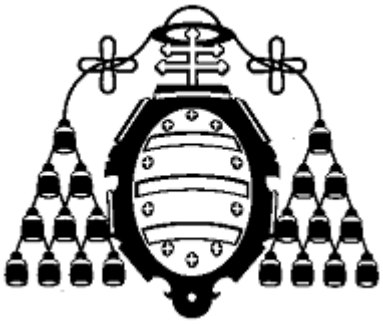

**UNIVERSIDAD DE OVIEDO**

**CENTRO INTERNACIONAL DE POSTGRADO**

# MASTER EN INGENIERÍA MECATRÓNICA

**TRABAJO FIN DE MÁSTER**

## Sistema robotizado para el transporte de materiales en los que es necesario el control de la orientación

**Julio 2015**

**Alfonso Lago Rodríguez** **Juan Carlos Álvarez Álvarez**

## RESUMEN

El presente proyecto trata de llevar a cabo el control de la orientación de la herramienta de un robot ABB IRB120 utilizando para ello, una unidad de medida inercial (IMU). Este IMU es el encargado de proporcionar los datos de orientación obtenidos simplemente por el movimiento rotacional de dicho aparato. Una de las aplicaciones más directas es la de transporte de material en el que sea necesario mantener una orientación determinada, como por ejemplo, el robot camarero.

En primer lugar, fue necesario un periodo de aprendizaje que permitiese adquirir los conocimientos necesarios para el manejo del robot. Una vez adquiridos estos conceptos, se lleva a cabo la puesta en marcha del sistema y se procede a implementar el código que permita el movimiento del manipulador. A continuación se establece un protocolo de comunicación entre el controlador del robot y el software empleado para elaborar la aplicación (Matlab). Finalmente, se desarrolla el código de captura de los datos de orientación. El último paso del proyecto se centra en el testeado del sistema y en la evaluación de los resultados obtenidos.

**PALABRAS CLAVE**

Comunicación OPC- Xsens- Controlador IRC5- IRB120- Ángulos de Euler

# ÍNDICE GENERAL

## ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# 1. INTRODUCCIÓN

En este capítulo se tratará, de forma general, el trabajo llevado a cabo a modo de introducción de la memoria descriptiva. Así, se tratarán, la propuesta de proyecto, su repercusión, y finalmente, una breve estructuración de la memoria.

## *1.1. Objeto*

El objetivo del proyecto es la elaboración de un sistema que permita el control de la orientación en 3D, en tiempo real, de un manipulador ABB IRB 120 .Se trata de programar un robot industrial para el transporte de productos en los que es importante en control de su orientación. Se pretende implementar un control en cadena cerrada, en base a la información sobre la orientación proporcionada en tiempo real por un sensor inercial tipo IMU de la marca Xsens.

## *1.2. Alcance*

El presente proyecto abarcará todos los puntos relacionados con la puesta en marcha, integración de todos los elementos de los que dispone el sistema e implementación del control para el desarrollo de un sistema robotizado que permita la orientación 3D de la herramienta del manipulador.

El manipulador contará con una herramienta que permita el posicionamiento del sensor inercial tipo IMU. Deberá disponer de un diseño flexible, que permita la colocación de otro tipo de sensores, así como en diferentes situaciones de la misma. El peso de esta herramienta no deberá sobrepasar los 3 Kg.

El material a utilizar será el aluminio. Todas las piezas serán fabricadas por el departamento de fabricación, utilizando las máquinas-herramienta disponibles.

La programación del manipulador se llevará a cabo en el propio lenguaje programación de ABB para este controlador, este recibe el nombre de lenguaje RAPID. Para la implementación del código encargado de la captura de los datos necesarios para determinar la orientación de la herramienta, se utilizará el software Matlab. Será necesario establecer un tipo de comunicación de cara a la lectura o escritura de datos de forma remota a través de internet.

## *1.3. Antecedentes*

La evolución de los robots industriales desde sus orígenes ha sido vertiginosa. La investigación y desarrollo sobre robótica industrial ha permitido que los robots tomen posiciones en casi todas las áreas productivas y tipos de industria. Principalmente están destinados a la realización de tareas repetitivas y laboriosas definidas de antemano, sin embargo, no aportan la flexibilidad y autonomía necesarias para adaptarse al cambio, al menos que sean reprogramados.

Por ello, con el fin de obtener la flexibilidad requerida, es necesario que exista una completa interacción Hombre-Robot dentro de las cadenas productivas.

En el presente proyecto, se pretende definir una orientación a partir de los datos recibidos de un sensor inercial. Se consigue con esto, cerrar el lazo de control, permitiendo así una adaptación del manipulador frente a la situación con la que se encuentre.

Esto permitiría el transporte de diversos materiales en el que es necesario el control de la orientación

Un claro ejemplo es el robot camarero, el cual debe mantener su herramienta (bandeja), en una orientación determinada, sea cual sea su posición, evitando así derramar el contenido de la bandeja

*Figura 1.1: Robot camarero.*

## *1.4. Requisitos del sistema*

Con el fin de acotar el proyecto desarrollado, se ha elaborado una serie de requisitos mínimos que el prototipo elaborado debe reunir. A continuación se detallará la lista de exigencias previamente definida:

| LISTA DE EXIGENCIAS | | |
|---|---|---|
| **Nº** | **Exigencia/Deseo** | **Descripción** |
| **SOFTWARE** | | |
| 1 | E | Sencilla estructuración del programa |
| 2 | D | Bajo tiempo de ejecución |
| 3 | E | Programación modular en Rapid. |
| 4 | E | Empleo del software Matlab y RobotStudio |
| 5 | E | Seleccionar y establecer comunicación remota más adecuada |
| CINEMÁTICAS | | |
| 1 | E | Control de la orientación en tres dimensiones |
| 2 | E | Rápida velocidad de ejecución del movimiento |
| DISEÑO, FABRICACIÓN, MATERIALES | | |
| 1 | E | Herramienta del manipulador de rápido montaje |
| 2 | E | Peso máximo de 3kg(ligera) |
| 3 | E | Herramienta flexible que permita variar el posicionamiento del sensor |

*Tabla 1.1: Requisitos del sistema.*

## *1.5. Organización del proyecto*

Para un correcto transcurso durante el desarrollo del proyecto se ha elaborado un esquema de planificación del mismo. Para ello, se ha dividido la ejecución del proyecto en varias tareas principales, que a su vez se subdividen en otras tareas de menor índole, no por ello menos importantes.

Una vez establecida la división de tareas, se procede a la implementación del diagrama Gantt mediante el software *GanttProject*. Se ha establecido unos tiempos de ejecución que en un principio parecen los más adecuados para la realización de dichas tareas. Esta planificación ha sido modificada a lo largo del proyecto, adecuándolo al desarrollo real del proyecto.
La planificación inicial se puede consultar en el anexo pertinente (Anexo 2).

## *1.6. Estructura de la memoria*

Para de facilitar la comprensión de la memoria, esta se ha estructurado del siguiente modo:

- Introducción: donde se explica de forma general el objeto y alcance del proyecto. Se establecen los requisitos a seguir para llevar a cabo el proyecto. Por otro lado, se determina la organización del mismo.

- Descripción del sistema: Durante este apartado se realizará una descripción de todos los elementos con los que cuenta el sistema, tanto de hardware como de software, incluyendo además el diseño de la herramienta del manipulador para la representación de la orientación.

- Protocolo para el calibrado de sensores: donde se establece un protocolo que permite la calibración de sensores inerciales tipo IMU.

- Software de control: en este apartado se explica el diseño del software desarrollado para conseguir el control del sistema.

- Resultados y Conclusiones.

# 2. DESCRIPCIÓN DEL SISTEMA

En el presente capitulo se ilustrará detalladamente todos los elementos por los que está formado el sistema, tanto a nivel de hardware como de software, así como la herramienta implementada para el manipulador IRB120.

## *2.1. Descripción general*

La función principal del sistema, es la generación de una determinada orientación de la herramienta del manipulador ABB IRB120. Para ello, se establece un sistema de control en lazo cerrado, en este caso manual, permitiéndonos regular su comportamiento con el fin de reducir las probabilidades de fallo y obtener los resultados deseados.

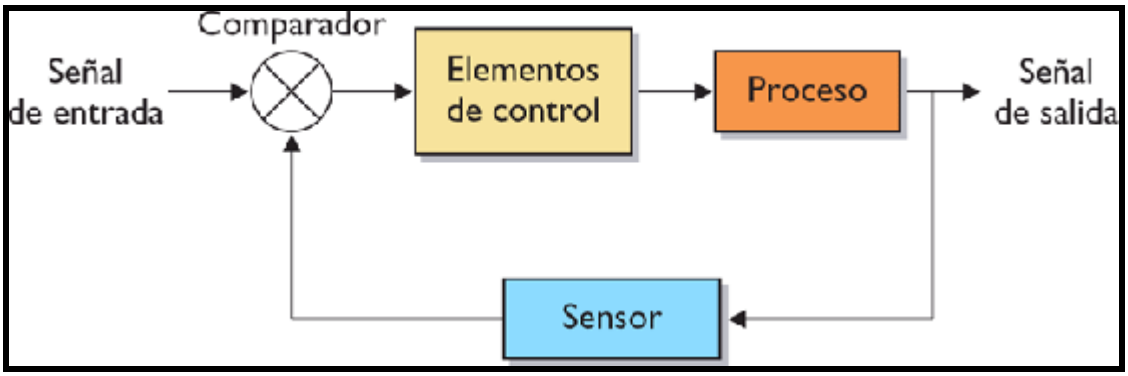


*Figura 2.1: Esquema de control en lazo cerrado.*

Los elementos por los que está formado el lazo de control son:

- Sistema a controlar o proceso: Manipulador.
- Elementos de control: Controlador IRC5.
- Medidor o sensor: Sensor inercial tipo IMU.

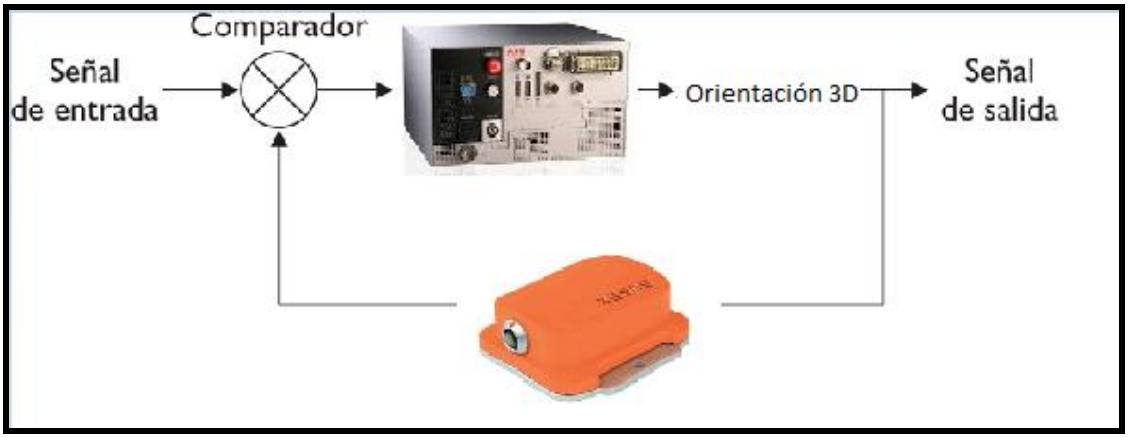


*Figura 2.2: Esquema de control en lazo cerrado. Elementos.*

Mediante el Software *Matlab*, instalado en un ordenador ajeno al robot, se captura las señales generadas por un sensor inercial de la marca Xsens, en este caso, estas señales serán obtenidas en forma de ángulos de Euler (Roll, Pitch, Yaw). Será el controlador IRC5 el que reciba estos datos e implemente el movimiento pertinente en el manipulador.

Para la conectividad remota que permite tanto la escritura como la lectura de variables en el controlador, se ha instalado en el mismo ordenador donde se capturan las señales, tanto el *OPC client* como el *OPC Server ABB* que es quien solicita los datos al robot a través de internet. A continuación se muestra un esquema de la comunicación remota así como del conexionado total del sistema:

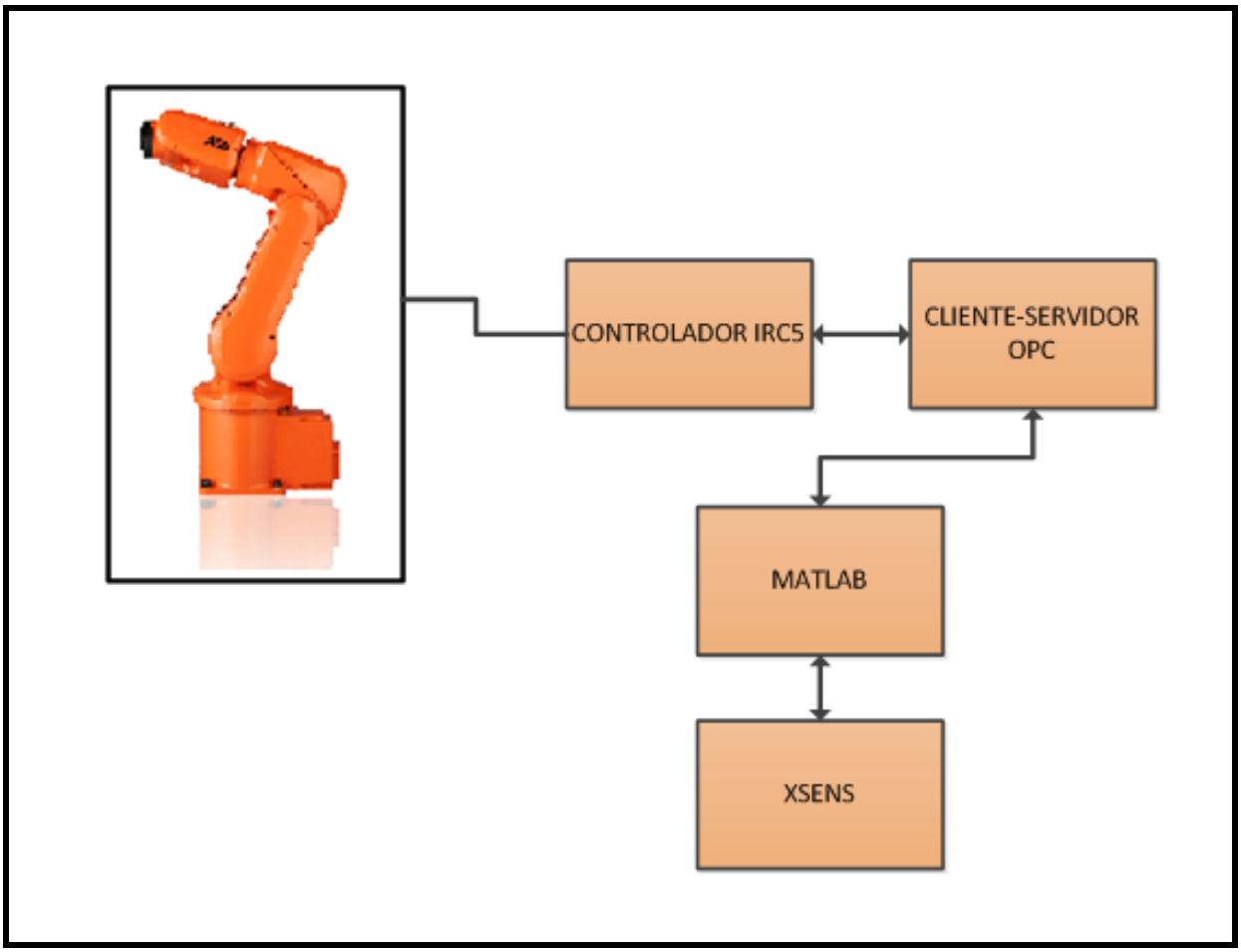


*Figura 2.3: Diagrama de bloques del sistema*

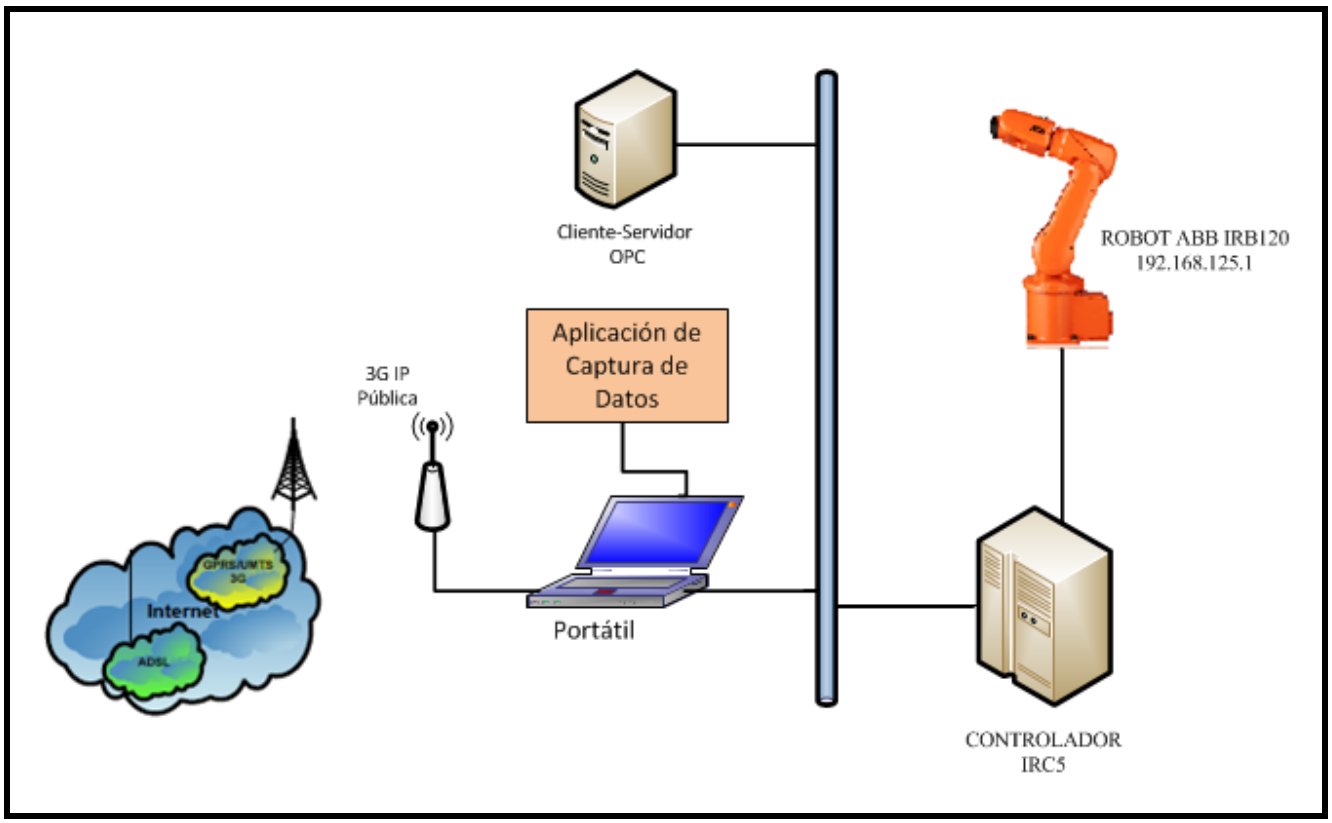


*Figura 2.4: Esquema de conexión remota.*

## 2.2. Descripción del hardware utilizado

### 2.2.1. CONTROLADOR IRC5

El controlador utilizado en el presente proyecto, es una variante del controlador IRC5 denominado *IRC5 compact.* Este es un controlador de robot de escritorio diseñado para segmentos como por ejemplo el mercado 3C, está pensado para ser montado en un cuadro eléctrico, por ello todos sus elementos de accionamiento y de conexión están instalados en la parte frontal del mismo. El grado de protección del controlador compacto es la clase IP20, de acuerdo con la norma IEC60529. Dicho armario de control contiene los elementos electrónicos necesarios para controlar el manipulador, los ejes adicionales y equipos periféricos.

El controlador IRC5 se compone de los módulos siguientes:

- Módulo de accionamiento, que contiene el sistema de accionamiento
- Control Module, que contiene el ordenador principal (incluidas cuatro ranuras PCI para tarjetas de extensión), el panel del operador, el interruptor principal, las interfaces de comunicación, la conexión para FlexPendant, los puertos de servicio y cierto espacio para los equipos del cliente, por ejemplo tarjetas de E/S de ABB. El controlador también contiene el software de sistema, es decir RobotWare-OS, que incluye todas las funciones

básicas de manejo y programación. Sobre RobotWare-OS es posible instalar un número de opciones con funcionalidad adicional.

A continuación analizaremos los principales elementos utilizados en la aplicación con los que cuenta el controlador en su frontal.

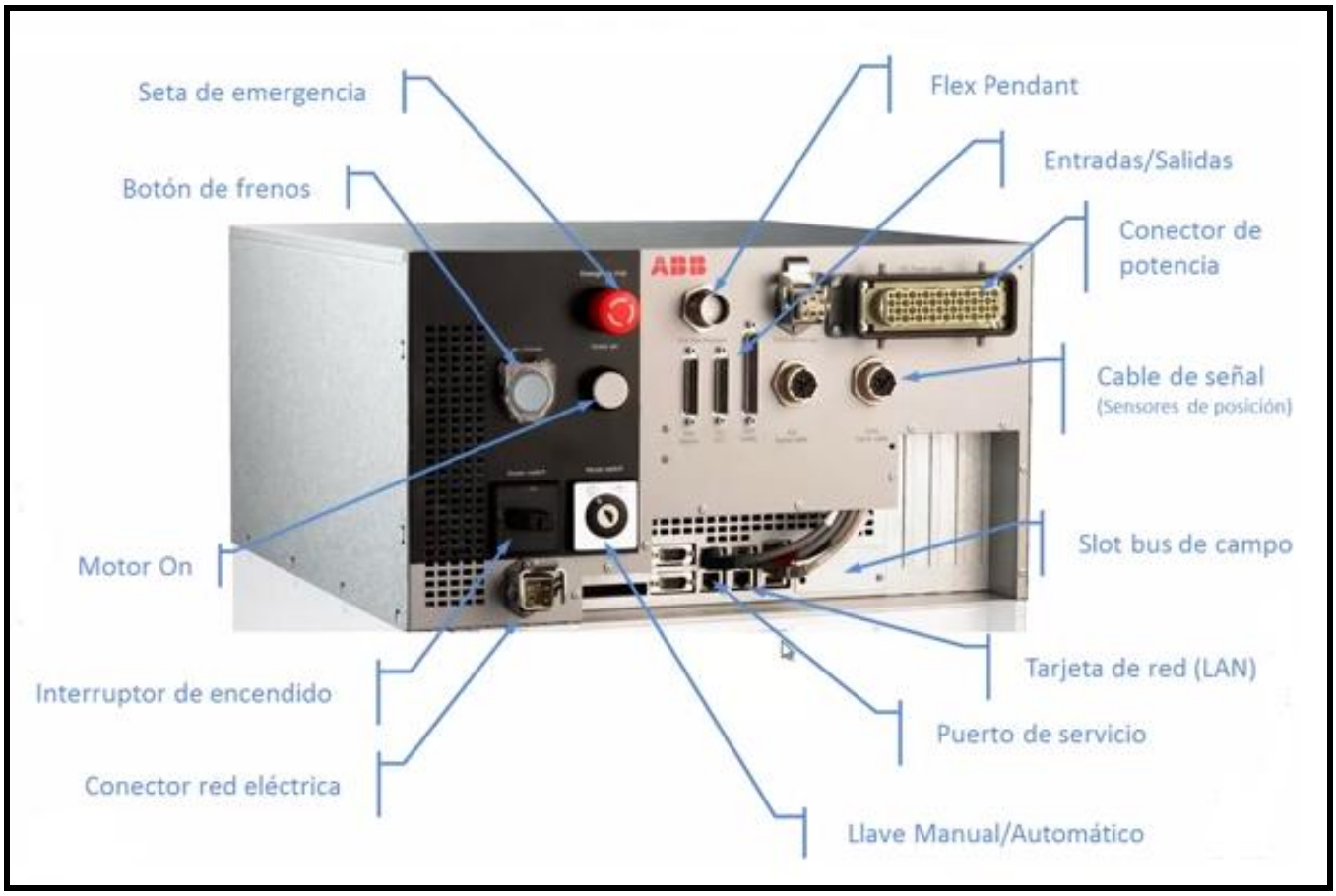


*Figura 2.5: Controlador IRC5.*

- Conector red eléctrica: Se trata de un conector monofásico a 230V.
- Interruptor de encendido: Interruptor principal y control remoto de la alimentación de los módulos de accionamiento.
- Seta de emergencia: Pulsador de emergencia que permite detener el movimiento del robot en cualquier momento.
- Motors ON: Este pulsador sirve para activar los motores.
- Botón de frenos: Permite el bloqueo o desbloqueo de los motores.
- Selector de modo de funcionamiento: Dispone de dos posiciones permitiendo seleccionar modo manual o automático.

- Puerto de servicio: Conector RJ45 con el que se realiza la puesta en servicio, sirve para conectar el ordenador y configurar el equipo desde RobotStudio. El cable a utilizar debe ser un cable de red de tipo cruzado.
- Conexión de FlexPendant: Se trata de un conector circular que sirve para conectar la consola FlexPendant.
- Conector de Potencia: conector que permite la alimentación de los elementos de potencia, es decir, los motores.
- Conector de señal (sensores de posición): Envía información de los sensores de posición a la controladora.

Este controlador cuenta con una consola de programación portátil que recibe el nombre de FlexPendant. A continuación se exponen los elementos de los que consta esta consola:


*Figura 2.6: Consola de programación portátil o FlexPendant.*

- Pantalla táctil: Permite navegar por el entorno del sistema pudiendo tener varias ventanas abiertas al mismo tiempo.
- Seta de emergencia: Como hemos visto en el controlador, el FlexPendant también dispone de una seta de emergencia.
- Joystick: Es un elemento de mando que permite el posicionamiento del manipulador.
- Teclas de ejecución: Permiten iniciar y parar el programa.
- Dead man button: La función de este pulsador es no permitir el movimiento a menos que este sea activado, evitando así, posibles movimientos inesperados al tropezar con el joystick.

### 2.2.2. *MANIPULADOR ABB IRB120*

El IRB 120 es miembro de la generación más reciente de ABB Robotics de robots industriales de 6 articulaciones, todas ellas de tipo rotativo, que le confieren un total de 6 grados de libertad de movimiento en su efector final, con una carga útil de 3 kg y diseñado específicamente para industrias de fabricación que utilizan una automatización flexible basada en robot y para industrias 3C. Tiene un peso de 25 Kg. La repetitividad que estos robots pueden llegar a ofrecer es de 0.01 mm. El robot tiene una estructura abierta especialmente adaptada para un uso flexible y presenta unas grandes posibilidades de comunicación con sistemas externos.

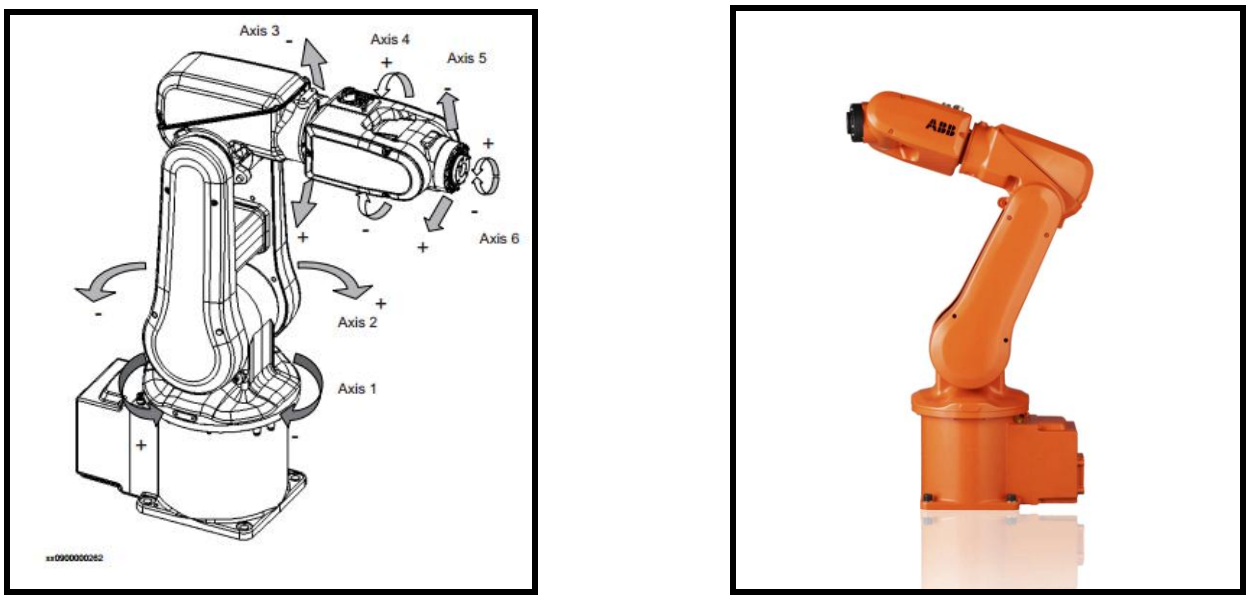


*Figura 2.7: Manipulador ABB IRB120.*

Las pequeñas dimensiones y el poco peso con el que cuenta, permite una colocación del mismo muy diversa.

*Figura 2.8: Posicionado del manipulador.*

El manipulador puede realizar movimientos con un alcance horizontal máximo de 580 mm y con una velocidad lineal de movimiento de la punta de la herramienta del orden de 6000 mm/s. En la Figura 2.9 se detallan las áreas de trabajo del manipulador donde las posiciones extremas del brazo de manipulador se especifican respecto al centro de la muñeca.

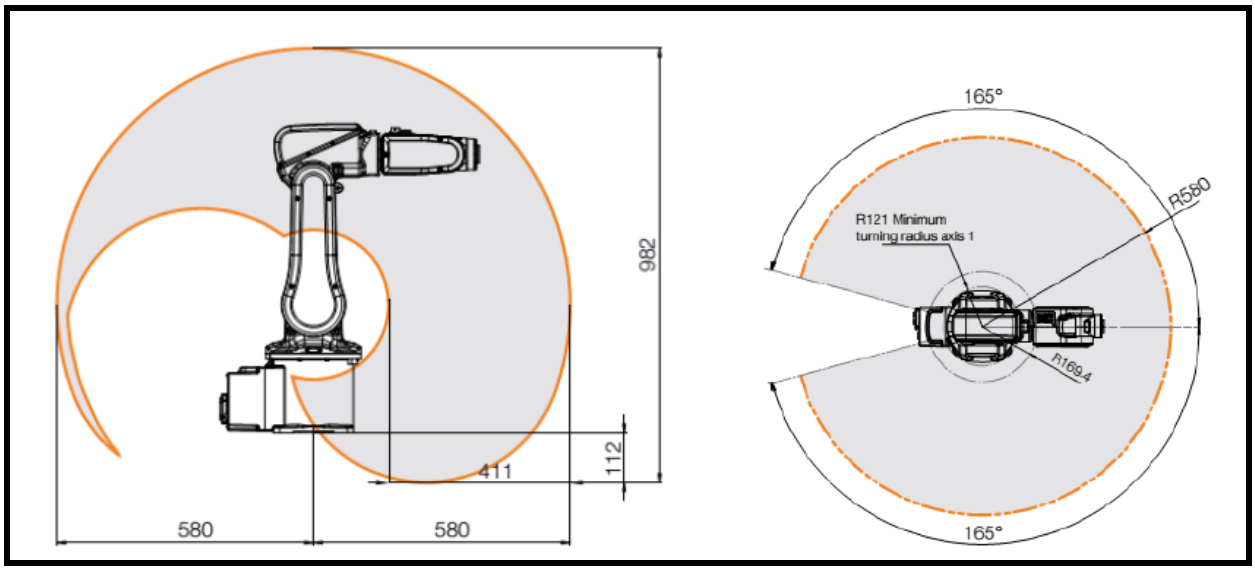


*Figura 2.9: Área de trabajo del manipulador.*

### 2.2.3. *SENSOR IMU MTI XSENS*

El MTi (Motion Tracker) es un sensor de medición inercial con magnetómetros 3D integrados, con un procesador integrado capaz de calcular 'Roll', 'Pitch' y 'Yaw' en tiempo real, así como, la salida calibrada 3D de aceleración lineal, velocidad angular (giroscopio) y los datos del campo magnético.

*Figura 2.10: Unidad de medida inercial (IMU) de Xsens.*

En la siguiente figura se muestran la arquitectura general del sistema, es decir, sus módulos y conexiones internas.

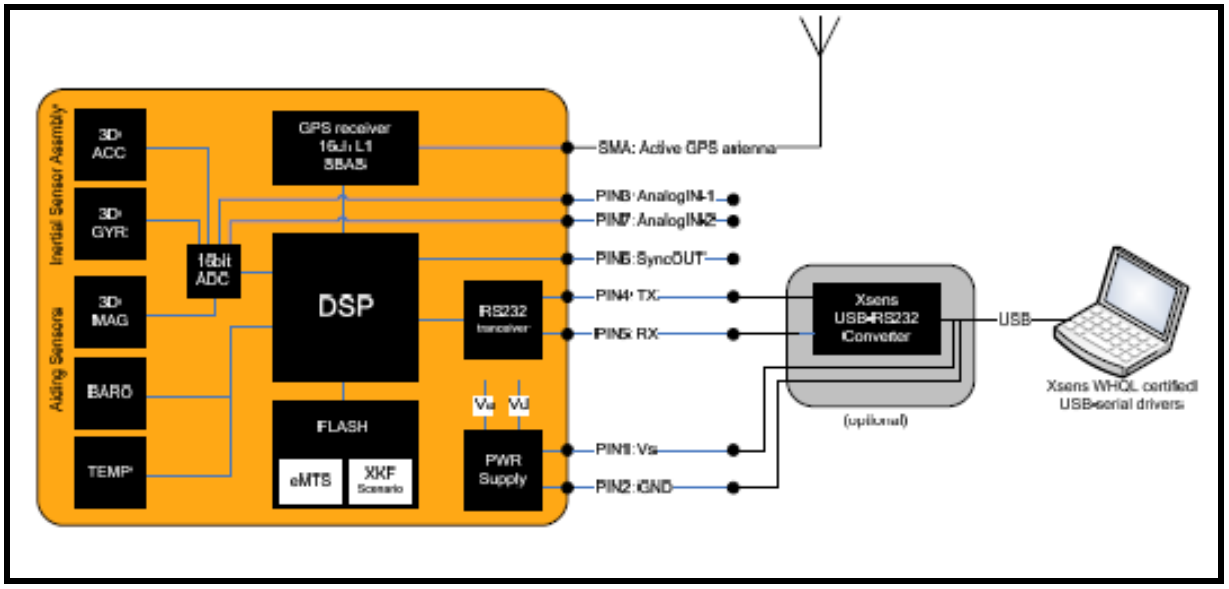


*Figura 2.11: Arquitectura interna del IMU.*

La orientación del MTi se calcula por el filtro de Kalman de 3 grados de libertad (XKF-3 por sus siglas en ingles). XKF-3 utiliza las señales de los giroscopios, acelerómetros y magnetómetros para calcular una estimación estadística óptima de la orientación en 3D en alta precisión sin la interferencia de los desvíos tanto estáticos como dinámicos. El diseño del algoritmo XKF-3 puede ser explicado como un algoritmo de fusión sensorial donde la medición de la gravedad (por los acelerómetros 3D) y el norte magnético de la Tierra (por los magnetómetros 3D) se compensan mutuamente lentamente, pero sin límites, incrementando (por deriva) errores de la integración de datos de tasa de giro.

Con el fin de estabilizar la medida de la inclinación el filtro de Kalman se sirve de la aceleración de la gravedad. Un acelerómetro mide la aceleración de la gravedad, más la aceleración debida al movimiento del objeto con respecto a su entorno.

El filtro de Kalman de 3 grados de libertad utiliza la hipótesis de que el promedio de la aceleración debida al movimiento es cero. Suponiendo esta hipótesis, la dirección de la gravedad puede ser observada y utilizada para estabilizar la medida de la posición. La clave aquí es la cantidad de tiempo durante la cual debe ser la aceleración promedio para que la hipótesis sea cierta. Durante este tiempo, los giroscopios deben ser capaces de seguir la orientación con un alto grado de precisión. En la práctica, esto limita la cantidad de tiempo durante el cual el supuesto es cierto. Para los giroscopios utilizados en el MTi, este periodo de tiempo es de aproximadamente 10-20 segundos como máximo.

Por otro lado, para estabilizar el *'heading'* (Yaw) se hace uso del campo magnético terrestre. Si el campo magnético de la Tierra local se encuentra temporalmente perturbado, XKF-3 hará un seguimiento de esta alteración en vez de asumir incorrectamente que no hay perturbación. Sin embargo, en caso de perturbaciones estructurales magnéticas (> 10 a 20 segundos), el *'heading'* calculara lentamente una solución con el nuevo norte magnético local. Tenga en cuenta que el campo magnético no tiene efecto directo en la estimación de inclinación.
En el caso especial de que el MTi este rígidamente atado a un objeto que contiene materiales ferro-magnéticos, entonces las perturbaciones magnéticas estarán presentes. El uso de un *'magnetic field mapping'* hace que las perturbaciones magnéticas sean eliminadas, permitiendo que el MTi pueda ser utilizado como si no estuviera unido a un objeto que contiene materiales ferro-magnéticos.

La comunicación a desarrollar con el MTi será la comunicación serie mediante el adaptador de Xsens USB-serie.

## *2.3. Descripción del software utilizado*

### 2.3.1. *LENGUAJE RAPID*

El programa está compuesto por un conjunto de instrucciones que describen la actividad del robot. Por tanto, existen instrucciones específicas para los distintos comandos, por ejemplo una para mover el robot, otra para seleccionar una salida, etc.

Por lo general, las instrucciones llevan asociado un conjunto de argumentos que definen qué debe ocurrir con una instrucción concreta. Por ejemplo, la instrucción utilizada para restablecer una salida contiene un argumento que define qué salida debe restablecerse, por ejemplo Reset do5.

Existen tres tipos de rutinas: procedimientos, funciones y rutinas TRAP.

- Los procedimientos se utilizan como subprogramas.
- Las funciones devuelven un valor de un tipo concreto y se utilizan como argumento de una instrucción.
- Las rutinas TRAP proporcionan una forma de responder a las interrupciones. Las rutinas TRAP pueden asociarse con una interrupción determinada. Por ejemplo, al establecer una entrada, se ejecuta automáticamente si se produce dicha interrupción en concreto.

La información también puede almacenarse en datos, por ejemplo los datos de las herramientas (que contienen información sobre las herramientas, como su TCP y su peso) y datos numéricos (que pueden usarse por ejemplo para contar el número de piezas que deben procesarse). Los datos se agrupan en distintos tipos de datos que describen los distintos tipos de información, como herramientas, posiciones y cargas. Dado que es posible crear estos datos y asignarles nombres arbitrarios, no existe ningún límite en el número de datos (excepto el límite impuesto por la memoria disponible). Estos datos pueden existir de forma global en el programa o sólo localmente dentro de una rutina.

Existen tres tipos de datos: constantes, variables y persistentes

- Las constantes representan valores fijos y la única forma de asignarles un nuevo valor es manualmente.
- La asignación de nuevos valores a las variables puede realizarse durante la ejecución del programa.
- Un valor persistente puede describirse como una variable “persistente”. Cuando se guarda un programa, el valor de inicialización corresponde al valor actual del valor persistente.

Otras características del lenguaje son:

- Parámetros de rutinas
- Expresiones aritméticas y lógicas
- Gestión automática de errores
- Programas modulares
- Multitasking

El programa se ejecuta secuencialmente como una regla, es decir, una instrucción tras otra. En ocasiones, se requieren instrucciones que interrumpen esta ejecución secuencial y que llaman a otra instrucción, para enfrentarse a las distintas situaciones que pueden darse durante la ejecución.

El flujo del programa puede controlarse acorde con cinco principios diferentes:

- Llamar a otra rutina (procedimiento) y, una vez ejecutada dicha rutina, continuar la ejecución con la instrucción que sigue a la llamada a la rutina.
- Ejecutar instrucciones diferentes en función de si se cumple o no una condición determinada.

- Repetir una secuencia de instrucciones un número determinado de veces o hasta que se cumple una condición determinada.
- Ir a una etiqueta dentro de la misma rutina.
- Detener la ejecución del programa.

La memoria RAM del robot, se organiza en módulos, y por defecto crea siempre dos módulos:

- Módulo de programación: Por ejemplo *MainModule.mod*: módulo principal que se utiliza para organizar el programa y desde el que se llama a las subrutinas.
- Módulo de sistema: Por ejemplo *Base.sys*: módulo donde se guardan datos imborrables como tool0 o el sistema de coordenadas mundo. Se pueden guardar datos creados por el usuario

El programador puede crear más módulos tanto de programación como de sistema creando una estructura como la que se refleja en la siguiente figura:

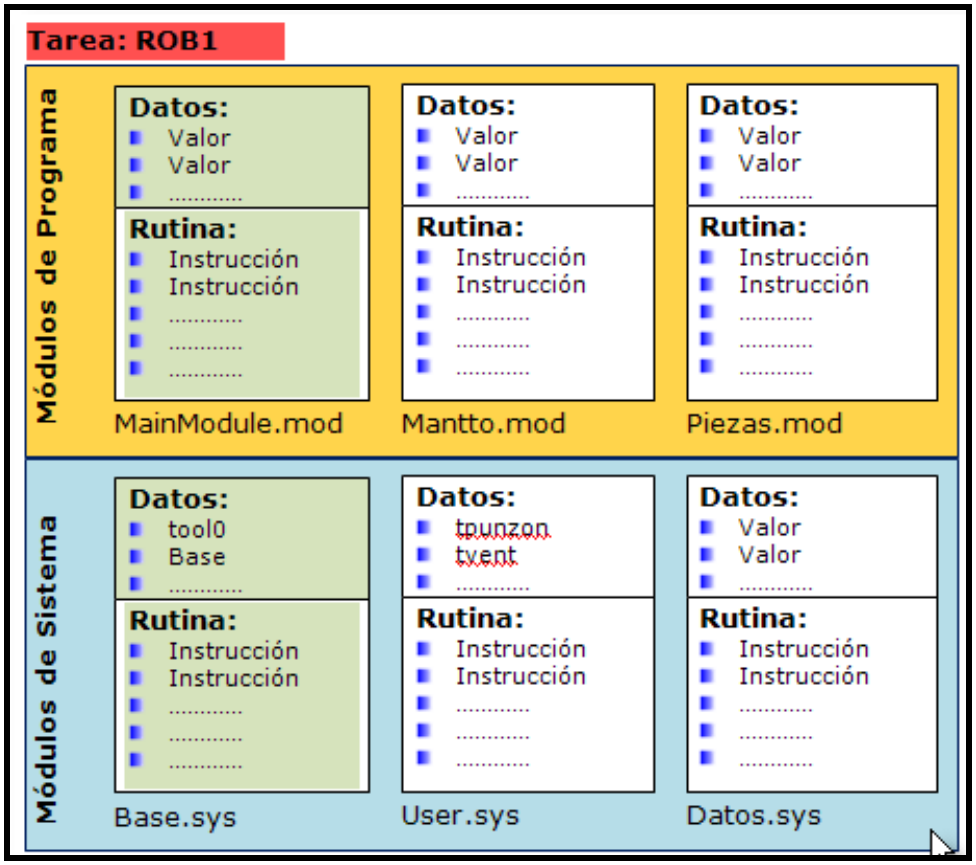


*Figura 2.12: Estructura del leguaje de programación RAPID.*

### 2.3.2. *MATLAB*

Es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OSX y GNU/Linux.

### 2.3.3. *ROBOTSTUDIO*

El software asociado a los robots ABB que permite la programación y simulación de los mismos para comprobar su comportamiento, se denomina RobotStudio.

RobotStudio es una aplicación de PC destinada al modelado, la programación fuera de línea y la simulación de células de robot. RobotStudio le permite trabajar con un controlador fuera de línea, que constituye un controlador IRC5 virtual que se ejecuta localmente en su PC. Este controlador fuera de línea también se conoce como el controlador virtual (VC). RobotStudio también le permite trabajar con un controlador IRC5 físico real, que simplemente se conoce como el controlador real. Cuando RobotStudio se utiliza con controladores reales, se conoce como el modo online. Al trabajar sin conexión a un controlador real o mientras está conectado a un controlador virtual, se dice que RobotStudio se encuentra en el modo fuera de línea.
Además, el programa consta de una vista gráfica en 3D donde se pueden crear simulaciones del entorno. También es posible la importación de archivos CAD, la generación automática y optimización de trayectorias, el análisis del alcance del manipulador a determinadas posiciones, la detección de colisiones, entre otras funcionalidades.
En la figura que se muestra a continuación, se puede observar el entorno de trabajo de RobotStudio, donde se puede diferenciar claramente el entrono de programación y el de simulación.

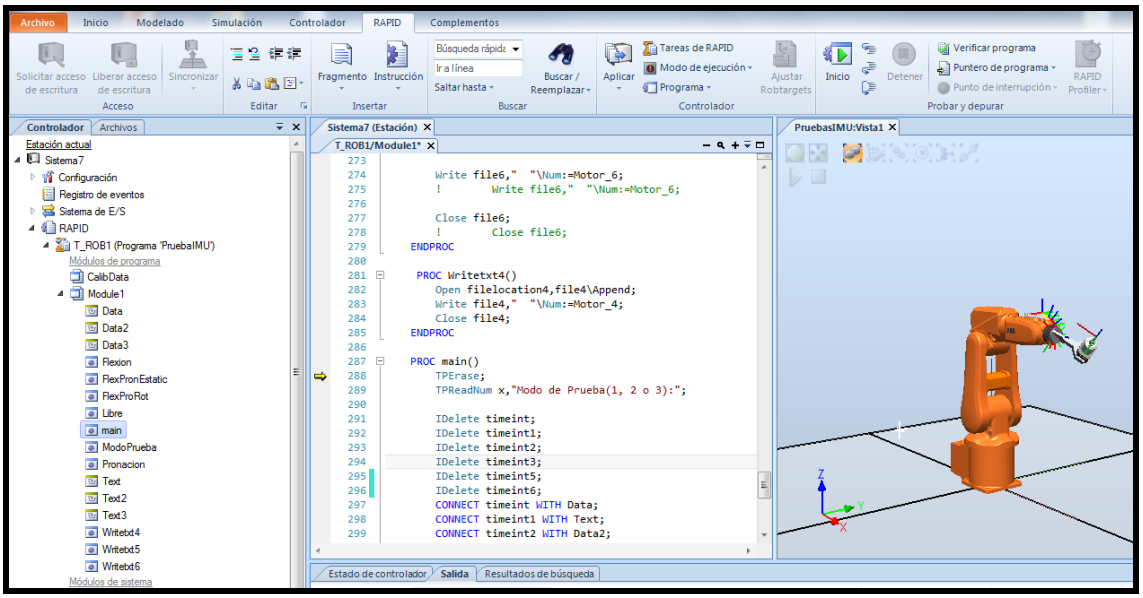


*Figura 2.15: Entorno de trabajo de RobotStudio.*

### 2.3.4. *COMUNICACIÓN REMOTA*

El sistema robot dispone de una serie de opciones de cara a la lectura o escritura de datos de forma remota a través de Internet. Estas opciones van a servir para poder manejar principalmente datos, para su posterior tratamiento tanto en temas relacionados tanto con la gestión, como en el mantenimiento, es lo que en este proyecto se denomina telegestión y telemantenimiento o lo que es lo mismo la obtención de forma remota de datos que sirvan para dichos fines.

Las posibilidades de conectividad remota a través de internet que ofrecen los robots ABB se resumen en:

- Conectividad a través de **ftp**.
- Conectividad mediante **OPC**.
- Conectividad mediante **RobotStudio**.
- Conectividad mediante **sockets**.

Tras barajar todas las posibilidades existentes, se decidió que los dos tipos de comunicación que mejor se ajustaban a nuestras necesidades eran la comunicación OPC y socket. Finalmente se selecciono el tipo de comunicación OPC. Esto se debe a que ABB proporciona un servidor OPC. Además, a diferencia de comunicación por Socket, no es preciso que la lectura y la escritura estén sincronizadas

OPC es el estándar de interoperabilidad para el intercambio seguro y fiable de datos en el espacio de la automatización industrial y en otras industrias. Es independiente de la plataforma y garantiza un flujo continuo de información entre los dispositivos de múltiples proveedores. La Fundación OPC es responsable del desarrollo y mantenimiento de esta norma.

El estándar OPC es una serie de especificaciones desarrolladas por proveedores de la industria, los usuarios finales y desarrolladores de software. Estas especificaciones definen la interfaz entre clientes y servidores, así como servidores y servidores, incluido el acceso a los datos en tiempo real, monitoreo de alarmas y eventos, acceso a datos históricos y otras aplicaciones.

Se pueden encontrar dos tipos de tecnologías OPC:

- Clásica: Se basan en la tecnología de Microsoft Windows mediante el COM / DCOM (Distributed Component Object Model) para el intercambio de datos entre los

componentes de software. Las especificaciones proporcionan definiciones distintas para acceder a los datos de proceso, alarmas y datos históricos.

- Arquitectura unificada: Es una arquitectura orientada a servicios independiente de la plataforma que integra toda la funcionalidad de las especificaciones individuales OPC clásicos en un solo marco extensible.

Sin embargo a pesar de las ventajas de una arquitectura abierta y multiplataforma, una gran parte de los fabricantes de equipos utilizan el sistema operativo Windows e implementan solo el estándar OPC clásico. Con lo que el intercambio de información entre componentes de software se realiza a través de la tecnología de Microsoft COM/DCOM (*Distributed Component Object Model*).

La imagen que aparece a continuación, muestra un ejemplo sobre un ambiente de control y monitoreo industrial basado en el concepto OPC. Los servidores OPC actúan como interfaz de comunicación entre los distintos clientes OPC; así como entre estos y los dispositivos de campo. Las vías de conexión entre los dispositivos de campo y los servidores OPC son diversas, dependiendo de la tecnología utilizada por cada fabricante.

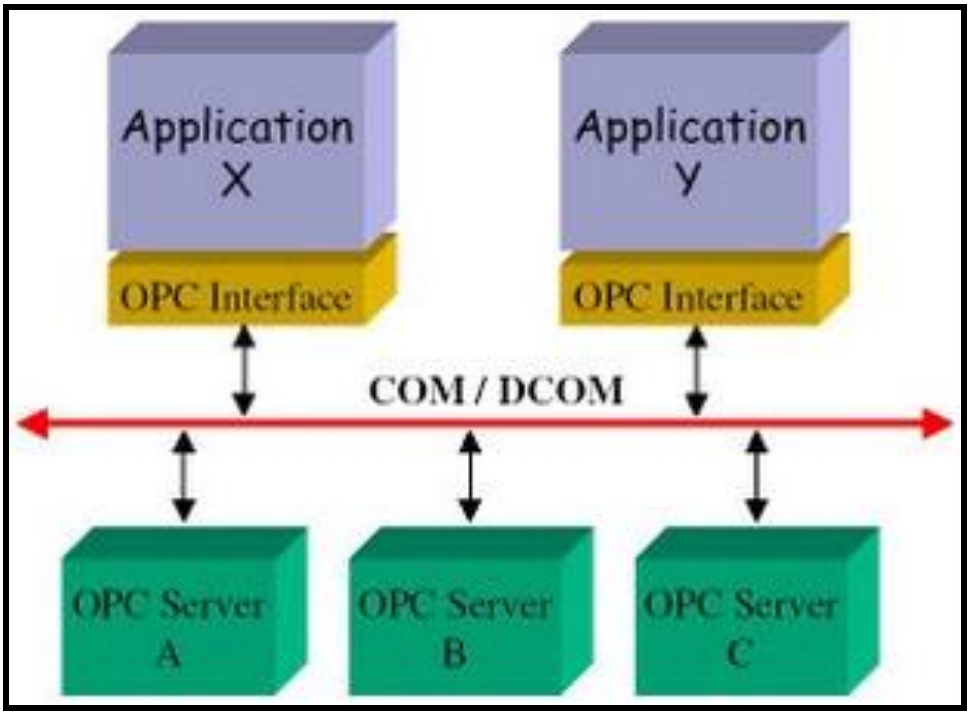


*Figura 2.16: Esquema de comunicación OPC.*

Para la implementación de la comunicación OPC fue necesaria la instalación de un servidor OPC y el empleo de la Toolbox OPC de Matlab.

### Servidor OPC ABB IRC5

El controlador IRC5 permite la comunicación vía OPC DA v3.0 y OPC AE v1.0 por medio de un servidor OPC a instalar en una PC conectada a uno o varios controladores IRC5 virtuales o reales.

La cantidad de controladores que se pueden conectar a un mismo servidor OPC está limitada por el rendimiento del sistema. ABB aconseja conectar un máximo de 30 a una PC dedicada únicamente como servidor.

El servidor OPC DA contiene variables (*tags*) predefinidas que proveen información concerniente al estado actual de los controladores conectados. Además de estas, se pueden disponer de hasta 1000 variables conectadas a señales de entradas/salidas de controladores IRC5 y hasta 200 variables que reflejen los valores de variables persistentes existentes en los programas de los controladores.

La cantidad de clientes OPC a conectarse se encuentra limitado por el rendimiento del sistema. Se permiten tiempos de *polling* del servidor entre 10-6000ms. Se calcula que normalmente un cambio en una variable será notificado al cliente entre 150-300ms aunque depende de la prioridad de las tareas que esté ejecutando el controlador y la velocidad de la conexión de red. Se recomienda una configuración de 1000ms por el fabricante. Por otro lado, este asegura que la cantidad de clientes conectados tiene muy poca influencia en el rendimiento del servidor.

El servidor OPC utiliza "*report mode*" para la publicación de las variables OPC (*OPC tags*) de manera que cuando una variable o señal cambia de valor será reportado al cliente tan pronto este la consulte en el próximo tiempo de muestreo. De esta manera los clientes pueden incluso configurar un tiempo de muestreo de 0ms, lo que significa que el servidor notificará el cambio tan pronto como sea posible.

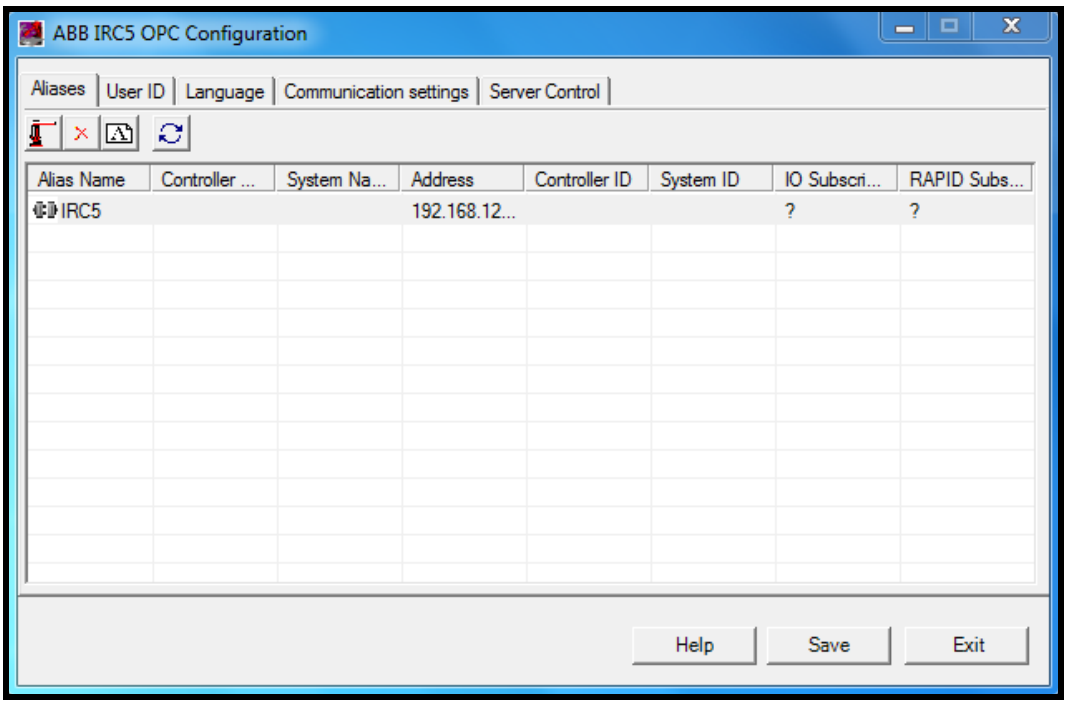


*Figura 2.17: Interfaz del servidor ABB IRC5 OPC.*

### Toolbox OPC Matlab

El software OPC Toolbox implementa un enfoque jerárquico orientado a objetos para la comunicación de Matlab con los servidores OPC utilizando los estándares Historical Data Access y OPC Data Access.

Mediante el uso de las funciones de la Toolbox, se crea un objeto OPC Data Access (DA) y Historical Data Access (HDA) que representan la conexión entre MATLAB y un servidor OPC. El uso de propiedades de los objetos cliente nos permite controlar varios aspectos de la conexión de comunicación, tales como el estado de la conexión, y el almacenamiento de eventos asociados con ese cliente.

La implementación de estas funciones se puede llevar a cabo directamente con la interfaz de OPC Toolbox o elaborando el código pertinente, nos hemos decantado por esta última opción.

En temas posteriores se detallará paso a paso la puesta en marcha del servidor así como la implementación de OPC Toolbox.

# 3. DISEÑO DE LA HERRAMIENTA Y PUESTA EN MARCHA DEL SISTEMA

En el presente capítulo se explicará detalladamente el proceso para la obtención de la herramienta del manipulador así como las nociones necesarias para poner el sistema en funcionamiento.

## *3.1. Diseño se la herramienta del manipulador*

Durante el desarrollo del proyecto fue necesario llevar a cabo el diseño y fabricación de una herramienta para el manipulador. La función principal de este elemento es la de proporcionar soporte a sensores, principalmente de tipo inercial.

### 3.1.1. *DESCRIPCIÓN DEL DISEÑO ELEGIDO*

Uno de los requisitos a tener en cuenta para el diseño de la herramienta, es que esta posea cierta flexibilidad, es decir, que permita posicionar sobre si misma sensores de diversos tipos y tamaños. Además, estos sensores deben poder posicionarse sobre los tres planos con la orientación requerida. Con el fin de cumplir estas expectativas, se opto por una geometría en forma de cruz tridimensional.

*Figura 3.1: Geometría de la herramienta.*

Esta herramienta soporte deberá contar con un sistema de sujeción que permita fijarla a la brida del robot, además dispondrá de un soporte donde ira ubicado el sensor a utilizar.

### 3.1.2. *MODELADO DEL DISEÑO ELEGIDO*

El modelado 3D se lleva a cabo empleando el software de diseño SolidWorks. Mediante el modelado 3D se ha llevado a cabo un proceso iterativo de diseño, ya que se ha ido adaptando con el fin de obtener el sistema más adecuado, es decir un diseño simple y de fácil fabricación, teniendo en cuenta, además, las limitaciones espaciales requeridas.

Tras varias modificaciones el resultado final es el que se muestra a continuación:

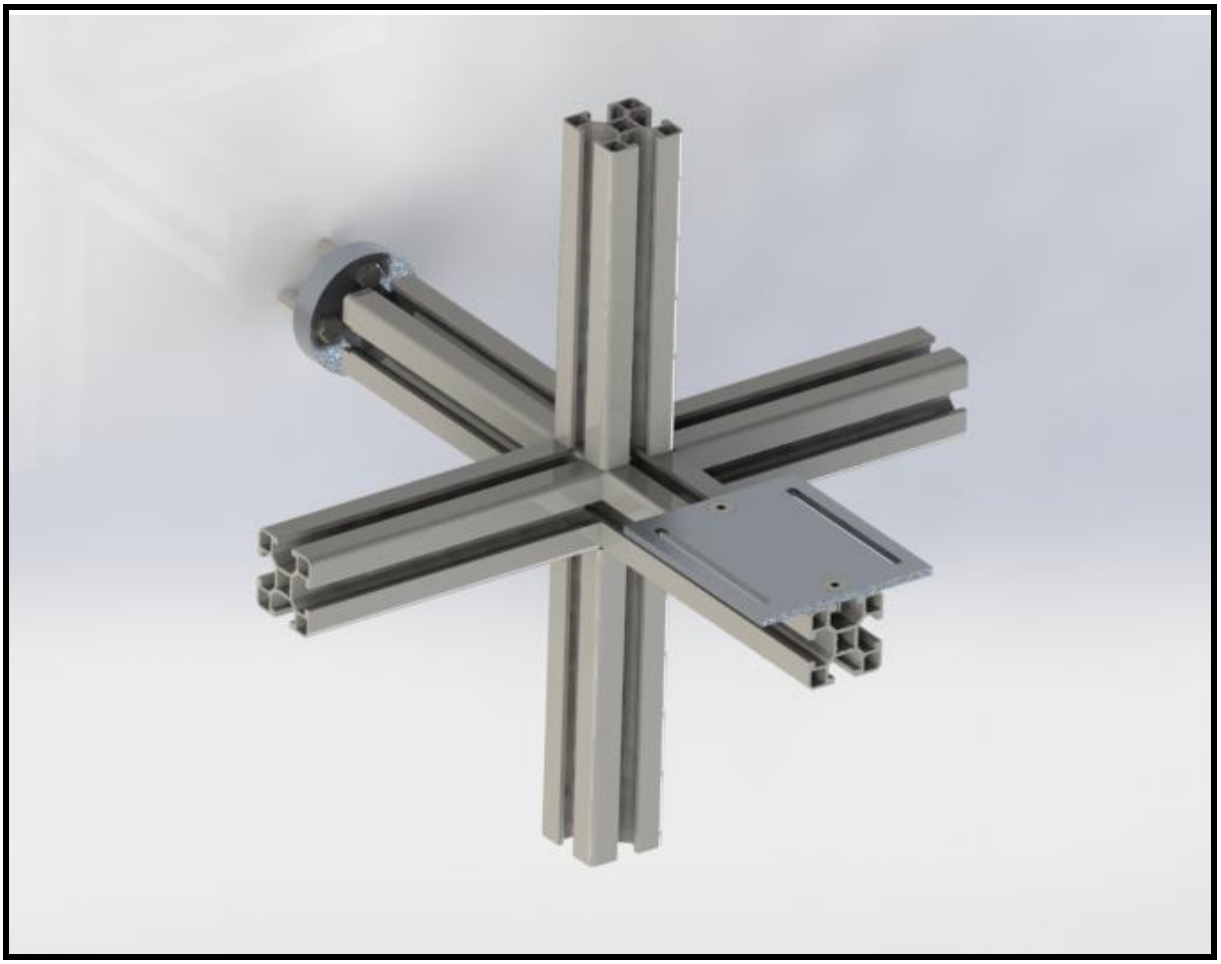

*Figura 3.2: Modelado 3D de la herramienta.*

A continuación se muestra cada uno de los elementos que conforman el modelo.

- **Vía principal**

Es el elemento en el que se apoyan las demás vías. Es un perfil modular de 30x30 mm con una longitud de 270 mm.

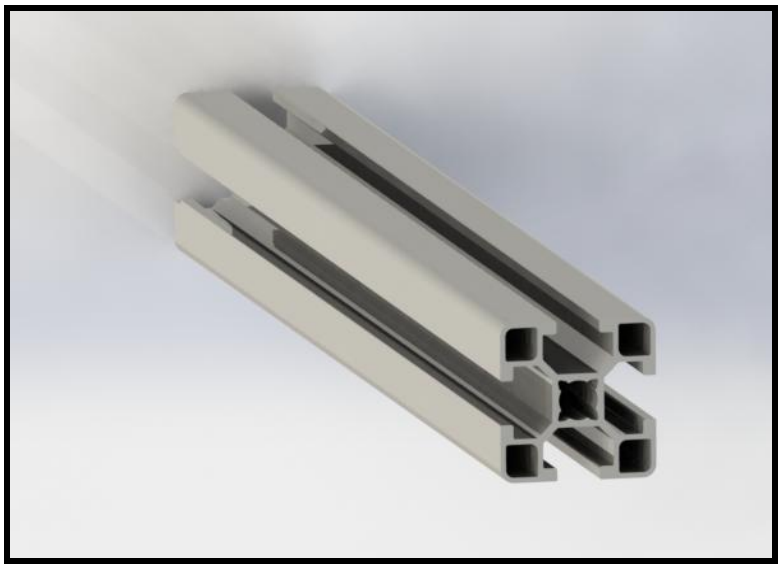

*Figura 3.3: Modelo de la vía principal.*

- **Vía Lateral**

Estas vías van fijadas a la vía principal por medio de uniones cruzadas. Al igual que la vía principal, es un perfil de 30x30 mm con una longitud de 120 mm.

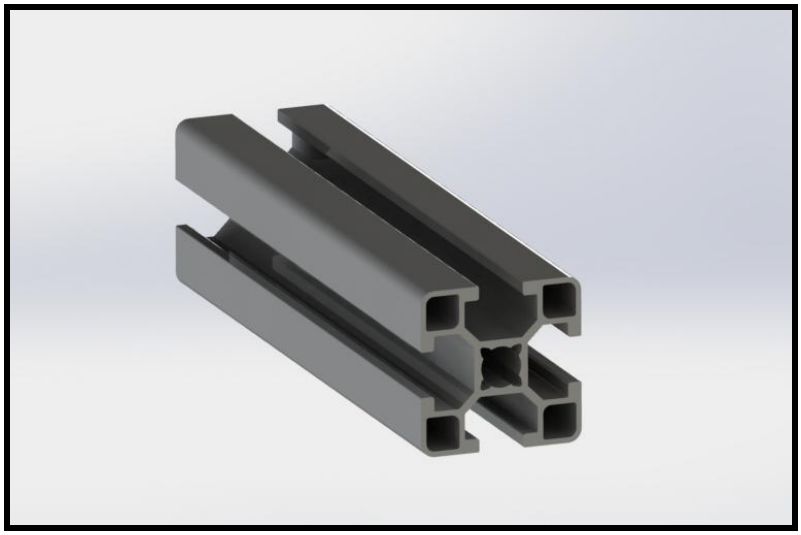

*Figura 3.4: Modelo de la vía lateral.*

- **Soporte Sensor**

Es el elemento donde se posicionará el sensor a usar. A su vez, este soporte puede ser colocado en cualquiera de las seis vías. Dispone de un par de ranuras que permiten la fijación entre el sensor y el soporte.

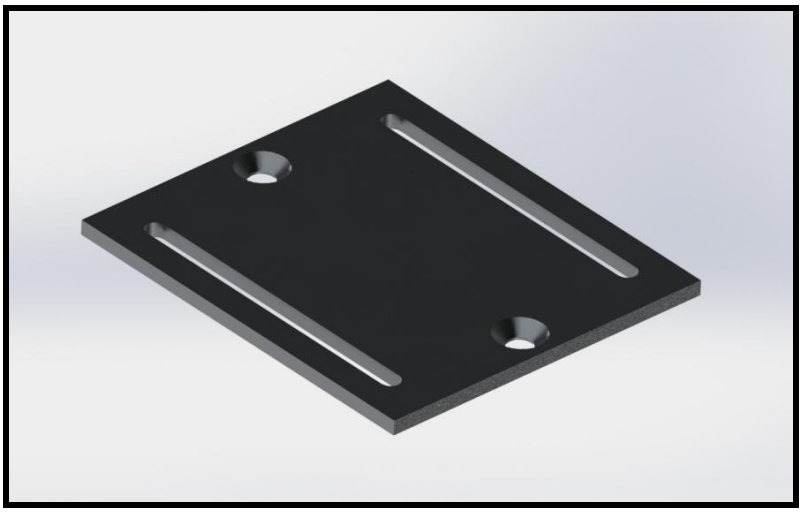

*Figura 3.5: Modelo del soporte del sensor inercial.*

- **Brida**

La función principal de este elemento, es la unión de la herramienta con el robot. La geometría de esta pieza queda condicionada por la brida del la muñeca del manipulador.

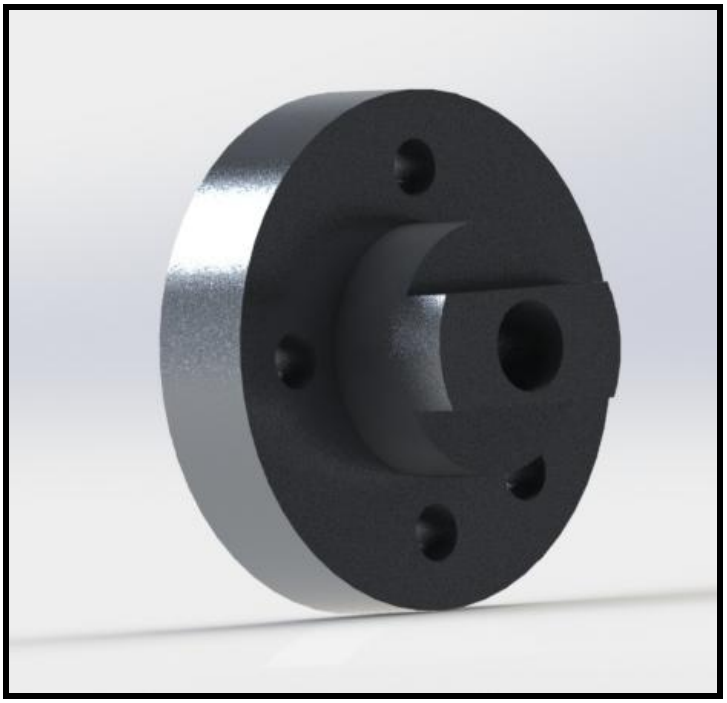

*Figura 3.6: Modelo de la brida.*

### 3.1.3. *MATERIALES Y PROCESOS DE FABRICACIÓN*

El material elegido para la mayor parte de las piezas ha sido aluminio debido a la relación ligereza-resistencia y facilidad de mecanizado . Las piezas realizadas en éste material son:

- Brida
- Soporte Sensor
- Vía Principal y Lateral.

La Brida se realizó por torneado, y posteriormente se mecanizaron los agujeros para fijarla a la muñeca del robot.

El Soporte del Sensor y el acondicionamiento de las vías obtenidas del perfil modular se realizador en primer lugar por corte, seguidas de un proceso de fresado ya que se requiere tolerancia para que ensamblen correctamente, así como el soporte de las ruedas locas

### 3.1.4. *ELEMENTOS COMERCIALES*

Los elementos comerciales empleados para la fabricación de la herramienta han sido:

- **Tapa Perfil 30x30**

Se han utilizado un total de cinco tapas, una por cada vía de la herramienta, excepto la vía que va en contacto con la brida del robot. Estas tapas han sido seleccionadas del catálogo de la empresa MINITEC, cuya referencia de artículo se corresponde con N° 22.1146/1

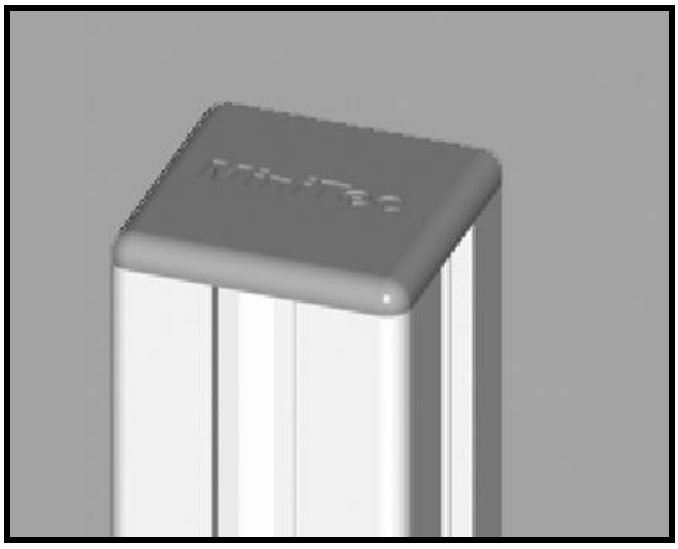

*Figura 3.7: Tapa MiniTec 30x30.*

- **Fijación Perfil 30**

Para la unión de todas las vías de manera que formen 90 grados entre sí, se ha optado por un total de 4 tornillos para uniones cruzadas. Estos han sido seleccionados del catálogo MINITEC con referencia N° 21.0016/0

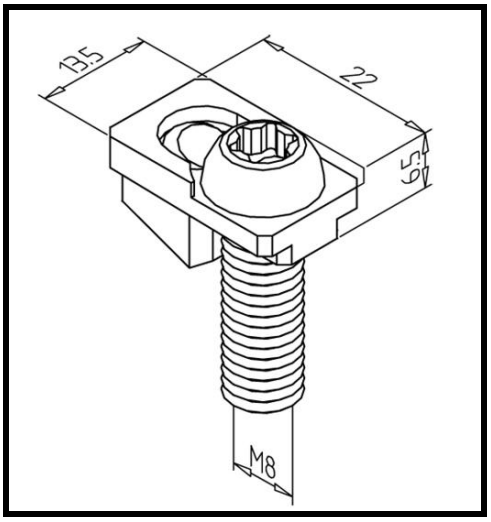


*Figura 3.8: Fijación MiniTec para perfiles de 30x30*

- **Turca deslizante M5**

La sujeción del soporte del sensor a la estructura se lleva a cabo por medio de las guías de las que dispone dicha estructura. Se utiliza para ello, un total de dos tuercas deslizantes de M5. Estas turcas han sido seleccionadas del catálogo MINITEC con referencia N° 21.1320/0.

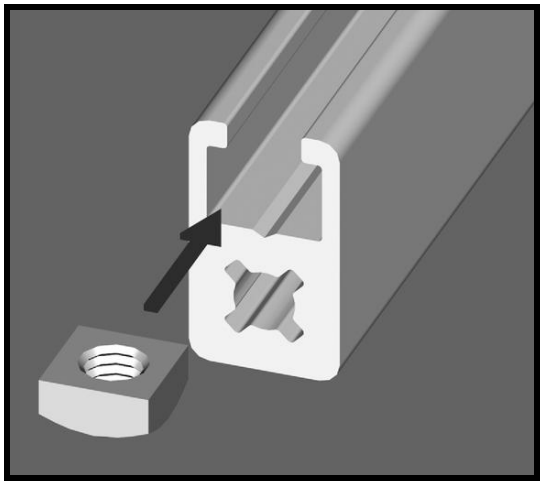

*Figura 3.9: Tuerca M5 MiniTec para perfiles de 30x30*

- **Perfil 30x30**

La estructura portante de la herramienta se fabricará a partir de un perfil modular de sección 30x30 mm. Este perfil se ha obtenido del catálogo de MINITEC con referencia Nº 20.1068/0.

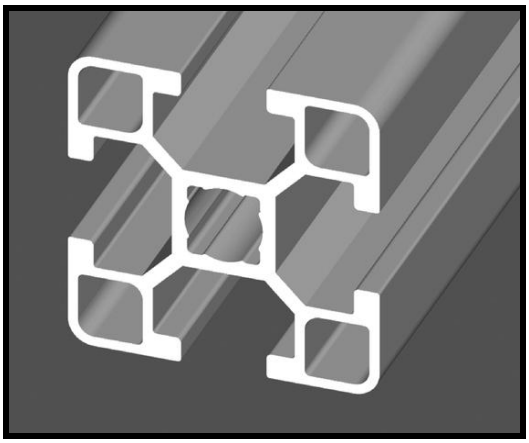

*Figura 3.10: Perfil MiniTec de 30x30*

- **Tornillería**

La unión entre la estructura portante y sujeción que conecta con la brida del motor se realiza por medio de un tornillo avellanado de M8 DIN 7991.

### 3.1.5. *IMPLEMENTACIÓN EN EL ENTORNO ROBOTSTUDIO*

Puesto que se ha elaborado una nueva herramienta que va a ser utiliza con el robot real, surge la necesidad de crear una herramienta en RobotStudio que se corresponda con la real en cuanto a dimensiones con el fin de obtener las posiciones deseadas. A continuación, se muestra una imagen de la herramienta diseñada en RobotStudio.

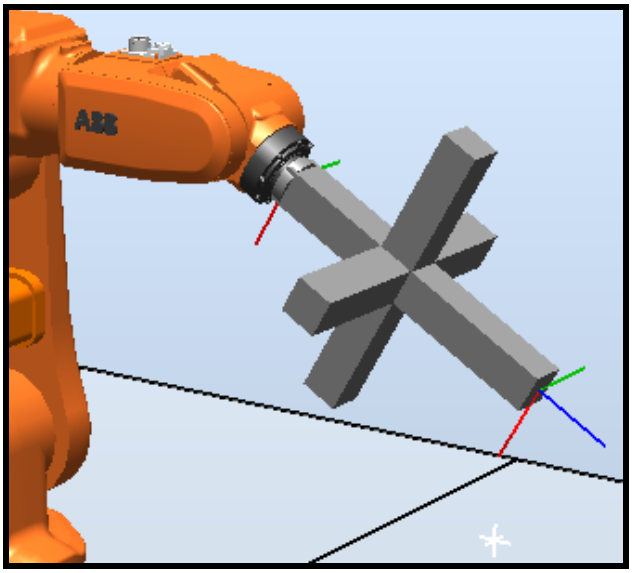


*Figura 3.11: Diseño de la herramienta en RobotStudio.*

## *3.2. Puesta en marcha del servidor OPC*

Una vez instalado el servidor ABB IRC5 en el ordenador donde se dispone del programa que va a generar las variables de orientación a usar por el manipulador, es necesario llevar a cabo una serie de acciones que permitirán adquirir esas variables en el controlador del robot.

En primer lugar, se procederá a conectar el ordenador al controlador por medio del puerto de servicio. Esta conexión se lleva a cabo mediante un cable de red Ethernet RJ45 proporcionado por el fabricante del robot.

El puerto de servicio tiene como finalidad que los ingenieros y programadores de servicio técnico se conecten directamente al controlador con un PC.

El puerto de servicio está configurado con una dirección IP fija, que es la misma para todos los controladores y no puede modificarse, y un servidor DHCP que asigna automáticamente una dirección IP al PC conectado. La dirección IP configurada para este controlador es 192.168.125.1. A continuación se puede ver una imagen de la ubicación de dicho puerto:


*Figura 3.12: Situación del puerto de servicio.*

Tras realizar esta conexión, se procede a poner en marcha el robot mediante el interruptor de encendido disponible en la parte frontal del controlador y el software RobotStudio con el programa a ejecutar. A continuación, se establece la comunicación entre RobotStudio y el controlador. Para ello, en la pestaña "Controlador", se selecciona la opción de "Añadir controlador con un clic" y elegimos la dirección IP que aparece por defecto (1192.168.125.1), puesto que solo disponemos de un controlador. Por último, mediante la pestaña "Crear relación" se transfieren todos los módulos del programa al controlador. Cabe destacar la necesidad de

seleccionar la opción de PC interface en el sistema creado, ya que de esta manera se permite la comunicación con el servidor.

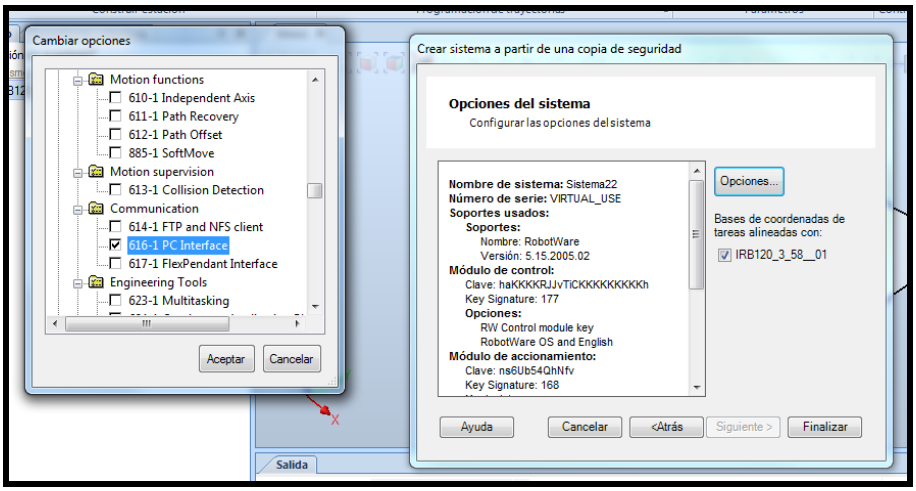


*Figura 3.13: Activación de la opción PC interface.*

Además, el servidor solo puede leer y escribir variables en RAPID de tipo Persistente (PERS), por lo que las variables a obtener del servidor han de ser configuradas como tal.

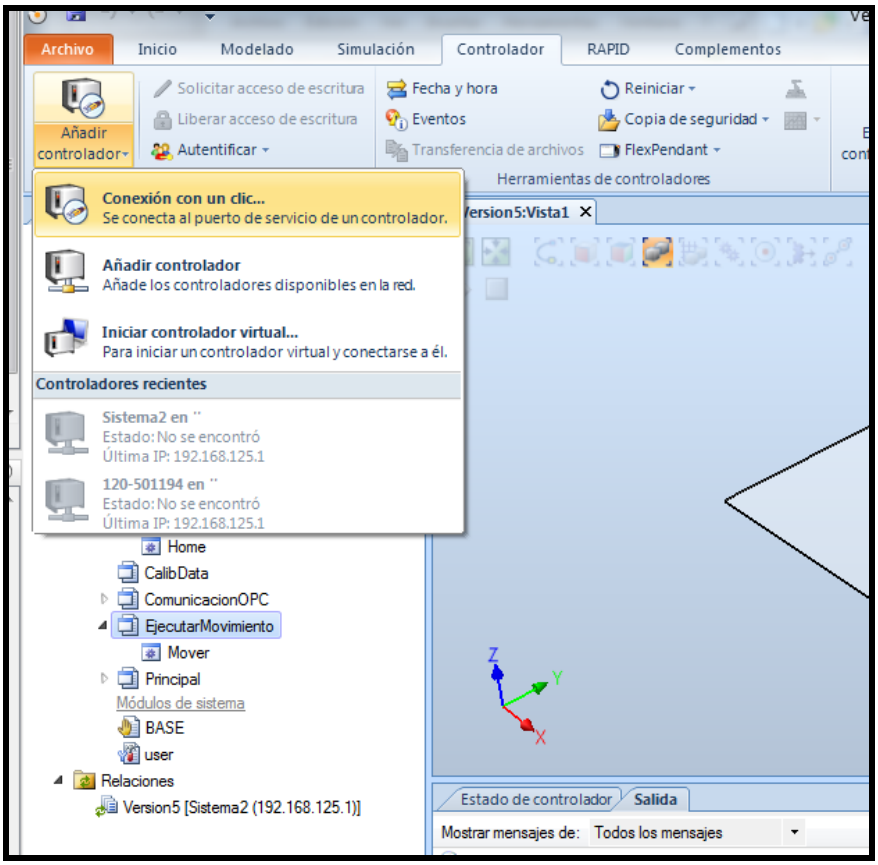


*Figura 3.14: Añadir controlador.*

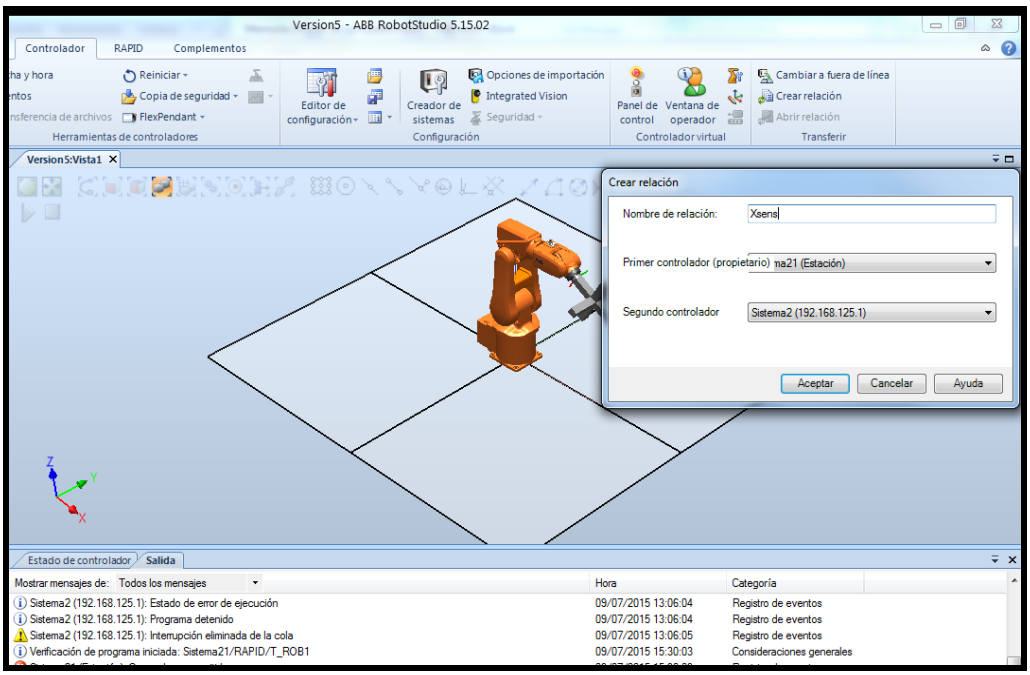


*Figura 3.15: Creación de la relación entre RobotStudio y el controlador.*

Es necesario configurar los ajustes de de acceso DCOM ya que un cliente OPC en la misma máquina que el IRC5 Servidor OPC puede fallar al conectar con el servidor.

El siguiente paso consiste en ejecutar la pantalla del Servidor ABB IRC5 con el fin de configurar el alias del robot.

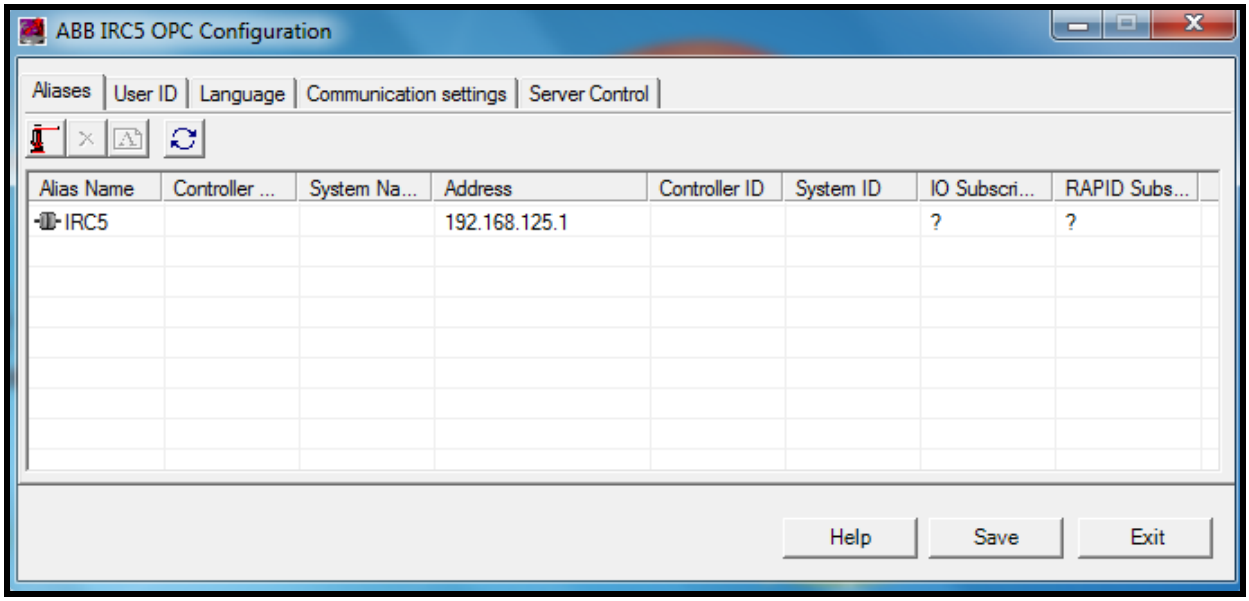


*Figura 3.16: Configuración del servidor IRC5.*

Se establece el nombre del Alias con el que poder reconocer la conexión y se aplica la dirección IP del controlador.

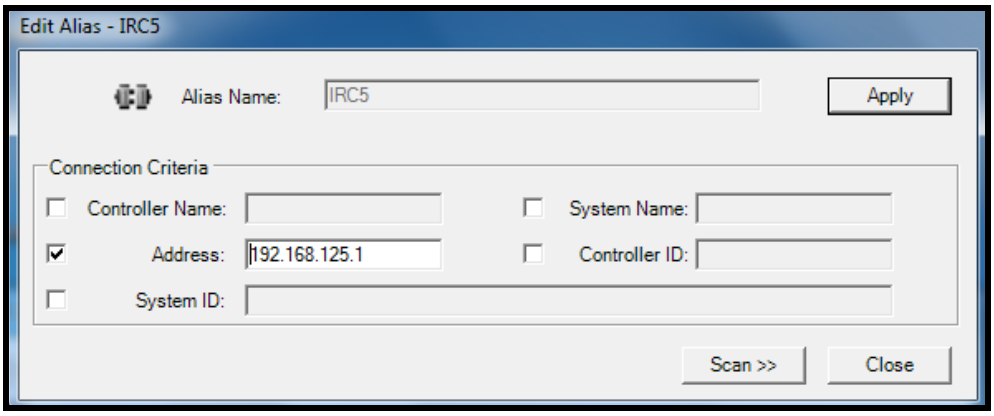


*Figura 3.17: Creación del alias.*

Finalmente, en la pestaña "Server Control" se ejecuta la aplicación OPCconfig mediante el botón "Start".

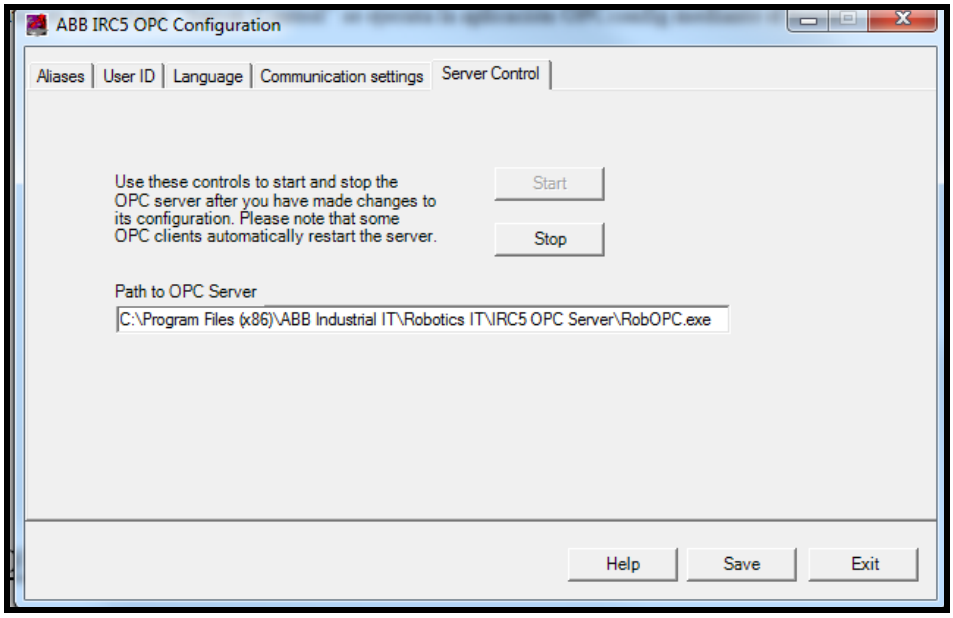


*Figura 3.18: Ejecución de la aplicación OPC Config.*

# 4. PROTOCOLO PARA EL CALIBRADO DE SENSORES

Durante el desarrollo del proyecto, ya que se lleva a cabo el empleo de sensores de tipo inercial, surgió la necesidad de elaborar un protocolo que permitiera el calibrado de dicho tipo de sensores. Por ello, sirviéndonos del robot IRB120 y de la herramienta desarrollada para el mismo, se elaboró un plan de pruebas preliminar para aplicación de contrastes con los sistemas Xsens y Robot ABB.

## *4.1. Plan de pruebas*

Antes de empezar con la calibración, fue necesaria la implementación de un protocolo a seguir, el cual nos asegurase obtener los datos pertinentes para una correcta calibración. A continuación se muestra una tabla del plan de pruebas elaborado:

| Items de evaluación | Movimiento Simple Estático | Movimiento Simple Dinámico | Movimiento Complejo Estático | Movimiento Complejo Dinámico |
|---|---|---|---|---|
| Ángulos(˚) | Flexión: 30˚, 60˚, 90˚<br>Pronación: +/-30˚,+/-60˚ | Flexión: 30˚, 60˚, 90˚<br>Pronación: +/-30˚,+/-60˚ | Flexión: 30˚, 60˚, 90˚<br>Pronación: +/-30˚,+/-60˚<br>Giro:0, 90˚ | Flexión: 30˚, 60˚, 90˚<br>Pronación: +/-30˚,+/-60˚<br>Giro: 0,90˚ |
| Velocidades (˚/s) | 10, 28'65, 57'3 | 10, 28'65, 57'4 | 10, 28'65, 57'5 | 10, 28'65, 57'6 |
| Número de repeticiones | 2 | 2 | 2 | 2 |
| Frecuencia de muestreo | 10 Hz | 10 Hz | 10 Hz | 10 Hz |

*Tabla 4.1: Plan de pruebas para el calibrado.*

Como se puede observar, se dispone de cuatro tipos de movimiento:

- Movimiento Simple Estático: Se trata de un movimiento simple que consiste en realizar un movimiento de flexión, mediante el eje cinco del robot, seguido de un movimiento de pronación por medio del eje seis. Se considera un movimiento estático ya que el robot permanecerá parado durante dos segundos en cada posición.
- Movimiento Simple Dinámico: Al igual que el anterior, se ejecuta un movimiento de flexión y pronación. Sin embargo, no se establece ningún tiempo de parada en cada posición
- Movimiento Complejo Estático: Este tipo de movimiento, consiste en realizar el proceso de flexión y pronación pariendo de dos posiciones iniciales diferentes. En

un primer lugar, la ejecución de este movimiento tiene lugar con el codo del robot (eje cuatro) posicionado en 0˚ y a continuación, en 90˚.

- Movimiento Complejo Dinámico: Al igual que el anterior, se ejecuta un movimiento de flexión y pronación en dos posiciones diferentes. Sin embargo, no se establece ningún tiempo de parada en cada posición

En la siguiente imagen, se puede observar el movimiento, representado por el movimiento del brazo humano, que lleva a cabo el robot.

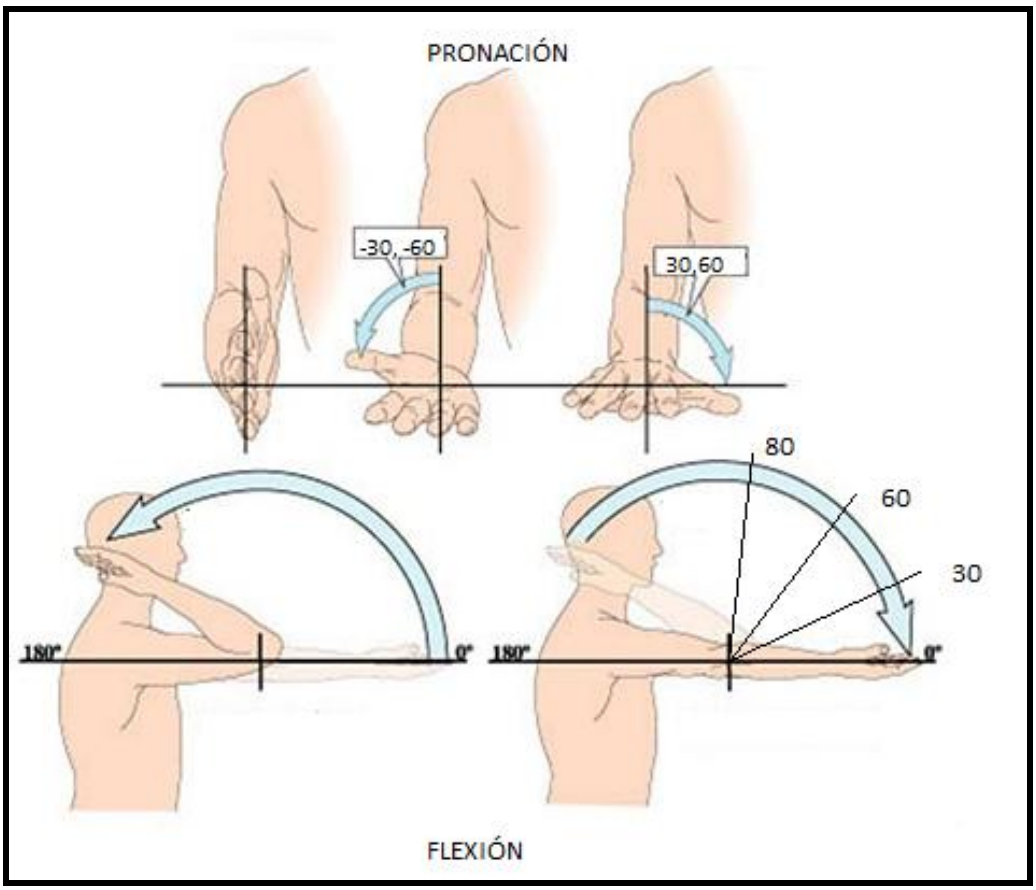


*Figura 4.1: Movimientos ejecutados por el robot.*

Se efectuarán un total de dos repeticiones para todos los tipos de movimiento. Las velocidades empleadas para el análisis serán de 10, 28'65, 57'3 (˚/s). Es necesario obtener los datos de los ángulos girados, empleando para ello una frecuencia de muestreo de 10 Hz.


*Figura 4.2: Movimientos de flexión y pronación ejecutados por el robot.*

Para el desarrollo del calibrado, la unidad inercial (IMU) se posicionará en la vía principal de la herramienta, en el extremo más alejado de la misma.

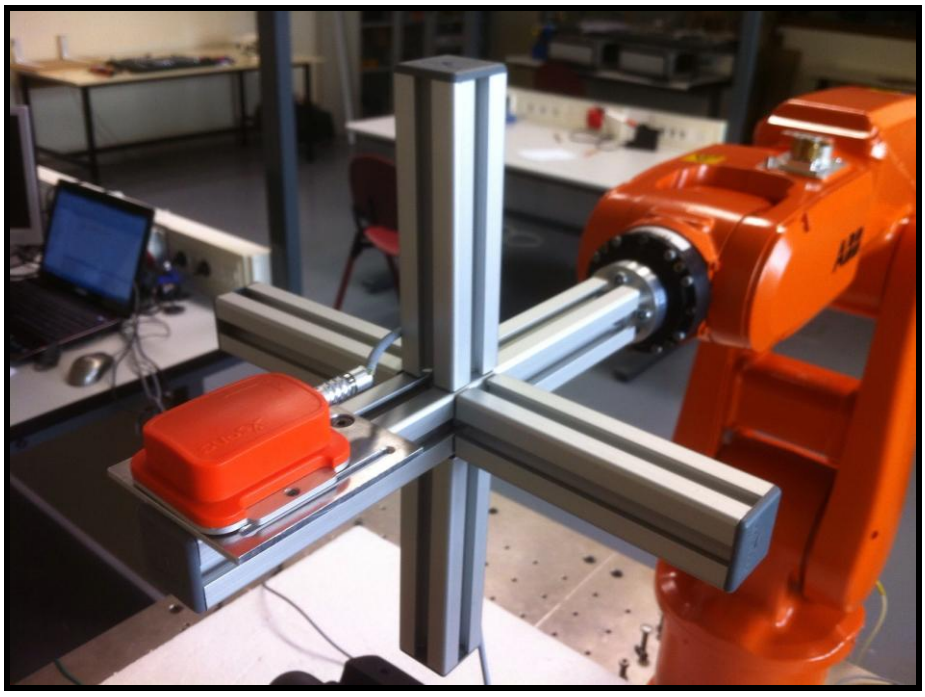

*Figura 4.3: Posicionamiento del IMU.*

## *4.2. Software de Calibración*

Una vez definido el plan de pruebas, se procedió a la elaboración de un código de control que nos permitiese implementar estos movimientos. A continuación, se muestra un diagrama de flujo del mismo:

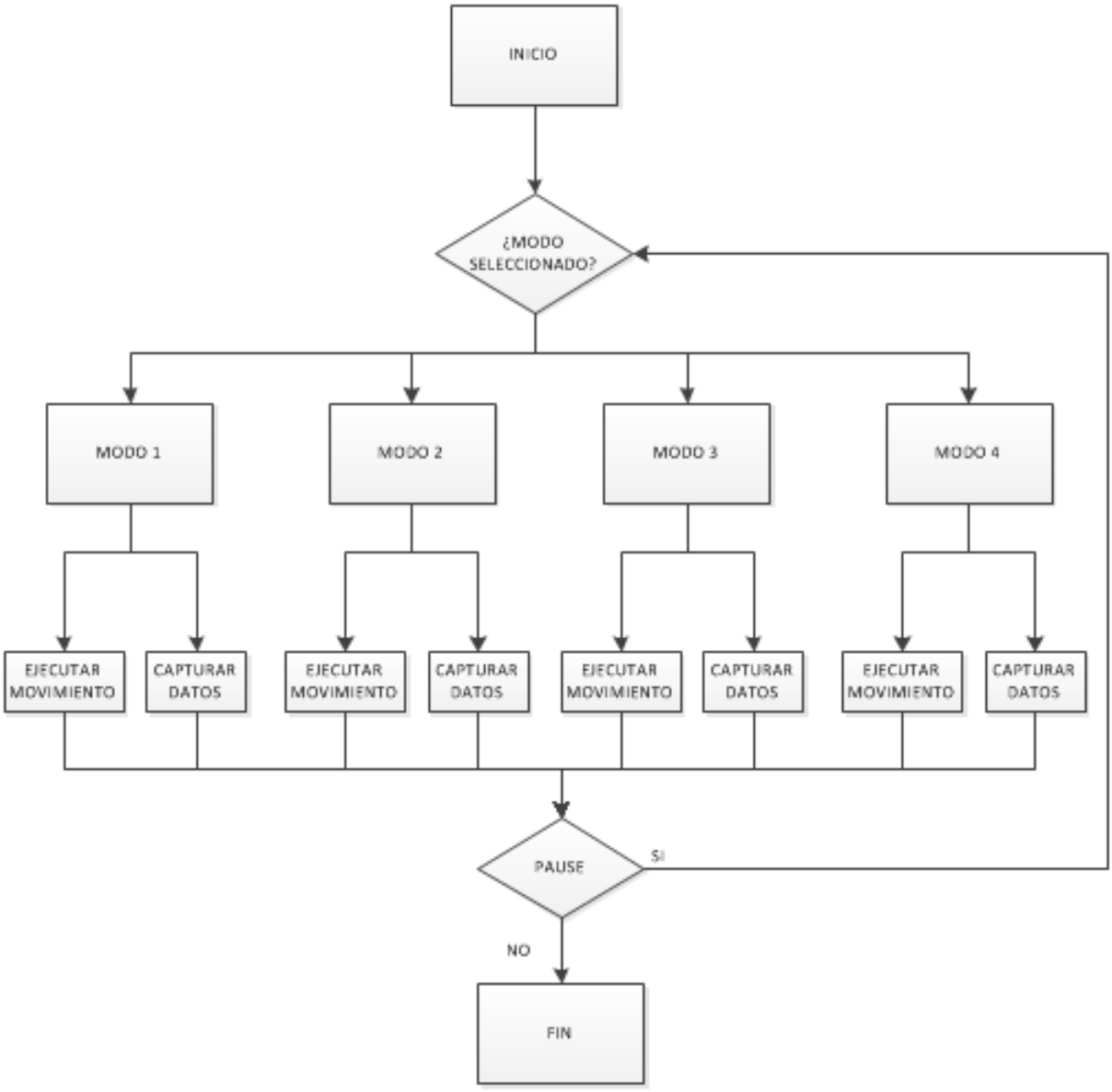


*Figura 4.4: Diagrama de flujo del software de calibración.*

Inicialmente, el manipulador se mantiene en la posición inicial, conocida como la posición de calibración. A continuación, se determina el modo de funcionamiento a realizar. Si el modo de funcionamiento es el modo 1, se ejecutará el tipo de movimiento Simple Estático, si el modo empleado es el 2, se realizará el tipo de movimiento Simple Dinámico. Si por el contrario se selecciona el modo 3 y 4, el tipo de movimiento a implementar serán el movimiento Complejo Estático y el movimiento Complejo Dinámico respectivamente. Durante la ejecución de una de las cuatro subrutinas, se lleva a cabo una captura de los datos deseados, en este caso, el ángulo

girado de los motores que intervienen en el movimiento. Finalmente, si el comando seleccionado es el botón de "Pause", se detiene la ejecución del programa. Si por el contrario, no se activa dicho comando, será posible la selección del tipo de funcionamiento deseado. Para una información más detallada consúltese Anexo 2.

## *4.3. Comparación de datos obtenidos*

Durante este apartado, se lleva a cabo un estudio para comparar la similitud entre los datos obtenidos del IMU de Xsens y el robot ABB. Para ello se ha realizado una serie de pruebas mediante el protocolo de calibración elaborado. A continuación se detallarán gráficamente los datos obtenidos con el fin de comprobar si ambos datos se corresponden, los datos tanto en el IMU como en el ABB se han obtenido con una frecuencia de muestreo de 10 Hz.

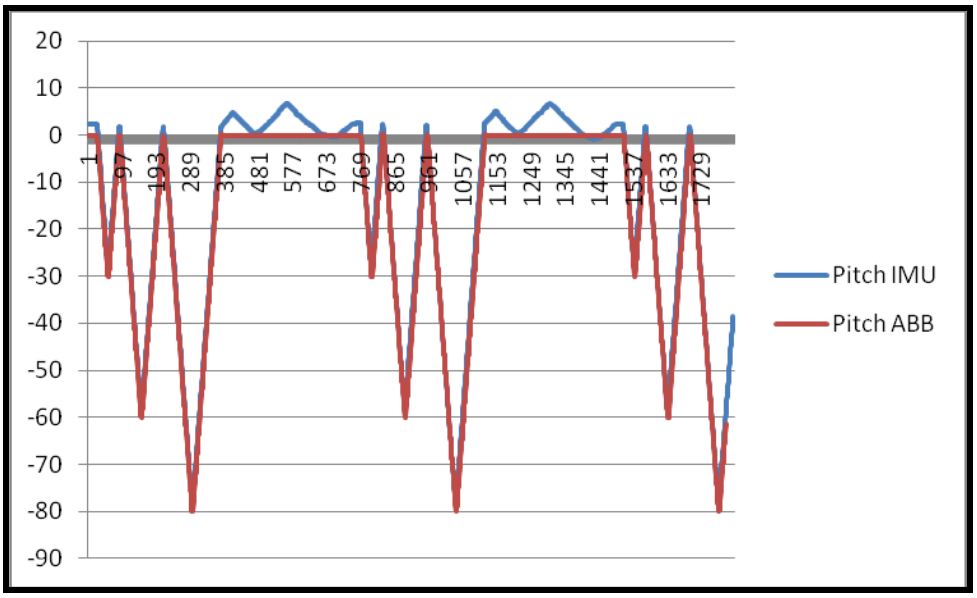


*Figura 4.5: Movimiento Flexión (velocidad de 10 grados/segundo).*

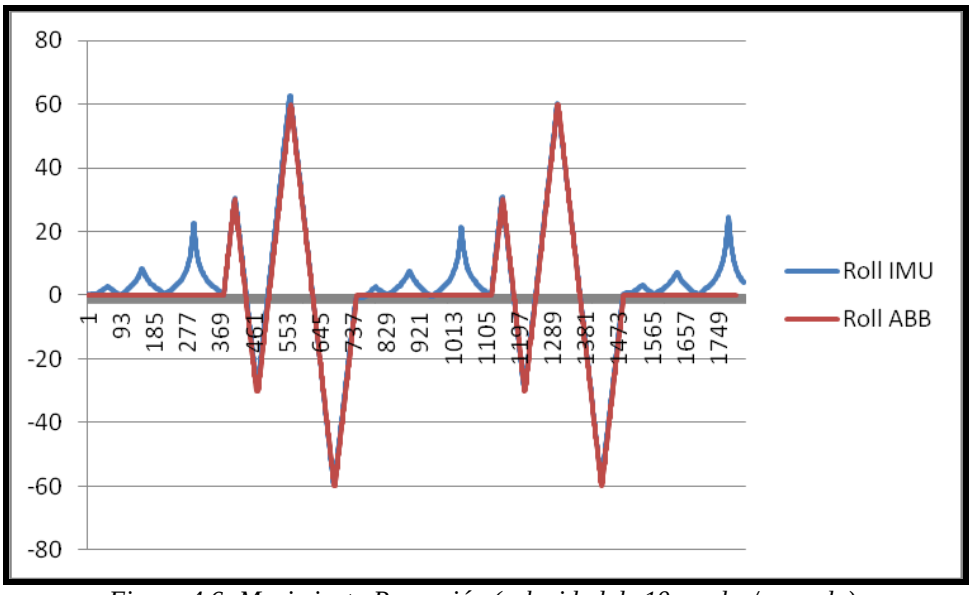


*Figura 4.6: Movimiento Pronación (velocidad de 10 grados/segundo).*

Observando las gráficas podemos apreciar un pequeño error entre ellas que se hace notable en el momento en el que el robot permanece estacionario. Este error apenas sobrepasa los 20º en Roll mientras que en Pitch no alcanza los 10º. Esto puede ser debido al entorno en el que se encuentra el IMU, ya que están situados cerca de los motores del manipulador lo que puede provocar ciertas alteraciones.

Se han comprobado que para velocidades altas, a partir de 150º/s, se produce errores de mayor índole, aumentando proporcionalmente según aumenta la velocidad. A continuación se detalla gráficamente.

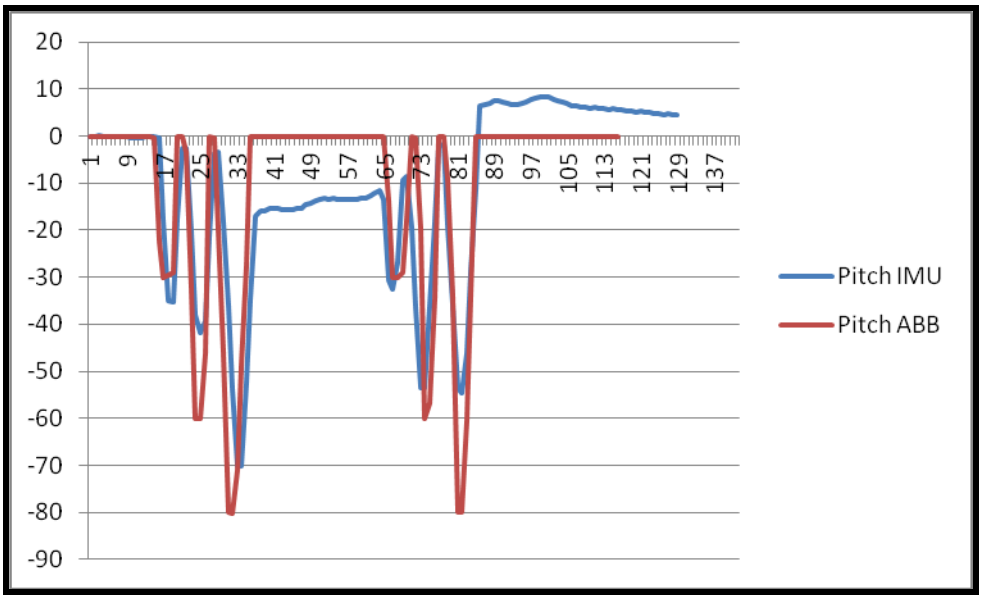


*Figura 4.7: Movimiento Flexión (velocidad de 180 grados/segundo).*

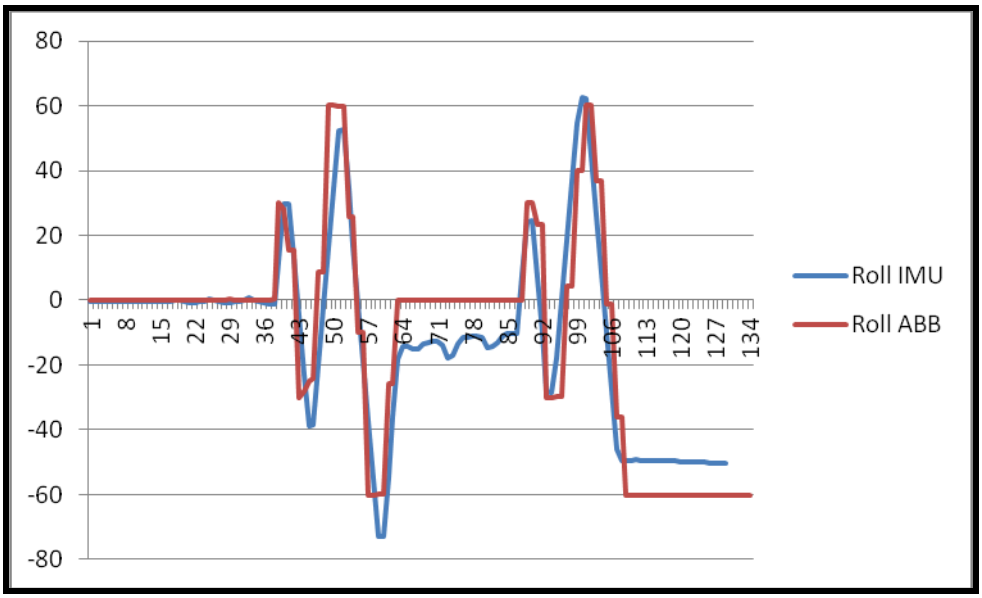


*Figura 4.8: Movimiento Pronación (velocidad de 180 grados/segundo).*

# 5. SOFTWARE DE CONTROL

A lo largo de este capítulo, se explicará detalladamente el desarrollo del software de control elaborado, tanto de la programación del controlador del manipulador, como la programación para la captura y envío de las señales de orientación.

## *5.1. Programación del controlador IRC5*

Puesto que el robot recibirá datos de la unidad de medida inercial (IMU), surge la necesidad de implementar un código de control que permita al controlador ejecutar los movimientos pertinentes.

Para ello, el robot deberá leer los datos enviados para generar el movimiento, mientras tanto, no debe permitir la escritura de nuevos datos ya que se producirían errores en la generación del objetivo del manipulador. Con el fin de limitar la escritura de nuevas variables se creó un grupo de variables cuya función es la de actuar como semáforos de lectura y escritura (OPC_semWrite/OPC_semRead). De esta manera, se controla el flujo de datos permitiendo que el servidor escriba nuevos datos en las variables destinadas a almacenarlos si el semáforo de escritura está habilitado (OPC_semWrite =TRUE) e impidiéndolo si el semáforo de lectura está habilitado (OPC_semRead =TRUE). Además, se establecen varias variables en las que se almacenarán los datos de posición y de orientación. En la siguiente tabla, se muestran una descripción de las variables creadas:

| NOMBRE | TIPO | DESCRIPCIÓN |
|---|---|---|
| OPC_X | num | Variable donde se almacena la coordenada X del punto dado |
| OPC_Y | num | Variable donde se almacena la coordenada Y del punto dado |
| OPC_Z | num | Variable donde se almacena la coordenada Y del punto dado |
| OPC_RX | num | Variable donde se almacena el giro en X de la orientación dada |
| OPC_RY | num | Variable donde se almacena el giro en X de la orientación dada |
| OPC_RZ | num | Variable donde se almacena el giro en X de la orientación dada |
| OPC_semWrite | Bool | Variable que habilita la escritura de datos |
| OPC_semRead | Bool | Variable que habilita la lectura de datos |

*Tabla 5.1: Conjunto de variables creadas para el controlador.*

El código se divide en varias subrutinas, las cuales serán explicadas detalladamente a continuación.

La rutina principal es la encargada de ejecutar de manera continua, manteniendo el orden de llegada de los datos, el movimiento del manipulador. En primer lugar, se posiciona el robot en una posición definida como la posición de calibración, esta será el punto de partida cada vez que se inicie el control. El siguiente paso será iniciar la comunicación para permitir la entrada de datos al controlador y finalmente ejecutar el movimiento deseado. A continuación se muestra el flujograma de la rutina principal del controlador.

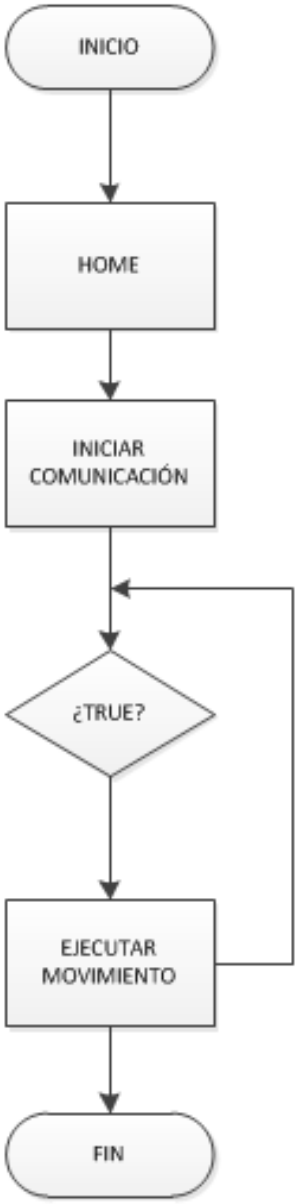


*Figura 5.1: Diagrama de flujo de la rutina principal del software de del controlador.*

La captura de datos se implementa por medio de una subrutina de interrupción. Esta interrupción se produce por el cambio en la variable de lectura (OPC_semRead). En primer lugar se establece la conexión de la interrupción, es decir, se conecta la variable de lectura con la interrupción a ejecutar. Seguidamente, se procede a deshabilitar la escritura puesto que el siguiente paso es la creación del objetivo que el manipulador debe alcanzar. Finalmente se habilita la opción de escritura. En la siguiente imagen, se muestra un diagrama de flujo del la rutina de interrupción.


*Figura 5.2: Diagrama de flujo de la rutina de interrupción.*

Con el fin de realizar el movimiento del manipulador, se elaboró una rutina que simplemente se encarga de recibir los datos proporcionados por el servidor y ejecutar la instrucción de movimiento necesaria para obtener el objetivo deseado.

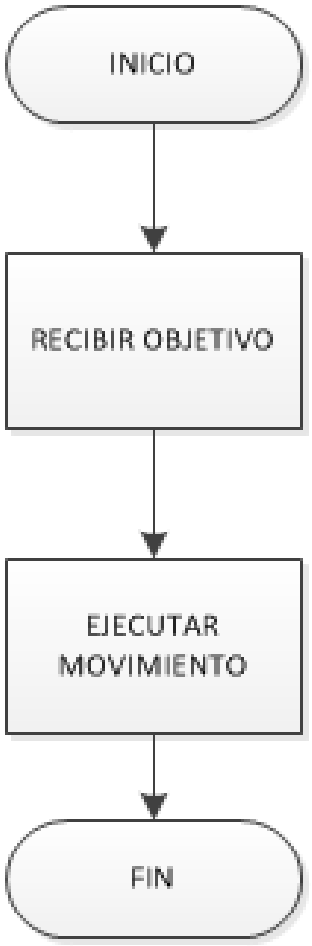


*Figura 5.3: Diagrama de flujo de la rutina de ejecución del movimiento.*

## 5.2. Programación para la comunicación mediante OPC Toolbox de Matlab y el algoritmo de captura de datos

La implementación de este código tiene lugar mediante el software Matlab. Para ello no servimos de diversas librerías, una ya implementada en el propio software (OPCToolbox) y otra elaborada por el Departamento de Ingeniería de sistemas y Automática (InerSensToolbox). Para una información del código más detallada, ver Anexo 4.

### 5.2.1. COMUNICACIÓN OPC CON MATLAB

Para establecer la comunicación del código encargado de la captura de datos con el servidor OPC ABB IRC5, se emplea el cliente OPC de Matlab. Se trata de una librería disponible en Matlab que permite resolver el problema de la comunicación entre ambos sistemas. Dicha comunicación puede implementarse por medio de la interfaz de la Toolbox o directamente elaborando el código pertinente, ambos casos se muestran en la ayuda de Matlab (ver referencia [12]).

Para el caso que nos concierne, se ha optado por establecer la comunicación mediante comandos. A continuación se muestran detalladamente los pasos seguidos.

En primer lugar, es necesario definir una conexión. Para ello se determina la dirección IP (hostname) con el fin de identificar el equipo dentro de la red. Esta dirección IP es utilizada por los protocolos OPC Data Access para determinar los servidores OPC disponibles en ese equipo, y para comunicar con el ordenador para establecer una conexión con el servidor. Puesto que la conexión al servidor tiene lugar en la misma máquina que el cliente, se ha utilizado "*localhost*" como hostname. El segundo paso para definir la conexión es determinar la ID del servidor a usar, en este caso el servidor ABB IRC5. Para conocer la designación del servidor a usar se puede ejecutar en Matlab el siguiente comando:

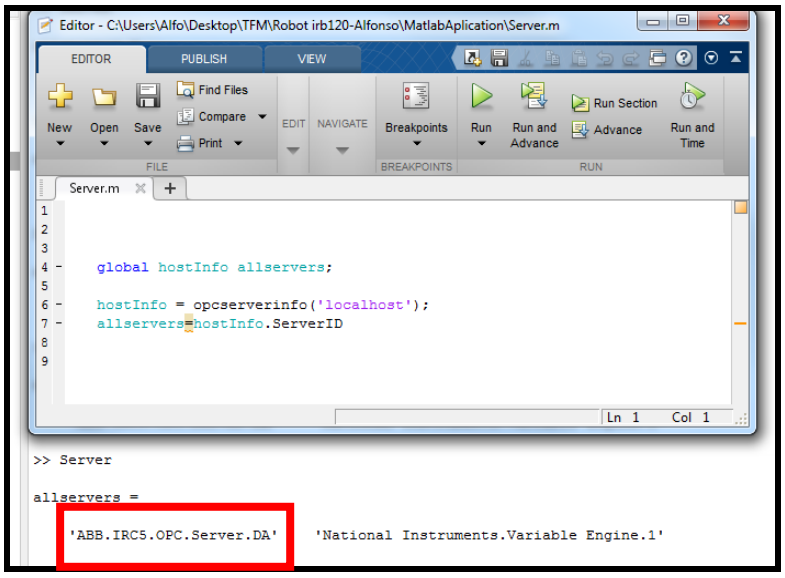


*Figura 5.4: Determinación del servidor.*

Una vez determinado el nombre de host y el ID para conectarse al servidor, se procede a crear un objeto cliente OPC Data Access y establecer la conexión. El cliente controla el estado de la conexión con el servidor, y almacena los eventos que se producen a partir de ese servidor (como la notificación de los datos cambiantes) en el registro de eventos.

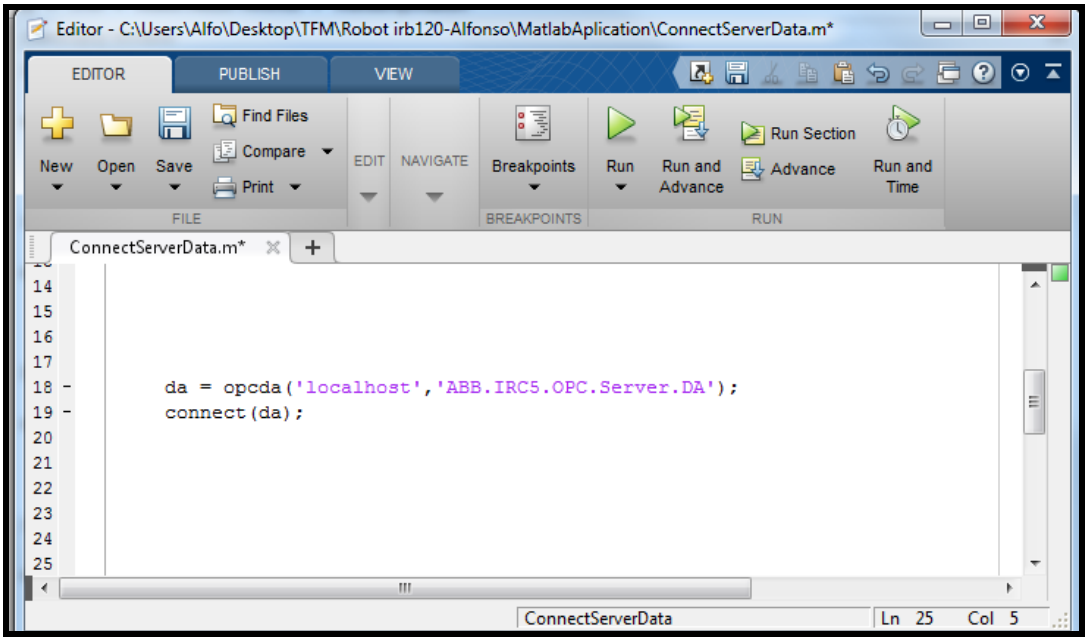


*Figura 5.5: Creación del objeto y conexión.*

Tras establecer la conexión, es necesario crear un grupo de objetos de acceso de datos que permita disponer de una colección de elementos y a continuación, añadir los elementos deseados al grupo. Solamente una vez que estos elementos son agregados a un grupo pueden ser manejados. Para ello se emplean los siguientes comandos:

```
18
19
20 -    group = addgroup(da);
21 -    item_x = additem(group, strcat(tagsPath,'.OPC_X'));
22 -    item_y = additem(group, strcat(tagsPath,'.OPC_Y'));
23 -    item_z = additem(group, strcat(tagsPath,'.OPC_Z'));
24 -    item_Rx = additem(group, strcat(tagsPath,'.OPC_RX'));
25 -    item_Ry = additem(group, strcat(tagsPath,'.OPC_RY'));
26 -    item_Rz = additem(group, strcat(tagsPath,'.OPC_RZ'));
27
```

*Figura 5.6: Creación del grupo de objetos.*

Finalmente, la escritura y lectura de los datos se lleva a cabo mediante las funciones *write* y *read*.
Con el fin de agilizar el proceso de conexión y escritura de los elementos se han implementado una serie de funciones. Las variables creadas para la comunicación son las siguientes:

| NOMBRE | DESCRIPCIÓN |
|---|---|
| tagsPath | Variable donde se almacena el directorio de las variable a escribir |
| da | Objeto cliente |
| group | Grupo de objetos donde se guardan los elementos deseados |
| item_x | Elemento donde se almacena la coordenada X del punto dado |
| item_y | Elemento donde se almacena la coordenada Y del punto dado |
| item_z | Elemento donde se almacena la coordenada Z del punto dado |
| item_Rx | Elemento donde se almacena el giro en X de la orientación dada |
| item_Ry | Elemento donde se almacena el giro en Y de la orientación dada |
| item_Rz | Elemento donde se almacena el giro en Z de la orientación dada |
| item_semRead | Elemento que habilita/deshabilita la lectura |
| item_semWrite | Elemento que habilita/deshabilita la escritura |

*Tabla 5.2: Conjunto de variables creadas para la comunicación.*

.

### 5.2.2. *REPRESENTACIÓN DE LA ORIENTACIÓN*

Como anteriormente hemos explicado en el apartado *2.2.3*, los Xsens son capaces de medir aceleraciones, velocidades de giro e intensidad del campo magnético en tres ejes perpendiculares. Mediante la Toolbox InerSens, el Xsens nos proporciona de uno a tres archivos de datos por sensor, estos datos pueden ser:

- Datos crudos sin calibrar.
- Datos calibrados.
- Datos de orientación.

Los datos de orientación pueden ser obtenidos en tres formatos:

- Cuaternios
- Matriz de Rotación
- Ángulos de Euler

El robot ABB IRB120 trabaja con Cuaternios para definir la orientación de la herramienta de trabajo. La orientación de un sistema de coordenadas puede describirse mediante una matriz de rotación que describe la dirección de los ejes del sistema de coordenadas respecto de un sistema de referencia.

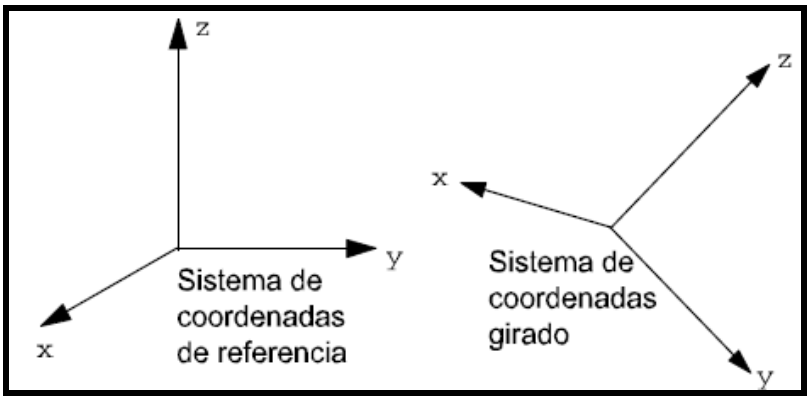


*Figura 5.7: Sistema de coordenadas de referencia.*

Los ejes del sistema de coordenadas girado (x, y, z) son vectores que pueden expresarse en el sistema de coordenadas de referencia de la forma siguiente:

**x** = (x1, x2, x3)

**y** = (y1, y2, y3)

**z** = (z1, z2, z3)

Esto significa que el componente x del vector x del sistema de coordenadas de referencia será x1, el componente y será x2, etc.

Estos tres vectores pueden reunirse en una matriz (una matriz de rotación) en la que cada uno de los vectores compone una de las columnas.

Un cuaternio es sólo una forma más concisa de referirse a esta matriz de rotación. Los Cuaternios se calculan partiendo de los elementos de la matriz de rotación:

| | |
|---|---|
| $q1 = \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2}$ | |
| $q2 = \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2}$ | sign q2 = sign (y 3-z 2) |
| $q3 = \frac{\sqrt{y_2 - x_1 - z_3 + 1}}{2}$ | sign q3 = sign (z 1-x 3) |
| $q4 = \frac{\sqrt{z_3 - x_1 - y_2 + 1}}{2}$ | sign q4 = sign (x 2-y 1) |

*Figura 5.8: Cálculo de los Cuaternios.*

Sin embargo el propio lenguaje del robot dispone de funciones que permiten realizar directamente la conversión de los Ángulos de Euler a Cuaternios.

Por la facilidad de interpretación gráfica que representan dichos ángulos frente a los Cuaternios, se ha optado por normalizar la orientación entorno a los Ángulos de Euler.

### 5.2.3. *ADAPTACIÓN DEL ORDEN DE GIRO*

Los ángulos de Euler constituyen un conjunto de tres coordenadas angulares que sirven para especificar la orientación de un sistema de referencia de ejes ortogonales, normalmente móvil, respecto a otro sistema de referencia de ejes ortogonales normalmente fijos.

Desafortunadamente no existe un estándar para la nomenclatura de los Ángulos de Euler. Por este motivo, definiremos como el giro en el eje X como "Roll", giro en Y como "Pitch" y giro en Z como "Yaw".

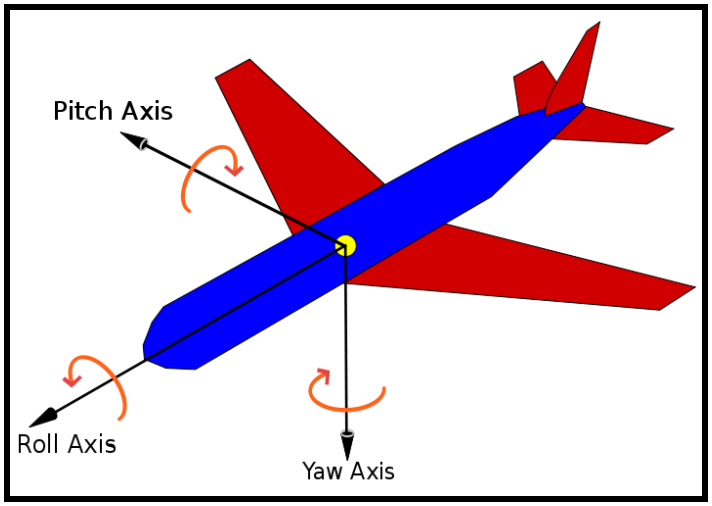


*Figura 5.9: Nomenclatura de los ángulos de Euler.*

Existe un total de 24 convenciones de los ángulos. En los Xsens, las rotaciones se llevan a cabo en el orden Roll ($\alpha$), Pitch ($\beta$) y Yaw ($\gamma$). A esta convención que permite especificar una orientación se denomina Ángulos Fijos X-Y-Z. Reciben la denominación de "fijos" debido a que las rotaciones se especifican sobre la trama de referencia fija.

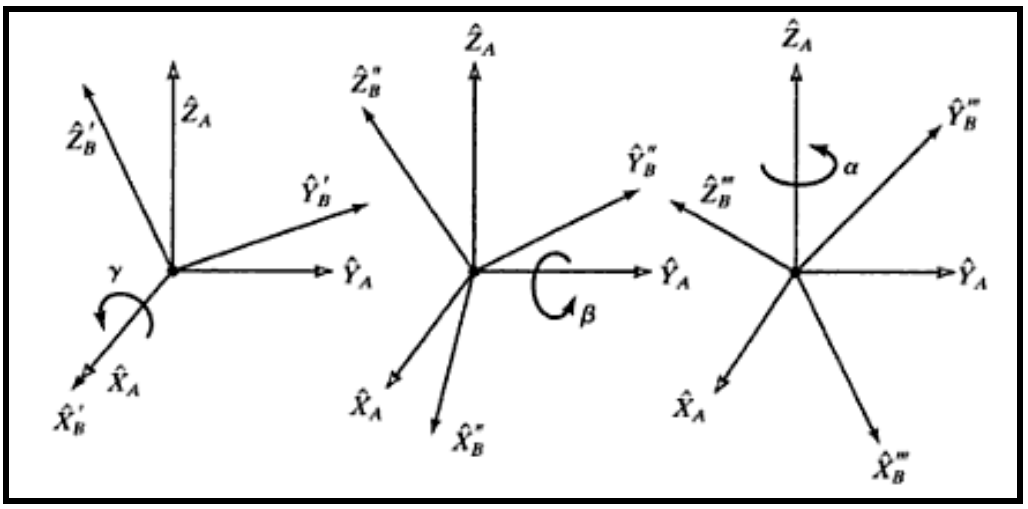


*Figura 5.10: Orden de las rotaciones.*

Por otro lado, en el robot ABB IRB120 las rotaciones no tienen lugar en el mismo orden. Al igual que en los Xsens la rotaciones se producen sobre una trama de referencia fija, sin embargo, en este caso el orden es Yaw (γ), Pitch (β) y Roll (α). Es por esto, por lo que es necesario llevar a cabo una adaptación de los ángulos de giro obtenidos de los Xsens para que la herramienta realice los giros de orientación en el mismo orden. A continuación, se detalla el desarrollo de esta adaptación.

Una vez capturados los tres ángulos (Roll, Pitch, Yaw) mediante el Xsens, se selecciona la convención correspondiente, en este caso XYZ y se sustituyen en la matriz de rotación, obteniéndose una matriz de 3x3 con todos sus elementos conocidos.

$$R_{XYZ}(\gamma,\beta,\alpha) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

*Figura 5.11: Convención XYZ Ángulos de Euler.*

El paso siguiente es extraer los ángulos los ángulos ZYX equivalentes a partir de la matriz de rotación anteriormente calculada. Se trata de igualar dicha matriz con la matriz de rotación de la convención ZYX obteniendo un conjunto de nueve ecuaciones, seis de ellas dependencias, por lo que en realidad disponemos de tres ecuaciones con tres incógnitas.

$$R_{ZXY}(\gamma,\beta,\alpha)=\begin{bmatrix} s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & s\alpha c\beta \\ c\beta s\gamma & c\beta c\gamma & -s\beta \\ c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha c\beta \end{bmatrix}$$

*Figura 5.12: Convención ZYX Ángulos de Euler.*

$$\begin{aligned} \beta &= \text{Atan2}(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}), \\ \alpha &= \text{Atan2}(r_{21}/c\beta, r_{11}/c\beta), \\ \gamma &= \text{Atan2}(r_{32}/c\beta, r_{33}/c\beta), \end{aligned}$$

*Figura 5.13: Obtención de los nuevos ángulos.*

Finalmente, una vez el calculados  los nuevos ángulos, serán enviados al manipulador.

### 5.2.4. *PROGRAMA DE CAPTURA DE DATOS*

Tras el estudio detallado de la física interna del IMU de Xsens como de la del robot ABB, se procedió a la elaboración del software. Para ello nos servimos de la Toolbox InerSens, se trata de una librería que dispone de diversas funciones la cuales permiten obtener todo tipo de datos referentes al IMU de Xsens además de otros tipos de sensores inerciales. Estas funciones permiten desarrollar diversas aplicaciones siguiendo siempre los mismos pasos, cinco en concreto:

- initsilop(): Esta función lleva a cabo la inicialización del sistema de procesamiento de las aplicaciones estándar de la Toolbox. Debe ser el primer comando utilizado en estas aplicaciones permitiendo así la adición de los sensores y de los algoritmos necesarios. Esta función no requiere de argumentos de entrada. A continuación se muestra un ejemplo de uso:

```
> initsilop();
```

- addimu(): Mediante esta función definimos el IMU a usar en la aplicación. Se debe incluir la lista completa de IMUs a usar antes de realizar la conexión. Esta función cuenta con tres argumentos de entrada:

```
addimu(posicion,numserie,orientacion);
```

  - Posición: Es una cadena de texto que contiene la posición en la que está el sensor. Para este caso recibirá la cadena "*COG"*.

  - Numserie: Se refiere al número de serie del IMU a utilizar, esta información aparece detallada en la parte posterior de dicho sensor.

  - Orientación: Vector que contiene la relación entre los ejes del acelerómetro y los ejes anatómicos que se usarán en la aplicación. Si el acelerómetro se sitúa en el COG, por defecto tendrá un valor de [3,-2,1]. Lo que indica que el eje 3 (Z) del acelerómetro será nuestro eje 1(antero-posterior o X) el eje -2(-Y) del acelerómetro será nuestro eje 2 (vertical o Y) y el eje 1 (X) del acelerómetro será nuestro eje 3 (Vertical o Z). En otros puntos vale [1, 2,3] por defecto, es decir, no se produce reorientación. Se aceptan valores negativos para indicar que el eje anatómico y el del acelerómetro son opuestos.

  A continuación se muestra un ejemplo de uso:

```
> addimu('COG', 303904,[3,-2,1]);
```

- connectsilop(): Permite conectar el sistema de procesamiento de las aplicaciones con la fuente de datos, es decir, conecta el XbusMaster de acuerdo a la configuración del IMU previamente seleccionada. Cuenta con cuatro argumentos de entrada:

```
connectsilop(driver, source, freq, updateeach);
```

  - driver: Indica el modo de funcionamiento, para ello se indica el tipo de IMU a usar, en este caso IMU MTi.

  - source: Determina el puerto del que leer los datos, es decir, el puerto al que está conectado el Xsens, en este caso el puerto COM7.

  - freq: Especifica la frecuencia de muestreo de datos. Se ha definido una frecuencia de 100 Hz.

  - Updateeach: Determina el tiempo de procesado de los datos recibidos.

A continuación se muestra un ejemplo del modo de uso:

```
> connectsilop ('MTiG','COM7', 100, 3, [115200,2]);
```

- addalgoritmo(): Añade un algoritmos al sistema de procesamiento de aplicaciones. Esta función debe ejecutarse tras conectar el sistema y antes de llevar a cabo el procesamiento de datos. Dispone de un total de cuatro parámetros de entrada:

```
addalgoritmo (nombre,n_valores_retorno,senhales,params);
```

  - nombre: Nombre que recibe el algoritmo.
  - n_valores_retorno: Se trata de un "*Cell Array*" que contiene los nombres de las señales generadas por el algoritmo.
  - senhales: Se trata de un "*Cell Array"* con los nombres de las señales a usar.
  - params: vector con los datos que necesita el algoritmo.

  Posteriormente se explicará con más detalle el desarrollo del algoritmo generado para la captura de los datos pertinentes
- playsilop(): Esta función pone en funcionamiento el sistema, inicia el procesamiento de datos y no finaliza hasta que se pulse la tecla "*ESC*". Esta función dispone de argumentos de entrada que permiten generar y guardar archivos, sin embargo ha sido utilizada sin esta configuración.

```
Playsilop (salvar,fichero);
```

  A continuación se muestra un ejemplo:

```
> playsilop();
```

Una vez determinadas las funciones necesarias para la implementación del código, se procedió a la elaboración del algoritmo de captura de datos de orientación. Para ello, es necesario conocer el funcionamiento del núcleo del Silop. Este, consiste en una matriz que contiene los datos de los IMU empleados, insertando el dato más nuevo en la última fila de la matriz. Cada columna de la matriz se corresponde con una variable medida por el IMU. El tiempo total de datos disponible se corresponde con el parámetro *updateeach* de *connectsilop()*.

Por cada algoritmo generado se obtiene una matriz donde se almacenan los resultados. El número de columnas es configurable, permitiendo la adaptación a las necesidades concretas del algoritmo.

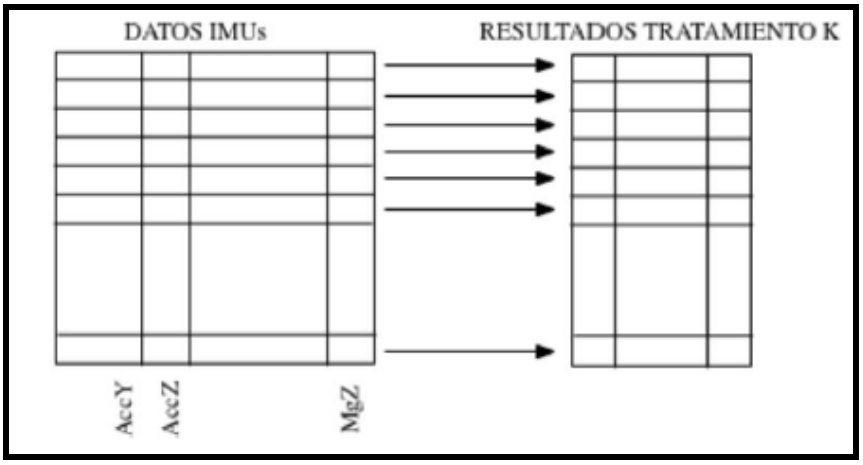


*Figura 5.14: Matriz de almacenamiento de datos.*

En el momento en que se reciba una nueva muestra de los IMUs, la ventana de datos se desplaza para dejar espacio a los nuevos datos, los cuales serán insertados por el final.

Todas las ventanas de resultados asociadas a los distintos tratamientos se desplazan en la misma proporción rellenándose con valores NaN las filas correspondientes a las nuevas muestras. De esta manera, un valor NaN en una determinada posición de la ventana de tratamientos ha de interpretarse como un dato aún no calculado.

Se procede a invocar secuencialmente a todos los algoritmos de tratamiento. Cada algoritmo recibirá como parámetro (entre otros) el estado actual de la matriz de resultados correspondiente, que ha de procesar y devolver. Recibirá además una submatriz construida a partir de determinadas columnas de la matriz de datos muestreados, ventanas de resultados de otros algoritmos, y parámetro específico de configuración, todo ello configurable a partir de la función addalgoritmo.

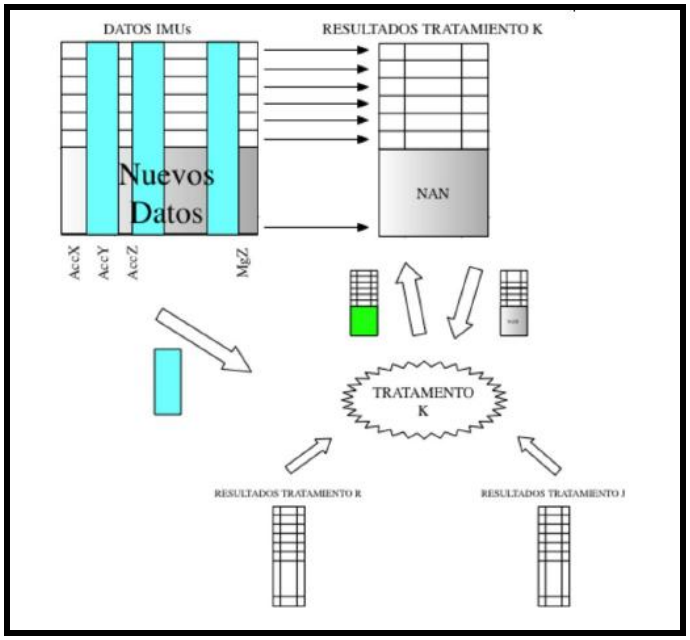


*Figura 5.15: Tratamiento de los datos recogidos.*

Para ello se implementó una función con tres argumentos de entrada:

```
alg_captura_senhales (previos, senhales, params)
```

Esta función recibe los datos de los Ángulos de Euler, los cuales son almacenados en una matriz donde la primera columna se corresponde con el ángulo girado en X (Roll), la segunda con el ángulo girado en Y (Pitch) y la tercera con el ángulo girado en Z (Yaw). Como anteriormente hemos explicado, el dato más reciente se sitúa en la última fila de la matriz

```
if (~isempty(senhales))

   Roll=(senhales(end,1));%A(end, : )
   Pitch=(senhales(end,2));
   Yaw=(senhales(end,3));
end
```

*Figura 5.16: Comando para la obtención de los Ángulos de Euler.*

A continuación lleva a cabo la adaptación del orden de giro para finalmente ser enviados al robot. Para ello se elaboró una función que se encarga de realizar la conversión de ángulos.

```
ConvencionAngulosEuler (Roll, Pitch, Yaw)
```

Finalmente, se procede a la escritura de los datos de orientación a los datos que serán enviados al robot.

```
try
     writeasync (item_Rx,Roll);
     writeasync (item_Ry,Pitch);
     writeasync (item_Rz,Yaw);
     writeasync (item_x,x);
     writeasync (item_y,y);
     writeasync (item_z,z);

catch theError
      rethrow (theError);
end
```

*Figura 5.17: escritura de datos.*

# 6. RESULTADOS

Durante el desarrollo de este proyecto se ha llevado a cabo el control de la orientación en 3D del robot ABB IRB120 por medio de una unidad de medida inercial (IMU) de la marca Xsens. Para ello nos hemos servido de diversas herramientas típicas de la robótica industrial.

El protocolo de comunicación utilizado (OPC) funciona correctamente ya que la aplicación implementada en Matlab proporciona loas datos de orientación con una frecuencia de 1 Hz, por lo que el tiempo de envío de datos nunca será inferior al tiempo recomendado por el fabricante (200ms).

El robot ejecuta la orientación proporcionada por el IMU de Xsens, llevando a cabo el mismo orden de giro de los ángulos de orientación (ZYX). Sin embargo, debido a la Toolbox de la que se dispone para obtener dichos datos en la aplicación de Matlab, no se permite inicializar el IMU de Xsens en una orientación determinada, obteniendo posteriormente los movimientos referenciados a dicha orientación. Por este motivo no se obtiene una percepción visual de que ambas orientaciones, las del Xsens y la de la herramienta del robot, coincidan.

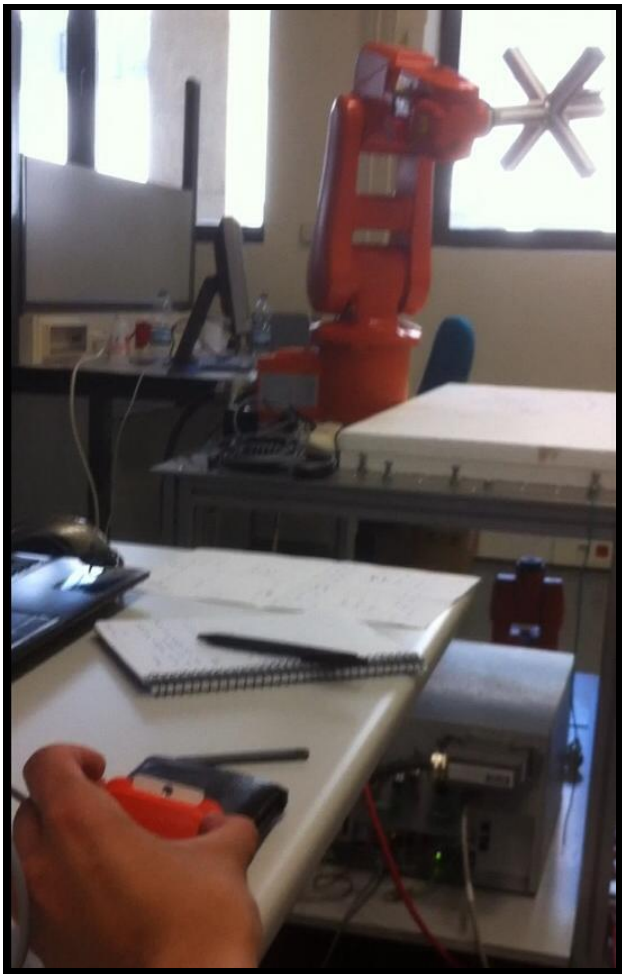

*Figura 6.1: Orientación de la herramienta del robot.*

*Figura 6.1: Orientación del IMU de Xsens*

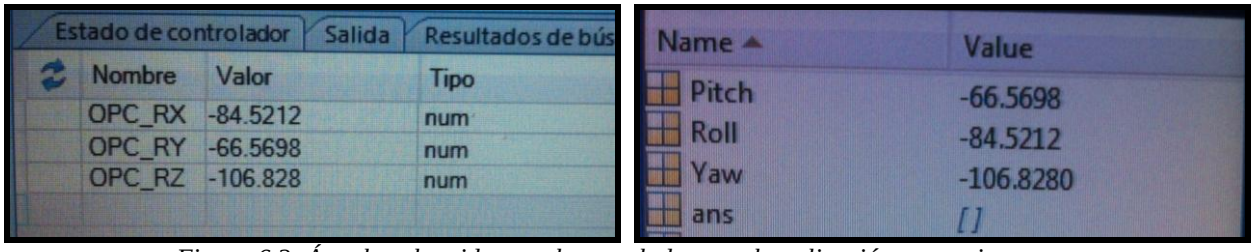


*Figura 6.2: Ángulos obtenidos en el controlador y en la aplicación respectivamente.*

Como se observa en las figuras anteriores, los ángulos obtenidos así como el orden de giro en ambos subsistemas coinciden, sin embargo, visualmente no es posible determinar si la orientación es la correcta, es necesario examinar estos ángulos para determinar la coincidencia. Debido a esto, se ha decidido disminuir la velocidad de ejecución de la instrucción de movimiento con el fin contar con tiempo suficiente para poder parar el sistema, ya que en ciertos momentos el robot puede salirse de su rango de trabajo y colisionar con la mesa que lo soporta o consigo mismo. Esta disminución de la velocidad, provoca cierto retraso en la ejecución del movimiento deseado, sin embargo es necesario para evitar dañar el robot.

La problemática de establecer un movimiento que está fuera del rango de trabajo del robot, se debe a que en algunas ocasiones el robot no dispone de una configuración cinemática para alcanzar cierto punto del espacio con una orientación determinada.

# 7. CONCLUSIONES

Una vez finalizado el proyecto, se extraen las siguientes conclusiones:

- Se ha implementado un sistema que cumple el objetivo general del proyecto, así como los requisitos de mínimos detallados al inicio del proyecto.

- El diseño de la herramienta cumple su función de permitir el posicionamiento del IMU con un alto grado de flexibilidad.

- La Comunicación OPC es un tipo de comunicación remota viable para este tipo de aplicaciones. Además de disponer de un servidor proporcionado por el propio fabricante del robot, proporciona los tiempos de envío/recepción de datos necesarios para cerrar el lazo de control.

- La aplicación elaborada en Matlab permite llevar a cabo la captura y el envío de datos al manipulador de manera satisfactoria.

- La orientación obtenida en el robot coincide con la generada por el IMU de Xsens. Sin embargo, es posible corregir el problema de percepción visual de la orientación mediante el empleo de dos IMU, estableciendo uno fijo que trabaja como referencia y actuando sobre el otro para determinar la orientación.

# 8. PRESUPUESTO

A lo largo de este capítulo, se analizarán los costes producidos durante la realización del proyecto. Este presupuesto consta de:

- Coste de Materiales.
- Coste de Amortización de Equipos.
- Coste de Mano de Obra.
- Coste Total.

## *8.1. Coste de Materiales*

Estos costes están referidos a los materiales necesarios para la fabricación de la herramienta del manipulador. En la siguiente tabla se muestra detalladamente las características técnicas de cada componente.

| **MÁSTER DE INGENIERÍA MECATRÓNICA** | CONTROL DE LA ORIENTACIÓN PARA EL ROBOT ABB IRB120 MEDIANTE UN SENSOR INERCIAL Mti XSENS | | | | Pag 1 COSTE DE MATERIALES | |
|---|---|---|---|---|---|---|
| Nº ORDEN | CONCEPTOS | DESCRIPCIÓN | Nº UNIDADES | FABRICANTE | PRECIO UNITARIO MATERIAL | TOTAL |
| 1.1 | Perfil 30x30 | 201068/0 1x1m | 1 | MiniTec | 6,3 | 6,3 |
| 1.2 | Tapa Perfil 30x30 | 221146/1 | 7 | MiniTec | 0,59 | 4,13 |
| 1.3 | Fijación Minitec 30 | 210014/0 | 8 | MiniTec | 2,7 | 21,6 |
| 1.4 | Tuerca 30 M5 con freno | 211568/0 | 4 | MiniTec | 0,62 | 2,48 |
| 1.5 | Placa aluminio 3 mm 80x70x3 mm | - | 1 | Alustock | 3,1 | 3,1 |
| 1.6 | Redondo aluminio 50x40 mm | - | 1 | Alustock | 6,5 | 6,5 |
| 1.7 | TTE transporte | - | 1 | TNT | 9 | 9 |
| **COSTE TOTAL DE MATERIALES** | | | | | | **53,11** |

*Tabla 7.1: Coste de materiales.*

## *8.2. Coste de Amortización de Equipos*

El coste de amortización hace referencia al proceso financiero por medio del cual se reduce una deuda. En el presente proyecto, este tipo de coste aparece en el empleo de los equipos que permiten el desarrollo del sistema. Con el fin de determinar el nivel de utilización de todos los equipos, se ha aplicado un coeficiente de amortización que responde a la siguiente ecuación:

$$Coef.\ Amortización = Tiempo\ de\ Proyecto / Tiempo\ estimado\ de\ amortización$$

En la siguiente tabla aparecen detallados dichos costes.

| **MÁSTER DE INGENIERÍA MECATRÓNICA** | CONTROL DE LA ORIENTACIÓN PARA EL ROBOT ABB IRB120 MEDIANTE UN SENSOR INERCIAL Mti XSENS | | | Pag 2<br>COSTE DE AMORTIZACIÓN DEL EQUIPO | | |
|---|---|---|---|---|---|---|
| Nº ORDEN | CONCEPTOS | DESCRIPCIÓN | FABRICANTE | PRECIO(€) | COEF. AMORTIZACIÓN | TOTAL |
| 2.1 | Hardware | Robot IRB120 | ABB ROBOTICS | 10.954 | 0,1 | 1095,4 |
| 2.2 | Hardware | Ordenador Portátil | ASUS | 559 | 0,09 | 50,31 |
| 2.3 | Hardware | IMU Mti | Xsens | 2.528,50 | 0,06 | 151,71 |
| 2.4 | Software | Matlab R2013b | MathWorks | 6.000 | 0,08 | 480 |
| 2.5 | Software | Robotware & RobotStudio 15.02 | ABB ROBOTICS | - | | |
| 2.6 | Software | MT Manager 2010 | Xsens | - | | |
| 2.7 | Software | SolidWorks 2014 | SolidWorks Corp | 6.500 | 0,05 | 325 |
| 2.7 | Software | Office Profesional 2010 | Microsoft | 539 | 0,07 | 37,73 |
| **COSTE TOTAL DE AMORTIZACIÓN DEL EQUIPO** | | | | | | **2140,15** |

*Tabla 7.2: Coste de amortización de equipos.*

## *8.3. Coste de Mano de Obra*

En este apartado se desarrollarán los costes derivados de la implementación del proyecto, es decir, todos aquellos costes producidos por el trabajo físico y mental de cada unos de los procesos del mismo. A continuación, se presenta una tabla en la que se detallan dichos costes.

| **MÁSTER DE INGENIERÍA MECATRÓNICA** | CONTROL DE LA ORIENTACIÓN PARA EL ROBOT ABB IRB120 MEDIANTE UN SENSOR INERCIAL Mti XSENS | Pag 3 COSTE DE MANO DE OBRA | | |
|---|---|---|---|---|
| **Organización del Proyecto** | | | | |
| Nº ORDEN | CONCEPTOS | PRECIO(€/h) | HORAS | TOTAL |
| 3.1 | Aprendizaje del manejo del Robot | 30 | 100 | 3000 |
| 3.2 | Planificación | 30 | 25 | 750 |
| 3.3 | Documentación | 30 | 150 | 4500 |
| **Diseño Mecánico y Fabricación** | | | | |
| Nº ORDEN | CONCEPTOS | PRECIO(€/h) | HORAS | TOTAL |
| 3.4 | Diseño | 30 | 30 | 900 |
| 3.5 | Modelado 3D | 30 | 25 | 750 |
| 3.6 | Fabricación y Montaje | 30 | 7 | 210 |
| **Diseño de Software** | | | | |
| 3.7 | Diseño del software del controlador | 30 | 55 | 1650 |
| 3.8 | Diseño del software de captura de datos | 30 | 50 | 1500 |
| 3.9 | Comunicación OPC | 30 | 30 | 900 |
| 3.10 | Diseño Software de Calibración | 30 | 50 | 1500 |
| **Test** | | | | |
| 3.11 | Test Controlador | 30 | 20 | 600 |
| 3.12 | Test captura de datos | 30 | 23 | 690 |
| 3.13 | Test Comunicación | 30 | 10 | 300 |
| 3.14 | Test Sistema completo | 30 | 40 | 1200 |
| **Coste Total de Mano de Obra** | | | | **18450** |

*Tabla 7.3: Coste de mano de obra.*

## *8.4. Coste Total*

Finalmente, una vez sumados los costes anteriormente calculados, es necesario aplicar el beneficio industrial y el impuesto sobre el valor añadido (IVA).

| **MÁSTER DE INGENIERÍA MECATRÓNICA** | CONTROL DE LA ORIENTACIÓN PARA EL ROBOT ABB IRB120 MEDIANTE UN SENSOR INERCIAL Mti XSENS | Pag 4<br>COSTE TOTAL |
|---|---|---|

| CONCEPTOS | TOTAL |
|---|---|
| Coste de Materiales | 53,11 |
| Coste de Amortización de Equipos | 2140,15 |
| Coste de Mano de Obra | 18450 |
| **Coste Total de Ejecución del Proyecto** | 20643,26 |
| **Beneficio Industrial (10%)** | 2064,33 |
| **IVA (21%)** | 4768,59 |

| **COSTE TOTAL** | **27476,18** |
|---|---|

*Tabla 7.1: Coste de total.*

El precio Total de proyecto asciende a:

**VEINTISIETEMIL CUATROCIENTOS SETENTA Y SEIS CON DIECIOCHO.**

# 9. BIBLIOGRAFÍA

## *9.1. Manuales*

[1] ABB ROBOTICS. *IRB 120 Datasheet*. 2014.

[2] ABB ROBOTICS. *Especificaciones del producto Controller IRC5 con FlexPendant*. 2014.

[3] ABB ROBOTICS. *Especificaciones del producto IRB120*. 2014.

[4] ABB ROBOTICS. *Instrucciones, Funciones y tipos de datos en Rapid*. 2014.

[5] ABB ROBOTICS. *Manual del operador. RobotStudio*. 2014.

[6] ABB ROBOTICS. *Manual del operador, Introducción a Rapid*. 2014.

[7] ABB. *Manual de referencia técnica de Rapid*. 2014.

[8] ABB ROBOTICS. *Instalación de Manual del producto IRB120*. 2014.

[9] ABB ROBOTICS. *Release Notes OPC Server 6.00.01*. 2014.

[10] ABB ROBOTICS. *Application manual, IRC5 OPC Server help*. 2014.

[11] Mathworks. *Matlab OPC Toolbox Help*. 2013.

[12] Xsens. *MT Manager User Manual*. 2010.

[13] Xsens. *Mti User Manual and Technical Documentation*. 2010.

[14] D. Álvarez. *InerSens Toolbox.* Departamento de Ingeniería de Sistemas y Automática. Universidad de Oviedo. 2009.

## *9.2. Libros*

[15] A. Herreros, E. Baeyens. *Curso de Programación en Matlab*. Departamento de Ingeniería de Sistemas y Automática. Universidad de Valladolid. 2011.

[16] J.Craig. *Robótica*. Pearson Education, 3ª Ed, 2006.

## *9.3. Artículos*

[17] P. Cardou, G. Fournier, P. Gagnon. A Nonlinear Program for Angular-VelocityEstimation From Centripetal-Acceleration Measurements. IEEE/ASME Transactions on Mechatronics. Vol 16. 2011.

## *9.4. Webs*

[18] *Automation and Power Techologies*. En: ABB ROBOTICS. Disponible vía DIALOG. https://forums.robotstudio.com/.

[19] *Actualidad y recursos sobre automatización*. En: Automatización Industrial. Disponible vía DIALOG. http://www.infoplc.net/.

[20] J.C.M Castillo, *Revista de electricidad, Electrónica y Automática.* En: Revista de electricidad, Electrónica y Automática. Disponible vía DIALOG. http://reea-blog.blogspot.com.es/.