UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA TÈCNICA SUPERIOR ENGINYERIA INDUSTRIAL VALÈNCIA

TREBALL FINAL DE GRAU EN ENGINYERIA EN TECNOLOGIES INDUSTRIALS

DESENVOLUPAMENT D'UNA INTERFÍCIE BASADA EN LABVIEW PER AL POSICIONAMENT GLOBAL I GENERACIÓ DE TRAJECTÒRIES D'UN ROBOT PARAL·LEL MITJANÇANT L'ÚS DE MARCADORS

AUTOR: Eros Iván Costa Andrés

TUTORA: Marina Vallés Miquel

COTUTOR: Rafael José Escarabajal Sánchez

Curs Acadèmic: 2019-20

"A la meua família, per donar-me suport i creure sempre en mi,
als meus amics i companys, per tot el viscut en aquests anys inoblidables,
a la meua tutora Marina, pels seus consells i tota la confiança depositada."

RESUMEN

En este proyecto se presenta el diseño de una interfaz programada en LabVIEW para verificar y visualizar, en tiempo real, la posición y orientación de objetos en el espacio 3D, usando el sistema de captura de movimiento óptico con marcadores OptiTrack.

El objetivo del presente trabajo es crear un control de los errores en el posicionamiento de la plataforma de un robot paralelo de 4 grados de libertad, examinando la ubicación real frente a la referencia. Además, puede ayudar a solucionar el problema de las configuraciones singulares que se dan en el robot.

La aplicación se conecta con el software de las cámaras mediante la tecnología .NET para acceder a los datos. Al mismo tiempo, estos datos se pueden almacenar y enviar a otros dispositivos usando los protocolos TCP/IP.

Adicionalmente, se ha desarrollado un módulo para la unidad de control del robot. Éste envía a través de una conexión TCP/IP, la posición de la plataforma, calculada a partir de las medidas de los sensores internos, mediante un modelo cinemático dado.

Palabras Clave: LabVIEW, marcadores, OptiTrack, robot paralelo, posicionamiento, .NET, TCP.

RESUM

En aquest projecte es presenta el disseny d'una interfície programada amb LabVIEW per verificar i visualitzar, en temps real, la posició i orientació d'objectes en l'espai 3D, usant el sistema de captura de moviment òptic amb marcadors Optitrack.

L'objectiu del present treball és crear un control dels errors en el posicionament de la plataforma d'un robot paral·lel de 4 graus de llibertat, examinant la ubicació real front a la referència. A més, pot ajudar a solucionar el problema de les configuracions singulars que es donen al robot.

L'aplicació es connecta amb el software de les càmeres mitjançant la tecnologia .NET per accedir a les dades. Al mateix temps, aquestes dades es poden emmagatzemar o enviar a altres dispositius utilitzant els protocols TCP/IP.

Addicionalment, s'ha desenvolupat un mòdul per a la unitat de control del robot. Aquest envia a través d'una connexió TCP/IP, la posició de la plataforma, calculada partint de les mesures dels sensors interns, mitjançant un model cinemàtic donat.

Paraules clau: LabVIEW, marcadors, OptiTrack, robot paral·lel, posicionament, .NET, TCP.

ABSTRACT

This project presents the design of an interface programmed in LabVIEW to verify and visualize, in real time, the position and orientation of objects in 3D space, using an OptiTrack optical motion capture system with markers.

The objective of the present paper is to create a control of the errors in the positioning of the platform of a 4-degree-of-freedom parallel robot, examining the real location versus the reference. Moreover, it can help to solve the problem of the singular configurations that occur in the robot.

The application connects to the camera's software using .NET technology to access the data. At the same time, it can store and send this data to other devices using the TCP/IP protocols.

Additionally, a module for the robot control unit has been developed. It sends through a TCP/IP connection, the position of the platform, calculated from the internal sensor's measurements, using a given kinematic model.

Keywords: LabVIEW, markers, OptiTrack, parallel robot, positioning, .NET, TCP.

ÍNDEX

DOCUMENTS CONTINGUTS EN EL TFG

- Memòria Descriptiva
- Pressupost
- Manual de l'Usuari
- Annex: Manual del programador

ÍNDEX DE LA MEMÒRIA DESCRIPTIVA

ÍNDEX DEL PRESSUPOST

ÍNDEX DEL MANUAL DE L'USUARI

ÍNDEX DE L'ANNEX: MANUAL DEL PROGRAMADOR

TREBALL FINAL DE GRAU EN ENGINYERIA EN TECNOLOGIES INDUSTRIALS

MEMÒRIA DESCRIPTIVA

AUTOR: Eros Iván Costa Andrés

TUTORA: Marina Vallés Miquel

COTUTOR: Rafael José Escarabajal Sánchez

Curs Acadèmic: 2019-20

ÍNDEX DE LA MEMÒRIA DESCRIPTIVA

CAPÍTOL 1. INTRODUCCIÓ

1.1. OBJECTE

El present treball final de grau es basa en programar una interfície que permeta a l'usuari contrastar en temps real la posició global, i les trajectòries, de la plataforma del robot paral·lel. Ho aconsegueix visualitzant per pantalla la posició i orientació real front a la referència. Per una part, es grafica la ubicació teòrica, que s'obté en cada moment amb el model cinemàtic i les mesures dels sensors de les potes del robot. D'altra banda, es grafica la posició real, entregada per un sistema de càmeres capaç de rastrejar els moviments d'uns marcadors, situats sobre la plataforma mòbil.

Concretament, l'aplicació fa accessibles les dades de posició que entrega el sistema de rastreig OptiTrack. Es connecta al software que controla les càmeres, usant la tecnologia .NET. Paral·lelament, estableix una connexió TCP/IP amb la unitat de control del robot, per tal d'enviar i/o rebre dades de posició i orientació del sòlid amb el que es treballe.

A més, s'ha programat un mòdul per a la unitat de control del robot, encarregat de calcular la ubicació teòrica i connectar-se a la interfície de visualització. A partir de les mesures dels sensors interns del robot, calcula com varia la posició i orientació de la plataforma. Posteriorment, transmet totes les dades a la pròpia aplicació del projecte.

1.2. JUSTIFICACIÓ

El posicionament dels robots és un pilar fonamental de la robòtica, i en moltes ocasions de costosa resolució. Més encara si es tracta d'un robot paral·lel, que degut a com està configurada la seua estructura, és a dir, com són les relacions entre les diferents cadenes d'articulacions, la cinemàtica es torna molt més complicada. Moltes vegades és necessari recórrer a mètodes iteratius per poder resoldre els sistemes, com en el cas del robot d'estudi. Aquests no deixen de ser mètodes matemàtics, que troben una aproximació òptima amb més o menys precisió a la solució real; que per als casos dels punts singulars, pot no ser única o ni tal sols existir.

Addicionalment, s'ha de tenir en compte que les dades de partida per a la resolució dels sistemes són les dades entregades pels sensors del robot, que concretament són encorders incrementals. Aquests, com qualsevol tipus de sensor, poden afegir errors en la mesura, deguts per exemple a un mal calibratge o defectes en la pròpia fabricació. Aquests possibles errors es tradueixen en una major inexactitud en el posicionament final de la plataforma.

Per últim, és impossible conèixer les coordenades globals de la plataforma, la causa és que sols es pot calcular la posició relativa de la plataforma, perquè el model cinemàtic utilitza les mesures dels encoders, els quals són sensors incrementals. Ho aconsegueix amb les dimensions del robot i les diferents variacions en la configuració dels seus components. Per poder obtenir la posició global final, s'ha d'introduir inicialment unes coordenades com a referència, per aplicar-li després la variació relativa que es produisca.

Per tots aquests motius, és necessari disposar d'un sistema de sensors externs capaç de monitoritzar la posició global exacta del robot en cada moment. D'aquesta forma, es pot conèixer el seu moviment en coordenades globals, i comparar-lo amb l'obtingut segons el model teòric. Així, l'error sistemàtic que puga donar-se es pot reduir o inclús arribar a corregir-se, si es detecta l'origen. Més encara, una verificació en temps real permet un millor estudi del comportament del model davant canvis en l'escenari de treball, com per exemple, els diferents esforços que haja d'aplicar o suportar la plataforma en cada tipus d'assaig.

El disseny de l'aplicació sorgeix de la necessitat d'haver de treballar simultàniament amb les dades del sistema de sensors externs de l'OptiTrack, i les de la unitat de control del robot. Malgrat tots els avantatges que presenta el software de control de les càmeres, també presenta una gran limitació. Degut al seu caràcter tancat, no permet treballar conjuntament amb informació provinent de fonts externes a les càmeres. Aquest fet, impossibilita utilitzar aquest software per poder validar les dades de la manera necessària.

L'aplicació d'aquest projecte constitueix un mecanisme de control del posicionament, en aquest cas d'un robot paral·lel, que permet afinar els resultats dels assajos que es realitzen. La interfície és una ferramenta d'investigació útil per poder perfeccionar el comportament del robot, proporcionant la possibilitat d'identificar errors de diversos tipus en el posicionament global d'aquest.

1.3. OBJECTIUS

L'objectiu principal que persegueix aquest Treball Final de Grau és la programació d'una aplicació que connecte les dades de posició del sistema OptiTrack amb les dades de la unitat de control del robot paral·lel, amb la finalitat de supervisar i millorar la fiabilitat dels resultats en la ubicació de la plataforma en temps real.

A més de l'objectiu principal, es busca que l'aplicació i el seu codi siguen, en la mesura del possible, sostenibles i reutilitzables. La intenció és poder dotar-la d'utilitat en el màxim d'escenaris possibles d'investigació on intervinga el sistema de captura de moviment, i es requerisca processar o comparar els moviments de diversos sòlids en temps real, ja siga rebent les posicions d'altres aplicacions o carregant les trajectòries d'experiments anteriors.

Per poder aconseguir aquests propòsits, l'aplicació ha de complir amb els següents requeriments:

- Graficar la posició i orientació de diferents sòlids rígids, de forma simultània i en temps real.
- Comunicar-se amb la unitat de control del robot, per obtenir la ubicació que deuria tindre la plataforma segons el model. Així com poder enviar-li les dades de posició de les càmeres en els casos necessaris.

- Connectar-se al software de l'OptiTrack per accedir a la ubicació real dels sòlids rígids que s'hagen definit.
- Poder guardar i carregar diferents assajos, per tal de comparar les diferents trajectòries resultants obtingudes en diverses sessions.
- Ser flexible i versàtil, estant dissenyada al màxim possible, de forma modular i independent de l'ús específic que se li done en cada moment.

Finalment, respecte del mòdul addicional de la cinemàtica, es necessita que siga capaç de calcular les coordenades finals de la plataforma, perquè es puga visualitzar de forma efectiva el moviment. A més, ha de poder enviar aquesta informació a la interfície per contrastar els resultats obtinguts.

D'aquesta manera, els diferents objectius que ha de complir aquest mòdul són:

- Calcular la posició i orientació finals del centre geomètric o CG de la plataforma, partint de les mesures dels sensors interns mitjançant el model cinemàtic donat.
- Calcular 3 punts pertanyents a la plataforma, per formar així una superfície i poder simular la plataforma de forma efectiva.
- Comunicar-se amb la interfície desenvolupada, i enviar-li totes les dades de posició calculades.

CAPÍTOL 2. FONAMENTS DEL PROJECTE

2.1. ROBOT PARAL·LEL

Els robots són un tecnologia que ha revolucionat, i modificat de forma permanent, la manera de concebre molts, i cada vegada més, aspectes de la vida tal i com avui la coneguem. En l'actualitat, tenen un paper vital i imprescindible en l'automatització de processos industrials, ajudant a més a augmentar la qualitat i quantitat dels productes fabricats. Destaquen especialment en les industries automobilístiques, electròniques, alimentàries i farmacèutiques, encara que avui en dia ja estan presents en tots els sectors. Cada dia que passa, els avanços tecnològics fan possible que els robots siguen cada vegada més precisos, intel·ligents, ràpids i efectius.

Podem entendre un robot com un manipulador reprogramable i multifuncional, capaç de moure ferramentes/eines, parts o dispositius determinats per poder realitzar un sèrie de tasques concretes [1]. Seguidament, es presenten quins són els elements que els formen, una classificació d'aquests, i les bases per determinar la seua posició.

2.1.1. Components d'un robot

Un robot està composat bàsicament per diferents sòlids rígids, units per diverses articulacions. Dins dels sòlids rígids que el composen, existeixen dos que tenen un nom especial: la base i l'efector final. La base, es tracta del sòlid que està unit amb la primera articulació, i que s'encarrega de fixar el robot. L'efector final o terminal, és l'element més allunyat de la base, on es sol col·locar el component funcional, com podria ser una pinça, un trempant o una plataforma.

Abans de seguir aprofundint en les parts que composen els robots, és necessari conèixer que representen els graus de llibertat d'un sòlid rígid. Els graus de llibertat o GDL són el nombre de coordinades independents que es necessiten per poder descriure completament la posició i orientació del sòlid [2]. Un sòlid rígid en l'espai pot moure's, bàsicament, usant moviments de translació i/o rotació. Altra forma d'entendre els GDL, és veure'ls com la quantitat de moviments dels que disposa un sòlid en l'espai [3].

Segons la dimensió on es treballe els GDL disponibles per al sòlid varien. Si es tracta de 2D, els graus del llibertat que posseeix són 3, podria definir-se completament la seua posició en el pla amb, per exemple, dues coordenades de posició i una d'orientació. Si es treballa en 3D, el sòlid disposa de 6 GDL. Aquests també representen els possibles moviments que pot realitzar, que són: 3 translacions sobre tres eixos i 3 rotacions al voltant d'aquests tres. Ambdós casos mostren en la figura següent:

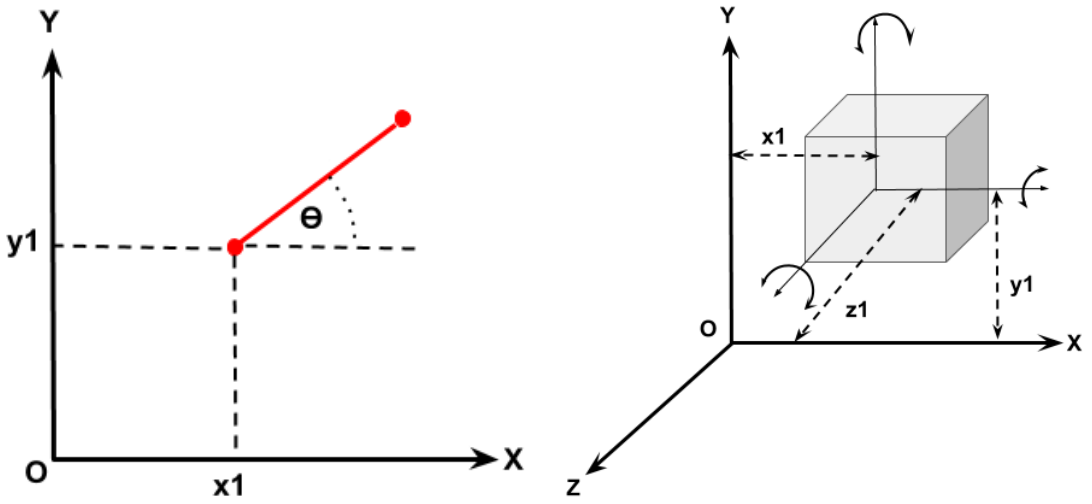

***Fig. 1*. Graus de llibertat en 2D (esquerra) i en 3D (dreta) d'un sòlid rígid**

Nota. Adaptada de *Degrees of Freedom*, per SMLease Design, Mechanism (https://www.smlease.com/entries/mechanism/what-is-degree-of-freedom-dof-in-mechanics/).

La mobilitat que posseeix un robot, ve donada per la quantitat de graus de llibertat que presenta el seu efector final [3]. Aquests GDL vindran imposats per la forma que estiga dissenyat el robot, la configuració dels seus sòlids rígids, i del tipus i nombre d'articulacions que tinga.

Les articulacions, són les connexions entre dos o més sòlids rígids, que permeten un determinat moviment relatiu entre aquests. Existeixen diferents tipus d'articulacions, cadascuna imposa més o menys restriccions o reduccions en els GDL dels sòlids que uneix. Per exemple, per al cas de l'articulació de revolució, permet el moviment de rotació sobre un eix, és a dir, que presenta solament un 1 GDL. D'aquesta forma, imposa 5GDL de restricció, perquè bloqueja les restants 3 translacions i 2 rotacions, que posseeixen els sòlids en l'espai tridimensional. A continuació, en la taula 1 i en la figura 2, es presenten els tipus d'articulacions més comuns:

Taula 1. Articulacions típiques i les seues característiques

Articulació	Nomenclatura	GDL	Restriccions	Descripció
Revolució	R	1	5	Permet la rotació al voltant d'un eix
Prismàtica	P	1	5	Permet la translació rectilínia sobre un eix
Helicoidal	H	1	5	Permet simultàniament la rotació i translació sobre el mateix eix
Cilíndrica	C	2	4	Permet la rotació al voltant d'un eix i, independentment, la translació sobre un altre
Universal	U	2	4	Permet la rotació al voltant de dos eixos independents
Esfèrica	S	3	3	Permet la rotació respecte del centre d'una esfera, al voltant de tres eixos independents

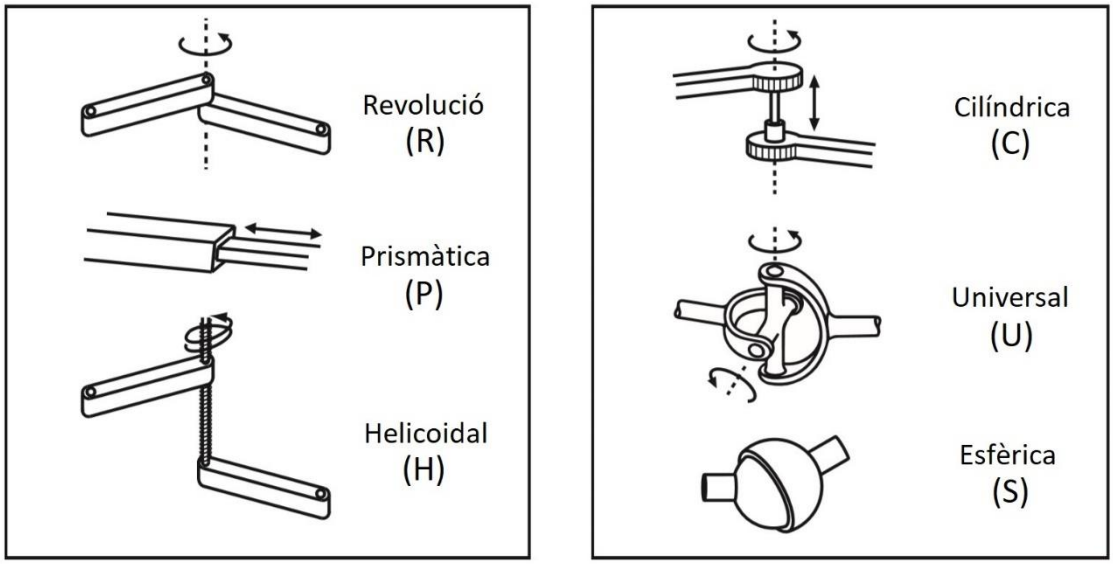

Figura 2. **Representació de les articulacions més típiques**
Nota. Adaptada de *Modern Robotics* (p. 16), per Lynch, K.M. i Park, F.C, 2017, Cambridge University Press.

En cas d'existir el sensor i/o l'actuador, es troba situat dins de l'articulació pertinent. Per una part, el sensor mesura la posició relativa de l'articulació, i normalment són potenciòmetres o encoders incrementals. Per altra banda està l'actuador, que s'encarrega de moure l'articulació i habitualment és un component electromecànic, com per exemple, un servomotor. L'acoblament o unió de diversos sòlids mitjançant diverses articulacions, s'anomena cadena cinemàtica [2]. En la figura següent, estan representats els dos tipus de cadenes diferents que existeixen:

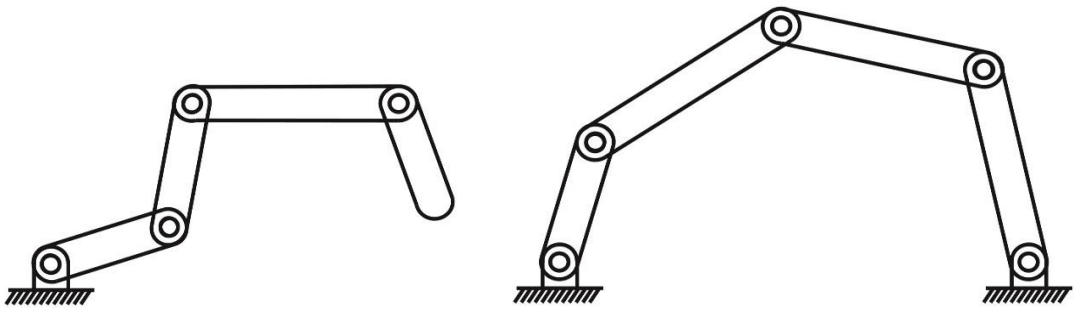

Fig 3. **Cadena cinemàtica oberta (esquerra) i tancada (dreta)**
Nota. Adaptada de *Modern Robotics* (p. 19), per Lynch, K.M. i Park, F.C, 2017, Cambridge University Press.

Finalment, definir que un mecanisme es tracta d'un sistema, format per una o més cadenes cinemàtiques, que converteix els moviment i forces aplicades sobre aquestes, en determinats moviments i forces sobre l'efector final [2]. Un mecanisme pot anomenar-se robot, en cas de que es puga controlar automàticament el seu moviment. Llavors, es pot concloure que és el nombre i tipus de cadenes cinemàtiques que composen al robot, els que determinen els graus de llibertat que l'efector final, i conseqüentment el robot, posseeix.

2.1.2. Classificació

En funció dels criteris que es seleccionen, els robots es poden dividir en multitud de tipus. Per al cas d'interès, es classifiquen atenent al tipus de cadenes cinemàtiques que els formen. D'aquesta forma trobem: els robots sèrie, composats per cadenes cinemàtiques obertes, i els robots paral·lels, formats per cadenes tancades. També existeix un tercer tipus, els robots híbrids, que estan format per cadenes cinemàtiques obertes i tancades, amb l'objectiu d'obtenir els avantatges que presenta cadascun [4].

2.1.2.1. Robots sèrie

Els robots sèrie estan formats per una cadena cinemàtica oberta, és a dir, una successió de sòlids rígids, on cadascun està unit al anterior i al següent mitjançant una articulació, a excepció de la base i l'efector. Al tractar-se de cadenes cinemàtiques obertes, totes les articulacions estan actives en cada moment [5]. Altra característica d'aquest tipus, és la seua estructura antropomòrfica, degut a que s'assimilen a un braç humà, composant-se de diferents articulacions com si fora el muscle, colze i monyica [6].

L'exemple més famós en la industria és el robot SCARA, que significa *Selective Compilance Articulated Robot Arm*. Aquest robot poseeix 4 GDL i està format per 3 articulacions de revolució verticals i una articulació prismàtica. Es tracta d'un tipus de robot dissenyat per a l'enssamblatge, la inspecció, el pick-and-place o l'embalatge, entre altres. En la figura 4, es pot observar l'estructura d'un exemple de robot SCARA. Concretament, es tracta del model SR-6iA desenvolupat per la corporació FANUC, el qual compta amb un espai de treball complet de 360º [7]:

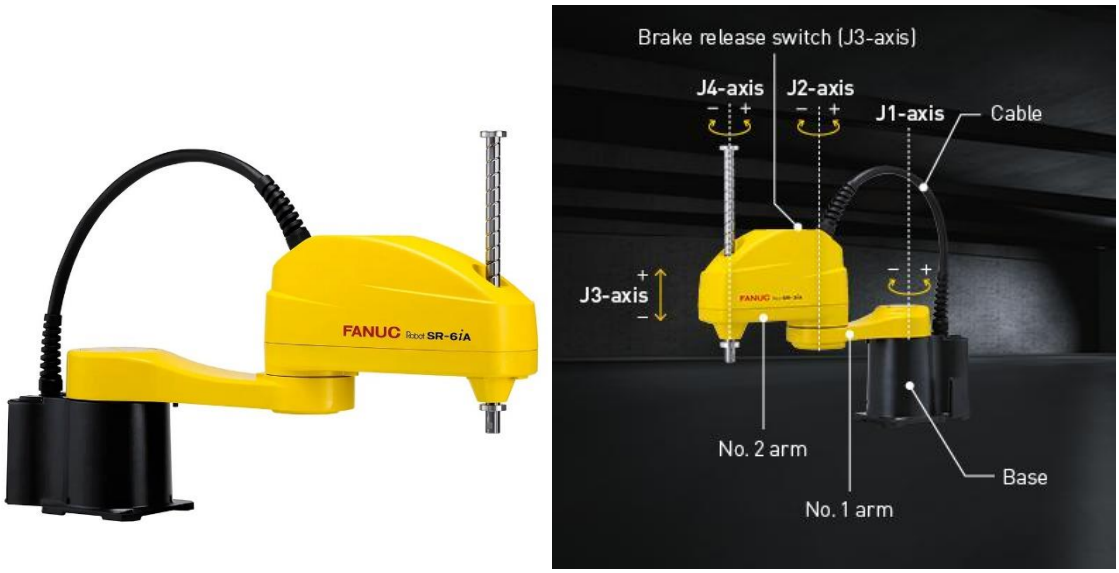

***Fig. 4*. Robot SCARA model SR-6iA de FANUC (a la dreta amb la representació dels GDL)**
Nota. Adaptada de *SCARA SR-6iA*, per FANUC, 2020, SCARA Robots (https://www.fanuc.eu/uk/en/robots/robot-filter-page/scara-series/scara-sr-6ia).

Un dels avantatges dels robot sèrie és l'ampli espai de treball que presenten, és a dir, que existeix una gran volum de l'espai on l'efector final pot actuar sense problemes. A continuació, es mostra en la figura 5, l'espai de treball de l'exemple de robot sèrie presentat anteriorment. També destaquen per la relativa facilitat que presenten a l'hora de determinar la posició de l'efector final, partint de la posició de les articulacions, el que s'anomena problema cinemàtic directe.

***Fig. 5*. Espai de treball del robot SCARA model SR-6iA de FANUC**
Nota. Adaptada de *Work Envelope*, per FANUC, 2020, SCARA Robots Selection Suport (https://www.fanuc.eu/de/en/robots/robot-filter-page/scara-series/selection-support).

Un dels principals inconvenients és la relativa incapacitat de transportar càrregues elevades, que es deu a la seua arquitectura. La raó és que la seua estructura obliga a que cada segment que la composa haja d'aguantar el pes del segment anterior, i així successivament; llavors tot quest pes condiciona la màxima càrrega que pot suportar l'efector. L'altre gran inconvenient que presenta és la seua poca precisió, ja que ocorre de forma pareguda que amb la càrrega. La seua arquitectura magnifica qualsevol error que es done en els sensors de les articulacions. Un petit error en la mesura del sensors de la primera articulació, pot convertir-se en un error considerable en el posicionament de l'efector final.

2.1.2.2. Robots paral·lels

Els robots paral·lels es composen de cadenes cinemàtiques tancades, que són els mecanismes que formen un bucle tancat entre les seues cadenes. En el cas de les cadenes cinemàtiques tancades, a diferència de les obertes, no totes les articulacions estan actives per a cada configuració, algunes d'aquestes seran passives [5]. La seua arquitectura està formada per una base fixa unida a l'efector mòbil, mitjançant diferents cadenes cinemàtiques, que són una tipus de potes o braços que el suporten.

Un exemple molt conegut d'aquest tipus de robot és la plataforma Stewart-Gough. Aquesta presenta 6 GDL i ho aconsegueix amb 6 actuadors prismàtics, units en parells a tres posicions diferents de la plataforma mitjançant articulacions universals. Un exemple d'aquest tipus és el radiotelescopi AMiBA, *Array for Microwave Background Anisotropy*. Es tracta del telescopi astronòmic hexàpode més gran operatiu actualment, format per una plataforma de 6m de fibra de carboni. Està dissenyat per captar la radiació electromagnètica romanent de les etapes primerenques de l'univers i per captar la distribució dels aglomerats galàctics [8]. A continuació, en la figura 6, es mostra la seua arquitectura:

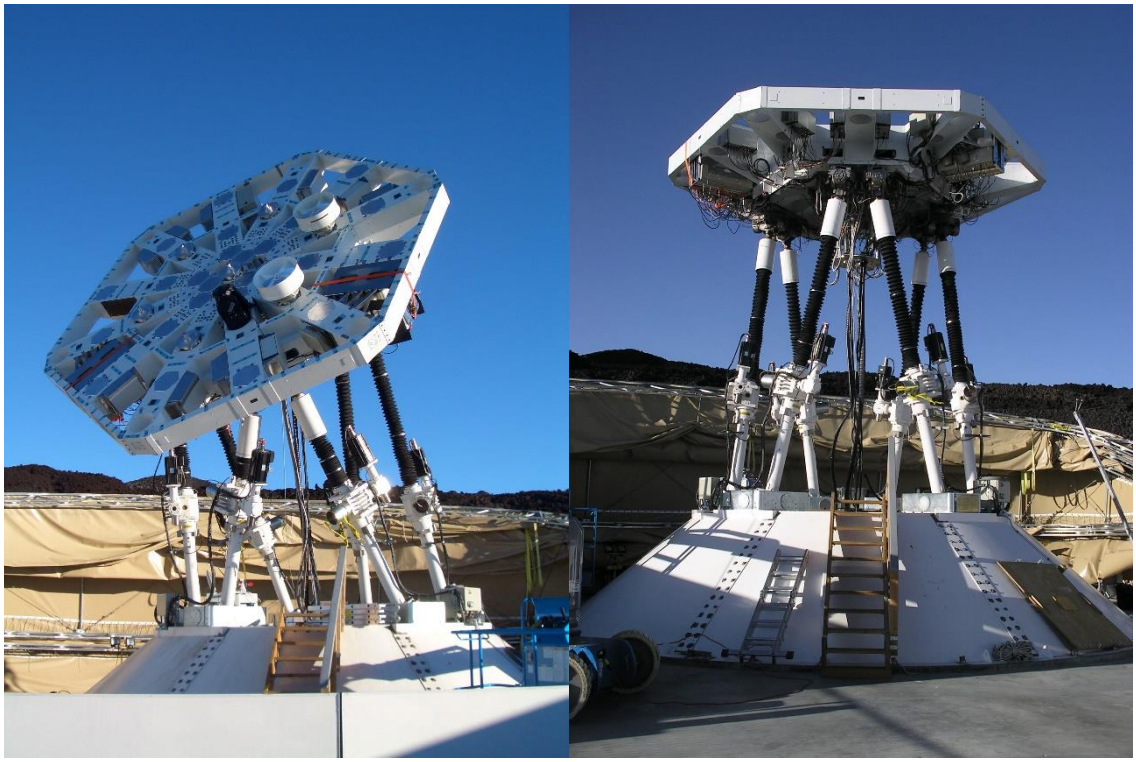

Fig. 6. **El robot paral·lel hexàpode AMiBA en el volcà Mauna Loa de Hawaii**
Nota. Adaptada de "The AMiBA hexapod telescope mount", per Koch, P. M., Kesteven, M., Nishioka, H., Jiang, H., Lin, K. Y., Umetsu, K., ... & Chereau, G. Anuc, 2009, *The Astrophysical Journal*.

Altre tipus de robot paral·lel molt estès és el Delta, que compta amb 4 GDL: 3 translacions i 1 rotació. És molt usat en les operacions de pick-and-place de la industria per la seua elevada velocitat i precisió. El robot d'aquest tipus més ràpid actualment, es tracta del model Quattro 650H dissenyat per l'empresa de robòtica Omron. Compta amb capacitat per poder realitzar més de 200 pick-and-place per minut, és a dir, agafar i col·locar un producte en menys de 0,3s [9]. L'estructura del robot paral·lel Quattro presentat es pot observar en la següent figura:

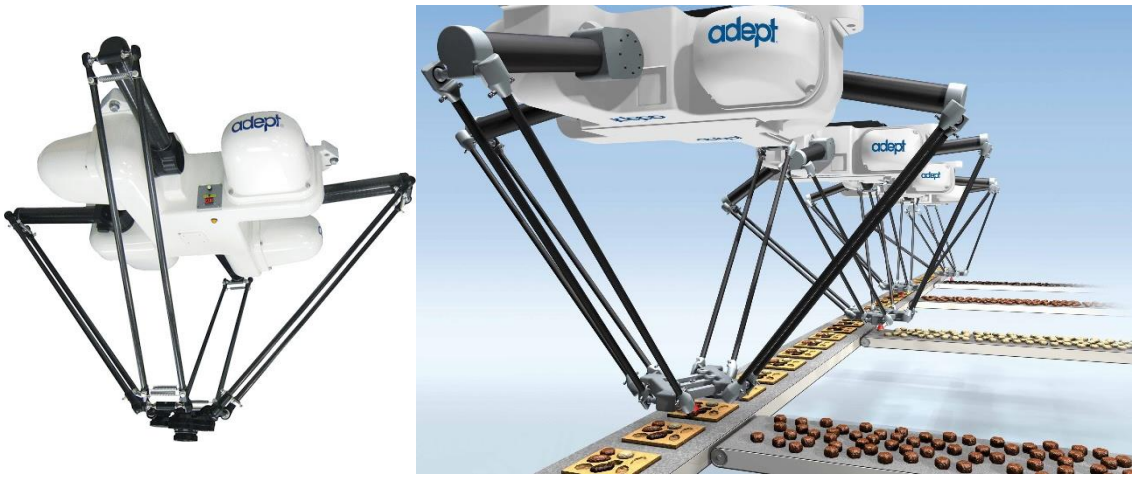

***Fig.* 7. Robot paral·lel model Quattro 650H de Omron**

Nota. Adaptada de *Adept Quattro s650H Robot User's Manual* (p. 13), per Omron (https://assets.omron.com/m/61547d4849409d86/original/Quattro-S650H-Parallel-Robot-Users-Manual.pdf).

Aquest tipus de robots presenten diferents avantatges, un d'ells es la seua capacitat per treballar amb elevats pesos, al contrari que els robot sèrie. En part es deu a la robustesa de la seua estructura, que fa que les cadenes cinemàtiques estiguen sotmeses majoritàriament a esforços de tracció-compressió, amb un flexió molt menys severa que en els de cadena oberta. A més, aquesta capacitat per suportar majors càrregues en el seu efector final, també es deu a que aquesta es divideix, aproximadament d'igual forma, entre totes les cames o braços que la uneixen a la base [3].

Una altra característica a destacar és la seua alta precisió. En aquest tipus de robot, l'error no es propaga de la mateixa forma que ho feia amb els sèries, fent que un error de mesura tinga relativament poc impacte sobre el càlcul de la posició final. D'aquesta forma, si tots els sensors presentaren el mateix error, l'error final que es trobaria a la solució tindria la mateixa magnitud que en un sensor, en compte de tres vegades aquest.

Pel que respecta als inconvenients, trobem el reduït espai de treball que presenten. A causa de la seua estructura, l'espai físic que l'efector final té a l'abast és molt limitat. Un altre desavantatge és l'enorme dificultat que presenta la resolució del problema cinemàtic directe. L'arquitectura del robot complica enormement els càlculs, havent de recórrer a algoritmes iteratius que convergisquen a una solució.

2.1.3. Prototip del laboratori

El prototip del laboratori es tracta d'un robot paral·lel, en procés de desenvolupament, dissenyat per poder dur a terme rehabilitacions de genoll. Aquest compta amb 4 GDL: 2 translacions, en els eixos X i Z; i 2 rotacions, de *yaw* o guinyada (eix Y) i de *roll* o balanceig (eix Z) [10]. La seua estructura es basa en una base fixada al terra unida a la plataforma mòbil, que és l'efector final, mitjançant 4 extremitats o potes. En la següent figura, es mostra una representació esquemàtica dels components i els tipus d'articulacions que posseeix el robot:

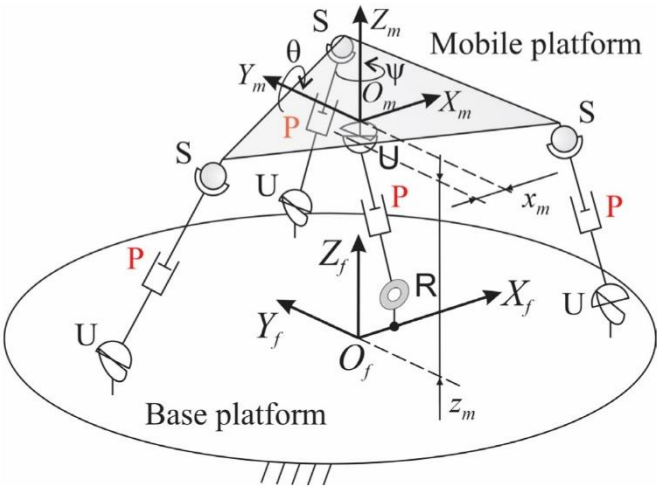

***Fig. 8*. Representació simplificada del robot paral·lel del tipus 3UPS+RPU**

Nota. Adaptada de "Experimental Analysis of Type II Singularities and Assembly Change Points in a 3UPS+RPU Parallel Robot", per Pulloquinga, J. L., Mata, V., Valera, Á., Zamora-Ortiz, P. Díaz-Rodríguez, M. y Zambrano, I., 2020, *Mechanism and Machine Theory*.

Com es pot observar, les 4 extremitats que el formen són cadenes cinemàtiques no idèntiques. Les tres que subjecten els extrems estan formades per la unió del tipus d'articulacions: universal (U), prismàtica (P) i esfèrica (S), és a dir, són unes potes del tipus UPS. La quarta extremitat té una configuració diferent, està formada per una articulació rotacional (R), prismàtica (P) i universal (U); es tracta d'una extremitat del tipus RPU. Aquesta última pota, a banda de tindre una configuració diferent, tampoc està situada als extrems com la resta, sinó que es troba enganxada al centre de la plataforma [10].

D'aquesta forma queda definit el prototip com a un robot paral·lel del tipus 3UPS+RPU, degut al nombre i tipus de cadenes cinemàtiques que el formen. Destacar, que es subratlla la lletra P, al igual que en la figura 8 està marcada en roig, perquè és on estan situats els actuadors, que es tracten de servomotors, i el sensors interns que mesuren la posició de l'articulació, que són encoders incrementals.

És important comentar que existeixen diferents configuracions del robot, cadascuna amb uns paràmetres geomètrics diferents. Aquest fet no afecta al model cinemàtic, ja que està preparat per rebre'ls com a dades d'entrada a l'hora de realitzar els càlculs.

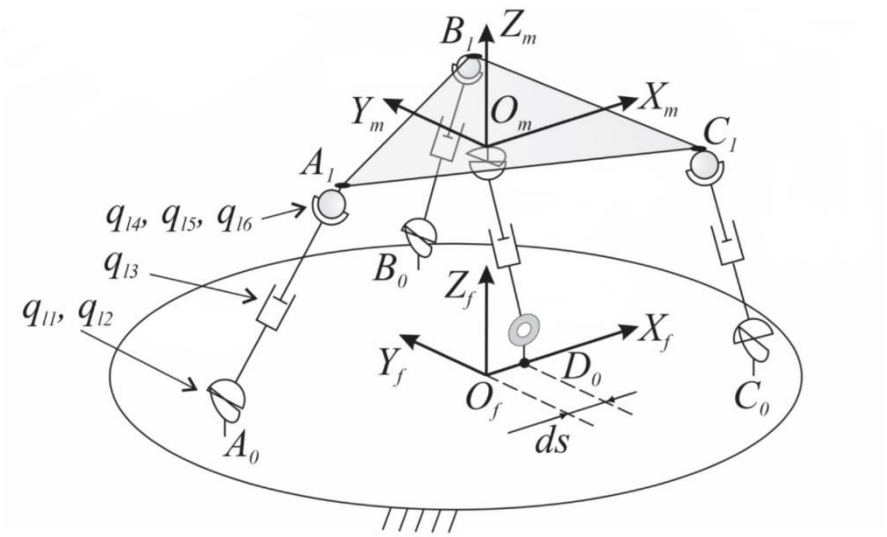

***Fig. 9*. Representació mecànica del robot paral·lel del tipus 3UPS+RPU**

Nota. Adaptada de "Experimental Analysis of Type II Singularities and Assembly Change Points in a 3UPS+RPU Parallel Robot", per Pulloquinga, J. L., Mata, V., Valera, Á., Zamora-Ortiz, P. Díaz-Rodríguez, M. y Zambrano, I., 2020, *Mechanism and Machine Theory*.

En la figura 9, es pot observar un segon esquema del robot, però amb la configuració mecànica del mateix. Sobre la mateixa es troben les variables de posició de les articulacions, que es corresponen amb els GDL que posseeix cada articulació, i que es denoten com q. A més, també es poden observar els tres vèrtex de les plataformes fixa (A_0, B_0, C_0) i mòbil (A_1, B_1, C_1), i la distància ds.

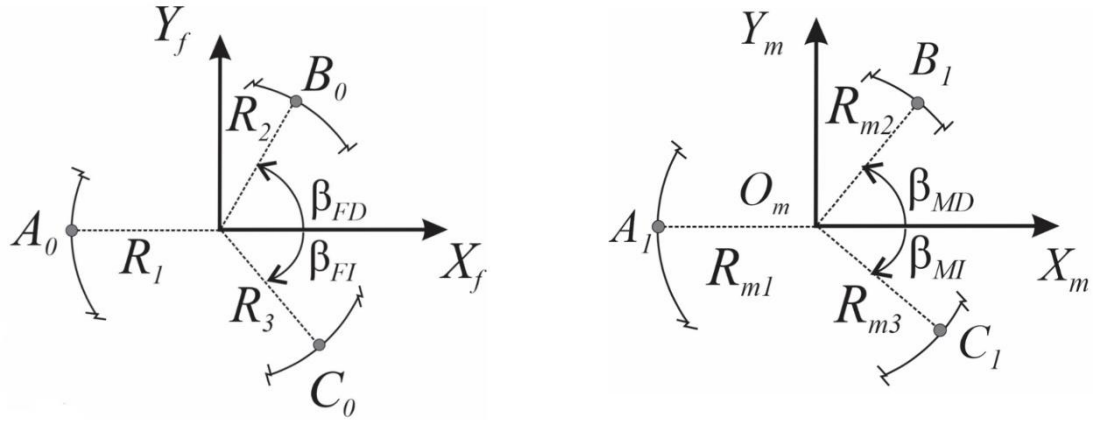

***Fig. 10*. Esquema geomètric de la plataforma fixa (esquerra) i mòbil (dreta)**

Nota. Adaptada de "Experimental Analysis of Type II Singularities and Assembly Change Points in a 3UPS+RPU Parallel Robot", per Pulloquinga, J. L., Mata, V., Valera, Á., Zamora-Ortiz, P. Díaz-Rodríguez, M. y Zambrano, I., 2020, *Mechanism and Machine Theory*.

En la figura 10, estan representats els paràmetres geomètrics que determinen la forma de la plataforma mòbil i fixa. Aquests, varien segons la configuració amb la que es treballe en cada moment. Actualment, els assajos es realitzen en la configuració nº1, els paràmetres geomètrics dels qual es troben representats en la taula 2, que seran els paràmetres d'entrada que s'introduiran en el model per resoldre la cinemàtica del mateix.

Taula 2. Paràmetres geomètrics del robot 3UPS+RPU de la configuració nº1

$R_1\ (m)$	$R_2\ (m)$	$R_3\ (m)$	β_{FD} (°)	β_{FD} (°)	$ds\ (m)$
0'4	0'4	0'4	90	45	0'15
$R_{m1}\ (m)$	$R_{m2}\ (m)$	$R_{m3}\ (m)$	β_{MD} (°)	β_{MI} (°)	
0'3	0'3	0'3	50	90	

Finalment, mencionar que el robot es dirigeix mitjançant la unitat de control. Aquesta es basa en un ordinador industrial d'alt rendiment equipat amb targetes d'adquisició, les quals llegeixen les senyals que envien els encoders incrementals. Al mateix temps, proporcionen les accions de control sobre els diferents actuadors que posseeix el robot, degut a que el moviment de la plataforma mòbil es producte de la combinació de les accions que realitzen els diferents actuadors [10].

2.1.4. Cinemàtica

El problema cinemàtic consisteix en conèixer la posició i velocitat del robot. Concretament, es basa en determinar com afecten les variacions en la longitud o rotació de les articulacions, al moviment de l'efector final, o viceversa. Per poder trobar la relació que existeix entre aquestes coordenades, és necessari determinar el model matemàtic del robot. Aquest model està format per un conjunt d'equacions matemàtiques que descriuen les propietats físiques del sistema, permetent determinar el seu moviment [2].

D'aquesta manera existeixen dues possibilitats diferents segons en el sentit que es necessiten transformar les coordenades. En primer lloc, es troba el problema cinemàtic directe, aquest consisteix en obtenir les coordenades de l'efector final (espai de treball) a partir de la posició i orientació de les diferents articulacions (espai articular). Si pel contrari es necessita obtenir quina ha de ser la configuració de les diferents articulacions, partint d'una posició de l'efector final coneguda, es tracta del problema cinemàtic invers.

Normalment, el que s'ha de resoldre és el problema cinemàtic directe. En aquest cas, les variables conegudes són les coordenades de les diferents articulacions, i el que es desconeix és la posició i orientació de l'efector final. Aquesta és la forma habitual de treball, degut a que la posició de les articulacions es pot obtenir directament a través dels diferents sensors que posseeixen els seus actuadors, però no es possible determinar directament la posició de l'efector final.

2.1.4.1. Matriu Jacobiana

Per trobar les relacions entre el moviment dels actuadors i el moviment de l'efector final, s'ha de definir un conjunt d'equacions de restricció ($\vec{\Phi}$) per al cas de les cadenes cinemàtiques tancades [10]. Aquestes defineixen la relació entre les coordenades generalitzades dels actuadors ($\vec{q}_{ind}$) i els GDL de la plataforma ($\vec{X}$):

$$\vec{\Phi}\left(\vec{X}, \vec{q}_{ind}\right) = \vec{0}$$

Derivant l'expressió anterior respecte del temps, es troben les equacions de velocitat del problema cinemàtic per a un robot paral·lel [10]:

$$J_D\dot{\vec{X}} + J_I\dot{\vec{q}}_{ind} = \vec{0}$$

En aquesta expressió, J_D i J_I representen respectivament la matriu jacobiana del problema cinemàtic directe i invers. Les relacions entre les velocitats de l'espai articular i l'espai de treball s'obtenen mitjançant la matriu Jacobiana. Aquesta representa, en forma matricial, el sistema d'equacions, que permet transformar el moviment de les articulacions al moviment de l'efector final. L'expressió per poder resoldre el problema cinemàtic invers, llavors és de la forma:

$$\dot{\vec{X}} = -{J_D}^{-1}J_I\dot{\vec{q}}_{ind} = J\dot{\vec{q}}_{ind}$$

La matriu Jacobiana (J) es composa per les diferents derivades parcials respecte de cada variable. En el sentit físic, representa l'aportació en direcció, però no en magnitud, de les diferents articulacions sobre la velocitat de l'efector final. En altres paraules, relaciona com afecten els moviments dels actuadors de les cadenes cinemàtiques sobre la plataforma.

La matriu Jacobiana a banda de servir per calcular la cinemàtica, també proporciona informació sobre la dinàmica, i serveix per estudiar la singularitat. La singularitat es tracta d'un problema que sorgeix en algunes configuracions concretes del robot. El que ocorre en aquests punts, on la configuració del robot és singular, és que la matriu Jacobiana té determinant igual a zero. Aquesta propietat indica que la matriu és singular, el que significa que no existeix la seua inversa, degut a que per a eixa configuració la matriu Jacobiana disminueix de rang [11].

El rang d'una matriu ens indica el nombre de files o columnes, en definitiva equacions, linealment independents. Llavors, quan el rang de la matriu Jacobiana disminueix, significa que alguna de fila o columna és combinació lineal de les altres [12]. El que significa que el nombre d'equacions del sistema és menor, existint d'aquesta manera més incògnites que equacions. El problema quan açò ocorre es que la unitat de control no trobarà o donarà múltiples solucions possibles al problema cinemàtic.

2.1.4.2. Configuracions singulars

Els punts singulars realment representen un problema, perquè per a aquestes configuracions, el sistema perd el control sobre el moviment del robot. Les configuracions singulars que pot sofrir un robot paral·lel es poden dividir en dues classes diferents [10]:

- Tipus I: el robot perd almenys un GDL, fent que no puga moure's la plataforma encara que els actuadors sí que tinguen velocitats diferents de zero. Es produeix quan s'anul·la el determinant de la matriu Jacobiana del problema cinemàtic invers: $\|J_I\| = 0$
- Tipus II: el robot guanya com a mínim un GDL, el que causa que siga capaç de moure's la plataforma encara que tots els actuadors estiguen bloquejats. Ocorre quan s'anul·la el determinant de la matriu Jacobiana del problema directe: $\|J_D\| = 0$

El segon tipus de singularitat és el més crític, perquè causa que la plataforma siga incapaç de suportar qualsevol esforç extern, a pesar de que tots els seus actuadors estan bloquejats. Aquest fet causa la pèrdua del control sobre el moviment d'aquesta, sent perillós tant per a l'usuari com per al propi robot [10].

Existeixen diferents solucions possibles per resoldre o pal·liar les configuracions singulars. Una d'aquestes consisteix en dissenyar un robot redundant, de forma que de que estiga composat per més actuadors que graus de llibertat requereix. D'aquesta manera guanya llibertat de moviment, podent evitar passar per les configuracions singulars que es puguen donar en l'espai de treball. El principal inconvenient d'aquest alternativa, és que l'augment d'actuadors i sensors repercuteix directament en un augment en el cost de fabricació i manteniment del robot [6].

Una altra solució possible és la utilització de sensors externs, que són sistemes capaços de captar el moviment del robot. D'aquesta manera els sensors determinen quina és la configuració del robot en cada moment. Llavors, quan la unitat de control es troba amb un punt singular, el sistema de captura de moviment, li proporciona la posició real, per comparar-la amb les solucions obtingudes dels càlculs. D'aquesta forma pot resoldre l'ambigüitat entre les diferents configuracions proporcionades per les equacions.

Un sistema extern de sensors, és útil també a l'hora d'estudiar i ajustar el model. El moviment de la plataforma es realitza a través de l'actuació de les articulacions que conformen les diferents potes. La posició de cadascuna d'elles s'obté mitjançant els sensors incrementals, els quals inevitablement afegeixen xicotets errors. Aquests es deuen al marge o tolerància que tenen les mesures, o també als possibles errors en el calibratge o en la seua pròpia fabricació. A més, també es poden generar errors causats per les folgances i imperfeccions en la fabricació dels components mecànics del robot. Aquests errors, al mateix temps, es veuen amplificats per possibles imprecisions que puga presentar el model cinemàtic del robot.

Tot aquest procés condiciona que la posició teòrica que s'obtinga, no coincidisca exactament amb la que posseeix en realitat la plataforma. A més, encara que teòricament els punts singulars es donen quan el determinant de la matriu Jacobiana és igual a zero, en la realitat es donen per a valor pròxims a zero, degut a la introducció d'aquesta classe d'errors en el model [10].

Per a intentar resoldre el problema de les singularitats i millorar la precisió del robot prototip del laboratori es va optar per instal·lar un sistema de captura de moviment. Per una banda podria solucionar les configuracions singulars, proporcionant les coordenades correctes en els instants de conflicte. Al mateix temps, proporciona una feedback continu de les posicions, que permet verificar i corregir el model, així com la detecció dels possibles errors de mesura.

Finalment, per poder avaluar i comparar les posicions calculades amb el model, amb les dades obtingudes dels sensors òptics, és necessari crear una aplicació capaç d'obtenir i treballar amb aquestes dades. Sorgeix llavors, la necessitat de desenvolupar una interfície que connecte el programa que controla el robot amb el software dels sensors, per poder realitzar el control en temps real, de les coordenades del robot, amb la finalitat de millorar la qualitat dels assajos que es realitzen.

2.2. SISTEMES DE CAPTURA DE MOVIMENT

La captura de moviment, també coneguda com a MOCAP o *Motion Capture*, es tracta del procés de traslladar el moviment d'un cos del món real a un model digital del mateix, mitjançant una seqüència de coordinades cartesianes de l'espai 3D [13]. Actualment està en auge, i destaca per les seves aplicacions en el camp militar, esportiu, mèdic i sobretot, en el cinematogràfic i en el desenvolupament de videojocs.

En funció dels tipus de tecnologia que utilitzen, existeixen diferents sistemes per dur a terme captura de moviment. Aquests sistemes es poden dividir en dos grans grups: els òptics i no òptics. Dins de cada tipus, trobem a l'hora una nova subdivisió, com es presenta en la figura 11. Tot seguit, s'aprofundeix en cada sistema, i en els avantatges i inconvenients que presenta.

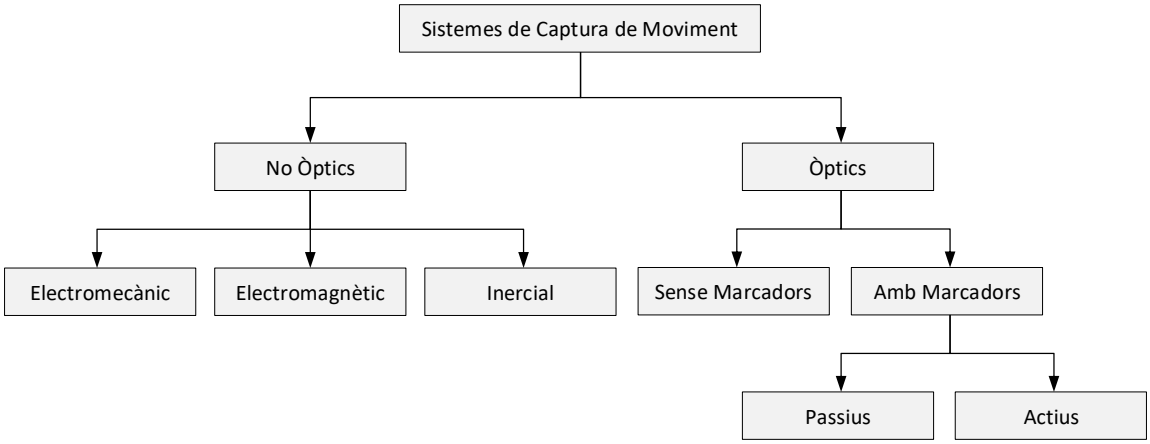

***Fig. 11*. Classificació dels sistemes de captura de moviment**

2.2.1. Sistemes no òptics

Els sistemes no òptics es caracteritzen per prescindir de l'ús de càmeres externes però necessitar un exoesquelet amb sensors, que habitualment és poc ergonòmic i limitant per a realitzar certs moviments. No obstant, els últims avanços han aconseguit millorar la lleugeresa d'aquests sistemes, i reduir la quantitat total de cables que posseeixen.

Aquests tipus presenten un cost menor que els òptics com a norma general, però també ofereixen una menor precisió que aquests. Una gran avantatge és la portabilitat que presenten, permetent així l'opció de poder realitzar estudis fora dels laboratoris [14]. La comunicació entre els sensors i receptor pot ser mitjançant un cablejat, o també utilitzant ones de ràdio per transferir la informació.

Atenent a la tecnologia en què es basen per dur a terme la captura de moviment, es subdivideixen en tres tipus diferents: electromecànics, electromagnètics i inercials. Tot seguit es presenten les característiques de cadascun, encara que de forma breu, perquè ningun d'aquests es tracta del sistema empleat en el projecte.

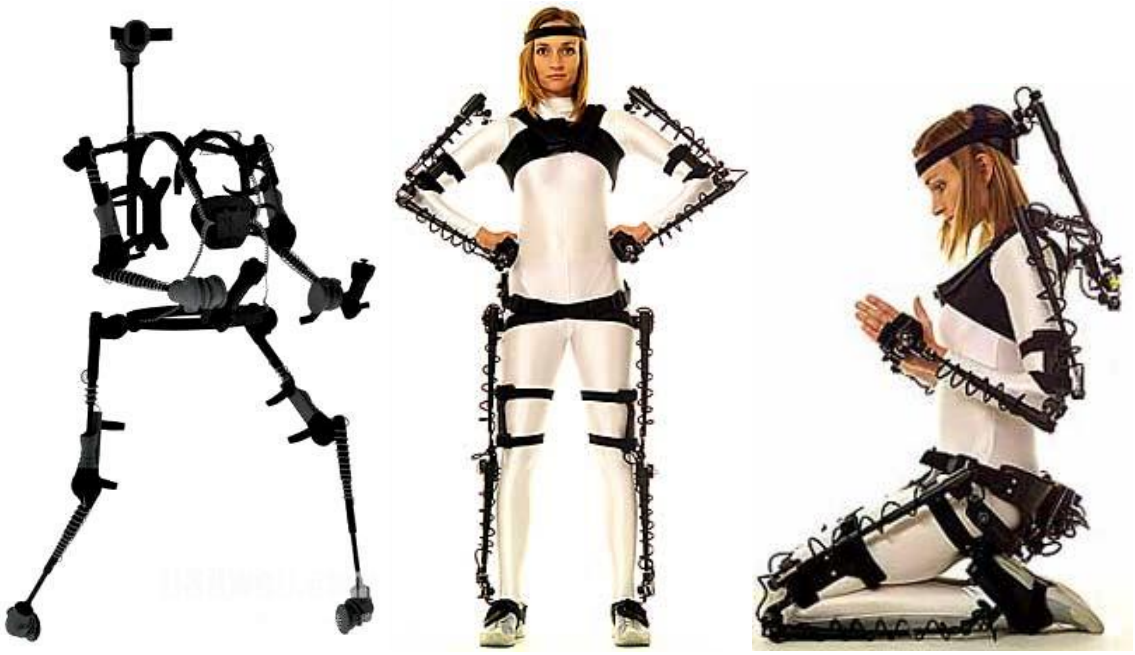

***Fig.12* . Exemple d'un sistema MOCAP electromecànic**

Nota. Adaptada de *Gypsy 7 Motion Capture Suit*, per META Motion (https://metamotion.com/gypsy/gypsy-motion-capture-system-workflow.htm).

- Sistema electromecànic: és tracta d'un exoesquelet de varetes rectes i articulades, unides amb potenciòmetres, els quals rastregen directament la variació dels angles entre les diferents varetes [15].

***Fig. 13*. Exemple d'un sistema MOCAP electromagnètic**

Nota. Adaptada de "Automatic Joint Parameter Estimation from Magnetic Motion Capture Data", per O'Brien J., Bodenheimer R., Brostow G., and Hodgins J., 2000, Proceedings of Graphics Interface (http://graphics.berkeley.edu/papers/Obrien-AJP-2000-05/).

- Sistema electromagnètic: són un conjunt de sensors, cadascun d'ells s'encarrega de mesurar el flux d'un camp electromagnètic generat, per tal de determinar la seua pròpia posició i orientació; existeixen tant de corrent continua com de corrent alterna [15].

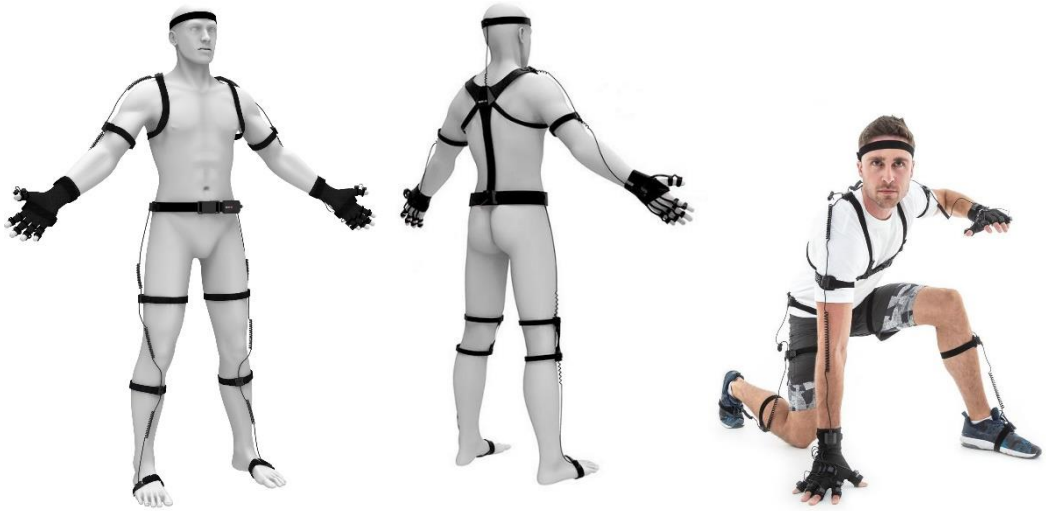

***Fig. 14*. Exemple d'un sistema MOCAP inercial**

Nota. Adaptada de *Full Body*, per Noitom, 2020, Perception Neuron (https://neuronmocap.com/file/355).

- Sistema inercial: es de creació relativament recent, i es basa en sensors inercials que combinen: giroscopis, magnetòmetres i acceleròmetres; els quals mesuren les tasses de rotació i capturen els sis graus de llibertat de l'espai tridimensional [14].

2.2.2. Sistemes òptics

Aquest altre tipus de sistema, utilitza les dades recollides per sensors òptics d'imatge per obtenir la posició d'un element en l'espai. Es tracta d'una o més càmeres sincronitzades que proporcionen projeccions simultànies del cos d'estudi [15]. Actualment el sistema permet l'obtenció en temps real del moviment amb algunes limitacions. Hi trobem dues modalitats, segons si s'usen o no marcadors pel posicionament dels elements.

2.2.2.1. Sense marcadors

Per una banda es troben els sistemes sense marcadors, que utilitzen uns tipus especials de càmeres capaces d'estimar la geometria 3D de l'element i els seus moviments. Un exemple d'aquest tipus és la càmera Microsoft Kinect, l'últim model d'aquest sistema al mercat s'anomena Azure Kinect. La figura 15 és un representació dels elements que el composen: un sensor de profunditat de llum infraroja amb 1 MP (1), una matriu de 7 micròfons (2), una càmera de vídeo RGB de 12 MP (3), un sensor inercial d'orientació, format per un acceleròmetre i un giroscopi (4), i diverses connexions per facilitar la sincronització (5) [16]. Les fortaleses d'aquest sistema són la facilitat d'ús i el reduït cost que presenta. No obstant, existeixen limitacions en la fiabilitat i precisió dels resultats, front l'alternativa amb marcadors, per als estudis més complexos [14].

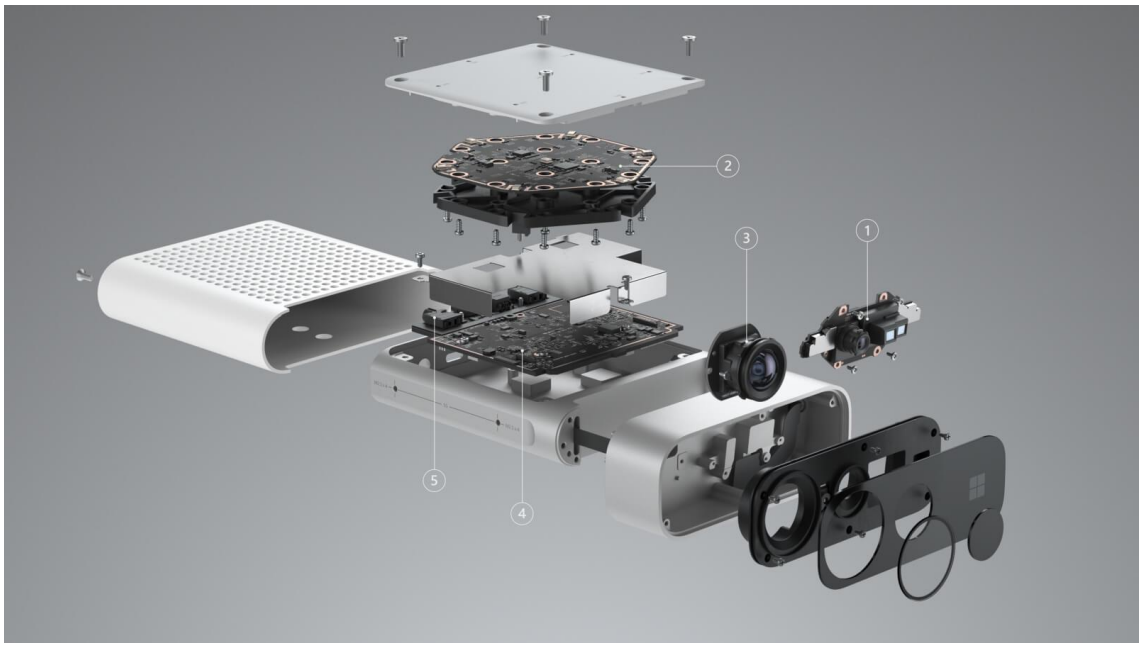

***Fig. 15*. Components de la càmera Azure Kinect de Microsoft**

Nota. Adaptada de *Contenido de Azure Kinect DK*, per Microsoft, 2020, Azure Kinect (https://azure.microsoft.com/es-es/services/kinect-dk/#industries).

2.2.2.2. Amb marcadors

L'altre tipus de sistema es basa en la captura d'imatges 2D usant diverses càmeres, i produint un model 3D per triangulació, dels marcadors situats sobre l'element d'estudi [14]. El sistema entrega les coordenades cartesianes (X, Y, Z) de cada marcador, necessitant un mínim de dues càmeres per fer-ho. D'altra banda, també és capaç de proporcionar l'orientació de qualsevol superfície formada per almenys tres marcadors [15]. D'aquesta forma, les dades dels diferents marcadors, obtingudes durant un temps determinat, permeten aconseguir una reproducció virtual del moviment efectuat pel cos.

A continuació, s'explica el procés per aconseguir la posició d'un sol marcador, tenint en compte que el procediment és anàleg per a la resta d'aquests. Primerament, s'obté de les imatges capturades per dues càmeres diferents, la posició 2D del marcador. Partint d'aquestes dues coordenades, una per cada càmera, s'obté la tercera coordenada per triangulació. Aquesta triangulació es basa en fer passar una recta, d'origen en el centre de cada càmera fins la posició 2D del marcador obtinguda. D'aquesta manera, el punt d'intersecció de les dues, constitueix la coordenada 3D del marcador en qüestió [13]. Aquest procediment es realitzat pel software cada instant, en la figura 16 es pot observar una representació d'aquest.

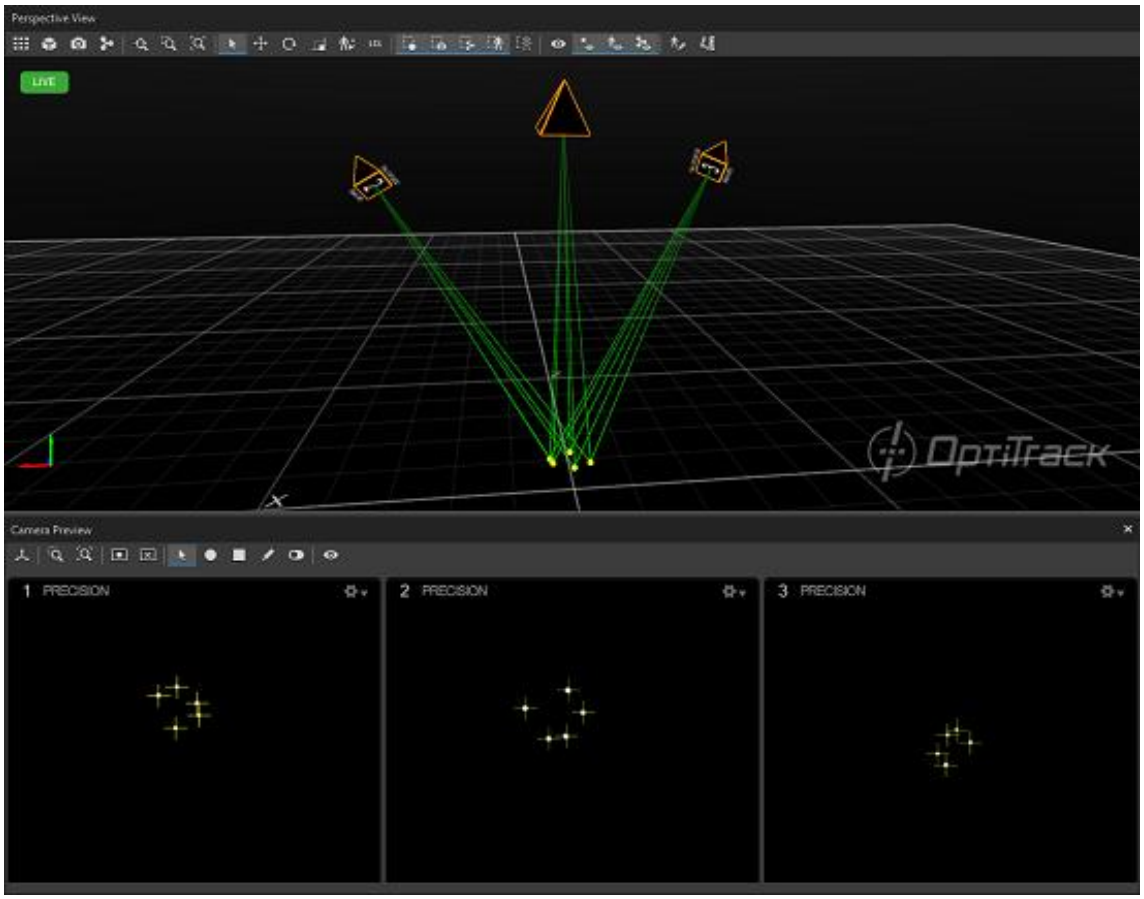

***Fig. 16*. Procés de reconstrucció 3D dels marcadors partint de les imatges 2D capturades amb el software Motive**

Nota. Adaptada de *3D Markers Reconstructed*, per NaturalPoint Inc., Optitrack Wiki Documentation (https://v22.wiki.optitrack.com/index.php?title=Reconstruction_and_2D_Mode).

Cal remarcar que es tracta d'una explicació simplificada i idealitzada de la triangulació realitzada pel software, perquè en la realitat el que ocorre es que degut al soroll captat per les càmeres, les rectes no arriben mai a intersecar-se [13].

Les principals característiques que fan destacar el sistema amb marcadors són l'alta precisió de les mesures i la possibilitat d'usar un gran nombre de marcadors, fent-lo l'adequat per a estudis d'alt grau de complexitat. Un exemple d'aquests estudis més complexos és la captura dels moviments facials, que requereixen de major nombre de marcadors, com es mostra en la figura 17. Dins de les limitacions que presenta es troba l'elevat cost d'adquisició i la necessitat d'haver de treballar en un espai controlat, lluny de la llum groga i sorolls reflexius, que pertorben les imatges capturades [17].

***Fig. 17*. Exemples de captura dels moviments facials amb marcadors passius**
Nota. Adaptada de *Visual Effects of "Star Wars: The Force Awakens"*, per Club Jade, 2016 (https://clubjade.net/a-look-at-the-force-awakens-motion-capture/).

El problema fonamental que presenta és l'oclusió, que consisteix en una interrupció i pèrdua de dades durant el seguiment de la posició d'algun marcador. La causa d'aquest fenomen és l'obstrucció de la línia de visió d'un o més marcadors quan es realitzen certs moviments que l'oculten [15].

La majoria de software de postprocessament està preparat, i compta diferents tècniques per d'interpretar i substituir el buit d'informació amb dades d'un marcador virtual amb la posició que teòricament li correspondria, sempre que el període d'oclusió no siga excessivament llarg [15]. L'inconvenient d'aquesta solució és que la majoria d'aquests mètodes no funcionen per a la captura en temps real, perquè el software requereix de cert temps per poder realitzar els càlculs. No obstant, en l'actualitat sí que existeixen alguns capaços d'aconseguir-ho. Principalment, es basen en establir restriccions constants de distàncies entre els marcadors, per poder aproximar així la distància entre ells en cas de que falten dades [18].

Una altra solució menys eficient és col·locar marcadors redundants, per poder compensar les possibles oclusions d'aquests. Actualment cap destacar que s'estan provant sistemes híbrids que combinen sensors inercials amb sensors òptics per tal de millorar la capacitat de rastreig i evitar així aquest problema.

Dins dels sistemes òptics amb marcadors, trobem una nova subdivisió: existeixen els sistemes actius i els passius, la diferència entre els dos resideix en el paper que tenen els marcadors. Si aquests són els elements actius i les càmeres passives, el sistema és actiu; mentre que si els marcadors són passius i les càmeres actives es tracta del passiu.

***Fig. 18*. Exemples de sistemes de MOCAP amb marcadors actius**
Nota. Adaptada de *WorleyWorks*, per PhaseSpace, 2013 (https://www.phasespace.com/clients_worleyworks.html).

En el sistema actiu, són els marcadors els encarregats d'emetre llum mitjançant diferents LEDs, i les càmeres sols registren tot el que els arriba. Per aconseguir la correcta identificació de cada marcador en cada instant, s'envien senyals als diferents LEDs, per aconseguir una sincronització precisa entre l'exposició de les càmeres i la il·luminació de cadascun [19]. Aquest sistema permet arenes de gravació de majors dimensions i disminueix la quantitat de soroll rebut, augmentant així la resolució final. No es va a profunditzar en aquesta tecnologia, perquè el cas d'interès no utilitza els marcadors actius.

***Fig. 19*. Exemple d'un sistema de MOCAP amb marcadors passius**
Nota. Adaptada de "FIFA 16 Introduces Female Footballers For The First Time", per Vikki Blake, 2015, *IGN* (https://sea.ign.com/fifa-16/89815/news/fifa-16-introduces-female-footballers-for-the-first-time).

Per al sistema passiu, els marcadors funcionen com a elements reflectors, a diferència de l'actiu, i en aquest cas són les càmeres les encarregades d'enviar i rebre, els rajos infrarojos. Per aquest motiu, aquests marcadors prescindeixen de cables o de qualsevol tipus de sistema electrònic. Es tracta del cas d'estudi, perquè és aquesta tecnologia la que utilitza el sistema OptiTrack instal·lat. En l'apartat següent s'expliquen amb detall tots els components que el formen.

2.3. ELEMENTS DEL SISTEMA OPTITRACK

Les captures de moviment que es duen a terme en un laboratori del departament d'Ingenieria de Sistemes i Automàtica de la Universitat Politècnica de València, i es realitza amb un sistema de marcadors passius de la companyia OptiTrack.

Els recursos necessaris per dur a terme la captura de moviment amb un sistema òptic amb marcadors són: els propis marcadors, les càmeres sincronitzades, el software de processament d'imatge i l'àrea de captura. A continuació, es comenten les característiques i funcions més importants de cada recurs, enfocat a les especificacions del sistema OptiTrack instal·lat al laboratori.

2.3.1. Marcadors

Els marcadors passius tenen forma esfèrica i estan recoberts d'un material retroreflectant, ja que són els encarregats de reflectir els rajos incidents que envien les càmeres. Els marcadors es col·loquen en els punts d'interès on es vol capturar el moviment.

Concretament, els que s'han col·locat sobre el robot es tracten d'un model de marcadors que s'uneixen a les bases mitjançant una rosca de mètrica M4. Aquestes bases, al mateix temps estan fixades al punt d'interès (la plataforma mòbil del robot paral·lel) mitjançant uns adhesius acrílics. L'esquema del muntatge d'aquest model de marcadors és el que es mostra en la figura següent:

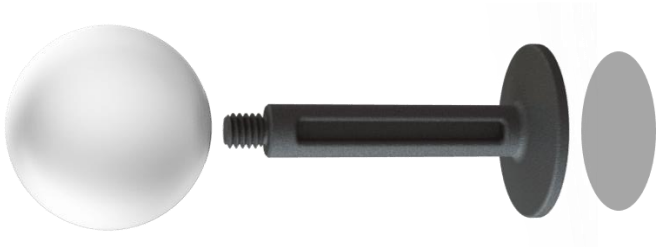

***Fig. 20*. Marcador passiu reflectant, base de marcador i adhesiu acrílic de la marca OptiTrack**
Nota. Adaptada de *Marker Configurator*, per NaturalPoint Inc., 2020 (https://www.optitrack.com/accessories/).

2.3.2. Càmeres

Les càmeres són sensors fotosensibles que capturen la llum incident. Per fer-ho, usen un matriu de cèl·lules fotoelèctriques, els píxels, i mesuren la intensitat en cadascuna d'aquestes cèl·lules [17].

Les matrius de píxels que poden entregar les càmeres varia de entre 128x128 en baixa resolució, fins a és de 4096x4096. Les càmeres que utilitza el sistema OptiTrack instal·lat són el model FLEX13, i compten amb un 1.3MPíxels, el que vol dir que cadascuna entrega una matriu de 1280x1024. Per obtenir una bona precisió en els resultats és fonamental la resolució i qualitat de les càmeres, ja que és el que ens donarà els detalls de la representació digital de la imatge.

Altre aspecte rellevant és la velocitat de gravació, perquè representa la quantitat de mostres o imatges que es prenen per segon. El model instal·lat és capaç de captar 120FPS (els FPS és la notació habitual que significa *Frames Per Second*), és a dir, que aquestes càmeres capten fins a 120 imatges per segon. Es tracta d'un paràmetre de rellevant importància en aquells casos on els cossos es menegen ràpidament, o si els moviments d'estudi són molt subtils.

***Fig. 21*. Càmera FLEX13 de la marca OptiTrack**
Nota. Adaptada de *Flex 13 Gallery*, per NaturalPoint Inc., 2020, OptiTrack Flex 13 (https://www.optitrack.com/products/flex-13/).

El nombre càmeres d'alta velocitat que solen tindre aquests sistemes sol oscil·lar entre un mínim de 4 i un màxim de 32 [17]. El sistema del laboratori compta actualment amb 10 càmeres, fixades de forma estratègica al voltant de l'àrea de gravació. Aquestes estan sincronitzades entre elles, i són les que defineixen el volum de captura on es podrà registrar el moviment dels elements.

Cada càmera d'aquest model compta amb 28 LEDs que emeten llum infraroja de longitud d'ona de 850nm. La finalitat és captar solament el reflex d'aquests sobre els marcadors, perquè el recobriment retroreflectant fa que tinguen un guany major a aquests rajos que la resta d'elements. A més, la majoria de càmeres sol requerir un graduació per definir un llindar d'intensitat de llum determinada. L'objectiu és que es centren solament en rebre la llum reflectida pels marcadors i ignoren la resta.

A més, aquestes càmeres permeten la gravació de forma paral·lela en escala de grisos. Aquesta característica serveix per poder validar les dades obtingudes, tant pel feedback instantani com a posteriori de la realització. Permet comparar el moviment digital obtingut, amb una representació visual de l'assaig real, com es pot observar en la següent figura:

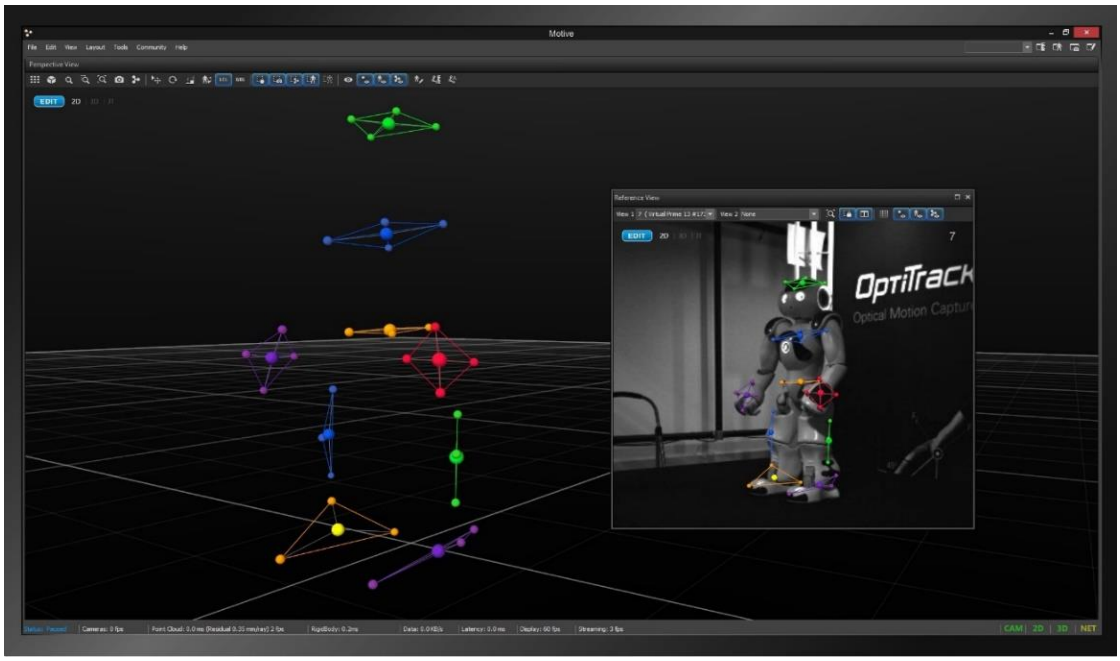

***Fig. 22*. Visualitzador del moviment amb la referència en grisos del software Motive**
Nota. Adaptada de *Screen Reference Video*, per NaturalPoint Inc., 2020, Motive:Tracker (https://www.optitrack.com/products/motive/tracker/indepth.html).

Periòdicament s'han de calibrar les càmeres, per obtenir la posició i orientació de cadascuna, perquè inicialment es desconeguda. La precisió en la calibració es deteriora de forma natural amb el temps, per factors ambientals com la temperatura, o en cas de que es produïsca algun canvi en la seua configuració [20].

***Fig. 23*. Vareta i quadrat de calibratge de la marca OptiTrack**
Nota. Adaptada de *Callibration Wand and Square*, per NaturalPoint Inc., 2020, OptiTrack Products (https://www.optitrack.com/products/tools/).

El procediment de calibració normalment es tracta de rastrejar un objecte de dimensions conegudes, com un quadrat o una vara amb diversos marcadors, com es mostra en la figura 23. Aquest element es mou per tota l'arena mentre les càmeres prenen captures, fins que arriben al nombre de mostres necessàries per realitzar els càlculs amb bona precisió. Posteriorment, combinant les vistes de totes les càmeres amb les dimensions de l'objecte conegut, es calcula la posició exacta de cada càmera en l'espai [17].

***Fig. 24*. Concentrador d'USB OptiHub de la marca OptiTrack**
Nota. Adaptada de *OptiHub Gallery Views*, per NaturalPoint Inc., 2020, OptiTrack Products (https://www.optitrack.com/products/optihub/).

Les càmeres es comuniquen mitjançant una connexió tipus USB, però no ho fan directament a l'ordinador, es connecten a un sistema concentrador d'USB anomenat OptiHub. És l'intermediari entre les càmeres i la computadora que controla el software, i s'encarrega d'agrupar totes les senyals. També ofereix suport per a la sincronització i adquisició per dispositius externs si fora el cas [21]. Cada OptiHub suporta com a màxim 6 càmeres, llavors com la configuració del sistema del laboratori compta amb 10, s'usen dues unitats. L'esquema de connexions queda de la forma que mostra la figura 25:

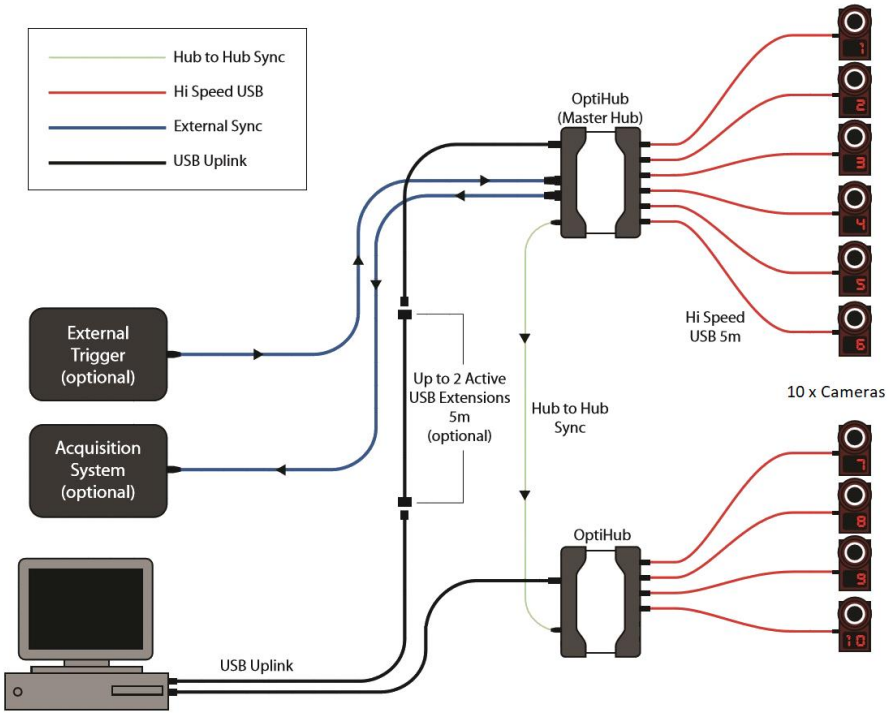

***Fig. 25*. Esquema de les connexions del sistema OptiTrack del laboratori**
Nota. Adaptada de OptiHub Quick Start Guide (p. 1), per NaturalPoint Inc. (https://www.optitrack.com/public/documents/OptiHub%20Quick%20Start%20Guide.pdf).

2.3.3. Àrea de captura

Les seues dimensions venen determinades pel nombre de càmeres i la col·locació d'aquestes, generalment a major nombre de càmeres, major dimensió. En el cas d'aquest sistema amb 10 càmeres, l'arena de gravació que es té al laboratori és, aproximadament, de $10m^2$.

Altre aspecte a destacar, és que amb l'objectiu de millorar l'eliminació del fons de les imatges captades, la zona està rodejada per cortines negres opaques i el sòl està tapat amb moqueta. Així, s'aconsegueix uniformitzar la il·luminació de l'escena per tal de reduir al màxim els enlluernaments i les ombres que es puguen produir.

2.3.4. Software

L'últim component clau és el software de processament de dades, el que utilitza el sistema Optitrack instal·lat rep el nom de Motive:Tracker. És l'element del sistema que controla totes les càmeres, rep totes les seues imatges i s'encarrega del tractament de les dades per identificar les posicions. A la següent figura es pot observar la interfície general:

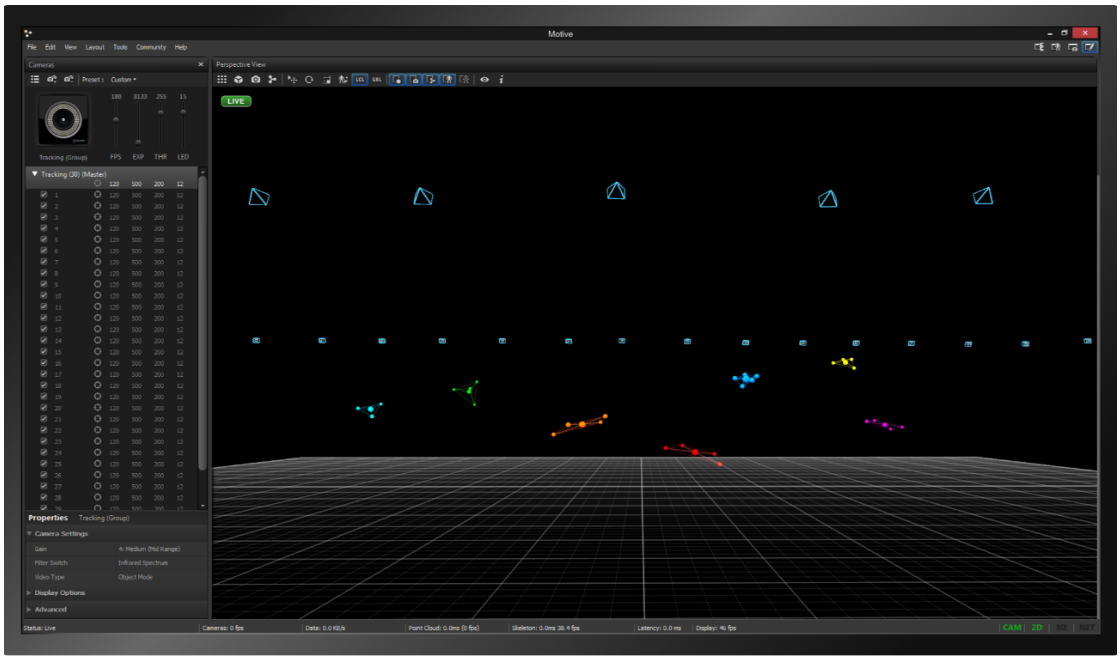

***Fig. 26*. Interfície del visualitzador del software Motive:Tracker**
Nota. Adaptada de *Tracker Screen*, per NaturalPoint Inc., 2020, Motive:Tracker (https://www.optitrack.com/products/motive/tracker/indepth.html).

Dins de les seues funcions es troba la visualització dels cossos rígids i marcadors durant l'assaig i a posteriori de la realització. És també l'encarregat de definir el nombre, nom i tipus d'objectes rígids, entre altres aspectes de personalització, per adequar els resultats al nostre assaig particular. El software també permet editar i millorar la gravació eliminant parts que no interessen, limitant el valor màxim dels bots en la posició o reassignant marcadors a un set, en cas de que es desconfiguren, entre altres opcions. En la figura 27, es mostra l'aspecte de la interfície d'edició del Motive.

A més, compta amb la possibilitat d'exportar totes les dades generades en format CSV, una extensió de Microsoft Excel, i C3D, un tipus d'arxiu especial per al Motive i compatible amb la majoria de software de captura de moviment.

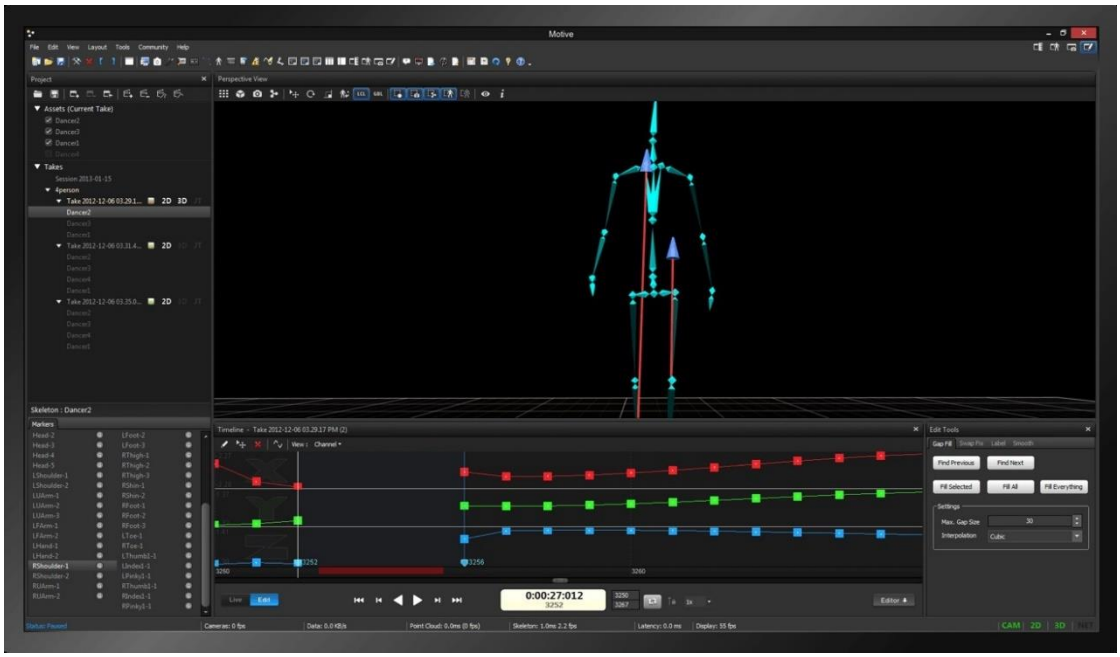

***Fig. 27*. Interfície del editor de dades del software Motive**

Nota. Adaptada de *Data Management*, per NaturalPoint Inc., 2020, OptiTrack Movement Sciences (https://optitrack.com/motion-capture-movement-sciences/).

No obstant, la missió principal és calcular la posició dels cossos en tot moment partint de les imatges obtingudes. En l'apartat de captura de moviment, s'ha exposat en línies generals el procediment de triangulació per obtenir les coordenades 3D d'un marcador. Seguidament, es torna a abordar el procés d'identificació de la posició amb una perspectiva més centrada en el propi funcionament del software.

El procés comença per obtenir una representació clara dels marcadors a partir de cada imatge, com està representat en la figura 28. S'aconsegueix minimitzant el soroll i aïllant de l'ambient als marcadors, eliminant els píxels que sobrepassen o no arriben al llindar (o *threshold*) lumínic predeterminat [17].

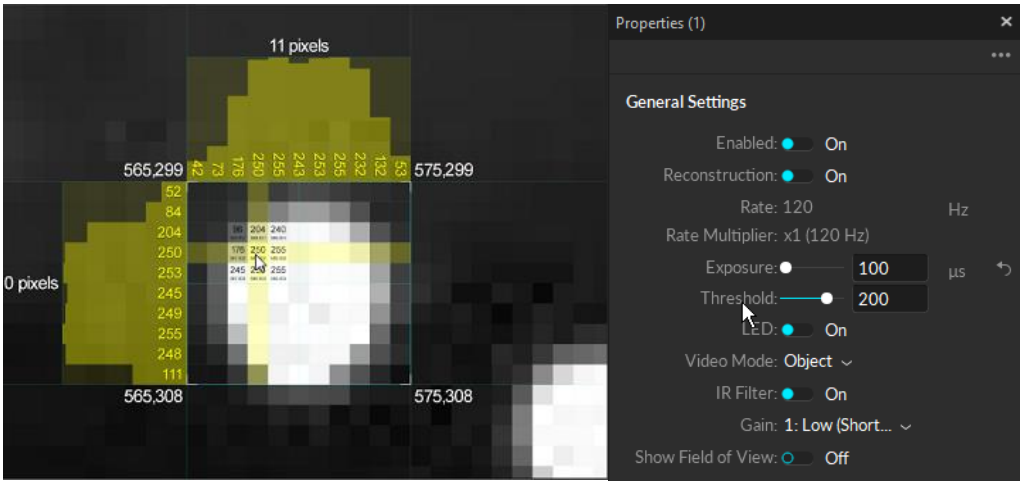

***Fig. 28*. Detecció del marcador amb Motive, usant el llindar lumínic predefinit**

Nota. Adaptada de *Analyzing píxel brightness*, per NaturalPoint Inc., Optitrack Wiki Documentation (https://v22.wiki.optitrack.com/index.php?title=Reconstruction_and_2D_Mode).

El següent pas és la detecció de cada marcador, determinant les seues coordinades 2D per cada imatge. El software s'encarrega de resoldre la correspondència espacial, identificant el mateix marcador en les diferents imatges, presses el mateix instant de temps, però amb diferents perspectives [13].

Posteriorment, s'aplica la triangulació entre els diferents sets de posicions 2D, un set per cada càmera. D'aquesta forma es combinen i formen les coordenades 3D de cada marcador, responent a una base de restriccions de col·linealitat, com es mostra en la següent figura:

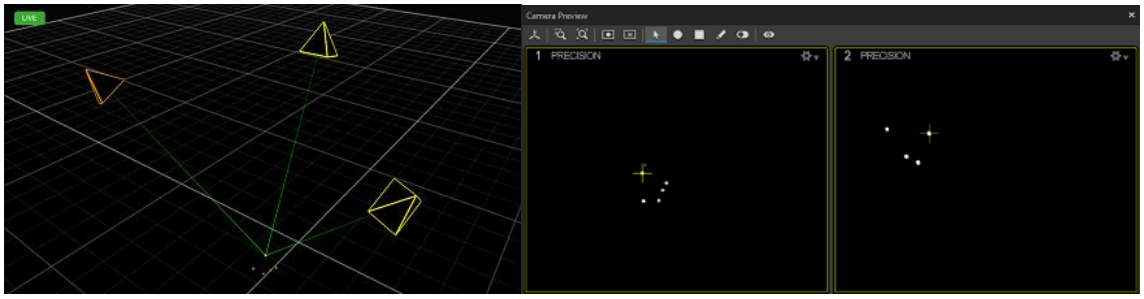

***Fig. 29*. Obtenció de la posició 3D d'un marcador partint de les imatges 2D amb el Motive**
Nota. Adaptada de *Tracked Rays*, per NaturalPoint Inc., Optitrack Wiki Documentation (https://v22.wiki.optitrack.com/index.php?title=Reconstruction_and_2D_Mode).

Per últim, es resol la correspondència temporal, es rastreja cada marcador a llarg de les diverses imatges presses. D'aquesta manera es relacionen els diferents núvols o sets de representacions 3D dels marcadors, generats a partir de diferents captures [13].

En l'última etapa, es genera una trajectòria completa de cada marcador, com a resultat de concatenar en temps les seues diferents coordinades, permetent així al software poder aconseguir una reproducció virtual del moviment efectuat pel cos en la realitat. En aquest punt si perd el rastre d'algun marcador degut a l'oclusió, s'intenta resoldre les coordinades que hauria de tindre basant-se en les dades dels fotogrames adjacents [17].

D'aquesta manera, les dades entregades són directament uns vectors de coordenades cartesianes dels diferents marcadors, per cada instant de temps mostrejat. Si ens interessa obtenir l'orientació d'una superfície, el programa requereix la posició relativa d'almenys tres marcadors per poder fer-ho. Aquestes dades resultants es poden post processar amb diferents tipus de software.

La principal limitació que trobem del Motive:Tracker és que es tracta d'un software tancat, el que significa que no permet ningun tipus de modificacions en les seues funcions. Aquesta característica fa que siga impossible adaptar-lo a qualsevol necessitat que vaja més enllà de visualitzar, editar o exportar les dades obtingudes en un format determinat.

D'altra banda, sí que és un software flexible a l'hora de transmetre les dades en temps real, disposa de diversos mètodes per fer-ho. Alguns d'aquests són el VRPN o el Trackd, que s'usen per interaccionar amb els dispositius de realitat virtual. Altre mètode de transmissió és mitjançant el NatNet SDK, el qual resulta ser l'opció més interesant per a la poder accedir a les dades de forma externa, com es veurà a continuació.

2.3.4.1. NatNet SDK

El NatNet SDK és, com el seu nom indica, d'un *Software Development Kit* creat per NaturalPoint. Es tracta d'un conjunt de ferramentes que permeten integrar en noves aplicacions personalitzades, les dades de seguiment en temps real que proporciona el sistema OptiTrack.

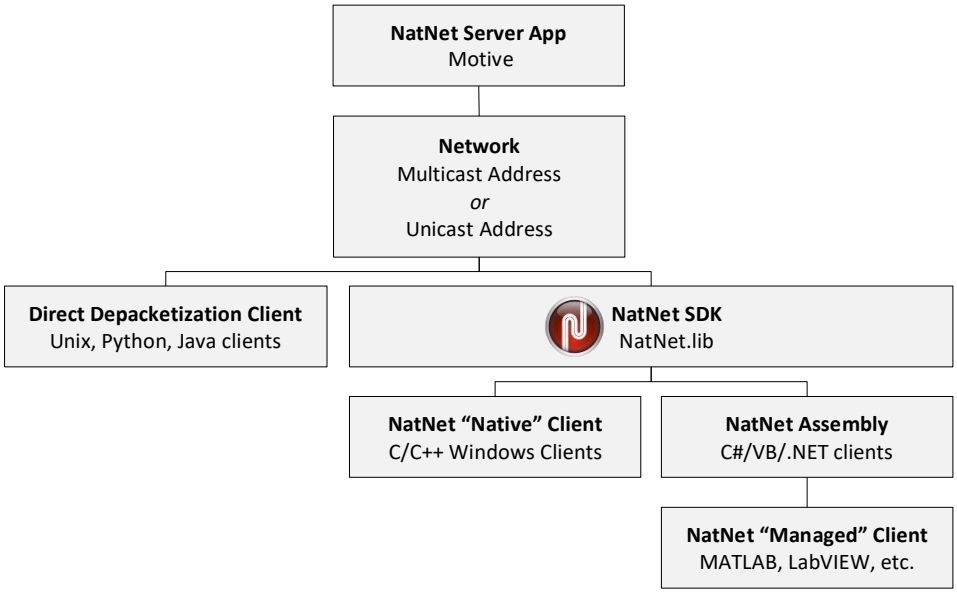

***Fig. 30*. Esquema general dels elements de NatNet**
Nota. Adaptada de *NatNet API User's Guide* (p. 3), per Naturalpoint Inc., 2016 (https://www.optitrack.com/public/documents/natnet-api-user-guide-2.10.0.pdf).

Com es pot observar en la figura 30, la transmissió de dades en temps real es basa en una arquitectura on el software Motive exerceix de servidor, i una o més aplicacions externes actuen com a clients d'aquest. Per realitzar la connexió entre les computadores es necessita que estiguen connectades a la mateixa xarxa d'Internet, per exemple, mitjançant Wi-Fi o Ethernet. Per transmetre la informació, utilitza un protocol anomenat UDP que s'explica a l'apartat del TCP/IP (punt 2.5.3.). A més, la comunicació pot ser unicast, on cada ordinador té una direcció IP, o multicast, on una sola IP té assignada un grup de computadores [22].

Posteriorment, per accedir a la informació que proporciona el servidor es pot realitzar de forma directa, mitjançant llenguatges com Unix, Phyton o Java entre altres; o usant el NatNet SDK. Utilitzant aquest últim mètode, es poden llegir les dades directament programant amb les funcions que ofereix el kit de desenvolupament de software amb C o C++. Alternativament, també es pot accedir a les mateixes funcions a través de l'assemblatge o *assembly* amb programes com MATLAB o LabVIEW, com en el cas d'estudi.

El NatNet SDK està format, per una banda, per llibreries que són bàsicament recursos per al desenvolupament de programes com: parts de codi, subrutines o dades, entre altres. També inclou l'assemblatge NatNetML.dll, que es tracta d'un recurs molt similar a les llibreries, però que usa la tecnologia .NET. Per últim, també conté alguns exemples i executables, que permeten verificar la connexió amb el Motive, i comprovar que s'estan rebent les dades correctament [22].

En síntesi, el NatNet SDK proporciona la possibilitat de crear un programa personalitzat que accedisca, de forma remota i en temps real, a tota la informació sobre el moviment proporcionat pel Motive. Aquests són els motius que el fan òptim per usar-lo en la aplicació de projecte, i així fer possible la connexió amb el software de l'OptiTrack.

2.4. INTERFÍCIE D'USUARI

La interfície d'usuari de qualsevol programa és la via de comunicació entre l'usuari i les diferents funcions que aquest ofereix [23]. Normalment està formada per diversos elements interactius com pestanyes, quadres de diàleg, taules, botons o altres elements gràfics que faciliten l'ús i el control de la aplicació.

Primerament, es considera que una interfície està ben dissenyada si, en línies generals, es comporta de la mateixa manera que l'usuari espera que ho faça [24]. Una bona interfície no sols ha de ser capaç de cobrir tots els requisits funcionals del programa, també ha de ser intuïtiva, lògica i amigable, per facilitar a l'usuari aconseguir els seus objectius de la forma més fàcil i efectiva possible. Un exemple del que es pot considerar una bona interfície, es mostra en la següent figura:

***Fig. 31*. Exemple d'un bon disseny: la interfície de Windows 8**

Nota. Adaptada de *Windows 8 Start Screen*, per CS Odessa Corp., 2020 (https://www.conceptdraw.com/samples/software-windows-8-user-interface).

D'altra banda, es considera que una interfície està mal dissenyada si està sobrecarregada d'elements innecessaris, posseeix una organització il·lògica o és excessivament difícil d'entendre. Aquest tipus de disseny pot ser responsable d'un major cost d'aprenentatge, augment dels errors comesos, generar frustració o una pèrdua de productivitat; degut a la concentració addicional requerida per poder utilitzar-la [23]. Una representació del que seria una interfície mal dissenyada, és la de l'exemple de la figura 32.

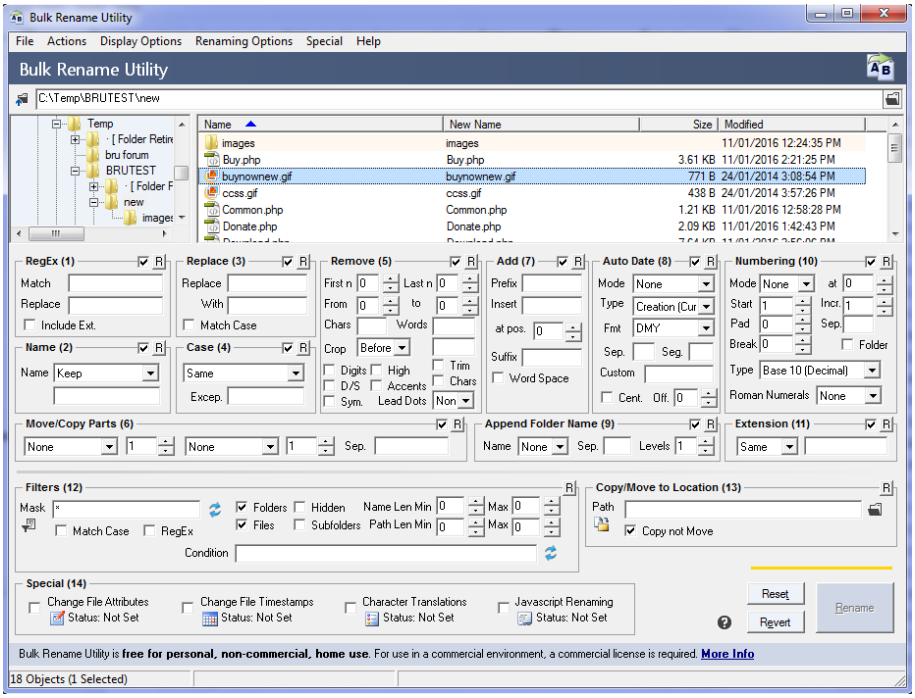

***Fig. 32*. Exemple d'un mal disseny: la interfície de Bulk Rename Utility**

Nota. Adaptada de *Main Screen*, per TGRMN Software, 2020 (https://www.bulkrenameutility.co.uk/).

En definitiva, la interfície d'usuari es tracta d'un element vital de les aplicacions, perquè connecta a l'usuari amb totes les funcions que té a l'abast. El disseny d'aquesta és fonamental, perquè en la majoria de casos, és la responsable de generar satisfacció, o pel contrari estrès, durant l'ús del programa. Per aquest motiu, es busca que les interfícies tinguen un disseny funcional, atractiu i amigable.

2.5. LABVIEW

LabVIEW és l'acrònim de *Laboratory Virtual Instrument Engineering Workbench*, es tracta al mateix temps d'un llenguatge i un entorn de programació desenvolupat per la companyia National Instruments [25]. Actualment, és àmpliament utilitzat en l'àmbit de l'enginyeria de sistemes per les seues característiques i el gran ventall de possibilitats que ofereix.

Principalment ha estat desenvolupat per a l'adquisició de dades, el condicionament de senyals, l'automatització, el control d'instruments i l'anàlisi de mesures; així com la representació d'aquestes. Utilitza un tipus de programació gràfica d'alt nivell, anomenat llenguatge G, que en compte de estar basat en text com la majoria, el codi que es programa s'expressa mitjançant diagrames de blocs [26].

L'element o arxiu que genera aquest software s'anomena *Virtual Instrument* o VI, que consisteix en: el *Block Diagram*, on es programa el codi, el *Front Panel*, que mostra l'aspecte de la interfície de l'aplicació, i la icona que representa visualment el VI i les seus entrades i eixides. En la figura 33, es mostra l'aspecte del *Front Panel* i el *Block Diagram* en un programa exemple:

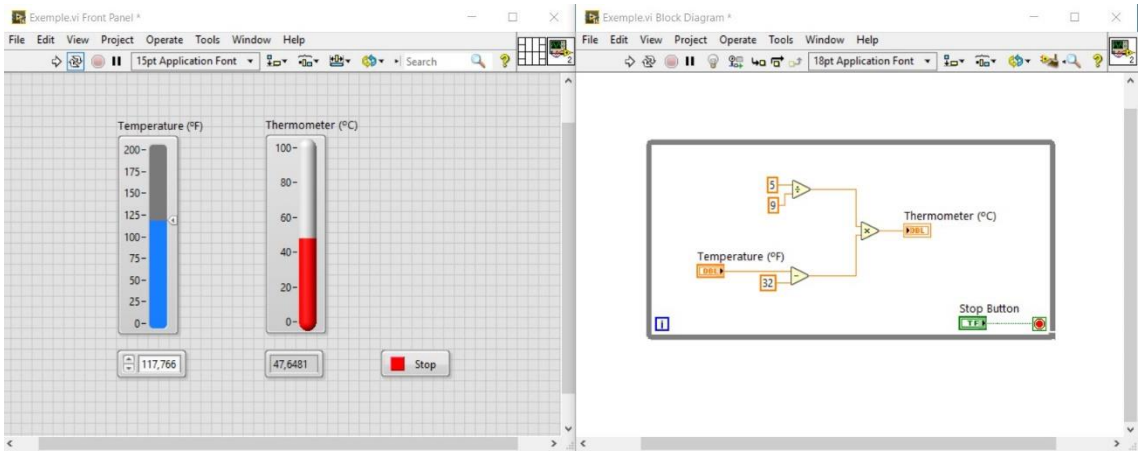

***Fig. 33. Front Panel* (esquerra) i *Block Diagram* (dreta) del LabVIEW**

Altres llenguatges de programació, com C o BASIC, utilitzen funcions i subrutines com a elements de programació; LabVIEW utilitza blocs i els propis VI, perquè aquests es poden usar tant com a interfície de comunicació amb l'usuari, com a funció o subrutina dins d'un altre VI [26].

La programació visual que ofereix és un dels principals avantatges que presenta, perquè simplifica i facilita enormement la creació i comprensió del codi. Les funcions i les seues connexions varien de forma i color segons el tipus de dades que manipulen. Permetent així que programadors inexperts tinguen la possibilitat de crear projectes relativament complexos.

No obstant, també trobem el LabVIEW en l'àmbit professional degut a que ofereix programar aplicacions molt elaborades amb milers de nodes i VIs, cadascun dels quals pot treballar amb un gran nombre d'entrades i eixides de dades.

Una altra fortalesa a remarcar és la rapidesa a l'hora de programar amb LabVIEW front a altres llenguatges. Gràcies a la gran quantitat de funcions o blocs predissenyats que ofereix, aconsegueix estalviar tot el temps que requeriria programar-les des de zero.

A més, compta en el *Front Panel* amb indicadors, controls, menús o gràfics entre moltes altres opcions de personalització de la interfície. D'aquesta manera permet millorar la interacció amb l'usuari final, amb la finalitat de fer l'aplicació més intuïtiva i senzilla d'usar.

Finalment, destacar la gran capacitat d'interacció amb diferents llenguatges i aplicacions, com DLL, .NET, Multisim, AutoCAD o MATLAB, entre altres. Addicionalment, proporciona suport específic per a totes les plataformes de hardware de NI, com la CompactRIO o el CompactDAQ. A banda, disposa de milers de controladors de dispositius a l'abast, i una gran quantitat de biblioteques, oferint així multitud d'habilitats i funcions especialitzades per a cada tipus determinat d'aplicació.

Per tots aquets motius, LabVIEW és el programa que s'ha elegit per desenvolupar l'aplicació del projecte, destacant l'accessibilitat i facilitat a l'hora de desenvolupar nous programes. També degut a la multitud de possibilitats que ofereix per interconnectar-se amb altres programes,

ordinadors o instruments; característica fonamental i necessària en el disseny de la interfície. A continuació, s'aprofundeix especialment en les tecnologies .NET, ActiveX i TCP/IP compatibles amb LabVIEW, perquè són elements claus en el desenvolupament de la connectivitat del projecte.

2.5.1. .NET

Aquesta tecnologia es tracta d'una plataforma de desenvolupament de software que s'utilitza per connectar informació, sistemes o dispositius mitjançant l'ús dels serveis Web [26]. La tecnologia .NET ha estat creada per Microsoft, i està basada en una programació totalment orientada a objectes. Es caracteritza per poder accedir, a través de la Internet, al codi de multitud d'aplicacions, independentment del sistema operatiu o del llenguatge de programació. Tot seguit, es mostra en la figura 34 un esquema de l'operativitat de .NET, que s'explicarà a continuació:

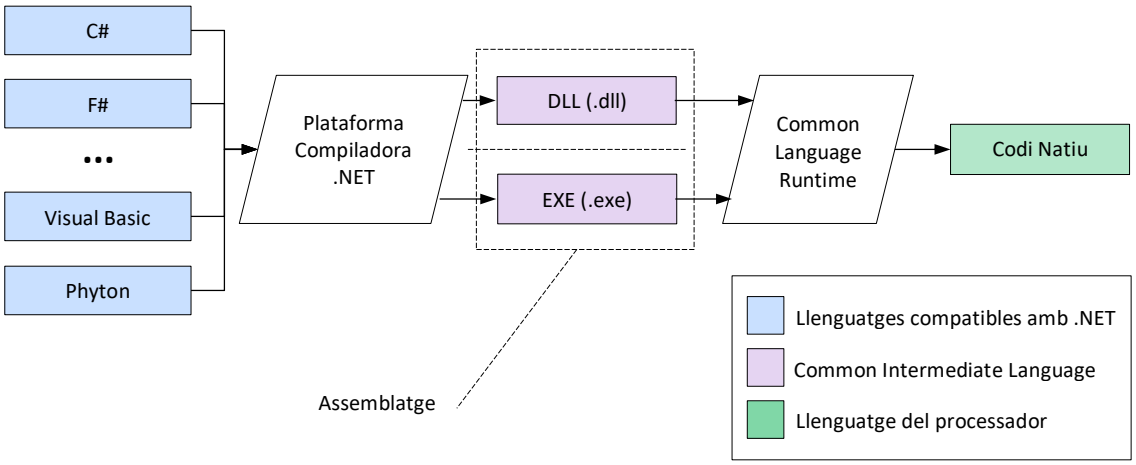

***Fig.34*. Esquema del funcionament de la tecnologia .NET**

El funcionament d'aquesta tecnologia comença amb un programa escrit en qualsevol dels llenguatges compatible amb .NET, com C#, F#, C++, Visual Basic o Phyton entre molts altres. Quan aquest codi es compila, es genera un assemblatge o *assembly*, que es tracta d'un arxiu que pot ser una llibreria (.dll) o un executable (.exe), segons com s'haja especificat al compilador. Aquest assemblatge és un mòdul composat per un nou codi i metadades. El nou codi de l'assemblatge, està escrit en *Common Intermediate Language* o CIL, que és un llenguatge de programació totalment independent de la plataforma, molt paregut al de tipus assemblador. Per altra banda estan les metadades, que contenen informació com la versió, els permisos, les dependències o els recursos de l'aplicació. [25]

Per tal d'executar o accedir al programa, s'ha de tornar compilar l'assemblatge per transformar-lo del CIL al llenguatge natiu del processador. Aquest pas el realitza el *Common Language Runtime* o CLR, que es tracta d'una màquina virtual que s'encarrega d'interpretar el llenguatge intermedi CIL i traduir-lo a ordres concretes de la plataforma de treball. [25]

LabVIEW permet accedir al *Global Assembly Cache* o GAC, que es tracta d'un registre controlat de tots els assemblatges disponibles dins del sistema operatiu [26]. D'aquest manera, carregant l'assemblatge desitjat del GAC, té la possibilitat d'accedir de forma remota a les propietats i els mètodes dels objectes d'infinitat de programes. Per connectar les aplicacions s'usen protocols

estàndards d'Internet, on LabVIEW actua com a client, i la resta de programes com a servidors. Per poder realitzar aquesta comunicació es necessari conèixer, per part del client, la direcció IP de l'ordinador o dispositiu que actue com a servidor, per a identificar on s'ha de connectar.

Per al projecte, aquesta tecnologia s'usa per accedir a les funcions i als atributs de cada tipus d'objecte dels que disposa Motive, permetent controlar certs mètodes i llegir les dades necessàries, gràcies a carregar en el LabVIEW la llibreria NatNet.dll. La tecnologia .NET, cal destacar que és la que fa possible la comunicació entre el sistema OptiTrack i l'aplicació de LabVIEW, permetent que puguen treballar de forma conjunta.

2.5.2. ActiveX

És una tecnologia que es tracta d'un entorn de programació, creat amb la finalitat de poder compartir recursos, basats en objectes, entre diferents aplicacions [25]. ActiveX permet la comunicació entre alguns programes concrets, oferint la possibilitat de controlar diversos components o serveis que aquests posseeixen. D'aquesta manera, es pot reutilitzar el codi d'alguns objectes ja programats, que són mòduls funcionals independents, que poden ser embeguts dins d'altres aplicacions.

Entre les tecnologies ActiveX i .NET existeixen moltes característiques en comú, com haver estat desenvolupades per Microsoft o estar basades en la programació orientada a objectes. Ambdues comparteixen també l'habilitat d'accedir (com a clients) als mètodes i propietats de diferents programes (que actuen com a servidors) compartint els seus components.

No obstant, presenten diverses diferències, una d'elles és el fet de que ActiveX no requereix d'Internet per compartir els components entre les aplicacions. El motiu és que aquesta tecnologia sols treballa de forma local, per tant sols disposa dels recursos disponibles al propi sistema, no està dissenyada per connectar-se amb altres ordinadors o dispositius externs, com sí ho estava .NET. Aquest fet representa un factor molt limitant de l'ActiveX, perquè, encara que no tot, la gran majoria sols funciona amb un sistema operatiu Windows de 32 bits [25].

El principal avantatge d'aquesta tecnologia és la reducció en temps de desenvolupament de noves aplicacions. El seu ús estalvia haver d'escriure des de zero nous codis, o haver d'aprendre nous llenguatges per crear controls concrets [26]. Un clar exemple del tipus de mòduls que es poden compartir seria la "Revisió ortogràfica i gramatical". Aquest objecte s'utilitza en el Microsoft Word, PowerPoint i Outlook, i en totes funciona de forma idèntica. Seria ineficient haver de definir-lo en cada aplicació, mitjançant la implementació d'aquesta tecnologia el problema es soluciona.

El primer requeriment per poder utilitzar un control ActiveX es que ha d'estar registrat en el sistema. Quan s'instal·len aplicacions com Word, Excel o qualsevol compatible, tots els objectes o mòduls disponibles per al control queden registrats [26]. És molt similar al que ocorria amb el GAC i els assemblatges del .NET, el sistema coneix tots els ítems disponibles per a ser usats.

El procediment per a la implementació es basa en col·locar, dins de la nova aplicació, un contenidor on el component a controlar estarà embegut. Aquest contenidor, conté la referència del nom on l'objecte del programa està emmagatzemat [26]. D'aquesta manera la nova aplicació accedeix directament al control del mòdul des de la seua pròpia interfície, no s'ha d'obrir l'altre programa de forma explícita per usar eixe control concret. Usant els contenidors obtenim una integració total d'aquelles funcions més útils dins del nou programa. Aquest funcionament permet unir el mateix mòdul a diverses aplicacions.

Un altre exemple de l'ús de l'ActiveX, el trobem en el navegador Internet Explorer. Quan en una pàgina Web apareix un vídeo, l'aplicació crida al Windows Media Player per a que el reproduïsca; o de forma anàloga amb el Adobe Flash Player, per a la visualització d'un PDF al propi navegador [26]. En aquest cas, el programa Internet Explorer actua com a client dels servidors Media Player o Flash Player.

La tecnologia ActiveX li permet a LabVIEW actuar a la vegada com a servidor o client, és a dir, pot accedir a diferents components externs, al mateix temps que altres aplicacions accedeixen a certs mòduls seus. No ocorria el mateix amb .NET, perquè aquesta sols permetia que LabVIEW fora client d'altres aplicacions, no podia exercir ell mateix com a servidor.

En l'aplicació del projecte, ActiveX ha permès utilitzar tecnologia del sistema Windows per la visualització dels gràfics. El seu ús ha sigut indispensable degut a què les opcions de graficar disponibles al LabVIEW no satisfeien els requeriments mínims per a la correcta representació dels diferents sòlids rígids, i dels seus marcadors.

2.5.3. TCP/IP

El TCP/IP consisteix en un parell de protocols que estableixen una comunicació entre computadores mitjançant la xarxa d'Internet. Les seues sigles es corresponen a *Transmission Control Protocol* i a *Internet Protocol*. Es va desenvolupar el 1972, pels investigadors del departament de defensa dels EEUU. Representa la base del que actualment coneixem com a Internet [25].

La connexió TCP/IP és, bàsicament, la creació d'un canal de comunicació directa entre un ordinador emissor i un altre receptor. L'arquitectura d'aquesta connexió està formada per diferents nivells o capes, el que significa que les dades es van encapsulant a mesura que travessen els diferents nivells. Per cada nivell que passa, al missatge que s'envia, se li afegeix una capçalera d'informació. La finalitat és poder proporcionar certs serveis, com comprovar l'existència d'errors, o encaminar les dades al destinatari correcte. El mateix procés que realitzen les dades en el servidor, el realitzaran en l'ordre contrari al arribar al client. Aquest procediment d'intercanvi d'informació es pot observar a la figura 35:

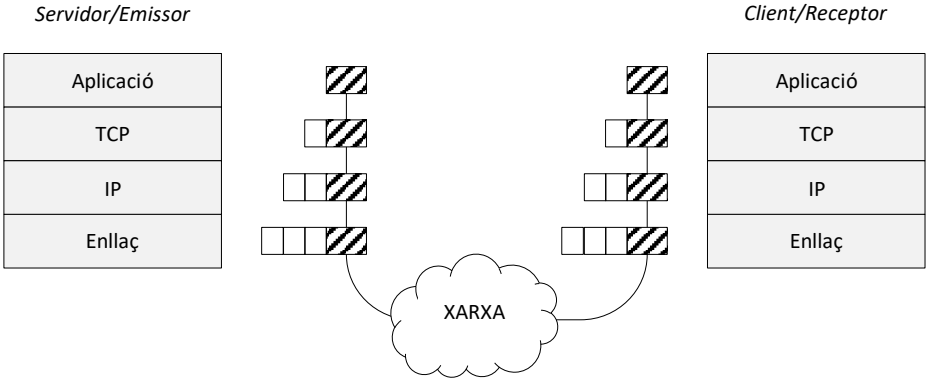

Fig. 35. **Esquema funcionament protocols TCP/IP**

Nota. Adaptada de *LabVIEW: entorno gráfico de programación* (p. 188), per Lajara Vizcaino, J.R. i Pelegrí Sebastiá, J., 2017, Marcombo.

La capa o nivell inferior correspon a la de transport, que el realitza el protocol TCP. Aquest s'encarrega de dividir el missatge que s'ha de enviar en diferents paquets, a la vegada que detecta possibles errors en l'enviament. Aquestes funcions les acompleix gràcies a la informació afegida en la capçalera. Aquesta capçalera, per una part conté dades sobre l'ordre o la posició de cada paquet dins del missatge, i per altra banda un codi especial per verificar que no hi hagen errors. D'aquesta manera, quan els missatge arriba al receptor, el protocol TCP es capaç d'ordenar correctament els diferents paquets de dades. Al mateix temps, valida si el missatge és correcte, i si es dona el cas que de que no ho és, sol·licita al servidor que torne a enviar el paquet [27].

El nivell superior correspon al nivell de ret, i el realitza l'IP. Aquest protocol és l'encarregat de definir la mesura que tindran els diferents paquets, i d'encaminar-los fins la direcció IP del destinatari. Per aquest motiu el protocol afegeix en les dades de la capçalera la mesura i l'adreça IP del receptor [28].

Per últim, es troba el nivell d'enllaç. La capçalera que afegeix conté informació sobre el tipus de connexió a la xarxa de l'ordinador en qüestió, indicant si aquesta és a través de Wi-Fi o d'Ethernet, entre altres tipus [28].

Per poder realitzar una connexió d'aquest tipus, l'ordinador receptor o client, ha de conèixer prèviament la direcció IP de l'emissor o servidor de dades. Aquesta adreça és única per a cada ordinador, és tracta d'un nombre enter de 32 bits, normalment representat per notació decimal de 8 bits separats per punts [26]. Aquesta part és anàloga al que ocorria per establir una connexió amb la tecnologia .NET, no obstant, ara no sols es necessita la IP, també es requereix conèixer el port virtual i la quantitat de bits que posseeix el missatge a rebre.

Altra característica del protocol TCP és que no comença a manar dades al client o receptor fins que la connexió no s'ha validat. En aquest tipus de connexió, primer és el servidor el que es manté a l'espera, fa de *listener* o oient, fins trobar algun ordinador que es connecte a la seua adreça IP i al mateix port. Per fer-ho, ambdues parts es manen un missatge indicant que estan preparades per enviar/rebre informació.

El motiu d'aquest procediment és que la xarxa d'Internet no és un mitjà de comunicació segur, a causa de que no s'assegura que la informació manada arribe al seu destinatari de forma correcta, ja que es pot perdre o arribar de forma desordenada al receptor [29]. No obstant, gràcies a aquest procediment, i als comentats anteriorment, el protocol TCP/IP sí que és capaç de crear un canal fiable de connexió.

En tot moment s'ha fet referència a una connexió entre un únic servidor i un únic client perquè es l'arquitectura típica de les aplicacions, però no l'única. És important comentar que també existeix un cas especial anomenat la multicast o multidifusió. En aquest tipus de connexió, les direccions IP no fan referència a un sol ordinador, sinó a un conjunt d'aquests.

Finalment, és rellevant destacar que existeix un altre protocol del nivell de transport, alternatiu al TCP, anomenat UDP, que és el que utilitza el software Motive per retransmetre les dades. Les sigles són de *User Datagram Protocol*, i es tracta d'una simplificació al màxim del TCP prèviament definit. Aquest no inclou informació a capçaleres, llavors no pot assegurar el correcte enviament del paquet [29]. L'avantatge que presenta es que al tractar-se d'un protocol més senzill, és més ràpid en l'enviament de dades. Els dos protocols són igualment útils, tot depèn de la necessitat de transmissió de dades que es requerisca.

El protocol TCP/IP s'ha implementat per crear un canal de comunicació entre la computadora que realitza el control del robot paral·lel i la que estiga executant l'aplicació del projecte. Amb l'ús d'aquest protocol s'aconsegueix establir una connexió fiable per rebre i/o enviar les dades de posició a través de la Internet i assegurant que no es puga perdre informació.

CAPÍTOL 3. DESCRIPCIÓ DE LA SOLUCIÓ ADOPTADA

3.1. CODI PRINCIPAL DE L'APLICACIÓ

L'objecte principal d'aquest apartat és exposar el disseny darrere de la interfície desenvolupada. Per poder donar solució al problema del posicionament del robot, i cobrir així tots els objectius requerits, s'ha desenvolupat una aplicació capaç de realitzar les múltiples funcions necessàries.

Primerament, es descriu l'estructura del codi principal per entendre, en línies generals, el funcionament. Aquest codi es pot dividir en dues parts, cadascuna amb una funcionalitat diferent. La primera s'encarrega de l'actualització dels valors les variables, i la segona gestiona l'activació dels diferents mòduls que intervenen en l'execució. Ambdues parts es mostren en la figures 36 i 37.

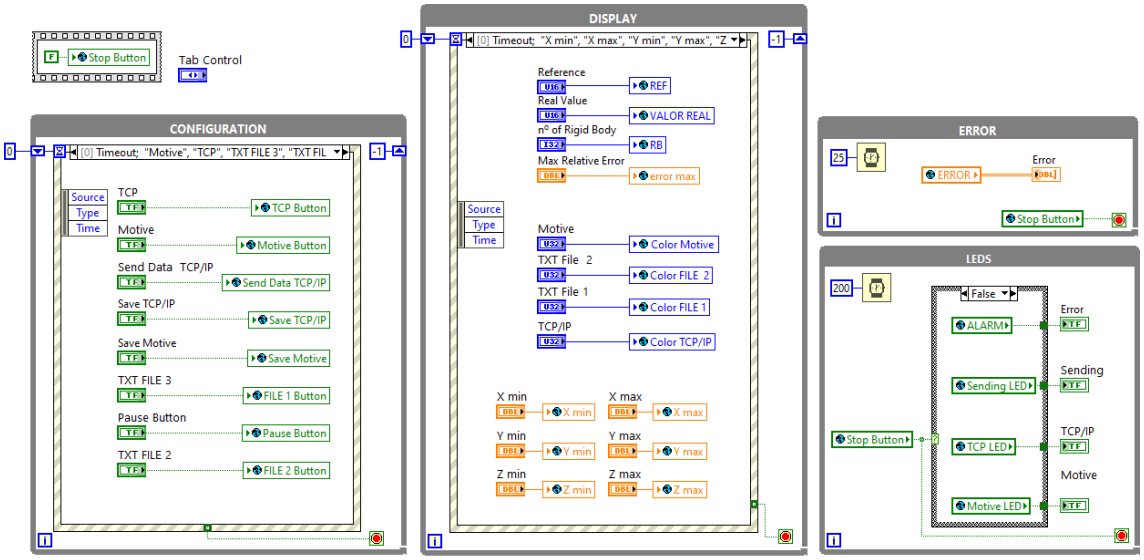

Fig. 36. **Codi Principal 1: Actualització de les variables**

Aquesta primera part del codi s'encarrega d'actualitzar en tot moment el valor dels diferents controls i indicadors amb els que treballa la interfície. Com aquesta utilitza diferents mòduls o subVIs per dur a terme les seues funcions, s'han hagut d'usar variables globals. El motiu és que aquestes són l'únic tipus que permet la comunicació entre diversos subVIs. D'aquesta forma l'aplicació es controla mitjançant l'ús de diversos botons situats en la interfície d'usuari.

S'han incorporat un total de 5 variables globals, cadascuna conté agrupats els elements amb finalitat similar. Les variables mencionades són: *Global Configuration*, *Global Graph*, *Global LEDs*, *Global Error Control* i *Global Error Indicator*. Els elements que conformen cadascuna es poden intuir en la figura 36, ja que en el codi es troben agrupats segons la variable a la que corresponen.

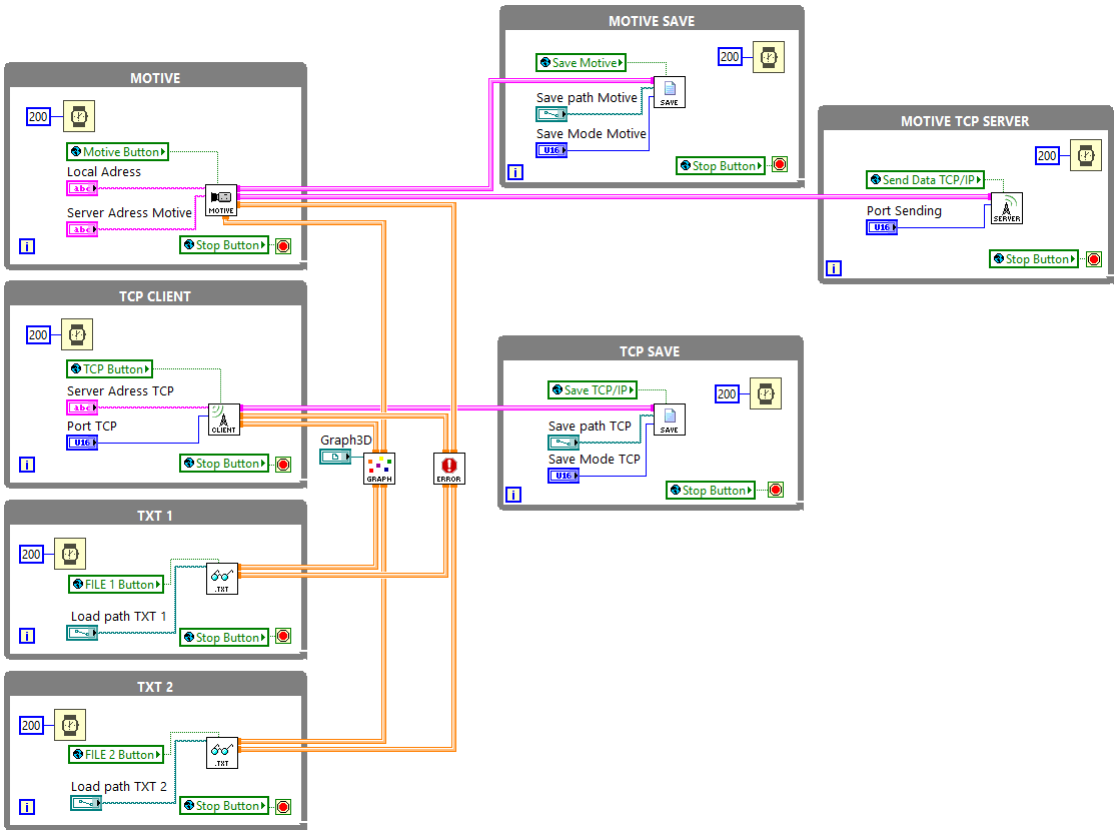

Fig. 37. **Codi Principal 2: Control dels mòduls**

Aquesta part del codi s'encarrega de gestionar l'activació dels diferents mòduls necessaris per dur a terme la funcionalitat definida per l'usuari. També és responsable de subministrar les dades necessàries per a la correcta execució de cada mòdul. La forma en que estan connectats els diferents mòduls es deu a la operativitat que s'espera que tinga el programa. En la figura 37, es pot observar les diferents rutes que poden prendre les dades de posició segons les necessitats.

La figura 37, també mostra que existeixen 4 mòduls que reben o llegeixen informació, els *productors*, i 5 mòduls que reben i processen aquestes dades, els *consumidors*. Totes les dades arriben sempre als mòduls *Graph* i *Error Calculator*, perquè la interfície està dissenyada de forma que sempre visualitza els moviments dels cossos i calcula l'error entre ells, siga quina siga la font que suministre les dades. No obstant, en funció del mòdul ho faça, el camí d'aquestes serà diferent.

Segons la procedència de les dades, existeixen diferents rutes; no es el mateix si provenen, per exemple, del Motive que del TCP Client. Per al cas de la informació que es rep del Motive, a més de graficar i calcular l'error, es pot elegir transferir les dades en temps real amb una connexió TCP/IP i/o guardar la sessió en un fitxer de text. Per a les posicions rebudes via TCP/IP, està preparat per guardar-les al mateix temps que les mostra al gràfic.

3.1.1. Mòduls

El codi de la interfície s'ha desenvolupat, en el màxim del possible, de forma modular per intentar que així fos més sostenible i reutilitzable, per a aplicacions diferents o modificacions de

la mateixa. Per aquest motiu existeixen diferents mòduls, cadascun dels qual realitza una funció diferent cobrint una necessitat determinada. Tot seguit es presenta, en línies generals, l'operativitat de cadascun.

En primer lloc, trobem els mòduls productors de dades, la icona dels quals podem observar en la figura 38. Aquest grup es composa dels mòduls:

- Motive: es connecta amb el software Motive, mitjançant a tecnologia .NET, i proporciona les dades de posició dels sòlids rígids que entreguen les càmeres del sistema OptiTrack.
- *TCP Client*: es comunica en la unitat de control del robot, mitjançant els protocols TCP/IP, i subministra les coordenades que teòricament tindria la plataforma.
- *Read*: llegeix un fitxer de tipus TXT, entregant les trajectòries guardades de sessions anteriors. Existeixen 2 mòduls pràcticament idèntics, *Read TXT 1* i *Read TXT 2*, sols es diferencien en la variable d'activació que controla el bucle intern.

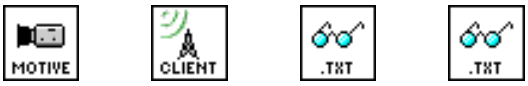

Fig.38. **Mòduls productors (Motive, *TCP Client*, *Read TXT 1* i *Read TXT 2*)**

El segon grup són els mòduls consumidors de la informació que proporcionen els anteriors, la icona d'aquests es mostra en la figura 39. Els mòduls que formen aquest grup són:

- *Graph*: grafica la posició i orientació dels diferents sòlids rígids simultàniament i en temps real. Utilitza per dur-ho a terme la tecnologia ActiveX, ja que no es podien complir els objectius usant les funcions de graficar predeterminades oferides pel LabVIEW.
- *Error Calculator*: calcula l'error existent entre la posició i orientació dels diferents cossos, per quantificar la diferencia entre la referència i el valor real.
- *TCP Server*: envia en temps real les dades obtingudes del Motive, mitjançant els protocols TCP/IP a qualsevol client compatible.
- *Save*: guarda en un fitxer TXT la trajectòria descrita pels sòlids rígids, durant el període de temps que desitge l'usuari. Existeixen dos formats d'escriure les dades: el format Normal i el *Only RB*. De forma anàloga al que passava amb el mòdul *Read*, també trobem *Motive Save* i *TCP Save*, idèntics a diferència de la variable d'activació que controla el procés intern.

Fig. 39. **Mòduls consumidors (*Graph*, *Error Calculator*, *TCP Server*, *Motive Save* i *TCP Save*)**

A continuació s'explica més en detall el funcionament del mòdul Motive, degut a que es tracta de l'element clau de la interfície desenvolupada.

3.1.1.1. Motive

Aquest és el mòdul més important de l'aplicació, perquè és l'encarregat d'establir una connexió amb el Motive, amb l'objectiu d'accedir a les dades de posició dels sòlids. S'aconsegueix usant la tecnologia .NET, mitjançant el NatNetSDK que proporciona NaturalPoint.

Primerament, com .NET es basa en una programació orientada a objectes s'ha de desenvolupar el codi de forma que s'accedisca als mètodes i atributs que ens interessen. Tots els objectes disponibles es troben dins de l'arxiu NatNetML.dll, que es tracta de l'assemblatge que s'ha de carregar al LabVIEW usant les funcions de la paleta .NET. A continuació, en la figura 40, es mostra l'esquema de la forma seguida per accedir les dades dels sòlids i marcadors:

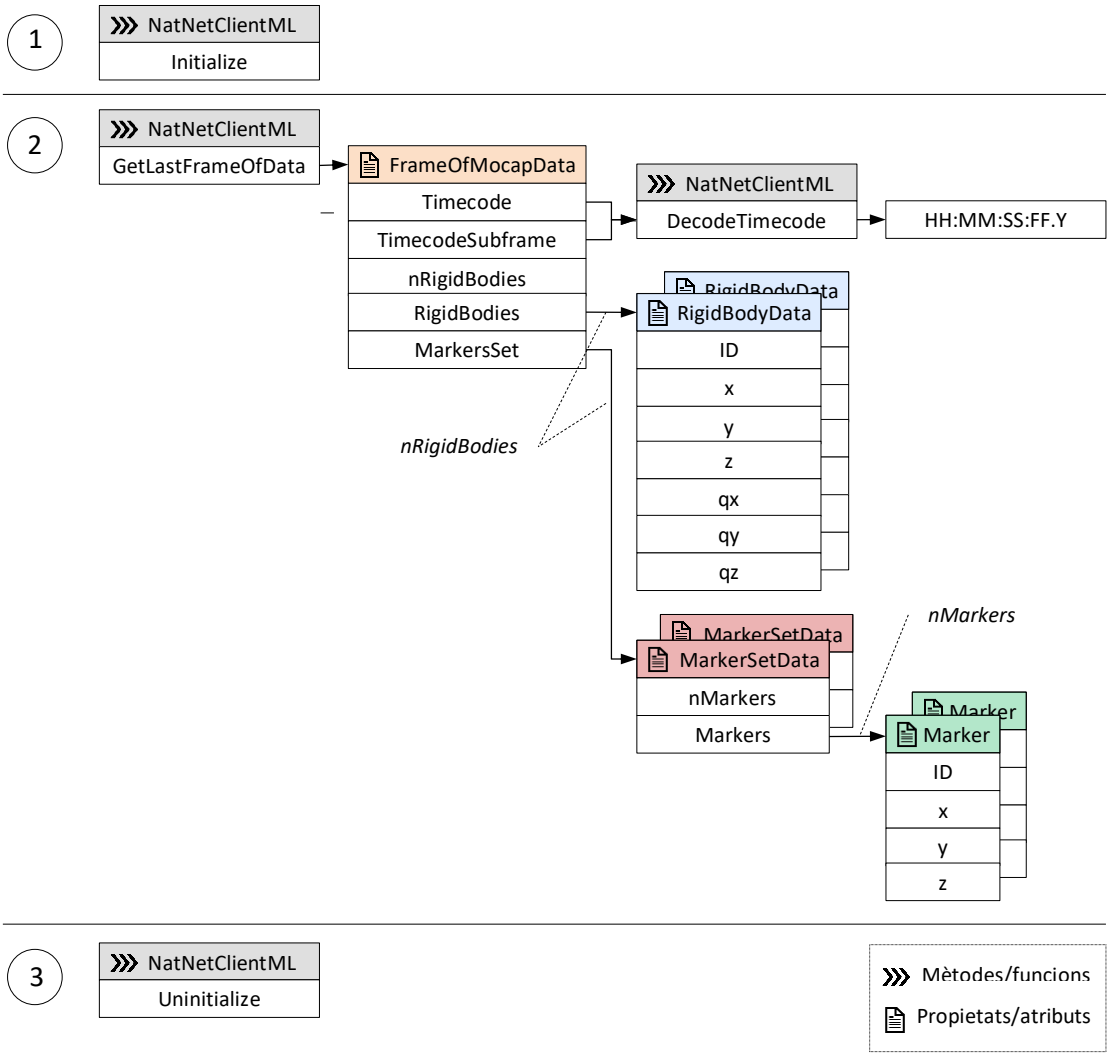

***Fig. 40*. Esquema del procés d'accés a les dades del NatNetClientML.dll**

Com es pot observar la forma d'accedir a tota la informació sobre la posició, és usant els mètodes i les propietats de cada classe d'objecte. Abans de tot, s'ha d'executar l'element que crea la referència poder accedir a cada classe d'objecte, es tracta del constructor *NatNetClientML*. Una vegada es crea, ja es poden usar les seues funcions i atributs.

Posteriorment al constructor, s'ha de seguir la seqüència indicada en la figura 40. Primer, s'executa el mètode o funció *Initialize*, que crea una connexió amb el Motive. Seguidament, s'executa el mètode *GetLastframeOfData*, que s'encarrega d'accedir a tota la informació calculada pel software de les càmeres. Finalment, quan es vol finalitzar la comunicació es realitza el tercer pas, que consisteix en cridar a la funció *Uninitilize*, per tancar la connexió creada.

El codi del mòdul Motive desenvolupat es pot observar en la figura 41, i segueix l'esquema anterior processar les dades. Algunes parts del codi s'han agrupat en diversos submòduls, com són:

- *Timecode To String*: transforma la informació del software sobre el *Timecode*, que és l'hora exacta quan s'ha pres la captura, a una cadena d'*strings*. Usa el mètode *DecodeTimecode* mostrat en la figura 40.
- *Get XYZ*: accedeix als atributs o propietats de cada sòlid rígid per accedir a la seua posició i orientació, així com la dels marcadors que el formen. Utilitza els objectes *RigidBodyData* i *Marker* ja mostrats.
- *Send Data*: organitza tota la informació obtinguda en forma de clúster o matriu, segons el destí, i la mana a traves dels canals de transmissió de dades a la resta de mòduls.

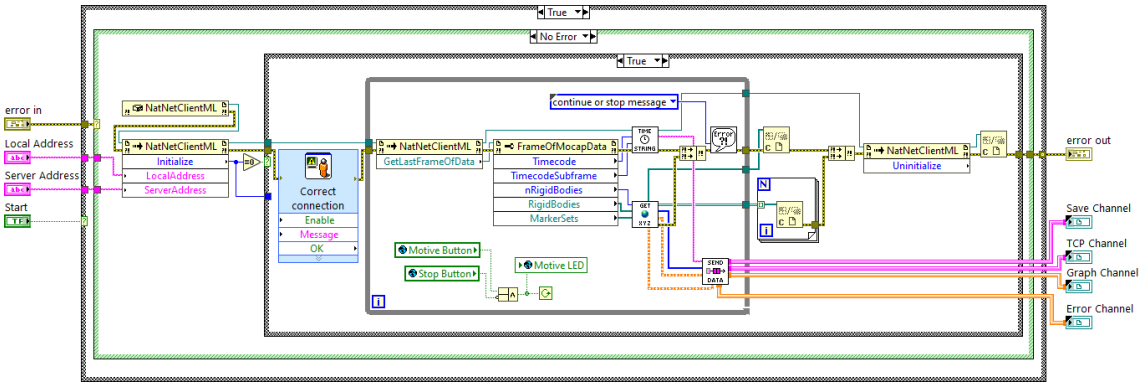

***Fig. 41*. Codi del mòdul Motive**

El funcionament, en essència, es basa en establir una connexió amb el Motive, mitjançant l'adreça IP local (on s'executa la interfície) i la IP del servidor (on s'executa el Motive), proporcionades per l'usuari. Posteriorment, entra en un bucle on es llegeix i s'envia tota la informació a la resta del programa, fent ús dels diferents submòduls presentats. Per últim, es finalitza el bucle i es tanca la connexió amb el servidor del Motive, quan l'usuari així ho elegeix mitjançant els controls del *Front Panel*.

Per aprofundir en la composició i el funcionament del codi dels submòduls i mòduls que formen l'aplicació es recomana consultar el Manual del Programador, on està detalladament explicat.

3.2. INTERFÍCIE D'USUARI

Aquest apartat presenta l'aspecte i les funcionalitats que ofereix la interfície dissenyada. La qual visualitza, en temps real, la posició dels sòlids rígids que rep del Motive, de la connexió TCP/IP i/o dels fitxers TXT. També permet a l'usuari gravar les parts que li interessen de la trajectòria dels sòlids, oferint dos formats d'escriptura diferents. La propia interfície es troba representada en la figura 42. Principalment, es pot dividir en dues parts diferenciables: el panell de control, a l'esquerra, i el visualitzador dels sòlids, a la dreta.

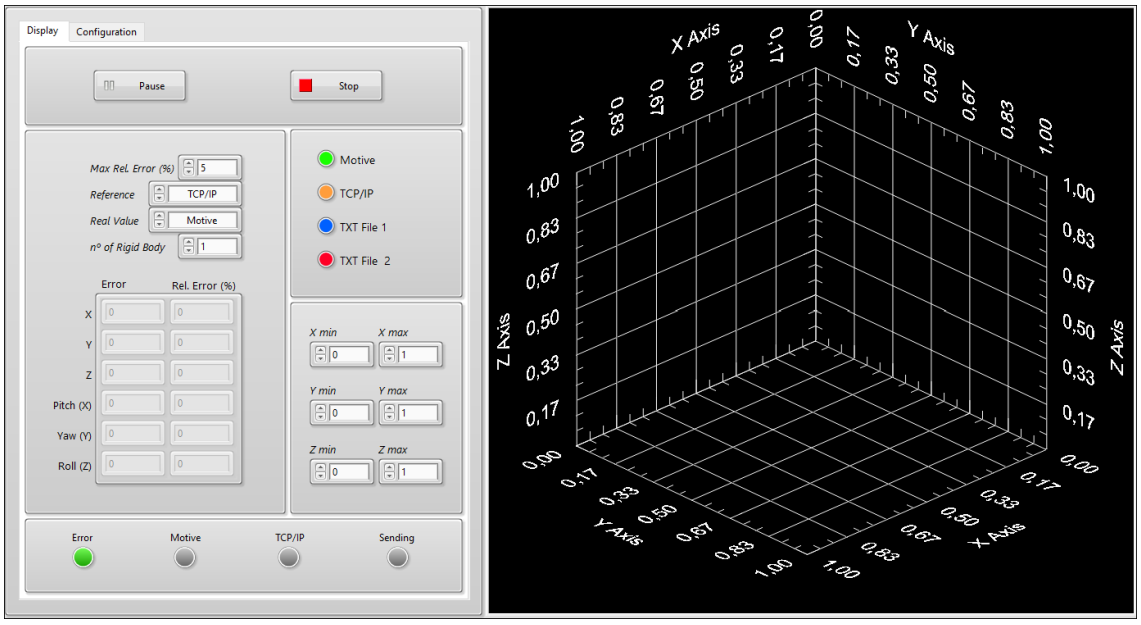

***Fig. 42*. Interfície d'usuari de l'aplicació**

El propi panell de control, compta amb dos menús diferents *Display* i *Configuration*, cadascun d'ells amb uns tipus de controls diferents, ambdós es troben representats en la figura 43. El menú *Display* permet a l'usuari controlar tots els aspectes relacionats amb la representació, com el color de cada sòlid rígid, el rang dels eixos XYZ, el botó *Pause* i *Stop*. A més, també és on s'introdueix de quin sòlid es vol calcular l'error i quin és el màxim acceptable abans de que s'active l'alarma.

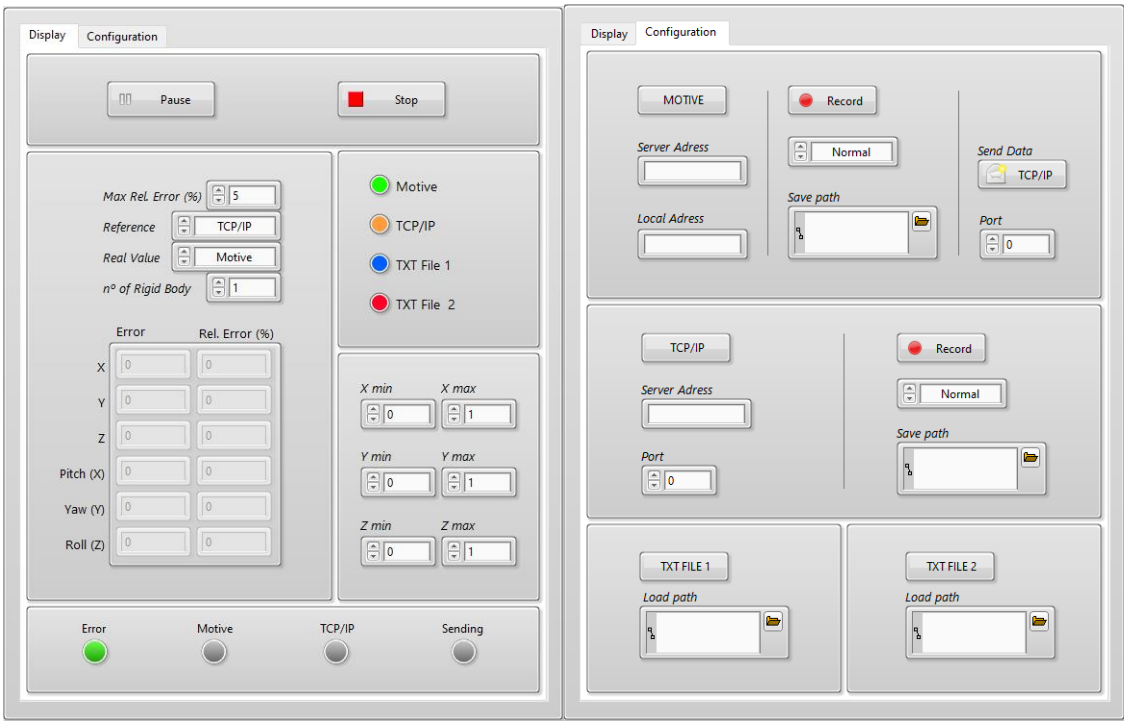

***Fig. 43*. Menús *Display* (esquerra) i *Configuration* (dreta) del panell de control**

En la segona pestanya, trobem el menú *Configuration*, aquest permet gestionar tota la informació relacionada amb les dades de posició: d'on prové, on s'envia i com es guarda. L'usuari pot elegir, en qualsevol moment, quines de les 4 fonts possibles està activa prement el botó corresponent. Les opcions són el Motive, la connexió TCP i els 2 fitxer TXT. També permet enviar les dades, usant els protocols TCP/IP, o guardar-les elegint el format i la part de l'assaig que resulte interessant.

A l'hora de gravar les trajectòries descrites pels sòlids, com s'ha comentat, existeixen dos formats d'escriure les dades: *Normal* i *Only RB*. La primera opció, està representada en la figura 44, i es tracta de l'opció predefinida. Aquest mode guarda la posició del CG de la plataforma, així com la de tots els marcadors que el formen. També afegeix una capçalera amb la data, el nombre de sòlids i marcador de cadascun, així com títols en les columnes de dades. Els fitxers guardats amb aquest mode són compatibles amb la interfície desenvolupada, permetent tornar a reproduir sessions anteriors.

```
18/7/2020 RB=1 M1=3
time         Nº ID      X       Y        Z        qx       qy       qz
19:43:15,794 1 1000000 0,007401 0,000000 0,620800 0,000000 0,135089 -0,103150
19:43:15,794 1 1000001 -0,588271 0,030890 0,580396
19:43:15,794 1 1000002 0,520143 0,208736 0,661203
19:43:15,794 1 1000003 0,273778 -0,298405 0,661203
19:43:15,802 1 1000000 0,007399 0,000000 0,620900 0,000000 0,135088 -0,103146
19:43:15,802 1 1000001 -0,588273 0,030889 0,580497
19:43:15,802 1 1000002 0,520140 0,208737 0,661303
19:43:15,802 1 1000003 0,273777 -0,298406 0,661303
19:43:15,812 1 1000000 0,007400 0,000000 0,621000 0,000000 0,135089 -0,103151
19:43:15,812 1 1000001 -0,588272 0,030891 0,580596
19:43:15,812 1 1000002 0,520142 0,208736 0,661404
19:43:15,812 1 1000003 0,273777 -0,298405 0,661404
```

Fig. 44*. Fitxer guardat en mode *Normal

L'altre mode s'anomena *Only RB*, i s'encarrega de guardar solament la posició i rotació del CG de la plataforma. Aquest, prescindeix de la capçalera i les dades corresponents als marcadors, per facilitar la lectura i processament de la informació en programes externs, com per exemple MATLAB. Degut a la falta d'informació en la capçalera de l'arxiu, el fitxer no es pot llegir correctament amb l'aplicació, aquest mode està enfocat solament a la exportació de dades. Un exemple del mateix es mostra en la següent figura:

```
09:16:55,590 1 1000000 1,007400 1,000000 1,635600 0,000000 0,135088 -0,103151
09:16:55,599 1 1000000 1,007400 1,000000 1,635700 0,000000 0,135089 -0,103148
09:16:55,607 1 1000000 1,007400 1,000000 1,635800 0,000000 0,135088 -0,103150
09:16:55,615 1 1000000 1,007400 1,000000 1,635900 0,000000 0,135089 -0,103146
09:16:55,623 1 1000000 1,007400 1,000000 1,635999 0,000000 0,135089 -0,103154
09:16:55,632 1 1000000 1,007399 1,000000 1,636100 0,000000 0,135089 -0,103150
09:16:55,640 1 1000000 1,007399 1,000000 1,636200 0,000000 0,135087 -0,103145
09:16:55,648 1 1000000 1,007401 1,000000 1,636300 0,000000 0,135088 -0,103150
```

Fig. 45*. Fitxer guardat en mode *Only RB

Per obtenir més informació sobre com usar totes les funcionalitats que ofereix la interfície, es recomana consultar el Manual de l'Usuari, on s'expliquen tots els passos a seguir per aconseguir-ho.

3.3. MÒDUL KINEMATICS

Aquest punt pretén exposar el mòdul desenvolupat per a la unitat de control del robot. El mòdul *Kinematics* s'encarrega de calcular, a partir de les dades proporcionades pels sensors i usant el model cinemàtic donat, la posició del CG i dels tres vèrtexs de la plataforma. Posteriorment, les envia usant els protocols TCP/IP a la interfície dissenyada. Per realitzar aquesta funció s'ha reutilitzat el mòdul de l'aplicació principal.

Per poder utilitzar el model matemàtic donat, programat en MATLAB, s'ha descarregat el *MathScript Module* per al LabVIEW. Aquest, crea un entorn, dins de la funció amb el seu mateix nom, on es poden carregar les funcions tipus M, d'extensió .m, directament.

Per facilitar la presentació del codi, s'ha dividit en dos blocs, el primer d'ells es mostra en la figura 46. Aquesta part s'encarrega de l'actualització de les variables que intervenen, tant dels indicadors com dels controls. A més, també llegeix totes les dades, és a dir, les mesures dels encorders, que estan contingudes en fitxers de tipus TXT.

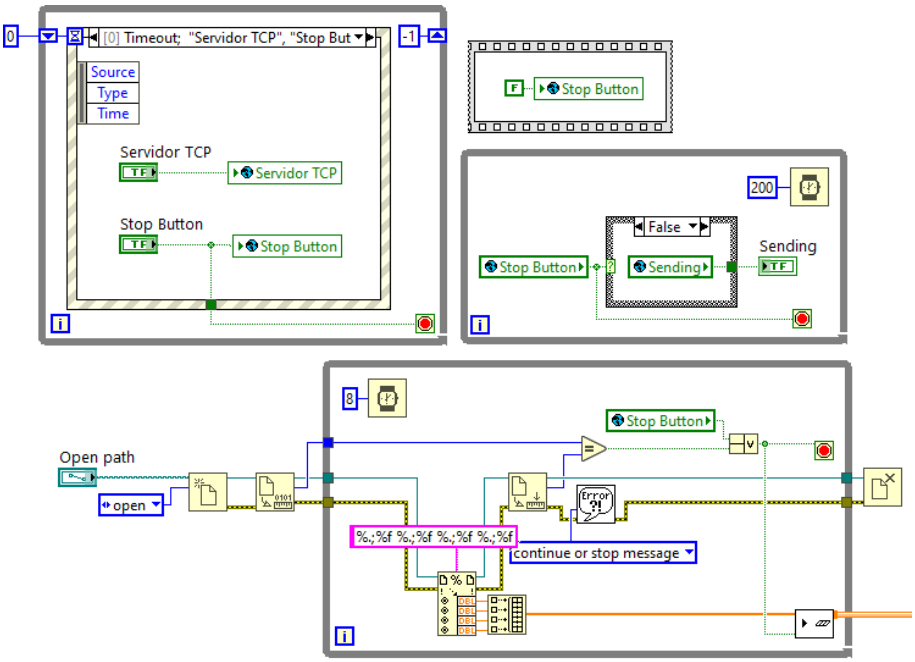

***Fig. 46*. Codi *Kinematics* 1: Actualització de variables i lectura de fitxers**

La segona part del codi es presenta en la figura 47, els passos principals que es segueixen en l'execució d'aquesta part es presenten a continuació:

- Primerament, es llegeix la informació obtinguda de l'arxiu TXT, i es defineixen els paràmetres necessaris com la posició global inicial o la geometria del robot.
- A continuació, es calcula el CG de la plataforma usant la funció *CinDirectaPos3UPS_RPU*, que es tracta del model cinemàtic donat. Carregant aquesta funció mitjançant el *MathScript Node.*
- Posteriorment, es calculen els vèrtex A, B i C de la plataforma mòbil usant la funció desenvolupada *Increment* i el CG obtingut anteriorment.
- Seguidament, s'obté l'hora exacta en eixe instant, es defineix el vector amb el nombre de sòlids i marcadors, i s'organitzen totes aquestes dades en el clúster de dades

- Finalment, s'envia la informació al mòdul TCP Server, el qual la transmetrà via Internet a la interfície presentada anteriorment.

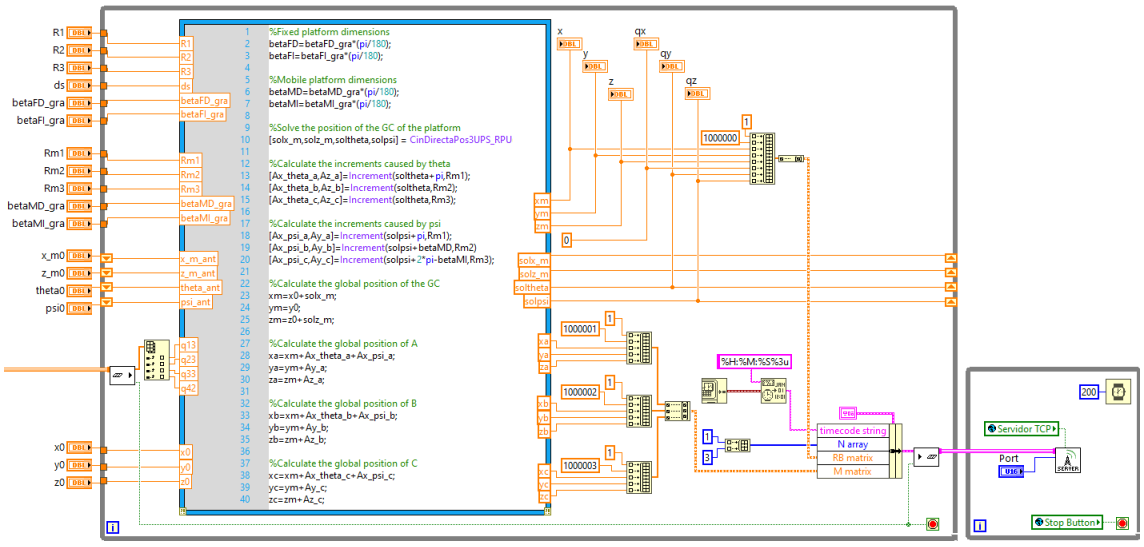

***Fig. 47.* Codi *Kinematics* 2: *MathScript Node* i enviament de dades**

Tot seguit, en la figura 48, es troba representat el *Front Panel* d'aquest mòdul, existeixen dues parts diferenciables: els *Inputs* i els *Outputs & Controls*. El primer d'ells, els *Inputs*, es correspon a la part on s'introdueixen tots els paràmetres necessaris per dur a terme els càlculs. Entre ells es troba: la ruta de l'arxiu que conté les mesures dels encoders, un punt proper a la solució per a que convergisca el model, la posició global inicial del CG de la plataforma i els paràmetres geomètrics d'aquesta. Els últims depenen del tipus de configuració que tinga el robot, al panell s'ha predefinit els valor de la configuració nº1, presentats en la taula xx, que és amb la que s'està treballant actualment.

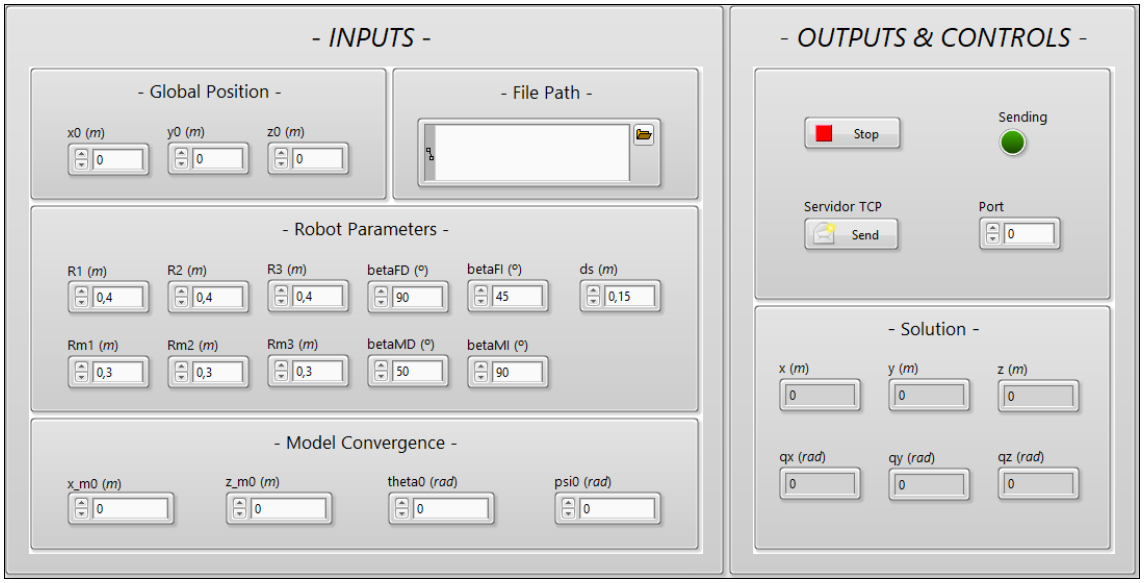

Fig. 48. Front Panel* del mòdul *Kinematics

El segon bloc del panell s'encarrega de controlar l'enviament de les dades via TCP/IP, així com de si es vol detenir el procés. També es troben situats en aquesta part els indicadors que mostren els valors d'eixida de la posició i orientació del CG de la plataforma.

És important destacar que el programa desenvolupat, carrega un arxiu TXT amb les mesures reals que entreguen els encoders incrementals. No obstant, en realitat no serà aquesta la seua forma final, ja que les dades que ara es llegeixen dels fitxers es substituiran per les que entreguen els sensors en temps real. Aquest fet serà possible perquè el mòdul creat s'afegirà a la unitat de control, encarregada d'activar els diferents actuadors, al mateix temps que llegeix les mesures dels encorders.

Finalment, si es requereix més informació sobre el codi o l'ús del *Front Panel* d'aquest mòdul, es convenient consultar el Manual del Programador o el Manual de l'Usuari, on es troba cada part molt més desenvolupada.

CAPÍTOL 4. CONCLUSIONS

4.1. RESULTATS OBTINGUTS

En aquest apartat s'avalua el comportament de la interfície en l'execució de les diferents funcions per a les que ha estat dissenyada. S'ha anat analitzant el funcionament de cada mòdul per separat, a mesura que s'anava programant la interfície. Una vegada s'ha arribat a la versió final de l'aplicació, s'ha provat el funcionament de forma conjunta. A continuació, es presenten cronològicament els assajos realitzats i els resultats obtinguts.

Primerament, s'ha verificat que el mòdul Motive funcione correctament, aconseguint connectar-se a l'ordinador amb el software de l'Optitrack, mitjançant un altre ordinador amb una versió preliminar de l'aplicació. L'assaig realitzat ha consistit en l'establiment d'una connexió .NET amb el software de l'OptiTrack, i la visualització de les dades entregades en temps real. Per dur-ho a terme, ambdós ordinadors s'han connectat a la mateixa xarxa d'Internet, generada pel mateix router. Posteriorment, s'han introduït les adreces IP assignades i s'ha polsat el botó per a connectar-se. Tot seguit, s'ha verificat que sí que accedeix a les dades entregades de forma satisfactòria, perquè la interfície ha permès visualitzar el moviment que experimentaven els marcadors d'una barra de calibració.

A continuació, s'han posat a prova els mòduls *Read* i *Save*. Per fer-ho, s'ha creat un mòdul que genera dades fictícies imitant el mateix format entregat pel Motive, i s'han aconseguit guardar de forma exitosa en un fitxer tipus TXT, usant el mòdul *Save*. S'ha comprovat que el mòdul era capaç d'escriure els dos modes (*Normal* i *Only RB*) correctament. Posteriorment, s'ha comprovat el mòdul *Read*, obrint el mateix fitxer TXT generat amb el mode *Normal*, i verificant que el visualitzador de la interfície graficara satisfactòriament el moviment dels marcadors ficticis.

El següent mòdul que s'ha provat és *Graph*, l'encarregat de graficar. En els assajos anteriors aquest mòdul es tractava d'una versió molt rudimentària amb capacitat solament per graficar punts. Després de desenvolupar el mòdul *Graph* al complet, mitjançant l'ús de la tecnologia ActiveX, s'ha verificat que funcionen totes les opcions que ofereix. S'ha provat que sí que permet canviar els colors dels sòlids a priori, així com l'escala dels eixos, tant abans com en meitat de la representació. A més, s'ha observat que genera correctament la silueta dels sòlids rígids, i que és capaç de pausar la visualització del moviment en qualsevol moment, i després poder seguir graficant.

Posteriorment, s'han comprovat els mòduls encarregats de crear el canal de comunicació TCP/IP, és a dir, el *TCP Server* i *TCP Client*. L'assaig realitzat ha consistit en transmetre les dades de posició d'uns marcadors ficticis, entre dos ordinadors diferents. Un d'ells s'ha encarregat d'enviar la informació d'un fitxer de text usant el mòdul *TCP server*, mentre que l'altre ha guardat totes les dades rebudes mitjançant el *TCP Client*. S'ha verificat la correcta transferència d'informació comprovant que ambdós fitxers TXT, el que llig el servidor i el que guarda el client, són idèntics.

L'últim mòdul creat i provat ha estat el mòdul *Error Calculator*, s'ha comprovat que obté l'error entre els sòlids rígids que estiguen seleccionats en cada moment. S'ha verificat l'alarma sonora i l'indicador s'activen en quant es supera el llindar de l'error establert, i es desactiven quan no ho fa; així com que l'error màxim es pot modificar en qualsevol moment de la sessió.

Finalment, s'ha provat l'operativitat de la versió final de l'aplicació, que inclou tots el mòduls descrits treballant de forma conjunta. S'ha verificat que les accions de control: *Pause* i *Stop*, són efectives i tenen el comportament esperat. S'ha verificat que la interfície funcione adequadament quan s'executen diferents mòduls de forma simultània, així com la correcta sincronització entre ells a l'hora de visualitzar els moviments perquè el mòdul *Graph* sols grafica quan disposa de les dades de tots els mòduls actius en eixe moment.

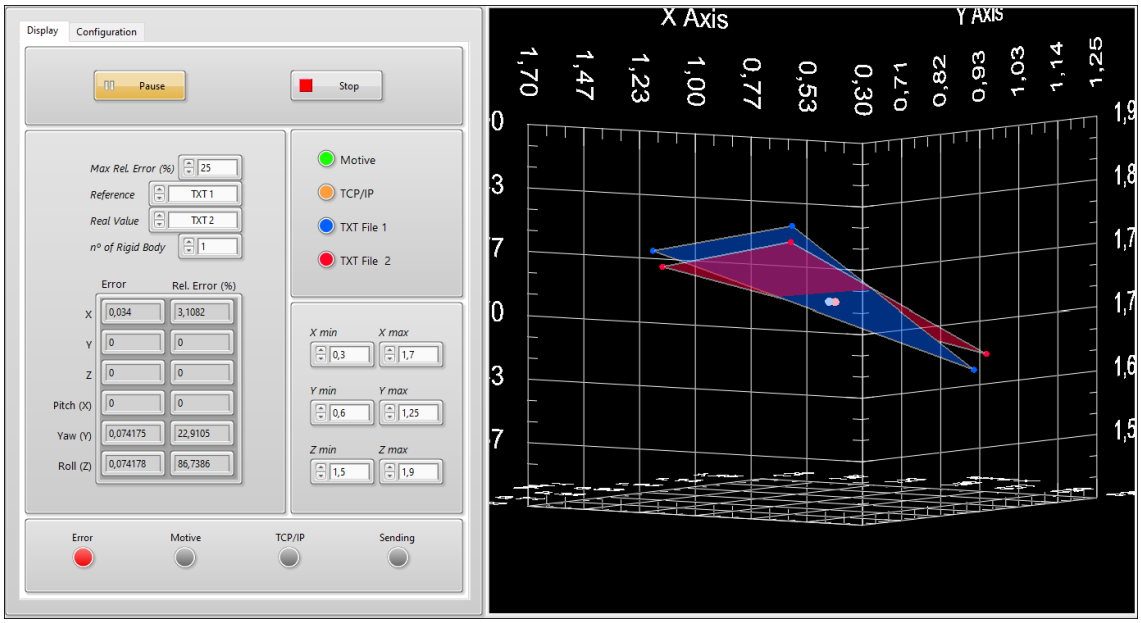

***Fig. 49*. Interfície d'usuari durant una sessió**

En aquest punt, s'ha constatat que a la freqüència de 120FPS, és a dir, cada 8'33ms, el processament de dades funciona perfectament, però amb la visualització. La funció de graficar no es capaç de treballar a aquesta velocitat, llavors es produeix un retard a l'hora de visualitzar els moviments. La freqüència de 120FPS, és un paràmetre imposat pel software Motive, perquè és la màxima velocitat a la que pot entregar les dades. Aquesta freqüència indica que l'aplicació té capacitat per realitzar una iteració, en el tots mòduls i de forma simultània, almenys cada 8,33ms (a excepció del mòdul *Graph*).

Per aquesta raó, s'ha optimitzat i reduït al màxim el codi corresponent a *Graph*, degut a que es tracta de la part més crítica en temps d'execució de tota el programa. Inclús així, havent reduït

el temps respecte de la duració que tenia inicial, la màxima velocitat per iteració que s'ha arribat a aconseguir és de 33ms. Com no s'ha aconseguit reduir fins als 8,33ms, la solució per la que s'ha optat és graficar solament 1 de cada 3 mostres de les que li arriben a l'aplicació. El motiu, és que si s'haguera de graficar tota la informació que rep l'aplicació amb eixes velocitats, es produiria un retard cada vegada major entre la realitat i la visualització dels moviments, impossibilitant realitzar un control en temps real sobre la posició de la plataforma.

Finalment, s'ha verificat l'execució del mòdul *Kinematics*, encarregat d'obtenir la cinemàtica de la plataforma. Per dur-ho terme, s'ha carregat un fitxer amb un exemple de les mesures reals dels encoders incrementals del robot, i s'han enviat els resultats obtinguts a l'aplicació desenvolupada. S'ha pogut observar la representació del moviment que teòricament realitzaria la plataforma, al mateix temps que es guardaven totes les dades en un nou fitxer de text. Aquest fet ha permès comprovar que el fitxer TXT obtingut és idèntic al generat per MATLAB, amb el mateix model cinemàtic; confirmant així el correcte processament i enviament de dades del mòdul.

4.2. CONCLUSIONS

L'objecte principal d'aquest apartat és analitzar el disseny de la interfície, per avaluar si compleix amb els objectius establerts. Per aquest motiu, es comparen els resultats obtinguts amb els requeriments necessaris per a dur a terme les funcions demandades. S'ha de tenir en compte la limitació a l'hora de desenvolupar l'aplicació que ha suposat l'aparició del COVID-19. Aquest fet ha impossibilitat accedir als laboratoris, on es trobava el robot paral·lel i el sistema de càmeres OptiTrack, i el seu corresponent software Motive. Per aquesta raó no s'ha pogut provar l'última versió de l'aplicació amb aquest software i amb els moviments reals de la plataforma.

El primer requisit per a supervisar la ubicació de la plataforma, és visualitzar el moviment d'aquesta. La interfície és capaç de dur a terme aquesta funció, encara que no a la velocitat que s'havia plantejat inicialment. El codi del mòdul *Graph*, ha sigut una de les que més s'ha revisat per optimitzar-la, i així millorar el temps de resposta de cada iteració. Malgrat això, ha sigut impossible reduir els 33ms de temps d'execució, front als 8'33ms que és la màxima velocitat a la que les càmeres proporcionen dades. No obstant, 33ms signifiquen 30FPS, és a dir, que proporciona 30 imatges cada segon per tal de simular el moviment del robot, el que aconsegueix igualment una visualització fluida d'aquest.

L'aplicació també aconsegueix realitzar una connexió amb un altre ordinador, per tal de rebre i/o enviar les dades sobre la posició del sòlid rígid. Aquesta s'estableix mitjançant els protocols TCP/IP, que permeten una comunicació segura a través d'Internet. L'altra aplicació en qüestió es tracta del mòdul *Kinematics*, desenvolupat per afegir-se a la unitat de control del robot. Aquest també s'adequa perfectament a l'objectiu, ja que entrega la posició i orientació del CG de la plataforma, i dels seus vèrtexs (A_1, B_1, C_1), partint de les mesures dels sensors interns del robot.

Altre requeriment que l'aplicació ha superat amb èxit és comunicar-se amb el software Motive de les càmeres. La connexió es realitza mitjançant la tecnologia .NET i la xarxa d'Internet, accedint a tota la informació que entrega el sistema OptiTrack, usant les llibreries NatNetML. D'aquesta forma la interfície pot treballar amb totes les dades de posició dels sòlids rígids

definits, així com dels marcadors que els formen. Malgrat que es tracta d'un aspecte que no s'ha pogut provar en la última versió de l'aplicació, sí que s'ha comprovat el correcte funcionament amb una versió molt similar del mòdul Motive final.

La interfície també havia de disposar de les opcions de guardar i carregar els assajos que s'estigueren realitzant, per poder comparar les trajectòries de la plataforma a posteriori. La versió final, permet començar a llegir i graficar, en el moment que l'usuari desitge. El mateix ocorre amb l'opció de guardar la sessió, ja que està preparat per poder començar i acabar en qualsevol moment, permetent així gravar de tota la trajectòria, la part que més interese. A més, permet guardar diversos fitxers amb diferents trajectòries durant la mateixa sessió, o bé sobreescriure el mateix fitxer les vegades que es necessite.

Un altre aspecte a destacar és la independència de la interfície respecte del cas concret d'estudi, el moviment del robot paral·lel. L'aplicació permet visualitzar el moviment de 2 sòlids rígids per cada font d'informació, les quals són: *Motive*, *TCP Server*, *Open TXT 1* i *Open TXT 2*; el que significa que la interfície pot graficar fins a 8 sòlids rígids. A més, té capacitat per fer-ho independentment de la forma que tinga cada sòlid, i del nombre de marcadors que formen cadascun. Aquestes característiques li permeten ser un software flexible i útil per diferents escenaris d'estudi amb el sistema OptiTrack, garantint-li certa continuïtat en futures investigacions que requereixen d'algunes de les funcions que ofereix.

Per últim, el caràcter modular del codi permet que siga reutilitzable en altres aplicacions que requereixen de funcions idèntiques o similars. Un exemple és el mòdul *TCP Server*, ja que s'han usat dues còpies pràcticament idèntiques en el projecte, una en el mòdul *Kinematics* de la unitat de control del robot i l'altra en la pròpia interfície. El caràcter modular també el converteix en un codi sostenible, perquè permet realitzar modificacions fàcilment per adequar l'aplicació a nous contexts de treball. Si es necessitara realitzar canvis, com per exemple, augmentar el nombre de fitxers a carregar, les connexions TCP/IP o els sòlids a graficar, sols seria necessari copiar els mòduls o submòduls ja programats i connectar-los correctament.

En quant a les limitacions, en el desenvolupament de la interfície trobem principalment tres. La primera, es tracta de la limitació dels 33ms en la velocitat de visualització dels sòlids. Imposada pel mòdul *Graph*, que condiciona que no es puga representar el moviment amb tota la fluïdesa que les dades permeten. La segona limitació, es correspon al mòdul *Motive* que es connecta al software de l'OptiTrack, ja que sols s'ha provat per separat i en una versió preliminar, no amb la resta de mòduls en la versió final de la interfície. L'última limitació és sobre el mòdul *Kinematics*, perquè no s'ha pogut assajar amb les dades entregades per la unitat de control del robot en temps real, perquè per fer-ho es necessitarà modificar la font que li proveeix les dades, ja que actualment s'obtenen obté d'un fitxer TXT.

Addicionalment, al llarg del desenvolupament d'aquest projecte, s'han adquirit diferents coneixements a nivell personal. S'ha adquirit certa experiència de programació amb el LabVIEW, gràcies al disseny de la interfície i dels diferents mòduls necessaris. Aquest fet, ha permès també obtenir una base sobre el funcionament de les tecnologies .NET, ActiveX i els protocols TCP/IP. A més, s'han pogut donar unes pinzellades sobre la robòtica i el complex problema que pot arribar a presentar l'obtenció de la cinemàtica d'alguns robots. Altre aspecte a comentar és el coneixement adquirit sobre el funcionament del sistema de captura de moviment òptic amb marcadors, així com les diferents alternatives que existeixen al sistema utilitzat.

Finalment, és important mencionar que en tot moment s'ha desenvolupat l'aplicació intentant fer-la una interfície amigable. S'ha elaborat amb un disseny el més intuïtiu i senzill possible, per a que resulte atractiu i funcional per a l'usuari final. Per tots aquests motius, es pot afirmar que s'han aconseguit tots els objectius esperats de la interfície.

4.3. FUTURES MILLORES

En aquest últim apartat, es presenten algunes millores que podrien realitzar-se a l'actual interfície per augmentar les seues prestacions. Degut a que l'aplicació ha sigut creada per al control de l'error en el posicionament, qualsevol element gràfic que ajude a la visualització d'aquest resultaria interesant. Podria resultar molt útil unes gràfiques que mostraren la posició i orientació per cada eix, en gràfics tipus *Waveform* o similar (que estan disponibles al LabVIEW). Aquest tipus de gràfic es mostra en la figura 50, existirien 6 en total (3 translacions i 3 rotacions) i cada color estaria assignat a un sòlid diferent. D'aquesta manera es visualitzaria ràpidament el signe i magnitud de la diferència entre els valors dels sòlids que es seleccionen.

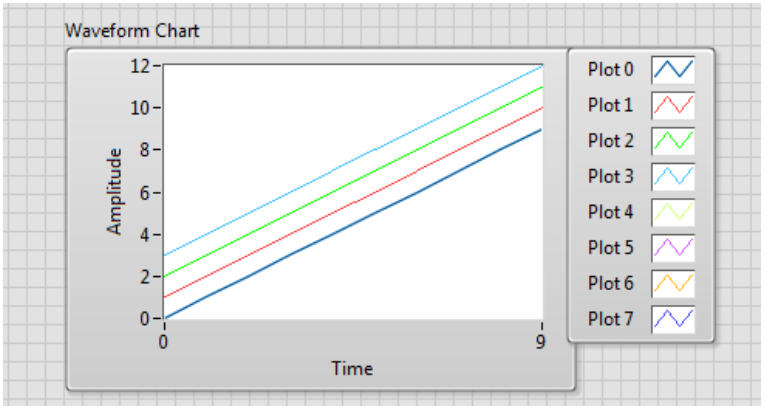

***Fig. 50*. Exemple del tipus de gràfica *Waveform* del LabVIEW**

També podria ser interesant poder guardar, en un fitxer independent o en el dels propis sòlids, els errors que es calculen al mòdul *Error Calculator*. Malgrat que el mòdul, sí que pot treballar a la freqüència imposada pel Motive, actualment els errors calculats sols es mostren per pantalla.

Per últim, un altre aspecte que també es podria millorar és l'opció de guardar els assajos. Quan existeixen dos sòlids per cada mòdul productor, seria molt pràctic poder elegir guardar les trajectòries dels dos sòlids en fitxers separats. D'aquesta manera s'obtindrien dos arxius TXT cadascun corresponent als moviments del CG de sòlid però ambdós provinents del Motive o del *TCP Server*. Aquesta característica sempre la podria configurar l'usuari, i resultaria molt interesant a l'hora de poder treballar les dades amb programes externs, ja que amb la versió actual, la posició dels dos sòlids es guarda en el mateix fitxer, dificultant la seua lectura.

CAPÍTOL 5. BIBLIOGRAFIA

[1] NOF, S.Y. (1999). *Handbook of industrial robotics*. John Wiley & Sons. ISBN 0-471-17783-0.

[2] INTERNATIONAL FEDERATION FOR THE PROMOTION OF MECHANISM AND MACHINE SCIENCE. *Dictionary of terminology.* <http://www.iftomm-terminology.antonkb.nl/2057_1031/frames.html> [Consulta: 21 de juny de 2020]

[3] MERLET, J.-P. (2006). *Parallel robots*. Springer Science & Business Media. ISBN 1-4020-4133-0. <https://link.springer.com/book/10.1007%2F1-4020-4133-0> [Consulta: 14 de juny de 2020]

[4] TANEV, T.K. (2000). Kinematics of a hybrid (parallel–serial) robot manipulator. *Mechanism and machine theory*, vol. 35, no. 9, pp. 1183-1196. ISSN 0094-114X. <https://www.sciencedirect.com/science/article/abs/pii/S0094114X99000737> [Consulta: 14 de juny de 2020]

[5] LYNCH, K.M. Y PARK, F.C. (2017). *Modern Robotics*. Cambridge University Press. ISBN 1-107-15630-0. <http://hades.mech.northwestern.edu/images/7/7f/MR.pdf> [Consulta: 4 de juny de 2020]

[6] TAGHIRAD, H.D. (2013). *Parallel robots: mechanics and control*. CRC press. ISBN 1-4665-9928-6. <https://learning.oreilly.com/library/view/parallel-robots/9781466555778/?ar> [Consulta: 9 de juny de 2020]

[7] FANUC AMERICAN CORPORATION (2018). *Robot SR-3iA/SR-6iA specifications.* <https://www.fanucamerica.com/cmsmedia/datasheets/SR%20Series%20SCARA%20Robot%20Product%20Info%20Sheet_262.pdf > [Consulta: 31 de juny de 2020]

[8] KOCH, P. M., KESTEVEN, M., NISHIOKA, H., JIANG, H., LIN, K. Y., UMETSU, K., ... & CHEREAU, G. ANUC (2009). The AMiBA hexapod telescope mount. *The Astrophysical Journal*, *694*(2), 1670. <https://iopscience.iop.org/article/10.1088/0004-637X/694/2/1670/pdf> [Consulta: 3 de juliol de 2020]

[9] OMRON INDUSTRIAL AUTOMATION. *Quattro 650H / HS specifications.* <http://www.omron-ap.com.my/products/family/3513/specification.html> [Consulta: 11 de juliol]

[10] PULLOQUINGA, J. L., MATA, V., VALERA, Á., ZAMORA-ORTIZ, P. DÍAZ-RODRÍGUEZ, M. Y ZAMBRANO, I. (2020) Experimental Analysis of Type II Singularities and Assembly Change Points in a 3UPS+RPU Parallel Robot. *Mechanism and Machine Theory*.

[11] ELVIRA IZURRATEGUI, C. (2015). "Configuraciones singulares de un robot". CEAIR de la Universitat de la Rioja en *Youtube.* <https://www.youtube.com/watch?v=maFoe6515D4> [Consulta: 28 de juny de 2020]

[12] HERRERO DEBÓN, A. (2014). *Rango*. PoliMedia de la Universitat Politècnica de València. <https://media.upv.es/#/portal/video/b7eec73a-23ca-1840-98ed-a58dad822c36> [Consulta: 1 de juliol de 2020]

[13] GUERRA-FILHO, G. (2005). *Optical Motion Capture: Theory and Implementation*. (Vol. 12). RITA. <https://pdfs.semanticscholar.org/0714/7486b65d12c4326ccb3ad54ca612b52e1ac3.pdf> [Consulta: 10 d'abril de 2020]

[14] ECHEVERRY, L. L., JARAMILLO HENAO, A. M., RUIZ MOLINA, M. A., VELÁSQUEZ RESTREPO, S. M., PÁRAMO VELÁSQUEZ, C. A. I SILVA BOLÍVAR, G. J. (2018). Sistemas de captura y análisis de movimiento cinemático humano: una revisión sistemàtica. *Prospectiva*, Vol. 26, N° 2, p.24-34. <http://www.scielo.org.co/pdf/prosp/v16n2/1692-8261-prosp-16-02-00024.pdf> [Consulta: 19 d'abril de 2020]

[15] BRAVO M., D.A., RENGIFO R., C.F. I AGREDO R., W. (2016). Comparación de dos Sistemas de Captura de Movimiento por medio de las Trayectorias Articulares de Marcha. *Revista Mexicana de la Ingeniería Biomédica*, Vol. 37, N° 2, p. 149-160. <http://www.scielo.org.mx/pdf/rmib/v37n2/2395-9126-rmib-37-02-00149.pdf> [Consulta: 28 de març de 2020]

[16] MICROSOFT CORPORATION. *Azure Kinect DK hardware specifications.* <https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification> [Consulta: 30 de març de 2020]

[17] PARENT, R., EBERT, D.S., GOULD, D., GROSS, M., KAZMIER, C., LUMSDEN, C.J., KEISER, R., MENACHE, A., MÜLLER, M. I MUSGRAVE, F.K. (2009). *Computer animation complete: learn motion capture, characteristic, point-based, and Maya winning techniques*. Morgan Kaufmann. ISBN 0-12-378564-2. <https://learning.oreilly.com/library/view/computer-animation-complete/9780123750785/?ar> [Consulta: 5 d'abril de 2020]

[18] ARISTIDOU, A. I LASENBY, J. (2013). Real-time marker prediction and CoR estimation in optical motion capture. *The Visual Computer*, vol. 29, no. 1, pp. 7-26. <https://link.springer.com/article/10.1007/s00371-011-0671-y> [Consulta: 14 d'abril de 2020]

[19] NATURALPOINT INC. *OptiTrack Documentation Wiki: Active Marker Tracking.* <https://v22.wiki.optitrack.com/index.php?title=Active_Marker_Tracking> [Consulta: 25 d'abril de 2020]

[20] NATURALPOINT INC. *OptiTrack Documentation Wiki: Calibration.* <https://v22.wiki.optitrack.com/index.php?title=Calibration#Overview> [Consulta: 28 d'abril de 2020]

[21] NATURALPOINT INC. *OptiHub User Manual*. (Versió 0.9.4). <https://www.optitrack.com/public/documents/OptiHub%20User%20Guide.pdf> [Consulta: 30 d'abril de 2020]

[22] NATURALPOINT INC. (2016). *NatNet API User's Guide*. (Versió 2.10.0). <https://www.optitrack.com/public/documents/natnet-api-user-guide-2.10.0.pdf> [Consulta: 27 de gener de 2020]

[23] STONE, D., JARRETT, C., WOODROFFE, M. I MINOCHA, S. (2005). *User interface design and evaluation*. Elsevier. ISBN 0-08-052032-4.

[24] SPOLSKY, A.J. (2008). *User interface design for programmers*. Apress. ISBN 1-4302-0857-0. <https://learning.oreilly.com/library/view/user-interface-design/9781893115941/?ar> [Consulta: 24 de març de 2020]

[25] LAJARA VIZCAINO, J.R. I PELEGRÍ SEBASTIÁ, J. (2017). *LabVIEW: entorno gráfico de programación*. (3ª ed.). Marcombo. ISBN 978-84-267-2436-6.

[26] BITTER, R., MOHIUDDIN, T. I NAWROCKI, M. (2017). *LabVIEW: Advanced programming techniques*. CRC press. ISBN 1-4200-0491-3. <https://www.taylorfrancis.com/books/9781315222097> [Consulta: 9 de febrer de 2020]

[27] REBOLLO PEDRUELO, M. (2008). *TCP/IP*. PoliMedia de la Universitat Politècnica de València. <https://media.upv.es/#/portal/video/18c7b620-56fe-6146-aa5f-cf72dfa85061> [Consulta: 28 de febrer de 2020]

[28] BAYDAL CARDONA, M. E. (2015). *Tipos de servicio del nivel de transporte en la arquitectura TCP/IP*. PoliMedia de la Universitat Politècnica de València. <https://media.upv.es/#/portal/video/960e7809-00f3-4a19-befb-9479c9ba8e3b> [Consulta: 28 de febrer de febrer]

[29] BAYDAL CARDONA, M. E. (2018). *Nivel de red en la arquitectura TCP/IP.* PoliMedia de la Universitat Politècnica de València. <https://media.upv.es/#/portal/video/25b3b720-77de-11e8-8de6-6d7aa0bef545> [Consulta: 28 de febrer]

TREBALL FINAL DE GRAU EN ENGINYERIA EN TECNOLOGIES INDUSTRIALS

PRESSUPOST

AUTOR: Eros Iván Costa Andrés

TUTORA: Marina Vallés Miquel

COTUTOR: Rafael José Escarabajal Sánchez

Curs Acadèmic: 2019-20

ÍNDEX DEL PRESSUPOST

1. QUADRE DE PREUS Nº1: MÀ D'OBRA

Taula 1. Quadre de preus de la mà d'obra

Codi	Unitats	Denominació	Salari (€/h)
MO.GITI	h	Enginyer/a Tècnic Industrial	19
MO.MUII	h	Enginyer/a Industrial	25

2. QUADRE DE PREUS Nº2: MATERIALS

2.1. SOFTWARE

Taula 2.1. Quadre de preus del software

Codi	Unitats	Denominació	Preu (€/any)	Preu (€/h)
MS.LV	u	Llicència LabVIEW Base	406	0,23
MS.MATH	u	Llicència MathScript Module	614	0,35
MS.MOT	u	Llicència Motive:Tracker	603,79	0,34
MS.O365	u	Llicència Microsoft Office 365	0	0

Per obtenir el període d'amortització de les llicències, s'ha considerat un total de 220 dies laborables anuals, amb una jornada laboral completa de 8h:

$$Hores\ laborals\ anuals = \frac{8\ h}{1\ dia} \cdot \frac{220\ dies}{1\ any} = 1.760\ h/any$$

Destacar que la llicència del software Microsoft Office 365, és gratuïta per a tots els estudiants i docents de les institucions acadèmiques.

2.2. HARDWARE

Taula 2.2. Quadre de preus del hardware

Codi	Unitats	Denominació	Estimació (€)	Preu (€/h)
MH.OP	u	Ordenador Portàtil	900	0,1
MH.OS	u	Ordinador de Sobretaula	700	0,08

Per al càlcul de l'amortització s'ha tingut en compte un període de vida útil dels ordinadors de 5 anys, amb les hores laborals anuals calculades anteriorment:

$$Període\ d'amortització\ total = 5\ anys \cdot \frac{1.760\ h}{1\ any} = 8.800\ h$$

2.3. INVESTIGACIÓ: SISTEMA OPTITRACK

Taula 2.3. Quadre de preu del material d'investigació

Codi	Unitats	Denominació	Import (€)	Preu (€/h)
MI.OPTI	u	Sistema OptiTrack	18.289,15	1,04

Per al càlcul de l'amortització de l'equip d'investigació OptiTrack, s'ha considerat un període de vida útil de 10 anys, amb les hores laborals anuals calculades anteriorment:

$$Període\ d'amortització\ total = 10\ anys \cdot \frac{1.760\ h}{1\ any} = 17.600\ h$$

L'import total del sistema de captura de moviment es troba desglossat en la pàgina següent, on es presenta el pressupost detallat de tots els ítems instal·lats.

A continuació, es presenta el pressupost complet de l'equip OptiTrack instal·lat en el laboratori:

Taula 2.4. Pressupost de l'equip Optitrack del laboratori

Denominació	Unitats	Preu (€/u)	Import (€)
Càmera FLEX13, 800nm Long-Pass (IR), 1,3 Mpíxel	10	999	9.990
USB Cable High Grade, Down Angle	10	10	100
OptiHub	2	299	598
Cable de connexió USB	2	5	10
Extensió de 5m del Cable USB 2.0 Active	4	20	80
SYNC Cable	1	10	10
Quadrat de calibració CS-200	1	149	149
Kit de calibració CW-500	1	299	299
Marcador: 14.0mm (9/16), rosca M4, paquet de 10	2	50	100
Marcador: 19.0mm (3/4), rosca M4, paquet de 10	2	60	120
Base de marcador: rosca M4, 15mm base, 15mm alçada	20	1	20
Base de marcador: rosca M4, 20mm base, 30mm alçada	20	1	20
Punt adhesiu de la base - 14mm (9/16") - Acrylic	200	0,25	50
Ultra-gran angular 850nm Infraroig LED - 10 count	1	20	20
Clau del Hardware	1	99	99
Despeses d'importació, aranzels, ports	1	950	950
Instal·lació i posada en marxa	1	2.500	2500
		Subtotal	15.115
		IVA (21%)	3.174,15
		Total	18.289,15€

3. QUADRE DE PREUS Nº3: PREUS UNITARIS

Taula 3.1. Quadre de preus unitaris del Capítol 1

Nº Ordre	Unitats	Descripció d'unitats d'obra	Preu (€)
1	CAP. 1	Planificació i disseny de les funcions de la interfície	
1.1	u	Estudi d'alternatives i recerca d'informació	118,05
1.2	u	Reunió tècnica	22,66

Taula 3.2. Quadre de preus unitaris del Capítol 2

2	CAP. 2	Desenvolupament de la interfície i del mòdul *Kinematics*	
2.1	u	Programació del codi en LabVIEW	3.146,25
2.2	u	Programació del codi en LabVIEW amb MathScript	648,71
2.3	u	Reunió tècnica via Teams	22,71
2.4	u	Test estàndard	19,91
2.5	u	Test connexió TCP/IP	20,31
2.6	u	Test connexió Motive	47,17

Taula 3.3. Quadre de preus unitaris del Capítol 3

3	CAP. 3	Recopilació i tractament de la informació	
3.1	u	Elaboració dels documents del projecte	2.197,83

4. QUADRE DE PREUS Nº4: PREUS DESCOMPOSTOS

Taula 4.1. Quadre de preus descompostos del Capítol 1

Codi	Unitats	Descripció d'unitats d'obra	Rendiment	Preu	Import
1	CAP. 1	Planificació i disseny de les funcions de la interfície			
1.1	u	Estudi d'alternatives			
MO.GITI	h	Enginyer Tècnic Industrial	6	19	114
MH.OP	h	Ordinador Portàtil	6	0,10	0,61
				CD	114,61
				CDC (3%)	3,44
				Cost total	118,05 €
1.2	u	Reunió tècnica			
MO.GITI	h	Enginyer Tècnic Industrial	0,5	19	9,50
MO.MUII	h	Enginyer Industrial	0,5	25	12,50
				CD	22
				CDC (3%)	0,66
				Cost total	22,66 €

Taula 4.2. Quadre de preus descompostos del Capítol 2

Codi	Unitats	Descripció d'unitats d'obra	Rendiment	Preu	Import
2	CAP. 2	Desenvolupament de la interfície i del mòdul *Kinematics*			
2.1	u	Programació del codi en LabVIEW			
MO.GITI	h	Enginyer Tècnic Industrial	158	19	3.002
MH.OP	h	Ordinador Portàtil	158	0,10	16,16
MS.LV	h	LabVIEW	158	0,23	36,45
				CD	3.054,61
				CDC (3%)	91,64
				Cost total	3.146,25 €
2.2	u	Programació del codi en LabVIEW amb MathScript			
MO.GITI	h	Enginyer Tècnic Industrial	32	19	608
MH.OP	h	Ordinador Portàtil	32	0,10	3,27
MS.LV	h	LabVIEW	32	0,23	7,38
MS.MATH	h	MathScript Module	32	0,35	11,16
				CD	629,82
				CDC (3%)	18,89
				Cost total	648,71 €
2.3	u	Reunió tècnica via Teams			
MO.GITI	h	Enginyer Tècnic Industrial	0,5	19	9,50
MO.MUII	h	Enginyer Industrial	0,5	25	12,50
MH.OP	h	Ordinador Portàtil	0,5	0,10	0,05
MS.O365	h	Microsoft Office 365	0,5	0	0
				CD	22,05
				CDC (3%)	0,66
				Cost total	22,71 €

Codi	Unitats	Descripció d'unitats d'obra	Rendiment	Preu	Import
2.4	u	Test estàndard			
MO.GITI	h	Enginyer Tècnic Industrial	1	19	19
MH.OP	h	Ordinador Portàtil	1	0,10	0,10
MS.LV	h	LabVIEW	1	0,23	0,23
				CD	19,33
				CDC (3%)	0,58
				Cost total	19,91 €
2.5	u	Test connexions TCP/IP			
MO.GITI	h	Enginyer Tècnic Industrial	1	19	19
MH.OP	h	Ordinador Portàtil	1	0,10	0,10
MS.LV	h	LabVIEW	1	0,23	0,23
MH.OS	h	Ordinador de Sobretaula	1	0,08	0,08
				CD	19,41
				CDC (3%)	0,58
				Cost total	20,31 €
2.6	u	Test Motive			
MO.GITI	h	Enginyer Tècnic Industrial	1	19	19
MO.MUII	h	Enginyer Industrial	1	25	25
MH.OP	h	Ordinador Portàtil	1	0,10	0,10
MS.LV	h	LabVIEW	1	0,23	0,23
MH.OS	h	Ordinador de Sobretaula	1	0,08	0,08
MS.MOT	h	Motive	1	0,34	0,34
MI.OPTI	h	Sistema Optitrack	1	1,04	1,04
				CD	45,79
				CDC (3%)	1,37
				Cost total	47,17 €

Taula 4.3. Quadre de preus descompostos del Capítol 3

Codi	Unitats	Descripció d'unitats d'obra	Rendiment	Preu	Import
3	CAP. 3	Recopilació i tractament de la informació			
3.1	u	Redacció de la memòria descriptiva i dels manuals			
MO.GITI	h	Enginyer Tècnic Industrial	96	19	1.824
MO.MUII	h	Enginyer Industrial	12	25	300
MH.OP	h	Ordinador Portàtil	96	0,10	9,82
MS.O365	h	Microsoft Office 365	96	0	0
				CD	2.133,82
				CDC (3%)	64,01
				Cost total	2.197,83 €

5. PRESSUPOSTOS PARCIALS

Taula 5.1. Pressupost parcial Capítol 1

Nº Ordre	Unitats	Descripció d'unitats d'obra	Mesura	Preu	Import
1	CAP. 1	Planificació i disseny de les funcions de la interfície			
1.1	u	Estudi d'alternatives i recerca d'informació	4	118,05	472,21
1.2	u	Reunió tècnica	4	22,66	90,64
				Total Capítol 1	562,85 €

Taula 5.2. Pressupost parcial Capítol 2

2	CAP. 2	Desenvolupament de la interfície i del mòdul *Kinematics*			
2.1	u	Programació del codi en LabVIEW	1	3.146,25	3.146,25
2.2	u	Programació del codi en LabVIEW amb MathScript	1	648,71	648,71
2.3	u	Reunió tècnica via Teams	4	22,71	90,85
2.4	u	Test estàndard	5	19,91	99,56
2.5	u	Test connexió TCP/IP	2	20,31	40,61
2.6	u	Test connexió Motive	1	47,17	47,17
				Total Capítol 2	4.073,15 €

Taula 5.3. Pressupost parcial Capítol 3

3	CAP. 3	Recopilació i tractament de la informació			
3.1	u	Elaboració dels documents del projecte	1	2.197,83	2.197,83
				Total Capítol 3	2.197,83 €

6. PRESSUPOST TOTAL

Taula 6.1. Pressupost total d'Execució Material

Capítol 1: Planificació i disseny de les funcions de la interfície	562,85
Capítol 2: Desenvolupament de la interfície i del mòdul *Kinematics*	4.073,15
Capítol 3: Recopilació i tractament de la informació	2.197,83
Total Execució Material	6.833,83 €

Taula 6.2. Pressupost total d'Execució per Contracta

Pressupost total d'Execució Material (PEM)	6.833,83
Benefici industrial (6%)	410,03
Despeses generals (13%)	888,40
Total Execució per Contracta	8.132,26 €

Taula 6.3. Pressupost total (Base de licitació)

Pressupost total d'Execució per Contracta	8.132,26
IVA (21%)	1.707,77
Total Base de Licitació	**9.840,04 €**

Ascendeix el pressupost projectat, a l'expressada quantitat de NOU MIL HUIT-CENTS QUARANTA EUROS I QUATRE CÈNTIMS.

TREBALL FINAL DE GRAU EN ENGINYERIA EN TECNOLOGIES INDUSTRIALS

MANUAL DE L'USUARI

AUTOR: Eros Iván Costa Andrés

TUTORA: Marina Vallés Miquel

COTUTOR: Rafael José Escarabajal Sánchez

Curs Acadèmic: 2019-20

ÍNDEX DEL MANUAL D'USUARI

CAPÍTOL 1. INTERFÍCIE

Aquest manual té com a finalitat guiar a l'usuari durant l'ús de l'aplicació, per aquest motiu s'explica quins tipus i quina funcionalitat té cadascun dels controls i indicadors que conté la interfície. Abans de tot, és important remarcar que l'ús d'aquesta aplicació està limitat a aquells ordinadors que disposen del sistema operatiu Windows de 32 o 64 bits. La interfície dissenyada, principalment es pot dividir en dues zones, el panell de control i el propi visualitzador dels sòlids rígids a estudiar. Aquesta distribució es present en la següent figura:

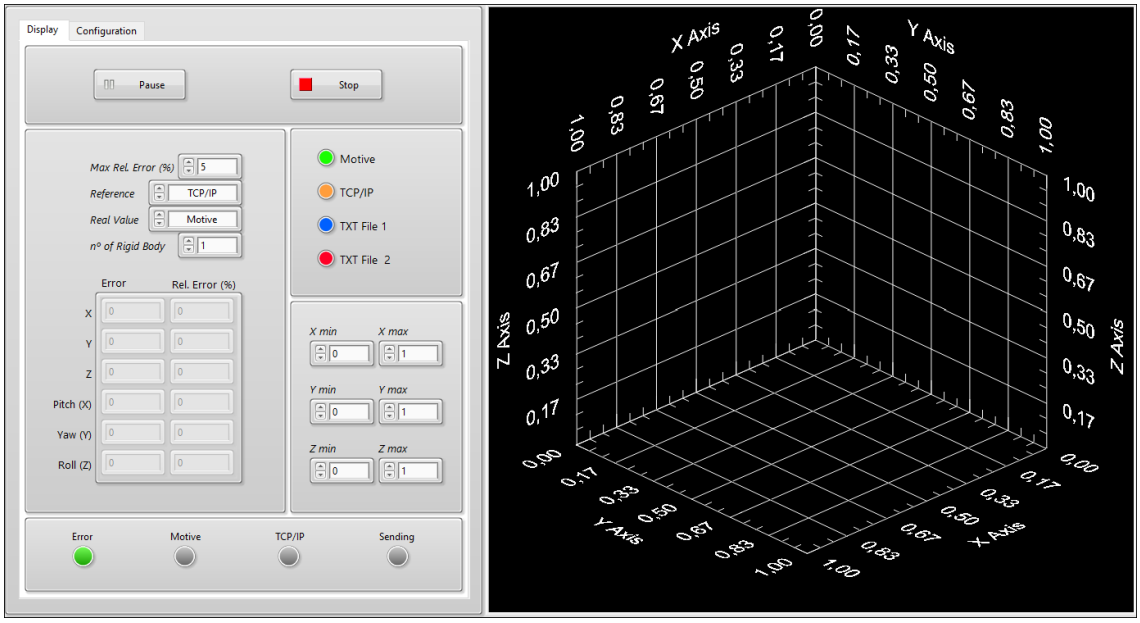

***Fig. 1*. Interfície de l'aplicació**

Com es pot observar el visualitzador es situa a la part dreta de la interfície, i en la part esquerra es troba el panell de control, el qual es divideix en dos menús diferents *Configuration* i *Display*. Cadascun fa referencia a un tipus de controls diferents, *Configuration* està enfocat a gestionar les dades de posició, mentre que *Diplay* controla aspectes relacionats amb la visualització del gràfic. Per poder canviar entre els dos menús s'ha de clicar en la finestra corresponent, ambdós es troben en la part superior esquerra de la interfície. A continuació es presenta més en detall com usar totes les funcionalitats de cadascun.

1.1. CONFIGURATION MENU

Aquesta part del panell de control s'encarrega de gestionar tota la informació sobre la posició: d'on prové, on s'envia, com i on es guarda. El menú es divideix en 4 quadrats diferents, que representes les 4 possibles fonts d'on pot provindre la informació, aquests són: el Motive, la connexió TCP/IP i dos arxius TXT. Aquestes característiques es poden observar en la figura 2, on està representat el *Configuration Menu*:

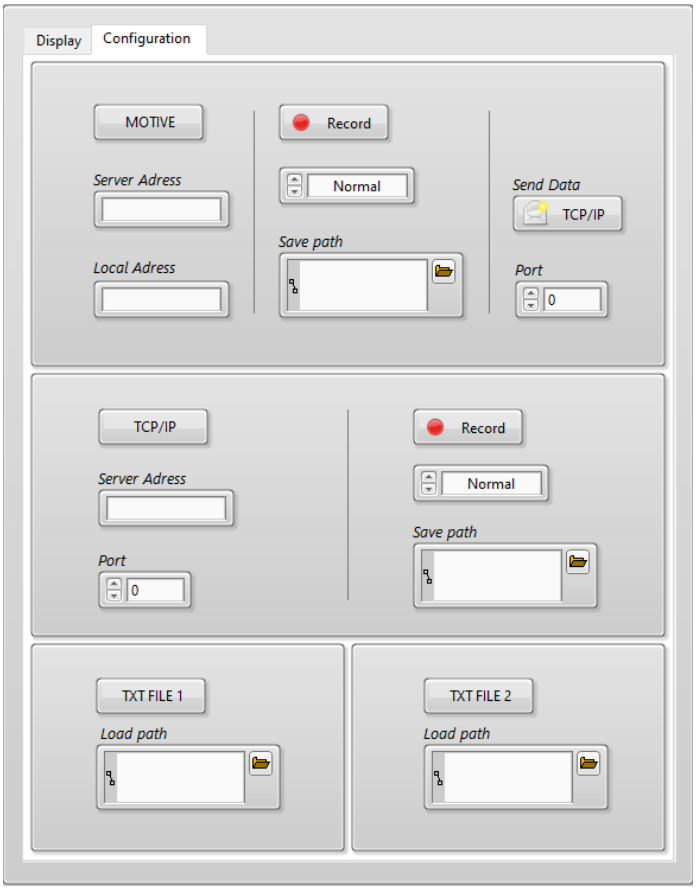

Fig. 2. Configuration Menu

Per seleccionar qualsevol d'aquestes fonts, sols s'ha de polsar el botó corresponent i emplenar les dades requerides. Tot seguit, s'explica més detalladament la informació necessària i les opcions que ofereix cada tipus.

1.1.1. Connexió amb Motive

Per connectar-se correctament al servidor Motive, simplement s'ha d'introduir la direcció IP de l'ordinador amb el Motive (*Server Adress*) i la que té la interfície en qüestió (*Local Adress*). Posteriorment, clicant en el botó Motive s'estableix la connexió i es comencen a graficar les dades.

A més, també existeix la possibilitat de grabar els instants de la trajectòria del cos que ens interessen. Sempre que el botó *Record* està actiu, l'aplicació està guardant totes les dades fins que el botó es desmarca. Per fer-ho, es requereix de la ruta i el nom del fitxer on es vol guardar l'assaig. S'ha d'introduir en *Save Path*, i es pot fer escrivint directament en l'espai blanc, o bé es pot clicar en la icona de la carpeta i navegar per l'explorador d'arxius.

L'aplicació ha estat dissenyada de forma que es pot grabar més d'una trajectòria durant una mateixa sessió. D'aquesta manera, s'ha d'introduir la ruta del nou fitxer on en vol guardar la següent trajectòria. Pel contrari, si no s'indica ninguna nova, el programa guarda la nova trajectòria en el mateix fitxer, sobreescrivint les dades anteriors que contenia.

La funció de guardar ofereix 2 formats diferents, segons les dades que es vuiguen escriure. De forma predefinida està el mode Normal, però també existeix el mode *Only RB*, que sols guarda la informació corresponent al CG dels sòlids rígids sense capçalera, per facilitar que es puga el processament de l'arxiu en altres plataformes com MATLAB, entre altres. Els dos formats es presenten en les següents figures:

```
18/7/2020 RB=1 M1=3
time         Nº ID      X        Y        Z        qx       qy       qz
19:43:15,794 1 1000000 0,007401 0,000000 0,620800 0,000000 0,135089 -0,103150
19:43:15,794 1 1000001 -0,588271 0,030890 0,580396
19:43:15,794 1 1000002 0,520143 0,208736 0,661203
19:43:15,794 1 1000003 0,273778 -0,298405 0,661203
19:43:15,802 1 1000000 0,007399 0,000000 0,620900 0,000000 0,135088 -0,103146
19:43:15,802 1 1000001 -0,588273 0,030889 0,580497
19:43:15,802 1 1000002 0,520140 0,208737 0,661303
19:43:15,802 1 1000003 0,273777 -0,298406 0,661303
19:43:15,812 1 1000000 0,007400 0,000000 0,621000 0,000000 0,135089 -0,103151
19:43:15,812 1 1000001 -0,588272 0,030891 0,580596
19:43:15,812 1 1000002 0,520142 0,208736 0,661404
19:43:15,812 1 1000003 0,273777 -0,298405 0,661404
```

Fig. 3*. Fitxer guardat en mode *Normal

```
09:16:55,590 1 1000000 1,007400 1,000000 1,635600 0,000000 0,135088 -0,103151
09:16:55,599 1 1000000 1,007400 1,000000 1,635700 0,000000 0,135089 -0,103148
09:16:55,607 1 1000000 1,007400 1,000000 1,635800 0,000000 0,135088 -0,103150
09:16:55,615 1 1000000 1,007400 1,000000 1,635900 0,000000 0,135089 -0,103146
09:16:55,623 1 1000000 1,007400 1,000000 1,635999 0,000000 0,135089 -0,103154
09:16:55,632 1 1000000 1,007399 1,000000 1,636100 0,000000 0,135089 -0,103150
09:16:55,640 1 1000000 1,007399 1,000000 1,636200 0,000000 0,135087 -0,103145
09:16:55,648 1 1000000 1,007401 1,000000 1,636300 0,000000 0,135088 -0,103150
```

Fig. 4*. Fitxer guardat en mode *Only RB

Per últim, es troba la funció que permet enviar totes les dades rebudes usant els protocols TCP/IP. Per dur-ho a terme, s'ha de marcar la opció *Send Data*, aquesta funció necessita que s'introduisca el nombre del port (*Port*) desde el que es vol enviar la informació. Aquest nombre ha d'estar contingut entre el 0 i el 50000. A continuació, es mostra en la figura 5 un exemple on s'ha establert una connexió amb el Motive, al mateix temps que estan enviant les dades.

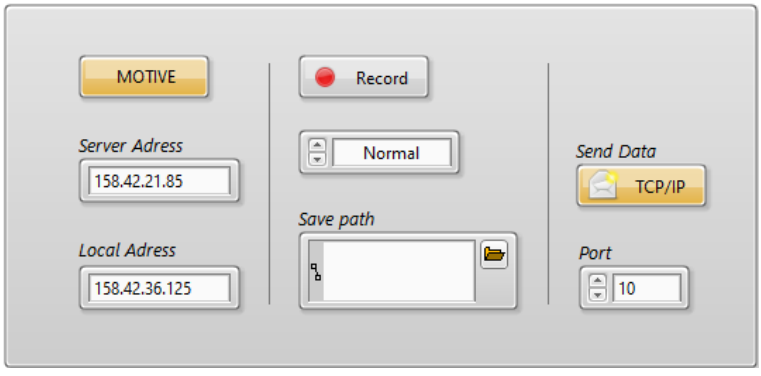

***Fig. 5*. Exemple de connexió Motive**

1.1.2. Connexió via TCP/IP

Aquesta connexió permet connectar la interfície amb un servidor del tipus TCP/IP, i poder visualitzar tota la informació que rep. Per poder establir aquesta connexió és necessari definir la direcció IP del l'ordinador que transmet les dades (*Server Adress*), i el port des del que ho està realitzant (*Port*). Seguiment, en la figura 6, és presenta un exemple de connexió TCP/IP, on s'estan guardant les dades amb el format *Only RB*:

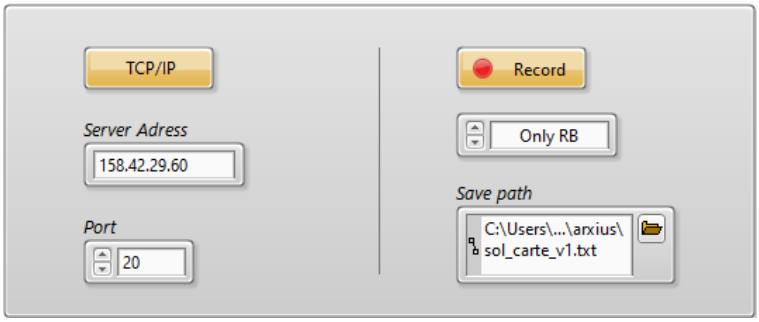

***Fig. 6*. Exemple connexió TCP/IP**

Com es mostra en la figura anterior, aquesta connexió també compta amb la funció de poder gravar una part determinada de l'assaig que es realitze. El procediment és idèntic al del Motive, comentat en l'apartat anterior.

1.1.3. Carregar fitxers TXT

Aquesta és l'última opció que trobem en el *Configuration Menu*, permet carregar arxius de tipus TXT, amb dades sobre trajectòries anteriors. Per a poder carregar sessions anteriors, els fitxers TXT han de tindre el mateix format que els arxius guardats per la interfície, és a dir, el mode *Normal*; en cas contrari l'aplicació no pot processar la informació.

La interfície ofereix 2 entrades diferents, permetent carregar simultàniament dos arxius TXT. Per fer-ho, sols s'ha d'introduir la ruta on es troba el fitxer a carregar i polsar el botó. Si quan finalitza la lectura del document, el botó TXT 1 segueix polsat, l'aplicació torna a graficar l'arxiu de nou des de el principi. En la figura següent es pot observar un exemple:

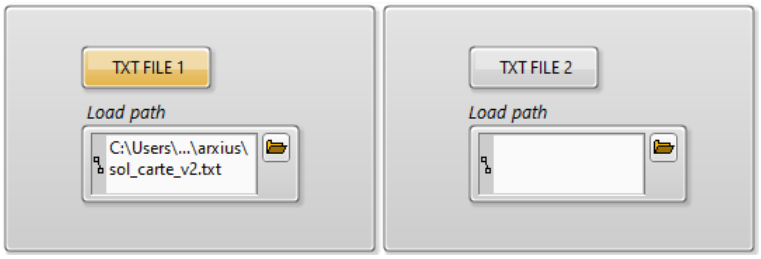

***Fig. 7*. Exemple de carregar un fitxer**

1.2. DISPLAY MENU

Aquesta segona part del panell de control és responsable de controlar diferents aspectes del visualitzador com: el color de cada sòlid rígid, el rang dels eixos XYZ, l'error, el botó *Pause* i *Stop*. El menú es divideix en 5 quadrats, cadscun referit a una funcionalitat diferent, trobem: els botons de control, els colors, l'error, el rang del eixos i els indicadors LED. En la figura 8 es mostra el menú *Display* complet, on es pot observar la distribució de les diferents funcionalitats, les quals s'expliquen a continuació.

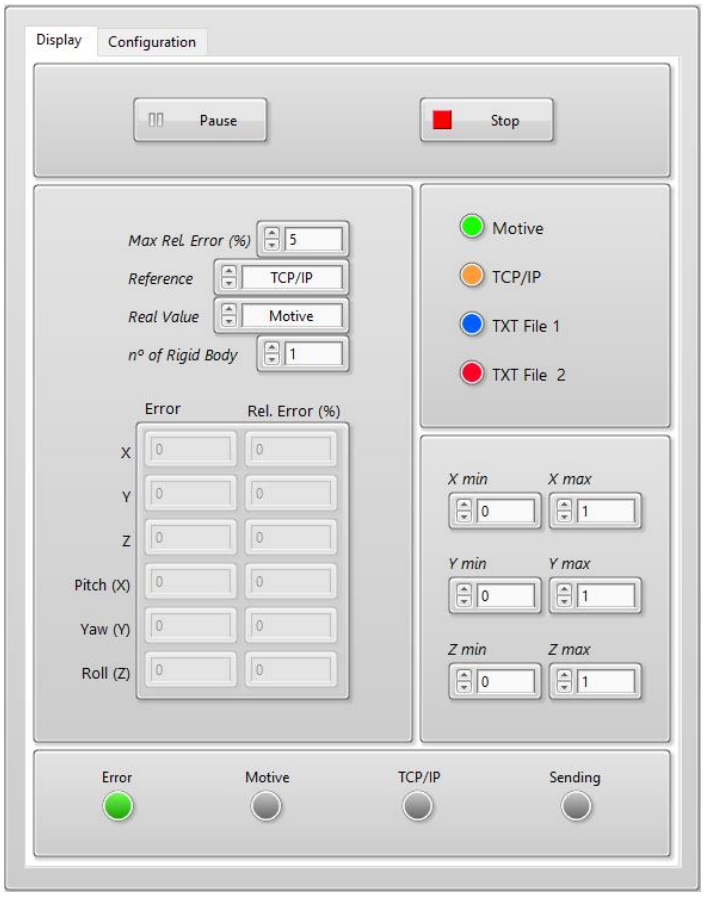

***Fig. 8*. Display Menu**

1.2.1. Controls

En aquest menú també trobem el botons de *Pause* i *Stop*, que controlen l'execució de l'aplicació. El *Pause* congela la visualització de la trajectòria dels cossos que s'estaven representant fins que aquest es desmarca. Quan ocorre, la representació continua en el mateix punt on s'havia quedat, per al cas dels fitxer TXT, i torna a mostrar-se en temps real les dades que li arriben, per al cas del Motive i TCP/IP.

Fig. 9*. Botons *Pause* i *Stop

Altre control d'aquest mFronenú és el color de cada sòlid rígid que es grafica. La interfície permet canviar, a priori, el color que tindrà cada cos. Per fer-ho sols s'ha de polsar sobre el cercle de color del sòlid que volem canviar, i es mostra una paleta on s'ha d'elegir el nou color.

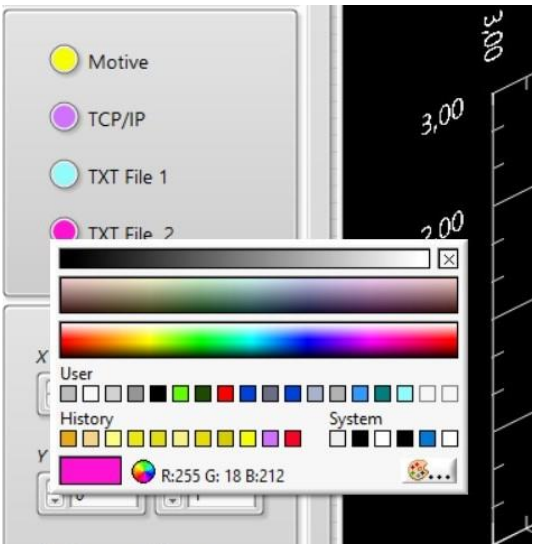

***Fig. 10*. Canvi de color dels cossos a graficar**

Finalment, tenim el control del rang de representació del visualitzador. Inicialment, el rang de cada eix varia entre 0 i 1, però es pot canviar l'escala en qualsevol moment de la visualització. S'aconsegueix introduint directament el valor amb el teclat, o bé polsant les fletxes fins a arribar al valor desitjat.

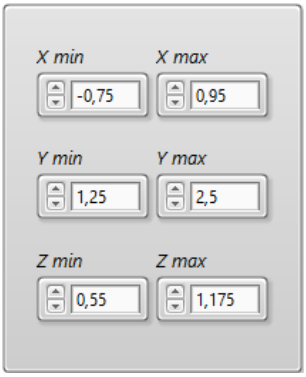

***Fig. 11*. Control del rang dels eixos XYZ**

1.2.2. Indicadors LED

Els indicador estan situats en la part inferior del menú i proporcionen informació instantània sobre quines connexions estan actives en cada moment. Existeix un LED per a cada tipus de connexió possible: Motive, TCP/IP com a client (rebent dades) o TCP/IP com a servidor (enviant) Mostren un color gris, quan no existeix cada connexió, que canvia a groc quan està activa.

A més es disposa d'un quart LED que indica si l'error ha excedit el valor definit, canviant el color a roig. Aquest fet es pot observar en la figura 12, on existeix una connexió activa amb el Motive i TCP/IP, al mateix temps que l'error ha excedit el màxim introduït per l'usuari:

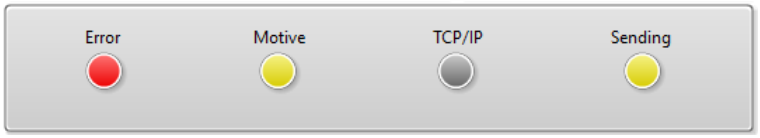

***Fig. 12*. Indicadors LED**

1.2.3. Error

Per últim, el menú proporciona un visualitzador de l'error en temps real, mostrant l'error amb el seu signe i com a error relatiu. L'usuari ha d'introduir el valor d'error màxim acceptable, en percentatge, perquè el sistema compta amb una alarma sonora i un indicador LED, que avisen quan es supera aquest valor definit.

Altre aspecte que s'ha d'elegir és entre quins sòlids rígids es vol calcular, de forma predefinida la interfície té el del TCP/IP com a valor de referència i el del Motive com a valor real, però es pot canviar en qualsevol moment. El mateix ocorre amb el sòlid rígid del que es vol mostrar, per al cas on existeixen dos, es pot elegir l'error de quin es vol visualitzar.

Tot seguit, en la figura 13, es mostra un exemple de selecció de l'error entre les dades del TCP/IP i un fitxer carregat en el TXT 1 del sòlid rígid nº1. L'error màxim relatiu que s'ha definit en aquest cas es del 25%.

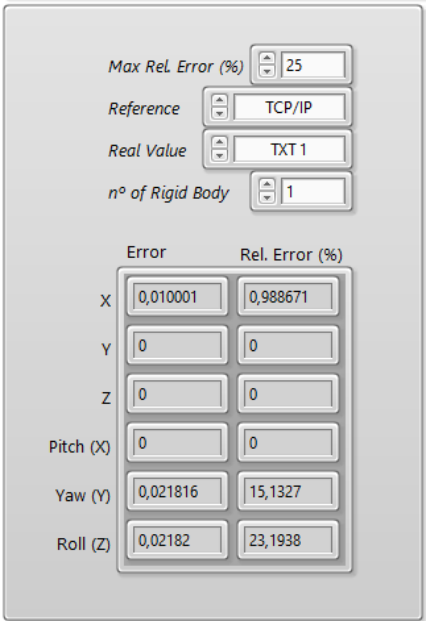

***Fig. 13*. Exemple de l'error**

CAPÍTOL 2. MÒDUL *KINEMATICS*

Aquest segon capítol del manual, introdueix a l'usuari en l'ús del mòdul *Kinematics* desenvolupat, explicant la funcionalitat tenen cada control i indicador que conté la interfície, així com el procediment d'emplenament dels paràmetres d'entrada al model.

La interfície del mòdul es pot dividir en dues zones, el panell dels inputs, i el dels outputs i els controls. Aquesta distribució es pot observar en la següent figura:

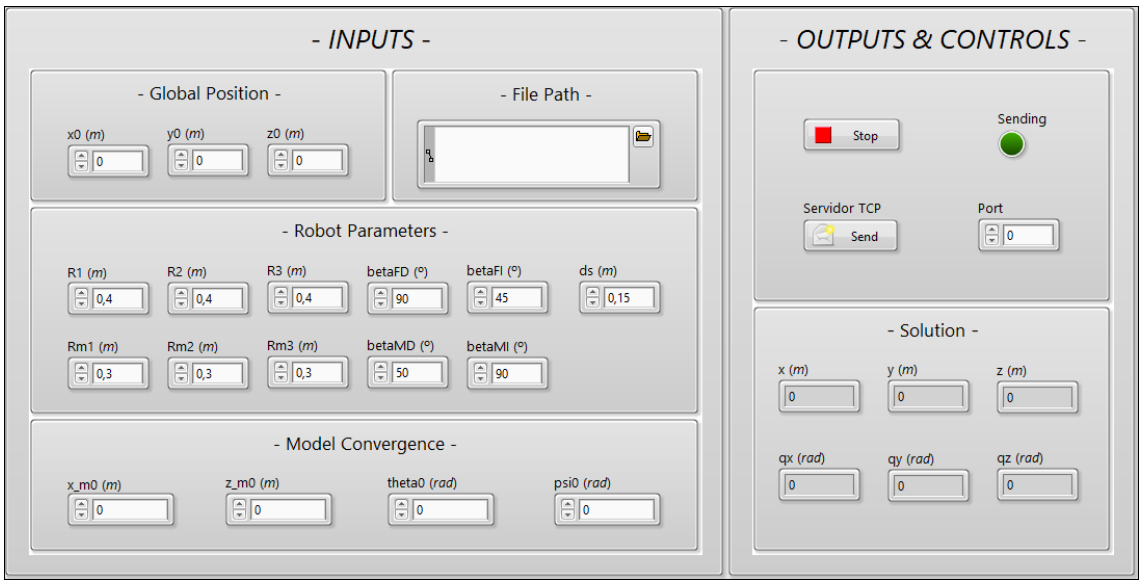

Fig. 14. Front Panel* del mòdul *Kinematics

2.1. PANELL DELS INPUTS

En aquest apartat es descriu el panell dels inputs, el qual s'encarrega de transmetre totes les dades necessàries que necessita el mòdul per al correcte funcionament. Existeixen 4 tipus diferents de controls, dividits per la finalitat de cadascun, són: *Global Position*, *File Path*, *Robot Parameters* i *Model Convergence*. En tots aquests, es troben indicades les unitats del valor d'entrada, a la dreta i entre parèntesis. Les funcions dels diferents tipus de controls es presenten més detalladament ens el següents apartats.

Aquests 4 tipus de controls, es troben agrupats al panell formant 4 quadrats independents. Cadascun dels qual té el títol i els controls als que fa referència. La forma del panell es mostra, més en detall, en la següent figura:

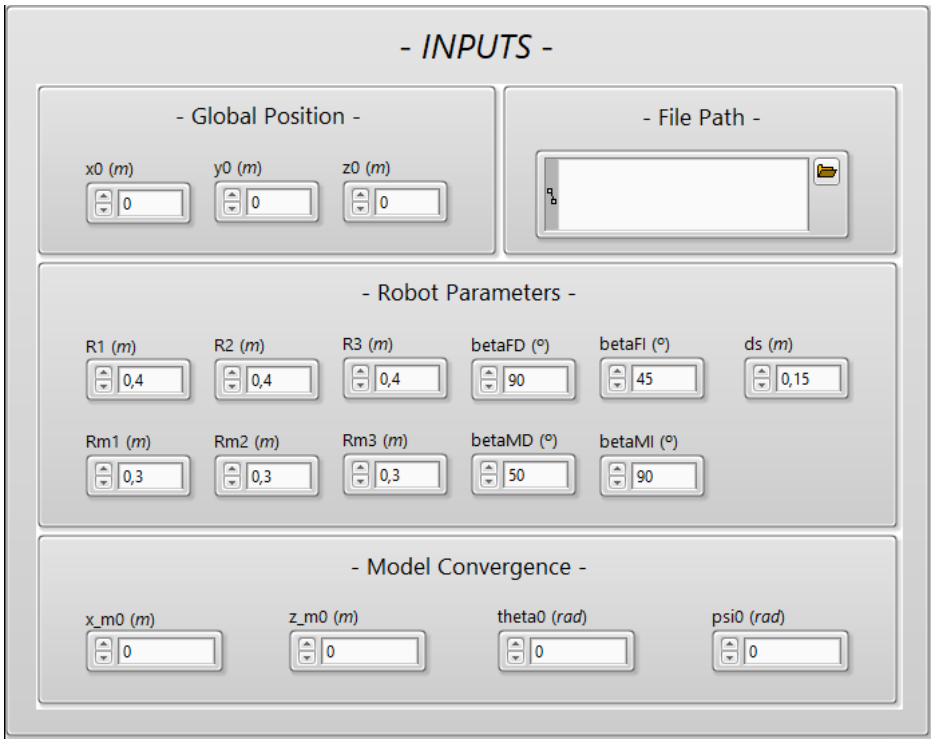

***Fig. 15*. Panell dels Inputs**

2.1.1. Global Position

En aquests controls s'han d'introduir els valors de la posició inicial de la plataforma. Degut a que el sensors són incrementals, el mòdul *Kinematics* requereix de la posició global, per poder així sumar-la a la relativa que es calcula usant el model. Tot seguit, es pot observar una posició global exemple:

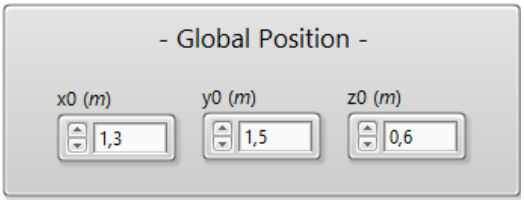

Fig. 16*. Exemple del *Global Position

2.1.2. File Path

Es tracta del control on s'ha d'indicar la ruta de l'arxiu, tipus TXT, que conté les mesures dels sensors incrementals. A partir de les dades d'aquest fitxer, el mòdul va calculant les successives posicions que pren la plataforma. En la figura 17 es mostra l'exemple de la ruta:

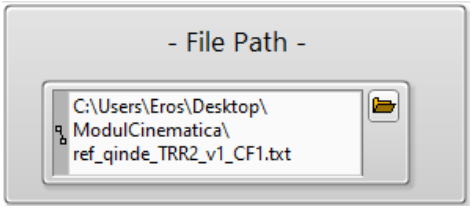

Fig. 17*. Exemple del *File Path

2.1.3. Robot Parameters

Aquests controls, són els paràmetres geomètrics corresponents a la configuració del robot en la que s'estiga treballant, els quals defineixen la forma que tindran les plataformes del robot.

De forma predefinida estan els valors geomètrics de la configuració nº1, però es poden canviar en qualsevols moment que es necessite. Seguiment, a la figura 18, es troben els valors per defecte, presentats en la Taula 2 de la Memòria Descriptiva:

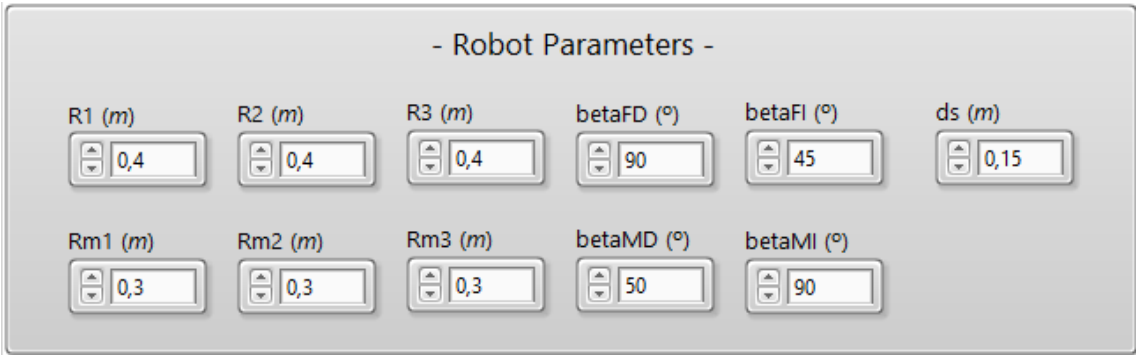

***Fig. 18*. Exemple dels Robot Parameters**

2.1.4. Model Convergence

Els últims controls del panell dels Inputs es corresponen a la convergència del model cinemàtic. Aquests, són els valors inicials propers a la solució que necessita el model per convergir, i així trobar la posició teòrica del CG de la plataforma. A continuació, en la figura 19, estan representats uns valors exemple.

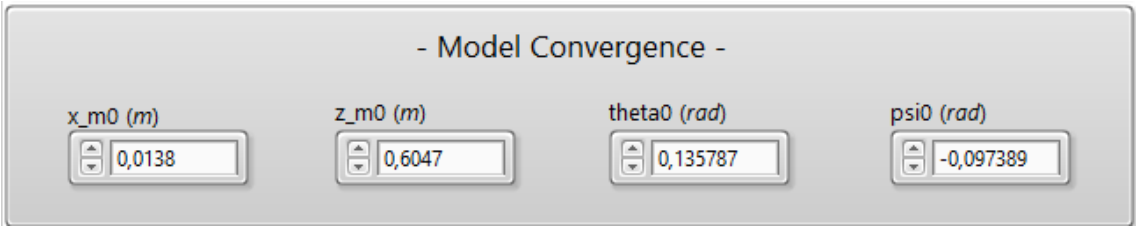

Fig. 19*. Exemple del *Model Convergence

2.2. PANELL DELS OUTPUTS I DELS CONTROLS

Aquest és el segon panell del mòdul *Kinematics*, controla el funcionament de l'aplicació i mostra els resultats obtinguts del model cinemàtic. Es troba dividit en dues parts: els Controls, en la part superior, i els Outputs, en la inferior; tal i com es pot veure en la figura 20:

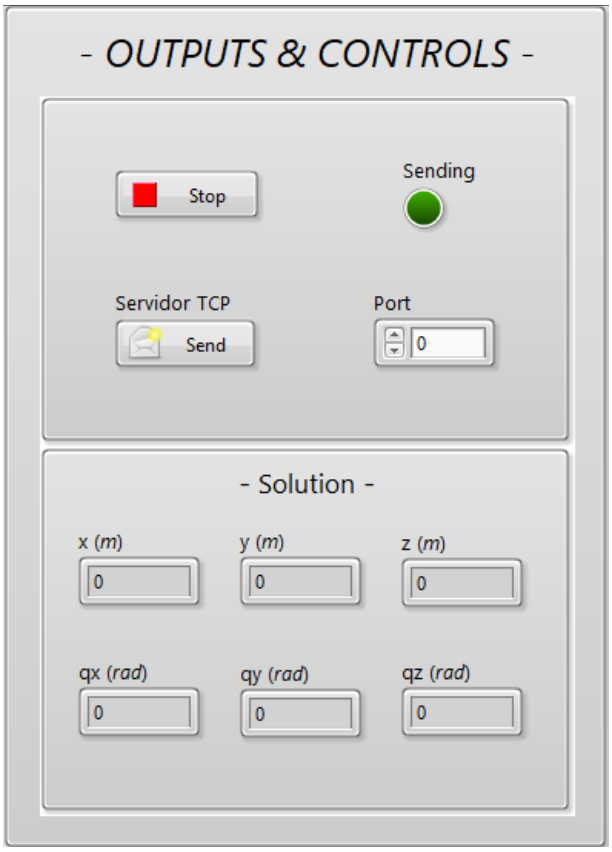

***Fig. 20*. Panell dels Outputs i dels Controls**

2.2.1. Controls

Primerament, es troben agrupats el controls del funcionament del mòdul. Està el botó de *Stop*, encarregat del finalitzar el tractament de dades, i el botó *TCP Server*. Respecte d'aquest últim, també es troba el control del port, des del que volem transmetre les dades via TCP/IP, així com un indicador LED indicant que aquesta comunicació s'està produint.

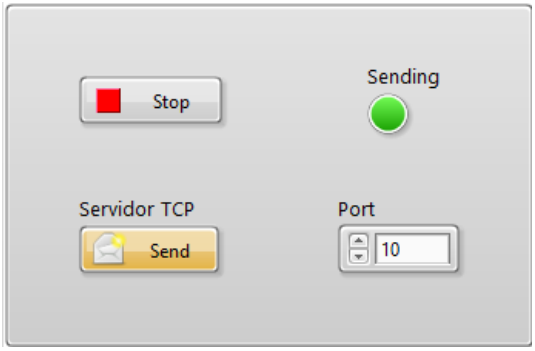

***Fig. 21*. Exemple dels Controls**

2.2.2. Outputs

Per últim, la segona part del panel es correspon als Outputs del mòdul, és a dir, a la posició final de la plataforma. Encara que el mòdul també calcula 3 punts més d'aquesta, els indicadors sols mostren la posició i orientació del CG de la plataforma. Tot seguit, en la figura 22, es poden observar uns valors de solució possibles:

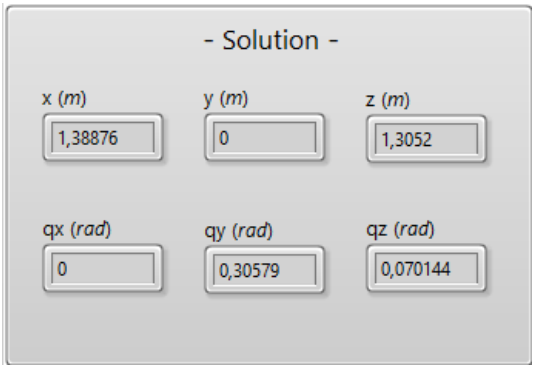

***Fig. 22*. Exemple dels Outputs**

TREBALL FINAL DE GRAU EN ENGINYERIA EN TECNOLOGIES INDUSTRIALS

ANNEX: MANUAL DEL PROGRAMADOR

AUTOR: Eros Iván Costa Andrés

TUTORA: Marina Vallés Miquel

COTUTOR: Rafael José Escarabajal Sánchez

Curs Acadèmic: 2019-20

ÍNDEX DE L'ANNEX: MANUAL DEL PROGRAMADOR

CAPÍTOL 1. APLICACIÓ PRINCIPAL

L'objecte d'aquest capítol és exposar detalladament el codi corresponen a la interfície desenvolupada, i dels mòduls i submòduls que la formen. Primerament, es presenta el codi principal del mòdul de l'aplicació. Aquest és l'encarregat de coordinar i gestionar l'actuació dels diferents subVIs o mòduls, que fan possibles totes les funcions que posseeix el programa. El comportament d'aquest en cada moment depèn de les opcions que l'usuari haja elegit. D'aquesta forma, per a poder oferir un bon funcionament, aquesta part del codi ha de transmetre eixa informació en tot moment als diferents mòduls. Per a facilitar l'explicació, és subdividirà el codi en dues parts, una es centrarà en l'actualització de variables i indicadors, i l'altra al control dels diferents mòduls. Ambdues parts es poden veure en les figures 1 i 2, respectivament.

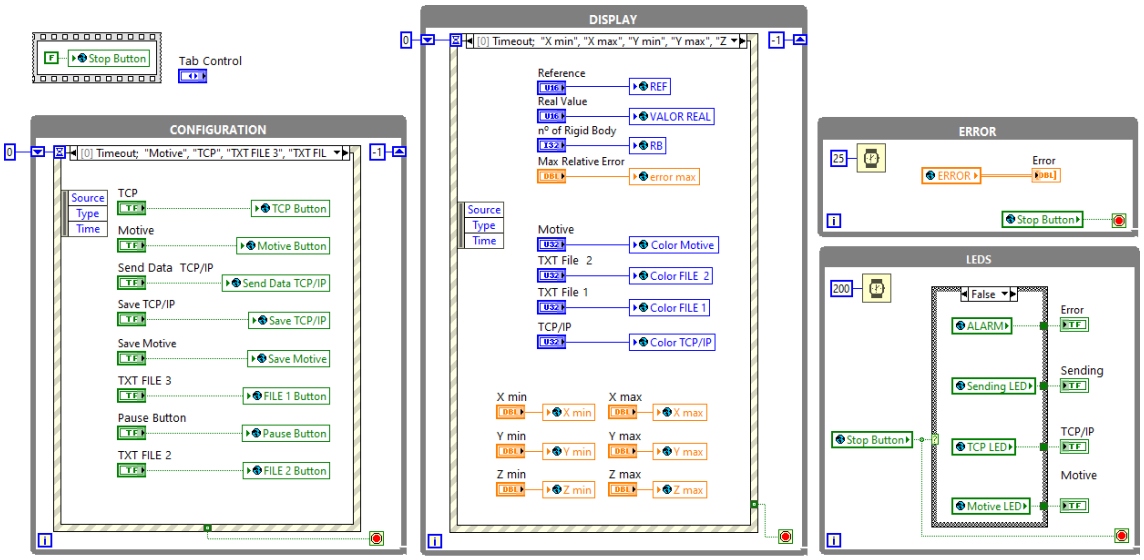

Fig. 1. **Codi Principal 1: Actualització de controls i indicadors**

La primera part del codi, és l'encarregada d'actualitzar en tot moment les opcions seleccionades per l'usuari, que determinaran el funcionament de l'aplicació. Com s'observa en la figura 1, totes les variables de control disponibles s'actualitzen en les dos *Even Structures*, situades en la part esquerra. S'han dividit en dos diferents, les corresponents a la finestra del *Display* i les de la *Configuration*. La raó principal és minimitzar les vegades que s'escriu en la variables, ja que cada vegada que alguna es polsada en el *Front Panel*, s'actualitza el valor de totes les que es troben en la mateixa *Even Structure*.

S'han elegit aquest tipus d'estructura perquè així es minimitza al màxim el temps que el processador passa en aquesta part del codi. És un funcionament equivalent a l'*Interrupt Handler*, és a dir, sols s'executa aquesta estructura quan es polsa o es canvia el valor per part de l'usuari.

La tècnica contraria és el *time polling*, que consistiria en posar directament les variables sobre un bucle *While*, fent que cada cert temps la CPU comprove i escriga el valor de cada variable, sense que s'haja produït ningun canvi.

Per aquest motiu, s'han de seleccionar totes les variables situades dins del la estructura com a *Events*, per tal de que l'estructura reaccione davant de qualsevol canvi en el seu valor. Per fer-ho, s'ha de clicar el botó dret damunt de l'*Even Structure,* seleccionar *Edit Events Handled by This Case...* i elegir les variables corresponents. Si es va a necessitar més d'una execució de l'estructura durant la mateixa execució del programa s'ha de situar dins d'un bucle *While*, ja que pel contrari sols s'executa una vegada.

Els restants dos bucles, situats a la dreta, són els responsables d'actualitzar els indicadors. El de baix s'encarrega dels LEDs del *Display*. S'ha elegit una freqüència d'actualització de 200ms, perquè no es necessària més velocitat en aquesta funció, i així s'allibera més capacitat del processador. El bucle superior actualitza els valors dels errors en el *Front Panel*, ho fa cada 25ms per la mateixa raó exposada anteriorment.

Per últim, podem observar que dalt del tot es troba una *Flat Sequence Structure*, amb una *Frame* solament. El motiu és que el primer que s'ha d'executar és la inicialització a *false* del valor de la variable *Stop*. La raó és que del contrari el compilador de LabVIEW podria començar per qualsevol altra part del codi, i llegiria com a *true* la variable *Stop* (ja que fou l'últim valor que va tindre abans de finalitzar la sessió anterior). Aquest comportament generaria un mal funcionament de la interfície.

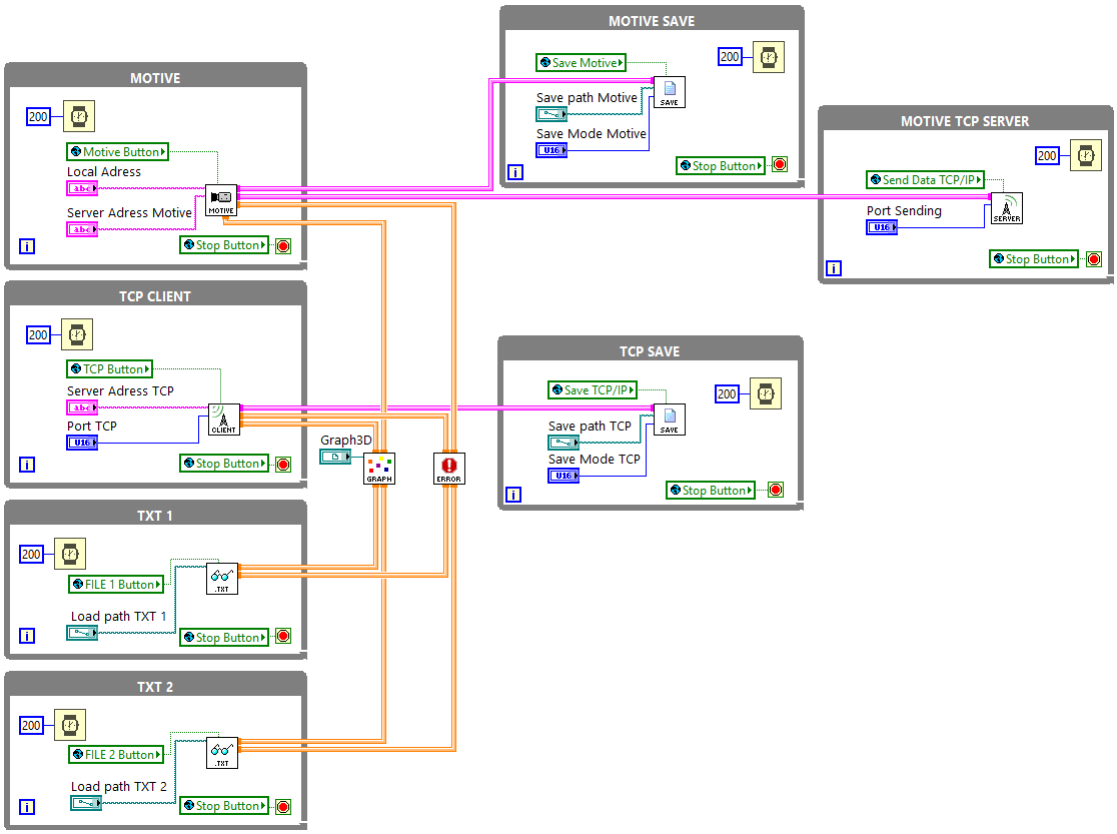

Fig. 2. **Codi Principal 2: Control dels mòduls**

Aquesta segona part del codi, s'encarrega de controlar els diferents mòduls que intervenen. Com es pot observar en la figura 2, existeixen 4 bucles productors d'informació i 5 bucles consumidors d'aquesta. Cadascun d'ells s'executa quan la variable corresponent d'activació (*Motive Button, TCP Button*...) es torna *true*, a excepció del mòdul *Graph* i *Error Calculator*. Aquests sempre estaran actius, la causa és que independentment de les opcions que l'usuari seleccione, sempre serà necessari visualitzar la trajectòria i calcular l'error. Els diferents mòduls productors i consumidors es poden observar en les figures 3 i 4, respectivament:

Fig. 3. **Mòduls productors (Motive, *TCP Client*, *Read TXT 1* i *Read TXT 2*)**

Fig. 4. **Mòduls consumidors (*Graph*, *Error Calculator*, *TCP Server*, *Motive Save* i *TCP Save*)**

Cada mòdul està situat dins d'un bucle *While* que comprova cada 200ms l'estat de la variable d'activació. Per eixir del bucles, s'ha de polsar el *Stop*, deixant així de llegir el valor de les variables d'activació. A més, aquesta part del codi transmet als mòduls informació necessària per realitzar cada funció, com les adreces IP, els ports TCP o la ruta de l'arxiu TXT a carregar. A la paleta *Functions > Programming > Structures* es on podem trobar els bucles *For*, *While*, les *Case* i *Event Structure*, i també la *Flat Sequence*.

El codi mostra quines connexions existeixen, i com flueix la informació, entre els diferents mòduls. S'observa que totes les dades viatgen, sense excepció, als mòduls *Graph* i *Error Calculato*r. D'aquesta forma, aquesta és l'única ruta per a la informació de dels fitxers TXT. El *TCP Client* utilitza aquesta ruta, i també les pot enviar al mòdul *TCP Save*. Per últim, en el cas de Motive, a banda d'enviar-les al propi *Motive Save*, també les pot enviar al mòdul *TCP Server*, capaç d'enviar tota la informació usant aquests protocols.

Els canals de comunicació entre els diferents mòduls, es tracten de *High Speed Stream channels*. Es tracta d'un tipus de canal *Stream*, preparat per suportar major velocitats, a costa de prescindir d'algunes funcionalitats. És un canal que s'assegura que no hi hagen pèrdues en l'enviament, ja que disposa d'un buffer, limitat o no segons elegisca l'usuari. Per crear-los s'ha de polsar el botó dret sobre el cable de dades que es vuiga enviar, i *Create > Channel Writer > High Speed Stream channel*.

El funcionament és molt paregut a les *Queue Operations*, però més fàcils d'implementar, ja que s'evita haver d'obrir i destruir les cues que es creen, a més de teòricament poder treballar amb majors velocitats de transmissió de dades. Per aquests motius, s'ha seleccionat com la millor opció per cobrir aquesta funció, degut a que es treballa amb altes velocitats de dades, perquè s'entreguen nous paquets cada 8,33ms.

Finalment, destacar que es transmeten dos formats de dades diferents, segons la informació viatja a *Graph* i *Error Calculator*, o a qualsevol altre. El motiu és reduir el temps d'execució del mòdul *Graph*, ja que és la part més crítica temporalment. El format de dades que li arriba és simplement una matriu de dades on la primera filera es correspon al vector N, les següents a la

matriu RB i posteriorment la matriu M. Aquest format de dades es diferencia per la rapidesa i facilitat a l'hora d'accedir a la informació front a l'altre format. També és el format del mòdul *Error Calculator*, perquè s'adequa a les seues demandes, ja que tampoc requereix de l'hora exacta de la mostra (el *TimeCode*) per calcular l'error.

L'altre format usat per la transmissió de dades es tracta d'un clúster de dades, està format per: *TimeCode* o l'hora exacta de la captura en format *string*, el vector N de tipus *int*, i de forma separada les matrius RB i M, ambdós de tipus *float*. Aquest format, serveix per organitzar de forma més ordenada, i costosa, de les dades per als mòduls que obligatòriament requereixen del moment exacte de quan s'ha pres la captura de moviment.

1.1. VARIABLES GLOBALS

La majoria de variables que s'usen en el codi principal de l'aplicació es tractes de variables globals. La causa és que són l'únic tipus de variable que existeix capaç de comunicar-se entre els diferents mòduls o subVIs. Es tracten d'un tipus de fitxer VI, similars al subVIs, però solament amb el *Front Panel* perquè sols són capaços d'emmagatzemar i canviar el valor d'aquells controls o indicadors que el formen. Aquests fitxers, no disposen de *Block Diagram*, és a dir, no tenen cap codi ja que sols escriuen o llegeixen el valor de les variables que contenen.

Per poder crear noves variables globals s'ha d'accedir amb el botó dret a *Functions > Programming > Structures > Global Variable*. En aquest moment es crea automàticament el nou *Front Panel* que tindrà la variable, on es col·loquen els elements d'interès. En tot el codi de la interfície s'usen un total de 5 variables globals diferents. A continuació, es presenta cada tipus, on intervenen i els elements que les formen.

1.1.1. Global configuration

La primera variable es correspon a *Global Configuration*, la seua estructura està representada en la figura 5. És la variable més important de tota l'aplicació, perquè conté totes les variables booleanes d'activació de cada mòdul, així com els controls de *Pause* i *Stop.* Intervé principalment en el panell de configuració de la interfície, per això el seu nom, encara que l'*Stop* i el *Pause* estan en el menú *Display*. Les variables s'escriuen en el codi principal, i es lligen cadascuna en el mòdul que li correspon, per avaluar si han d'estar o no actius.

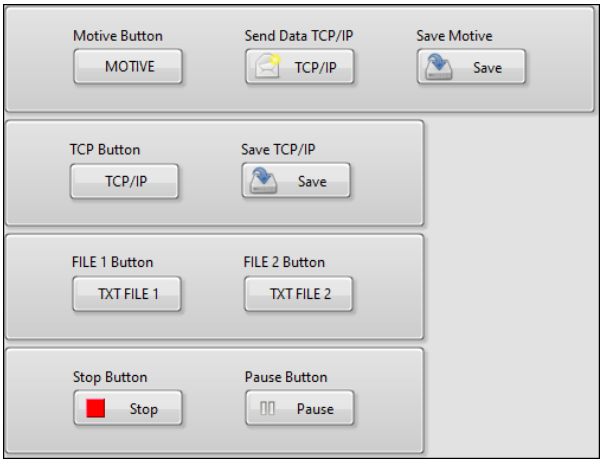

Fig. 5. **Variable *Global Configuration***

1.1.2. Global Graph

La segona variable s'anomena *Global Graph*, és l'encarregada d'emmagatzemar les dades corresponents al funcionament exclusiu del gràfic. Els elements que el formen es poden observar en la figura 6. Controla l'escala dels eixos XYZ del gràfic, així com el color de cada sòlid rígid que es visualitze. Les diferents variables s'escriuen el codi principal, com la resta en les *Even Structures*; i es lligen dins del mòdul *Graph*.

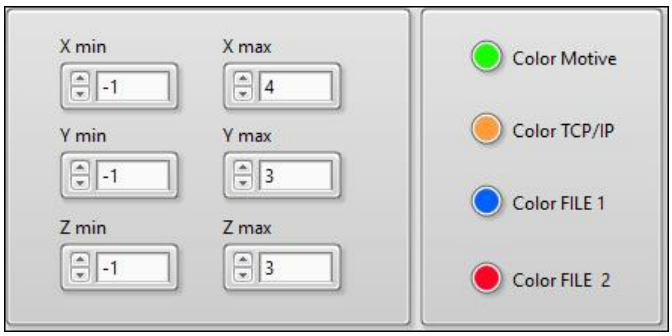

Fig. 6*. Variable *Global Graph

1.1.3. Global LEDs

La següent variable és la *Global LEDs*, la seua funció és transmetre l'estat de les connexions: al Motive, al servidor i client del tipus TCP/IP. La seua estructura s'observa en la figura 7, on es representen les diverses variables booleanes que mostren l'estat de cada canal de comunicació. L'estat de cadascuna pot ser activa o inactiva, s'actualitza dins del mòdul corresponent, i es llig en el codi principal de l'aplicació per mostrar-lo en el *Display* del *Front Panel*.

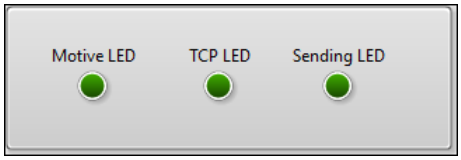

Fig. 7*. Variable *Global LEDs

1.1.4. Global Error Control

Seguidament, trobem la variables *Global Error Control*, encarregada d'emmagatzemar els valor necessaris pel funcionament del mòdul *Error Calculator*. Els diferents elements que el formen estan representat en la figura 8. Aquesta variable conté: quin element serà la referencia, quin el valor real, quin sòlid rígid es vol mostrar i l'error relatiu màxim que s'accepta. Totes aquestes dades s'obtenen de l'usuari que les elegeix en el *Display* de la interfície, i es traslladen al submòdul que calcula l'error.

***Fig. 8*. Variable Global Error Control**

1.1.5. Global Error Indicator

Per últim, es troba la variable *Global Error Indicator*, la seua estructura es pot observar en la figura 9. Està formada per un vector de 2D (equivalent a una matriu) que emmagatzema l'error normal i relatiu, i un booleà que indica si s'ha sobrepassat el valor definit per l'usuari. El funcionament es molt similar a l'anterior. La diferència es que ho fan de forma inversa, és a dir, aquesta variable s'escriu en el submòdul *Error*, i es llig en el codi principal, per poder representar els valors en el *Display* de la interfície.

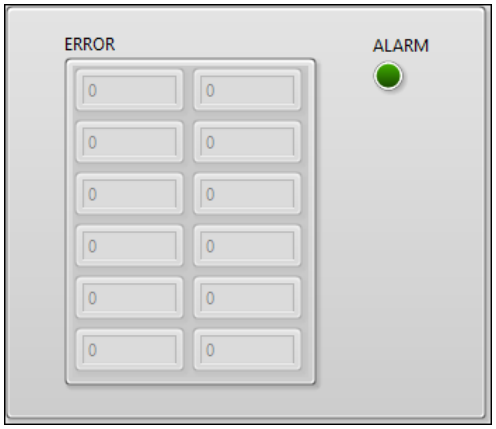

Fig. 9*. Variable *Global Error Indicator

1.2. MÒDULS DEL PROGRAMA

La programació de les diferents funcionalitats que presenta la interfície s'ha realitzat mitjançant diferents mòduls. El motiu és la sostenibilitat del codi, ja que d'aquesta manera es facilita la replicació de les parts del codi que interessen, així com la comprensió d'aquest. En futures modificacions o en el desenvolupament de nous programes, sols s'han copiar directament els mòduls que realitzen les funcions desitjades, les quals es van a presentar de cadascun ens els següents apartats.

A banda de la pròpia funcionalitat específica de cada mòdul o submòdul, s'han desenvolupat tots seguint, dins del màxim possible, la mateixa programació d'algunes de les funcions pròpies del LabVIEW. Aquestes incorporen el codi del control de l'error, a més del que correspon a la funció que realitzen. En les figures 10 i 11, es pot veure el codi corresponent a la funció *Compare Two Paths* que ofereix el LabVIEW.

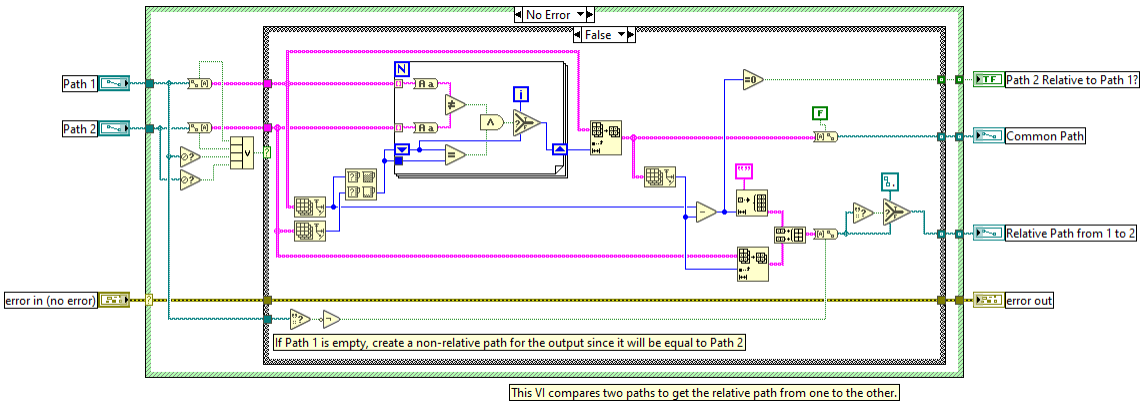

***Fig. 10*. Codi del *false Case (No Error)* de la funció *Compare Two Paths.vi* del LabVIEW**

Aquesta funció es pot trobar en *Programming > Functions > File I/O > Adv File Funcs* . S'ha programat instal·lant en primer lloc, un *Case Structure*, que comprova l'estat del canal de l'error. Com es pot observar en les figures la funció es comporta de forma diferent segons s'haja produït un error anterior o no. En cas de aquest existisca, el mòdul no realitza les seues funcions normals, i sols s'encarrega de transmetre l'error fins l'eixida. En cas contrari, s'executa el codi normal, i el mòdul realitza les funcions per a les que ha estat dissenyat.

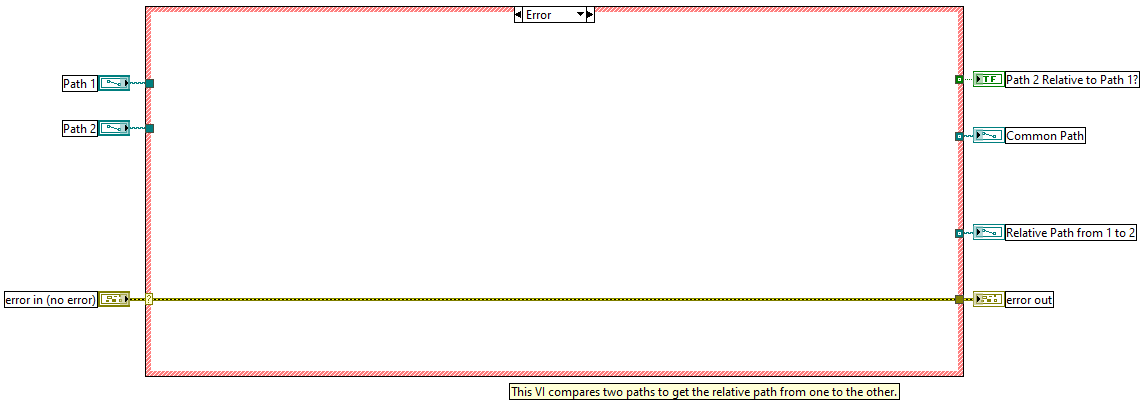

***Fig. 11*. Codi del *true Case (Error)* de la funció *Compare Two Paths.vi* del LabVIEW**

Aquesta és la programació en la que s'ha basat el codi de tots els mòduls i submòduls desenvolupats, a excepció d'alguns del *Graph*, que al tractar-se del mòdul més crític amb el temps d'execució, s'han simplificat per intentar augmentar la velocitat de processament. Abans d'explicar cadascun en detall, es presenta en la figura 12, un esquema de la jerarquia que posseeix el codi de la interfície desenvolupada:

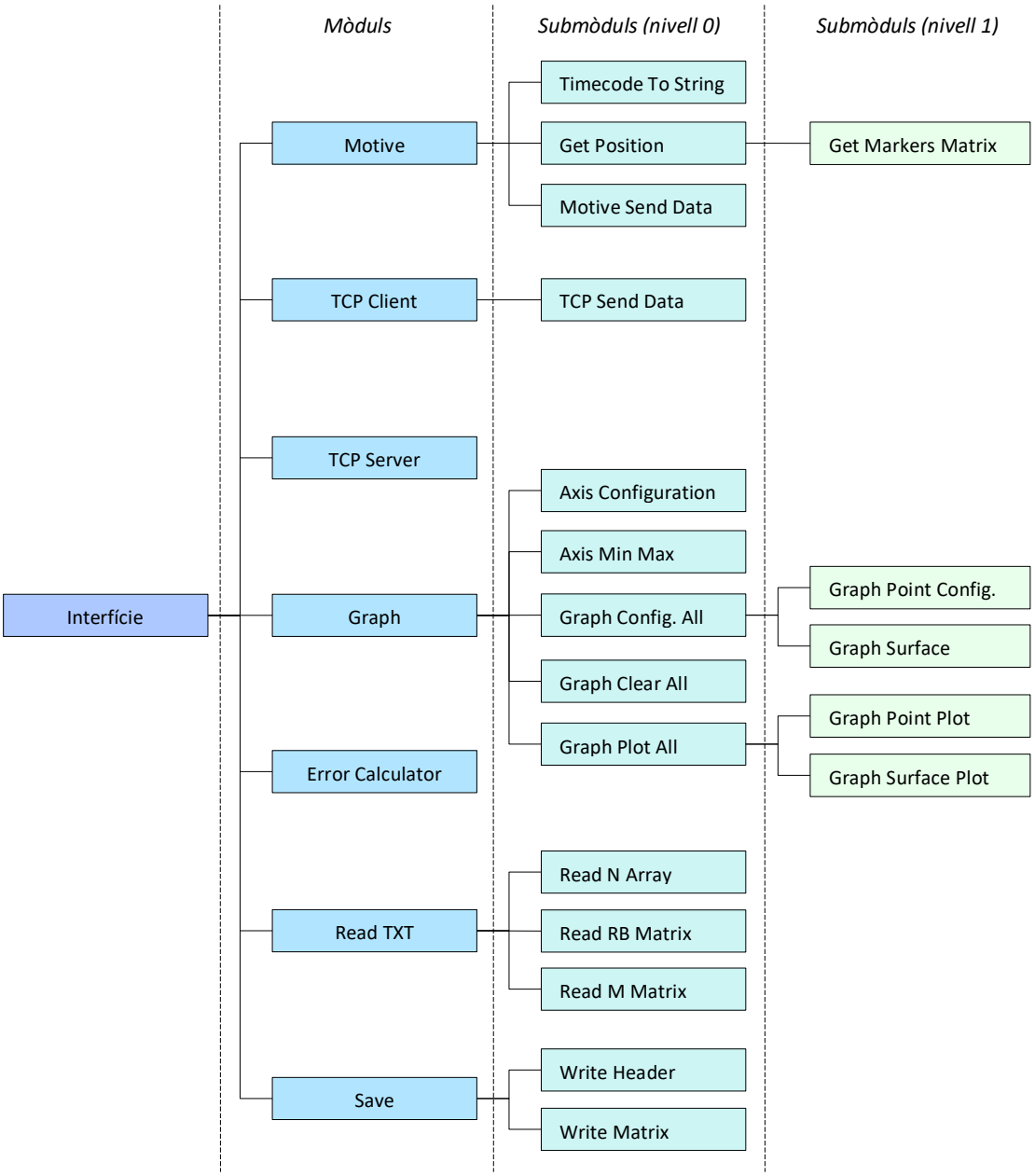

***Fig. 12*. Jerarquia dels mòduls i submòduls de la interfície**

1.2.1. Motive

Aquest mòdul crea el canal de connexió mitjançant la tecnologia .NET, entre l'aplicació i el software Motive. A través d'aquest, s'aconsegueix accedir a les dades de posició dels marcadors i dels sòlids rígids que entreguen les càmeres. Al mateix temps que li arriba tota aquesta informació, s'encarrega d'enviar-la als mòduls pertinents. En la figura 13 podem observar la icona i les connexions o paràmetres d'entrada i eixida del mòdul:

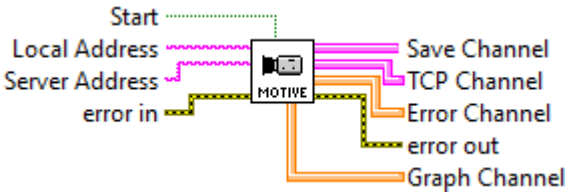

***Fig. 13*. Icona i connexions del mòdul Motive**

En la figura 13 podem veure que els paràmetres d'entrada que necessita són: el booleà *Start* que indica si s'ha d'activar o no, la direcció IP local i la del servidor del Motive, i el canal d'error (si es que existeix). D'altra banda, els paràmetres d'eixida del mòdul són la informació sobre la posició dels sòlids que es mana per quatre canals diferents, segons viatgen al mòdul *Motive Save*, *TCP Client, Error Calculator* o *Graph*. També, torna a tindre un canal d'error d'eixida, per si s'ha produït algun duran el desenvolupament de les seues funcions.

La tecnologia .NET es basa en una programació orientada a objectes, per aquest motiu el codi del mòdul ha d'accedir a certes funcions i atributs per obtenir les dades d'interès. En la figura 14 es presenta el procediment seguit per accedir a la informació necessària. Bàsicament s'han d'escriure o llegir propietats, i executar funcions de diferents objectes per poder aconseguir-ho.

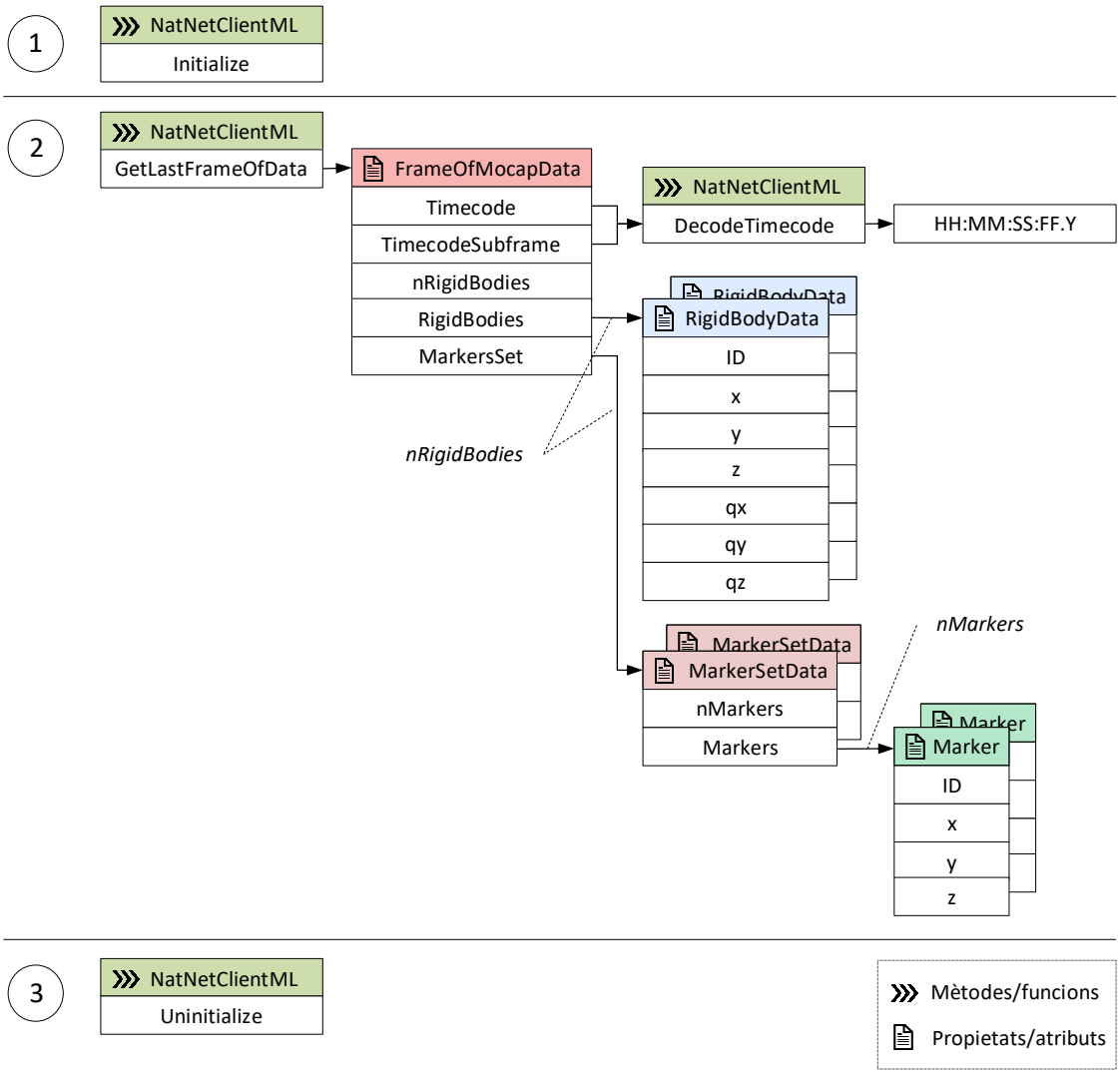

***Fig. 14*. Esquema del procediment d'accés a les dades del Motive**

A continuació, es presenta el codi del Motive en la figura 15. Aquest, està composat principalment per 3 *Case Strutures* (equivalents a les funcions *if else*) niats o *nested*, on l'últim conté un bucle *While* que s'encarrega de llegir i enviar totes les dades proporcionades pel Motive. Totes les funcions utilitzades per accedir als mètodes i atributis es poden trobar en *Functions > Connectivity > .NET.*

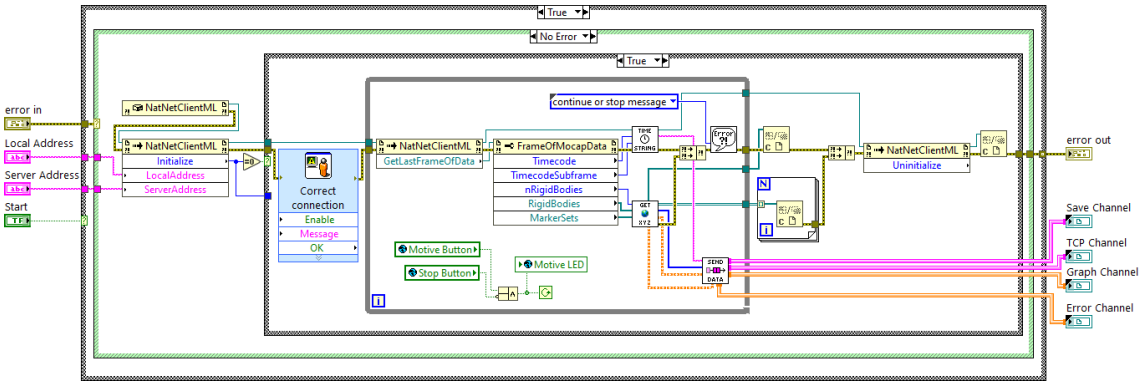

***Fig. 15*. Codi mòdul Motive**

El primer *Case Structure* comprova si s'ha d'activar o no el mòdul, avaluant el valor de la variable booleana *Start*. El segon serveix per veure si s'ha produït algun error anteriorment, si és així simplement passa el codi d'error a la eixida, en cas contrari el mòdul s'executa de forma normal. En aquest moment, és quan s'intenta inicialitzar una connexió amb el Motive amb les dades introduïdes per l'usuari. Per fer-ho, s'ha de carregar la llibreria NatNetML.dll al codi, per poder accedir al mètodes i atributs disponibles. El primer que el col·loca al diagrama és el Constructor Node *NatNetClientML*, posteriorment s'accedeix al mètode *Initialize*.

Si efectivament s'aconsegueix establir la connexió, la funció retorna un 0, en cas contrari el codi de l'error. El tercer i últim *Case Structure* avalua aquest nombre continuant amb l'execució normal si es 0, o passant al *false Case* si no ho és. En el *false Case* l'únic que s'executa és un missatge d'avís a l'usuari indicant-li que no s'ha pogut connectar amb el servidor del Motive, i quin és el codi d'error. Aquesta part del codi es pot observar en la figura 16:

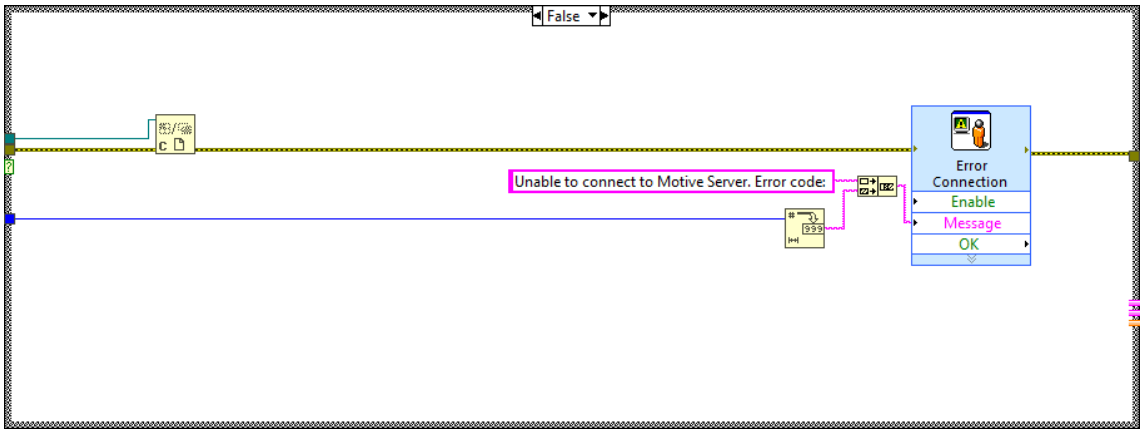

***Fig. 16*. *False Case* del *Case Struture* nº3 del mòdul Motive**

Si d'altra banda la connexió s'estableix correctament, també s'envia una finestra emergent a l'usuari indicant-ho. La següent part del codi és el bucle *While*, on s'accedeix, partint del mateix

objecte que abans, el *NatNetClientML*, a un mètode diferent anomenat *GetLastFrameOfData*. Aquest mètode entrega un nou objecte, *FrameOfMocapData*, on estan totes les dades de posició corresponents a l'últim mostreig que han realitzat les càmeres. Dins de tots els atributs o propietats que posseeix, es troben algunes d'interès que són:

- *Timecode*: és l'hora exacta a la que s'ha pres la captura de moviment.
- *Timecodesubframe*: és el nombre de la mostra que s'ha pres en 1s, varia entre 1 i 120.
- *nRigidBodies*: és el nombre total de sòlids rígids que s'han detectat.
- *RigidBodies*: són els propis objectes dels sòlids rígids, conten totes les dades de posició.
- *MarkersSets*: són els propis objectes dels marcadors, conten totes les dades de posició.

Aquesta informació s'envia als submòduls *Timecode To String* i *Get Position*, que s'encarreguen d'accedir a les dades pertinents i construir les matrius, vectors i *strings* que contenen tota la informació de la mostra. Aquestes dades s'envien al submòdul *Motive Send Data*, que crea els canals d'alta velocitat de transmissió i mana la informació. Al mateix temps, s'avalua si s'ha produït algun error i es comproven les condicions per finalitzar o continuar amb una nova iteració del bucle. Aquest processament de dades es pot observar millor en el codi sols del bucle *While* de la figura 17:

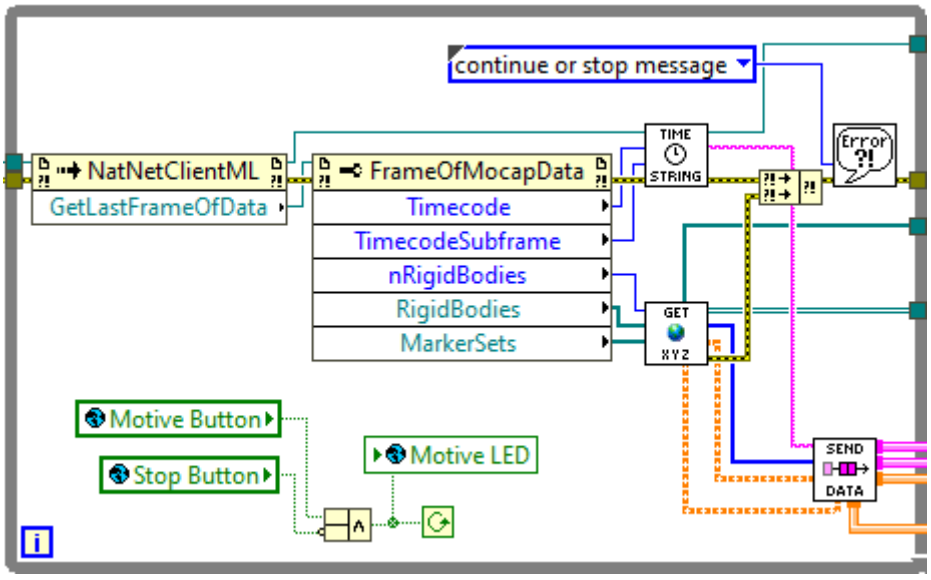

***Fig. 17*. Bucle *While* del mòdul Motive**

Quan es compleixen les condicions que detenen el bucle, és a dir, que s'ha polsat el botó de *Stop* o s'ha desmarcat l'opció del Motive; aquest finalitza deixant que s'execute la part final del codi. Aquesta part, tanca les referències de tots els objectes als que s'ha accedit, i es desconnecta del servidor Motive amb la funció o mètode *Unitialize*.

Per accedir a la informació d'interès sols s'han usat uns pocs mètodes i propietats del total que ofereix l'assemblatge NatNetML.dll. De forma prèvia al desenvolupament de la interfície, s'han estudiat totes les opcions que assemblatge ofereix. Per aquest motiu, s'ha el·laborat una llista amb cada tipus d'objectes i les seues possibilitats, la qual es troba en *Objectes .NET del Motive* del Manual de l'Usuari. A continuació, es presenta i s'explica el codi de tots els submòduls que intervenen en l'execució del Motive.

1.2.1.1. Timecode To String

Es tracta del submòdul que transforma la informació sobre l'hora exacta, en la que s'ha pres la mostra per les càmeres, en una cadena de *strings*, per facilitar el transport d'aquesta dada. Necessita com a paràmetres d'entrada: la referència de l'objecte de classe *NatNetClientML*, el nombre *TimeCode* i *TimeCodeSubFrame*, així com el possible error anterior. La icona i els connectors del *Timecode To String* les podem trobar en la figura 18:

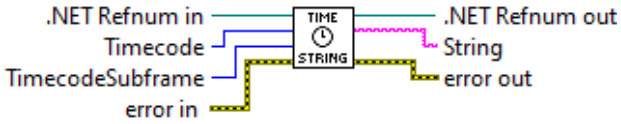

Fig. 18*. Icona i connexions del submòdul *Timecode To String

El codi del que es composa està representat en la figura 19. Com es pot observar, per traduir l'hora exacta s'ha de recórrer a un mètode anomenat *DecodeTimecode*, ja que no es pot accedir directament partint del *Timecode*. Aquest mètode entrega l'hora, minut, segon, mostra i submostra en la que s'ha produït la captura de moviment.

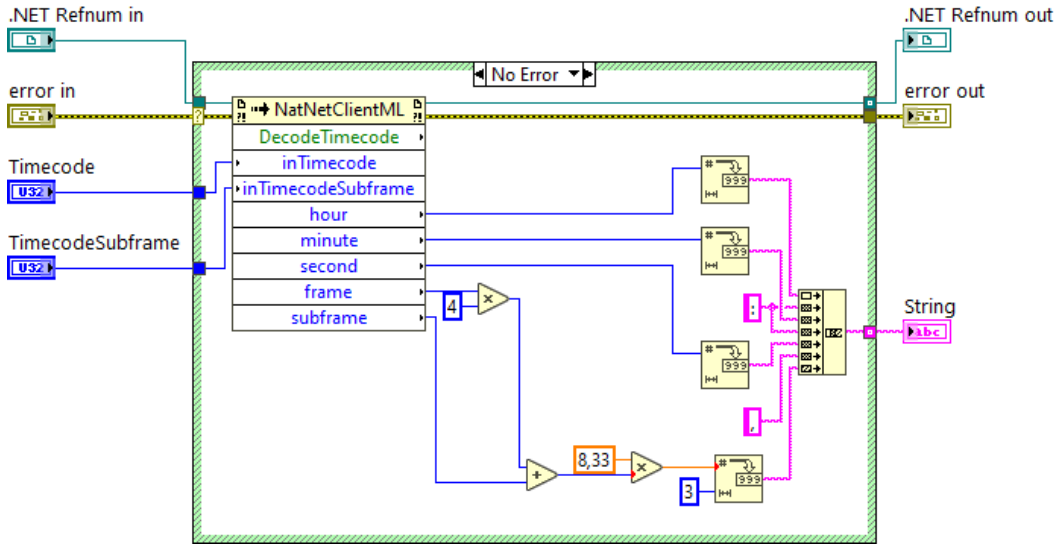

Fig. 19*. Codi del submòdul *Timecode To String

Posteriorment, es transforma el format de *int* a *string* i es realitza a conversió de la mostra i submostra als corresponents mil·lisegons. La raó és que el Motive no entrega directament els mil·lisegons, sinó que entrega el nombre de mostres en cada segon. De les 120fps, conta 30fps cadascuna amb 4 submostres, la forma de fer-ho es pot observar en la figura 20:

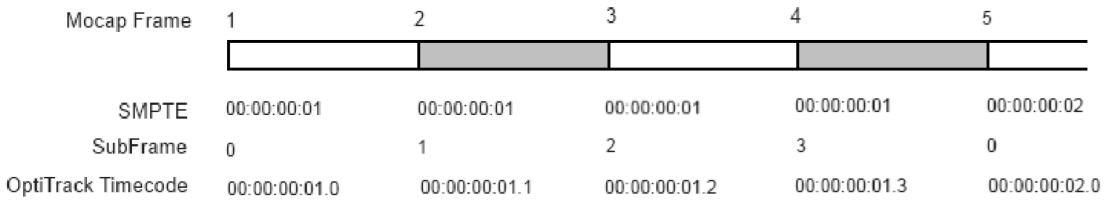

***Fig. 20*. Representació del *Timecode* típic d'OptiTrack**

Nota. Adaptada de *NatNet API User's Guide* (p. 17), per NaturalPoint Inc. , 2016, OptiTrack (https://www.optitrack.com/public/documents/natnet-api-user-guide-2.10.0.pdf).

D'aquesta forma el submòdul té com a paràmetres d'eixida un: *string* amb el moment exacte de la captura, la referència del *NatNetClientML* i el possible error generat.

1.2.1.2. Get Position

Aquest submòdul s'encarrega d'accedir als atributs o propietats de cada sòlid rígid per obtenir la seua posició i orientació, i la dels marcadors que el formen. Treballa usant el submòdul *Get Markers Matrix.* Els paràmetres d'entrada són: les referències del sòlids rígids i dels marcadors que els formen, també el nombre total de sòlids rígids que s'ha captat i l'error. En la següent figura es representa la icona i les connexions que necessita:

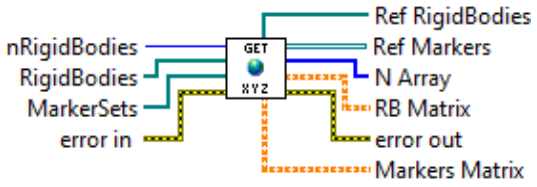

***Fig. 21*. Icona i connexions del submòdul GetPosition**

El codi es pot observar en la figura 22, es composa d'una bucle *for* amb el mateix nombre d'iteracions com sòlids rígids captats. Les funcions que s'utilitzen per a ordenar les dades es troben en *Functions > Programming > Array*, i *Functions > Programming > Array > Matrix*. Dins del bucle *for* s'accedeix als atributs dels objectes *RigidBodyData*, on es troba la posició X, Y, Z, la rotació qx, qy, qz, i l'identificador, entre altres paràmetres. Al mateix temps, la referència sobre el set de marcadors que el formen s'envia al submòdul *Get Markers Matrix*, que es veurà a continuació.

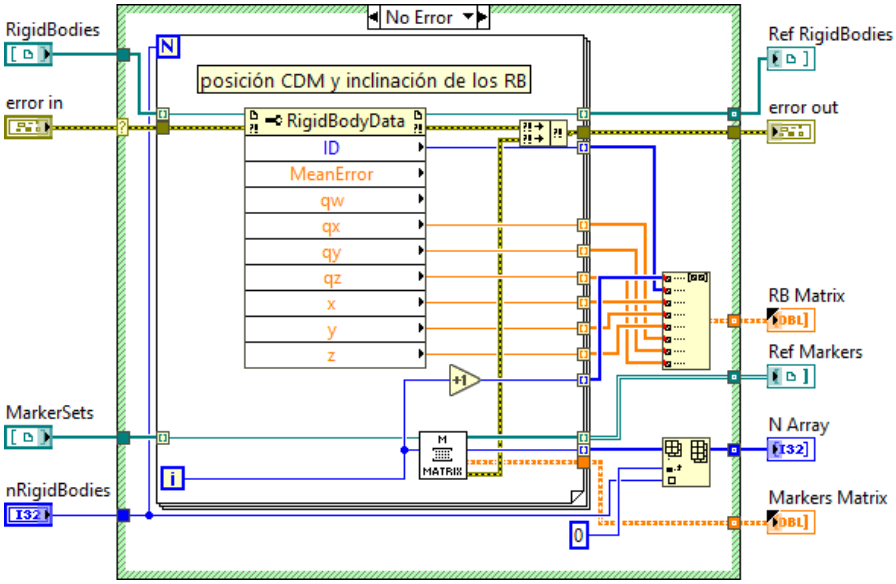

Fig. 22*. Codi del submòdul *Get Position

Per últim, quan finalitza el bucle s'ha creat un vector per cada atribut amb els valors corresponents a cada sòlid. Tota aquesta informació s'organitza en la matriu del sòlids rígids, que s'ha anomenat *RB Matrix,* i el vector amb el nombre de sòlids i marcadors de la mostra, anomenat *N array*. Aquestes variables junt amb la *M Matrix*, les referències i el possible error, són els paràmetres d'eixida del submòdul.

1.2.1.3. Get Markers Matrix

El submòdul *Get Markers Matrix* s'integra dins del *Get Position*, i realitza de forma anàloga el mateix procés que aquest, però amb el marcadors en compte dels sòlids rígids. Necessita per fer-ho: la refència del set de marcadors, el nombre del sòlid rigid i el possible error anterior. Es pot veure en la figura 23, la icona i els connectors que el formen:

Fig. 23*. Icona i connexions del submòdul *Get Markers Matrix

El codi d'aquesta funció està representat en la figura 24, el funcionament és molt similar a l'anterior submòduls. El procediment consisteix en accedir dins de cada objecte *MarkersSetData* a la referència de cada marcador per separat, que s'anomena *Marker*. Dins d'aquest tipus d'objecte, per a cada referència s'accedeix als atributs ID, X, Y, Z. Aquesta informació s'emmagatzema en un vector que posteriorment s'organitza en forma de matriu.

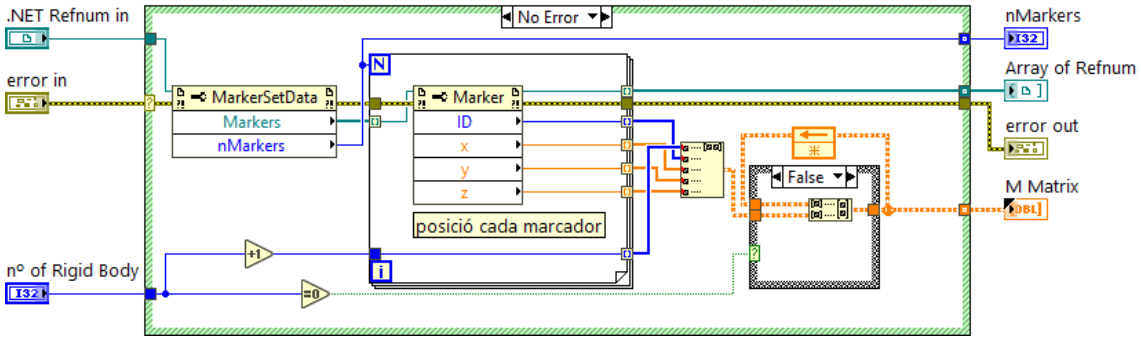

Fig. 24*. Codi del submòdul *Get Markers Matrix

D'aquesta forma, obtenim una matriu per cada sòlid rígid, amb les propietats de cada marcador. Per poder agrupar-les totes en una mateixa matriu, s'utilitza la part final del codi, que s'encarrega d'anar afegint cada nova en la part inferior de l'anterior. Finalment, entrega com a paràmetres d'eixida aquesta matriu, el nombre de marcadors en cada set, el vector de referències i l'error.

1.2.1.4. Motive Send Data

Es tracta de l'últim submòdul del Motive, agafa tota la informació obtinguda i l'organitza en forma de clúster o matriu, segons el destí, i l'envia a traves de canals de transmissió d'alta velocitat. Com a paràmetres d'entrada requerits són: l'hora exacta en format *string*, el vector N, la matriu de sòlids rígids RB i la dels marcadors, i el possible error anterior. Les connexions que necessita i la seua icona es poden observa en la figura 25:

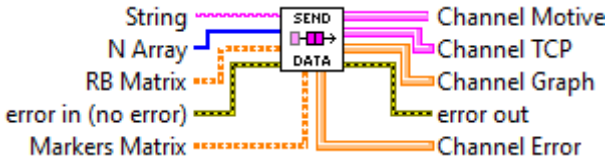

Fig. 25*. Icona i connexions del submòdul *Motive Send Data

El codi d'aquesta part es pot veure en la figura 26, es forma per diferents *Case Structures* niats, que segons les condicions en cada moment, envien la informació al lloc que pertoca. El primer d'ells avalua el valor de la variables booleana *Pause*. Si és *true*, simplement no ocorre res, tot el *true Case* està buit, en cas contrari s'executa tot el codi representat. El seguent *Case Structure*, comprova si existia una error anterior o no, si no s'ha produit anteriorment, continúa de forma normal. Seguidament, s'agrupen totes les dades per una banda en un clúster de dades, i per altra en una gran matriu que inclou dins seua el vector N, i les matrius RB i M.

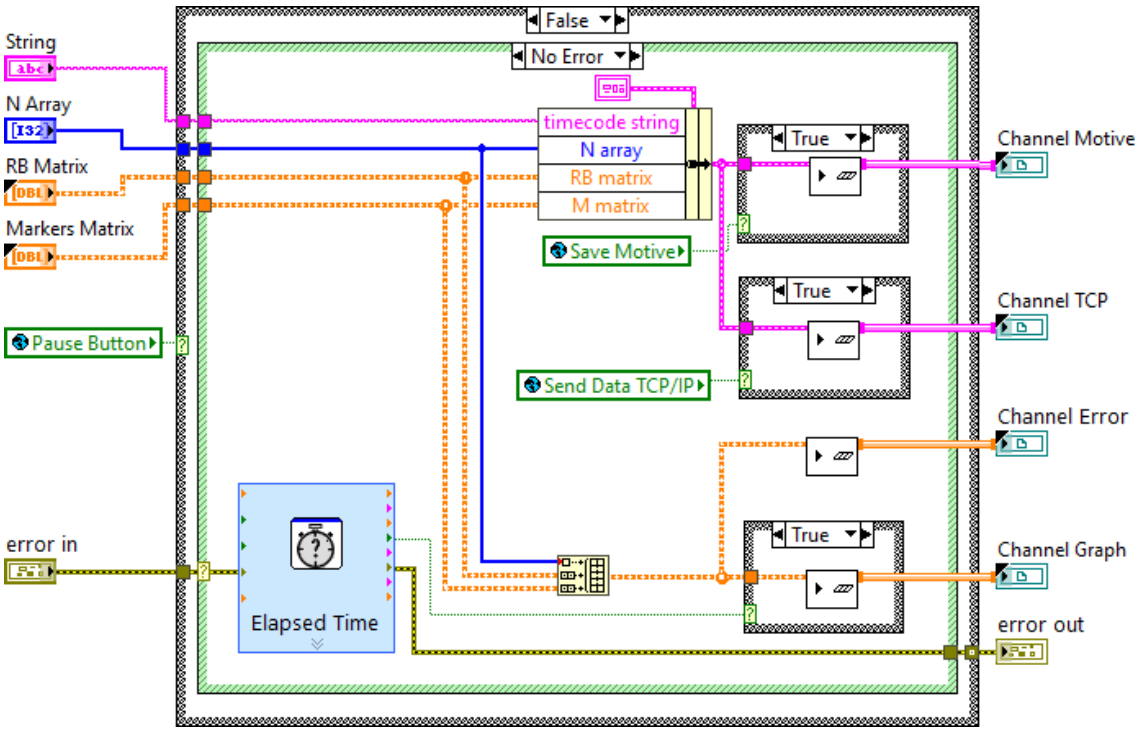

Fig. 26*. Codi del submòdul *Motive Send Data

A continuació, en el nivell final dels *Case Structures*, s'avalua individualment si està o no actiu el mòdul corresponent, *Motive Save* i *Motive TCP Server*, per enviar o no les dades. Per al cas de la informació que viatja al mòdul *Graph* i *Error Calculator*, sempre s'envia perquè sempre està actiu. El que ocorre es que degut a que sols es pot graficar com a màxim a 33ms, s'ha temporitzat l'enviament de dades amb la funció *Elapsed Time*. D'aquesta forma s'ha preestablert que sols s'envie el paquet de dades que estiga disponible als 33ms, descartant la resta encara que s'hagen llegit. Així, s'aconsegueix que la visualització de la interfície continue funcionant en temps real, sense cap retard en la representació dels sòlids. El paràmetres d'eixida són la informació que viatja pels canals d'alta velocitat i l'error.

1.2.2. TCP Client

El mòdul *TCP Client* s'encarrega de rebre totes les dades de posició que li arriben a l'aplicació via TCP/IP, i redirigir-les als mòduls corresponents. El funcionament es basa en l'establiment d'un canal de comunicació amb un servidor amb els protocols TCP/IP. Els paràmetres d'entrada que són: la variable d'activació corresponent, l'adreça IP i el port del servidor que envia les dades i el canal d'error d'entrada. La icona i els diferents connectors del mòdul les trobem representades en la figura 27:

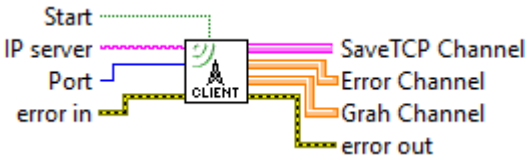

Fig. 27*. Icona i connexions del mòdul *TCP Client

El codi d'aquest submòdul es pot observar en la figura 28, es composa al començament dels 2 *Case Structures* habituals que comproven l'estat d'activació, i posteriorment, els possibles errors. Les funcions utilitzades per a realitzar la connexió es poden troben en *Functions > Data Communication > Protocols > TCP*. El següent mòdul, *TCP Open Connection*, obri el canal de connexió i es manté a l'espera fins detectar un client al que enviar les dades. El temps d'espera pot definir-se en mil·lisegons, admet qualsevol valor fins infinit (representat pel nombre -1), en aquest cas s'ha definit un temps de 60000ms. A continuació, s'executa el bucle *While*, un cop s'ha creat el canal de comunicació.

El procés que es realitza cada iteració és el mateix:

1. Llig un primer missatge de 8 bytes, que conté el nombre de bytes total que tindrà el paquet de dades a rebre, ja que s'ha de indicar a priori. Per fer-ho s'usa el *TCP Read.*
2. Introduint el nombre obtingut anteriorment, es llig el paquet amb dades de posició esperades. Es torna a utilitzar el *TCP Read.*
3. Aquest mateix paquet es torna a reenviar al servidor, per comunicar-li que s'ha rebut correctament el missatge. Ara s'utilitza el *TCP Write.*
4. Es comprova que la informació rebuda és diferent de la rebuda anteriorment, i no s'ha produït un enviament doble de la mateixa informació. Per fer-ho, es compara el valor del *Timecode string*, és a dir, l'hora exacta de eixa captura, amb l'hora de la captura anterior.
5. Si efectivament és diferent, es compleix la condició i la informació es mana al submòdul *TCP Send Data*. En cas contrari no ocorre res, perquè el *true Case* està buit.
6. Per últim, es comprova si s'ha produït algun error durant l'intercanvi de dades, amb el *Simple Error Handler*.

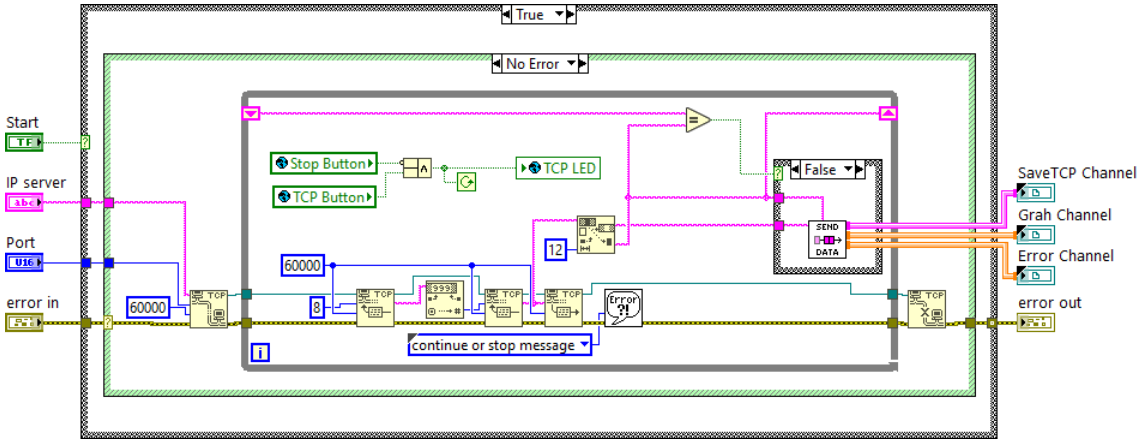

Fig. 28*. Codi del mòdul *TCP Client

Finalment, quan es polsa el botó de *Stop* o es desmarca l'opció de TCP/IP, el bucle *While* finalitza i es destrueix el canal de comunicació creat amb la funció *TCP Close Connection*. Els paràmetres d'eixida del mòdul són les dades que s'envien per el canals de transmissió i el canal de control de l'error.

1.2.2.1. TCP Send Data

Aquest submòdul del *TCP Client* és molt similar al *Motive Send Data*, es diferencien per el format d'entrada de la informació i els canals d'eixida que presenta. En línies generals, transforma i envia les dades de posició rebudes a la resta de mòduls que les necessiten. Els paràmetres d'entrada que posseeix són el *Timecode string*, la resta del missatge rebut via TCP i l'error. Aquests connectors i la icona del submòdul es pot observar en la següent figura:

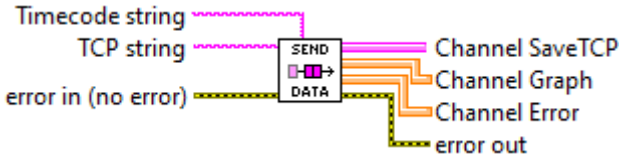

Fig. 29*. Icona i connexions del submòdul *TCP Send Data

Tot seguit, en la figura 30 està representat el codi del submòdul. Està composat per tres nivells diferents de *Case Structures* niats, els dos primers realitzen les mateixes funcions que el submòdul anàleg del Motive. Una vegada s'han comprovat les condicions, la primera part del codi el que realitza és una reorganització de la informació rebuda en el missatge de tipus *string* rebut. S'utilitza el *SpreadSheet String To Array*, perquè és el forma en el qual estan contingudes les dades. Aquesta funció es troba en *Functions* > *Programming* > *String*. A continuació, és secciona la matriu resultant en el vector N i les matrius RB i M, sabent el nombre de fileres que té cadascuna.

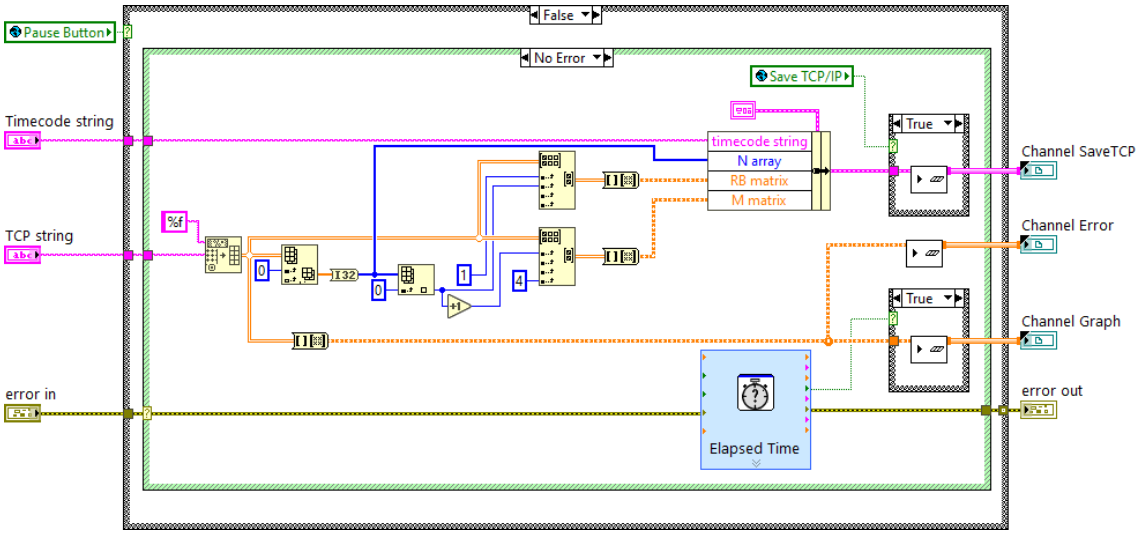

Fig. 30*. Codi del submòdul *TCP Send Data

Una vegada la informació està en el format habitual, aquesta segona part de codi la organitza segons el mòdul de destí. D'aquesta forma, s'agrupen en un clúster les que van a ser enviades al mòdul *TCP Save*, o en una matriu de nombres reals per al *Graph* i *Error Calculator*. L'enviament

de la informació es fa quan s'activa el booleà corresponent del mòdul, o cada 33ms per al cas del *Graph*, gràcies a la funció *Elapsed Time*. Els tres canals d'eixida de les dades, junt amb el possible error, són els paràmetres resultants del submòdul *TCP Send Data*.

1.2.3. TCP Server

Aquest mòdul s'utilitza en l'aplicació principal per enviar les dades de posició del Motive a qualsevol altre programa que les puga necessitar. El funcionament es basa en la creació d'un canal de comunicació mitjançant els protocols TCP/IP, on la interfície actua com a servidor d'informació. Representa el mòdul complementari a l'anterior, el TCP Client, ja que es necessita un servidor i un client per establir la connexió, els dos mòduls depenen de l'altre per fer-ho.

Els paràmetres d'entrada que requereix són: la variable d'activació *Start*, la pròpia informació a enviar (*Channel in*), el nombre del Port on es vol crear el canal TCP/IP i l'error. Aquestes entrades i la pròpia icona es poden observar en la següent figura:

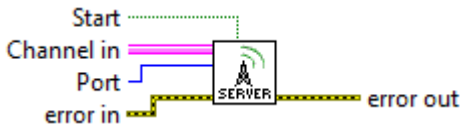

Fig. 31*. Icona i connexions del mòdul *TCP Server

El codi corresponent al *TCP Server* es troba en la figura 32, compta amb els dos *Case Structures* habituals i un bucle *While*, on s'envien els diferents paquets de dades. Al igual que les funcions del submòdul anterior, totes les funcions relacionades amb la creació del canal TCP es troben en *Functions > Data Communication > Protocols > TCP*. Primerament, es comprova que siga la variable *Start* està activa, i que no s'ha produït ningun error en l'execució del codi anterior. Tot seguit, es troba la funció *TCP Listen*, que s'encarrega d'esperar fins que es connecta alguna aplicació al Port indicat, i així establir el canal de comunicació TCP. Aquesta funció també permet establir el temps d'espera màxim en mil·lisegons, en el cas del projecte s'ha deixat el valor per defecte, infinit (que equival al -1).

Quan s'aconsegueix la connexió, s'entra dins del bucle *While*, on es converteix i s'envia la informació rebuda, la seqüencia que es realitza en cada iteració es descriu a continuació:

1. Es llig la informació que arriba pel *High Speed Stream channel* amb la funció *Read High Speed Channel*. Per accedir a ella s'ha de clicar el botó dret *Create > Channel Reader*.
2. S'accedeix a les dades contingudes en el clúster amb *Unbundle By Name*. Aquesta funció es troba en *Funtions > Programming > Cluster, Class & Variants*.
3. S'agrupen les dades i es transformen a *string*, ja que es el format en el que s'ha d'enviar les dades. Per fer-ho s'usa *Array To SpreadSheet String* i *Concatenate Strings*, que es troben en *Funtions > Programming > String*.
4. Es mesura i s'envia el nombre de bytes totals que tindrà el missatge amb les dades de posició.
5. S'envia el propi paquet d'informació de posició, amb la grandària indicada en el pas anterior.
6. Es verifica que el missatge s'ha rebut correctament. Per fer-ho, es compara el missatge rebut amb el que s'acaba d'enviar, per tal de que siguen idèntics.

7. Finalment, es comprova si s'ha produït algun error en la execució normal, si s'ha polsat el botó de *Stop* o si s'ha desmarcat l'opció que activa el mòdul.

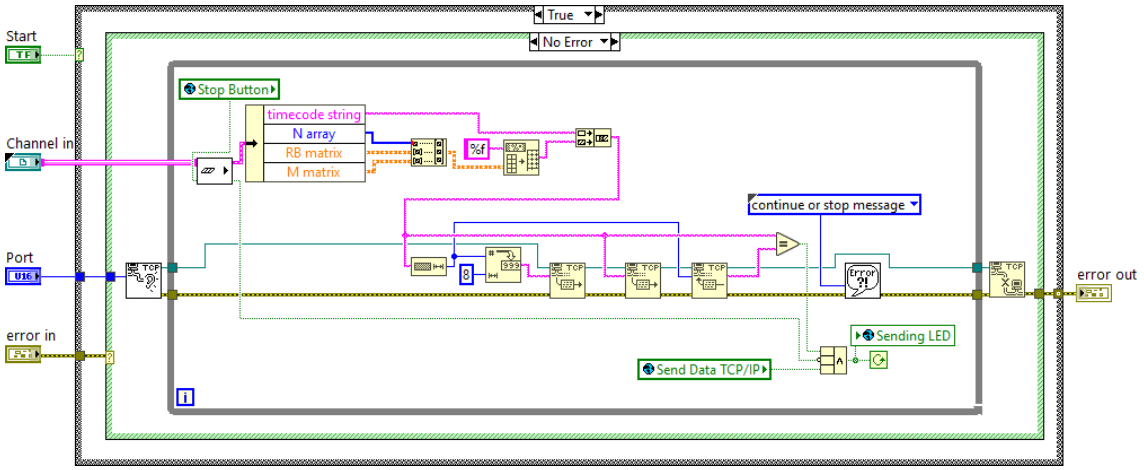

Fig. 32*. Codi del mòdul *TCP Server

Per últim destacar, que la variable *Stop* s'ha connectat al *Abort Before Read* del *Channel Reader* perquè en cas contrari es queda pendent de llegir i no permet finalitzar el bucle. Quan finalment ho fa, es tanca el canal TCP/IP creat, i per últim entrega com a paràmetres d'eixida el possible error que s'haja pogut produir.

1.2.4. Graph

El mòdul *Graph* s'encarrega de graficar tots els sòlids rígids que li envien els diferents mòduls productors. Es tracta de l'element més crític en temps d'execució, per aquest motiu s'ha optimitzat al màxim el seu codi. El funcionament es basa en llegir les dades de posició que li arriben, i graficar aquesta informació amb l'ajuda de la tecnologia ActiveX. Per aquest motiu, els paràmetres d'entrada són la referència del container ActiveX Graph3D, els diferents canal de transmissió de dades i el canal de control de l'error. A continuació, en la figura 33 es troba representada la icona i les diferents connexions que posseeix el mòdul presentat.

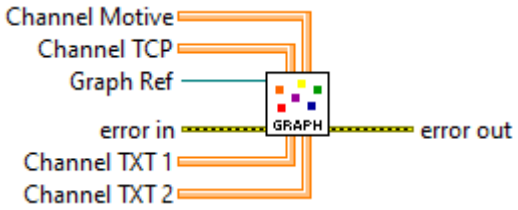

Fig. 33*. Icona i connexions del mòdul *Graph

El codi es basa en les funcions ActiveX del tipus CWGraph3D, les quals podem trobar en *Functions > Connectivity > ActiveX*. El funcionament per accedir als mètodes i atributs es molt similar al descrit en el mòdul Motive, amb la tecnologia .NET, degut a que ambdós es basen la programació orientada a objectes. Per poder definir el tipus d'objecte, *CWGraph3D*, s'ha de col·locar un container ActiveX en el *Front Panel* i clicar botó dret i elegir *Insert ActiveX Object... > CWGraph3D Control*. En eixe moment es crea la referència de la classe d'objecte desitjat, per

accedir a les funcions i propietats que presenta, s'usen els *Invoke Node* i *Property Node* de la paleta ActiveX. Els objectes, atributs i funcions que s'utilitzen es troben representades en els diferents codis que s'expliquen a continuació. No obstant, en els *Objectes ActiveX del CWGraph3D* del Manual, es troba una recopilació de tots els tipus que existeixen per al cas de l'ActiveX *CWGraph3D*.

El codi d'aquest mòdul es pot observar en la figura 34, està englobat en un *Case Structure*, que comprova l'estat de l'error anterior. Principalment es composa de: funcions que inicialitzen i configuren el gràfic, dos bucles *While* que s'encarreguen la correcta visualització, i quatre bucles *For* que, quan finalitzen els bucles *While*, tanquen totes les referències als objectes que s'ha obert.

Primerament, s'explica la part del codi inferior. Aquesta, s'encarrega de preparar el visualitzador de la interfície, esborra el sòlids rígis que pogueren quedar de la sessió anterior i configura els color de les quadrícules i el fons. Al mateix temps, s'accedeix a l'objecte *CWPlots3D* i es divideix aquesta rama en 2, una part correspon al control del eixos, i l'altra a la dels sòlids.

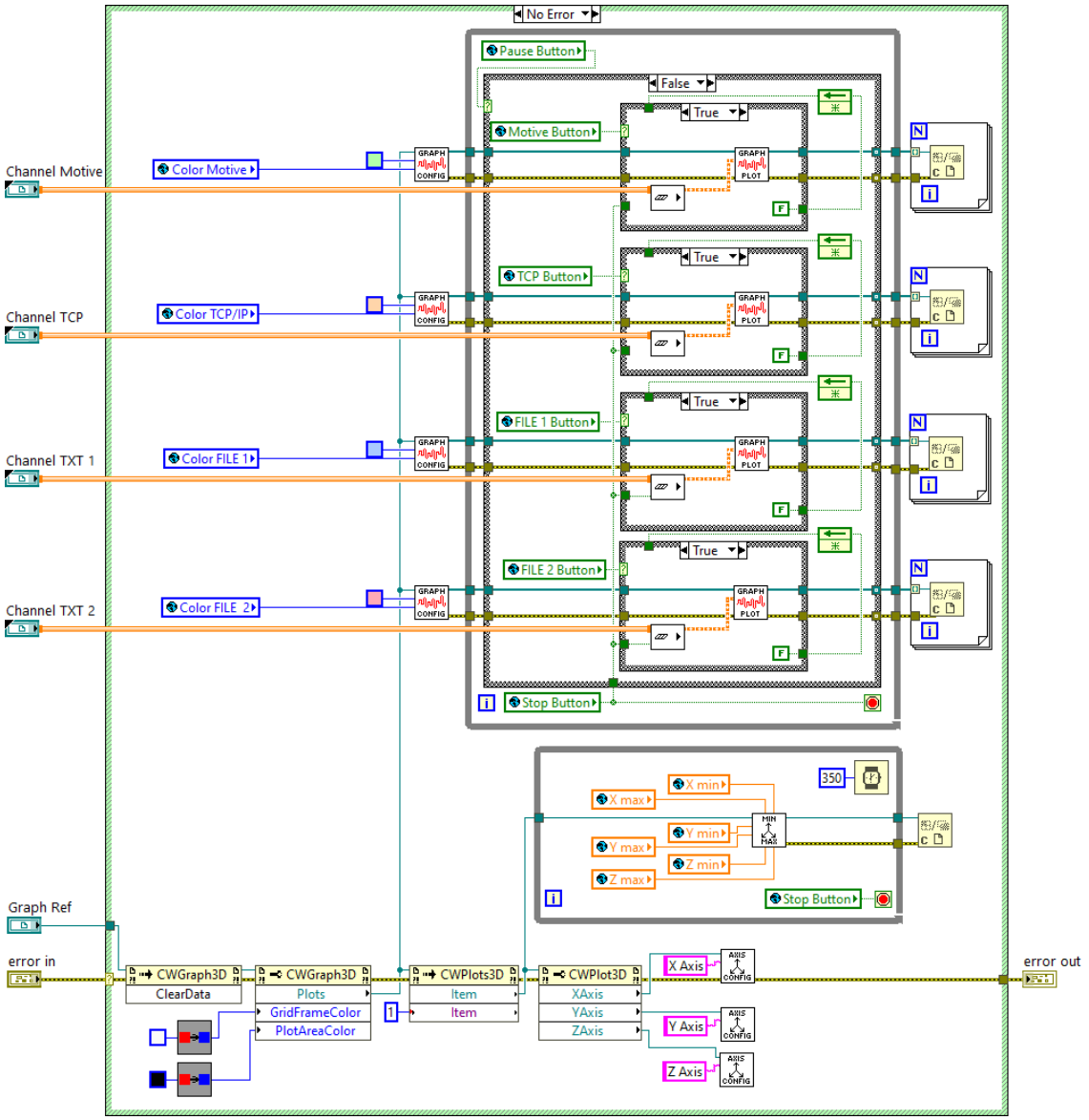

Fig. 34*. Codi del mòdul *Graph

La part dels eixos es correspon amb la referència que continua per la part inferior. En aquesta, s'accedeix mitjançant el mètode *Item* al nou objecte *CWPlot3D*, on es troben les propietats dels eixos. S'arriba a una nova subdivisió en la referència, per una banda va als submòduls *Axis Configuration*, que defineixen les característiques que tindran els eixos. D'altra banda, laa referència entra en el bucle *While* inferior, que s'encarregarà d'actualitzar cada 350ms els valor màxims i mínims que introdueixa l'usuari usant el submòdul *Axis Min Max*.

L'altra rama de la referència *CWPlots3D* correspon als sòlids rígids. Aquesta referència es replica 4 vegades, una per cada font productora d'informació té el programa: Motive, *TCP Client* i *Read TXT 1* i *Read TXT 2*. El primer que es troba és el submòdul *Graph All Configuration*, el qual inicialitza les característiques bàsiques de cada tipus de gràfic necessari. Aquest té connectat un color constant, el color del CG, i l'altre color que li arriba es correspon amb el que elegeix l'usuari en la interfície. Tot segui s'entra dins del bucle *While* encarregat de la visualització dels sòlids, en la figura 35 es mostra més en detall:

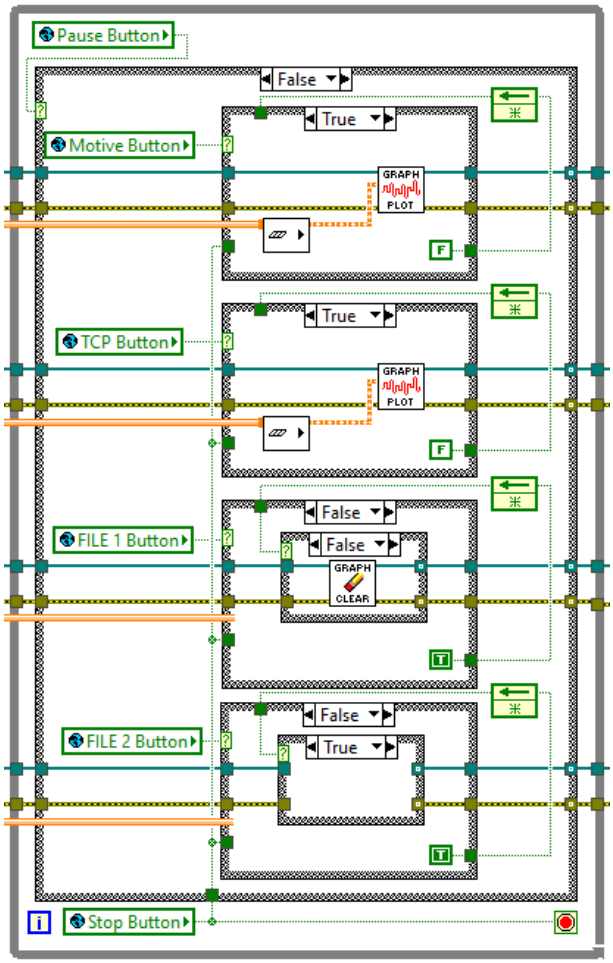

Fig. 35*. Bucle *While* dels sòlids rígids del mòdul *Graph

Com es pot observar el codi que s'executa en cada iteració, està format per 3 nivells de *Case Structures*, cadascun encarregat de comprovar un aspecte diferent. L'execució que es realitza cada iteració s'explica a continuació:

- El *Case* exterior comprova l'estat del *Pause Button*, per realitzar el codi mostrat o directament, en cas de que estiga actiu, no ocorrega res perquè el *true Case* està buit.
- Seguidament, es llegeix el valor de les variables d'activació de cada font productor dades, podent executar-se dos codis distints segons si està activada o desactivada:
    A. Si està actiu (casos de la figura del Motive i *TCP Client*): Les dades es llegeixen del canal, i s'envien al submòdul *Graph Plot All*, encarregat de graficar-les.
    B. Si està desactivat no llig les dades i s'arriba de nou amb un altre *Case Structure* que avalua el valor d'una variable que indica si s'ha esborrat o no el sòlid graficat:
        1. El valor que llegeix es *false* (cas de la figura TXT FILE 1): s'executa el submòdul *Graph Clear All* que neteja el sòlid graficat just en la iteració anterior.
        2. El valor que llegeix es *true* (cas de la figura TXT FILE 2): no ocorre res, ja que el *true Case* està buit, perquè el visualitzador no té graficat cap sòlid rígid d'aquesta font.

El motiu d'aquesta estructura de *Cases* niats és evitar execucions innecessàries dels submòduls, ja que això consumeix capacitat del processador, i el mòdul *Graph* és el més crític en temps d'execució. A continuació, es presenten tots el submòduls que intervenen en el mòdul presentat, fent possible totes les funcionalitats que aquest posseeix.

1.2.4.1. Axis Configuration

El submòdul *Axis Configuration* defineix el format que tindrà l'eix de la referència que se li connecte, configurant: el color, el nom, la grandària de les lletres, el nombre de decimals, entre altres funcions. Per fer-ho, necessita com a paràmetres d'entrada: la referència de l'eix a configurar, el nom que se li va a posar i el possible error anterior, el qual és l'únic paràmetres d'eixida. Aquests canals de comunicació del mòdul es poden veure a la figura següent:

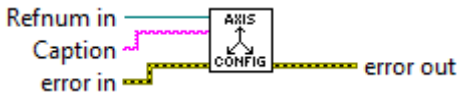

Fig. 36*. Icona i connexions del submòdul *Axis Configuration

El codi d'aquest submòdul es presenta en la figura 37, com es pot observar es basa en la funcions *Functions > Connectivity > ActiveX*. Es necessiten les funcions de *Property Node* perquè el codi s'encarrega d'accedir i escriure en el atributs o propietats de cada objecte, els valors necessàris per a la correcta visualització en el gràfic. Posteriorment, es tanca la referència dels objectes als que s'ha accedit, i es comunica el possible error generat.

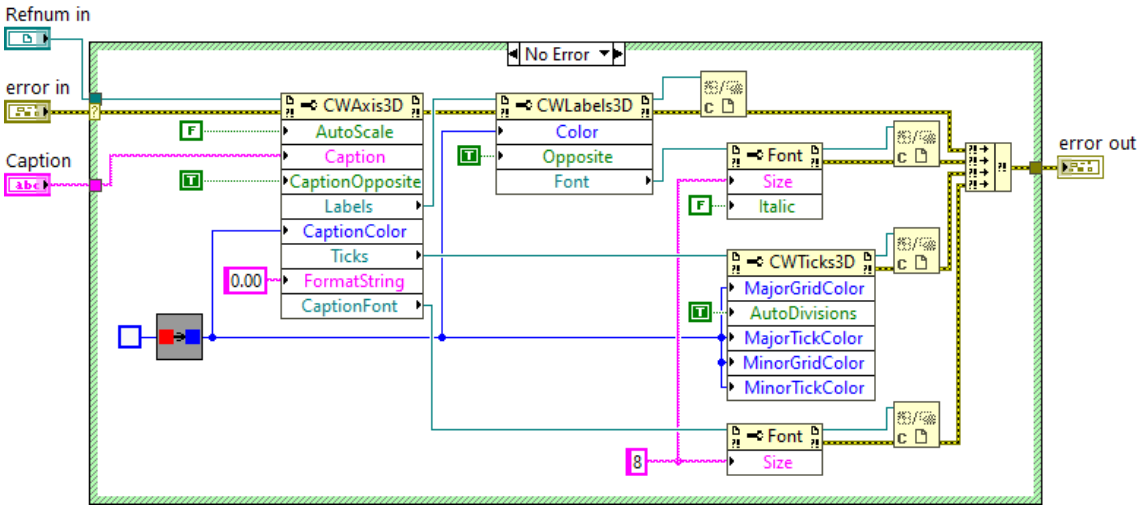

Fig. 37*. Codi del submòdul *Axis Configuration

1.2.4.2. Axis Min Max

El *Axis Min Max* és l'altre submòdul que gestiona els eixos del gràfic, aquest s'encarrega d'establir els valor màxim i mínim dels 3 eixos. Per poder dur-ho a terme necessita com a entrades: la referència l'objecte *CWPlot3D*, i els valors a establir en cada eix, a banda de l'habitual canal de l'error. Totes les connexions les podem observar en la figura 38.

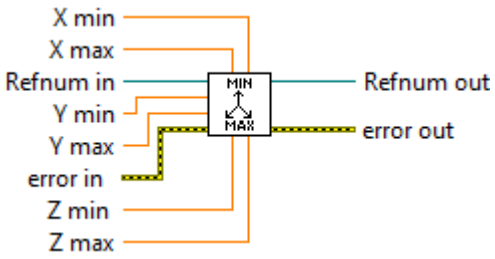

Fig. 38*. Icona i connexions del submòdul *Axis Min Max

El codi d'aquest submòdul està representat en la figura 39. Es basa simplement en usar el mètode *SetMaxMin*, usant la funció *Invoke Node* de la paleta ActiveX. S'han d'introduir els valors definits per l'usuari com entrada al mètode. Finalment, es tornen a tancar les referències obertes, i s'envia el possible error i la referència de l'objecte *CWPlot3D*, com a eixides del submòdul.

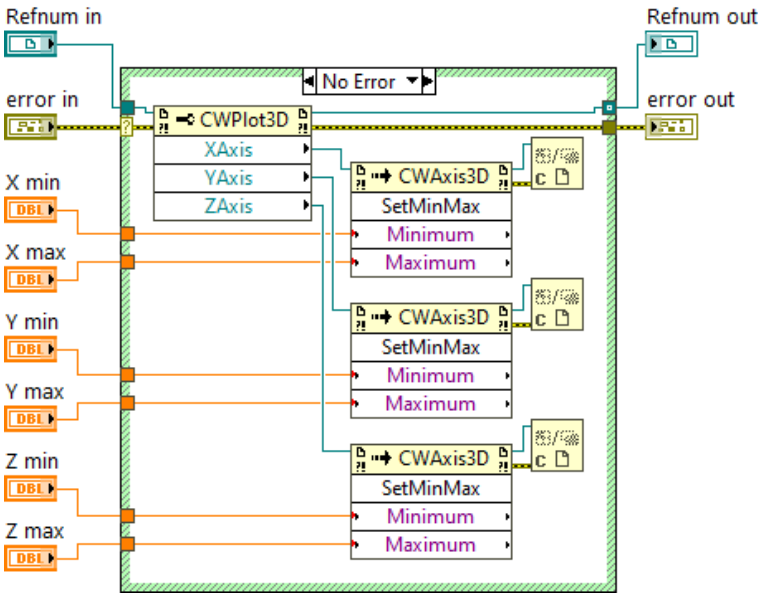

Fig. 39*. Codi del submòdul *Axis Min Max

1.2.4.3. Graph Configuration All

Aquest submòdul pertanyent a *Graph* s'utilitza cada vegada que s'inicia el mòdul, i es necessita un per cada font productora de dades. La seua funció és, mitjançant els submòduls *Graph Point Configuration* i *Graph Surface Configuration*, definir les característiques dels gràfics per tal de que s'aconseguisca un format que permeta la correcta visualització de tota la informació.

Com a paràmetres d'entrada trobem: la referència al tipus d'objecte *CWPlot3D*, el color que es vol que tinga el sòlid, així com el CG d'aquest, i el canal de control de l'error. Aquests canals d'entrada es mostren en la figura 40:

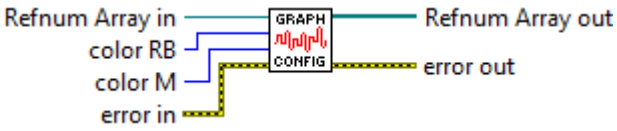

Fig. 40*. Icona i connexions del submòdul *Graph Configuration All

El codi de *Graph Configuration All*, es presenta en la figura 41. Aquest, s'encarrega de transmetre als corresponents submòduls, la referència de l'objecte *CWPlot3D* i els colors. S'observa que el primer de *Graph Point Configuration*, es correspon amb els CG dels sòlids, per això li arriba el color del RB i la grandària del punt és 4 (i no 3 com la dels marcadors). El següent *Graph Point Configuration*, sí que són els marcadors que formaran els sòlids. Finalment, trobem els submòduls *Graph Surface Configuration*, existeix un per cada sòlid rígid. En cas de que es necessitara augmentar el nombre de sòlids a graficar, sols s'ha de replicar aquest submòdul. Posteriorment, es forma un vector de referències, i s'uneix el canal d'error corresponent a cada submòdul.

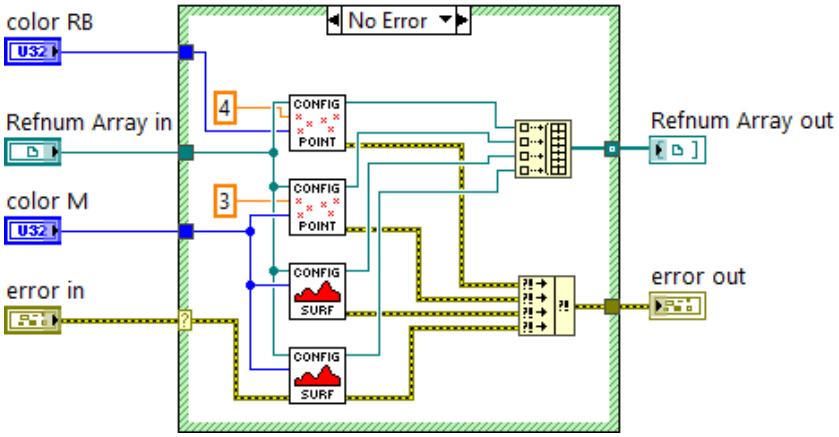

Fig. 41*. Codi del submòdul *Graph Configuration All

1.2.4.4. Graph Point Configuration

El submòdul *Graph Point Configuration* és l'encarregat de definir les característiques específiques que tindran els marcadors de la referència. Per aquest motiu, rep com a entrades la referència del *CWPlot3D*, la grandària de punt desitjada, el color i el possible error anterior. Els paràmetres d'entrada presentats es poden observar en la següent figura:

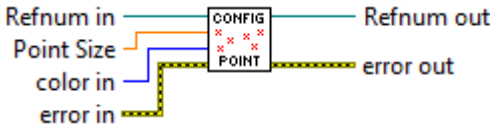

Fig. 42*. Icona i connexions del submòdul *Point Configuration

A continuació, en la figura 43, es presenta el codi d'aquest submòdul. El primer que s'executa és el mètode encarregat d'afegir una nova figura al visualitzador existent. A continuació, s'escriuen els valors de les propietats que es volen canviar per ajustar el gràfic a les necessitats presentades.

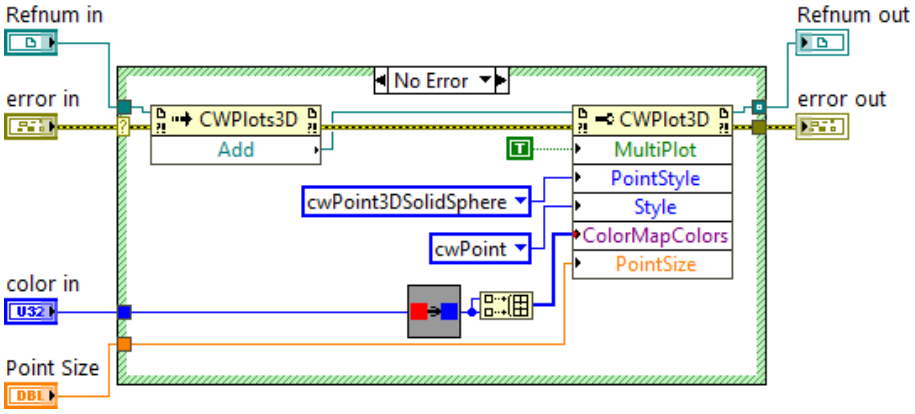

Fig. 43*. Codi del submòdul *Point Configuration

1.2.4.5. Graph Surface Configuration

Aquest submòdul és molt similar al presentat anteriorment, amb la mateixa funció: definir els paràmetres bàsics del gràfic. La diferència entre ambdós és l'estil del gràfic, i les configuracions derivades d'aquest fet. Com a paràmetres d'entrada rep: la referència de *CWGraph3D*, el color definit i l'error; aquests canals d'entrada es poden observar a la figura 44:

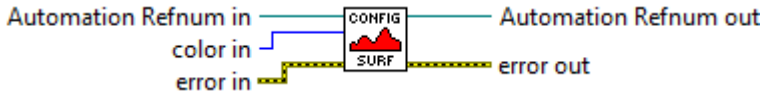

Fig. 44*. Icona i connexions del submòdul *Graph Surface Configuration

Tot seguit, en la figura 45, es mostra el codi corresponent al submòdul presentat. El funcionament és anàleg al del submòdul anterior. Es crida al mètode *Add*, i s'escriuen en les propietats del nou gràfic, els valors establerts per obtenir una bona visualització. Per últim, com a paràmetre d'eixida trobem la referència del nou objecte, *CWPlot3D*, i el canal de l'error.

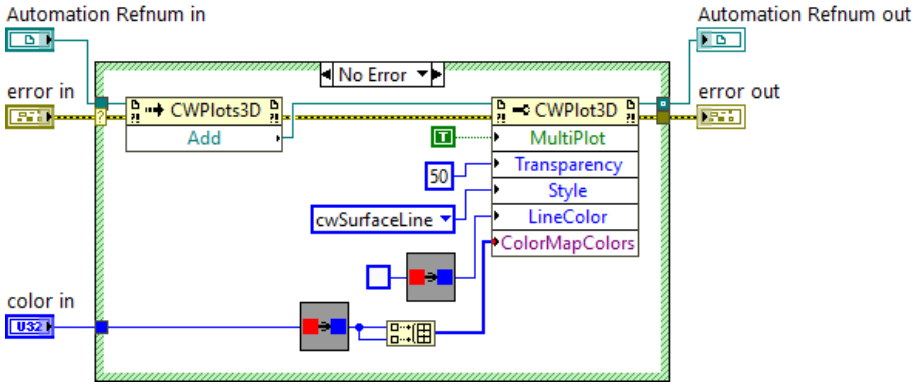

Fig. 45*. Codi del submòdul *Graph Surface Configuration

1.2.4.6. Graph Clear All

El submòdul *Graph Clear All* esborra del visualitzador els sòlids rígids corresponents a la referència que se li connecte. En el mòdul *Graph* és necessari fer-ho quan la font productor del sòlid no està activa, perquè si no s'executara aquest submòdul, es quedaria en el visualitzador la última posició graficada del cos. Els paràmetres d'entrada i eixida són els mateixos, i són: el vector de referències del gràfics i el canal de l'error; es pot observar en la figura 46.

Fig. 46*. Icona i connexions del submòdul *Graph Clear All

El codi corresponent a aquest submòdul es presenta en la figura 47. El funcionament es basa en accedir a cada referència del vector, i executar el mètode *ClearData*, per a cadascunna. Per última s'agrupen de nou les referències i es fusiona el canal d'error.

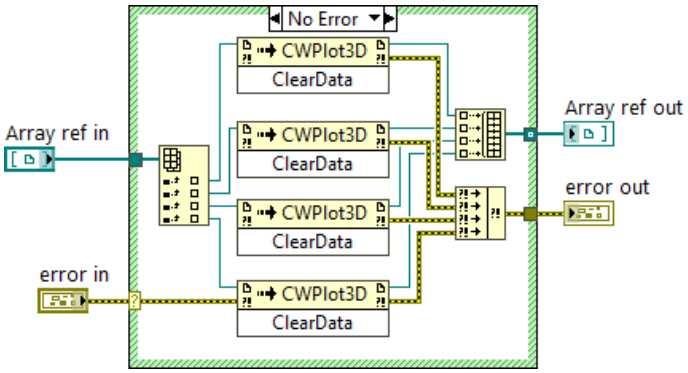

Fig. 47*. Codi del submòdul *Graph Clear All

1.2.4.7. Graph Plot All

Aquest submòdul s'utilitza per graficar, amb la forma ja definida, el sòlids rígids que proporciona cada font productora. Per aquest motiu, existeixen 4 còpies en el mòdul *Graph*, una per cada mòdul productor d'informació. Els paràmetres d'entrada que necessita són el vector de referències, la matriu amb totes les dades de posició i el canal de control de l'error, com es pot veure en la següent figura:

Fig. 48*. Icona i connexions del submòdul *Graph Plot All

Tot seguit, en la figura 49, es troba el codi d'aquest submòdul. El funcionament que es segueix és reorganitzar totes les dades que li arriben i manar-les als submòduls corresponents. La funcionalitat de cadascun es troba detallada a continuació:

- El submòdul *Graph Plot Point*, situat en la part superior, s'encarrega de graficar el CG dels sòlids. La matriu de dades que li entra es la *RB Matrix*.
- El segon *Graph Plot Point*, grafica tots els marcadors dels sòlids rígids que hi existeixen en eixa mostra d'informació. Li entra solament com a dades la *M Matrix*.
- El primer *Graph Plot Surface*, és el responsable de la forma del sòlid rígid nº1, és a dir, genera el cos o les superfícies que tinga a partir dels punts o marcadors que el defineixen.
- L'últim *Graph Plot Surface*, té la mateixa funció que l'anterior, però per al sòlid rígid nº2. Situat dins d'un *Case Structure* per evitar la seua execució en cas de que sols existisca un sòlid, i així estalviar-se l'execució del mòdul.

Destacar que, igual que en el *Graph All Configuration*, si s'haguera d'augmentar el nombre de sòlids a visualitzar sols s'hauria de replicar l'última part del codi, és a dir, el Case Structure amb *Graph Plot Surface*.

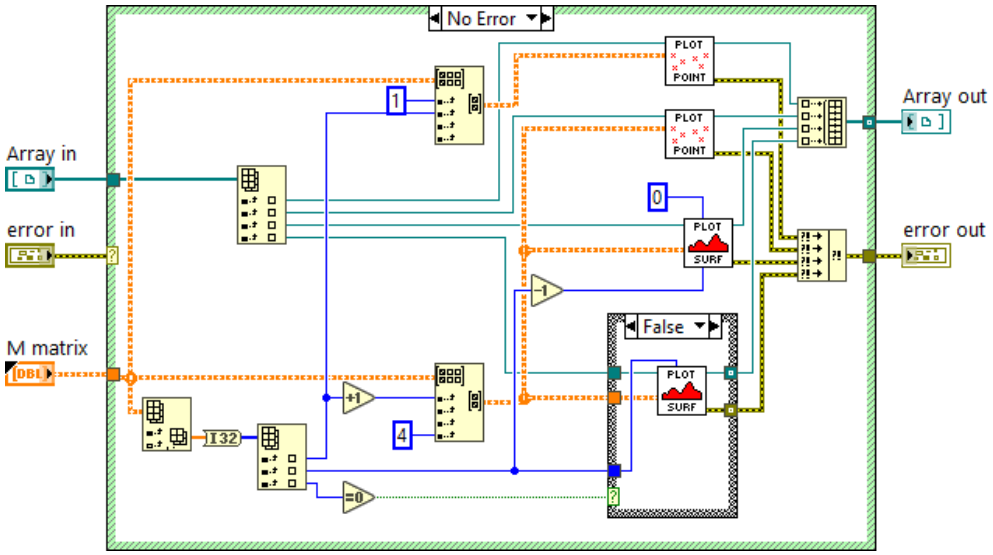

Fig. 49*. Codi del submòdul *Graph Plot All

1.2.4.8. Graph Point Plot

Es tracta del submòdul encarregat de graficar tots els marcadors que li arriben de la matriu donada. Per aquest motiu necessita la matriu amb les posicions que li corresponga, segons el comentat anteriorment, o la *RB Matrix* o la *M Matrix*. Aquest es un dels paràmetres d'entrada junt amb la referència de l'objecte *CWPlot3D* i el canal habitual de l'error, com està representat en a figura 50:

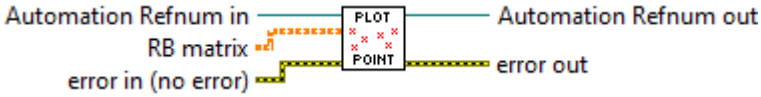

Fig. 50*. Icona i connexions del submòdul *Graph Point Plot

Seguidament, podem observar el codi del submòdul en la figura 51. Primer, s'accedeix a les dades corresponent a cada eix, i aquestes dades en forma de vector s'envien a l'entrada corresponent de la funció. La funció o mètode que s'executa es el *Plot3DMesh*, i com demanda les dades en forat *variant* s'utilitza el transformador de *float-variant* corresponent per fer més eficient el codi. Tampoc es comprova l'error anterior a l'entrada del submòdul, pel mateix motiu fer més ràpid i fluida l'execució del codi.

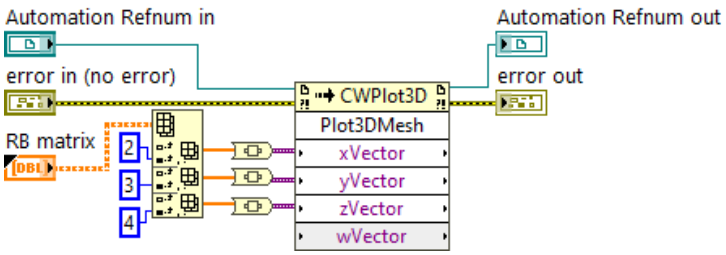

Fig. 51*. Codi del submòdul *Graph Point Plot

1.2.4.9. Graph Surface Plot

L'últim submòdul s'encarrega de graficar el cos o la superfície, en definitiva, la forma del sòlid rígid que defineixa la informació entrant dels marcadors. S'ha d'usar un per cada sòlid rígid, i per seleccionar quins marcadors són els que li corresponen, dins de tota la *M Matrix*, es requereix com a paràmetres d'entrada les fileres de la matriu on es troba la informació d'interès. Per aquest motiu, els paràmetres d'entrada són: els dos nombres comentats, la *M Matrix*, a referència *CWPlot3D* i el canal d'error, com està representat en la figura següent:

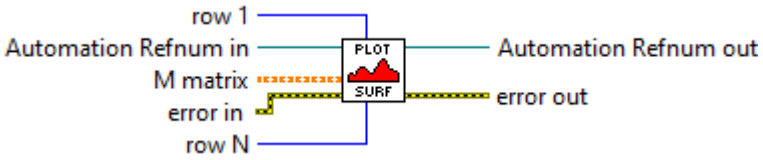

Fig. 52*. Icona i connexions del submòdul *Graph Surface Plot

El codi d'aquest submòdul es mostra en la figura 53. El funcionament és idèntic al presentat en el submòdul anterior, ja que el tipus de gràfic que anava a ser cadascun es determina en els submòdul *Graph Configuration*. L'única diferencia resideix en que ara es selecciona una submatriu a gaficar en funció dels valors d'entrada, que marquen de quina filera a quina es troba la informació necessària.

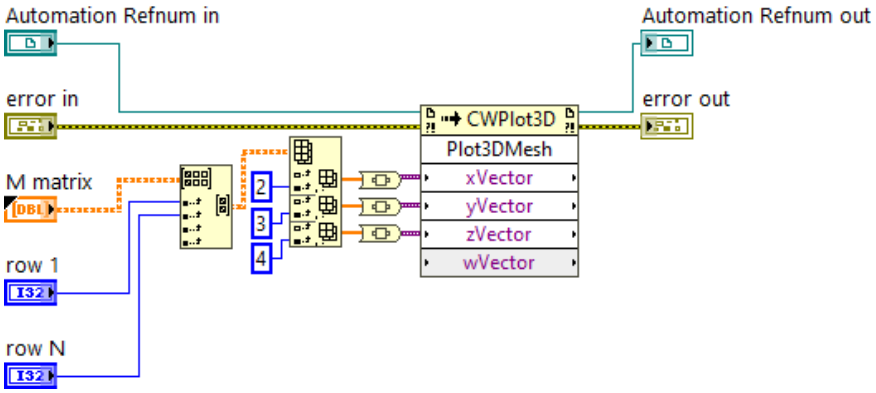

Fig. 53*. Codi del submòdul *Graph Surface Plot

1.2.5. Error Calculator

La funció principal del mòdul *Error Calculator* és obtenir la diferència entre a posició i orientació del sòlid rígid que l'usuari trie. Per poder dur-ho a terme, està preparat per rebre com a paràmetres d'entrada les dades de posició provinents de tots els mòduls productors. A estes quatre entrades, també s'afegeix l'habitual canal d'error. Tots els canals d'entrada es presenten en la següent figura:

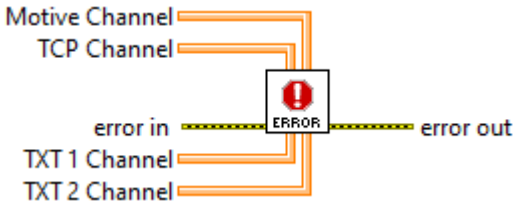

Fig. 54*. Icona i connexions del mòdul *Error Calculator

El codi corresponent a el mòdul *Error Calculator* es mostra en la figura 55. Després de comprovar la inexistència d'error anteriors, s'entra en el bucle *While*, on es troba el codi principal del mòdul. Aquest bucle s'executa contínuament fins que es polsa el botó *Stop*, al igual que *Graph*, sempre s'està calculant l'error independentment de les fonts. Seguidament, s'expliquen els diferents passos seguits en cada iteració del bucle:

- Primerament, es comprova l'estat de les variables d'activació de cada mòdul productor. Si està activa es llegeixen les dades del *High Speed Streaming Channel*.
- Es llegeix REF i VALOR REAL, variables que indiquen entre quins sòlids rígid es vol calcular l'error, i es deixa passar el valor dels seleccionats.
- S'extrau de tota la matriu de dades, solament la *RB matrix*. Per poder fer-ho primer s'extrau el *N array*.
- Es calcula la diferència entre les dues matrius, generant una matriu de l'error real.
- Partint d'aquesta, es calcula la matriu de l'error relatiu.
- Es compara el màxim valor d'aquesta amb el màxim error relatiu definit per l'usuari. Si es major s'encén el LED corresponent i s'activa l'alarma sonora. Aquesta la podem trobar en *Functions > Programming > Sounds & Graphics > Beep*.
- Paral·lelament, els vectors corresponents al sòlid rígid elegit s'agrupen en una sola matriu, i s'escriuen en la variable *Global Error*.

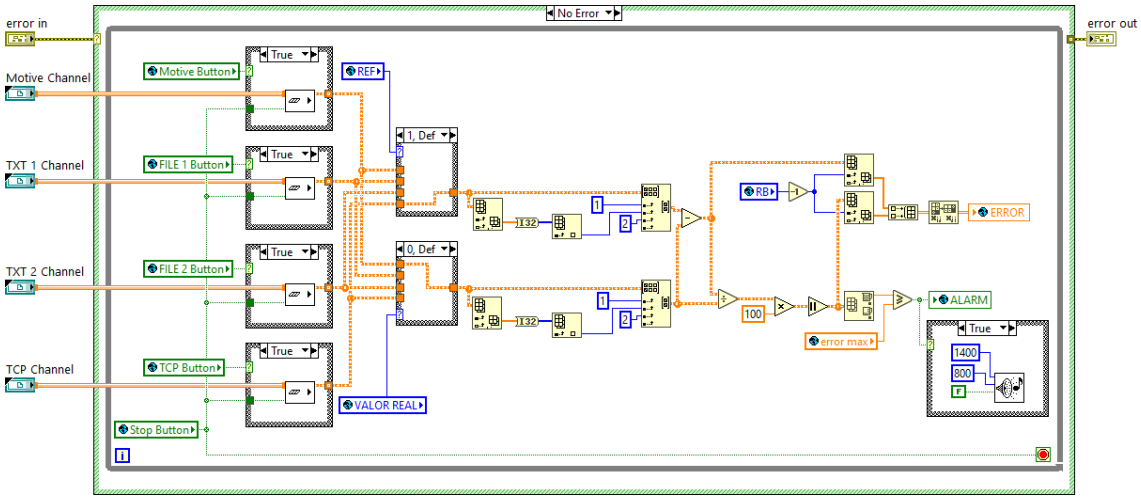

Fig. 55*. Codi del mòdul *Error Calculator

Finalment, destacar que l'únic paràmetre d'eixida és el canal d'error. Aquest sols pot estar generat per funcions o mòduls anteriors a ell, degut a ninguna de les funcions del codi presenten l'habitual canal del control de l'error, és a dir, que no poden generar un avís d'error.

1.2.6. Read TXT

El mòdul *Read TXT* s'encarrega d'obrir i llegir totes les dades de posició contingudes en un arxiu de tipus TXT, s'usa dues vegades diferents en l'aplicació, ja que aquesta té capacitat de treballar amb dos fitxers TXT diferents. El funcionament general es basa en obrir el fitxer(indicat per l'usuari en el *TXT path*), llegir i agrupar tota la informació que conté. Finalment envia, en el format adequat, les dades a la resta de mòduls.

Per aconseguir les funcions descrites, el mòdul ha de rebre com a paràmetres d'entrada: la variable d'activació, la ruta del fitxer on es troba la informació i el canal de l'error. A continuació, en la figura 56, es poden observar tots els canals d'entrada al mòdul.

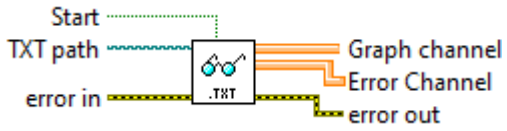

Fig. 56*. Icona i connexions del mòdul *Read TXT

Seguidament, es mostra en la figura 57, el codi del mòdul presentat. Es composa dels dos *Case Structure* habituals, que comproven l'error i l'activació. Quan ocorre, s'obre el fitxer i es mesura el nombre de bytes que té, al mateix temps s'obre l'arxiu i es llegeix el *N array* i s'ignora la resta de capçalera. En aquest punt s'entra el bucle *While*, la seqüència de passos que s'executen són:

- Primer, s'obté del fitxer, les dades corresponents a la *RB matrix*, usant el *Read RB Matrix*.
- S'obté del fitxer les dades corresponents a la *M matrix*, usant el *Read M Matrix*.
- Es mesura el nombre de bytes llegits fins al moment, i el compara amb el nombre total, per saber si s'ha acabat de llegir tot el document o no.
- Al mateix temps, s'agrupen i s'envien les dades llegides al canal dels mòduls *Graph* i *Error Caculator*.
- Per últim, es comprova el possible error generat.

Finalment, quan es donen les condicions determinades s'acaba l'execució del bucle, i es finalitza el mòdul.

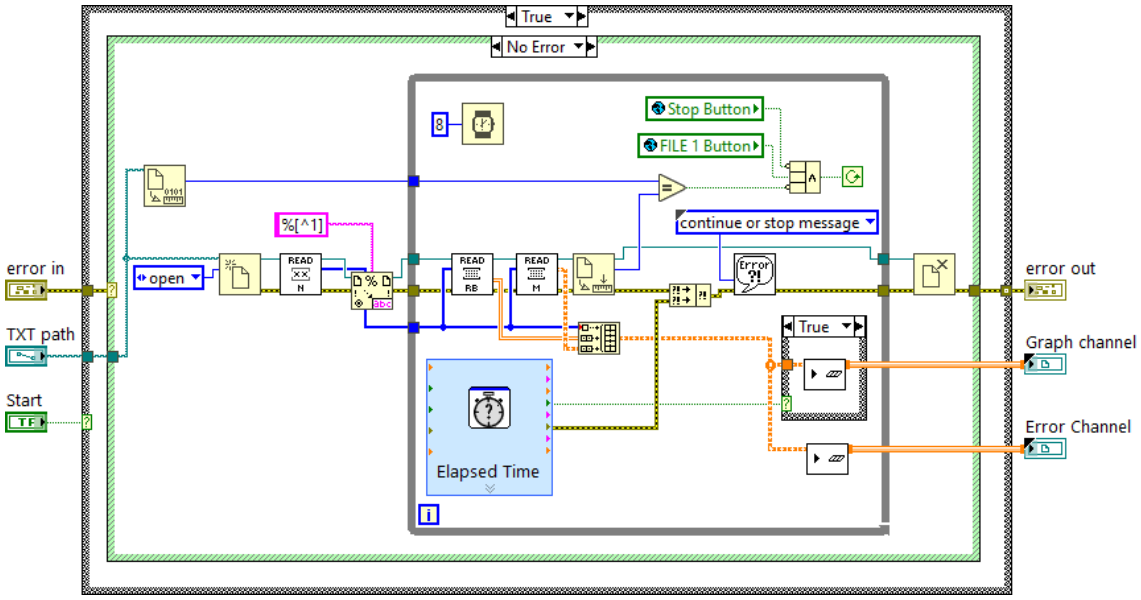

Fig. 57*. Codi del mòdul *Read TXT 1

Com a paràmetres d'eixida es troben el canal de l'error, i el *High Speed Streaming Channel* corresponents as mòduls *Graph* i *Error Calculator*.

Per últim, s'ha de destacar que el mòdul s'utilitza dues vegades diferents en la interfície, però el codi dels mòdul *Read TXT 1* i *Read TXT 2* no és idèntic. Encara que no existeix quasi ninguna

diferencia entre els dos, hi ha una variable que canvia d'un mòdul a un altre. Aquesta variable es tracta de la variable d'activació *FILE 1 Button* i *FILE 2 Button*. A causa de que s'usa per a controlar el *Stop* del bucle *While*, s'ha de canviar per tal de que cada mòdul responga correctament al seu propi control. La diferència entre els dos codi es pot observar comparant les figures 57 i 58, cadascuna corresponent a un dels mòduls.

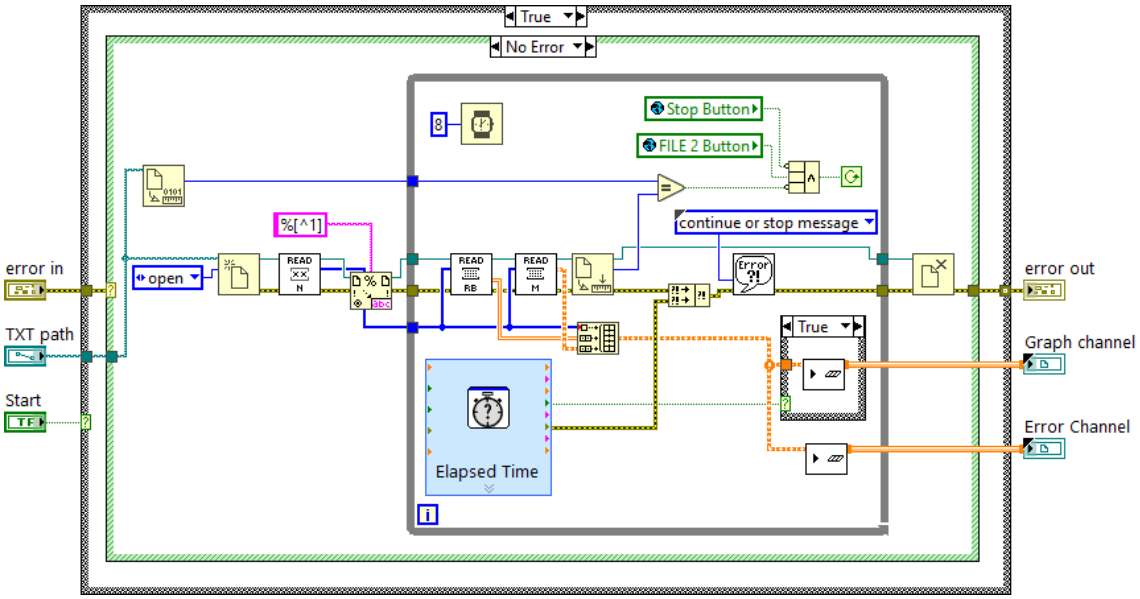

Fig. 58. Codi del mòdul Read TXT 2

1.2.6.1. Read N Array

Aquest submòdul s'integra dins del codi *Read TXT*, i s'encarrega d'extraure del fitxer el *N array*, que conté les dades sobre el nombre de sòlids rígids i marcadors hi ha i com s'organitzen. En la figura 59, es poden veure els canals d'entra al submòdul, que solament són la referència de l'arxiu a llegir i el canal d'error.

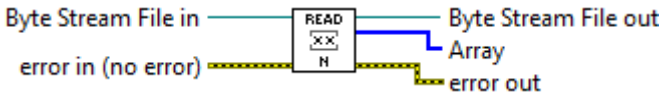

Fig. 59*. Icona i connexions del submòdul *Read N Array

Tot seguit, en la figura 60, es troba el codi corresponent al submòdul *Read N array*. El primer que es realitza és la lectura del nombre de sòlids rígids, i posteriorment (sabent la dada anterior) s'obté el nombre de marcadors que té cadascun. Per últim, s'ajunten tots els valor en un únic vector. D'aquesta forma, els paràmetres d'eixida són el propi *N array*, la referència del fitxer i el possible error.

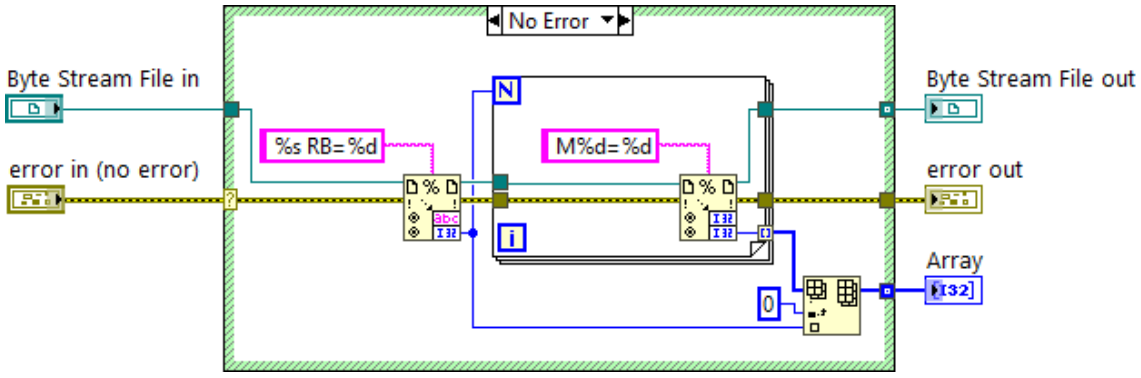

Fig. 60*. Codi del submòdul *Read N Array

1.2.6.2. Read RB Matrix

El submòdul *Read RB Matrix*, es basa en obtenir de l'arxiu TXT la matriu de dades corresponent als sòlids rígids. Per fer-ho, necessita com a paràmetres d'entrada: el *N array*, la referència del fitxer i el canal de control de l'error, com es mostra seguidment:

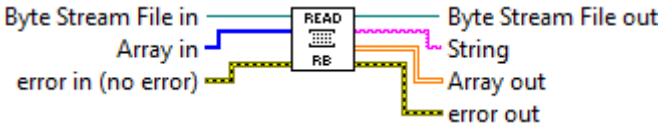

Fig. 61*. Icona i connexions del mòdul *Read RB Matrix

En la figura 62, es troba el codi del submòdul presentat. Està format, bàsicament, per un bucle *For* que per cada iteració llegeix de l'arxiu TXT: l'hora de la captura de moviment, la posició i l'orientació del sòlid. El nombre d'iteracions ve determinat pel nombre de sòlids que hi haja, valor que s'obté del *N array*. Finalment, tota la informació obtinguda s'envia per canals d'eixida, els quals són: la referència, el canal d'error, el *timecode string* i el *RB matrix*.

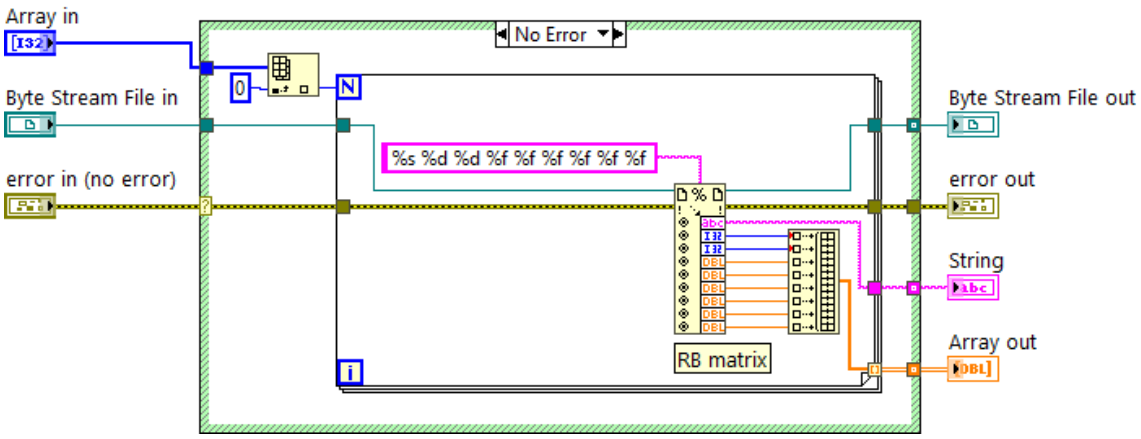

Fig. 62*. Codi del submòdul *Read RB Matrix

1.2.6.3. Read M Matrix

L'últim submòdul de *Read TXT*, és l'encarregat de llegir a matriu de marcadors completa de l'arxiu TXT que li arriba com a referència. Les entrades són: la pròpia referència al fitxer, el *N array* i l'error, com es pot veure en la figura 63:

Fig. 63*. Icona i connexions del submòdul *Read M Matrix

En la figura 64, es mostra el codi corresponent a *Read M Matrix*. Es composa principalment de dos bucles *For* niats, el primer es correspon al nivell dels sòlids rígids, i el segon al dels marcadors. El que significa que el primer bucle s'executa el nombre de vegades igual que sòlids rígids existeixen, i el segon el nombre de vegades igual que marcadors tinga cada sòlid. La informació sobre aquests valor s'extrau del *N array*, que és el primer que es llegeix.

Les matrius de marcadors que s'obtenen del segon bucle, es van concatenant en una gran matriu. Quan s'arriba a l'últim sòlid rígid, el *Case Structure* canvia al cas vertader i es connecta la matriu resultant a l'eixida i no a seguir concatenant-se. D'aquesta forma entrega com a paràmetres d'eixida la *M Matrix* completa, el possible error generat i la referència del fitxer TXT llegit.

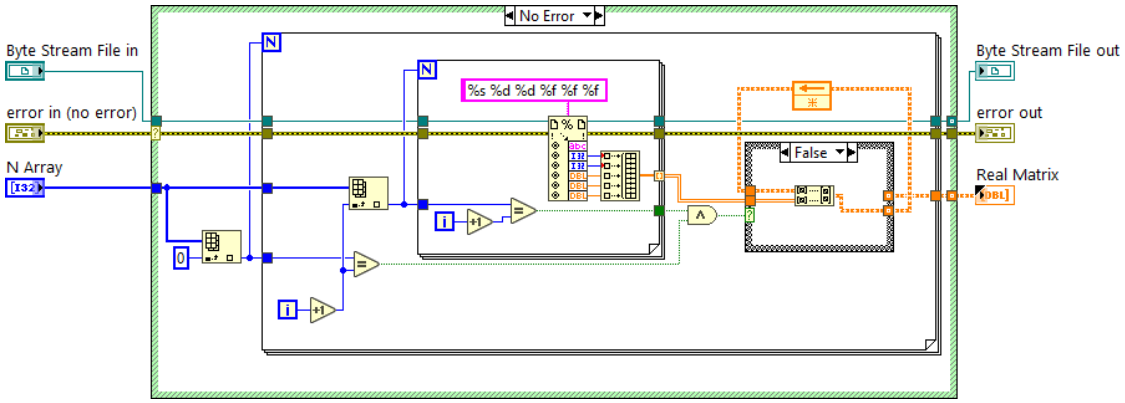

Fig. 64*. Codi del submòdul *Read M Matrix

1.2.7. Save

Aquest mòdul s'utilitza en l'aplicació dues vegades en la interfície, ja que guarda les dades de posició rebudes tant per part de la connexió TCP com del Motive. El funcionament es basa en crear o obrir un fitxer tipus TXT on s'escriu tota la informació que li arriba, en el format que indique l'usuari. Aquest, proporciona: el nom i la ruta de l'arxiu on es va a guardar, les dades de l'interval de temps d'estudi i els format de guardat de la informació: si sols són d'interès els CDM sòlids rígids o tots els marcadors que els formen.

Per complir amb aquesta funcionalitat el mòdul ha de rebre com a paràmetres la ruta del fitxer TXT, el mode o forma de guardat, el canal amb les dades de posició, i els habituals canal d'error i variable d'activació. En la següent figura 65, es presenten totes les connexions i la icona que posseeix el mòdul presentat:

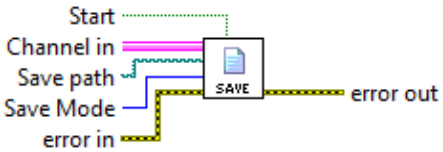

***Fig. 65*. Icona i connexions del mòdul Save**

Tot seguit, en la figura 66, es presenta el codi del mòdul *Save*, les funcions corresponents al tractament d'arxius es troben en la paleta *Functions > Programming > File I/O*. Com s'observa es forma per els dos Case Structures habituals, i a continuació es troba la funció *Open/Create File*, que obri l'arxiu i passa la referencia al bucle *While*, on s'escriuran un paquet de dades per cada iteració. La seqüencia que segueix el codi es la següent:

A. Primer, s'avalua l'estat de la variable d'activació i la del *Stop*, per comprovar que si s'ha de seguir escrivint, o en cas contrari pasar al *true Case* on no hi ha cap codi, llavors no ocorre res. A més, també és necessari comprovar l'estat, ja que són aquestes les variables que controlen la finalització del bucle.
B. Després, s'accedeix al paquet de dades de la mostra i es mana la informació al seguent *Case Structure*. En aquest, s'avalua el valor de la variable *Save Mode* que marcarà el format que tindrà el document. En la figura 66 es mostra el codi per al mode Normal (Case 0) i en la figura 67 el codi del mode Only RB (Case 1).
C. Segons el format seleccionat, les dades es manen a les funcions i als submòduls corresponents, que escriuen la informació al fitxer.
D. Finalment, es comprova si s'ha produït algun tipus d'error en la execució.

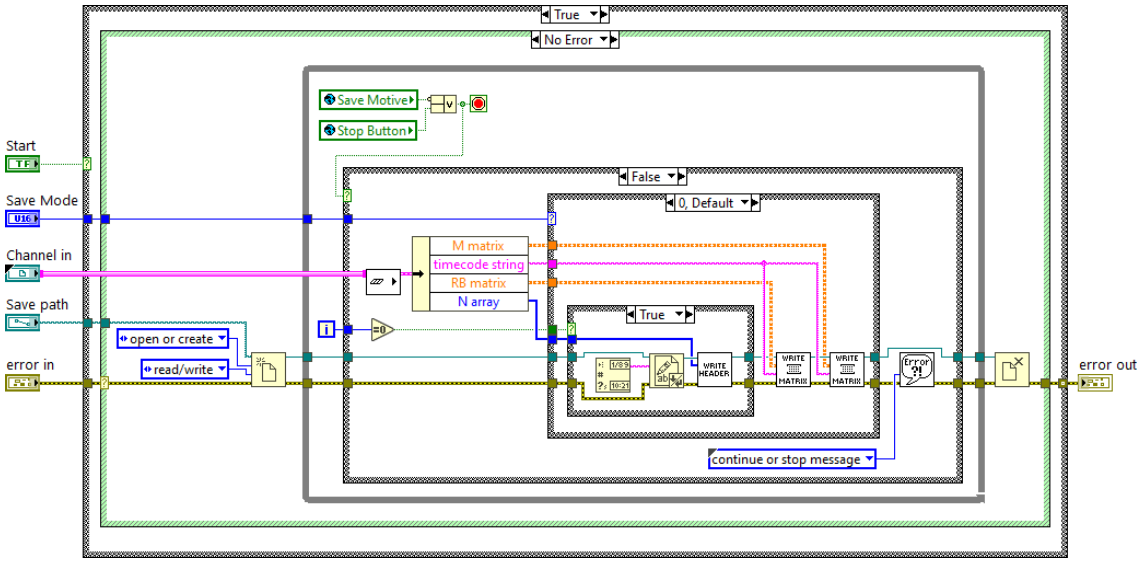

Fig. 66*. Codi del mòdul *TCP Save

Quan es compleixen les condicions mencionades anteriorment, es para el bucle i es tanca el fitxer i la referència oberta. Com a paràmetre d'eixida s'entrega el possible error generat en el procés.

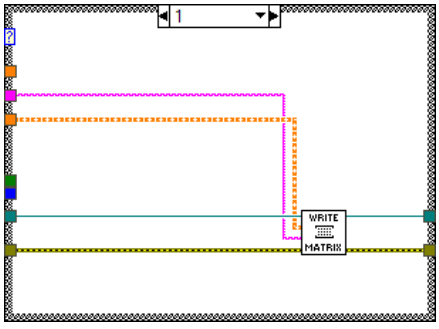

Fig. 67*. Codi del *Case 1* del 4t nivell de *Case Structure* corresponent al mode *Only RB

Cal remarcar, que la funció *Get Date/Time String* i el submòdul *Write Header,* sols s'executen en la primera iteració del bucle *While* d'escritura, ja que sols s'escriu la data i la capçalera del fitxer TXT una vegada. La funció encarregada d'obtenir la data actual d'execució es pot trobar en *Functions > Programming > Timming*.

Per últim, és important destacar que igual que amb el mòdul anterior, *Open TXT*, quan el mòdul s'usa per al TCP o Motive, la variable global associada s'ha de modificar per al cas concret. Aquest fet es deu a que la variable corresponent d'activació sí que intervé en el control del bucle *While*, llavors ha de ser la adient per a cada cas. També implica haver de guardar el mòdul amb altre nom, ja que el codi es modifica (encara que ínfimament) i no són estrictament el mateix arxiu.

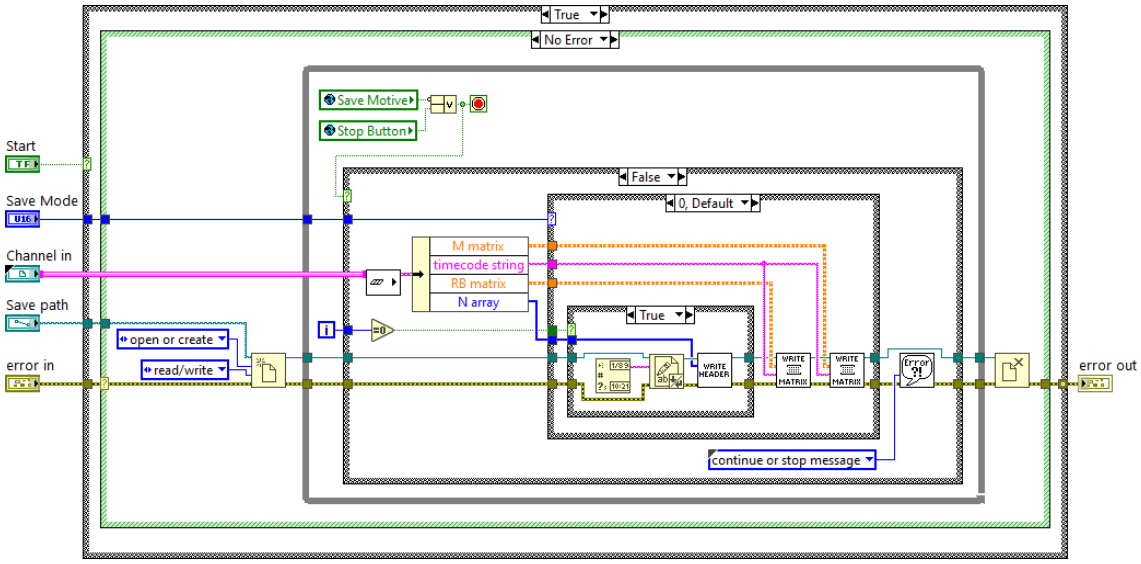

Fig. 68*. Codi del mòdul *Motive Save

1.2.7.1. Write Header

Aquest submòdul pertany a *Save*, i s'encarrega d'escriure en el fitxer TXT que li arriba com a referència, la capçalera de dades. Com a paràmetres d'entrada rep: la referència del l'arxiu amb el que ha de treballar i el possible error anterior, com es pot observar en la següent figura:

Fig. 69*. Icona i connexions del submòdul *Write Header

A continuació, en la figura 70, es presenta el codi que composa aquest submòdul. S'usen les funcions de les paletes *String i File I/O*. El primer pas quan li arriba la referència és escriure el *N array*, el primer element del qual (i=0) és el corresponent als RB, i el valor s'escriu en el format definit en el *true Case*; per a la resta d'elements el format és el mostrat en la figura 70.

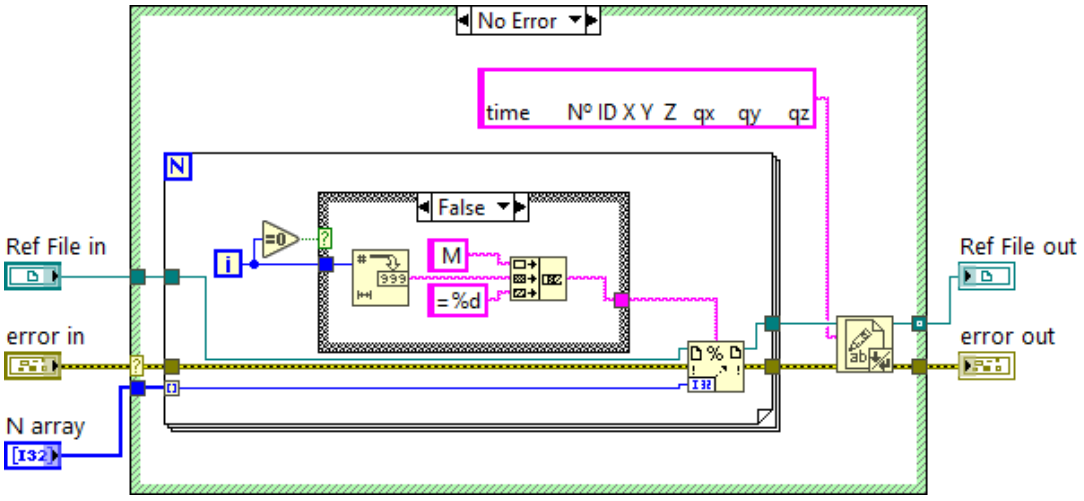

Fig. 70*. Codi del submòdul *Write Date

Quan es finalitza l'escriptura del *N array*, s'escriu la capçalera que tindrà el fitxer, definit com a una constant de *strings*. Finalment, es passa com a paràmetres d'eixida: la referència de l'arxiu TXT, i el canal de control de l'error.

1.2.7.2. Write Matrix

L'últim submòdul pertanyent al *Save*, és el responsable de guardar les matrius i el *timecode string* que li arriben, en el fitxer TXT. S'encarrega de guardar tant les *RB matrix*, com les *M matrix*. Com a paràmetres d'entrada està: la matriu que es vol escriure, el *timecode string*, la referència de l'arxiu i el canal d'error habitual. Aquestes entrades es poden observar en la figura 71:

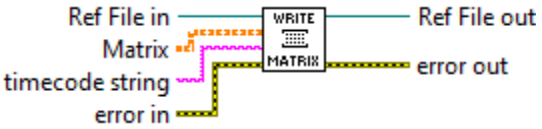

Fig. 71*. Icona i connexions del submòdul *Write Matrix

El codi de *Write Matrix* es presenta en la figura 72, es composa de dos bucles *For* niats. El primer transforma la matriu d'entrada en una successió de vectors, és a dir, cada iteració el bucle llegeix una filera de la matriu diferent. El segon bucle, treballa de forma idèntica a l'anterior, però transformant el vector d'entrada a una successió de valors, el que vol dir que en cada iteració llegeix un element del vector diferent.

A l'hora d'escriure cada valor, primer s'escriu un *Intro* al fitxer, i després s'avalua si el valor a escriure es correspon a un enter (*int*) o a un real (*float*). La raó és no escriure, per exemple, l'identificador del marcador (que és un nombre enter) amb una coma decimal seguida de zeros, perquè dificultaria el visionat posterior del fitxer. Finalment, estan el paràmetres d'eixida: la referència de l'arxiu i el possible error generat.

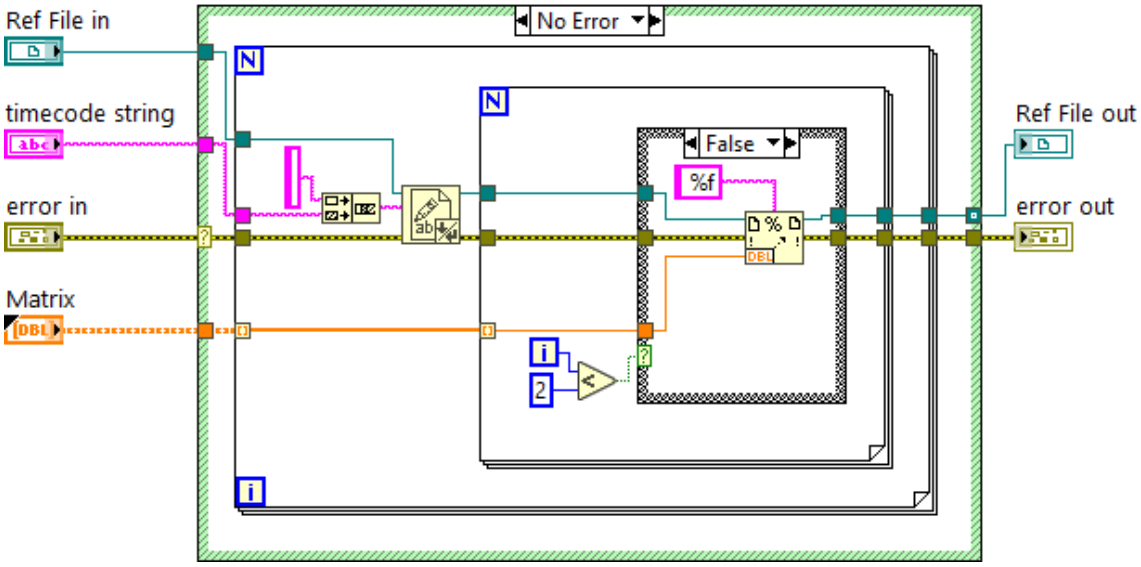

Fig. 72*. Codi del submòdul *Write Header

CAPÍTOL 2. MÒDUL *KINEMATICS*

En el present capítol, es pretén explicar en profunditat tot el codi corresponent al mòdul *Kinematics* desenvolupat. Aquest, calcula posició i orientació de la plataforma del robot, i la envia a la interfície dissenyada a través d'una connexió TCP/IP. Per realitzar els càlculs, parteix de les mesures dels encoders incrementals, les quals llegeix d'un fitxer del tipus TXT. El codi de mòdul s'ha dividit en dues parts, per facilitar la comprensió d'aquest.

La primera part és la que es mostra en la figura 73, que és la responsable d'actualitzar el valor de la variable global del mòdul, al mateix temps que obté totes les mesures dels sensors interns continguts en l'arxiu TXT. L'estructura és molt similar a la de la interfície, es troba una *Flat Sequence* que assegura que el primer que s'executa es la desactivació del botó *Stop*, per evitar comportaments indesitjats del mòdul. Seguidament, està el primer bucle *While* amb la *Even Structure* que actualitza el valor de les variables, que conté al interior, quan es polsa un dels botons. També hi ha un altre bucle *While*, que s'encarrega d'actualitzar l'indicador *Sending* del panell principal.

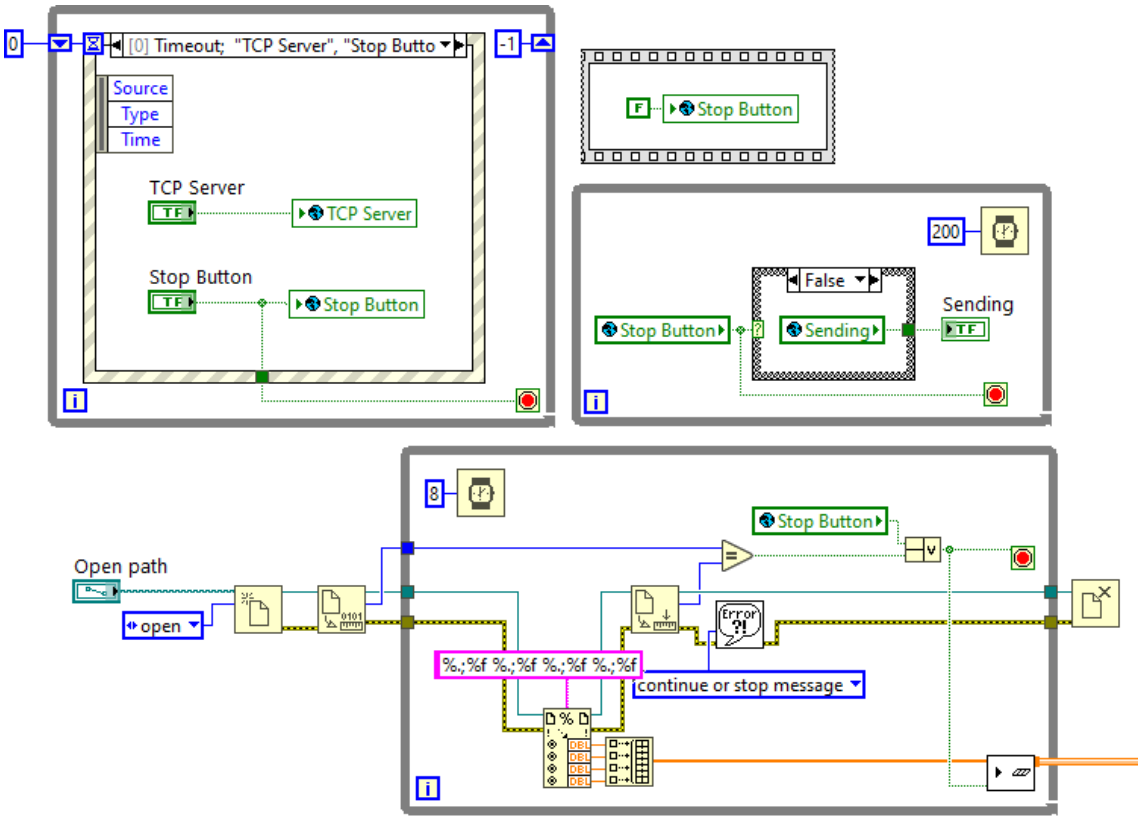

***Fig. 73*. Codi *Kinematics* 1: Actualització de variables i lectura de fitxers**

Per últim, existeix un tercer bucle *While* que obre el fitxer, de la ruta indicada per l'usuari, el qual llegeix les dades que conté. A més, s'encarrega d'enviar-els a la segona part del codi, utilitzant el *High Speed Straeming Channel* ja explicat. Totes les funcions que s'utlitzen en aquest bucle es troben en la paleta *Functions > Programming > File I/O*. Per saber quan s'ha acabat de llegir l'arxiu, es compara el nombre de bytes llegits versus el nombre de bytes totals del fitxer. Finalment, destacar d'aquesta part que degut a que els decimals dels fitxer estan separats per punts i no comes, s'ha hagut de definit el format de dades a llegir amb un punt i una coma.

La segona part del codi es troba representada en la figura 74. El bucle *While* de l'esquerra s'encarrega de calcular, a partir de les dades definides per l'usuari en el *Front Panel* i les dades del fitxer TXT, la posició del CG de la plataforma i de 3 punts més de la plataforma. Els passos que segueix en cada iteració són:

- Es llegeix la informació provenint de l'arxiu TXT i del *Front Panel*. S'envien al *MathScript Node* on es defineixen els paràmetres necessaris per a realitzar el càlculs.
- A continuació, es calcula el CG i els vèrtex A, B i C de la plataforma, usant diverses funcions definides en el *MathScript Node*.
- S'organitzen les dades obtingues del CG en la *RB Matrix*, i la informació calculada dels vèrtexs, en la *M Matrix* com marcadors de la plataforma.
- Per últim, s'obté l'hora exacta en eixe instant, es defineix el vector amb el nombre de sòlids i marcadors, i s'agrupen totes aquestes dades en el clúster de dades, per enviar-les usant el *High Speed Stream Channel*.

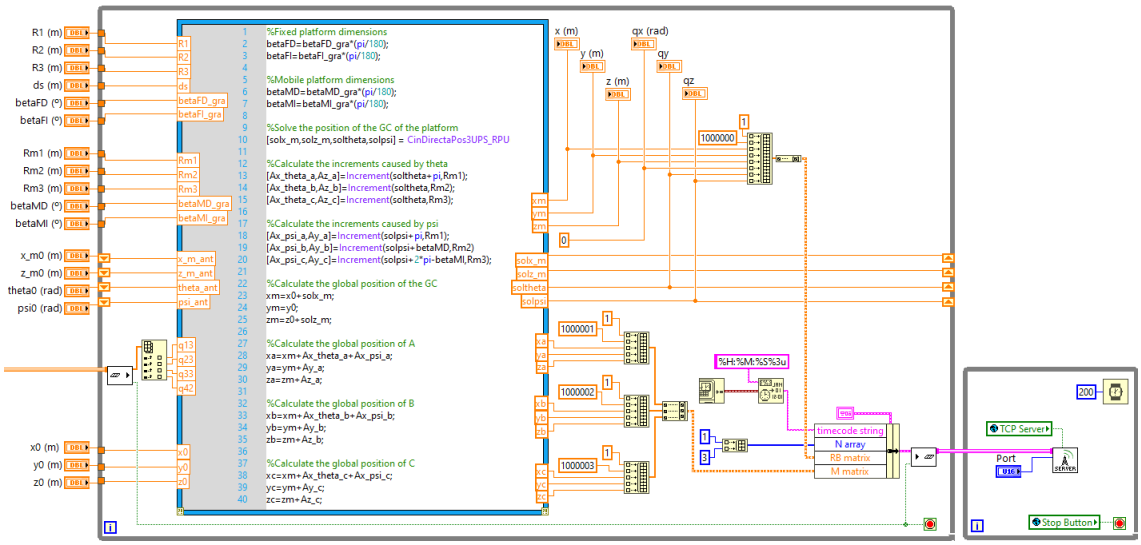

Fig. 74. **Codi *Kinematics* 2: *MathScript Node* i enviament de dades**

Es remarca l'ús del *Shift Register* del bucle, el qual s'encarrega de passar el valor entre la iteració anterior i l'actual de 4 variables, ja que el model cinemàtic les necessita per poder convergir a la solució.

Finalment, el clúster de dades s'envia al mòdul TCP Server, idèntic al de la interfície, encarregat de transmetre usant els protocols TCP/IP a la pròpia interfície comentada.

2.1. MATHSCRIPT NODE

Aquesta funció és la que conté tot el codi encarregat de realitzar el càlculs necessaris per obtenir la posició i orientació de la plataforma, partint dels paràmetres d'entrada requerits. Tot seguit, es comenta el procediment que es segueix cada vegada que s'executa el codi escrit dins del *MathScript Node*:

- Primerament, es defineixen els paràmetres geomètrics corresponents amb la configuració amb la que es treballe. Els paràmetres lineals (*R1, R2, R3, Rm1, Rm2, Rm3*), s'usen directament en el model cinemàtic, però per al cas del angulars (*betaFD, betaFI, betaMD, betaMI*) s'han de passar a radians.
- La resta de paràmetres també s'usen sense ninguna conversió d'unitats prèvia al model. Aquest són: la posció global (*x0, y0, z0*), les mesures dels encoders (*q13, q23, q33, q42*) i la convergència del model (*x_m_ant, z_m_ant, theta_ant, psi_ant*).
- A continuació, es crida a la funció *CinDirectaPos3UPS_RPU*, que amb totes les dades definides anteriorment es capaç de resoldre la posició i orientació del robot entregant com a solució: *solx_m, soly_m, soltheta* i *solpsi*.
- Es calcula com varien les coordenades de posició dels vèrtex de la plataforma, front al canvi d'orientació calculat. D'aquesta forma s'obtenen les variacions en els eixos XYZ de es 3 vèrtex introduint soltheta i solpsi, mitjançant la funció desenvolupada *Increments*.
- S'obté la posició final global de CG (*xm, ym, zm*), sumant-li a la posició global inicial l'increment calculat amb el model.
- Es fa el mateix procediment per als vèrtex, però a més afegint els increments calculats anteriorment, d'aquesta forma s'obté la posició final global d'aquest (*xa, ya, za*...).

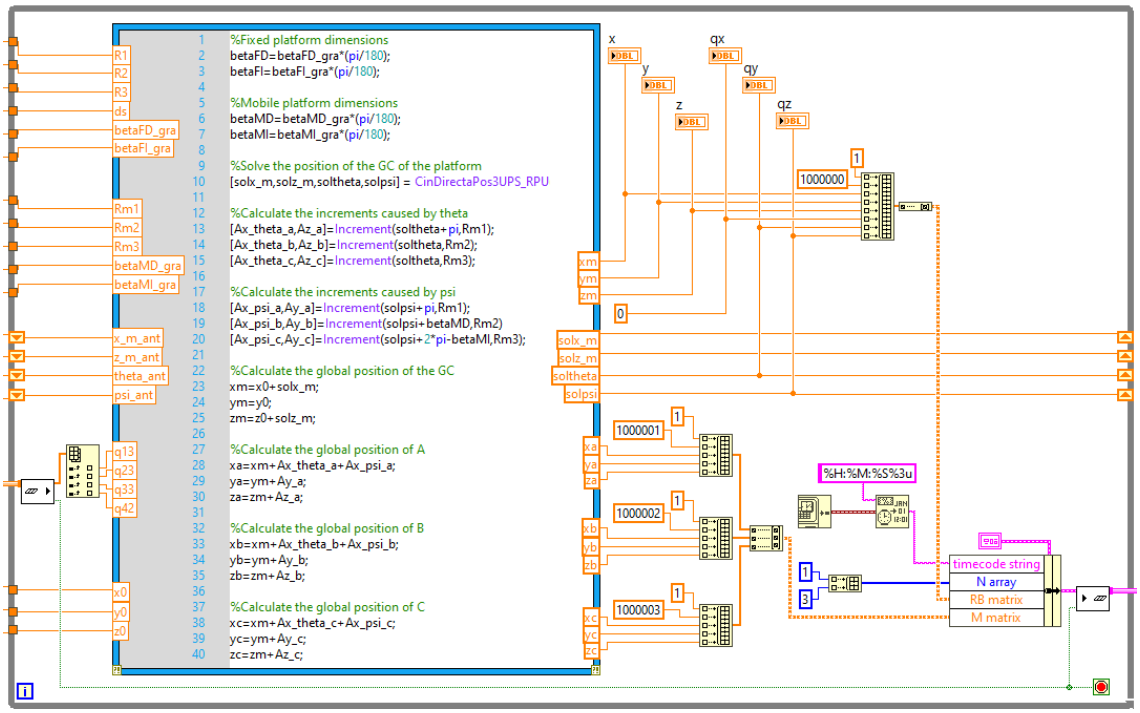

Fig 75*. Bucle *While* amb del *MathScript Node* del mòdul *Kinematics

El codi del MathScript Node, i el de totes les funcions que intervenen, es troben arreplegats en l'apartat de *Funcions matemàtiques tipus .m* del Manual.

2.2. TCP SERVER

El mòdul encarregat de comunicar-se amb la interfície dissenyada és el mòdul TCP ja presentat en capítols anteriors. Els canal d'entrada i eixia del mòdul estan representats en la figura 76, que es mostra a continuació:

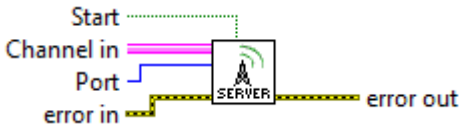

Fig. 76*. Icona i connexions del mòdul *TCP Server

Tot seguit, en la figura 77, es presenta el codi del mòdul *TCP Server*. L'estructura és idèntica a la presentada anteriorment, a diferencia de les variables que controlen el procés es corresponen a una variable global diferent que actua en el mòdul *Kinematics*, la qual es presenta en el següent punt.

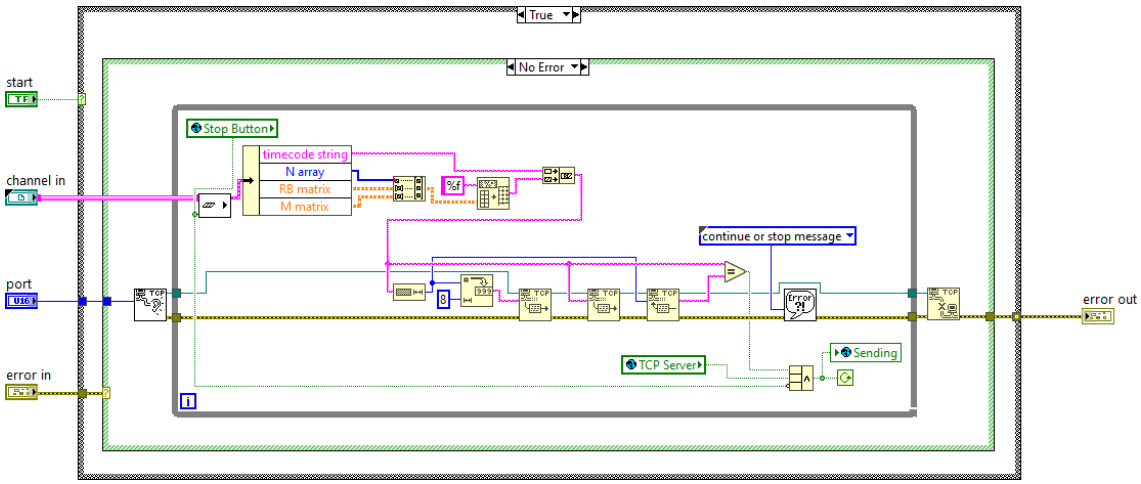

Fig. 77*. Codi del mòdul *TCP Server

2.3. VARIABLE GLOBAL *KINEMATICS*

Per a poder controlar el mòdul TCP Server, en el codi del mòdul *Kinematics* s'ha utilitza una variable global amb els botons de control: *TCP Server* i *Stop*, a més també es troba l'indicador *Sending*, que s'activa quan s'està produint la comunicació via TCP/IP. Aquests es poden observar en la següent figura:

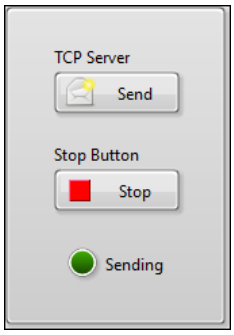

Fig. 78*. Variable *Global Kinematics

OBJECTES .NET DEL MOTIVE

Objecte	Constructor	Propietat	Mètode
AnalogChannelData	AnalogChannelData()	nFrames	Equals(Object obj)
	AnalogChannelData(AnalogChannelData source)	Values	GetHashCode()
			GetType()
			ToString()
ConnectionType	ConnectionType()	[]Multicast	CompareTo(Object target)
		[]Unicast	Equals(Object obj)
		value_	GetHashCode()
			GetType()
			GetTypeCode()
			HasFlag(Enum flag)
			ToString()
			ToString(IFormatProvider provider)
			ToString(String format)
			ToString(String format, IFormatProvider provider)
DataDescriptor	DataDescriptor()	type	Equals(Object obj)
			GetHashCode()
			GetType()
			ToString()
DataDescriptorType	DataDescriptorType()	[S]eDeviceData	CompareTo(Object target)
		[S]eForcePlateData	Equals(Object obj)
		[S]eMarkerSetData	GetHashCode()
		[S]eRigidbodyData	GetType()
		[S]eSkeletonData	GetTypeCode()
		value_	HasFlag(Enum flag)
			ToString()
			ToString(IFormatProvider provider)
			ToString(String format)
			ToString(String format, IFormatProvider provider)
Device	Device()	ChannelCount	Equals(Object obj)

Objecte	Constructor	Propietat	Mètode
		ChannelDataType	GetHashCode()
		ChannelNames	GetType()
		DeviceType	ToString()
		ID	
		Name	
		Serial	
		type	
DeviceData	DeviceData()	ChannelData	Equals(Object obj)
	DeviceData(DeviceData source)	ID	GetHashCode()
		nChannels	GetType()
		params	ToString()
DiscoveredServer	This class contains no public constructors.		
ForcePlate	ForcePlate()	CalMatrix	Equals(Object obj)
		ChannelCount	GetHashCode()
		ChannelDataType	GetType()
		ChannelNames	ToString()
		Corners	
		ID	
		Length	
		OriginX	
		OriginY	
		OriginZ	
		PlateType	
		Serial	
		type	
		Width	
ForcePlateData	ForcePlateData()	ChannelData	Equals(Object obj)
	ForcePlateData(ForcePlateData source)	ID	GetHashCode()
		nChannels	GetType()
		params	ToString()
FrameOfMocapData	FrameOfMocapData()	bRecoding	CopyArrays(FrameOfMocapData source)
	FrameOfMocapData(FrameOfMocapData source)	bTrackingModelsChanged	Equals(Object obj)
		CameraDataReceivedTimestamp	GetHashCode()
		CameraMidExposureTimestamp	GetType()
		Devices	InitArrays()
		ForcePlates	ToString()

Objecte	Constructor	Propietat	Mètode
		fTimestamp	
		iFrame	
		LabeledMarkers	
		MarkerSets	
		nDevices	
		nForcePlates	
		nMarkers	
		nMarkerSets	
		nOtherMarkers	
		nRigidBodies	
		nSkeletons	
		OtherMarkers	
		RigidBodies	
		Skeletons	
		Timecode	
		TimecodeSubframe	
		TransmitTimestamp	
FrameReadyEventHandler	FrameReadyEventHandler(Object A_0, IntPtr A_1)	Method	BeginInvoke(FrameOfMocapData data, NatNetClientML client, AsysncCallback, Object obj)
		Target	Clone()
			DynamicInvoke(Object[] args)
			EndInvoke(IAsyncResult result)
			Equals(Object obj)
			GetHashCode()
			GetInvocationList()
			GetObjectData(SerializationInfo info, StreamingContext context)
			GetType()
			Invoke(FrameOfMocapData data, NatNetClientML client)
			ToString()
FrameReadyEventHandler2	FrameReadyEventHandler2(Object A_0, IntPtr A_1)	Method	BeginInvoke(FrameOfMocapData data, NatNetClientML client, AsysncCallback, Object obj)
		Target	Clone()
			DynamicInvoke(Object[] args)
			EndInvoke(IAsyncResult result)
			Equals(Object obj)
			GetHashCode()
			GetInvocationList()
			GetObjectData(SerializationInfo info, StreamingContext context)
			GetType()
			Invoke(FrameOfMocapData data, NatNetClientML client)

Objecte	Constructor	Propietat	Mètode
			ToString()
Marker	Marker()	ID	Equals(Object obj)
	Marker(Marker source)	parameters	GetHashCode()
		residual	GetType()
		size	ToString()
		x	
		y	
		z	
MarkerSet	MarkerSet()	MarkersNames	Equals(Object obj)
		Name	GetHashCode()
		nMarkers	GetType()
		type	ToString()
MarketSetData	MarkerSetData()	Markers	Equals(Object obj)
	MarkerSetData(MarkerSetData source)	MarkerSetName	GetHashCode()
		nMarkers	GetType()
			ToString()
NATEulerOrder	NATEulerOrder()	[S]NAT_XYXr	CompareTo(Object target)
		[S]NAT_XYXs	Equals(Object obj)
		[S]NAT_XYZr	GetHashCode()
		[S]NAT_XYZs	GetType()
		[S]NAT_XZXr	GetTypeCode()
		[S]NAT_XZXs	HasFlag(Enum flag)
		[S]NAT_XZYr	ToString()
		[S]NAT_XZYs	ToString(IFormatProvider provider)
		value_	ToString(String format)
			ToString(String format, IFormatProvider provider)
NatNetClientML	NatNetClientML()		Connect(NatNetClientML+ConectParams Params)
	NatNetClientML(Int32 iConnectionType)		[S]DecodeID(Int32 inSourceID, Int32&entityID, Int32&memberID)
			[S]DecodeTimecode(UInt32inTimecode...)
			Disconnect()
			Dispose()
			Equals(Object obj)
			GetDataDescriptions()
			GetDataDescriptions(…)
			GetHashCode()
			GetLastFrameOfData()
			GetServerDescription(ServerDescription Desc)
			GetType()
			Initialize(String LocalAddress, String ServerAddress)
			Intialize(String LocalAddress, String ServerAddress, Int32 Port)

Objecte	Constructor	Propietat	Mètode
			NatNetVersion()
			[S]QuatToEuler(Single[] queternion, NATEulerOrder eulerOrder)
			SecondsSinceHostTimestamp(UInt64 hostTimestamp)
			SendMessageAndWait(String Message, Byte[]& ServerResponse, Int32& ResponseSize)
			SendMessageAndWait(String Message, Int32 TimeOut, Byte[]& ServerResponse, Int32& ResponseSize)
			SendMessageAndWait(String Message, Int32 &ResponseCode)
			ToString()
			Uninitialize()
NatNetClientML+ConnectParams	ConnectParams()	ConnectionType	Equals(Object obj)
		LocalAddress	GetHashCode()
		MulticastAddress	GetType()
		ServerAddress	ToString()
		ServerCommandPort	
		ServerDataPort	
NatNetEventArgs	NatNetEventArgs(NatNetClientML inClient, FrameOfMocapData inData)	client	Equals(Object obj)
		data	GetHashCode()
			GetType()
			ToString()
NatNetPacket	NatNetPacket()	MessageBytes	Equals(Object obj)
		MassageId	GetHashCode()
		Payload	GetType()
			ToString()
NatNetServerDiscovery	NatNetServerDiscovery()		EndDiscovery()
			Equals(Object obj)
			GetHashCode()
			GetType()
			StartDiscovery()
			ToString()
RigidBody	RigidBody()	ID	Equals(Object obj)
		MarkerPositions	GetHashCode()
		MarkerRequiredLabels	GetType()

Objecte	Constructor	Propietat	Mètode
		Name	ToString()
		nMarkers	
		offsetx	
		offsety	
		offsetz	
		parentID	
		type	
RigidBodyData	RigidBodyData()	ID	Equals(Object obj)
	RigidBodyData(RigidBodyData source)	MeanError	GetHashCode()
		qw	GetType()
		qx	ToString()
		qy	
		qz	
		Tracked	
		x	
		y	
		z	
ServerDescription	ServerDescription()	bConnectionInfoValid	Equals(Object obj)
		ConnectionDataPort	GetHashCode()
		ConnectionMultiCast	GetType()
		ConnectionMulticastAddress	ToString()
		HighResClockFrequency	
		HostApp	
		HostAppVersion	
		HostComputerAddress	
		HostComputerName	
		HostPresent	
		NatNetVersion	
ServerDiscoveredHandler	ServerDiscoveredHandler(Object A_0, IntPtr A_1)	Method	BeginInvoke(FrameOfMocapData data, NatNetClientML client, AsysncCallback, Object obj)
		Target	Clone()
			DynamicInvoke(Object[] args)
			EndInvoke(IAsyncResult result)
			Equals(Object obj)
			GetHashCode()
			GetInvocationList()
			GetObjectData(SerializationInfo info, StreamingContext context)
			GetType()

Objecte	Constructor	Propietat	Mètode
			Invoke(FrameOfMocapData data, NatNetClientML client)
			ToString()
Skeleton	Skeleton()	ID	Equals(Object obj)
		Name	GetHashCode()
		nRigidBodies	GetType()
		RigidBodies	ToString()
		type	
SkeletonData	SkeletonData()	ID	Equals(Object obj)
	SkeletonData(SkeletonData source)	nRigidBodies	GetHashCode()
		RigidBodies	GetType()
			ToString()
UnknownMessageEventHandler	UnknownMessageEventHandler(Object A_0, IntPtr A_1)	Method	BeginInvoke(FrameOfMocapData data, NatNetClientML client, AsysncCallback, Object obj)
		Target	Clone()
			DynamicInvoke(Object[] args)
			EndInvoke(IAsyncResult result)
			Equals(Object obj)
			GetHashCode()
			GetInvocationList()
			GetObjectData(SerializationInfo info, StreamingContext context)
			GetType()
			Invoke(FrameOfMocapData data, NatNetClientML client)
			ToString()

OBJECTES ACTIVEX DEL CWGRAPH3D

Objecte	Propietat	Mètode
CWGraph3D	AmbientLightColor	AboutBox
	Axes	ClearData
	BackColor	ControlImage
	Caption	ControlImageEx
	CaptionColor	ExportStyle
	ClipData	ImportStyle
	Cursors	Plot3DCurve
	Dither	Plot3DMesh
	Enabled	Plot3DParametricSurface
	FastDraw	Plot3DSimpleSurface
	Font	SetDefaultView
	GraphFrameColor	
	GraphFrameVisible	
	GridFrameColor	
	GridSmoothing	
	GridXY	
	GridXZ	
	GridYZ	
	ImmediateUpdates	
	KeyboardMode	
	Lighting	
	Lights	
	PlotAreaColor	
	Plots	
	PlotTemplate	
	ProjectionStyle	
	ReadyState	
	TrackMode	
	Use3DHardwareAcceleration	
	ViewAutoDistance	
	ViewDistance	
	ViewLatitude	
	ViewLongitude	
	ViewMode	
	ViewXCenter	
	ViewYCenter	
	ViewZCenter	
	Windowless	
Axes	Count	Item

Objecte	**Propietat**	**Mètode**
Cursos	Count	Add
		Item
		Remove
		RemoveAll
Font	Bold	
	Charset	
	Italic	
	Name	
	Size	
	Strikethrough	
	Underline	
	Weight	
Lights	Count	Item
Plots	Count	Add
		Item
		Remove
		RemoveAll
PlotTemplate	AutoScale	ClearData
	CacheData	Plot3DCurve
	ColorMapAutoScale	Plot3DMesh
	ColorMapColors	Plot3DParametricSurface
	ColorMapInterpolate	Plot3DSimpleSurface
	ColorMapLog	Plot3DSurface
	ColorMapStyle	
	ColorMapValues	
	Contours	
	CoordinateSystem	
	Enabled	
	FillColor	
	FillStyle	
	LineColor	
	LineStyle	
	LineWidth	
	MultiPlot	
	Name	
	PointColor	
	PointFrequency	
	PointSize	
	PointStyle	
	Transparency	
	ProjectionXY	
	ProjectionXZ	
	ProjectionYZ	
	ShowProjectionsOnly	
	Style	
	Transparency	
	Visible	

Objecte	Propietat	Mètode
	Xaxis	
	Yaxis	
	Zaxis	
Item	AutoScale	ClearData
	CacheData	Plot3DCurve
	ColorMapAutoScale	Plot3DMesh
	ColorMapColors	Plot3DParametricSurface
	ColorMapInterpolate	Plot3DSimpleSurface
	ColorMapLog	Plot3DSurface
	ColorMapStyle	
	ColorMapValues	
	Contours	
	CoordinateSystem	
	Enabled	
	FillColor	
	FillStyle	
	LineColor	
	LineStyle	
	LineWidth	
	MultiPlot	
	Name	
	PointColor	
	PointFrequency	
	PointSize	
	PointStyle	
	Transparency	
	ProjectionXY	
	ProjectionXZ	
	ProjectionYZ	
	ShowProjectionsOnly	
	Style	
	Transparency	
	Visible	
	Xaxis	
	Yaxis	
	Zaxis	
Contours	Anchor	Add
	AnchorEnabled	Item
	Basis	Remove
	Count	RemoveAll
	Interval	SetLabelColor
	LabelOrientationStyle	SetLabelFont
	LevelList	SetLabelFormat
	Levels	SetLabelVisible
		SetLineColor
		SetLineStyle
		SetLineWidth

Objecte	**Propietat**	**Mètode**
Xaxis	AutoScale	AutoScaleNow
	Caption	SetMinMax
	CaptionColor	
	CaptionFont	
	CaptionNormal	
	CaptionOpposite	
	CaptionOrientationStyle	
	FormatString	
	Inverted	
	Labels	
	Log	
	Maximum	
	Minimum	
	Name	
	Ticks	
	ValuePairs	
	Visible	
Labels	Color	
	Fonts	
	Normal	
	Opposite	
	OrientationStyle	
Ticks	AutoDivisions	
	Inside	
	MajorDivisions	
	MajorGrid	
	MajorGridColor	
	MajorTickColor	
	MajorTicks	
	MajorUnitsBase	
	MajorUnitsInterval	
	MinorDivisions	
	MinorGrid	
	MinorGridColor	
	MinorTickColor	
	MinorTicks	
	Normal	
	Opposite	
	Outside	
ValuePairs	Count	Add
	GridLines	Item
	LabelType	Remove
	Location	RemoveAll
	MajorTIcks	Swap

FUNCIONS MATEMÀTIQUES TIPUS .M

A. INCREMENT

```
function [Ah,Av] = Increment(angle,d)
%FUNCTION: Longitudinal increment from the rotation angle
%AUTHOR:   Eros Costa
%PROJECT: UPV - INTERFACE FOR GLOBAL POSITIONING OF A 3UPE+RPU
PARALLEL ROBOT
%DESCRIPTION:
%   Starting from the angle (angle) of rotation of the platform and
the
%   distance (d) between the GC and the vertex of the platform, it
calculates
%   the longitudinal increment in the horizontal and vertical axis
of the
%   rotation plane
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ah = d*cos(angle);
Av = d*sin(angle);
end
```

B. CINDIRECTAPOS3UPS_RPU

```
function [solx_m,solz_m,soltheta,solpsi] =
CinDirectaPos3UPS_RPU(q13,q23,q33,q42,R1,R2,R3,ds,betaFD,betaFI,Rm1,Rm
2,Rm3,betaMD,betaMI,x_m_ant,z_m_ant,theta_ant,psi_ant)
%FUNCIÓN: Cinematica Directa de Posición para Simulink
%AUTOR:   José Pulloquinga
%PROYECTO: UPV - ROBOT PARALELO 3UPE+RPU
%DESCRIPCIÓN:
%   A partir de la posición de las juntas activas (qi3 y q42 con
i=1,2,3)
%   se determina la posición (Xm, Zm) y la orientación (Theta y Psi)
del
%   centro de la plataforma del robot paralelo

%PARAMETROS PARA LA SOLUCIÓN DEL PROBLEM
%Tolerancia maxima
tol=1e-7;
%Numero maximo de iteracciones
iter=30;

%ASIGNACIÓN DE VARIABLES PARA LA RESOLUCIÓN
%posición de las juntas activas
qa=[q13;q23;q33;q42];
%vector de posición y orientación inicial de la plataforma
X=[x_m_ant;z_m_ant;theta_ant;psi_ant];
%error inicial en función del valor inicial de estimación
error=norm(CinDirEcPosicion(X(1),X(2),X(3),X(4),qa,R1,R2,R3,ds,betaFD,
betaFI,Rm1,Rm2,Rm3,betaMD,betaMI));
%Iteracion inicial
i=1;

%SOLUCIÓN PARA LA POSICIÓN DEL CENTRO DE LA PLATAFORMA
 while error>tol
     %Funcion con las ecuaciones de la cinematica directa

f=CinDirEcPosicion(X(1),X(2),X(3),X(4),qa,R1,R2,R3,ds,betaFD,betaFI,Rm
1,Rm2,Rm3,betaMD,betaMI);
     %Error de la solucióon actual
     error=norm(f);
     %Jacobiano directo

J=CDJacobian(X(1),X(2),X(3),X(4),R1,R2,R3,ds,betaFD,betaFI,Rm1,Rm2,Rm3
,betaMD,betaMI);
     %Calculo de la nueva respuesta
     Xn=X-J\f;
     %actualizo la respuesta de la posción y orientación del centro de
la
     %plataforma
     X=Xn;
     %incremento del contador de iteraciones
     i=i+1;
     %Condición para evitar bucles infinitos
     if i>iter
         break
     end
 end
solx_m = X(1);
solz_m = X(2);
soltheta = X(3);
solpsi = X(4);
```

C. CINDIRECPOSICION

```
function [f] =
CinDirEcPosicion(x_m,z_m,theta,psi,qa,R1,R2,R3,ds,betaFD,betaFI,Rm1,Rm
2,Rm3,betaMD,betaMI)
%FUNCIÓN: Ecuaciones de posición para la Cinematica Directa de
Posición
%AUTOR:   José Pulloquinga
%PROYECTO: UPV - ROBOT PARALELO 3UPE+RPU
%VERSIÓN: 1.0

%DESCRIPCIÓN:
%   A partir de la posición de las articulaciones activas (qi3 y q42
con
%   i=1,2,3) se determina la posición (Xm, Zm) y la orientación (Theta
y
%   Psi) del centro de la plataforma del robot paralelo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Asignación de las variables de las juntas activas
q13=qa(1);
q23=qa(2);
q33=qa(3);
q42=qa(4);

%Ecuaciones para resolver el problema de la cinematica directa
f = [
    q13 ^ 2 + 2 * cos(theta) * cos(psi) * R1 * Rm1 + 2 * cos(theta) *
cos(psi) * Rm1 * x_m - 2 * cos(psi) * Rm1 * sin(theta) * z_m - R1 ^ 2
- 2 * R1 * x_m - Rm1 ^ 2 - x_m ^ 2 - z_m ^ 2;
    q23 ^ 2 - 2 * sin(psi) * cos(theta) * cos(betaFD) * sin(betaMD) *
R2 * Rm2 + 2 * cos(theta) * cos(betaFD) * cos(psi) * cos(betaMD) * R2
* Rm2 + 2 * sin(psi) * cos(theta) * sin(betaMD) * Rm2 * x_m + 2 *
sin(psi) * sin(betaFD) * cos(betaMD) * R2 * Rm2 - 2 * Rm2 * sin(theta)
* sin(psi) * sin(betaMD) * z_m - 2 * cos(theta) * cos(psi) *
cos(betaMD) * Rm2 * x_m + 2 * sin(betaFD) * cos(psi) * sin(betaMD) *
R2 * Rm2 + 2 * cos(betaMD) * cos(psi) * Rm2 * sin(theta) * z_m + 2 *
R2 * x_m * cos(betaFD) - R2 ^ 2 - Rm2 ^ 2 - x_m ^ 2 - z_m ^ 2;
    q33 ^ 2 + 2 * cos(theta) * cos(betaFI) * sin(psi) * sin(betaMI) *
R3 * Rm3 + 2 * cos(theta) * cos(betaFI) * cos(psi) * cos(betaMI) * R3
* Rm3 - 2 * sin(psi) * cos(theta) * sin(betaMI) * Rm3 * x_m - 2 *
cos(theta) * cos(psi) * cos(betaMI) * Rm3 * x_m - 2 * sin(betaFI) *
sin(psi) * cos(betaMI) * R3 * Rm3 + 2 * sin(betaFI) * cos(psi) *
sin(betaMI) * R3 * Rm3 + 2 * Rm3 * sin(theta) * sin(psi) * sin(betaMI)
* z_m + 2 * cos(betaMI) * cos(psi) * Rm3 * sin(theta) * z_m + 2 * R3 *
x_m * cos(betaFI) - R3 ^ 2 - Rm3 ^ 2 - x_m ^ 2 - z_m ^ 2;
    -ds ^ 2 + 2 * ds * x_m + q42 ^ 2 - x_m ^ 2 - z_m ^ 2
    ];
end
```

D. CDJACOBIAN

```
function [J] =
CDJacobian(x_m,z_m,theta,psi,R1,R2,R3,ds,betaFD,betaFI,Rm1,Rm2,Rm3,bet
aMD,betaMI)
%FUNCIÓN: Matriz del jacobiano para resolver la Cinematica directa
%AUTOR:   José Pulloquinga
%PROYECTO: UPV - ROBOT PARALELO 3UPE+RPU
%DESCRIPCIÓN:
%   A partir de la posición de la posición (Xm, Zm) y la orientación
(Theta y
%   Psi) del centro de la plataforma del robot paralelo, se determina
la
%   matriz Jacobiana de las ecuaciones implicitas de la cinematica
directa,
%   es decir que las ecuaciones se igualan a cero. Es diferente del
%   Jacobiano directo.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Ecuaciones para la matriz del jacobiano directo
J = [
    2 * cos(theta) * cos(psi) * Rm1 - (2 * R1) - (2 * x_m) -2 *
sin(theta) * cos(psi) * Rm1 - (2 * z_m) -2 * cos(psi) * Rm1 *
(cos(theta) * z_m + sin(theta) * R1 + sin(theta) * x_m) -2 * sin(psi)
* Rm1 * (cos(theta) * R1 + cos(theta) * x_m - sin(theta) * z_m);
    2 * cos(theta) * sin(psi) * Rm2 * sin(betaMD) - 2 * cos(theta) *
cos(psi) * Rm2 * cos(betaMD) + 2 * R2 * cos(betaFD) - (2 * x_m) -2 *
sin(theta) * sin(psi) * Rm2 * sin(betaMD) + 2 * sin(theta) * cos(psi)
* Rm2 * cos(betaMD) - (2 * z_m) -2 * Rm2 * (-sin(theta) * sin(psi) *
cos(betaFD) * sin(betaMD) * R2 + sin(theta) * cos(psi) * cos(betaFD) *
cos(betaMD) * R2 + cos(theta) * sin(psi) * sin(betaMD) * z_m -
cos(theta) * cos(psi) * cos(betaMD) * z_m + sin(theta) * sin(psi) *
sin(betaMD) * x_m - sin(theta) * cos(psi) * cos(betaMD) * x_m) -2 *
Rm2 * (cos(theta) * sin(psi) * cos(betaFD) * cos(betaMD) * R2 +
cos(theta) * cos(psi) * cos(betaFD) * sin(betaMD) * R2 - cos(theta) *
sin(psi) * cos(betaMD) * x_m - cos(theta) * cos(psi) * sin(betaMD) *
x_m + sin(theta) * sin(psi) * cos(betaMD) * z_m + sin(theta) *
cos(psi) * sin(betaMD) * z_m + sin(psi) * sin(betaMD) * sin(betaFD) *
R2 - cos(psi) * cos(betaMD) * sin(betaFD) * R2);
    -2 * cos(theta) * sin(psi) * Rm3 * sin(betaMI) - 2 * cos(theta) *
cos(psi) * Rm3 * cos(betaMI) + 2 * R3 * cos(betaFI) - (2 * x_m) 2 *
sin(theta) * sin(psi) * Rm3 * sin(betaMI) + 2 * sin(theta) * cos(psi)
* Rm3 * cos(betaMI) - (2 * z_m) 2 * Rm3 * (-sin(theta) * sin(psi) *
sin(betaMI) * cos(betaFI) * R3 - sin(theta) * cos(psi) * cos(betaFI) *
cos(betaMI) * R3 + cos(theta) * sin(psi) * sin(betaMI) * z_m +
cos(theta) * cos(psi) * cos(betaMI) * z_m + sin(theta) * sin(psi) *
sin(betaMI) * x_m + sin(theta) * cos(psi) * cos(betaMI) * x_m) -2 *
Rm3 * (cos(theta) * sin(psi) * cos(betaFI) * cos(betaMI) * R3 -
cos(theta) * cos(psi) * sin(betaMI) * cos(betaFI) * R3 - cos(theta) *
sin(psi) * cos(betaMI) * x_m + cos(theta) * cos(psi) * sin(betaMI) *
x_m + sin(theta) * sin(psi) * cos(betaMI) * z_m - sin(theta) *
cos(psi) * sin(betaMI) * z_m + sin(psi) * sin(betaMI) * sin(betaFI) *
R3 + cos(psi) * cos(betaMI) * sin(betaFI) * R3);
    2 * ds - 2 * x_m -2 * z_m 0 0
    ];
end
```

E. MATHSCRIPT NODE

```
%Fixed platform dimensions
betaFD=betaFD_gra*(pi/180);
betaFI=betaFI_gra*(pi/180);

%Mobile platform dimensions
betaMD=betaMD_gra*(pi/180);
betaMI=betaMI_gra*(pi/180);

%Solve the position of the GC of the platform
[solx_m,solz_m,soltheta,solpsi] =
CinDirectaPos3UPS_RPU(q13,q23,q33,q42,R1,R2,R3,ds,betaFD,betaFI,Rm1,Rm
2,Rm3,betaMD,betaMI,x_m_ant,z_m_ant,theta_ant,psi_ant);

%Calculate the increments caused by theta
[Ax_theta_a,Az_a]=Increment(soltheta+pi,Rm1);
[Ax_theta_b,Az_b]=Increment(soltheta,Rm2);
[Ax_theta_c,Az_c]=Increment(soltheta,Rm3);

%Calculate the increments caused by psi
[Ax_psi_a,Ay_a]=Increment(solpsi+pi,Rm1);
[Ax_psi_b,Ay_b]=Increment(solpsi+betaMD,Rm2)
[Ax_psi_c,Ay_c]=Increment(solpsi+2*pi-betaMI,Rm3);

%Calculate the global position of the GC
xm=x0+solx_m;
ym=y0;
zm=z0+solz_m;

%Calculate the global position of A
xa=xm+Ax_theta_a+Ax_psi_a;
ya=ym+Ay_a;
za=zm+Az_a;

%Calculate the global position of B
xb=xm+Ax_theta_b+Ax_psi_b;
yb=ym+Ay_b;
zb=zm+Az_b;

%Calculate the global position of C
xc=xm+Ax_theta_c+Ax_psi_c;
yc=ym+Ay_c;
zc=zm+Az_c;
```