);
    }
  } // assignarReferenciaMatlab

  public void alliberarReferenciaMatlab ()
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("alliberarReferenciaMatlab", true);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
```

```
        _in = _ex.getInputStream ();
        String _id = _ex.getId ();
        throw new org.omg.CORBA.MARSHAL (_id);
      } catch (org.omg.CORBA.portable.RemarshalException _rm) {
        alliberarReferenciaMatlab ();
      } finally {
         _releaseReply (_in);
      }
    } // alliberarReferenciaMatlab

    public int getTipusControl () throws
FBK33200remotPackage.ExcepcioFBK33200remota
    {
      org.omg.CORBA.portable.InputStream _in = null;
      try {
        org.omg.CORBA.portable.OutputStream _out = _request
("getTipusControl", true);
        _in = _invoke (_out);
        int __result = _in.read_long ();
        return __result;
      } catch (org.omg.CORBA.portable.ApplicationException _ex) {
        _in = _ex.getInputStream ();
        String _id = _ex.getId ();
        if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
        else
             throw new org.omg.CORBA.MARSHAL (_id);
      } catch (org.omg.CORBA.portable.RemarshalException _rm) {
        return getTipusControl ();
      } finally {
         _releaseReply (_in);
      }
    } // getTipusControl


    //     long getAdrPCL812PG()raises (ExcepcioFBK33200remota);
    public float[] getParFiltrePosCarro () throws
FBK33200remotPackage.ExcepcioFBK33200remota
    {
      org.omg.CORBA.portable.InputStream _in = null;
      try {
        org.omg.CORBA.portable.OutputStream _out = _request
("getParFiltrePosCarro", true);
        _in = _invoke (_out);
        float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
        return __result;
      } catch (org.omg.CORBA.portable.ApplicationException _ex) {
        _in = _ex.getInputStream ();
        String _id = _ex.getId ();
        if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
        else
             throw new org.omg.CORBA.MARSHAL (_id);
      } catch (org.omg.CORBA.portable.RemarshalException _rm) {
        return getParFiltrePosCarro ();
      } finally {
         _releaseReply (_in);
      }
```

```
  } // getParFiltrePosCarro

  public float[] getParFiltreVelCarro () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getParFiltreVelCarro", true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getParFiltreVelCarro ();
    } finally {
        _releaseReply (_in);
    }
  } // getParFiltreVelCarro

  public float getConsigna () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getConsigna", true);
       _in = _invoke (_out);
       float __result = _in.read_float ();
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getConsigna ();
    } finally {
        _releaseReply (_in);
    }
  } // getConsigna

  public int getDivisorRTK () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getDivisorRTK", true);
       _in = _invoke (_out);
```

```
        int __result = _in.read_long ();
        return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
        _in = _ex.getInputStream ();
        String _id = _ex.getId ();
        if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
           throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
        else
             throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
        return getDivisorRTK ();
    } finally {
         _releaseReply (_in);
    }
  } // getDivisorRTK

  public float[] getHistorialTemps () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getHistorialTemps", true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getHistorialTemps ();
    } finally {
         _releaseReply (_in);
    }
  } // getHistorialTemps

  public float[] getHistorialAnglePendul () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getHistorialAnglePendul", true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
```

```
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getHistorialAnglePendul ();
    } finally {
        _releaseReply (_in);
    }
  } // getHistorialAnglePendul

  public float[] getHistorialVelPendul () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getHistorialVelPendul", true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getHistorialVelPendul ();
    } finally {
        _releaseReply (_in);
    }
  } // getHistorialVelPendul

  public float[] getHistorialPosCarro () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getHistorialPosCarro", true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getHistorialPosCarro ();
    } finally {
        _releaseReply (_in);
    }
  } // getHistorialPosCarro
```

```
  public float[] getHistorialVelCarro () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getHistorialVelCarro", true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getHistorialVelCarro ();
    } finally {
        _releaseReply (_in);
    }
  } // getHistorialVelCarro

  public float[] getHistorialAccioControlRelativa () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getHistorialAccioControlRelativa", true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getHistorialAccioControlRelativa ();
    } finally {
        _releaseReply (_in);
    }
  } // getHistorialAccioControlRelativa

  public float[] getHistorialConsigna () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getHistorialConsigna", true);
       _in = _invoke (_out);
```

```
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getHistorialConsigna ();
    } finally {
        _releaseReply (_in);
    }
  } // getHistorialConsigna

  public int getNumMostresBufer () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getNumMostresBufer", true);
       _in = _invoke (_out);
       int __result = _in.read_long ();
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getNumMostresBufer ();
    } finally {
        _releaseReply (_in);
    }
  } // getNumMostresBufer

  public float[] getParametresControl () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getParametresControl", true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
```

```
            throw new org.omg.CORBA.MARSHAL (_id);
        } catch (org.omg.CORBA.portable.RemarshalException _rm) {
           return getParametresControl ();
        } finally {
            _releaseReply (_in);
        }
      } // getParametresControl

      public float[] getParFiltrePosPendul () throws
    FBK33200remotPackage.ExcepcioFBK33200remota
      {
        org.omg.CORBA.portable.InputStream _in = null;
        try {
           org.omg.CORBA.portable.OutputStream _out = _request
    ("getParFiltrePosPendul", true);
           _in = _invoke (_out);
           float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
    (_in);
           return __result;
        } catch (org.omg.CORBA.portable.ApplicationException _ex) {
           _in = _ex.getInputStream ();
           String _id = _ex.getId ();
           if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
              throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
    (_in);
           else
                throw new org.omg.CORBA.MARSHAL (_id);
        } catch (org.omg.CORBA.portable.RemarshalException _rm) {
           return getParFiltrePosPendul ();
        } finally {
            _releaseReply (_in);
        }
      } // getParFiltrePosPendul

      public float[] getParFiltreVelPendul () throws
    FBK33200remotPackage.ExcepcioFBK33200remota
      {
        org.omg.CORBA.portable.InputStream _in = null;
        try {
           org.omg.CORBA.portable.OutputStream _out = _request
    ("getParFiltreVelPendul", true);
           _in = _invoke (_out);
           float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
    (_in);
           return __result;
        } catch (org.omg.CORBA.portable.ApplicationException _ex) {
           _in = _ex.getInputStream ();
           String _id = _ex.getId ();
           if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
              throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
    (_in);
           else
                throw new org.omg.CORBA.MARSHAL (_id);
        } catch (org.omg.CORBA.portable.RemarshalException _rm) {
           return getParFiltreVelPendul ();
        } finally {
            _releaseReply (_in);
        }
      } // getParFiltreVelPendul
```

```
  public float[] getPW () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request ("getPW",
true);
       _in = _invoke (_out);
       float __result[] = FBK33200remotPackage.Vector_FloatHelper.read
(_in);
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getPW ();
    } finally {
        _releaseReply (_in);
    }
  } // getPW

  public float getPeriodeMostreig () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("getPeriodeMostreig", true);
       _in = _invoke (_out);
       float __result = _in.read_float ();
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return getPeriodeMostreig ();
    } finally {
        _releaseReply (_in);
    }
  } // getPeriodeMostreig

  public int carregaLlibreria () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("carregaLlibreria", true);
       _in = _invoke (_out);
       int __result = _in.read_long ();
       return __result;
```

```
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       return carregaLlibreria ();
    } finally {
        _releaseReply (_in);
    }
  } // carregaLlibreria

  public void carregaControladorExtern (String fitxer) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("carregaControladorExtern", true);
       _out.write_string (fitxer);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       carregaControladorExtern (fitxer);
    } finally {
        _releaseReply (_in);
    }
  } // carregaControladorExtern


  //     void comptadorsAzero()raises (ExcepcioFBK33200remota);
  public void tempsAzero () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("tempsAzero", true);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       tempsAzero ();
    } finally {
        _releaseReply (_in);
```

```
    }
  } // tempsAzero

  public void setTipusControl (int tipus) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("setTipusControl", true);
       _out.write_long (tipus);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       setTipusControl (tipus);
    } finally {
        _releaseReply (_in);
    }
  } // setTipusControl


  //     void setAdrPCL812PG(in long adreca)raises
(ExcepcioFBK33200remota);
  public void setParFiltrePosCarro (float[] parametres) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("setParFiltrePosCarro", true);
       FBK33200remotPackage.Vector_FloatHelper.write (_out, parametres);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       setParFiltrePosCarro (parametres);
    } finally {
        _releaseReply (_in);
    }
  } // setParFiltrePosCarro

  public void setParFiltreVelCarro (float[] parametres) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("setParFiltreVelCarro", true);
```

```
        FBK33200remotPackage.Vector_FloatHelper.write (_out, parametres);
        _in = _invoke (_out);
     } catch (org.omg.CORBA.portable.ApplicationException _ex) {
        _in = _ex.getInputStream ();
        String _id = _ex.getId ();
        if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
           throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
        else
             throw new org.omg.CORBA.MARSHAL (_id);
     } catch (org.omg.CORBA.portable.RemarshalException _rm) {
        setParFiltreVelCarro (parametres);
     } finally {
         _releaseReply (_in);
     }
   } // setParFiltreVelCarro

   public void setConsigna (float r) throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
     org.omg.CORBA.portable.InputStream _in = null;
     try {
        org.omg.CORBA.portable.OutputStream _out = _request
("setConsigna", true);
        _out.write_float (r);
        _in = _invoke (_out);
     } catch (org.omg.CORBA.portable.ApplicationException _ex) {
        _in = _ex.getInputStream ();
        String _id = _ex.getId ();
        if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
           throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
        else
             throw new org.omg.CORBA.MARSHAL (_id);
     } catch (org.omg.CORBA.portable.RemarshalException _rm) {
        setConsigna (r);
     } finally {
         _releaseReply (_in);
     }
   } // setConsigna

   public void setDivisorRTK (int divisor) throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
     org.omg.CORBA.portable.InputStream _in = null;
     try {
        org.omg.CORBA.portable.OutputStream _out = _request
("setDivisorRTK", true);
        _out.write_long (divisor);
        _in = _invoke (_out);
     } catch (org.omg.CORBA.portable.ApplicationException _ex) {
        _in = _ex.getInputStream ();
        String _id = _ex.getId ();
        if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
           throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
        else
             throw new org.omg.CORBA.MARSHAL (_id);
     } catch (org.omg.CORBA.portable.RemarshalException _rm) {
        setDivisorRTK (divisor);
     } finally {
```

```
        _releaseReply (_in);
    }
  } // setDivisorRTK

  public void setParametresControl (float[] parametres) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("setParametresControl", true);
       FBK33200remotPackage.Vector_FloatHelper.write (_out, parametres);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       setParametresControl (parametres);
    } finally {
        _releaseReply (_in);
    }
  } // setParametresControl

  public void setParFiltrePosPendul (float[] parametres) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("setParFiltrePosPendul", true);
       FBK33200remotPackage.Vector_FloatHelper.write (_out, parametres);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       setParFiltrePosPendul (parametres);
    } finally {
        _releaseReply (_in);
    }
  } // setParFiltrePosPendul

  public void setParFiltreVelPendul (float[] parametres) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("setParFiltreVelPendul", true);
       FBK33200remotPackage.Vector_FloatHelper.write (_out, parametres);
       _in = _invoke (_out);
```

```
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       setParFiltreVelPendul (parametres);
    } finally {
        _releaseReply (_in);
    }
  } // setParFiltreVelPendul

  public void setPW (float[] parametres) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request ("setPW",
true);
       FBK33200remotPackage.Vector_FloatHelper.write (_out, parametres);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       setPW (parametres);
    } finally {
        _releaseReply (_in);
    }
  } // setPW

  public void setPeriodeMostreig (float ts) throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("setPeriodeMostreig", true);
       _out.write_float (ts);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       setPeriodeMostreig (ts);
    } finally {
        _releaseReply (_in);
    }
```

```
  } // setPeriodeMostreig

  public void iniciarAdquisicioDades () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("iniciarAdquisicioDades", true);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       iniciarAdquisicioDades ();
    } finally {
        _releaseReply (_in);
    }
  } // iniciarAdquisicioDades

  public void paraCarro () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request ("paraCarro",
true);
       _in = _invoke (_out);
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
       if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
          throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
       else
            throw new org.omg.CORBA.MARSHAL (_id);
    } catch (org.omg.CORBA.portable.RemarshalException _rm) {
       paraCarro ();
    } finally {
        _releaseReply (_in);
    }
  } // paraCarro

  public int descarregaLlibreria () throws
FBK33200remotPackage.ExcepcioFBK33200remota
  {
    org.omg.CORBA.portable.InputStream _in = null;
    try {
       org.omg.CORBA.portable.OutputStream _out = _request
("descarregaLlibreria", true);
       _in = _invoke (_out);
       int __result = _in.read_long ();
       return __result;
    } catch (org.omg.CORBA.portable.ApplicationException _ex) {
       _in = _ex.getInputStream ();
       String _id = _ex.getId ();
```

```
        if (_id.equals ("IDL:FBK33200remot/ExcepcioFBK33200remota:1.0"))
           throw FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read
(_in);
        else
             throw new org.omg.CORBA.MARSHAL (_id);
      } catch (org.omg.CORBA.portable.RemarshalException _rm) {
        return descarregaLlibreria ();
      } finally {
         _releaseReply (_in);
      }
    } // descarregaLlibreria

    // Type-specific CORBA::Object operations
    private static String[] __ids = {
      "IDL:FBK33200remot:1.0"};

    public String[] _ids ()
    {
      return (String[])__ids.clone ();
    }

    private void readObject (java.io.ObjectInputStream s)
    {
       try
       {
         String str = s.readUTF ();
         org.omg.CORBA.Object obj = org.omg.CORBA.ORB.init
().string_to_object (str);
         org.omg.CORBA.portable.Delegate delegate =
((org.omg.CORBA.portable.ObjectImpl) obj)._get_delegate ();
         _set_delegate (delegate);
       } catch (java.io.IOException e) {}
    }

    private void writeObject (java.io.ObjectOutputStream s)
    {
       try
       {
         String str = org.omg.CORBA.ORB.init ().object_to_string (this);
         s.writeUTF (str);
       } catch (java.io.IOException e) {}
    }
  } // class _FBK33200remotStub
```

## 13.35. ARXIU *FBK33200remotHelper.java*

```
/** FBK33200remotHelper.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
*/

abstract public class FBK33200remotHelper
{
  private static String  _id = "IDL:FBK33200remot:1.0";

  public static void insert (org.omg.CORBA.Any a, FBK33200remot that)
  {
    org.omg.CORBA.portable.OutputStream out = a.create_output_stream ();
    a.type (type ());
    write (out, that);
    a.read_value (out.create_input_stream (), type ());
  }
  public static FBK33200remot extract (org.omg.CORBA.Any a)
  {
    return read (a.create_input_stream ());
  }
  private static org.omg.CORBA.TypeCode __typeCode = null;
  synchronized public static org.omg.CORBA.TypeCode type ()
  {
    if (__typeCode == null)
    {
      __typeCode = org.omg.CORBA.ORB.init ().create_interface_tc
(FBK33200remotHelper.id (), "FBK33200remot");
    }
    return __typeCode;
  }
  public static String id ()
  {
    return _id;
  }
  public static FBK33200remot read (org.omg.CORBA.portable.InputStream istream)
  {
    return narrow (istream.read_Object (_FBK33200remotStub.class));
  }
  public static void write (org.omg.CORBA.portable.OutputStream ostream,
FBK33200remot value)
  {
    ostream.write_Object ((org.omg.CORBA.Object) value);
  }
  public static FBK33200remot narrow (org.omg.CORBA.Object obj)
  {
    if (obj == null)
      return null;
    else if (obj instanceof FBK33200remot)
      return (FBK33200remot)obj;
    else if (!obj._is_a (id ()))
      throw new org.omg.CORBA.BAD_PARAM ();
    else
    {
      org.omg.CORBA.portable.Delegate delegate =
((org.omg.CORBA.portable.ObjectImpl)obj)._get_delegate ();
      return new _FBK33200remotStub (delegate);
    }
  }
}
```

## 13.36. ARXIU *FBK33200remotHolder.java*

```
/**
* FBK33200remotHolder.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
*/

public final class FBK33200remotHolder implements
org.omg.CORBA.portable.Streamable
{
  public FBK33200remot value = null;

  public FBK33200remotHolder ()
  {
  }

  public FBK33200remotHolder (FBK33200remot initialValue)
  {
    value = initialValue;
  }

  public void _read (org.omg.CORBA.portable.InputStream i)
  {
    value = FBK33200remotHelper.read (i);
  }

  public void _write (org.omg.CORBA.portable.OutputStream o)
  {
    FBK33200remotHelper.write (o, value);
  }

  public org.omg.CORBA.TypeCode _type ()
  {
    return FBK33200remotHelper.type ();
  }

}
```

## 13.37. ARXIU *ExcepcioFBK33200remota.java*

```
package FBK33200remotPackage;
/**
* FBK33200remotPackage/ExcepcioFBK33200remota.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
*/

public final class ExcepcioFBK33200remota extends
org.omg.CORBA.UserException implements org.omg.CORBA.portable.IDLEntity
{

  public ExcepcioFBK33200remota ()
  {
  } // ctor

} // class ExcepcioFBK33200remota
```

## 13.38. ARXIU *ExcepcioFBK33200remotaHelper.java*

```
package FBK33200remotPackage;


/**
* FBK33200remotPackage/ExcepcioFBK33200remotaHelper.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
*/

abstract public class ExcepcioFBK33200remotaHelper
{
  private static String  _id =
"IDL:FBK33200remot/ExcepcioFBK33200remota:1.0";

  public static void insert (org.omg.CORBA.Any a,
FBK33200remotPackage.ExcepcioFBK33200remota that)
  {
    org.omg.CORBA.portable.OutputStream out = a.create_output_stream ();
    a.type (type ());
    write (out, that);
    a.read_value (out.create_input_stream (), type ());
  }

  public static FBK33200remotPackage.ExcepcioFBK33200remota extract
(org.omg.CORBA.Any a)
  {
    return read (a.create_input_stream ());
  }

  private static org.omg.CORBA.TypeCode __typeCode = null;
  private static boolean __active = false;
  synchronized public static org.omg.CORBA.TypeCode type ()
  {
    if (__typeCode == null)
    {
      synchronized (org.omg.CORBA.TypeCode.class)
      {
        if (__typeCode == null)
        {
          if (__active)
          {
            return org.omg.CORBA.ORB.init().create_recursive_tc ( _id );
          }
          __active = true;
          org.omg.CORBA.StructMember[] _members0 = new
org.omg.CORBA.StructMember [0];
          org.omg.CORBA.TypeCode _tcOf_members0 = null;
          __typeCode = org.omg.CORBA.ORB.init ().create_struct_tc
(FBK33200remotPackage.ExcepcioFBK33200remotaHelper.id (),
"ExcepcioFBK33200remota", _members0);
          __active = false;
        }
      }
    }
    return __typeCode;
  }

  public static String id ()
```

```
  {
    return _id;
  }

  public static FBK33200remotPackage.ExcepcioFBK33200remota read
(org.omg.CORBA.portable.InputStream istream)
  {
    FBK33200remotPackage.ExcepcioFBK33200remota value = new
FBK33200remotPackage.ExcepcioFBK33200remota ();
    // read and discard the repository ID
    istream.read_string ();
    return value;
  }

  public static void write (org.omg.CORBA.portable.OutputStream ostream,
FBK33200remotPackage.ExcepcioFBK33200remota value)
  {
    // write the repository ID
    ostream.write_string (id ());
  }

}
```

## 13.39. ARXIU *ExcepcioFBK33200remotaHolder.java*

```
package FBK33200remotPackage;

/**
* FBK33200remotPackage/ExcepcioFBK33200remotaHolder.java
* Generated by the IDL-to-Java compiler (portable), version "3.0"
* from FBK33200remot.idl
*/

public final class ExcepcioFBK33200remotaHolder implements
org.omg.CORBA.portable.Streamable
{
  public FBK33200remotPackage.ExcepcioFBK33200remota value = null;

  public ExcepcioFBK33200remotaHolder ()
  {
  }
  public ExcepcioFBK33200remotaHolder
(FBK33200remotPackage.ExcepcioFBK33200remota initialValue)
  {
    value = initialValue;
  }
  public void _read (org.omg.CORBA.portable.InputStream i)
  {
    value = FBK33200remotPackage.ExcepcioFBK33200remotaHelper.read (i);
  }
  public void _write (org.omg.CORBA.portable.OutputStream o)
  {
    FBK33200remotPackage.ExcepcioFBK33200remotaHelper.write (o, value);
  }
  public org.omg.CORBA.TypeCode _type ()
  {
    return FBK33200remotPackage.ExcepcioFBK33200remotaHelper.type ();
  }
}
```

## 13.40. ARXIU *FBK33200servent.java*

```
public class FBK33200servent extends _FBK33200remotImplBase
{
    private FBK33200local maqueta;

    public FBK33200servent(){}

          public void finalize()
   {
          if(maqueta!=null) maqueta.finalize();
   }

   public void assignarReferenciaMatlab () throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          try{
                synchronized(this) {
                maqueta=new FBK33200local();
                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
   }

   public void alliberarReferenciaMatlab ()
   {
                synchronized(this) {
                if(maqueta!=null)maqueta.finalize();
                }
   }

   public  int getTipusControl()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          int resultat=0;
          try{
                synchronized(this) {
                      if(maqueta!=null)resultat=maqueta.getTipusControl();
                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
   }

   public  float[] getParFiltrePosCarro()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          float[] resultat=null;
          try{
                synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getParFiltrePosCarro();
                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
```

```
   }

   public  float[] getParFiltreVelCarro()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         float[] resultat=null;
         try{
               synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getParFiltreVelCarro();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
   }


   public  float getConsigna()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         float resultat=0;
         try{
               synchronized(this) {
                     if(maqueta!=null)resultat=maqueta.getConsigna();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
   }

   public  int getDivisorRTK()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         int resultat=0;
         try{
               synchronized(this) {
                     if(maqueta!=null)resultat=maqueta.getDivisorRTK();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
   }


   public  float[] getHistorialTemps()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         float[] resultat=null;
         try{
               synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getHistorialTemps();

               }
         }catch(Exception e){
```

```
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
    }

    public  float[] getHistorialAnglePendul()throws
FBK33200remotPackage.ExcepcioFBK33200remota
    {
          float[] resultat=null;
          try{
                synchronized(this) {

    if(maqueta!=null)resultat=maqueta.getHistorialAnglePendul();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
    }

    public  float[] getHistorialVelPendul()throws
FBK33200remotPackage.ExcepcioFBK33200remota
    {
          float[] resultat=null;
          try{
                synchronized(this) {

    if(maqueta!=null)resultat=maqueta.getHistorialVelPendul();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
    }

    public  float[] getHistorialPosCarro()throws
FBK33200remotPackage.ExcepcioFBK33200remota
    {
          float[] resultat=null;
          try{
                synchronized(this) {

    if(maqueta!=null)resultat=maqueta.getHistorialPosCarro();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
    }

    public  float[] getHistorialVelCarro()throws
FBK33200remotPackage.ExcepcioFBK33200remota
    {
          float[] resultat=null;
          try{
                synchronized(this) {
```

```
   if(maqueta!=null)resultat=maqueta.getHistorialVelCarro();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
   }

   public  float[] getHistorialAccioControlRelativa()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          float[] resultat=null;
          try{
                synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getHistorialAccioControlRelativa();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
   }

   public  float[] getHistorialConsigna()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          float[] resultat=null;
          try{
                synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getHistorialConsigna();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
   }


   public  int getNumMostresBufer()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          int resultat=0;
          try{
                synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getNumMostresBufer();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
   }

   public  float[] getParametresControl()throws
FBK33200remotPackage.ExcepcioFBK33200remota
```

```
   {
         float[] resultat=null;
         try{
               synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getParametresControl();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
   }


   public  float[] getParFiltrePosPendul()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         float[] resultat=null;
         try{
               synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getParFiltrePosPendul();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
   }

   public  float[] getParFiltreVelPendul()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         float[] resultat=null;
         try{
               synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getParFiltreVelPendul();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
   }


   public  float[] getPW()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         float[] resultat=null;
         try{
               synchronized(this) {
                     if(maqueta!=null)resultat=maqueta.getPW();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
```

```
   }

   public  float getPeriodeMostreig()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         float resultat=0;
         try{
               synchronized(this) {

   if(maqueta!=null)resultat=maqueta.getPeriodeMostreig();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
   }

   public  int carregaLlibreria()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         int resultat=0;
         try{
               synchronized(this) {

   if(maqueta!=null)resultat=maqueta.carregaLlibreria();
               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
         return resultat;
   }

   public  void carregaControladorExtern(String fitxer)throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         try{
               synchronized(this) {

   if(maqueta!=null)maqueta.carregaControladorExtern(fitxer);

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
   }

   public  void tempsAzero()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         try{
               synchronized(this) {
                     if(maqueta!=null)maqueta.tempsAzero();

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
   }
```

```
   public  void setTipusControl(int tipus)throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         try{
               synchronized(this) {
                     if(maqueta!=null)maqueta.setTipusControl(tipus);

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
   }

   public  void setParFiltrePosCarro(float[] parametres)throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         try{
               synchronized(this) {

   if(maqueta!=null)maqueta.setParFiltrePosCarro(parametres);

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
   }

   public  void setParFiltreVelCarro(float[] parametres)throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         try{
               synchronized(this) {

   if(maqueta!=null)maqueta.setParFiltreVelCarro(parametres);

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
   }


   public  void setConsigna(float r)throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         try{
               synchronized(this) {
                     System.out.println("Nou valor Consigna: "+r+" ");
                     if(maqueta!=null)maqueta.setConsigna(r);

               }
         }catch(Exception e){
               throw new FBK33200remotPackage.ExcepcioFBK33200remota();
         }
   }

   public  void setDivisorRTK(int divisor)throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
         try{
               synchronized(this) {
```

```
                        if(maqueta!=null)maqueta.setDivisorRTK(divisor);

                  }
            }catch(Exception e){
                  throw new FBK33200remotPackage.ExcepcioFBK33200remota();
            }
      }

      public  void setParametresControl(float[] parametres)throws
FBK33200remotPackage.ExcepcioFBK33200remota
      {
            try{
                  synchronized(this) {

      if(maqueta!=null)maqueta.setParametresControl(parametres);

                  }
            }catch(Exception e){
                  throw new FBK33200remotPackage.ExcepcioFBK33200remota();
            }
      }


      public  void setParFiltrePosPendul(float[] parametres)throws
FBK33200remotPackage.ExcepcioFBK33200remota
      {
            try{
                  synchronized(this) {

      if(maqueta!=null)maqueta.setParFiltrePosPendul(parametres);

                  }
            }catch(Exception e){
                  throw new FBK33200remotPackage.ExcepcioFBK33200remota();
            }
      }

      public  void setParFiltreVelPendul(float[] parametres)throws
FBK33200remotPackage.ExcepcioFBK33200remota
      {
            try{
                  synchronized(this) {

      if(maqueta!=null)maqueta.setParFiltreVelPendul(parametres);

                  }
            }catch(Exception e){
                  throw new FBK33200remotPackage.ExcepcioFBK33200remota();
            }
      }


      public  void setPW(float[] parametres)throws
FBK33200remotPackage.ExcepcioFBK33200remota
      {
            try{
                  synchronized(this) {
                        if(maqueta!=null)maqueta.setPW(parametres);

                  }
            }catch(Exception e){
```

```
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
   }

   public  void setPeriodeMostreig(float ts)throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          try{
                synchronized(this) {
                      if(maqueta!=null)maqueta.setPeriodeMostreig(ts);

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
   }

   public  void iniciarAdquisicioDades()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          try{
                synchronized(this) {
                      if(maqueta!=null)maqueta.iniciarAdquisicioDades();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
   }

    public void paraCarro ()throws
FBK33200remotPackage.ExcepcioFBK33200remota
    {
        try {
             synchronized(this) {
                if(maqueta!=null)maqueta.paraCarro();

             }
        } catch(Exception e) {
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
        }
    }

   public  int descarregaLlibreria()throws
FBK33200remotPackage.ExcepcioFBK33200remota
   {
          int resultat=0;
          try{
                synchronized(this) {

   if(maqueta!=null)resultat=maqueta.descarregaLlibreria();

                }
          }catch(Exception e){
                throw new FBK33200remotPackage.ExcepcioFBK33200remota();
          }
          return resultat;
   }


}
```

## 13.41. ARXIU *ServidorFBK33200.java*

```
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;


import java.awt.*;
import java.awt.event.*;


class ServidorFBK33200
{

    public static void main(String args[])
    {
   try{

         FBK33200local instLocal= new FBK33200local();

       // creacio i inicialitzacio de l'objecte ORB
       ORB orb = ORB.init(args, null);

       /* creacio de l'objecte FBK33200local per controlar per part del
          servidor de la maqueta */


       // creacio de l'objecte servent i registre del mateix amb l'ORB
       FBK33200servent instRemota = new FBK33200servent();
       orb.connect(instRemota);

        // Obtenim el context de noms de l'arrel
       org.omg.CORBA.Object objRef =
         orb.resolve_initial_references("NameService");
       NamingContext ncRef = NamingContextHelper.narrow(objRef);

       // bateig de la referencia de l'objecte a la xarxa
       NameComponent nc = new NameComponent("FBK33200remot", "");
       NameComponent path[] = {nc};
       ncRef.rebind(path, instRemota);


       // espera d'invocacions per part de clients
        java.lang.Object sync = new java.lang.Object();
         FinestraControlServidorFBK33200 instFcsFBK33200;
         instFcsFBK33200=new FinestraControlServidorFBK33200(instLocal);
         System.out.println("Ja es poden processar comandes del client");

         synchronized (sync)
        {
         sync.wait();
        }

   } catch (Exception e) {
       System.err.println("ERROR: " + e);
       e.printStackTrace(System.out);
   }
    }
}
```

```
class FinestraControlServidorFBK33200 extends Frame implements
WindowListener
{
   private FBK33200local instLocal;

   public FinestraControlServidorFBK33200(FBK33200local instLocal)
   {
         this.instLocal=instLocal;
         super.setTitle("Control del servidor del Pont-Grua");
         super.add(new Label
                     ("Tanca la finestra per clausurar el servidor"));

         super.setSize(350,200);
         super.setLayout(new GridLayout(3,3,200,20));
         super.setVisible(true);

         this.addWindowListener(this);

   }


   public void windowClosing(WindowEvent we)
   {
         instLocal.finalize();
         System.exit(0);
   }
   public void windowOpened(WindowEvent we){}
   public void windowClosed(WindowEvent we){}
   public void windowIconified(WindowEvent we){}
   public void windowDeiconified(WindowEvent we){}
   public void windowActivated(WindowEvent we){}
   public void windowDeactivated(WindowEvent we){}

}
```

## 13.42. ARXIU *ClientFBK33200.java*

```
import org.omg.CosNaming.*;
import org.omg.CORBA.*;

import java.io.*;

public class ClientFBK33200
{
    public static void main(String args[])throws Exception
    {
       // creacio i inicialitzacio de l'objecte ORB
       ORB orb = ORB.init(args, null);

        // Obtenim el context de noms de l'arrel
        org.omg.CORBA.Object objRef =
                           orb.resolve_initial_references("NameService");
        NamingContext ncRef = NamingContextHelper.narrow(objRef);

        //Obtenim la referencia de l'objecte remot
        NameComponent nc = new NameComponent("FBK33200remot", "");
        NameComponent path[] = {nc};
        FBK33200remot instRemota =
                        FBK33200remotHelper.narrow(ncRef.resolve(path));

        //A partir d'aqui com si fos local

         instRemota.assignarReferenciaMatlab();

         float[] parPID=new float[7];
         parPID[0]=(float)0.1;
         parPID[1]=(float)-0.294;
         parPID[2]=(float)0;
         parPID[3]=(float)-0.0042;
         parPID[4]=(float)0.5;
         parPID[5]=(float)1;
         parPID[6]=(float)0.5;

         instRemota.setParametresControl(parPID);
         instRemota.setConsigna((float)0);
         instRemota.setPeriodeMostreig((float)0.01);
         instRemota.setTipusControl(FBK33200remot.PID);


         InputStreamReader teclat=new InputStreamReader(System.in);
         BufferedReader liniaTeclat = new BufferedReader(teclat);
         System.out.println("Escriu la posicio desitjada de -0.5 a +0.5");
         System.out.println("Tecleja SORTIR per acabar el programa");
         System.out.println
                        ("Tecleja ACTUAL per obtenir la posicio actual");

         String linia;
         float novaConsigna=0;
         float posicioActual=0;
         float[] historialPosicions;
         int mostresRebudes=0;
```

```
            while(true)
            {
                  linia=liniaTeclat.readLine();

                  if(linia.equalsIgnoreCase("sortir"))
                  {
                        break;
                  }

                  if(linia.equalsIgnoreCase("actual"))
                  {
                        do{
                          mostresRebudes=instRemota.getNumMostresBufer();
                        }while(mostresRebudes==0);

                        historialPosicions=instRemota.getHistorialPosCarro();
                        posicioActual=historialPosicions[mostresRebudes-1];
                        instRemota.tempsAzero();

                        System.out.println("POSICIO ACTUAL = "+posicioActual+" ");

                        continue;
                  }

                  try{

                        novaConsigna=Float.parseFloat(linia);

                        if(novaConsigna>0.5 || novaConsigna<-0.5)
                        {
                              System.out.println
                                    ("posicio especificada fora de rang!!");
                              System.out.println
                                    ("Posicio ha d'estar entre -0.5 i +0.5");
                        }
                        else
                        {
                              instRemota.setConsigna(novaConsigna);
                        }

                  }catch(NumberFormatException excepcioNumerica){

                        System.out.println("Format incorrecte!!!");
                        System.out.println
                              ("Escriu la posicio desitjada de -0.5 a +0.5");
                        System.out.println
                                 ("Tecleja SORTIR per acabar el programa");
                        System.out.println
                              ("Tecleja ACTUAL per obtenir la posicio actual");

                  }

            }

            instRemota.paraCarro();
            instRemota.alliberarReferenciaMatlab();

            System.out.println("S'ha acabat el programa");
      }
}
```