Universidad de Valladolid


UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

# Diseño de una estación con robots ABB colaborativos para simular servicios asistenciales.

Autor:

San José Cantalejo, Rubén

Tutor:

Herreros López, Alberto
Departamento de Ingeniería de Sistemas y Automática.

Valladolid, abril 2024.

## Resumen

La finalidad de este proyecto consiste en programar un robot colaborativo que permita simular servicios asistenciales y la coordinación de otros dos robots encargados de la preparación del servicio. Para ello, se ha empleado el software de ABB, RobotStudio, que permite la simulación y programación de estos robots.

Para la interacción entre la persona y el robot se ha diseñado una interfaz gráfica de usuario en AppDesigner de MATLAB que a través de un protocolo de comunicación OPC UA permite el intercambio de información con RobotStudio.

## Palabras clave

RobotStudio, CRB15000 GoFa, IRB 1200, OPC UA, simulación.

## Summary

The purpose of this project is to program a collaborative robot that allows the simulation of care services and the coordination of two other robots in charge of preparing the service. To do this, ABB's software, RobotStudio, has been used, which allows the simulation and programming of these robots.

For the interaction between the person and the robot, a graphic user interface has been designed in MATLAB AppDesigner thar allows the exchange of information with RobotStudio through an OPC UA Communication protocol.

## Key words

RobotStudio, CRB15000 GoFa, IRB 1200, OPC UA, simulation.

# ÍNDICE

# ÍNDICE ILUSTRACIONES

# INDICE DE TABLAS

# 1 Introducción

A lo largo de este informe, se procederá a describir y detallar la gestación de la estación mediante la implementación de robots colaborativos, con la intención de exhibir ciertas posibilidades inherentes al campo de la robótica.

Con la progresiva evolución de la sociedad hacia un entorno cada vez más tecnológico, la proliferación de robots y dispositivos automáticos que facilitan las actividades cotidianas se torna cada vez más común. Este ámbito se presenta como un terreno vasto, repleto de oportunidades aún por explorar.

El aumento en la relevancia de la robótica colaborativa en los últimos años encuentra su razón principal en el éxito de sus aplicaciones en la industria. Estas ventajas no se limitan únicamente a dicho sector, sino que también hallan aplicaciones en servicios asistenciales, como el enfoque desarrollado en este trabajo. Este aprovecha las capacidades de un robot colaborativo para llevar a cabo tareas que pueden simplificar la vida diaria.

## 1.1 Objetivos

El objetivo principal de este proyecto consiste en desarrollar una solución para facilitar la alimentación de personas con movilidad reducida, ampliando nuestros conocimientos en el software RobotStudio.

Este software se utiliza para crear de forma virtual el modelo de estación. En este entorno, se diseñan las herramientas, mecanismos y objetos necesarios para poder llevar a cabo la simulación. Además, se programan los movimientos e instrucciones a realizar por el robot colaborativo, el cual, es el encargado de servir a la persona. Las decisiones de esta, ya sea comer o beber, se simulan a través de la interfaz de usuario desarrollada en MATLAB y por ello es necesario establecer un protocolo de comunicación, en este caso, OPC UA.

Otro objetivo fundamental ha sido la aplicación de conocimientos adquiridos en el grado de Electrónica Industrial y Automática de la Universidad de Valladolid con un enfoque especial en el ámbito de la robótica.

En conclusión, con lo explicado anteriormente, se obtiene de forma más detallada los objetivos expuestos:

- Diseñar una estación en el entorno de simulación RobotStudio con los diferentes robots y los componentes necesarios para llevar a cabo los servicios asistenciales.

- Usando el lenguaje RAPID programar el código conveniente para crear los movimientos e instrucciones a realizar por los robots.
- Utilizar el protocolo OPC UA para establecer una conexión entre RobotStudio y MATLAB, con el objetivo de simular la interacción persona-robot.
- En MATLAB, desarrollar con App Designer una interfaz Hombre-Máquina en la que se controle toda la simulación de forma más sencilla.

Para la realización de este proyecto ha sido necesario emplear de la empresa tecnológica ABB, tres robots. Dos ABB IRB 1200 (Figura 1) y un ABB CRB 15000 GoFa (Figura 2).


*Ilustración 1. Robot ABB IRB 1200*


*Ilustración 2. Robot ABB CRB 15000 GoFa*

## 1.2 Motivación

El mundo de la robótica y todo lo que comprende a su alrededor en la ingeniería, así como la informática, física, mecánica, control, etc.., los cuales, son conocimientos adquiridos durante el grado cursado y tener la oportunidad de demostrarlos.

Otro factor que aumentó la motivación hacia este proyecto es conseguir facilitar la vida a aquellas personas que tienen dificultades para moverse y, con las tecnologías de hoy en día ver las distintas posibilidades.

Gracias a las motivaciones mencionadas anteriormente y al gusto hacia la robótica han sido los factores determinantes para desarrollar este Trabajo de Fin de Grado.

## 1.3 Estructura de la memoria

El contenido de este proyecto se abordará primeramente con una pequeña introducción y con los objetivos necesarios para desarrollar este trabajo. Acto seguido, se realizará una breve introducción al mundo de la robótica, profundizaremos en el entorno de los robots colaborativos y se hablará sobre los diferentes tipos de robots que existen hoy en día.

Después, será el turno de explicar el proyecto desarrollado y los distintos softwares que se han utilizado para llevarlo a cabo. A continuación, se pondrá en conocimiento cómo se ha modelado la estación y se explicarán los distintos componentes que la componen, tanto los robots como mecanismos o componentes inteligentes.

Seguidamente, se explicará cómo se ha elaborado la programación de los robots, la comunicación entre RobotStudio y MATLAB y el desarrollo de la interfaz Hombre-Máquina.

Por último, al finalizar la memoria se incluirá el funcionamiento de la aplicación con las correspondientes conclusiones y en la parte final de la memoria, se encontrará la bibliografía y distintos anexos.

# 2 Marco teórico y estado del arte

A lo largo de este capítulo se mostrará información acerca del mundo de la robótica, el cual, es un mundo que no para de crecer exponencialmente en todas las industrias debido a que su uso en asistencia y tareas de servicio provoca la ampliación de los campos de aplicación que abarcan desde la capacidad de realizar servicios asistenciales hasta complejas incursiones espaciales.

## 2.1 Origen de la robótica

El origen de la robótica como la conocemos hoy en día es una obra relativamente nueva de la tecnología y la ciencia. Sin embargo, sus raíces se remontan a la antigüedad [1] [2].

Antecedentes históricos, los primeros conceptos del mundo que conocemos como robótica se remontan a la antigüedad, cuando se inventaron los primeros autómatas [3]. Uno de los mayores creadores de autómatas en la antigua Grecia fue Herón de Alejandría quien construyó autómatas "teatrales" /" mágicos" movidos por mecanismos de agua, aire y vapor, gracias a ello se ganó el apodo "El mago" ya que a través de sus creaciones se reía de la gente con transformaciones de agua en vino o con uno de sus mayores hitos como fue que las puertas de los templos se abriesen solas.

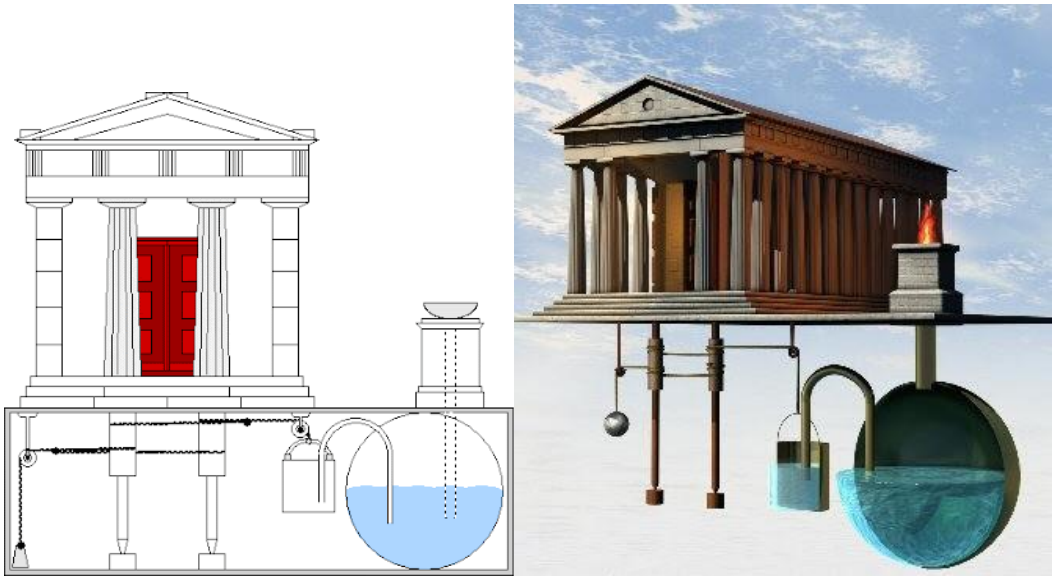


*Ilustración 3. Autómatas de Herón*

En el antiguo Egipto aprendieron a crear estatuas con movimiento que podían hacer distintas tareas como verter agua, estos pequeños autómatas fueron

considerados los primeros pasos de la robótica, debido a que su principal diseño era para hacer más fácil la vida de la gente.

Como su nombre indica, la Revolución Industrial fue el verdadero nacimiento de la robótica moderna donde se desarrolló maquinaria industrial que permitía realizar repetitivas tareas con eficiencia y precisión. Aunque no eran los robots que conocemos hoy en día, fueron las bases para el desarrollo de robótica que tenemos.

Gran ejemplo de ello fue el telar de Jacquard [4] creado en 1801, que usaba tarjetas perforadas para controlar el patrón de tejido. Este invento asentó las bases para la automatización y la programación en la historia de las máquinas, durante esta época también se desarrollaron conceptos como la línea de montaje y la automatización de la producción, conceptos fundamentales para la robótica actual.

Solo unos años más tarde Henri Maillardert creó la primera muñeca mecánica que era capaz de hacer dibujos.

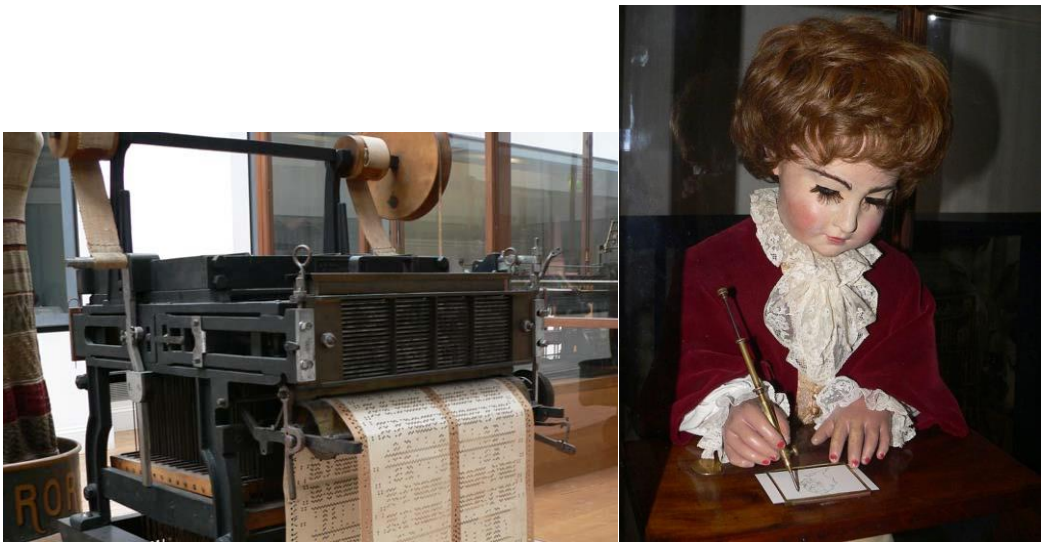


*Ilustración 4. El telar de Jacquard y la muñeca mecánica de Henri-Maillardert*

Aun habiendo sucedido todo lo anterior hasta los años 20 no aparece por primera vez la palabra “Robot”. Fue en la obra de teatro del checo Karel Capek “R.U.R.” (Rossum´s Universal Robots) que describe a los robots como trabajadores artificiales y dicha palabra proviene del término checo “robota” que significa servidumbre o trabajo forzado.

En los años 40, Isaac Asimov un prolífico escritor de ciencia ficción definió las “Tres leyes de la Robótica” [5] que se han convertido en la piedra angular en el campo de la robótica y la inteligencia artificial. Aparecieron en su relato “Runaround” en 1942 y son las siguientes:

- **Primera Ley:** Un robot no hará daño a un ser humano ni, por inacción, permitirá que un ser humano sufra daño.
- **Segunda ley**: Un robot debe cumplir las órdenes dadas por los seres humanos, a excepción de aquellas que entren en conflicto con la primera ley.
- **Tercera ley**: Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la primera o con la segunda ley.

Estas leyes están diseñadas para garantizar la seguridad de los humanos en su interacción con los robots. Además, se introdujo una "Ley Cero" que precede a las demás "Un robot no puede dañar a la humanidad o, por inacción, permitir que la humanidad sufra daños". Siendo esta ley para las situaciones en las que la acción o inacción de un robot puede afectar a la humanidad en su conjunto.

A partir de este momento, comienza una nueva época para la robótica. George Devol en 1954 fabricó el primer robot industrial, más conocido como "Unimate". Este robot fue instalado en la fábrica de General Motors en 1961, marcando el comienzo de la era de la robótica industrial. George Devol junto a Joseph F. Engelberger fundaron la empresa "Unimation" y desarrollaron el primer robot de transferencia programable, considerado el hijo del robot industrial tal como lo conocemos hoy en día.

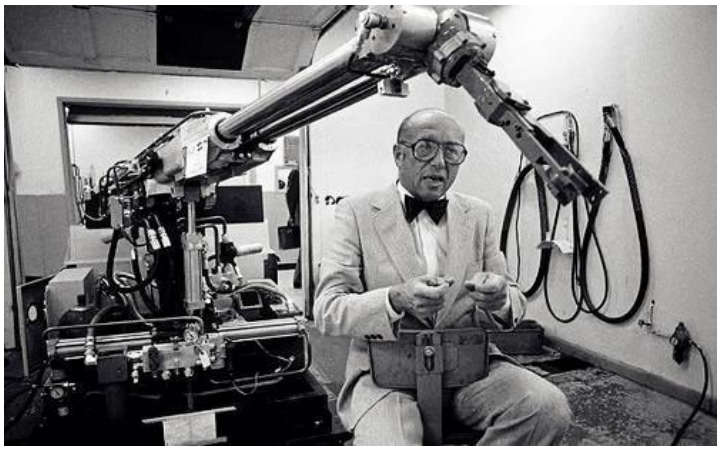


*Ilustración 5. Joseph F. Engelberger junto a una de sus creaciones.*

En los años 70 se nota un gran crecimiento de la robótica, debido a ello aparecen las primeras asociaciones como "RIA" (Asociación de Robótica Industrial) en Estados unidos o en Japón "JIRA". Empezó a usarse la robótica para las armas inteligentes y en la munición de precisión.

En 1973, Víctor Scheinman inventó el primer brazo robótico programable y se convirtió en la base de la mayoría de los robots industriales modernos, en ese mismo año la prestigiosa empresa japonesa FANUC desarrolló el primer robot

controlado por un microordenador, lo que permitió mayor precisión y control en la programación.

No fue hasta los años 80 cuando en 1985 se creó el primer robot médico que se utilizaba para realizar cirugías no invasivas, PUMA 560.

En 1997, el Mars Pathfinder de la NASA aterrizó en Marte con un pequeño robot llamado Sojourner, el cual, fue el primer rover en explorar otro planeta. Unos años más tarde se desarrollaría el primer robot Humanoide por Honda, ASIMO.

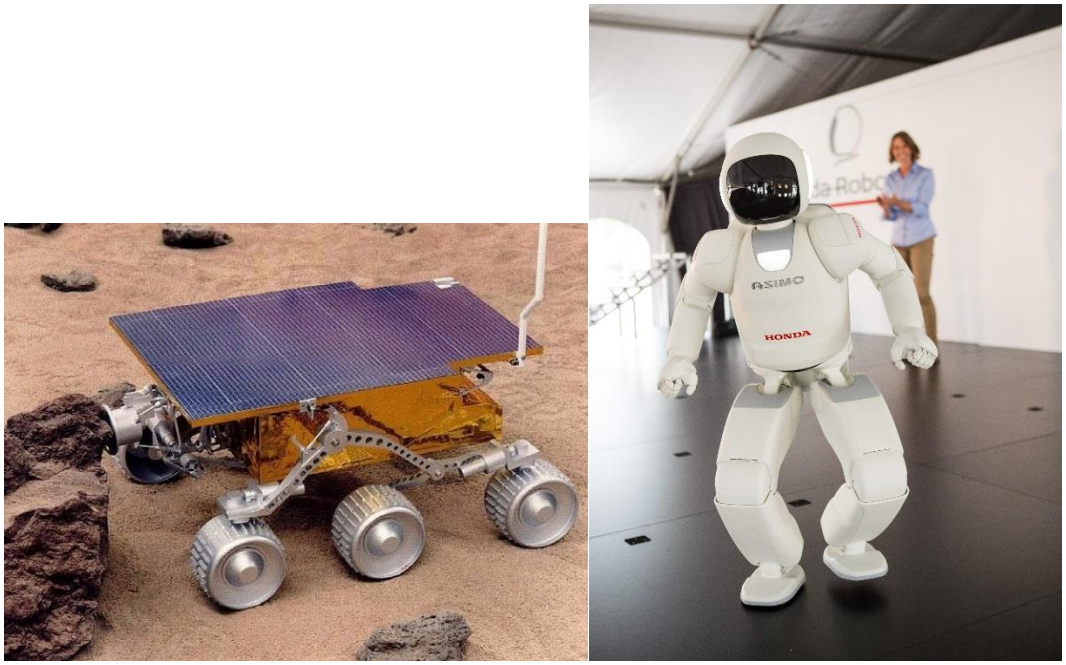



*Ilustración 6. A la izq. Sojourner primer rover espacial, a la drcha. ASIMO el primer robot Humanoide por Honda*

## 2.2 Desarrollo de la robótica colaborativa

Una vez introducida la robótica desde nuestros orígenes, se comentará el origen de los robots colaborativos o cobots que será de utilidad para entender este trabajo de fin de grado.

A diferencia de los robots industriales que suelen estar aislados de los trabajadores por razones de seguridad, los robots colaborativos permiten la interacción con los humanos en entornos compartidos.

Su origen gira en torno a 1996 pero la primera idea de que un robot pudiese trabajar con humanos viene de varias décadas hacia atrás en las líneas de montaje de vehículos de los años 80 pero estos robots no tenían la capacidad de interactuar con las personas de manera segura. Los cobots representan una

nueva generación de maquinaria que incorporan sensores de fuerza para garantizar la seguridad con los humanos.

Fue en 2008 cuando la empresa danesa Universal Robots vendió el primer robot industrial capaz de trabajar con personas de forma segura, el UR5.

*Ilustración 7. Robot colaborativo UR5*

## 2.3 Clasificación y definición de los tipos de robots

Después de examinar el inicio y evolución del mundo de la robótica se continuará viendo los distintos tipos de robots [6] y su clasificación en varias formas.

Según su cronología:

- Primera generación: robots manipuladores, los cuales, pueden coger y mover objetos, pero los movimientos los tienen muy limitados y se dedican a realizar una o varias tareas repetitivas.
- Segunda generación: son robots en aprendizaje que recogen la información del entorno para poder hacer movimientos complejos, primeros en incluir la retroalimentación.
- Tercera generación: robots reprogramables equipados con sensores que usan lenguajes de programación para varias sus funciones.
- Cuarta generación: corresponde con los primeros robots inteligentes capaces de interpretar el entorno en tiempo real y tomar decisiones más complejas, son los robots móviles.

- Quinta generación: robots con inteligencia artificial que se encuentran en desarrollo ya que se pretende imitar al ser humano y que sean autónomos.

Según su movilidad:

- Robots articulados o brazos robóticos: su capacidad es muy reducida, pero son muy buenos para mover productos, manipular herramienta, empaquetar, etc.
- Vehículo de guiado automático (AGV): Se mueven por recorridos predefinidos y en la mayoría de los casos necesitan supervisión humana.
- Robots móviles autónomos (AMR): pueden tomar decisiones y moverse solos en tiempo real, suelen ser enviados para lugares de difícil acceso.
- Humanoides: suelen ser un tipo de AMR con aspecto humano y movimientos parecidos a los humanos.
- Zoomórficos: se busca modelar el sistema de locomoción de diferentes seres vivos.
- Robots colaborativos o cobots: son usados para trabajar mano a manos con las personas sin peligro.

*Ilustración 8. Al izq. robot zoomórfico y a la drcha. robot AGV*

Según su función:

- Robots industriales: con una visión directa a las cadenas de producción para realizar tareas rutinarias y repetitivas.
- Robots militares: sirven de apoyo a los ejércitos para distintas operaciones aportando mayor precisión o correr menos peligro para las propias vidas de los militares en la detección de material explosivo.

- Robots de servicios: son robots que realizan trabajos peligrosos, sucios o repetitivos ayudando al ser humano en medicina, en restauración o en aspectos domésticos entre otros.
- Robots educativos o de investigación: están destinados al aprendizaje o desarrollo cognitivo de un tema de cara a facilitar el aprendizaje del usuario.
- Robots médicos: aportan mayor precisión a los médicos en operaciones quirúrgicas u ofrecer una ayuda a las personas con movilidad reducida entre otras cosas.
- Robots domésticos: dedicados a las tareas del hogar principalmente, también en la vigilancia tienen un papel importante.

De cara a los robots que se van a emplear en este proyecto, también se encuentra la clasificación correspondiente entre robots colaborativos o cobots y no colaborativos que normalmente son conocidos como robots industriales para ello se va a enumerar las diferencias en los siguientes aspectos [7].

| | No colaborativos | Colaborativos |
|---|---|---|
| Tamaño | Gran tamaño, es un problema para organizar una cadena de producción. | Son pequeños, por lo que son menos condicionantes en cuanto a espacio. |
| Zona de trabajo | Suelen estar fijos y necesitan más espacio para ejecutar sus tareas. | Se pueden trasladar con gran facilidad y no requieren mucho espacio de trabajo para hacerlo de manera eficaz. |
| Dificultad de programación | Máquinas complejas por lo que requieren expertos para su manejo. | Están diseñados para ser usados fácilmente de manera muy intuitiva. |
| Procesos de producción | Ideales para grandes producciones y movimientos pesados. | Se emplean para producciones más específicas ya que no soportan mucha carga. |
| Rentabilidad e implantación | Menos económicos y para amortizarlo se requiere más tiempo. | Son más económicos y el ROI es mucho menor. |

*Tabla 1. Comparativa entre robots colaborativos y no colaborativos.*

## 2.4 Medios de detección

En el presente proyecto se estudia la forma de poder implementar de manera virtual en RobotStudio como servir servicios asistenciales, para ello es necesario poder detectar los rostros de las personas. Debido a que en la Universidad de Valladolid no se dispone actualmente de los medios necesarios para poder recrear la situación, se realiza de manera virtual a través del protocolo de comunicación OPC UA.

Pero se ha de saber que existen varias formas en las que un robot puede detectar el movimiento de la cara de una persona [8]. Consiste en el uso de un robot con una cámara que se encuentra ejecutando continuamente un algoritmo de reconocimiento facial para comprobar si hay una persona delante o no, debido a que este procedimiento sobrecarga el servidor al estar continuamente enviando fotos y con el objetivo de reducir el ancho de banda, solo se envían imágenes cuando se encuentra una persona.

Normalmente los softwares de reconocimiento facial se simulan en la nube porque si no se sobrecargaría la CPU del robot con los correspondientes cálculos y aparte, se necesita mucha información de caras distintas para poder llevar a cabo el algoritmo. Hoy en día, entre los mejores proveedores de este software se encuentra, SenseTimeSenseStudio, Microsoft Azure Cognitive Services, Amazon Rekognition y Face++, entre otras [9].

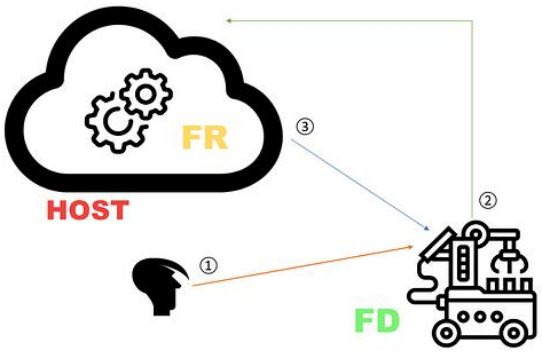



*Ilustración 9. Representación del modo de actuación de los softwares de reconocimiento facial.*

En cuanto a los dispositivos que se emplearían se enumeran las siguientes:

- **Seguimiento de movimiento con dispositivos ópticos**: son varios los proyectos como Opentrack [10] y Altrack que permiten rastrear el movimiento de la cabeza de una persona utilizando un dispositivo móvil que usa tecnología óptica que es útil para la realidad virtual, juegos, etc. Para el presente proyecto podría ser la más idónea.

- **Detección de movimiento con cámara Web**: existen varios proyectos en los que se ha desarrollado este prototipo como en el laboratorio de ciencias de la computación e inteligencia artificial del MIT donde fueron capaces de determinar con precisión el pulso cardiaco analizando el movimiento de sus cabezas.

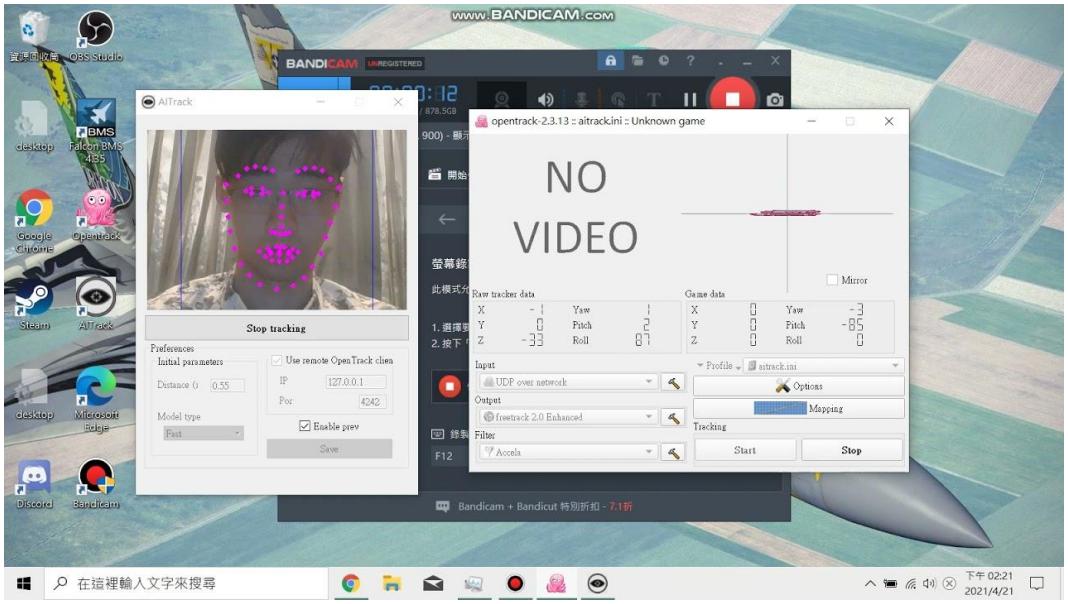



*Ilustración 10. Ejemplo de reconocimiento facial*

## 2.5 Estado del arte

Después de haber introducido el marco teórico en el que se ha desarrollado este proyecto, se procede a mostrar algunos proyectos realizados por antiguos alumnos de la Universidad de Valladolid relacionados con este proyecto.

En el año 2015, Juan Antonio Ávila Herrero [11] completó su proyecto de fin de grado al idear una célula robótica orientada a propósitos educativos. Esta célula, concebida para la aplicación práctica tanto en su forma física como en la simulada permitía a los estudiantes llevar a cabo distintas prácticas de robótica. Durante el diseño, se incorporaron diversos elementos, tales como una pinza destinada a la manipulación de objetos, una botonera para gestionar señales de entrada y salida, una caja equipada con rotuladores para la creación de dibujos, y dos mesas habilitadas para este mismo propósito.

*Ilustración 11. Célula de trabajo del TFG de Juan Antonio Ávila Herrero*

Al siguiente año (2016), Álvaro Galindo de Santos [12] completó su proyecto de fin de grado al modelar la célula robotizada presente en el departamento. Implementó una comunicación vía sockets entre el robot y una Tablet, permitiendo la ejecución de juegos previamente creados en proyectos anteriores, todos controlados de manera interactiva por el usuario.

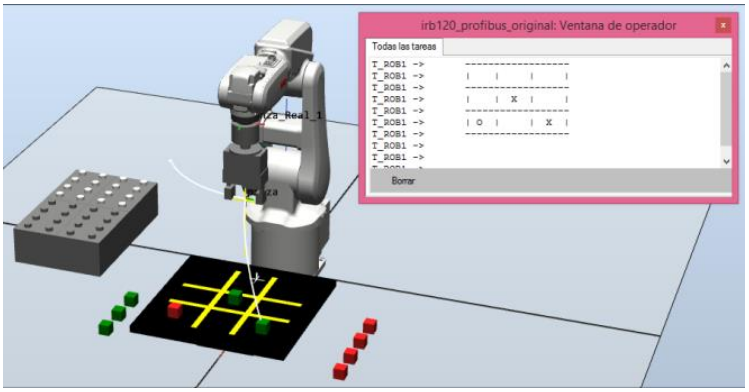



*Ilustración 12. Célula de trabajo del TFG de Álvaro Galindo de Santos*

En el año 2019, un estudiante de posgrado llamado Carlos Jiménez Jiménez [13] llevó a cabo un proyecto innovador para su tesis de máster. Creó un sistema robótico con fines educativos que permitía a los usuarios jugar al ajedrez contra un robot industrial. Los usuarios podían definir sus movimientos a través de una interfaz especialmente diseñada, y el robot se encargaba de mover las piezas en el tablero. Para facilitar la comunicación entre el usuario y el robot, se utilizó el protocolo TCP/IP. Este sistema podía funcionar tanto en un entorno simulado como con un robot real.

*Ilustración 13.Célula de trabajo del TFG de Carlos Jiménez Jiménez*

En 2022, Elena Pozas Mata [14] diseñó un proyecto innovador para su trabajo de fin de grado, en el cual, desarrolló la programación de un robot colaborativo YuMi ABB que tenía el objetivo de poder tocar el xilófono. El robot era capaz de interpretar las notas musicales que el usuario seleccionaba a través de una interfaz gráfica creada con App Designer.

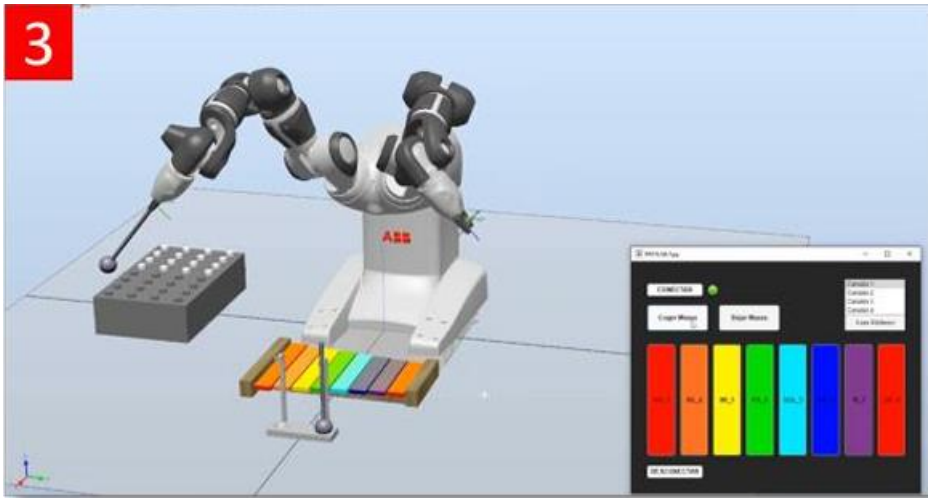



*Ilustración 14. Célula de trabajo del TFG de Elena Pozas Mata*

# 3 Metodología y software utilizado

Una vez realizada la introducción sobre la robótica y los diferentes proyectos realizados en la Universidad de Valladolid, es el turno de hablar de este trabajo de fin de grado en el que se ha buscado abarcar varios campos de la ingeniería que se han ido viendo a lo largo del grado y encontrar la forma de implementarlos de manera innovadora.

De una forma ilustrativa, observamos los campos que abarca el proyecto:

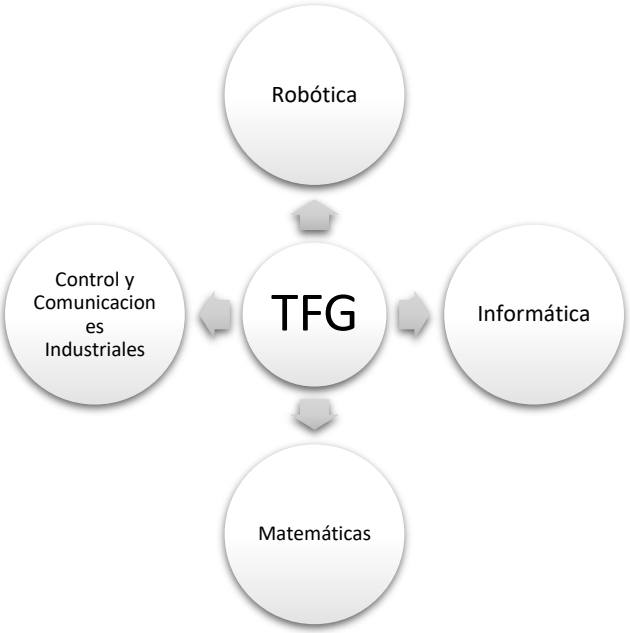



*Ilustración 15. Campos de la ingeniería que abarca el TFG.*

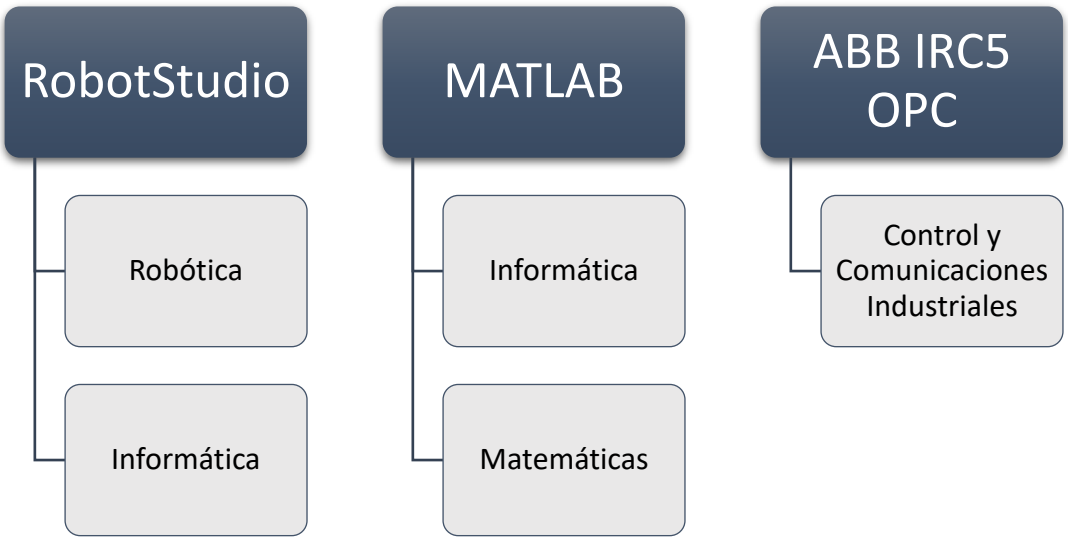



*Ilustración 16. Campos de la ingeniería que abarcan cada software.*

## 3.1 Desarrollo de los objetivos

Tras enumerar los campos sobre los que se desarrolla el proyecto, se continuará con los requisitos establecidos antes de empezar este trabajo de fin de grado, los cuales, son de buena práctica para orientar el proyecto hacia ellos. Los requisitos que ha considerado el autor son los siguientes:

- Buscar una solución para las personas con movilidad reducida a través del mundo de la robótica colaborativa.
- Coordinar y sincronizar varios robots con distintos controladores.
- Realizar la mayoría en lógica de estación de RobotStudio, empleando Smart Components.
- Establecer algún protocolo de comunicación industrial para poder comunicarse desde un programa externo.
- Integrar una interfaz de usuario práctica para poder controlar la simulación.

## 3.2 Software

### 3.2.1 RobotStudio

Es un software innovador desarrollado por ABB, una de las empresas líderes en tecnología de automatización industrial. RobotStudio permite diseñar, simular y probar soluciones robóticas en un entorno virtual antes de implementarlas en el mundo real.

*Ilustración 17. Logo RobotStudio 2023*

Su software se basa en la tecnología ABB Virtual Controller que es una réplica del controlador real de los robots ABB, lo que nos permite realizar las mismas tareas que en el mundo real.

Además, ofrece una interfaz de usuario muy intuitiva y fácil de usar que permite implementar fácilmente modelos creados con CAD y simular la estación real.

Cabe destacar el potente lenguaje de programación que tiene RobotStudio que sirve para programar tareas, movimientos, instrucciones, manejo automático de errores, multitarea, entre otras cosas, dicho lenguaje es conocido como RAPID.

Gracias a la Universidad de Valladolid se ha podido realizar el presente proyecto con este software.

### 3.2.2 MATLAB

Es un entorno de programación de alto nivel y con un lenguaje que permite a los usuarios realizar cálculos con facilidad. MATLAB es muy utilizado en la vida académica y en la industria para el análisis de datos, desarrollo de algoritmos, creación de modelos o la simulación. Fue desarrollado por MathWorks.

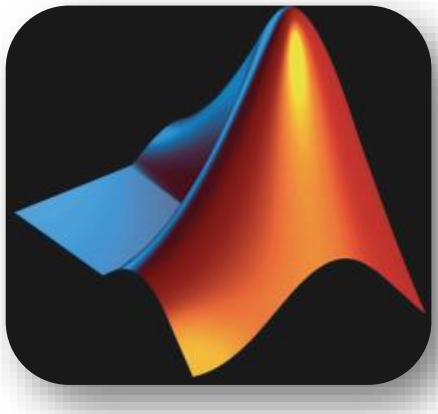


*Ilustración 18. Logo MATLAB 2023*

Además de sus capacidades de cálculo numérico también ofrece una variedad de herramientas para el análisis de datos, visualización y programación gráfica, entre las cuales se encuentra App Designer, una interfaz de diseño de aplicaciones interactiva que se empleará en este proyecto.

App Designer es una herramienta de diseño de interfaz gráfica de usuario (GUI) que permite a los usuarios crear aplicaciones de MATLAB de manera bastante intuitiva sin la necesidad de tener unos conocimientos muy elevados de programación gráfica.

En App Designer se distinguen dos apartados, uno para cada tarea:

- Un apartado para la distribución de los componentes visuales de la interfaz a crear por el usuario.

- Otro apartado en el que se encuentra la programación de la interfaz.

Otra herramienta que es de gran utilidad en este proyecto es 'OPC ToolBox' con la que podremos establecer la conexión OPC UA con la comunicación entre RobotStudio y MATLAB.

### 3.2.3 ABB IRC5 OPC

En los capítulos anteriores se ha comentado que para el presente proyecto se necesitaba establecer una comunicación entre la persona y el robot, simulando la forma de detección de movimiento pertinente. Para ello se empleó este software de ABB.

Este programa permite establecer comunicaciones entre RobotStudio y MATLAB. Antes de nada, se procederá a explicar que es OPC [18] (Open protocol Communication), es una tecnología que tiene una arquitectura cliente y servidor que permite la interoperabilidad entre distintos dispositivos de software y hardware. Podría simplificarse a algo más coloquial como que OPC realiza la función de traductor entre diferentes dispositivos permitiendo que se comuniquen entre sí de manera segura y operativa en diferentes secciones de la industria, principalmente en la automatización industrial.

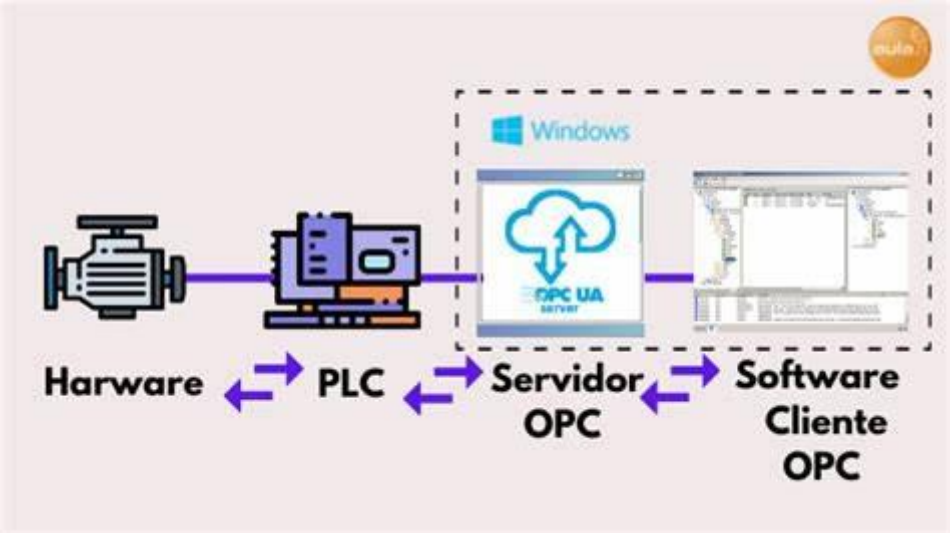



*Ilustración 19. Representación gráfica del funcionamiento OPC*

OPC emplea tecnologías propias de Microsoft como son OLE, COM y DCOM, pero en la versión más reciente, la cual se emplea en este proyecto, es independiente de Microsoft y es capaz de funcionar en otros sistemas operativos como Linux o MacOS, esta es conocida como OPC UA.

OPC UA al igual que se ha comentado con los otros tipos de OPC, también se basa en cliente-servidor que reemplaza el protocolo COM y DCOM por protocolos abiertos e independientes permitiendo su funcionamiento bajo otros S.O., sus principales características son:

- Funciona en cualquier sistema operativo.
- Permite la conexión con sistemas antiguos.
- Fácil mantenimiento y configuración.
- Mayor comunicación entre Firewall (no se dispone de DCOM).
- Mayor seguridad debido a la encriptación de los datos.
- Mayor seguridad
- Interoperabilidad.

En cuanto a la aplicación ABB IRC5 OPC se emplea para crear y organizar alias para los distintos controladores de ABB tanto para IRC5 como para Omnicore como se observará más adelante.

# 4 Diseño y desarrollo de la estación robótica

En el siguiente capítulo se describe más detalladamente los distintos softwares y como se ha desarrollado cada parte en cada uno para este proyecto, empezando por el diseño de la propia estación y terminando por la comunicación OPC UA entre RobotStudio y la persona (MATLAB).

## 4.1 Programación RobotStudio

RobotStudio se dividirá en dos apartados, primeramente, se comentará todo lo que compone la estación, el diseño de componentes inteligentes, mecanismos, entre otros. En la segunda parte, se centrará la atención en el potente lenguaje de programación que dispone RobotStudio, conocido como RAPID, donde se llevará a cabo un análisis de todas las instrucciones necesarias para realizar el presente proyecto.

### 4.1.1 Modelado de la estación

Una vez instalado el software en nuestro dispositivo para poder empezar a trabajar en él, se debe crear una estación. RobotStudio [19] nos ofrece varias opciones para ello, la primera es crear una “Estación vacía”, la segunda opción es crear un “Proyecto con estación y controlador virtual” y por último crear un “Proyecto con estación vacía”. En este caso se optó por crear una estación vacía.

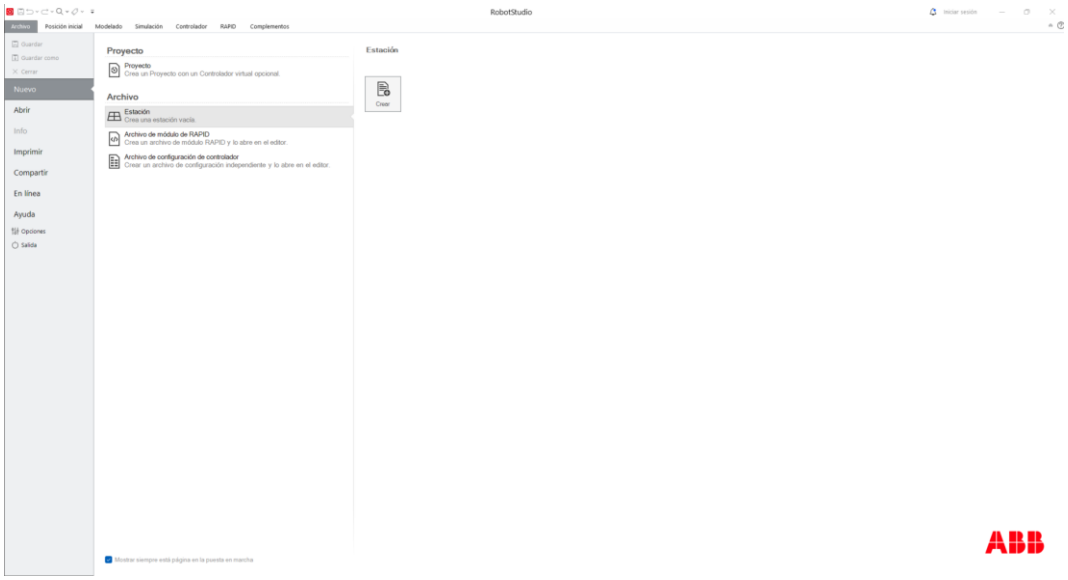



*Ilustración 20. Crear estación virtual.*

Creada la estación, el siguiente paso corresponde con la creación de los robots y sus respectivos controladores. Se accede a archivo, biblioteca ABB, lugar

donde se encuentran todos los robots que pueden ser simulados por ABB, posicionadores y tracks disponibles por RobotStudio.

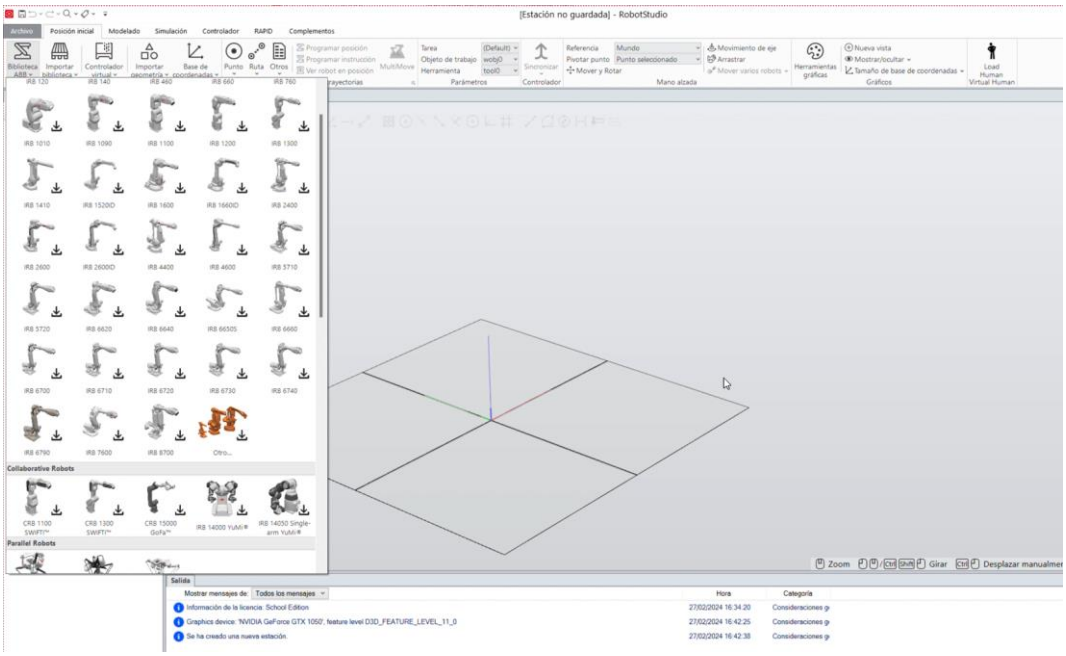


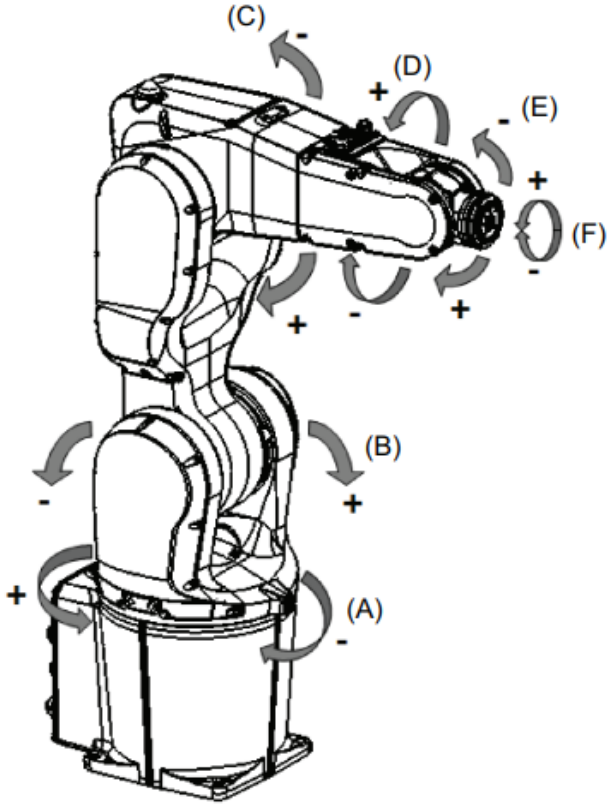

*Ilustración 21. Robots disponibles en RobotStudio.*

#### 4.1.1.1 IRB 1200 y Flexpendant

Para este trabajo de fin de grado se analizaron varios robots para la parte más "industrial" como podría ser la preparación de la comida, y se llegó a la conclusión que el IRB 1200 [20] era el robot más apropiado para este trabajo, debido a que es un robot compacto y ligero, cosa que facilita su integración en espacios de trabajo reducidos, posee una gran velocidad y precisión, gran versatilidad, disponen de una eficiencia energética muy buena y de la rama industrial son los más cercanos a poder colaborar con humanos en entornos de trabajo compartidos.

En las siguientes imágenes se puede observar los distintos ejes de giro que dispone y el área de trabajo que puede llegar a abarcar.

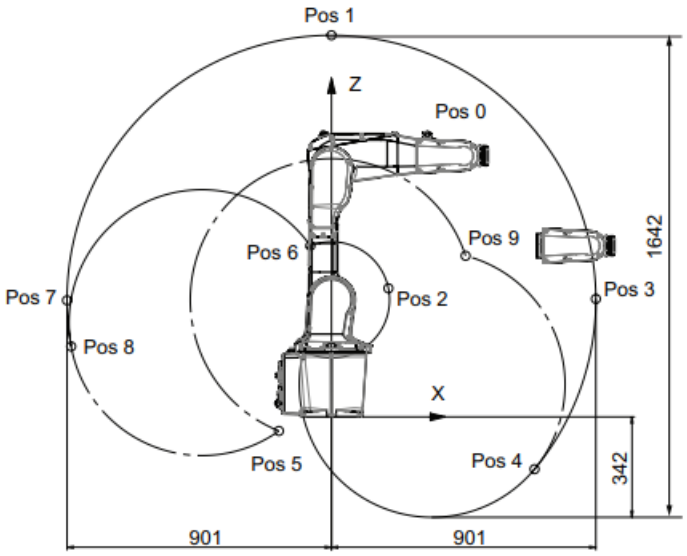
**Ejes del manipulador**




| Posición | Descripción | Posición | Descripción |
|---|---|---|---|
| A | Eje 1 | B | Eje 2 |
| C | Eje 3 | D | Eje 4 |
| E | Eje 5 | F | Eje 6 |

*Ilustración 22. Ejes de giro del ABB IRB 1200*

**Área de trabajo del IRB 1200-5/0.9 , posiciones en el centro de la muñeca y ángulo de los ejes 2 y 3**
**La figura muestra el área de trabajo sin restricciones del robot.**




| Posición en la figura | Posiciones en el centro de la muñeca (mm) | | Ángulo (grados) | |
|---|---|---|---|---|
| | X | Z | Eje 2 | Eje 3 |
| Pos0 | 451 | 889 | 0º | 0º |
| Pos1 | 0 | 1300 | 0º | -85º |
| Pos2 | 194 | 438 | 0º | +70º |
| Pos3 | 901 | 402 | +90º | -85º |
| Pos4 | 692 | -178 | +130º | -85º |
| Pos5 | -179 | -48 | -100º | -200º |
| Pos6 | -72 | 583 | -100º | +70º |
| Pos7 | -901 | 397 | -90º | -85º |
| Pos8 | -887 | 240 | -100º | -85º |
| Pos9 | 458 | 549 | +130° | -200° |

*Ilustración 23. Área de trabajo del ABB IRB 1200*

A continuación, seleccionamos uno a uno los dos robots IRB 1200 que se emplean en este proyecto para la parte de elaboración de la comida. Una vez posicionados en el lugar adecuado, es necesaria la creación de un "Controlador virtual" que podrá controlar ambos robots desde una misma FlexPendant. Para ello se selecciona en posición inicial, controlador virtual y se escoge una forma de creación. En este proyecto se optó por la creación desde diseño, en él se deberá elegir la ruta donde alojar el proyecto, así como elegir el RobotWare conveniente para cada robot (En nuestro caso para el IRB 1200 tiene RobotWare 6.12.03).

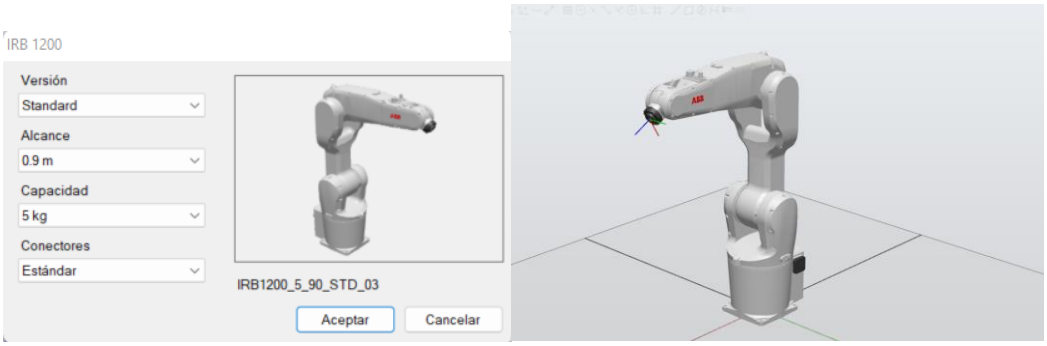



*Ilustración 24. Selección de las características del ABB IRB 1200*

En cuanto al controlador virtual del IRB 1200, emplea IRC5 FlexPendant un controlador muy eficiente que permite operar en manual o automático. En el lado derecho se encuentra el jostick de control, el paro de emergencia, entre otras cosas. En el, se pueden diseñar movimientos, crear señales, ver errores o incluso calibrar los distintos ejes del robot.

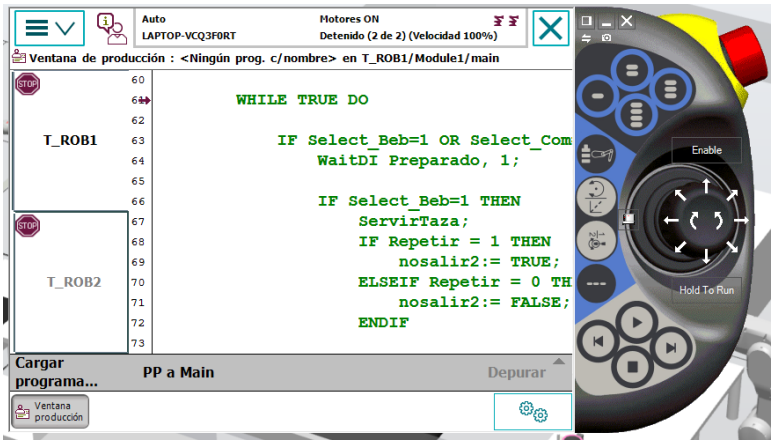



*Ilustración 25. Controlador IRC5 FlexPendant.*

### 4.1.1.2 CRB 15000 gofa y Omnicore FlexPendant

Al igual que para el IRB 1200, se analizaron varios robots colaborativos dando a lugar a una decisión contundentemente a favor del robot CRB 15000 Gofa [21] debido a sus características. Es un robot que dispone de una seguridad muy avanzada ya que dispone de señores de par integrados en cada una de las seis articulaciones, es el más veloz en su tipo con 2,2 metros por segundo. Dispone de una fácil configuración con una intuitiva FlexPendant llamada Omnicore que más adelante se comentará. Tiene un gran alcance (1,62 metros un 14% mayor que otros cobots similares) por lo que puede ofrecer una mayor productividad. Cuenta con la certificación de seguridad PL d Ct3 gracias a su diseño con geometría redondeada.

La figura muestra los sentidos positivo y negativo de cada eje al mover el robot en el sistema de coordenadas de la base.

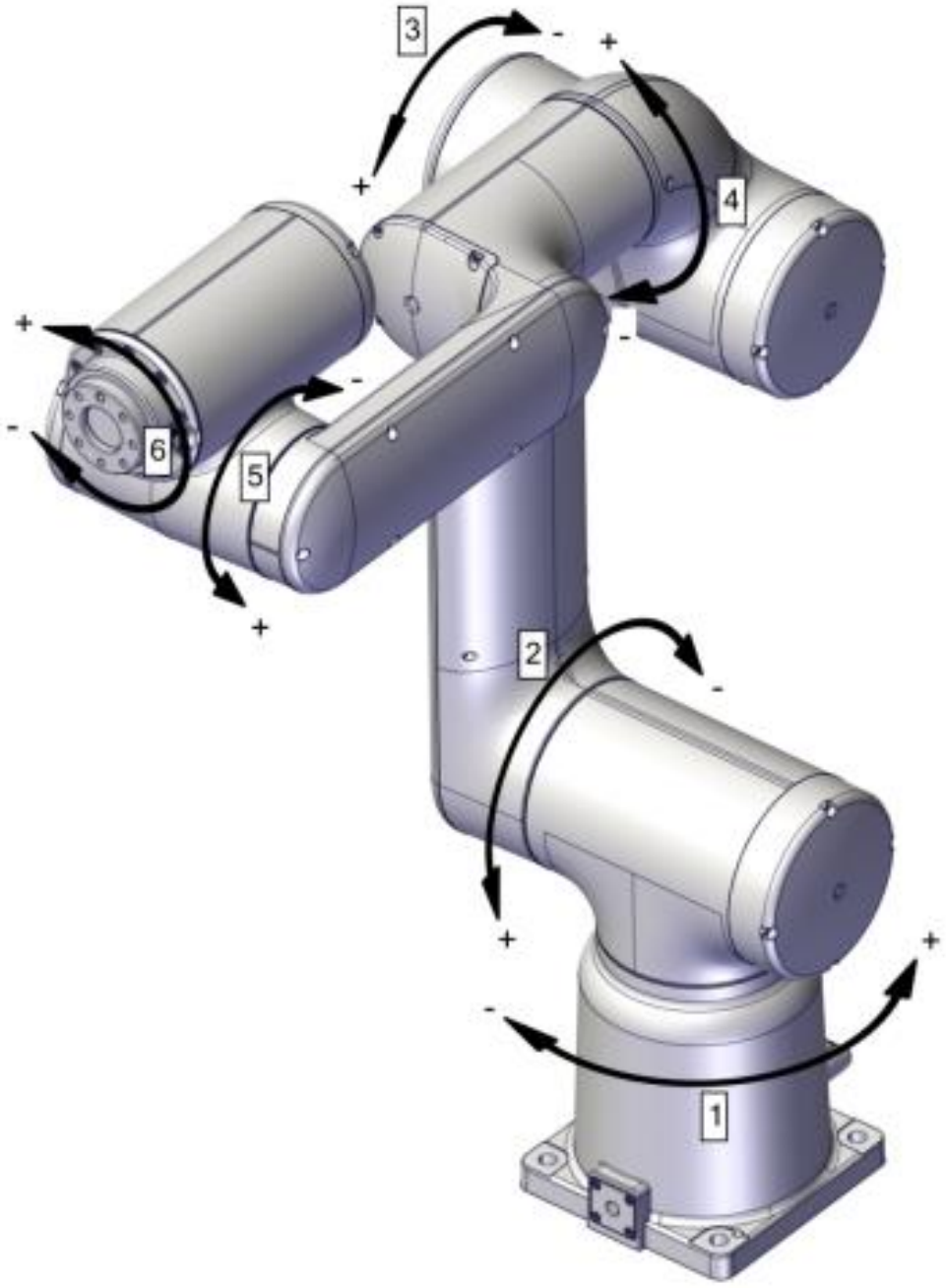
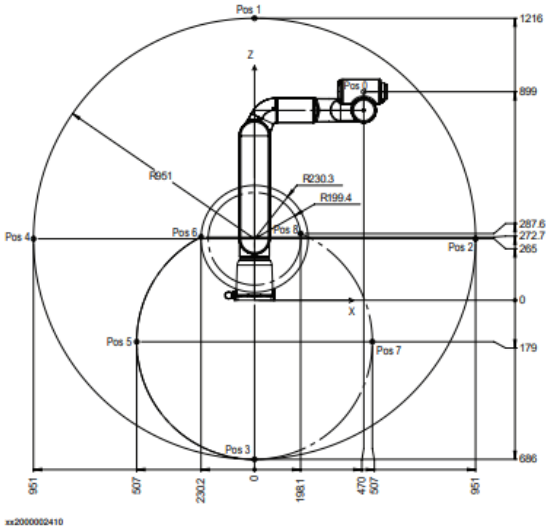



*Ilustración 26. Ejes de giro del ABB CRB 15000 GoFa.*

**Figura, área de trabajo de CRB 15000-5/0.95**
Esta figura muestra el área de trabajo del robot sin la restricción.




**Posiciones en el punto de inserción de ejes 4-5-6 y ángulo de ejes 2 y 3**

| Posición en la figura | Posiciones en el centro de la muñeca (mm) | | Ángulo (grados) | | |
|---|---|---|---|---|---|
| | X | Z | Eje 2 | Eje 3 | eje 5 |
| pos0 | 470 | 899 | 0° | 0° | 0° |
| pos1 | 0 | 1216 | 0° | -68° | 0° |
| pos2 | 951 | 265 | 90° | -68° | 0° |
| pos3 | 0 | -686 | 180° | -68° | 0° |
| pos4 | -951 | 265 | -90° | -68° | 0° |
| pos5 | -507 | -179 | 180° | 22° | 0° |
| pos6 | -230.2 | 272.7 | 180° | 85° | 0° |
| pos7 | 507 | -179 | 180° | -158° | 0° |
| pos8 | 198.1 | 287.6 | 180° | -225° | 0° |

*Ilustración 27. Área de trabajo del ABB CRB 15000 GoFa*

A continuación, del mismo modo que el IRB 1200 se crea en la estación el robot CRB 15000 Gofa con su respectivo controlador virtual, ya que no se permite el uso de un único controlador para estos dos distintos tipos de robots. En este caso, se crea el controlador virtual con RobotWare 7.7.0.

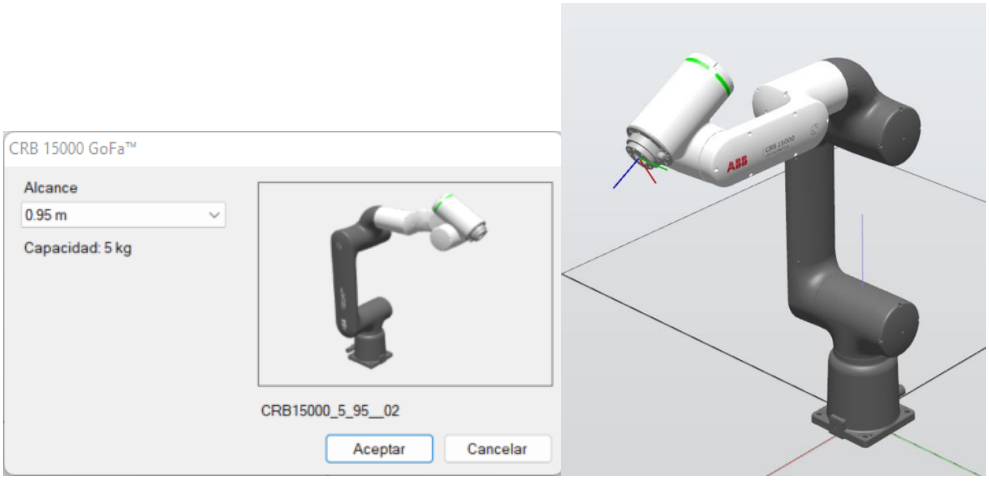



*Ilustración 28. Selección de las características del ABB CRB 15000 GoFa.*

Respecto al controlador del CRB 15000 Gofa, recibe el nombre de Omnicore. Es una actualización del IRC5 FlexPendant pero con una pantalla más grande y más intuitiva, tiene un mayor ahorro energético ya que reduce un 20 % del consumo.

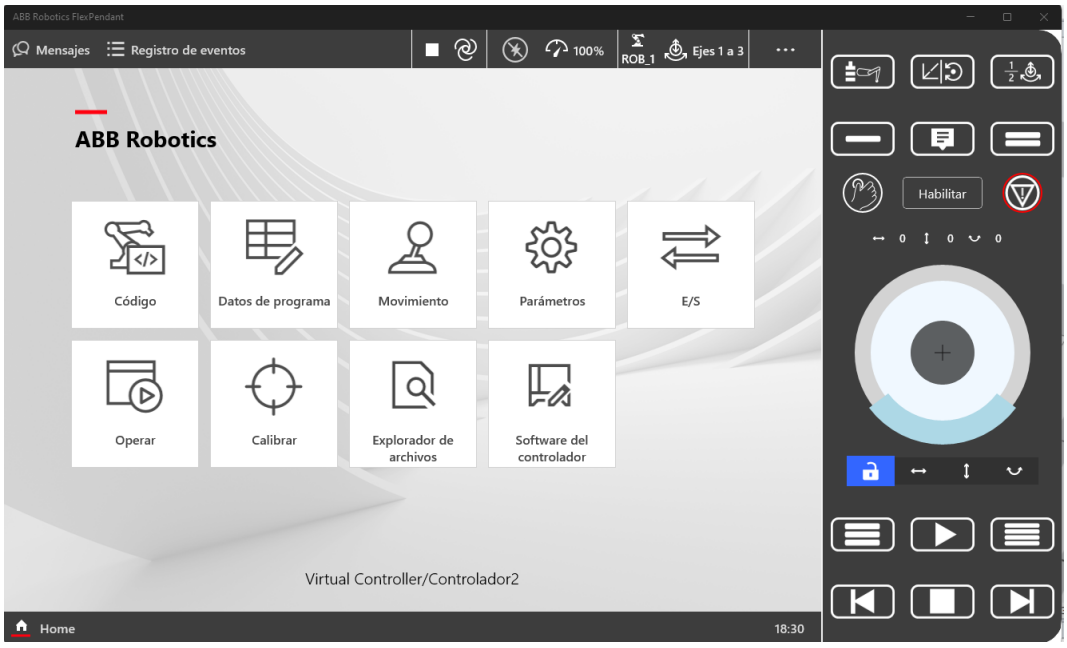



*Ilustración 29. Controlador Omnicore.*

### 4.1.1.3 Mecanismo Persona

Para llevar a cabo el presente proyecto, se debía implementar una persona para simular el servicio de asistencia, así que se pensó en cómo implementarlo. Primero de todo, se pensó en un componente que tiene RobotStudio que permite incorporar humanos a la estación, pero se descartó por razones de movimientos y porque el componente daba algunos errores de funcionamiento, pero es un componente muy interesante para el futuro porque la idea de su creación es para poder simular la estación virtual tanto con los robots como con las personas correspondientes.

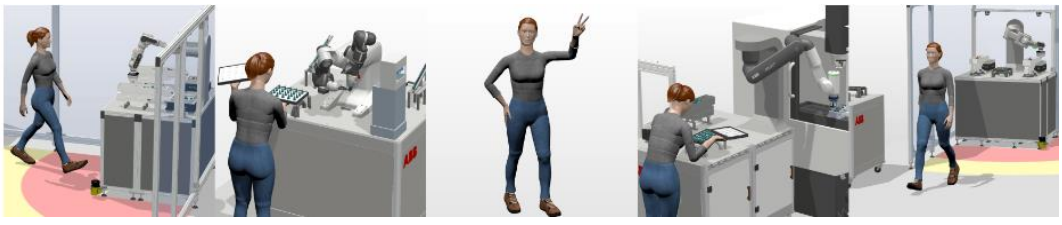


*Ilustración 30. Virtual human de RobotStudio.*

Después de buscar otras posibles soluciones, se decidió por diseñar un mecanismo que simule las prestaciones que nos haría una persona, dicho

mecanismo está compuesto por 5 ejes, pero de cara a este proyecto solo nos interesa uno que corresponde con el giro de la cabeza. También, se podría incluir otros movimientos como levantar una mano o un pie.

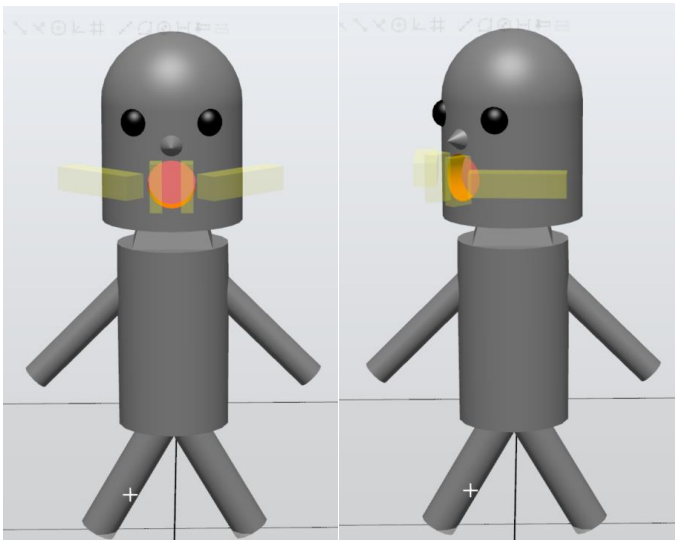


*Ilustración 31. Mecanismo propio de una persona*

Para conseguir la persona, se crea por partes la persona para después en "Crear mecanismo" poder unirlas, se debe seleccionar en "Tipo de mecanismo" que se va a crear un dispositivo. Acto seguido, se configurarán los sucesivos eslabones y ejes.

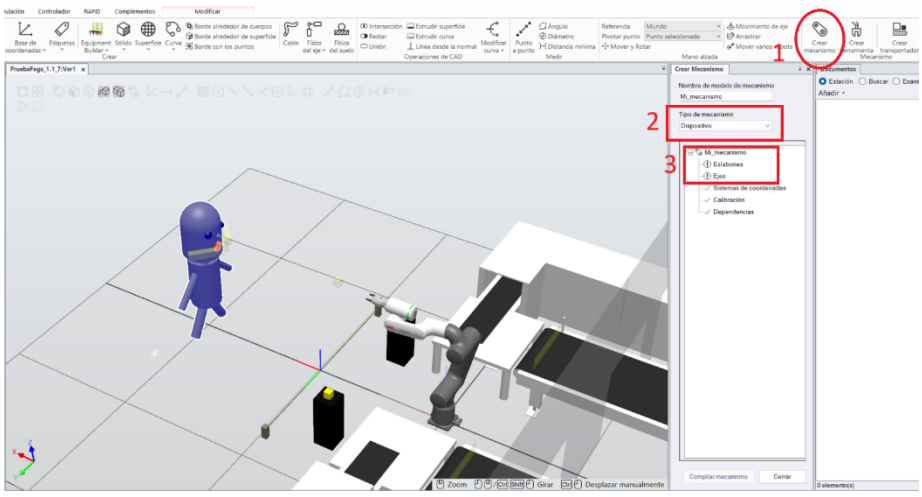



*Ilustración 32. Crear mecanismo.*

En el apartado de eslabones (imagen 1) se debe establecer como eslabón base el cuerpo de la persona debido a que es la única parte que no tendrá giro, se establecerá el nombre de cada eslabón con su componente asociada lo que permitirá después en el apartado de ejes (imagen 2) definir el tipo de movimiento y la amplitud que se desea en cada uno de los ejes. En esta parte,

se ha de establecer cada articulación como eslabón principal, el cuerpo, y como eslabón secundario, la cabeza, el brazo o la pierna.

Una vez configurados los eslabones y los ejes, y accionado compilar mecanismo ya estaría creada la persona que se necesita para simular el presente proyecto.

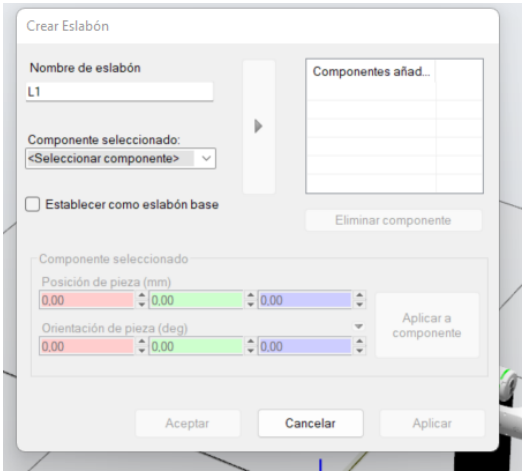



*Ilustración 33. Configurar eslabón.*

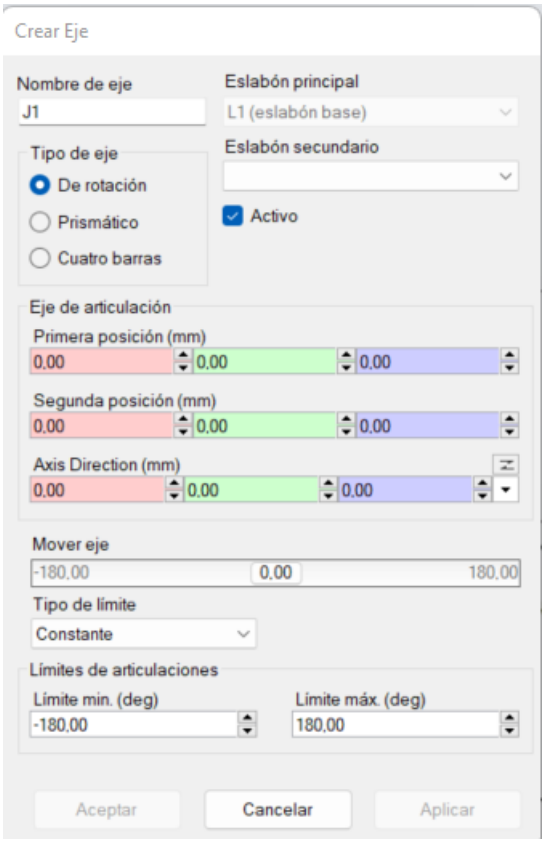



*Ilustración 34. Configurar ejes del mecanismo,*

#### 4.1.1.4 Pulsadores

Al igual que se ha realizado una comunicación entre la persona y el robot colaborativo CRB 15000 Gofa, es necesario hacer una conexión entre el robot colaborativo y los robots IRB 1200 para activar su funcionamiento. Se ha ideado otro mecanismo, pero en este caso, solo dispondrá de un eje lineal que simula el accionamiento de un botón real que será pulsado por el CRB 15000 Gofa una vez haya descifrado que desea la persona.

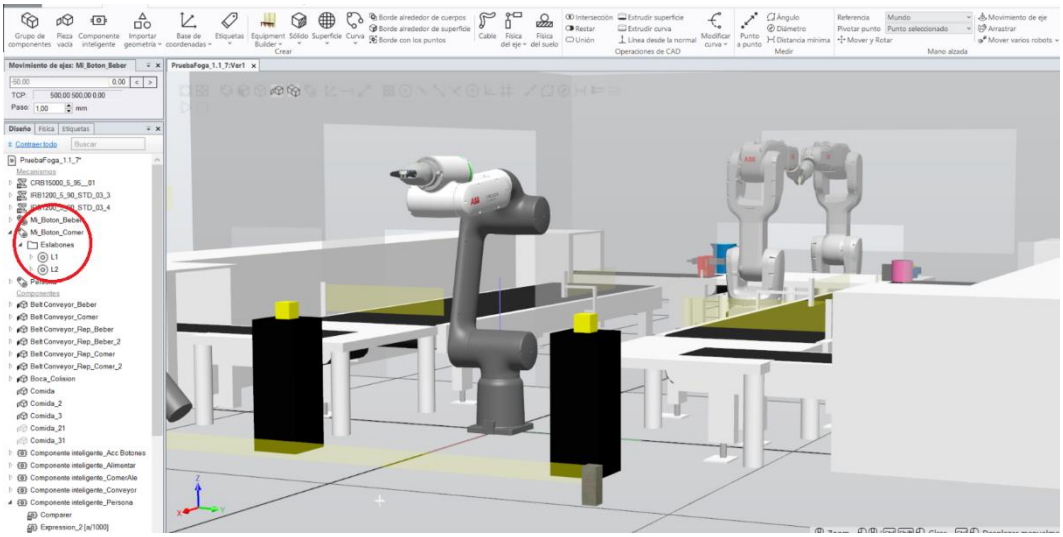


*Ilustración 35. Ubicación de los pulsadores en la estación.*

Siguiendo los pasos mencionados para crear el mecanismo de la persona, se aplica para este caso con dos eslabones y un eje como se observa en el margen izquierdo de la ilustración de arriba.

#### 4.1.1.5 Cintas transportadoras

En el presente proyecto es necesaria la construcción de cuatro cintas transportadoras con el objetivo de agilizar el proceso de elaboración del servicio sin la ayuda humana y permitiendo el suficiente espacio para que puedan trabajar libremente cada robot.

Desde la vista de planta de la estación observamos la ubicación de cada una de las cintas transportadoras. Repartidas dos a dos para separar el servicio de comida del de bebida y también, el proceso de preparación del proceso de recogida.

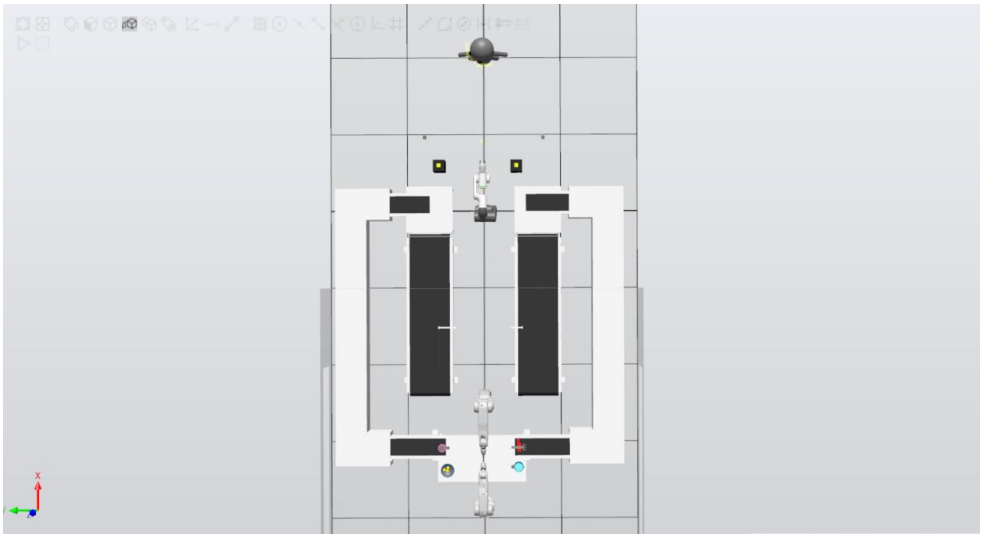


*Ilustración 36. Cintas transportadoras desde la perspectiva de planta.*

Como se puede apreciar en la siguiente imagen, las cintas transportadoras no están creadas con rodillos, si no, que están compuestas de una cinta de caucho debido a que para el servicio de hidratarse aporta mayor estabilidad que los rodillos y al tratarse de caucho aporta mayor fijación a la taza al tener un índice de fricción elevado (cerámica-caucho; taza-cinta) para que no se derrama el líquido en ningún momento.

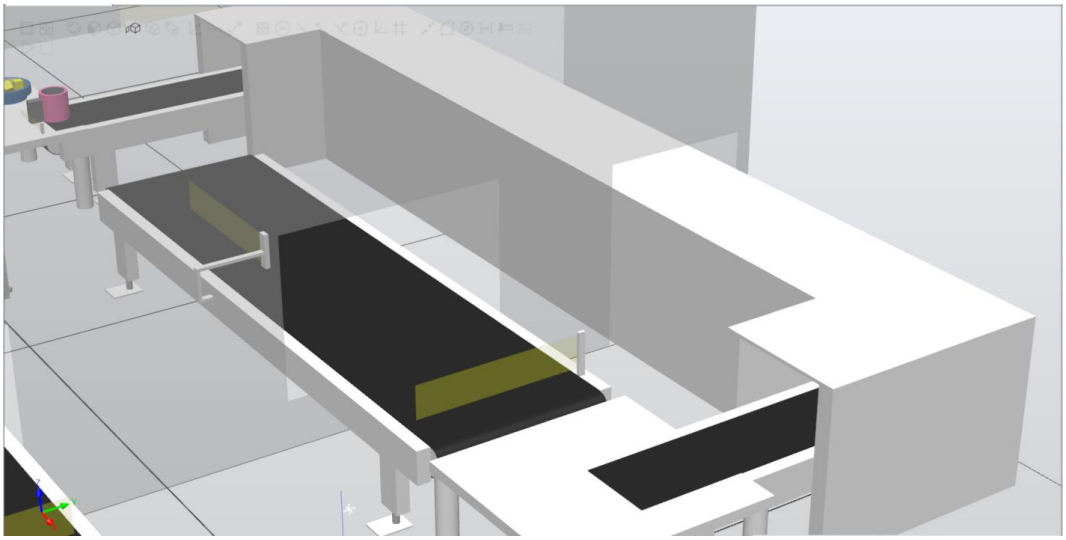


*Ilustración 37. Cinta transportadora estructura.*

La cinta principal dispone de una longitud de dos metros, una anchura de medio metro y se encuentra a una altura general de cero con tres metros, siendo la altura idónea para trabajar ambos robots. Su función consiste en transportar el servicio preparado por el IRB 1200 hasta el CRB 15000 Gofa para que este pueda servir.

La cinta secundaria dispone de una forma en U que permite el retorno de los objetos sin la manipulación de los humanos. En el propio RobotStudio se diseñó la cubierta de la cinta transportadora con el objetivo de optimizar la vuelta de los utensilios sin darse la posibilidad de que se extravíen.

#### 4.1.1.6 Utensilios

Junto con los robots, los objetos más importantes para llevar a cabo este proyecto son los utensilios para poder realizar el servicio correctamente ya que de ellos depende la forma de servir. Se han diseñado cinco objetos:

- Jarra: dispone de las correspondientes medidas. Diámetro 90 milímetros y altura de 150 milímetros.

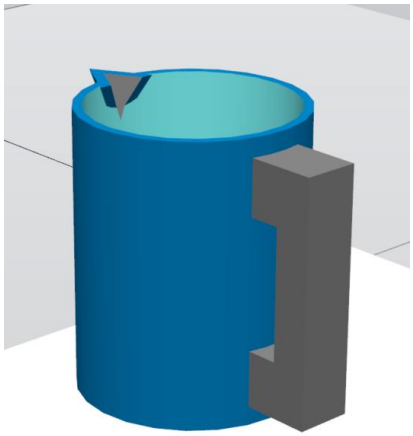


*Ilustración 38. Objeto Jarra.*

- Taza: dispone de las correspondientes medidas. Diámetro 70 milímetros y altura de 100 milímetros.

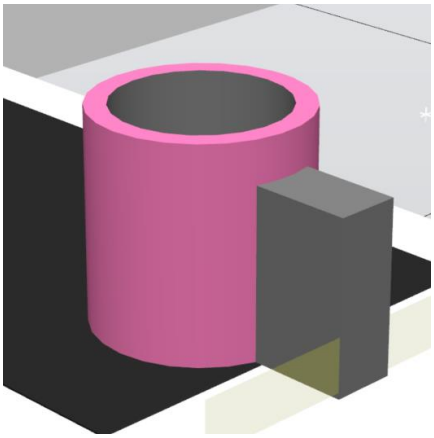


*Ilustración 39. Objeto Taza*

- Plato: dispone de las correspondientes medidas. Diámetro superior de 160 milímetros, diámetro inferior 100 milímetros y con una altura de 50 milímetros.

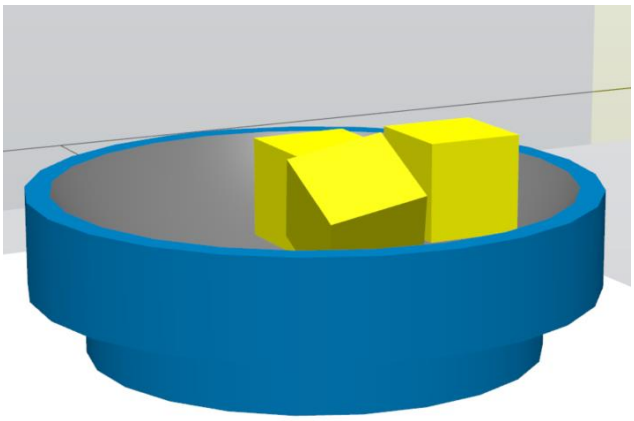


*Ilustración 40. Objeto Plato.*

- Soporte del tenedor: se diseñó con el objetivo de poder transportar la comida sin tener que llevar por separado plato y tenedor.

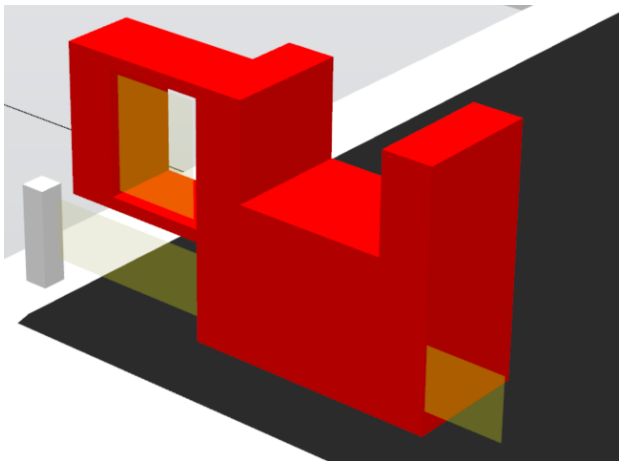


*Ilustración 41. Objeto. Porta-Tenedor.*

- Tenedor: dispone de unas medidas más grandes de lo habitual para facilitar el trabajo a los robots y poder realizar de mejor manera cada servicio. Longitud 200 milímetros y anchura 30 milímetros.

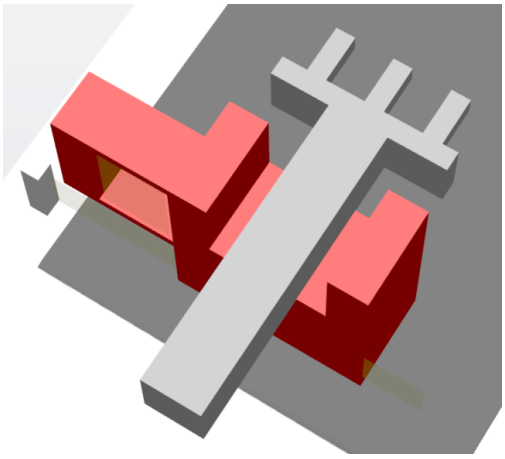
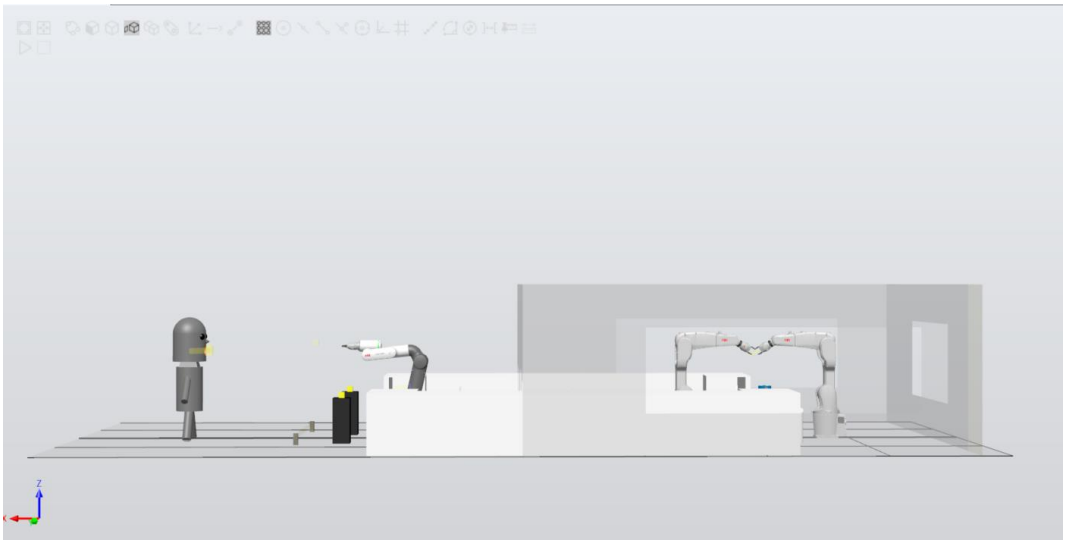
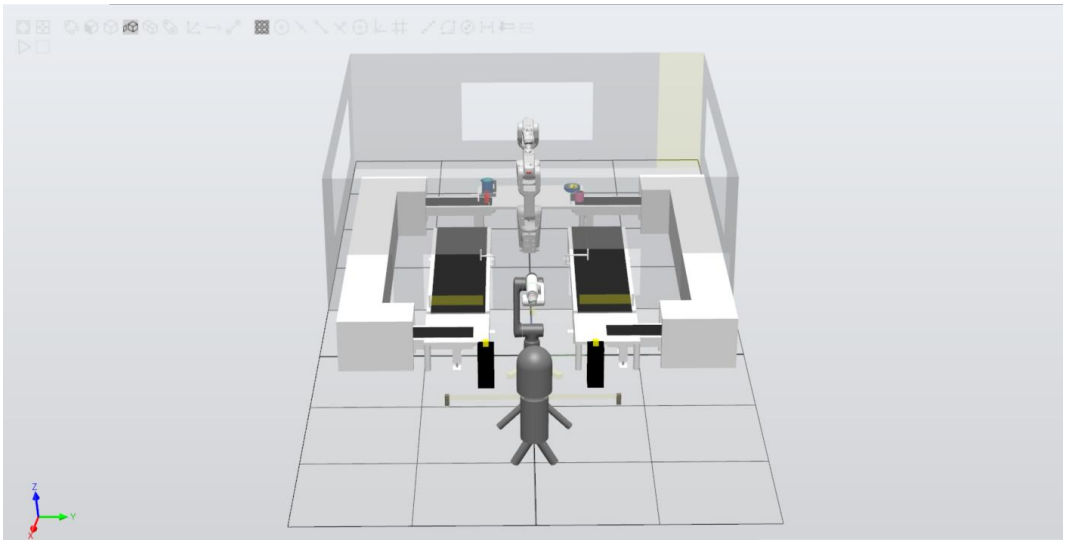


*Ilustración 42. Objeto Tenedor.*

Después de introducir todos los componentes utilizados para crear la estación se procede a mostrar unos planos generales de ella.





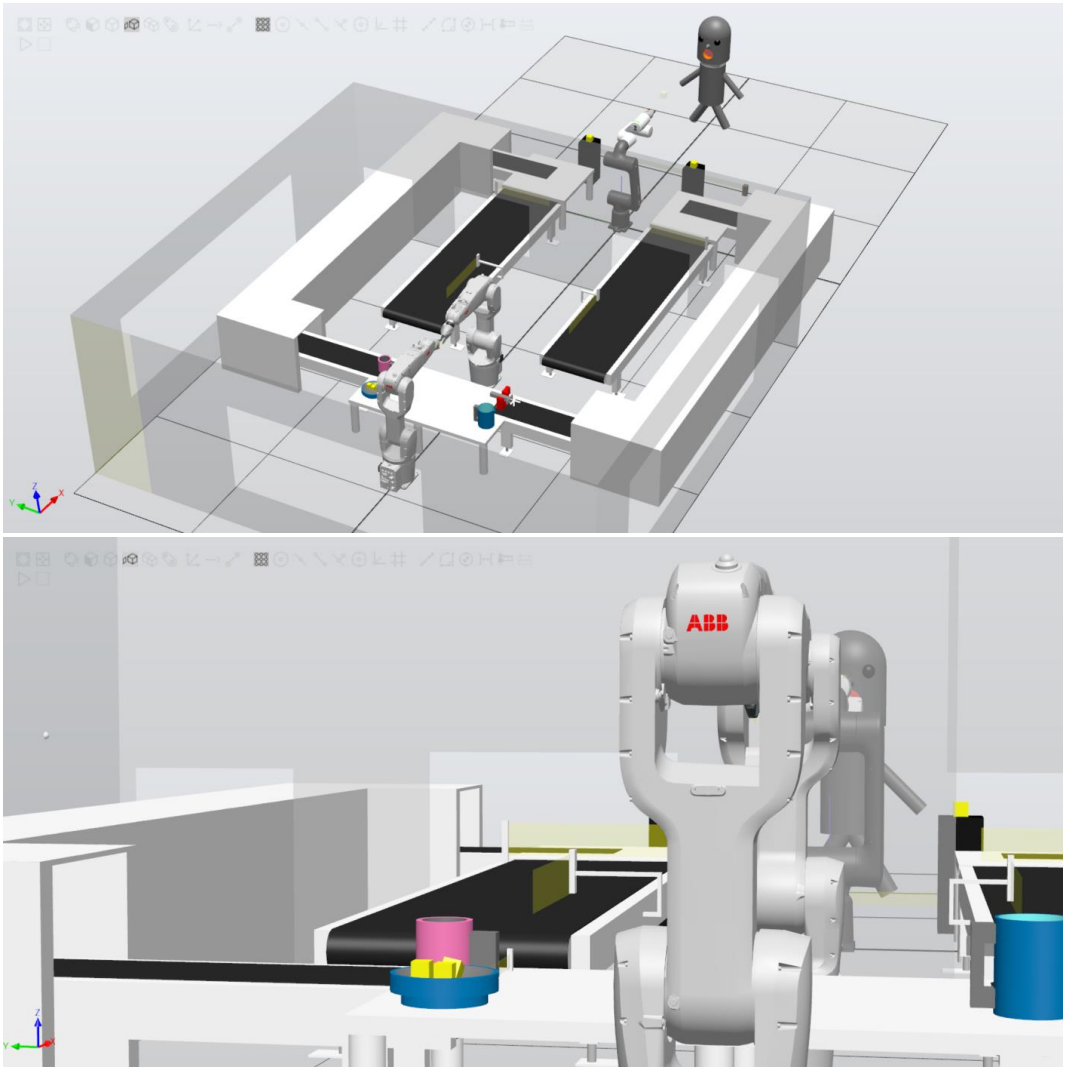



*Ilustración 43. Planos generales de la estación robótica.*

### 4.1.2 Lógica de estación

Una vez configurada la estación con cada componente para poder definir el comportamiento de cada uno durante la simulación, es necesario el uso de los componentes inteligentes de RobotStudio, son objetos con lógicas que permiten la entrada y salida de propiedades y señales correspondientes a cada componente.


*Ilustración 44. Abrir lógica de estación.*

En el apartado de diseño de la lógica de estación se puede observar el conjunto de componentes creados, pero primero de todo se deben de crear y saber qué categorías se pueden emplear. Son las siguientes:

- Señales y propiedades: encontraremos los bloques correspondientes a puertas lógicas como AND, OR, XNOR, etc., temporizadores, expresiones matemáticas, etc.
- Primitivos paramétricos: estos bloques permiten crear de forma automática cualquier tipo de sólido o líneas durante la simulación.
- Sensores: se encuentran distintos tipos de sensores que emitían una señal cuando un objeto los atraviese.
- Acciones. En esta categoría se pueden encontrar bloques muy útiles para simular la unión de dos objetos, crear o eliminar objetos, hacer visible o invisible, etc.
- Manipuladores: encontraremos bloques que permiten mover objetos o mecanismos en trayectorias lineales y circulares, también permiten mover los ejes de distintos mecanismos.
- Controlador: existe un bloque que se puede leer o escribir variables utilizadas en RAPID.
- Física: se puede asignar propiedades físicas a algún objeto o articulación para realizar la simulación de la forma más realista posible.
- PLC: este bloque nos permite establecer conexiones con PLCs de Siemens mediante OPC o SIMIT Connection.
- Otros: en esta categoría se encuentra una multitud de fe bloques que permiten diversas funcionalidades.

A continuación, después de presentar los distintos bloques de componentes inteligentes, se continua con la lógica de estación creada para el presente proyecto, donde se observa como cada componente esta conecta a su respectivo controlador, ya que se dispone de dos controladores, uno para el robot CRB 15000 Gofa y otro para los robots IRB 1200 como se observa en la siguiente imagen.

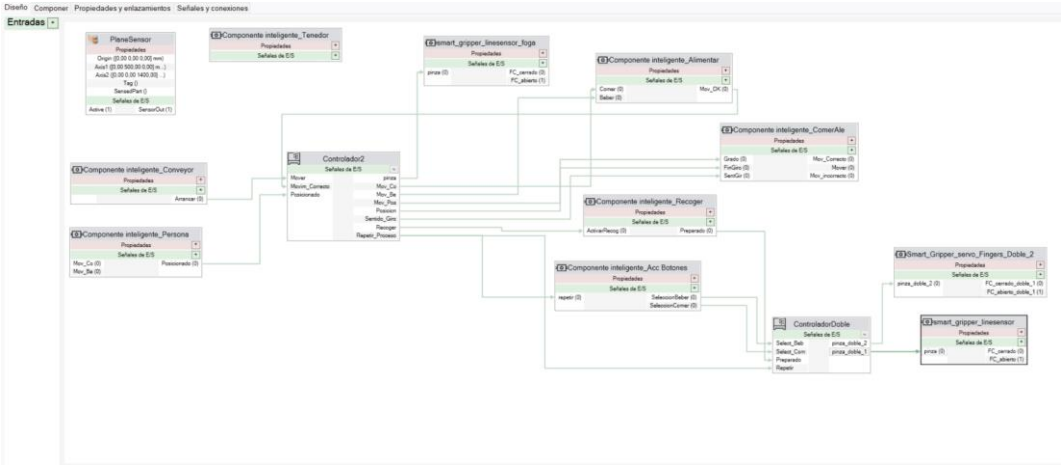



*Ilustración 45. Lógica de estación del TFG.*

#### 4.1.2.1 Pinza

Se ha diseñado un componente inteligente en base a la herramienta proporcionada por RobotStudio en la que se han empleado ocho componentes para su creación. Se ha creado una para cada robot, por lo que existen tres *Smart Components* (smart_gripper_linesensor_foga, smart_gripper_linesensor y Smart_Gripper_servo_Fingers_Doble_2).

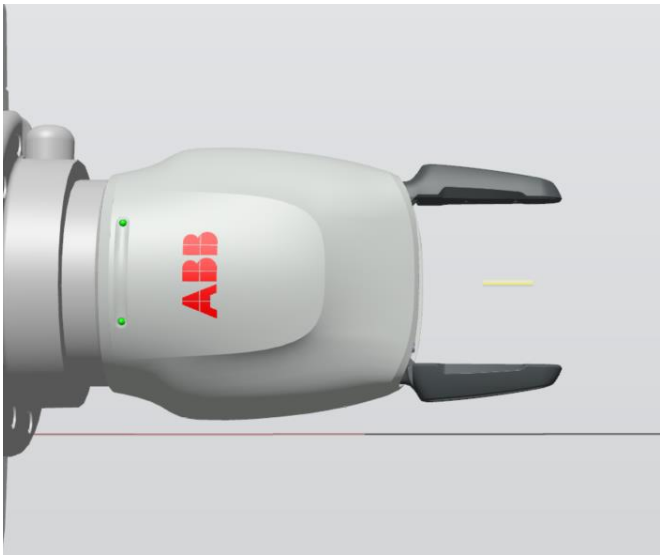



*Ilustración 46. Objeto inteligente. Pinza.*

Respecto a su lógica de estación, tenemos los siguientes componentes:

- LineSennsor: este componente corresponde a un sensor con el aspecto de una línea que se ha colocado en el interior de la pinza para detectar la colisión con el objeto a recoger. El aspecto de la línea se definirá en los dos primeros apartados (*Start* y *End*) donde se especificará el punto

de inicio y final respectivamente. El siguiente apartado a modificar es el correspondiente al radio que permitirá determinar el grosor de la línea. Con el parámetro SensedPart se conocerá el objeto que se encuentra en contacto con el sensor cuando este esté activo, es decir, el valor lógico este a '1'. Una vez se ha producido la colisión y el LineSensor está activo, se emitirá un uno lógico a la salida (SensorOut).


*Ilustración 47. Componente inteligente. LineSensor.*

- LogicGate: corresponde con un bloque de puertas lógicas que su salida se activa en función de la operación lógica definida. Para este caso, se emplea una puerta lógica NOT, la cual, devolverá el valor negado de su entrada.

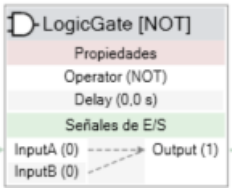



*Ilustración 48. Componente inteligente. LogicGate.*

- PoseMover: sirve para mover los ejes de un mecanismo hacia una posición indicada. Es necesario especificar el mecanismo a mover, en este caso, es la pinza (Smart gripper), también hay que indicar hasta que posición ha de moverse y la duración del movimiento. Se implementan dos, uno para el movimiento de abrirse y otro para el de cerrarse ya definidos en el mecanismo.

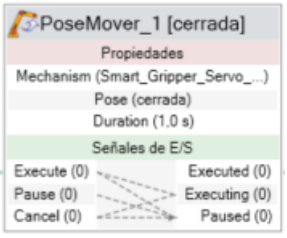



*Ilustración 49. Componente inteligente. PoseMover.*

- Attacher: componente que sirve para crear la conexión entre el objeto y la pinza (*Parent)*, debido a que conectará el objeto a ella cuando se active la señal de ejecución, Execute. Flange indica la herramienta a la que conectarse. Mediante el apartado, *child*, se especificará el objeto a conectar que en este componente inteligente lo determinará el LineSensor a través de SensedPart. Mount se declara falso ya que en el caso de que sea verdadero, el objeto a conectar se monta sobre el objeto superior de vinculación.


*Ilustración 50. Componente inteligente. Attacher*

- Detacher: sirve para desconectar objetos, es decir, es el componente contrario a *Attacher.* Pues en el momento que se ejecute (*Execute)* se desconectará el objeto y en el caso de que *KeepPosition* esté en falso, devolverá el objeto a la posición inicial, por eso se deja verdadero.


*Ilustración 51. Componente inteligente. Detacher.*

- LogicSRLacth: es un biestable asíncrono que se usará para almacenar información, pues nos permite tener dos entradas (Set y Reset) y dos salidas (Q y Q'). Si Set y Reset valen cero, estará en reposo y devolverá la salida que tenían anteriormente. Set activa la señal de salida y Reset desactiva la señal de salida.

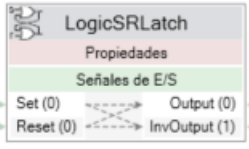



*Ilustración 52. Componente inteligente. LogicSRLacth.*

- SimulationEvents: bloque que permite emitir señales cuando se inicia o finaliza la simulación de la estación. Para la pinza se emplea de cara al inicio de la simulación para que se desconecte de cualquier objeto.


*Ilustración 53. Componente inteligente. SimulationEvents.*

#### 4.1.2.2 Persona

En el presente proyecto es necesario diseñar una persona que realice los movimientos necesarios para poder simular lo más real posible la estación. Por ello, se crean tres componentes inteligentes en lógica de estación.

El primer componente inteligente, Componente_inteligente_Persona, es el encargado de crear una simulación del andar de una persona y también, se modifica la envergadura de la persona para poder emplear el robot para todo tipo de estaturas.

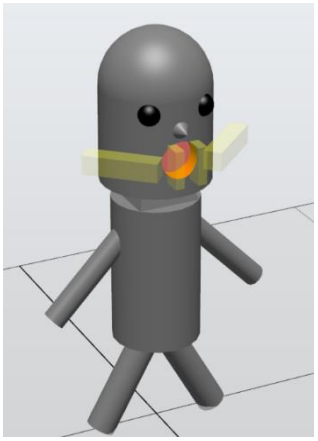


*Ilustración 54. Objeto inteligente. Persona.*

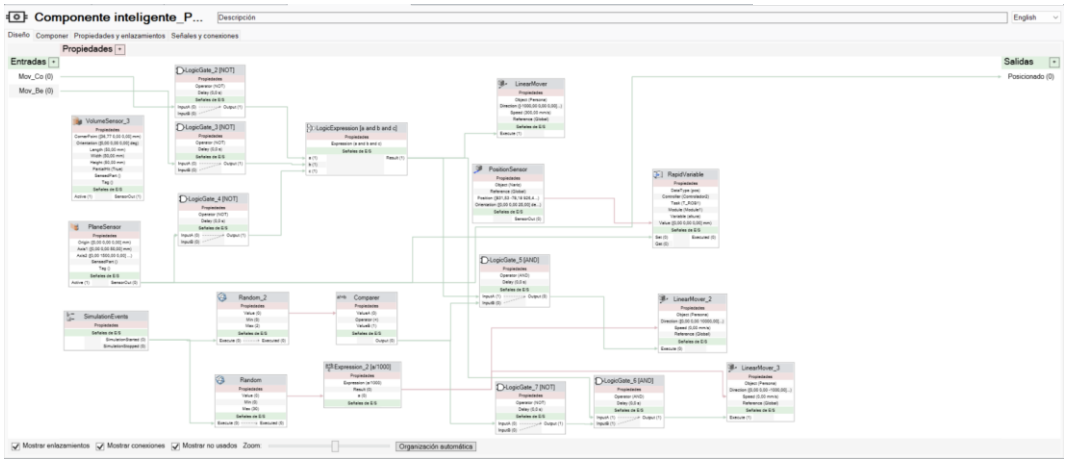



*Ilustración 55. Componente inteligente. Persona.*

Dispone dos partes, en la parte superior se crea el movimiento horizontal para acercarse al robot hasta una distancia de noventa centímetros, donde se detendrá y activará el robot CRB 15000 Gofa mediante el uso de un sensor.

Del mismo modo, en la parte inferior se genera el movimiento vertical mediante un generador de números aleatorios partiendo de una altura genérica de ciento setenta y cinco centímetros pudiendo alcanzar treinta centímetros por arriba o por abajo.

Debido a que para el presente proyecto RobotStudio no dispone de medios para identificar el rostro, se ha implementado un sensor en la nariz humana para determinar la altura de la persona y así simular la detección del rostro mediante una cámara o sistemas de IA.

Se emplean los siguientes componentes:

- PlaneSensor: este componente corresponde con un sensor plano al igual que el LineSensor, sirve para detectar los objetos que interfieren en el emitiendo una señal de salida (SensorOut) cuando se encuentran activos.


*Ilustración 56. Componente inteligente. PlaneSensor.*

- LogicExpression: permite evaluar una expresión lógica, permitiendo más entradas que LogicGate.


*Ilustración 57. Componente inteligente. LogicExpression.*

- LinearMover: realiza el movimiento del objeto declarado en *Object* a una velocidad indicada en *Speed* y hacia una dirección establecida en *Direction.* Para empezar el movimiento del objeto tendrá que estar activa la señal de entrada *Execute.*

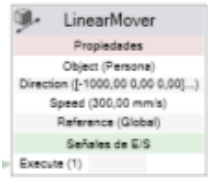



*Ilustración 58. Componente inteligente. LinearMover.*

- PositionSensor: capta la posición y orientación de un objeto establecido durante la simulación.

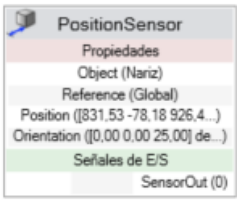



*Ilustración 59.. Componente inteligente. PositionSensor.*

- Random: componente que permite generar un número aleatorio en un intervalo establecido cuando se activa la señal de entrada.

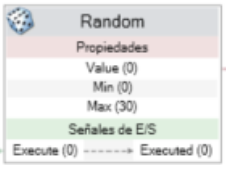



*Ilustración 60.. Componente inteligente. Random.*

- Comparer: componente lógico que compara dos entradas y en la salida devuelve un uno o un cero lógico dependiendo si se cumple la comparación o no.

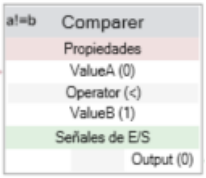



*Ilustración 61. Componente inteligente. Comparer.*

- Expression: evalúa una expresión matemática y devuelve el resultado de ella. El propio componente incluye operadores numéricos, funciones matemáticas, entre otras cosas.


*Ilustración 62. Componente inteligente. Expression.*

- RapidVariable: componente útil para establecer u obtener el valor de una variable de RAPID. Se debe establecer el tipo de dato que se va a usar para la variable, el controlador que la posee, al igual que la tarea y el módulo de RAPID que la contienen. Dispone de dos entradas que

corresponden a *Get* y a *Set,* la primera obtiene el valor de la variable y la segunda define el valor de la variable.

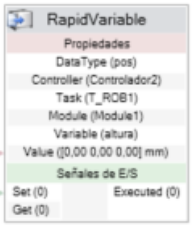



*Ilustración 63. Componente inteligente. RapidVariable.*

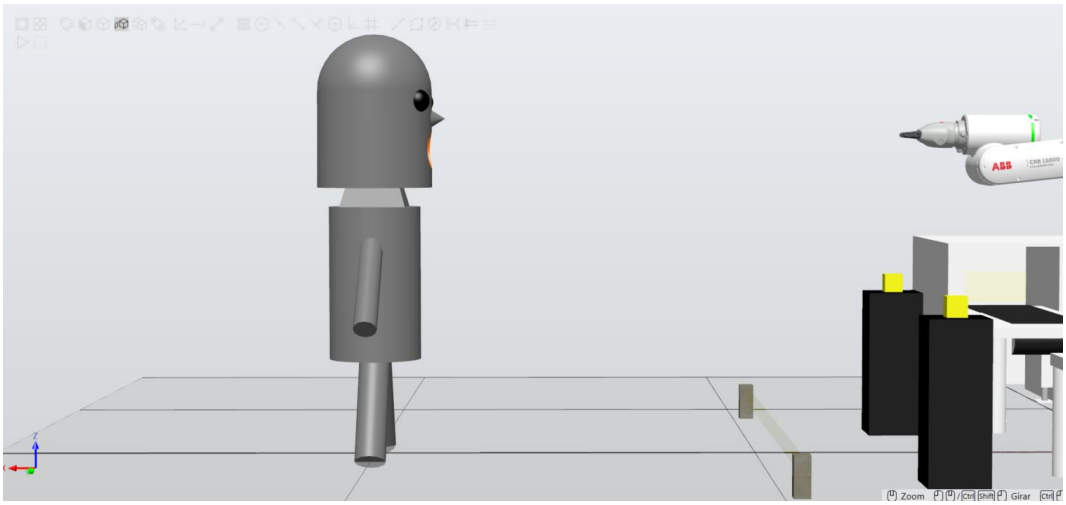


*Ilustración 64. Perspectiva lateral del objeto inteligente persona.*

El segundo componente inteligente es el llamado Componente_inteligente_Alimentar, es el encargado de permitir el giro de cabeza de la persona, en función de su decisión.

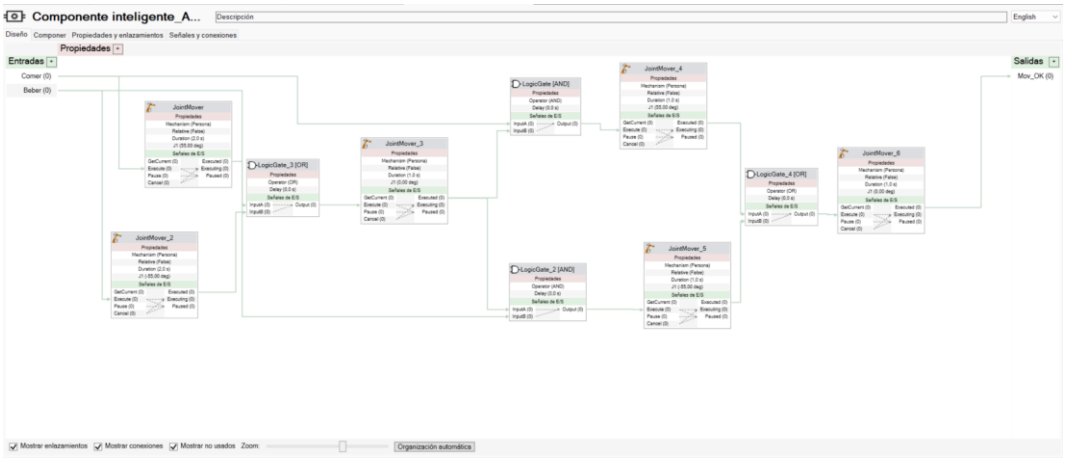



*Ilustración 65. Componente inteligente Alimentar.*

A mayores, se han empleado el siguiente componente que permite el giro de la cabeza:

- JointMover: se basa en las propiedades del mecanismo definido, permite establecer el giro del eje que corresponda y una duración indicada. Para poder llevar a cabo el giro del eje debe activarse la señal de entrada *Execute,* una vez alcanzada la posición requerida emite una señal de salida. También permite obtener los valores de los ejes actuales, pausar el movimiento y cancelar el movimiento con las señales de entrada.

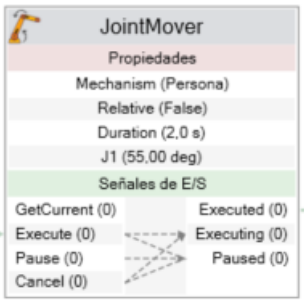



*Ilustración 66. Componente inteligente. JointMover.*

El tercer y último componente inteligente del mecanismo persona, es el encargado de simular el giro de la cara mientras se ejecuta el servicio asistencial.

Dispone de dos partes, la primera es la correspondiente al giro de la cabeza para posicionarse en el ángulo correspondiente. Y la segunda son sensores que actúan para verificar el correcto movimiento del robot para servir a la persona en el sitio correcto, es decir, en su boca. Emitiendo una señal de acierto o error mediante sensores.

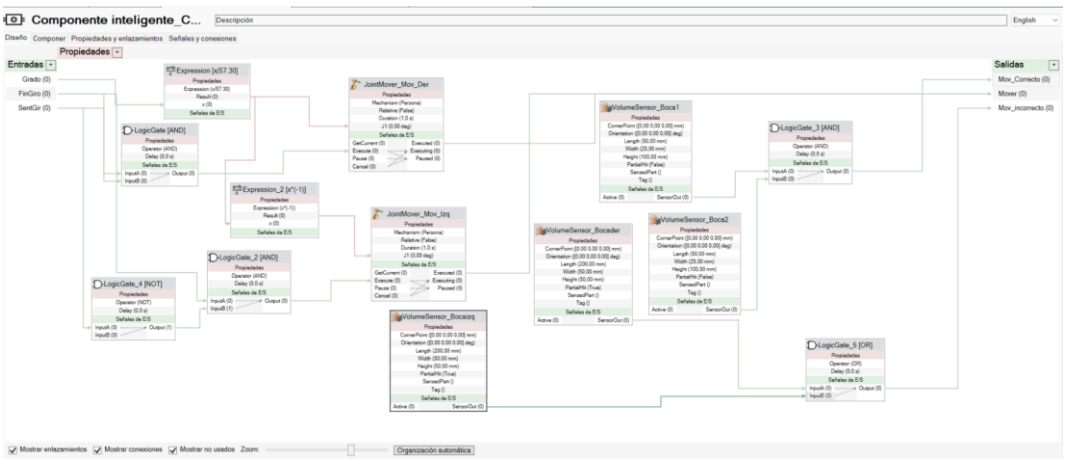



*Ilustración 67. Componente inteligente ComerAle*

#### 4.1.2.3 Botones

Para comunicarse con la "cocina" se diseña dos accionadores que permiten la comunicación entre el robot colaborativo y los robots industriales encargados de preparar el servicio. Se crean dos accionadores uno para comunicar que el humano quiere alimentarse y otro para hidratarse. Cada uno consiste en un plano de sensor y un mecanismo que emitirán la señal de salida cada vez que se activen.

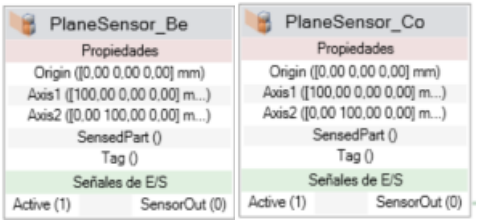



*Ilustración 68. Componentes inteligentes. PlaneSensor botones.*

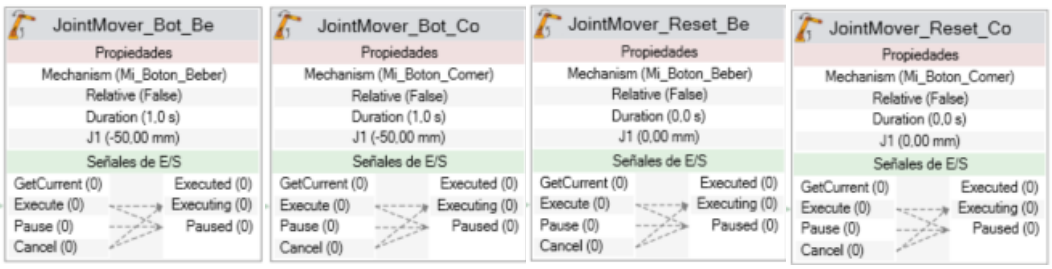



*Ilustración 69. Componentes inteligentes. JointMover Botones.*

Quedarán accionados hasta que se finalice la simulación o se desee repetir un servicio, por lo que volverán a su posición inicial con los componentes JointMover_Reset respectivo. La salida emitirá la elección de la persona que recibirá el controlador IRC5.

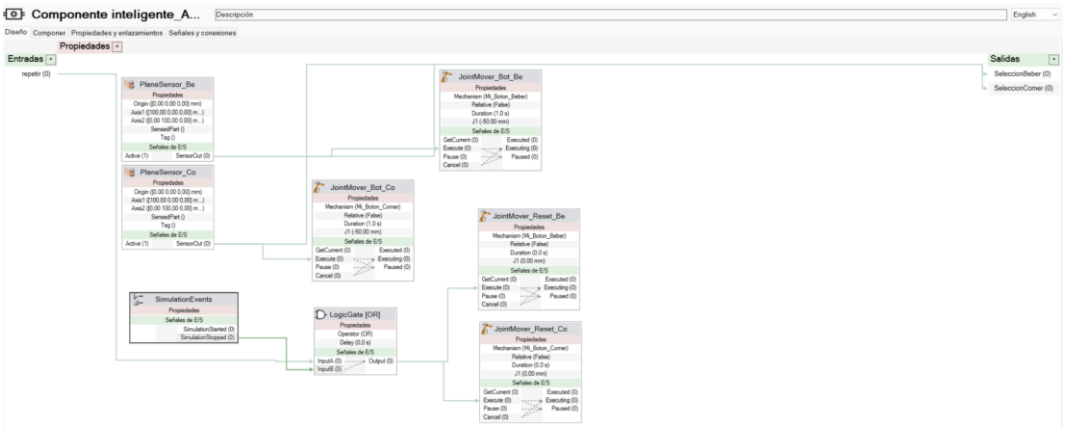



*Ilustración 70. Componente inteligente Acc_Botones*

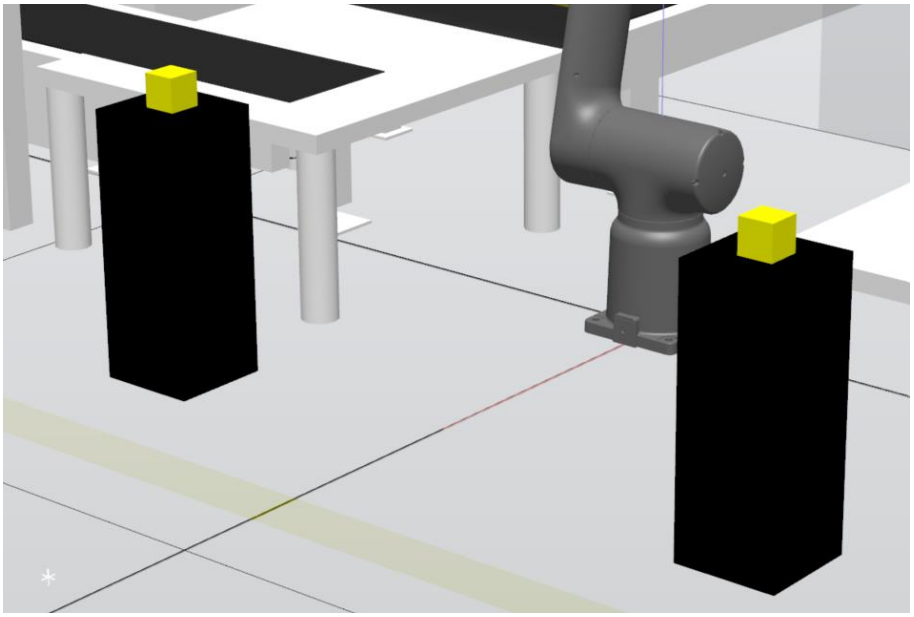


*Ilustración 71. Pulsadores en la estación.*

#### 4.1.2.4 Cintas transportadoras

Lo que respecta a las cintas transportadoras en el presente proyecto se ha divido en dos partes, el primer componente inteligente llamado Componente_inteligente_Conveyor corresponde con las dos cintas transportadoras para desplazar la comida desde los robots industriales hasta el robot colaborativo.

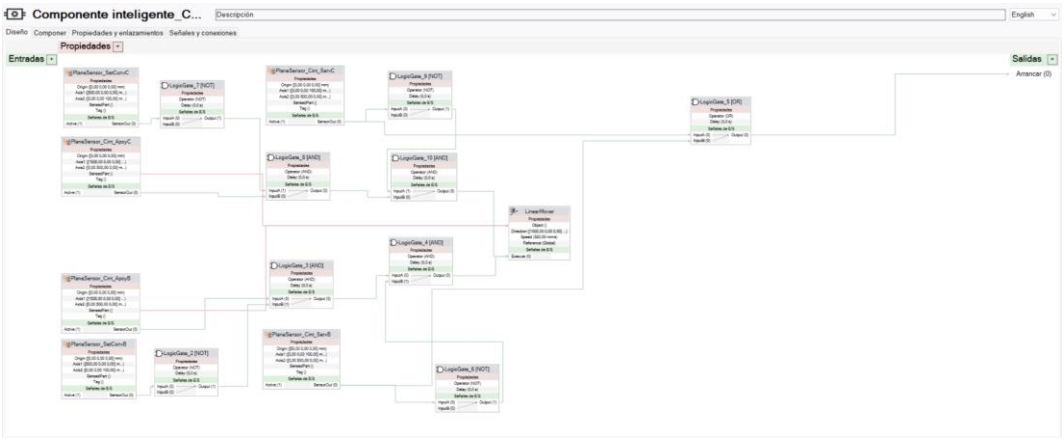



*Ilustración 72. Componente inteligente Conveyor*

Este componente dará la orden al robot colaborativo de que tiene el servicio disponible y puede ejecutar su asistencia, en el momento que la salida digital "Arrancar" se ponga a uno. Para ello es necesario que uno de los sensores finales de las cintas transportadoras se active porque han detectado un objeto.

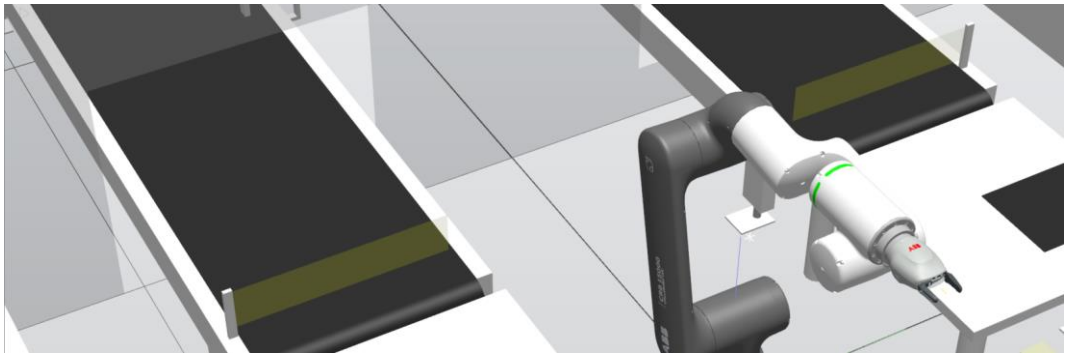


*Ilustración 73. Sensores finales cintas transportadoras iniciales.*

La entrada de cada PlaneSensor ha de estar activa durante la simulación:

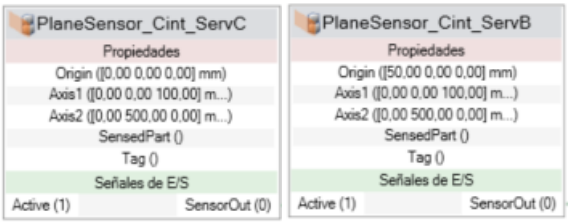



*Ilustración 74. Componentes inteligentes. PlaneSensor finales cintas transportadoras iniciales.*

Para poder llegar al final de la cinta transportadora primero tiene que activarse el movimiento de ella, por lo que se necesita que tanto el sensor integrado en la cinta se encuentre activo, como el sensor de movimiento y el sensor final no se encuentren activos.

La creación del sensor de movimiento tiene el objetivo de asegurar que el robot se ha retirado correctamente después de colocar el objeto correspondiente.

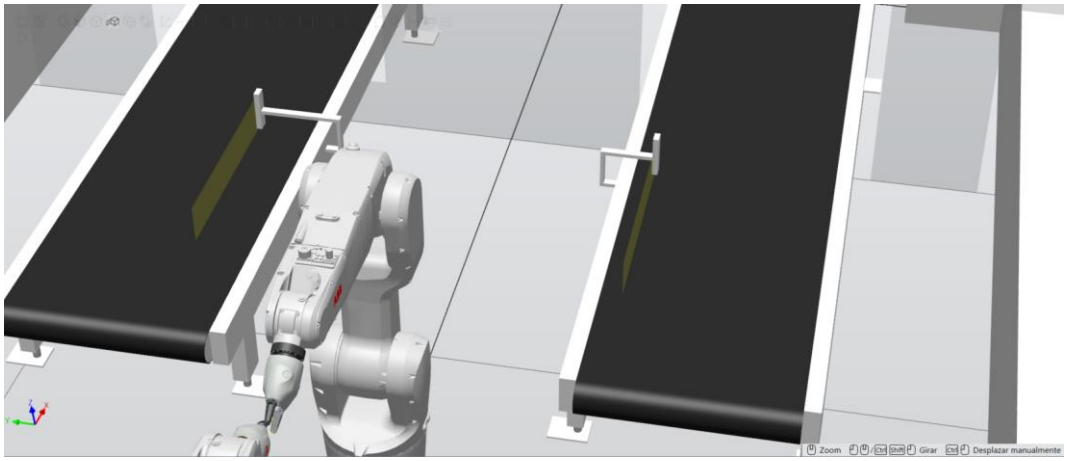


*Ilustración 75. Sensores iniciales cintas transportadoras iniciales.*

Lo que respecta con la lógica de estación se emplean varios componentes como los dos PlaneSensor (sensor integrado en la cinta y sensor de movimiento), dos puertas lógicas NOT y otras dos AND y el correspondiente LinearMover quien es el encargado de desplazar el objeto pertinente.

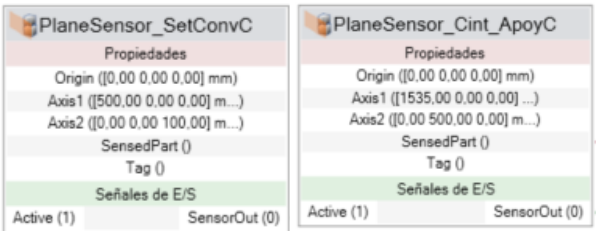



*Ilustración 76. Componentes inteligentes. PlaneSensor iniciales cintas transportadoras iniciales.*

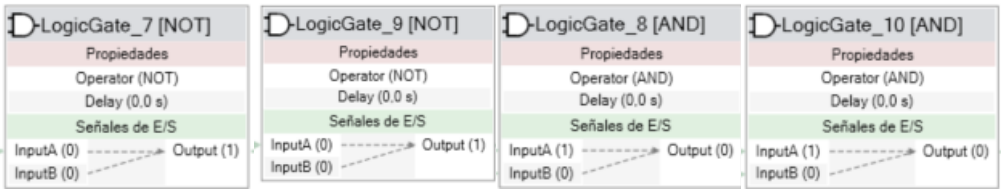



*Ilustración 77. Componentes inteligentes. LogicGate cintas transportadoras iniciales.*

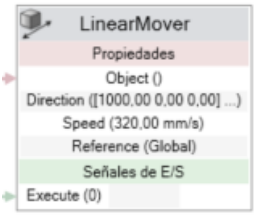



*Ilustración 78. Componente inteligente. LinearMover cintas transportadoras iniciales.*

En cuanto al segundo componente inteligente de las cintas transportadoras (Componente_inteligente_Recoger), es el encargado de recoger los objetos utilizados por el robot colaborativo y colocarlos en su posición inicial para que en el caso de que la persona desee realizar otro servicio, estos estén preparados.

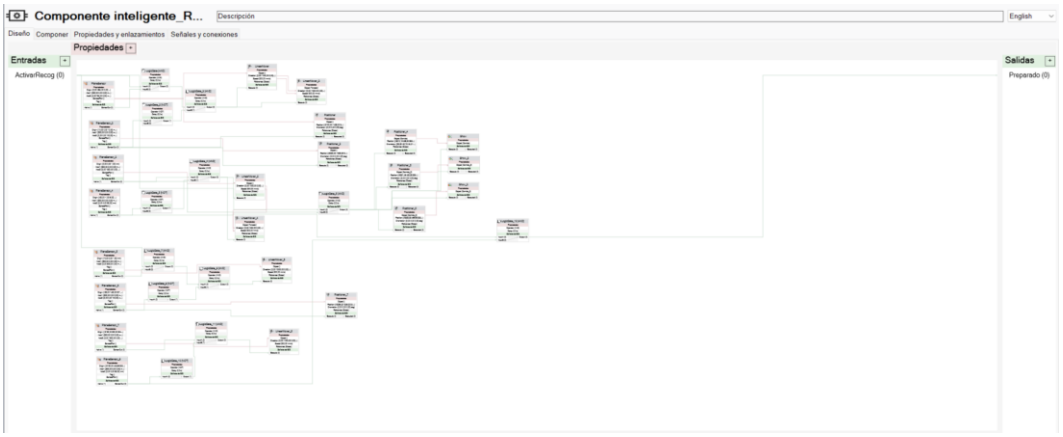



*Ilustración 79. Componente inteligente Recoger.*

A su vez, se dispone de dos partes diferenciadas, la cinta delantera y la cinta trasera. La primera cinta dispone del sensor integrado en ella que desplazará el objeto hacia el interior de la cubierta, y un sensor al final de ella para activar

el posicionador y colocar correctamente el objeto en la cinta trasera, siendo esta la más próxima a los robots industriales.

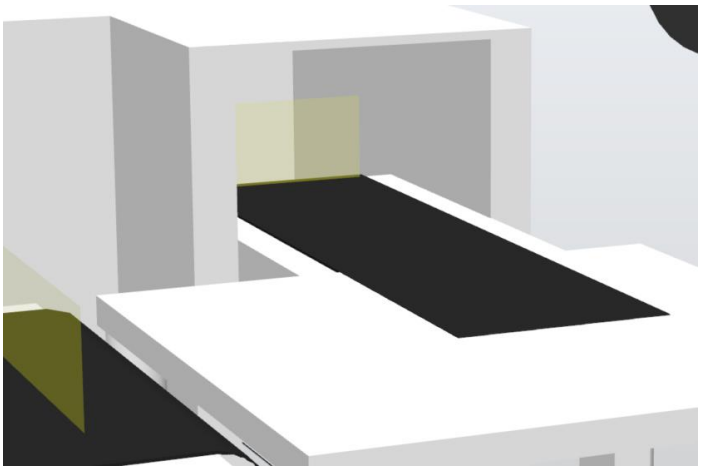


*Ilustración 80. Cinta transportadora retorno 1.*

La cinta trasera también dispone de un sensor integrado en ella para permitir el movimiento del objeto sobre ella hasta que se active el sensor final que detenga su movimiento y active su señal de salida para mandar la señal al robot de que los objetos están preparados.

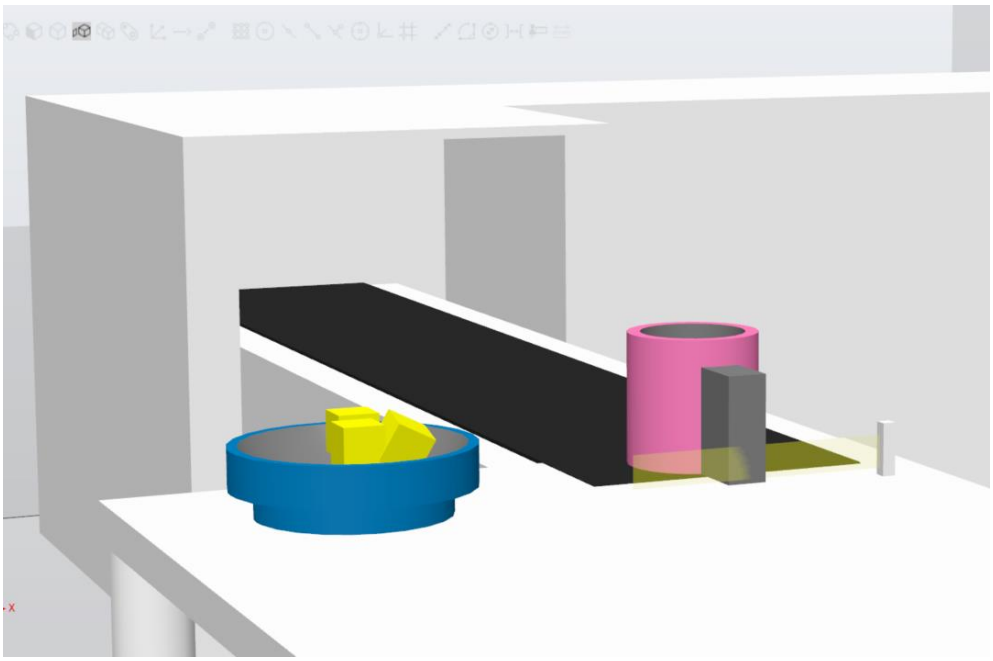


*Ilustración 81. Cinta transportadora retorno 2.*

A continuación, se muestra cómo sería para el caso del servicio de hidratación.

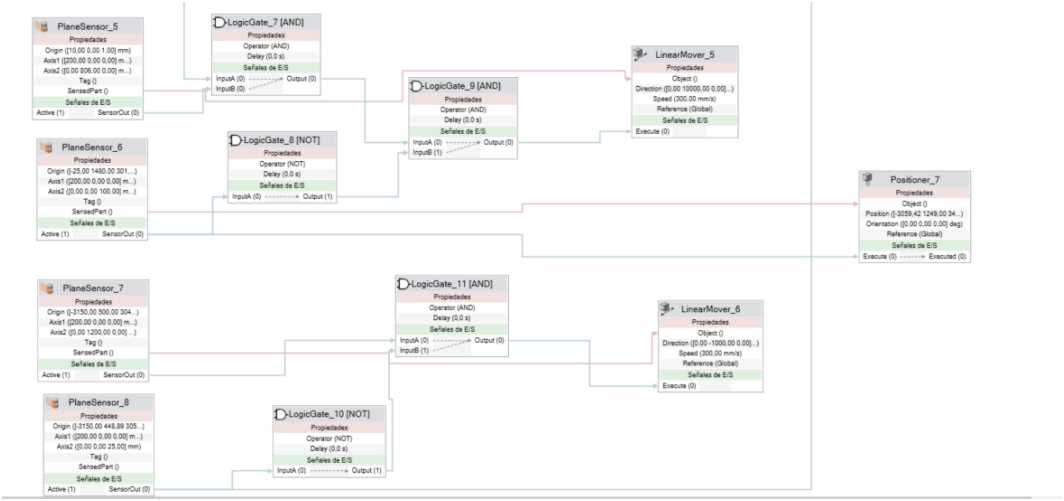



*Ilustración 82. Componente inteligente Recoger. Servicio Beber.*

Dentro del presente componte inteligente también, se diseña el proceso de devolución de la "comida" para poder realizar otra vez el servicio en el caso de que la persona lo desee. Esto consiste en el empleo de tres *Positioner* y tres componentes *Show* para mostrar los objetos que se escondieron en la boca de la persona para simular la alimentación.

Estos componentes se activan en el momento en que el sensor integrado en la cinta de recogida de alimentación y la salida digital "Recoger" se activan con un uno lógico. Los componentes son los siguientes:

- Positioner: es un componente que permite posicionar un objeto en una posición y una orientación indicada. En el momento que la señal de entrada *Execute* se activa, posiciona el objeto en las propiedades correspondientes respecto a la referencia elegida.


*Ilustración 83. Componente inteligente. Positioner.*

- Show: dispone de una señal de entrada que cuando se activa aparece el objeto del que se hace referencia en las propiedades.


*Ilustración 84. Componente inteligente. Show.*

#### 4.1.2.5 Tenedor

El objetivo de este componente inteligente es poder simular el acto de unión entre el cubierto ya sea un tenedor o una cuchara y el alimento, sólido o líquido, no se crea un componente inteligente de cuchara debido a que RobotStudio no permite la simulación de líquidos y quedaría de la misma manera que el proceso de un tenedor y sería duplicar el componente.

En cuanto al tenedor, no dispone de entradas ni salidas. Su funcionamiento se divide en dos secciones. La primera sección corresponde con el método de unión con el alimento, dicha unión se produce gracias a los bloques de sensores de colisión que más adelante se explicarán.

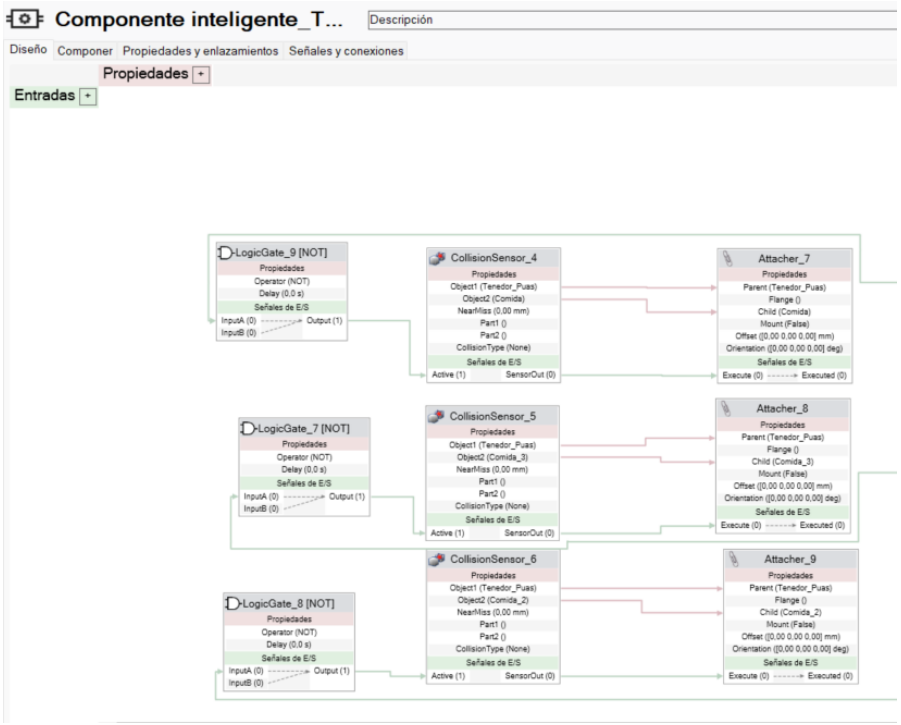



*Ilustración 85. Componente inteligente Tenedor parte 1.*

En la segunda sección del componente inteligente, se emplea un método bastante parejo, pero con la diferencia que en este es para la desunión del alimento del tenedor en el momento que se introduce en la boca de la persona. También, al no disponer de la simulación real del acto de masticar, se utilizan otros bloques para su simulación.

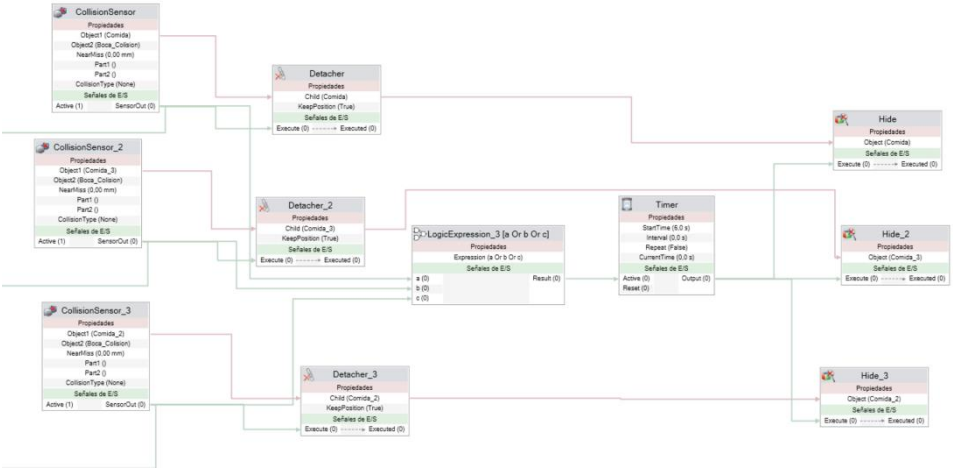



*Ilustración 86. Componente inteligente Tenedor parte 2.*

Ambas secciones están conectas debido a que para poder asegurar el correcto funcionamiento de la simulación no puede estar activo a la vez el componente *Detacher* y el componente *Attacher.* A continuación, se muestran las conexiones implementadas:

**Conexiones de E/S**

| Objeto de origen | Señal de origen | Objeto de destino | Señal o propiedad de destino |
|---|---|---|---|
| CollisionSensor_4 | SensorOut | Attacher_7 | Execute |
| CollisionSensor_5 | SensorOut | Attacher_8 | Execute |
| CollisionSensor_6 | SensorOut | Attacher_9 | Execute |
| CollisionSensor | SensorOut | LogicGate_9 [NOT] | InputA |
| LogicGate_9 [NOT] | Output | CollisionSensor_4 | Active |
| CollisionSensor_2 | SensorOut | LogicGate_7 [NOT] | InputA |
| LogicGate_7 [NOT] | Output | CollisionSensor_5 | Active |
| CollisionSensor_3 | SensorOut | LogicGate_8 [NOT] | InputA |
| LogicGate_8 [NOT] | Output | CollisionSensor_6 | Active |
| CollisionSensor_2 | SensorOut | Detacher_2 | Execute |
| CollisionSensor | SensorOut | Detacher | Execute |
| CollisionSensor_3 | SensorOut | Detacher_3 | Execute |
| Timer | Output | Hide_3 | Execute |
| Timer | Output | Hide_2 | Execute |
| Timer | Output | Hide | Execute |
| LogicExpression_3 [a Or b Or c] | Result | Timer | Active |
| CollisionSensor_3 | SensorOut | LogicExpression_3 [a Or b Or c] | c |
| CollisionSensor_2 | SensorOut | LogicExpression_3 [a Or b Or c] | b |
| CollisionSensor | SensorOut | LogicExpression_3 [a Or b Or c] | a |

*Ilustración 87. Conexiones componente inteligente Tenedor.*

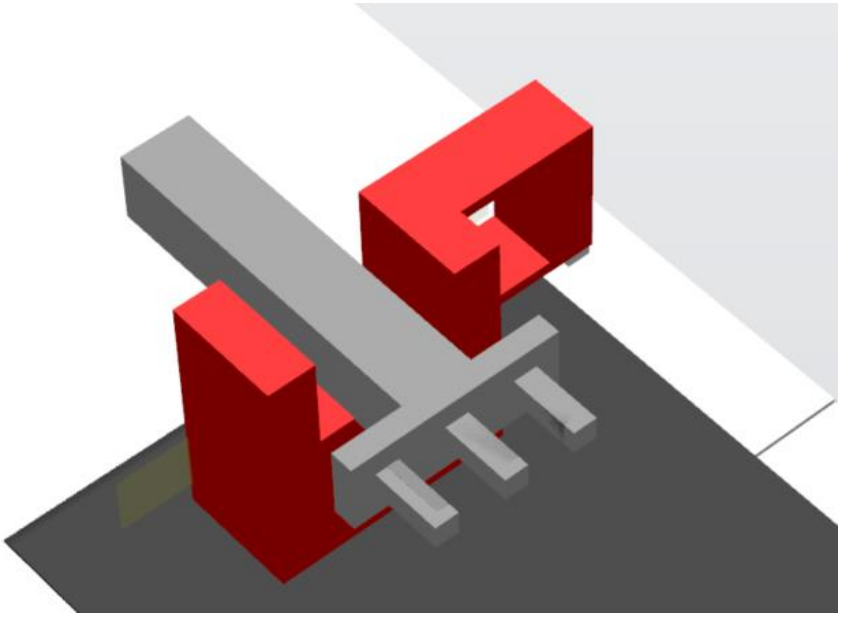


*Ilustración 88. Tenedor en la estación.*

Explicación correspondiente de los componentes restantes:

- CollisionSensor: es un sensor que detecta colisiones o aproximaciones entre los dos objetos seleccionados, salvo que solo se seleccione uno y entonces, ese objeto podrá ser detectado como colisión con toda la estación. Para poder emitir una salida lógica a nivel alto cuando se detecta una colisión, es necesario que la entrada lógica esté activa.

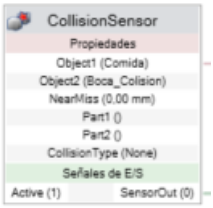



*Ilustración 89. Componente inteligente. CollisionSensor.*

- Timer: es un reloj que emite pulsos de salida a nivel lógico alto dependiendo de las características programadas. Se puede definir un intervalo de tiempo que este a su vez se repita o no. Cada vez que se cumple el intervalo emite el pulso de salida.


*Ilustración 90. Componente inteligente. Timer.*

- Hide: es el componente opuesto a Show, con este componente se pueden ocultar objetos cada vez que se activa la señal de entrada.


*Ilustración 91. Componente inteligente. Hide.*

#### 4.1.2.6 Entradas y salidas/conexiones de los controladores

Para el controlador correspondiente al CRB 15000 Gofa se han utilizado tres entradas y ocho salidas digitales que más adelante se explicará su funcionamiento. Este controlador prácticamente utiliza todas las salidas ya que hay algunas que son grupos de salidas y es el controlador principal. En cuanto a sus conexiones:

| Objeto origen | Señal origen | Objeto destino | Señal destino |
|---|---|---|---|
| **Controlador2** | Pinza | **Smart_gripper_linesensor_foga** | Pinza |

| | | | |
|---|---|---|---|
| **Componente inteligente_Conveyor** | Arrancar | **Controlador2** | Mover |
| **Controlador2** | Mov_Co | **Componente inteligente_Alimentar** | Comer |
| **Controlador2** | Mov_Be | **Componente inteligente_Alimentar** | Beber |
| **Componente inteligente_Alimentar** | Mov_Ok | **Controlador2** | Movim_Correcto |
| **Controlador2** | Mov_Pos | **Componente inteligente_ComerAle** | FinGiro |
| **Controlador2** | Posicion | **Componente inteligente_ComerAle** | Grado |
| **Componente inteligente_Persona** | Posicionado | **Controlador2** | Posicionado |
| **Controlador2** | Sentido_Giro | **Componente inteligente_ComerAle** | SentGir |
| **Controlador2** | Recoger | **Componente inteligente_Recoger** | ActivarRecog |
| **Controlador2** | Repetir_Proceso | **Componente inteligente_Acc Botones** | Repetir |
| **Controlador2** | Repetir_Proceso | **ControladorDoble** | Repetir |

*Tabla 2. Entradas y salidas controlador2.*

El controlador IRC5 dispone de las entradas y salidas que se muestran a continuación, cuatro entradas y dos salidas digitales.

| Objeto origen | Señal origen | Objeto destino | Señal destino |
|---|---|---|---|
| **ControladorDoble** | Pinza_doble_2 | **Smart_Gripper_Servo_Fingers_Doble_2** | Pinza_doble_2 |
| **ControladorDoble** | Pinza_doble_1 | **Smart_gripper_linesensor** | Pinza |
| **Componente inteligente_Acc Botones** | SelecciónBeber | **ControladorDoble** | Select_Beb |
| **Componente inteligente_Acc Botones** | SelecciónComer | **ControladorDoble** | Select_Com |
| **Componente inteligente_Recoger** | Preparado | **ControladorDoble** | Preparado |
| **Controlador2** | Repetir_Proceso | **ControladorDoble** | Repetir |

*Tabla 3. Entradas y salidas controladorDoble.*

## 4.2 Programación RAPID

La última parte que corresponde al software de RobotStudio para llevar a cabo la simulación, es la programación en RAPID que permite crear, modificar movimientos, rutinas, interrupciones, etc.

Para ello se deberá sincronizar con RAPID la estación o, ir creando módulos como se puede observar en la imagen inferior.

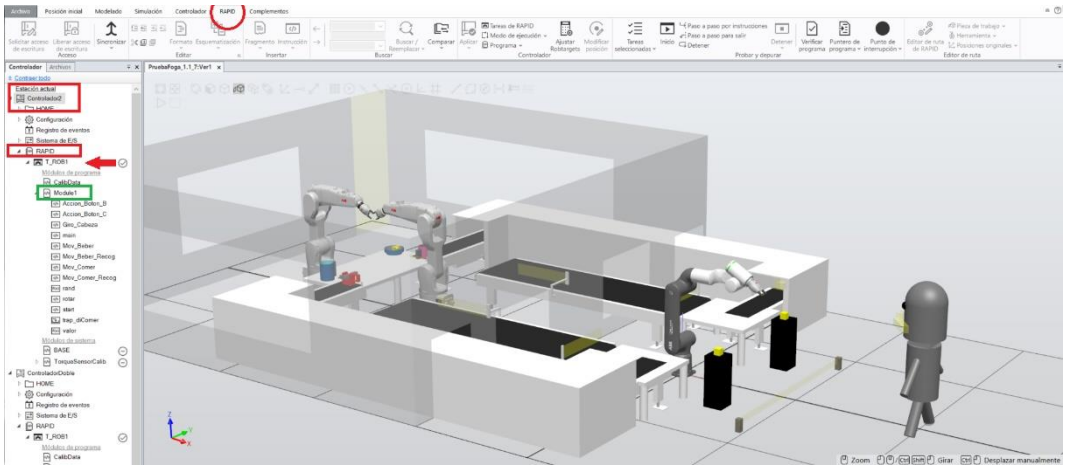


*Ilustración 92. Crear módulos RAPID.*

Debido a que el presente proyecto emplea dos controladores, se encuentran dos módulos principales. A continuación, se muestra y explica el primer controlador.

### 4.2.1 Controlador2

En él se puede encontrar la estructura principal del proyecto debido a que es el encargado de traducir los mensajes entre la persona y el robot.

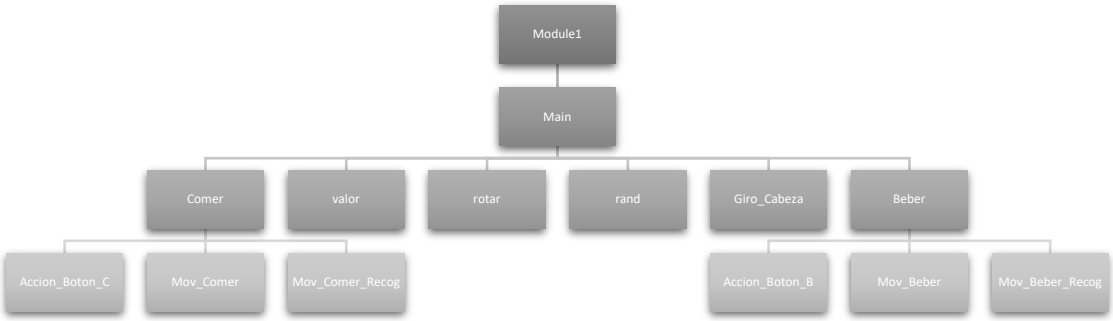



*Ilustración 93. Estructura módulos RAPID controlador2.*

Primero de todo, para poder llevar a cabo la programación del módulo se deben declarar las variables necesarias para el presente módulo mediante la sincronización de la estación con RAPID. Para ello, se realizará lo siguiente: Posición inicial, Sincronizar con RAPID y se elige lo que se quiere transmitir.

*Ilustración 94. Sincronizar con RAPID.*

Una vez pulsado Sincronizar con RAPID, aparecerá el siguiente recuadro donde se seleccionarán puntos, datos de herramienta y objetos de trabajo que se han creado para utilizar en RAPID.

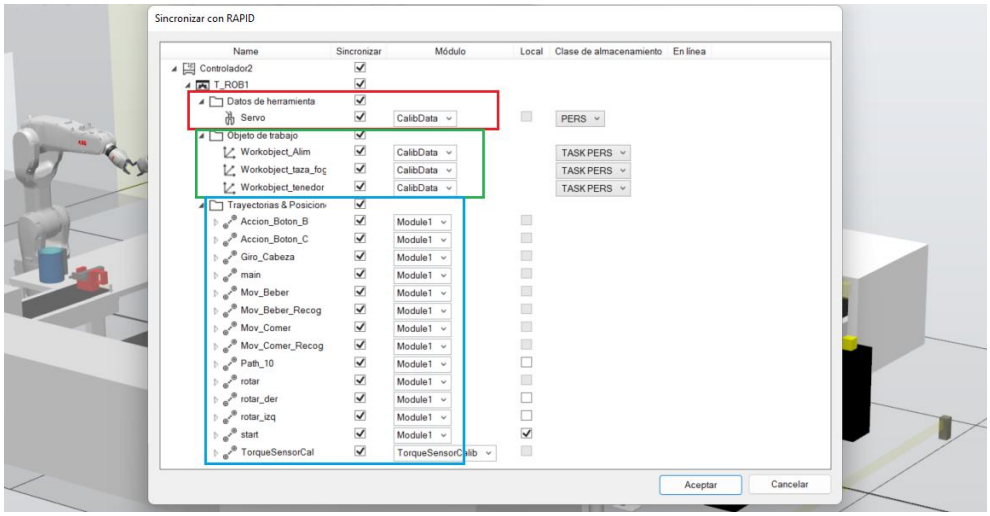



*Ilustración 95. Selección de objetos a sincronizar con RAPID.*

Todos los robtarget se han creado respecto a varios workobjects diseñados anteriormente, con el objetivo de separar los puntos de cada robot o tarea. Como se puede observar a continuación, para el CRB 15000 Gofa se crearon tres wobjdata, cada uno para su tarea correspondiente y también, se creó la herramienta de trabajo (Tooldata).

T_ROB1/Module1 | T_ROB1/CalibData

```
MODULE CalibData
    PERS tooldata Servo:=[TRUE,[[0,0,114.2],[1,0,0,0]],[0.215,[8.7,12.3,49.2],[1,0,0,0],0.00021,0.00024,0.00009]];
    TASK PERS wobjdata Workobject_taza_foga:=[FALSE,TRUE,"",[[-500,590,355],[0.5,-0.5,0.5,0.5]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata Workobject_tenedor:=[FALSE,TRUE,"",[[-565,-500,390],[0.5,0.5,0.5,-0.5]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata Workobject_Alim:=[FALSE,TRUE,"",[[1002.07,0,800],[0.707107,0,0.707107,0]],[[0,0,0],[1,0,0,0]]];
ENDMODULE
```

*Ilustración 96. Wobjdata y Tooldata. Controlador2.*

```
MODULE Module1

    CONST robtarget inicio:=[[597.468565782,0,837.782032303],[0.706638273,0,0.707574979,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_1:=[[-482.782027287,-83.111859452,-162.012832383],[0.894807681,-0.363774547,0.239762986,-0.097473092],[1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_2:=[[0,0,-87.157438849],[1,0,0,0],[1,-2,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_3:=[[-437.486663888,-0.000035911,-87.157480153],[1,0.000000132,0.000000014,0.000000124],[1,-2,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_4:=[[-458.722016747,-421.502770332,-35.075802176],[1,0.000000161,0.000000022,0.000000087],[1,1,-2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_5:=[[-458.722032314,-847.376628583,-55.535304701],[1,0.000000125,-0.000000006,0.00000014],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_boca:=[[712.068900091,0,800],[0.707106781,0,0.707106781,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_6:=[[-60.542357209,-587.764227631,144.424936003],[1,0.000000103,0.00000001,0.000000115],[0,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_7:=[[-16.380482628,-553.045488735,162.514683911],[1,0.0000001,0.000000003,0.000000169],[0,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_8:=[[-4.612319746,-550.857600319,195.890414431],[1,-0.0000001,-0.000000046,-0.000000036],[0,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_1_Ten:=[[13.352,115,19.023],[0.999206015,-0.000697578,0.039808443,0.001463188],[-2,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_2_Ten:=[[15,115,91.59],[1,0,0,0],[-2,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_3_Ten:=[[-426.618933014,136.149327583,19.023509367],[0.999206026,-0.000700185,0.039808139,0.001462981],[-2,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_4_Ten:=[[-426.61892242,687.487799313,19.023480011],[0.999206023,-0.000700019,0.039808203,0.001462978],[-1,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_5_Ten:=[[-207.747670102,747.184404197,19.023478637],[0.999206028,-0.000700131,0.039808062,0.0014631],[-1,-2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_6_Ten:=[[14.328609601,747.18440781,19.023473894],[0.999845362,0.016686716,-0.005337211,0.001523323],[-1,-2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_7_Ten:=[[14.329,747.184,148.452],[0.999999738,-0.000724312,0,0],[-1,-2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_8_Ten:=[[14.329,647.18,12.924],[0.9998044,-0.00072417,0.0197114,0.001448725],[-1,-2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_9_Ten:=[[14.329,647.18,73.95],[0.999999738,-0.000724312,0,0],[-1,-2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10_Ten:=[[-164.619186477,649.973816281,91.900820689],[0.999804387,-0.000729508,0.01971184,0.00144875],[-1,-2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_11_Ten:=[[-601.110333631,872.23720546,-7.961897335],[0.95830516,-0.285062386,0.019309822,-0.004217626],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Beb_Pos:=[[575,575,500],[0,0,1,0],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Beb_I:=[[575,575,470],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Beb_F:=[[575,575,420],[0,0,1,0],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Com_Pos:=[[575,-425,500],[0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Com_I:=[[575,-425,470],[0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Com_F:=[[575,-425,420],[0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_3_coger:=[[0,0,12.842561151],[1,0,0,0],[1,-2,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_9_dejar_10:=[[0,-550,233.449],[1,0,0,0],[0,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_A_10:=[[0,0,0],[1,0,0,0],[0,2,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

*Ilustración 97. Robtargets. Controlador2.*

Aparte de las variables anteriores, se han de crear variables locales para las distintas tareas como las variables de tipo PERS que permiten la comunicación con MATLAB ya que su valor se puede actualizar durante la ejecución del programa, también se emplean otras como Bool, Num o Pos para el resto de la programación.

```
local VAR bool nosalir:= false;
LOCAL VAR bool nosalir2:= FALSE;
LOCAL VAR num opcion:= 1;
LOCAL PERS num seed:= 27249;

PERS num ValorAleRec;
PERS num ValorNuevo;
PERS num Movimiento;
PERS pos altura;
PERS num Empezar:=0;
PERS num respuesta;
PERS num respuesta2;

VAR intnum iparada;
VAR intnum irespuesta;
VAR intnum irespuesta2;
VAR num contadorComer:= 0;
VAR num contadorBeber:= 0;
```

*Ilustración 98. Variables. Controlador.*

El siguiente paso corresponde con las funciones y procedimientos utilizados para la estación una vez definidas las variables.

La base del proyecto se encuentra en el procedimiento main del controlador2 (controlador Omnicore), pues es el encargado de administrar los principales movimientos, variables y señales de la estación para su funcionamiento. En él, primeramente, se inicializan todos los valores para que no den posibles fallos en la simulación, para acto seguido encapsular el resto dentro de un bucle condicional while que permita la simulación ininterrumpida como se busca hasta que se reciba la señal que rompa el bucle y finalice el programa.

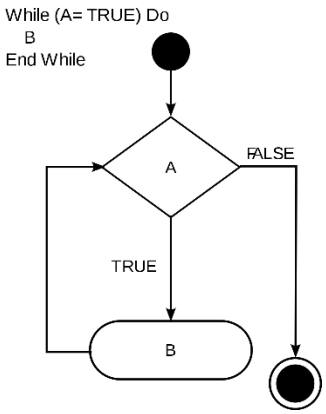



*Ilustración 99. Bucle While.*

Este procedimiento permite distinguir los movimientos en función de la señal recibida de la interfaz, en este caso es la llamada “Movimiento”, gracias al empleo de estructuras condicionales IF el controlador estará esperando a que la variable “Movimiento” actualice su valor para poder realizar el servicio asistencial de comer o de beber. Ambos movimientos tienen una estructura

similar pues, primero de todo activan la señal de salida “Arrancar” con el comando SetDO, el cual, cambia el valor lógico de una señal de salida.

Después, inicializan las correspondientes señales (Mov_co y Mov_be) de la misma forma que “Arrancar” y, se queda esperando a que se realicen los movimientos correspondientes al posicionamiento de la persona, para acto seguido llamar a la función Acción_Boton que ejecuta la acción para comunicar ambos controladores y se espera a que los robots industriales preparen el servicio y llegue a la posición correspondiente.

Una vez posicionado, se ha de conocer la posición de la boca, es decir, el ángulo en el que se encuentra la boca de la persona, para posteriormente simular el giro de la cabeza recibido de la interfaz. Se activa la señal que permite el movimiento del robot colaborativo para realizar el servicio en busca de la posición de la persona.

```
IF Movimiento=1 OR Movimiento=2 THEN
    TPWrite"movimiento: "\Num:=Movimiento;
    IF Movimiento=1 THEN
        SetDO Arrancar,1;
        WaitTime 1;
        WaitDI Posicionado,1;
        WaitTime 1;
        setdo Mov_Co,1;
        SetDO mov_be,0;
        WaitDI Movim_Correcto,1;
        Accion_Boton_C;
        WaitDI Mover,1;
        Nueva_Posicion;
        SetDO Mov_Pos,1;
        Mov_Comer;
        rotar;
        Mov_Comer_Recog;
        SetDO Recoger, 1;
```

*Ilustración 100. Procedimiento Main. Alimentar.*

Sin salir del bucle condicional IF se crea otro bucle condicional WHILE que permitirá esperar a si la persona quiere repetir o no, y, por ende, finalizar el programa o reinicializar los valores para empezar de nuevo un servicio. Está señal se leerá desde la interfaz mediante la variable PERS “respuesta”.

```
WHILE nosalir2 DO
    IF respuesta=0 THEN
        nosalir:=FALSE;
        nosalir2:=FALSE;
        EXIT;
    ELSEIF respuesta=1 THEN
        SetDO Repetir_Proceso,1;
        WaitDI Posicionado,1;
        nosalir2:=FALSE;
    ENDIF
    WaitTime 0.1;
ENDWHILE
```

*Ilustración 101. Procedimiento repetir proceso.*

Con la misma estructura se encuentra el caso de hidratarse, también se tiene una tercera opción que es totalmente aleatorio el servicio asistencial pues mediante el empleo del comando rand () en AppDesigner se escoge el servicio a realizar.

Al final del procedimiento Main se reiniciarán todas las variables a su valor original y se parará la ejecución de la simulación.

```
SetDO pinza, 0;
SetDO Repetir_Proceso, 0;
SetDO Mov_Be,0;
SetDO Mov_Co,0;
SetDO Mov_Pos,0;
SetGO Posicion,0;
SetDO Sentido_Giro, 0;
SetDO Recoger, 0;
Stop;
```

*Ilustración 102. Reiniciar variables.*

#### 4.2.1.1 Accionamiento de Botones

En la figura 103 se muestra el procedimiento creado para pulsar el botón correspondiente al servicio demandado y así poder activar la señal de selección en el controladorDoble como se observa en la figura XX (lógica de estación). Para realizar el movimiento se han empleado instrucciones “MoveJ”, “MoveL”, ”MoveLdo” y “MoveJdo”, estas dos últimas permiten establecer salidas digitales que en este caso primeramente se cierra la pinza para pulsar el botón, estableciendo el valor lógico “1” y, después se reestablece su valor.

```
PROC Accion_Boton_C()
    MoveJdo Target_Boton_Com_Pos,v500,z50,Servo\WObj:=wobj0, pinza, 1;
    MoveL Target_Boton_Com_I,v500,z50,Servo\WObj:=wobj0;
    MoveL Target_Boton_Com_F,v400,fine,Servo\WObj:=wobj0;
    MoveLdo Target_Boton_Com_I,v500,z50,Servo\WObj:=wobj0,pinza,0;
    MoveJ inicio,v500,fine,Servo\WObj:=wobj0;
ENDPROC
```

*Ilustración 103. Procedimiento accionar botón. Alimentar.*

Del mismo modo se crea la función correspondiente para activar el botón del servicio asistencial de hidratarse.

```
PROC Accion_Boton_B()
    MoveJdo Target_Boton_Beb_Pos,v500,z50,Servo\WObj:=wobj0,pinza,1;
    MoveL Target_Boton_Beb_I,v500,z50,Servo\WObj:=wobj0;
    MoveL Target_Boton_Beb_F,v300,fine,Servo\WObj:=wobj0;
    MoveLdo Target_Boton_Beb_I,v500,z50,Servo\WObj:=wobj0,pinza,0;
    MoveJ inicio,v500,fine,Servo\WObj:=wobj0;
ENDPROC
```

*Ilustración 104. Procedimiento accionar botón. Hidratar.*

#### 4.2.1.2 Movimiento servicio hidratarse

Para la realización de este servicio es muy importante la estabilidad y la realización de un movimiento horizontal rectilíneo durante el transporte de la taza por lo que en esta instrucción se emplean en su mayoría "MoveL" para las trayectorias creadas manualmente en la estación para garantizar el correcto funcionamiento sin derramar líquido. Se divide en dos procedimientos, el primero corresponde al acto de recoger la taza de la cinta transportadora y situarse en el punto previo al servicio, es decir, en posición inicial con la taza.

```
PROC Mov_Beber()
    MoveJ Target_taza_1,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_2,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveLdo Target_taza_3_coger,v300,fine,Servo\WObj:=Workobject_taza_foga, pinza, 1;
    WaitTime 2;
    MoveL Target_taza_2,v500,fine,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_3,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_4,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_5,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveJ inicio,v500,fine,Servo\WObj:=wobj0;

ENDPROC
```

*Ilustración 105. Procedimiento 1 hidratar. Controlador2.*

Entre medias de ambos procedimientos se ejecutará rotar que más adelante se comentará ya que es un procedimiento común a ambos servicios y, por tanto, se explica aparte. La segunda parte del servicio hidratarse corresponde con la parte de recogida de los utensilios por ello, mediante el empleo de instrucciones de movimiento y las posiciones creadas en la estación, después de servir la bebida se coloca la taza en la cinta transportadora de retorno y se desactiva la señal digital de salida de la pinza para soltar la taza dejando el valor lógico a "0".

```
PROC Mov_Beber_Recog()
    MoveJ Target_taza_5,v300,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_6,v300,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_7,v300,fine,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_8,v300,fine,Servo\WObj:=Workobject_taza_foga;
    MoveLdo Target_taza_9_dejar_10,v300,fine,Servo\WObj:=Workobject_taza_foga,pinza,0;
    WaitTime 2;
    MoveL Target_taza_8,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_7,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_6,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_5,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveJ inicio,v500,fine,Servo\WObj:=wobj0;
ENDPROC
```

*Ilustración 106. Procedimiento 2 hidratar. Controlador2.*

#### 4.2.1.3 Movimiento servicio alimentarse

De la misma manera que se ha comentado anteriormente el movimiento servicio hidratarse (hipervínculo) se obtiene que para la alimentación también se divide en dos procedimientos y, es necesario realizar movimientos estables y seguros para no desaprovechar la comida. El primer procedimiento permite al robot colaborativo alcanzar el porta-tenedor situado en la cinta transportadora, colocarlo en la mesa de apoyo para poder soltarlo y coger el

tenedor para alimentar a la persona. Como se observa en la imagen inferior 107 se emplean varios "MoveLdo" para establecer los valores lógicos correspondientes a los movimientos de abrir o cerrar la pinza. También se establecen unos tiempos de espera después de estos para asegurar el agarre de los utensilios.

```
PROC Mov_Comer()
    MoveL Target_1_Ten,v500,z50,Servo\WObj:=Workobject_tenedor;
    MoveLdo Target_2_Ten,v300,fine,Servo\WObj:=Workobject_tenedor,pinza,1;
    WaitTime 2;
    MoveL Target_1_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
    MoveL Target_3_Ten,v300,z50,Servo\WObj:=Workobject_tenedor;
    MoveL Target_4_Ten,v300,z50,Servo\WObj:=Workobject_tenedor;
    MoveL Target_5_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
    MoveL Target_6_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
    MoveLdo Target_7_Ten,v300,fine,Servo\WObj:=Workobject_tenedor, pinza,0;
    WaitTime 1;
    MoveL Target_6_Ten,v500,z50,Servo\WObj:=Workobject_tenedor;
    MoveL Target_8_Ten,v500,z50,Servo\WObj:=Workobject_tenedor;
    MoveLdo Target_9_Ten,v300,fine,Servo\WObj:=Workobject_tenedor, pinza,1;
    WaitTime 1;
    MoveL Target_10_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
    MoveL Target_11_Ten,v300,z100,Servo\WObj:=Workobject_tenedor;
    MoveL inicio,v300,fine,Servo\WObj:=wobj0;
ENDPROC
```

*Ilustración 107. Procedimiento 1 alimentar. Controlador2.*

Al igual que en el servicio de hidratación, entre medias de los dos movimientos de alimentación, se realiza el procedimiento rotar. El segundo movimiento es el encargado de la recogida del utensilio y su colocación en el porta-tenedor mediante instrucciones de movimiento "MoveL" y "MoveLdo", gracias a esta última se establece el valor lógico de salida "0" a la pinza a la hora de depositar el tenedor en el porta-tenedor.

```
PROC Mov_Comer_Recog()
    MoveL Target_11_Ten,v300,z100,Servo\WObj:=Workobject_tenedor;
    MoveL Target_10_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
    MoveLdo Target_9_Ten,v300,fine,Servo\WObj:=Workobject_tenedor, pinza,0;
    waittime 1;
    MoveL Target_10_Ten,v500,fine,Servo\WObj:=Workobject_tenedor;
    MoveL Target_11_Ten,v500,z100,Servo\WObj:=Workobject_tenedor;
    MoveL inicio,v500,fine,Servo\WObj:=wobj0;
ENDPROC
```

*Ilustración 108. Procedimiento 1 alimentar. Controlador2.*

#### 4.2.1.4 Procedimiento rotar

Para poder simular el reconocimiento facial de la persona se ha diseñado el siguiente procedimiento que permite alimentar o hidratar a una persona en la posición en la que se encuentre. Para ello se ha necesitado el uso de Robtargets además de un punto auxiliar. Primero de todo, se declaran los robtarget y las coordenadas para cada uno. El punto A es el punto auxiliar que se ha creado en la estación en la posición aproximada donde se detendrá la

persona para realizar el servicio, el punto B corresponde con el primer punto de aproximación, el punto C es el segundo punto de aproximación y el último punto, D, sirve para la rotación del servicio correspondiente. También, se actualiza el valor del ángulo de rotación que se recibe a través de la interfaz "ValorAleRec" con "ValorNuevo".

```
PROC rotar()

VAR robtarget rt_PointB;
VAR robtarget rt_PointC;
VAR robtarget rt_PointD;
VAR pose ps_PointA;
VAR pose ps_PointB;
VAR pose ps_PointC;
VAR pose ps_PointD;
VAR num valorZ;
VAR num valorY;
VAR num valorX;
VAR num n_RotationAngle ;

n_RotationAngle:= ValorNuevo;
ps_PointA := [target_a_10.trans,target_a_10.rot];
```

*Ilustración 109. Variables procedimiento rotar.*

En la siguiente imagen se puede apreciar de manera más visual la creación del punto auxiliar (en rojo), y los posibles puntos de aproximación que puede tomar B (azul) y C (negro) dependiendo del ángulo de giro que realice el cuello de la persona.

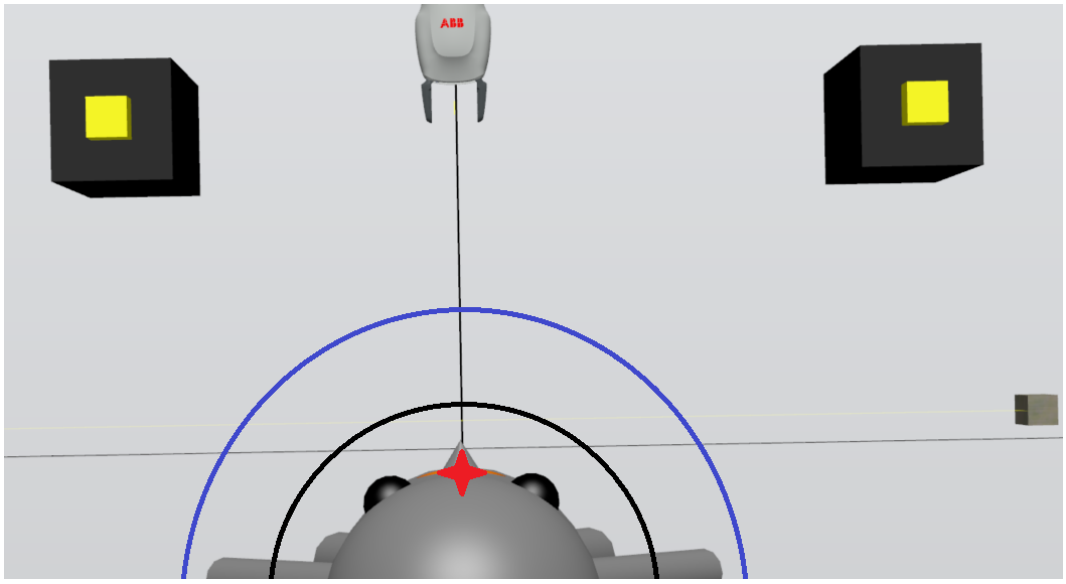


*Ilustración 110. Ángulos de servicio en procedimiento rotar.*

Como se comenta en anteriores apartados, se ha de dividir en dos sentidos el ángulo de giro recibido de la persona, por ello, se toman los valores de 0 a 40

para el sentido positivo de las agujas del reloj en el lado derecho de la imagen superior y de 40 a 80 para el sentido negativo para el lado izquierdo.

Para el cálculo de la posición de la herramienta del robot en la boca de la persona ha sido necesario emplear conocimientos de trigonometría como se muestra a continuación.

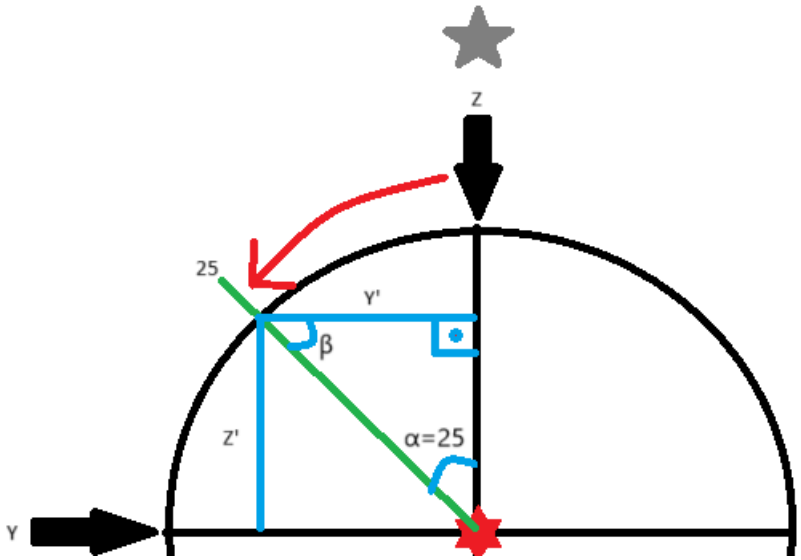



*Ilustración 111. Cálculo de la posición.*

Como se puede observar en imagen 111 primero de todo se necesita conocer el ángulo β, es decir, valorZ. Para obtener su valor se ha de saber que la suma de los ángulos interiores de un triángulo siempre debe sumar 180º, debido a ello se conoce el valor del ángulo recto (90º) y el valor procesado de la interfaz como "n_RotationAngle", conocidos estos valores ya se puede obtener el valor de β=180-90-n_RotationAngle.

Una vez calculados los ángulos correspondientes, el siguiente paso es conocer la posición de Z' e Y' respecto de la persona por ello, se aplica el teorema del seno. Por ejemplo, para el punto B, se sabe que el radio de la circunferencia adecuado para su ejecución es de 360 milímetros, correspondería con la línea verde de la imagen XX y su ángulo opuesto es el ángulo recto. Como se conocen el resto de los ángulos del triángulo, la forma de proceder es sencilla.

$$\frac{360}{Sin(90)} = \frac{Z'}{Sin(\beta)}$$

*Ecuación 1. Teorema del Seno.*

Se despeja el valor de Z',

$$Z' = \frac{360 * Sin(\beta)}{Sin(90)} = 360 * Sin(\beta)$$

*Ecuación 2. Despejar el valor requerido del Teorema del Seno.*

De la misma manera, se procede para obtener el valor de Y'. Obtenidas las posiciones de cada punto nuevo, se han de calcular las nuevas posiciones basadas en el punto auxiliar para ello se hace uso de la función "PoseMult" de RAPID que sirve para realizar operaciones de transformación o cálculos geométricos en el espacio de coordenadas del robot. También se ha de generar la orientación a partir de ángulos de Euler para poder multiplicarlos con el punto auxiliar mediante OrientZYX.

```
IF (ValorALErec>40 AND valorALErec<=80) THEN

    valorZ:=Sin(180-90-n_RotationAngle);
    valorY:=Sin(n_RotationAngle);
    valorX:= altura.z - 80;

    ps_PointB := PoseMult( [[(800-valorx),-(360*valory),-(360*valorz)],OrientZYX(0,0,-n_RotationAngle)], ps_PointA);
    ps_PointC := PoseMult( [[(800-valorx),-(340*valory),-(340*valorz)],OrientZYX(0,0,-n_RotationAngle)], ps_PointA);
    IF Mov_Be=1 AND Mov_Co=0 THEN
        ps_PointD := PoseMult(  [[(800-valorx-20),-(290*valory),-(290*valorz)],OrientZYX(0,0,-n_RotationAngle)], ps_PointA);
    ELSEIF Mov_Be=0 AND Mov_Co=1 THEN
        ps_PointD := PoseMult( [[(800-valorx-10),-(290*valory),-(290*valorz)],OrientZYX(0,0,-n_RotationAngle)], ps_PointA);
    ENDIF

ELSEIF(valorALErec <= 40 AND valorALErec>=0) THEN

    valorZ:=Sin(180-90-n_RotationAngle);
    valorY:=Sin(n_RotationAngle);
    valorX:= altura.z - 80;

    ps_PointB := PoseMult( [[(800-valorX),390*valory,-(390*valorz)],OrientZYX(0,0,n_RotationAngle)], ps_PointA);
    ps_PointC := PoseMult( [[(800-valorx),(330*valory),-(330*valorz)],OrientZYX(0,0,n_RotationAngle)], ps_PointA);
    IF Mov_Be=1 AND Mov_Co=0 THEN
        ps_PointD := PoseMult(  [[(800-valorx-20),(290*valory),-(290*valorz)],OrientZYX(0,0,n_RotationAngle)], ps_PointA);
    ELSEIF Mov_Be=0 AND Mov_Co=1 THEN
        ps_PointD := PoseMult( [[(800-valorx-10),(290*valory),-(290*valorz)],OrientZYX(0,0,n_RotationAngle)], ps_PointA);
    ENDIF
ENDIF
```

*Ilustración 112. Cálculos de las posiciones en función del ángulo en procedimiento rotar.*

Una vez realizado todos los pasos anteriores se han de actualizar las posiciones de translación y rotación de los robtarget B, C y D con los puntos creados.

```
rt_PointB.trans := ps_pointb.trans;
rt_PointB.rot := ps_pointb.rot;
rt_PointC.trans := ps_pointC.trans;
rt_PointC.rot := ps_pointC.rot;
rt_PointD.trans := ps_pointd.trans;
rt_PointD.rot := ps_pointd.rot;

MoveL inicio,v300,fine,Servo\WObj:=wobj0;
MoveL rt_PointB,v100,fine, Servo\WObj:=Workobject_Alim;
MoveL rt_PointC,v100,fine,Servo\WObj:=Workobject_Alim;
MoveL rt_PointD,v100,fine,Servo\WObj:=Workobject_Alim;
```

*Ilustración 113. Actualización Robtargets en procedimiento rotar.*

Por último, en la función rotar en el caso del servicio asistencial de hidratarse se ha de inclinar la taza lo suficiente para que la persona pueda beber adecuadamente. Esto se consigue realizando un desplazamiento respecto de la herramienta como se observa en la imagen inferior.

```
IF Mov_Be = 1 AND Mov_Co = 0 THEN

    Movel offs (rt_PointD,20,0,0),v100,fine,Servo\WObj:=Workobject_Alim;
    movel RelTool (rt_Pointd,-10,0,10,\Ry:=30),v100,fine,Servo\WObj:=Workobject_Alim;

ENDIF

WaitTime 2;
MoveL rt_PointC,v100,fine,Servo\WObj:=Workobject_Alim;

ENDPROC
```

*Ilustración 114. Rotación hidratarse en procedimiento rotar.*

#### 4.2.1.5 Procedimiento Nueva_Posicion

Como se menciona en el procedimiento rotar se ha empleado la lógica del giro positivo de las agujas del reloj para el lado derecho de 0 a 40 y el giro negativo para el lado izquierdo de 40 a 80, estos valores se reciben de la interfaz de usuario.

Debido a que durante la programación se emplean datos de tipo enteros se debe truncar el valor a solo la parte entera gracias a la función Trunc (). Para actualizar el valor del grupo de salidas digitales “Posición” que se emplea en la lógica de estación, es necesario quedarse solo con valores de 0 a 40, ya que el sentido de giro se pasa a través de la salida digital Sentido_Giro.

```
PROC Nueva_Posicion()

    IF (ValorALErec>40 AND valorALErec<=80) THEN
                    ValorAleRec:=Trunc(ValorAleRec);
                    ValorNuevo:= (valoraleRec-40);
                    SetDO Sentido_Giro, 1;
                    SetGO Posicion, ValorNuevo;
   ELSEIF (valorALErec <= 40 AND valorALErec>=0) THEN
                    ValorAleRec:=Trunc(ValorAleRec);
                    ValorNuevo:=valorALErec;
                    SetDO Sentido_Giro, 0;
                    SetGO Posicion, ValorNuevo;
    ENDIF
ENDPROC
```

*Ilustración 115. Procedimiento Nueva_Posicion.*

### 4.2.2 ControladorDoble

Este controlador es el encargado de coordinar los movimientos de los dos robots industriales debido a ello poseen estructuras muy similares y señales digitales compartidas. A continuación, se muestra el diagrama de bloques que contiene todos los procedimientos y funciones utilizadas.

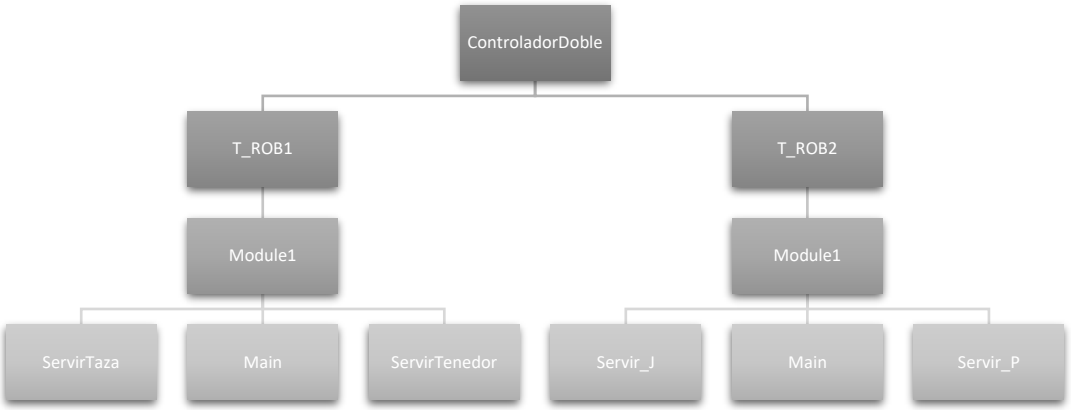



*Ilustración 116. Estructura módulos RAPID ControladorDoble.*

De la misma manera que en el Controlador2 (Ilustración 93) se sincroniza la estación con RAPID para transferir las posiciones y trayectorias creadas para cada robot. Para cada robot se ha creado un objeto de trabajo distinto y su herramienta correspondiente (“Tooldata”).

Como se puede observar en el diagrama de bloques se dispone de dos módulos separados, uno para cada robot, aunque trabajen de manera conjunta.

#### 4.2.2.1 T_ROB1

En lo que respecta al primer robot, es el encargado de actuar como nexo entre la zona de elaboración y la zona de servicio, debido a que posiciona cada utensilio en la cinta transportadora adecuada.

Primero de todo en el procedimiento main se han de inicializar las señales digitales de salida, así como restablecer el valor de la pinza. Una vez realizado se quedará esperando a que la señal digital de entrada se encuentre en el valor lógico '1' que indicará que la estación está preparada y esperando la orden de un servicio.

Dentro del bucle se irá actualizando hasta que una de las entradas digitales que se accionan mediante los (pulsadores) se activen y llamen a la función correspondiente. Si se recibe la orden desde la interfaz de usuario de querer repetir, se inicializarán de nuevo los valores y se quedarían esperando la señal de servicio otra vez.

```
PROC main()
    ConfJ\Off;
    ConfL\Off;
    SetDO Pos1R2,0;
    SetDO Pos2R2,0;
    SetDO Pos3R2,0;
    SetDO Pos4R2,0;
    SetDO pinza_doble_1,0;

    WaitDI Preparado,1;

    WHILE TRUE DO

        IF Select_Beb=1 AND Preparado=1 THEN
            ServirTaza;
            MoveJ Target_Inicio_R2,v300,fine,Servo\WObj:=Workobject_taza_doble;
        ENDIF
        IF Select_Com=1 AND Preparado=1 THEN
            ServirTenedor;
            MoveJ Target_Inicio_R2,v300,fine,Servo\WObj:=Workobject_taza_doble;
        ENDIF

        IF Repetir=1 THEN
            SetDO Pos1R2,0;
            SetDO Pos2R2,0;
            SetDO Pos3R2,0;
            SetDO Pos4R2,0;
            SetDO pinza_doble_1,0;
        ENDIF

        ConfJ\Off;
        ConfL\Off;
        WaitTime 0.01;
    ENDWHILE
```

*Ilustración 117. Procedimiento main robot_1. ControladorDoble.*

## ServirTaza

Este procedimiento se ejecutará cuando las señales de entrada "Select_Beb" y "Preparado" se activen con el valor lógico "1". Con el empleo de instrucciones de movimiento tipo "Move" ya sea de forma lineal o mediante el movimiento de ejes se consigue el movimiento requerido. Con el empleo de la función "MoveLdo" se establecerán los valores de las salidas digitales convenientes como abrir o cerrar la pinza para coger la taza.

Cada movimiento tiene que estar coordinado con el segundo robot y para ello se emplean las salidas digitales Pos1R2, Pos2R2… que indicarán el posicionamiento correcto de cada robot para la elaboración.

```
PROC ServirTaza()
    MoveJ Target_inicio_doble_1,v500,z50,Servo\WObj:=wobj0;
    MoveL Target_10_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_11_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_12_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_13_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_14_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_15_CogerTaza,v500,fine,Servo\WObj:=Workobject_taza_doble;
    MoveLdo Target_16_CogerTaza,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_doble_1,1;
    WaitTime 2;
    MoveL Target_20_ServirTaza,v500,fine,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_21_ServirTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_22_ServirTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    WaitDO Pos4R2,1;
    MoveLdo Target_23_ServirTaza,v300,fine,Servo\WObj:=Workobject_taza_doble,Pos3R2,1;
    Waitdo Pos4R2,0;
    MoveL Target_13_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_25_ServirTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_26_ServirTaza,v500,fine,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_190,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveLdo Target_200,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_doble_1,0;
    WaitTime 1;
    MoveL Target_190,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_170,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_26_ServirTaza,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_27_ServirTaza,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveJ Target_28_ServirTaza,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveJ Target_inicio_doble_1,v500,fine,Servo\WObj:=wobj0;
ENDPROC
```

*Ilustración 118. Procedimiento ServirTaza.*

## ServirTenedor

Para coger el tenedor y clavar la comida del plato, el procedimiento es muy parejo al de servir la taza, primero ha de esperar a la activación de las entradas digitales "Select_Com" y "Preparado" y después realizar las instrucciones de movimiento.

Primero de todo, coge el tenedor y lo lleva a la posición de espera para que el plato se sitúe, después, clava el tenedor y lo posiciona en el porta-tenedor para colocarlo en la cinta transportadora y poder realizar el servicio.

```
PROC ServirTenedor()
    MoveJ Target_inicio_doble_1,v500,fine,Servo\WObj:=wobj0;
    MoveLdo Target_40_CogerTene,v300,fine,Servo\WObj:=Workobject_taza_doble, pinza_doble_1,1;
    WaitTime 1.5;
    MoveL Target_41_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_42_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveLdo Target_43_CogerTene,v500,fine,Servo\WObj:=Workobject_taza_doble,Pos2R2,1;
    WaitDO Pos1R2,1;
    MoveL Target_44_CogerTene,v100,fine,Servo\WObj:=Workobject_taza_doble;
    WaitTime 1;
    MoveLdo Target_43_CogerTene,v500,fine,Servo\WObj:=Workobject_taza_doble,Pos2R2,0;
    MoveL Target_46_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_41_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveLdo Target_40_CogerTene,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_doble_1,0;
    WaitTime 1;
    MoveL Target_47_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_48_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveLdo Target_49_CogerTene,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_doble_1,1;
    WaitTime 1;
    MoveL Target_50_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_51_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_52_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveLdo Target_53_CogerTene,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_doble_1,0;
    waittime 1;
    MoveL Target_54_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_55_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_57_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_58_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveL Target_59_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
    MoveJ Target_inicio_doble_1,v500,z100,Servo\WObj:=wobj0;
ENDPROC
```

*Ilustración 119. Procedimiento ServirTenedor.*

#### 4.2.2.2 T_ROB2

La estructura del segundo robot se organiza de la misma manera que el primero, pues, como se ha comentado anteriormente deben de trabajar a la vez.

En el procedimiento principal al arrancar la estación se inicializarán las señales digitales de salida y la pinza del robot al valor lógico “0” y se quedará esperando a que la estación se encuentre preparada para iniciar el servicio, es decir, esperando a que la señal digital de entrada se active a “1”.

Una vez activada, entrará en el bucle While y comenzará a recorrerse hasta que una de las señales digitales de entrada sea activada a través del accionamiento del botón correspondiente a uno de los servicios.

```
PROC main()
    ConfJ\Off;
    ConfL\Off;
    SetDO Pos1R2,0;
    SetDO Pos2R2,0;
    SetDO Pos3R2,0;
    SetDO Pos4R2,0;
    SetDO pinza_doble_2,0;

    WaitDI Preparado,1;

    WHILE TRUE DO

        IF Select_Beb=1 AND Preparado=1 THEN
            Servir_J;
            MoveJ Target_Ini_R3,v300,fine,Servo\WObj:=Workobject_Servir;

        ENDIF
        IF Select_Com=1 AND Preparado=1 THEN
            Servir_P;
            MoveJ Target_Ini_R3,v300,fine,Servo\WObj:=Workobject_Servir;

        ENDIF

        IF Repetir=1 THEN

            SetDO Pos1R2,0;
            SetDO Pos2R2,0;
            SetDO Pos3R2,0;
            SetDO Pos4R2,0;
            SetDO pinza_doble_2,0;
        ENDIF
        waittime 0.1;
    ENDWHILE


ENDPROC
```

*Ilustración 120. Procedimiento main robot_2. ControladorDoble.*

## Servir_J

El movimiento de verter el líquido de la jarra en la taza se realiza cuando ambas entradas digitales “Select_Beb” y “Preparado” se activan. Las trayectorias de este procedimiento se realizan con instrucciones de movimiento “MoveL”, “MoveJ” y “MoveLdo”, esta última permite establecer valores a señales digitales de salida. Con “MoveLdo” se asigna el valor lógico a la pinza para abrir o cerrar y también, se emplea para avisar al otro robot que se ha posicionado y se encuentra esperando en la posición correcta.

```
PROC Servir_J()
    MoveL Target_Inicio_R3,v300,fine,Servo\WObj:=Workobject_Servir;
    MoveJ Target_3_1,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveJ Target_3_2,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_3,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveLDO Target_3_4,v500,fine,Servo\WObj:=Workobject_Servir,pinza_doble_2,1;
    WaitTime 2;
    MoveL Target_3_3,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_5,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveLDo Target_3_6,v500,fine,Servo\WObj:=Workobject_Servir,Pos4R2,1;
    waitDo Pos3R2,1;
    MoveL Target_3_7,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_8,v20,fine,Servo\WObj:=Workobject_Servir;
    WaitTime 2;
    MoveLdo Target_3_7,v500,fine,Servo\WObj:=Workobject_Servir,Pos4R2,0;
    MoveL Target_3_6,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_5,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_3,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveLDO Target_3_4,v500,fine,Servo\WObj:=Workobject_Servir,pinza_doble_2,0;
    WaitTime 2;
    MoveL Target_Inicio_R3,v300,fine,Servo\WObj:=Workobject_Servir;
ENDPROC
```

*Ilustración 121. Procedimiento Servir_J.*

## Servir_P

Este procedimiento se diseña con la misma estructura que el anterior, pero creando el movimiento de colocar el plato en la posición adecuada para que el tenedor pueda recoger la comida. Se activará en el momento que el robot colaborativo accione el botón del servicio alimentarse y la estación se encuentre preparada, es decir, ambas señales digitales de entradas con valor lógico a "1".

```
PROC Servir_P()
    MoveJ Target_Inicio_R3,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveJ Target_60,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveLDo Target_10,v500,fine,Servo\WObj:=Workobject_Servir,pinza_doble_2,1;
    WaitTime 1.5;
    MoveL Target_20,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_30,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_40,v500,fine,Servo\WObj:=Workobject_Servir;
    WaitDO Pos2R2,1;
    MoveLDo Target_50,v500,fine,Servo\WObj:=Workobject_Servir,Pos1R2,1;
    WaitDO Pos2R2,0;
    MoveJ Target_60,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveLDo Target_10,v500,fine,Servo\WObj:=Workobject_Servir,pinza_doble_2,0;
    WaitTime 1;
    MoveL Target_60,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_Inicio_R3,v500,fine,Servo\WObj:=Workobject_Servir;
ENDPROC
```

*Ilustración 122. Procedimiento Servir_P.*

## 4.3 Comunicación OPC UA

En el presente proyecto se ha empleado el protocolo de comunicación que se basa en una arquitectura unificada, OPC UA. Este protocolo permite el intercambio de datos en la comunicación industrial de una manera segura e independiente, intercambiando información en plataformas de hardware de distintos proveedores y distintos S.O.

Su funcionamiento se basa en el tipo Cliente-Servidor, lo que va a permitir intercambiar información de variables entre RobotStudio y MATLAB ya que ambos son clientes y los datos se transfieren a través de la aplicación ABB IRC5 OPC.

Iniciando ABB IRC5 OPC, lo primero que se debe realizar es crear un nuevo alias donde se podrá escoger los controladores que se van a utilizar. Para que aparezcan los posibles controladores, se ha de pulsar "Scan" que escaneará las rutas en busca de los controladores, una vez han cargado se seleccionan y se acciona "Create. Hasta este punto ya estarían creados y asignados los alias para cada controlador.

1. Añadir nuevo alias.

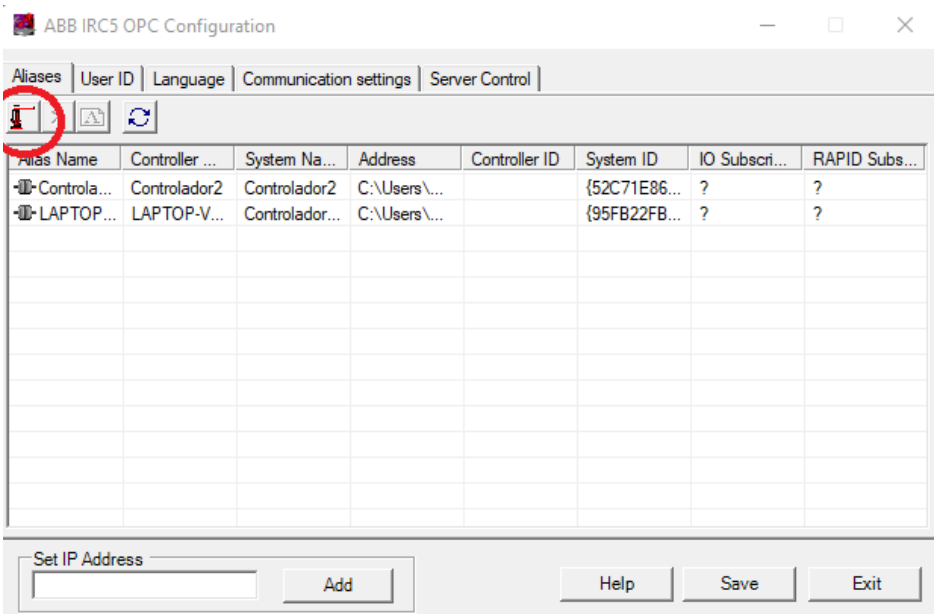



*Ilustración 123. Añadir alias.*

2. Escanear controladores.

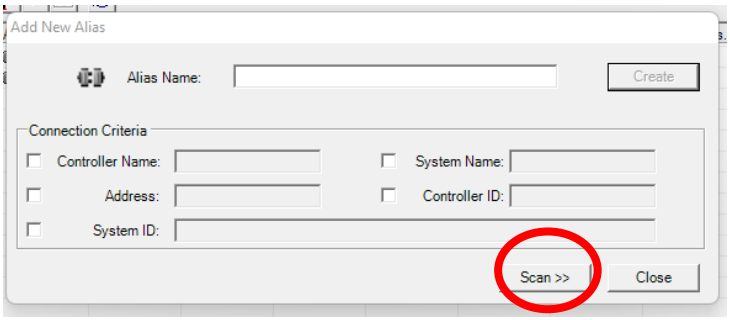



*Ilustración 124. Escanear controladores.*

3. Crear alias

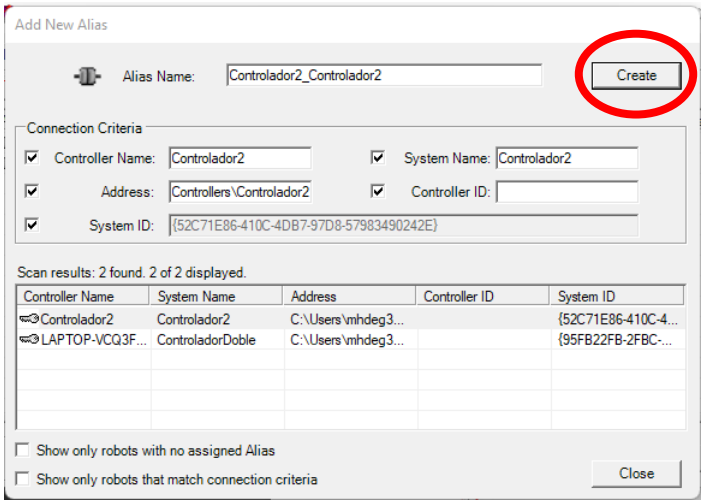



*Ilustración 125. Crear alias.*

El siguiente paso consiste en establecer la conexión con el servidor, para ello se pulsa en “Server Control” y seguidamente “Start”.

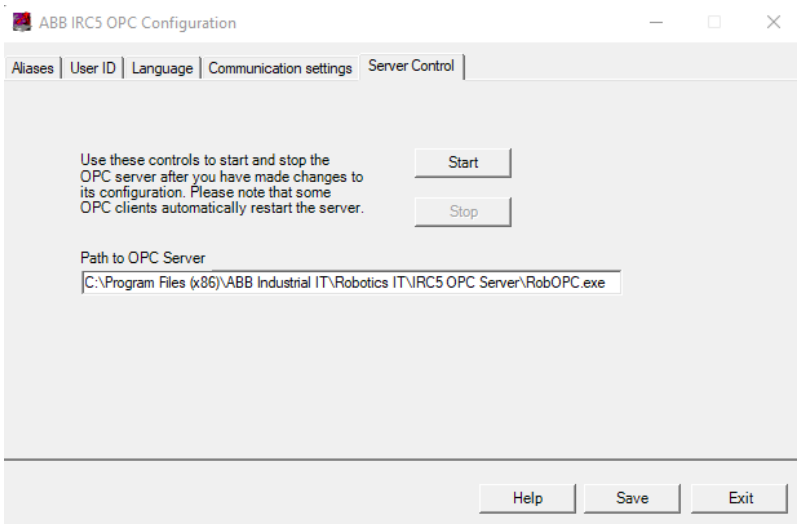



*Ilustración 126. Establecer conexión con el servidor.*

El siguiente objetivo corresponde con el establecimiento de la comunicación entre MATLAB y el servidor y así poder obtener las mismas posibilidades que se tienen en este momento entre RobotStudio y el propio servidor.

Una vez arrancado MATLAB se emplea el comando “OPCTool” para conseguir el nombre de las variables y señales de los controladores que se van a utilizar. Primeramente, se debe de crear un “host” que pertenece al nombre o la dirección IP de la máquina que alberga dicho servidor, para acto seguido crear un cliente y establecer la conexión tal y como se observa en la Imagen 127.

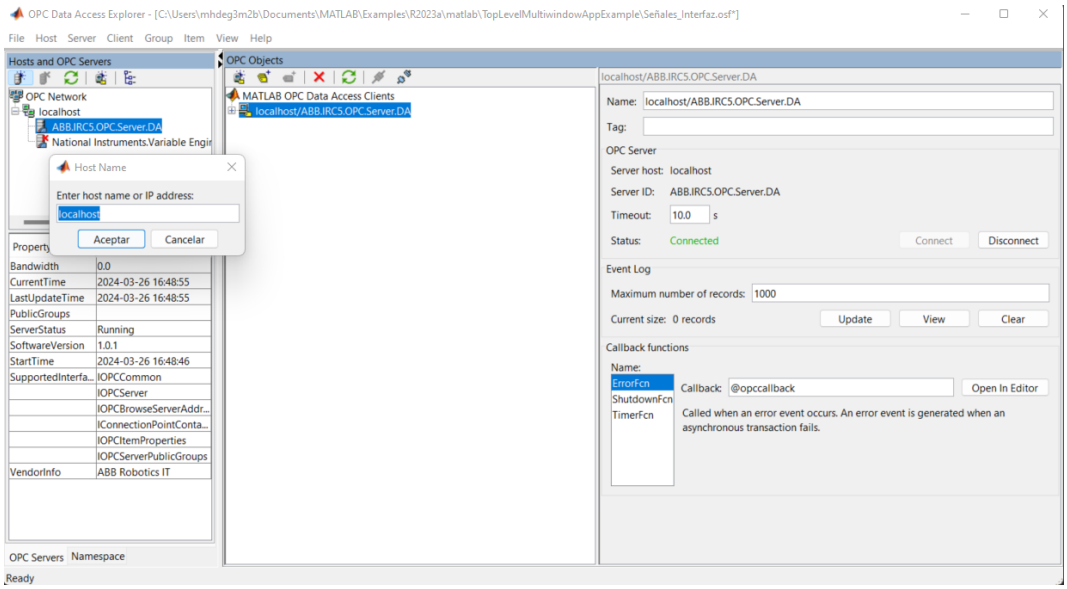



*Ilustración 127. Crear Host MATLAB.*

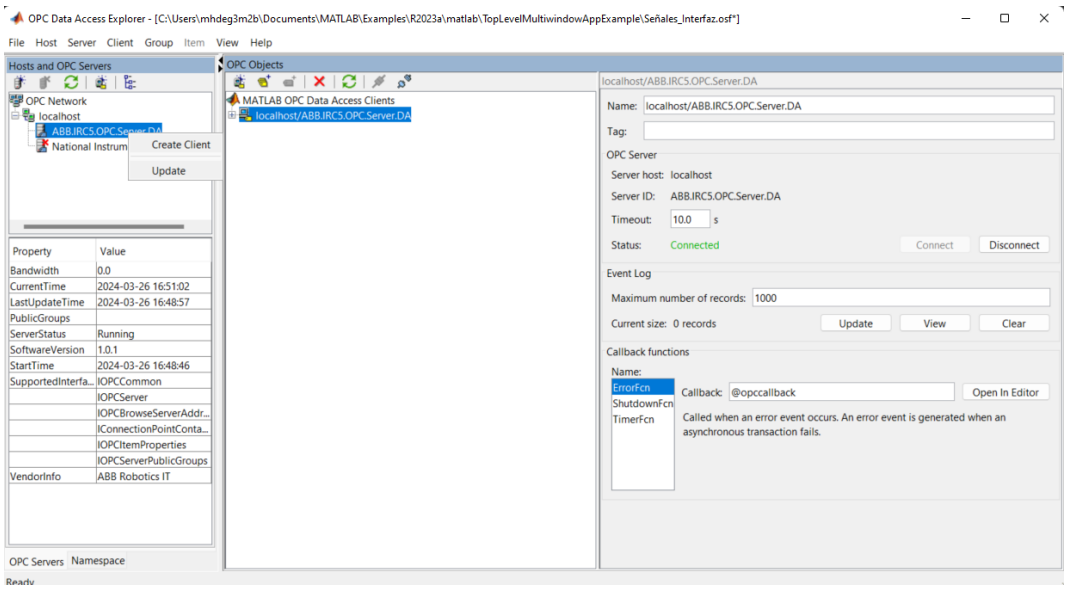



*Ilustración 128. Crear cliente MATLAB.*

El último paso consiste en añadir un grupo donde se añadirán todos los ítems que se van a emplear en la programación de AppDesigner, ocupando cada uno de estos ítems un espacio de nombre del servidor. De este modo se puede

controlar y modificar las variables y señales elegidas cuando se interacción con ellas en todo momento.

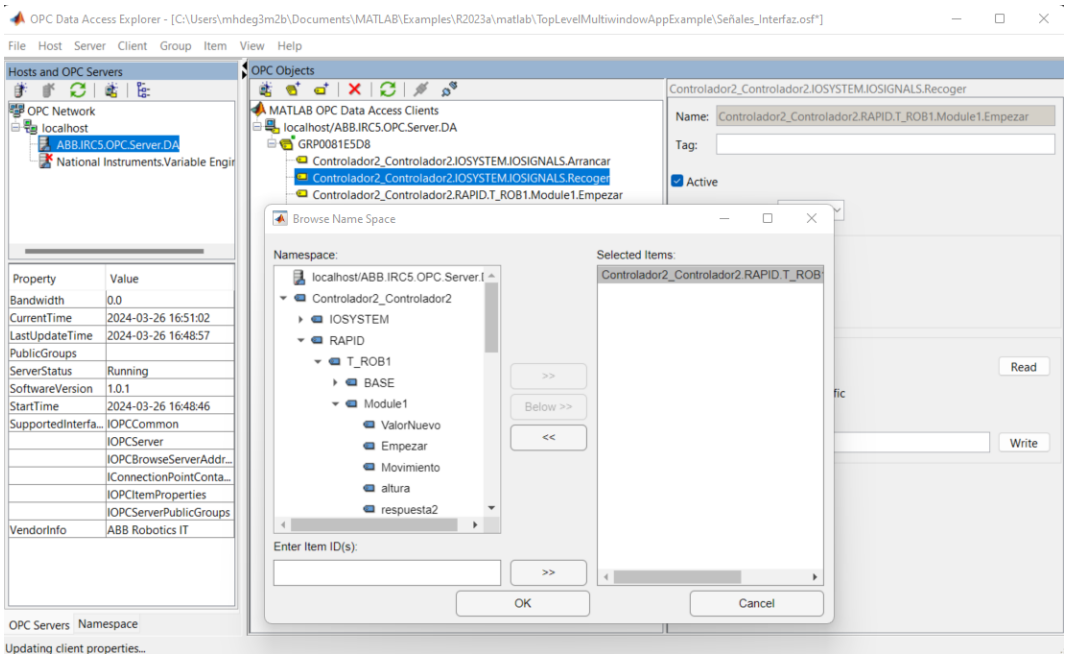



*Ilustración 129. Variables en el servidor OPC.*

Realizados los pasos anteriores, dentro de AppDesigner se programan los comandos convenientes para poder establecer la conexión y poder leer o escribir mediante “Read” y “Write”.

## 4.4 MATLAB, AppDesigner

De manera que las órdenes de la persona se reciban de una forma externa al robot colaborativo y así poder simular el reconocimiento facial, se ha diseñado una interfaz Hombre-Máquina con la que poder gobernar la simulación.

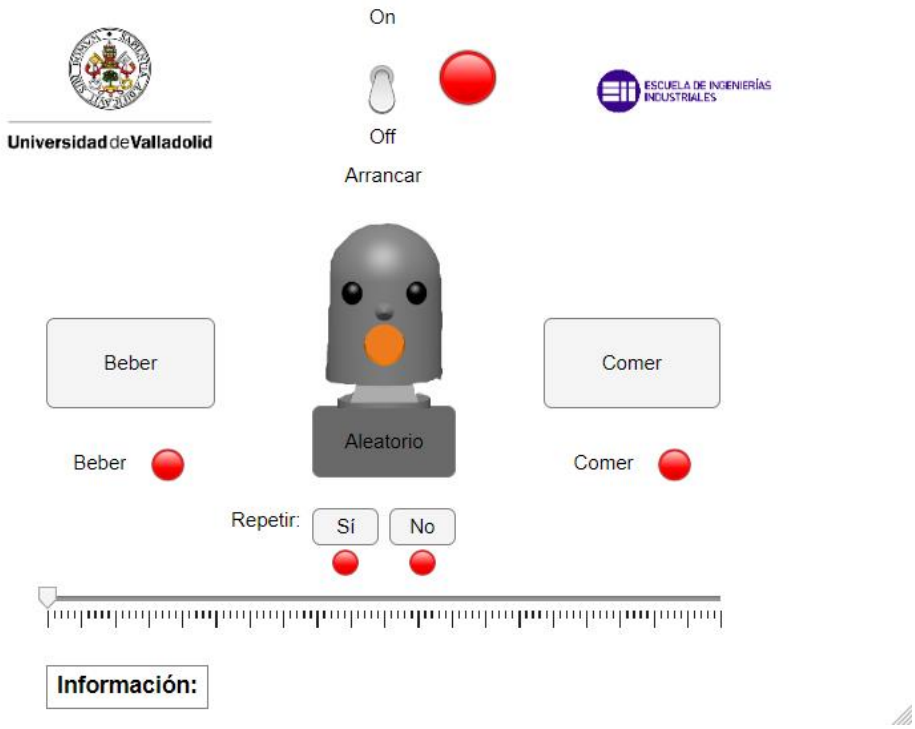



*Ilustración 130. Interfaz de usuario AppDesigner.*

La interfaz se ha diseñado con el objetivo de simular de la manera más realista la realización del movimiento de la persona, por ello, se dispone de un slider que permite elegir la posición de giro de la cabeza de la persona una vez eligió el servicio a través de uno de los tres botones (Beber, Comer o Aleatorio). Otros dos botones con sus respectivas lámparas para repetir y un interruptor que permite activar la interfaz y establecer la conexión.

Seguidamente se continua con la explicación de cada uno de los componentes creados y las respectivas funciones que conjuntamente permiten el correcto funcionamiento de la imagen 130.

### 4.4.1 Arrancar

Primero de todo se ha diseñado un interruptor ("Switch") que al encontrarse en el estado "On" se establecerá la comunicación para el intercambio de información de las variables entre ambos programas.

Para poder intercambiar información se han escrito las líneas de código que se observan en la imagen 131, mediante el comando "OPCda" se creará el cliente de acceso a datos del OPC utilizado, para posteriormente conectar el cliente al servidor OPC. A partir de ahora, ya se tiene conexión y se podría empezar a intercambiar información.

Al necesitar varias variables para el correcto funcionamiento se creará un grupo de objetos de acceso a datos OPC con el comando "addgroup" donde se establecerá el nombre del grupo en concreto que se va a utilizar y, posteriormente se configuraran las propiedades del grupo con "set".

Después de estas líneas de código se informará al usuario a través del recuadro de información que se ha conectado al servidor correctamente y, por tanto, está listo para intercambiar información.

Para ello se debe declarar las variables y añadirlas al grupo "Group_ok" escribiendo el ID correspondiente a cada una. Dicho ID se encuentra fácilmente como se explica en el apartado anterior en la imagen 129.

```
app.Lamp.Color = 'r';
app.da=opcda('localhost','ABB.IRC5.OPC.Server.DA');
connect(app.da);
app.group_ok=addgroup(app.da,'GRP0081E5D8');
set(app.group_ok,'UpdateRate',0.001);
pause(0.4);
app.Lamp.Color = 'y';
app.ComentariosLabel_2.Text= 'Conectado al servidor';


app.ServicioIt=additem(app.group_ok,'Controlador2_Controlador2.RAPID.T_ROB1.Module1.Movimiento');
app.PosicionIt=additem(app.group_ok,'Controlador2_Controlador2.RAPID.T_ROB1.Module1.ValorAleRec');
app.EmpezarIt=additem(app.group_ok,'Controlador2_Controlador2.RAPID.T_ROB1.Module1.Empezar');
recogerItem=serveritems(app.da,'Controlador2_Controlador2.IOSYSTEM.IOSIGNALS.Recoger');
app.RecogerIt=additem(app.group_ok,recogerItem);
app.RespuestaIt=additem(app.group_ok,'Controlador2_Controlador2.RAPID.T_ROB1.Module1.respuesta');
app.ArrancarIt=additem(app.group_ok,'Controlador2_Controlador2.IOSYSTEM.IOSIGNALS.Arrancar');
```

*Ilustración 131. Función Arrancar.*

Una vez establecidas las variables entrará en un bucle condicional While que se va a ejecutar continuamente hasta que la persona no quiera repetir que en ese caso se desconectará del servidor.

El programa dentro del bucle estará esperando a que se accione algún botón en la interfaz que active un servicio, una vez dentro se emplea el comando "Write" para escribir la información que queremos que se lea en las variables

PERS de RobotStudio. Escrito el servicio, se podrá escribir también el valor del ángulo de giro que mediante un comentario se mostrará el valor en el recuadro de información de la interfaz.

Para leer las variables de RobotStudio se emplea el comando “Read” con él se obtendrá la señal de que se ha terminado el servicio y se encuentra esperando a una nueva orden, es decir, llama a la función Repetir2 que más adelante se explicará.

```
while app.stop==0
    app.Lamp.Color = 'g';

    if app.BeberButton.Value == 1 ||app.ComerButton.Value == 1
        if app.BeberButton.Value == 1
            write(app.ServicioIt,'2');
            pause(0.1);
            value = app.Slider.Value;
            valor1=num2str(value);
            app.ComentariosLabel_2.Text= ['Gira :' num2str(valor1)];
            write(app.PosicionIt,value);
            pause(0.1);
            app.leer=read(app.RecogerIt);
            if app.leer.Value==1
                app.Repetir2();
            end

        end

        if app.ComerButton.Value == 1
            write(app.ServicioIt, '1');
            pause(0.1);
            value = app.Slider.Value;
            valor1=num2str(value);
            app.ComentariosLabel_2.Text= ['Gira :' num2str(valor1)];
            write(app.PosicionIt,value);
            pause(0.1);
            app.leer=read(app.RecogerIt);
            if app.leer.Value==1
                app.Repetir2();
            end
        end
    end
```

*Ilustración 132. Bucle While en función Arrancar.*

En la imagen 133 se observan las variables que han sido necesarias definir para el funcionamiento de la interfaz, pudiendo estas, ser utilizadas en cualquier parte del script ya que son declaradas variables globales. Se enumeran a continuación:

- La variable del objeto de cliente de acceso a datos
- Las variables utilizadas para detener los bucles while.
- Variable empleada para la lectura de información
- Variables empleadas para la escritura de datos OPC

```
properties (Access = private)
    da
    group_ok

    stop=0
    stop3=0

    leer

    ArrancarIt
    EmpezarIt
    RecogerIt
    ServicioIt
    PosicionIt
    RespuestaIt
end
```

*Ilustración 133. Propiedades interfaz de usuario.*

### 4.4.2 Repetir

Para que esta función entre en ejecución primero se ha de haber leído en Arrancar la señal que informe que el servicio se ha realizado correctamente. Si es así, entrará al bucle a esperar que el usuario pulse el botón de repetir o no repetir, en cualquiera de los casos mediante "RespuestaIt" se enviará la información y se mostrará en información la opción escogida.

En el caso de que quiera repetir será necesario reinicializar las variables para poder abordar correctamente el nuevo servicio acordado, por ello se llama a la función reiniciar.

En el caso opuesto de que no quiera repetir, se apagarán todos los botones y se desconectará del servidor mediante la función desconectar.

```
function Repetir2(app)

  app.stop3=0;
  app.ComerButton.Value=0;
  app.BeberButton.Value=0;

  while app.stop3==0
    if app.RepSiButton.Value==1 || app.RepNoButton.Value==1
        if app.RepSiButton.Value==1
            app.LampRepSi.Color= 'g';
            app.LampRepNo.Color= 'r';
            app.ComentariosLabel_2.Text = 'Quiere repetir';
            write(app.RespuestaIt,'1');
            pause(0.1);
            app.reiniciar();
            pause(0.1);
            app.RepSiButton.Value=0;

        elseif app.RepNoButton.Value==1
            app.LampRepSi.Color= 'r';
            app.RepSiButton.Value=0;
            app.LampRepNo.Color= 'g';
            write(app.RespuestaIt,'0');
            app.ComentariosLabel_2.Text = ' NO quiere repetir';
            app.ComerButton.Value=0;
            app.ComerLamp.Color= 'r';
            app.BeberButton.Value=0;
            app.BeberLamp.Color= 'r';
            pause(0.1);
            app.Desconetar();
        end
    end
    pause(0.1);
  end
end
```

*Ilustración 134. Función repetir.*

### 4.4.3 Reiniciar

Será utilizada al principio de cada nuevo servicio para restablecer a los valores iniciales de las variables empleadas y reiniciar los estados de los pulsadores.

```
function reiniciar(app)

        app.ComerButton.Value=0;
        app.BeberButton.Value=0;
        write(app.ServicioIt,'0');
        write(app.PosicionIt,'0');
        write(app.EmpezarIt,'0');
        write(app.RespuestaIt,'2');
        app.stop3=1;
        app.ComerLamp.Color= 'r';
        app.BeberLamp.Color= 'r';
        app.LampRepSi.Color= 'r';
        app.LampRepNo.Color= 'r';

end
```

*Ilustración 135. Función reiniciar.*

### 4.4.4 Desconectar

En el caso de que la persona desee no repetir o se accione el interruptor arrancar en el estado "Off", se establecerán a su valor inicial todas las variables utilizadas y se dejarán de leer, para acto seguido desconectarse del servidor y eliminar el objeto de cliente de acceso a datos OPC.

```
function  Desconetar(app)

        app.ComentariosLabel_2.Text = ' Desconectando del servidor...';
        write(app.ServicioIt,'0');
        write(app.PosicionIt,'0');
        write(app.EmpezarIt,'0');
        app.stop3 = 1;
        app.stop = 1;
        pause(2);
        disconnect(app.da);
        app.ComentariosLabel_2.Text = 'Desconectado';
        delete(app.da);
        app.LampRepSi.Color= 'r';
        app.LampRepNo.Color= 'r';
        app.BeberLamp.Color= 'r';
        app.ComerLamp.Color= 'r';
        app.Lamp.Color = 'r';
end
```

*Ilustración 136. Función desconectar.*

# 5 Funcionamiento de la estación

Después de explicar los entresijos de la presente estación y su desarrollo, de manera resumida se procede a mostrar los pasos para el correcto empleo de la simulación.

- Paso 1: Iniciar el servidor en la aplicación "ABB IRC5 OPC Configuration".
- Paso 2: Arrancar la simulación de la estación en RobotStudio.
- Paso 3: Arrancar la simulación de la interfaz de usuario en AppDesigner.
- Paso 4: Accionar el interruptor "Arrancar" al estado "On"
- Paso 5: Pulsar el botón del servicio que se desee.
- Paso 6: Seleccionar la posición de la cabeza de la persona a través del "slider".
- Paso 7: Dependiendo si se quiere repetir o no, accionar un botón u otro.
  - Paso 7.1: Si se selecciona repetir, se deberán realizar todos los pasos desde el paso 5 otra vez.
  - Paso 7.2: Pulsar no repetir, desconectará y finalizará la simulación.

A continuación, se mostará una secuencia de imágenes sobre el funcionamiento paso a paso durante la simulación del programa.

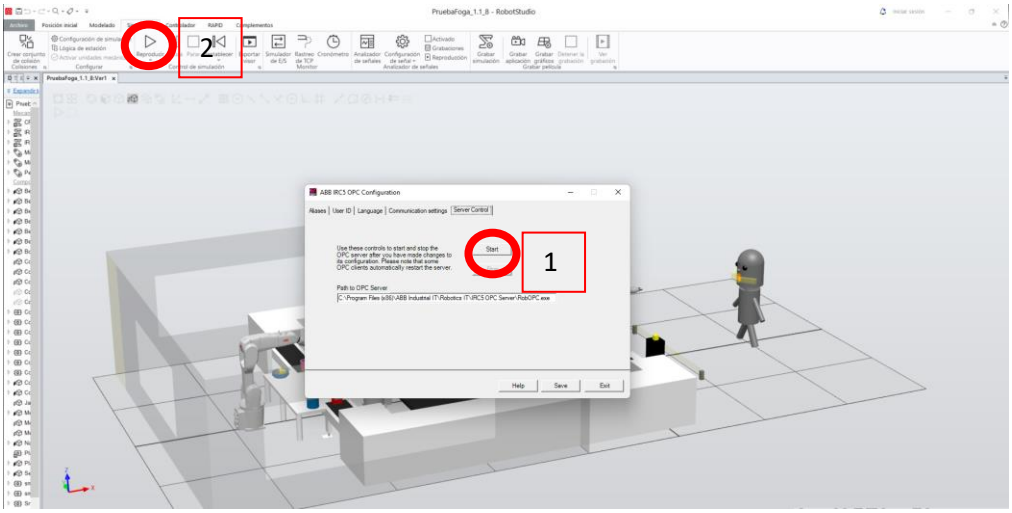



*Ilustración 137. Paso 1 y Paso 2.*

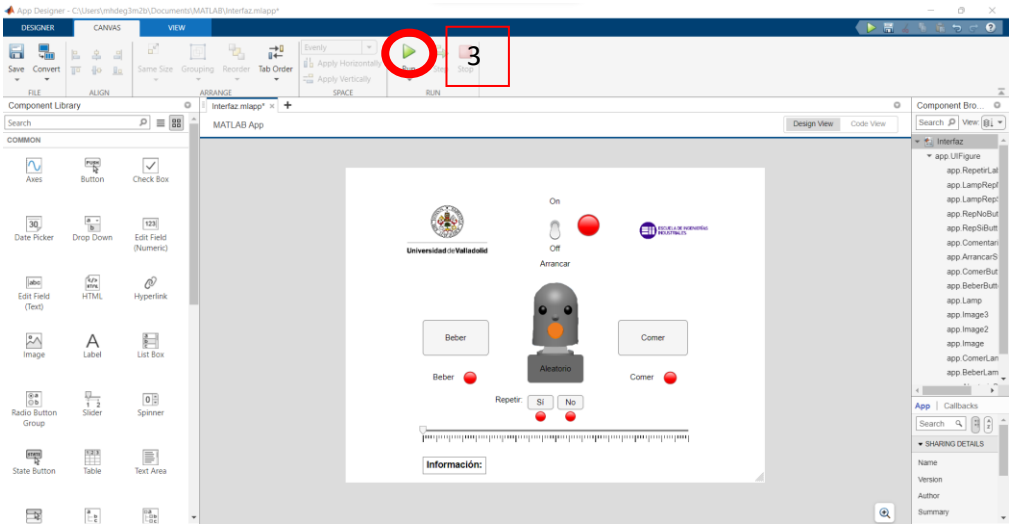



*Ilustración 138. Paso 3.*

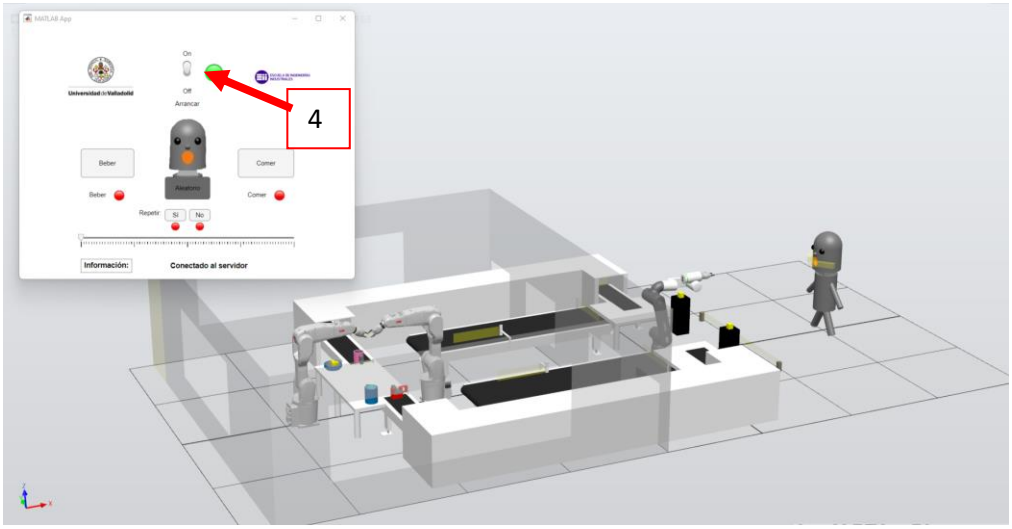



*Ilustración 139. Paso 4.*


*Ilustración 140. Paso 5.*


*Ilustración 141. Paso 6.*


*Ilustración 142.Paso 7 y Paso 7.1.*

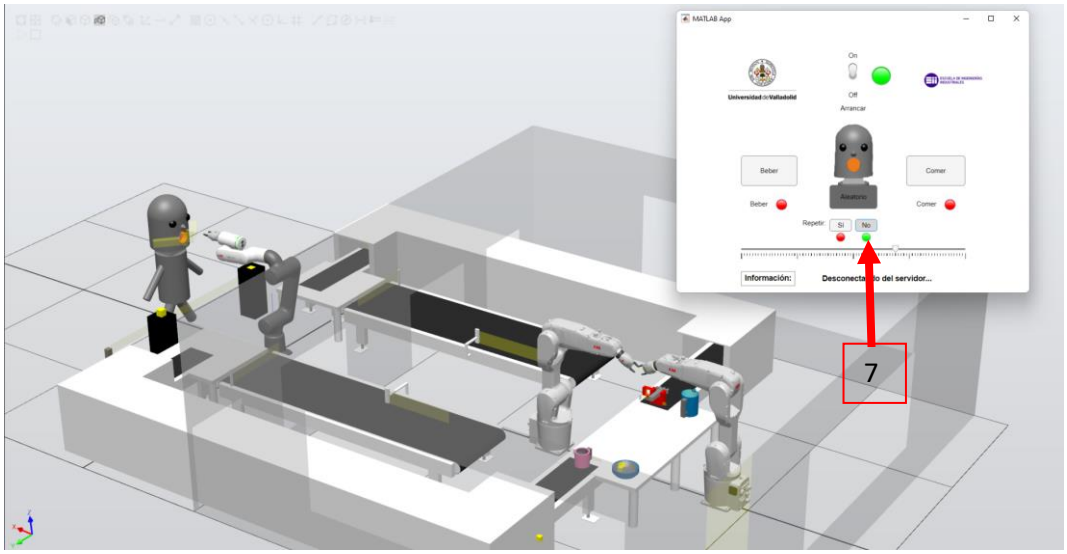



*Ilustración 143. Paso 7 y Paso 7.2.*


*Ilustración 144. Estación desconectada.*

# 6 Conclusiones

Realizado el presente trabajo de fin de grado y acorde a los resultados obtenidos, se pueden extraer un conjunto de conclusiones positivas acerca de los objetivos propuestos.

Durante todo el proyecto se han ido desarrollando y aplicando diferentes conocimientos en el campo de la ingeniería, en especial, en el ámbito de la robótica, cumplimentando así uno de los objetivos.

Se ha desarrollado la correcta simulación para un servicio asistencial hacia una persona pudiendo comprobar gracias a RobotStudio que su implementación puede ser posible en el mundo real.

Se ha diseñado una estación que emplease dos robots distintos (IRB 1200 y CRB 15000 GoFa) con dos controladores distintos (FlexPendant y Omnicore), investigando sobre cada una de sus características y sus parámetros. Además, se ha aprendido el diseño de la lógica de estación y la programación de Smart Components en RobotStudio, los cuales, son una gran herramienta para poder llevar la simulación a lo más real posible. Con ello, se consiguió ampliar nuestros conocimientos en el software RobotStudio.

Un gran inconveniente que se tuvo que solventar fue la creación de los puntos y trayectorias para poder realizar el servicio sin derramar el líquido ni la comida, evitando problemas de singularidad y aportando la mayor estabilidad posible.

Para la interacción de la persona con el robot se ha empleado el software, MATLAB, que se ha utilizado durante el grado y con este trabajo se han podido aumentar los conocimientos, como, por ejemplo, encontrando la posibilidad de comunicarse entre varios programas a través del protocolo de comunicaciones OPC UA.

También se ha conseguido desarrollar una interfaz Hombre-Máquina para simular las decisiones de la persona, gracias a AppDesigner.

Podemos concluir que se han alcanzado correctamente todos los objetivos que se propusieron al inicio del presente trabajo de fin de grado.

# 7 Líneas futuras

Concluyendo este trabajo de fin de grado y para aquellos lectores que estén interesados en posibles cambios o mejoras, se plantean los siguientes casos:

- Realizar la simulación con un robot colaborativo real, ya que esto no fue posible debido a que en la Universidad de Valladolid todavía no se dispone de un robot CRB 15000 GoFa, el cual, en un tiempo se prevé comprar. Siendo esto, una de las causas de estudio de sus características y funciones.

- Implementar el diseño con la detección de la persona de verdad. Para ello sería necesario disponer de una herramienta de detección como una cámara web u otro dispositivo visual e implementarlo al robot.

- La realización de la comunicación a través de IoT en vez de OPC UA, siendo esta una gran posibilidad gracias al nuevo software de ABB IoT Gateway que permite la transmisión de datos desde la planta de la fábrica, robots… o la nube, uniendo todo en uno.

# 8 Bibliografía

[1] Línea de Tiempo, «La evoluvión de la robótica a través del tiempo.,» [En línea]. Available: https://lineadetiempo.net/la-evolucion-de-la-robotica-a-traves-del-tiempo-una-linea-cronologica/. [Último acceso: 05 11 2023].

[2] Pcweb.info, «Historia de la robótica, cronología, línea de tiempo.,» 2023. [En línea]. Available: https://pcweb.info/historia-de-la-robotica-cronologia-linea-de-tiempo/?expand_article=1. [Último acceso: 05 11 2023].

[3] Antiquitas, 2010. [En línea]. Available: https://historiautomatas.blogspot.com/2010/06/grecia-iii-heron-de-alejandria.html. [Último acceso: 08 11 2023].

[4] Microsiervos, «El telar de Jacquard, la primera máquina programable de la historia,» 2013. [En línea]. Available: https://www.microsiervos.com/archivo/ordenadores/telar-jacquard-primera-maquina-programable-de-la-historia.html. [Último acceso: 09 11 2023].

[5] Wikipedia, «Tres leyes de la robótica,» [En línea]. Available: https://es.wikipedia.org/wiki/Tres_leyes_de_la_rob%C3%B3tica. [Último acceso: 14 11 2024].

[6] Telefónica, «Tipos de robots: Clasificación, aplicaciones y ejemplos.,» [En línea]. Available: https://www.telefonica.com/es/sala-comunicacion/blog/tipos-de-robots-clasificacion-aplicaciones-y-ejemplos/. [Último acceso: 14 11 2023].

[7] EDS robotics, «Diferencias entre robots y cobots,» [En línea]. Available: https://www.edsrobotics.com/blog/diferencias-robots-vs-cobots/. [Último acceso: 15 11 2023].

[8] C. González Valenzuela, «Así reconocen las cámaras con IA a los objetos y personas,» 2022. [En línea]. Available: https://computerhoy.com/tecnologia/reconocen-camaras-ia-objetos-personas-1169310. [Último acceso: 20 11 2023].

[9] G. Ermacora, «Top 4 Face Recognition Software for Robots,» [En línea]. Available: https://medium.com/@ermacora.gabriele/4-face-recognition-software-for-robot-as-a-service-538cc131a8fc. [Último acceso: 20 11 2023].

[10] L. Pardo, «Opentrack: Cómo rastrear los movimientos de tu cabeza con una webcam,» 2023. [En línea]. Available: https://www.neoteo.com/opentrack-como-

rastrear-los-movimientos-de-tu-cabeza-con-una-webcam/. [Último acceso: 20 11 2023].

[11] J. A. Ávila Herrero, «Modelado de una célula robótica con fines educativos usando el programa Robot-Studio,» 2015. [En línea]. Available: https://uvadoc.uva.es/handle/10324/14441. [Último acceso: 22 01 2024].

[12] A. Galindo De Santos, «Control remoto de una Célula robotizada mediante tecnología WI-FI,» 2016. [En línea]. Available: https://uvadoc.uva.es/handle/10324/17056. [Último acceso: 22 01 2024].

[13] C. Jiménez Jiménez, «Diseño de un sistema robótico educativo para jugar al ajedrez con robots industriales,» 2019. [En línea]. Available: https://uvadoc.uva.es/handle/10324/36877. [Último acceso: 22 01 2024].

[14] E. Pozas Mata, «Simulación de un Robot Colaborativo YuMi (ABB) en entorno RobotStudio comandado desde MATLAB mediante protocolo OPC UA para tocar un Xilófono,» 2022. [En línea]. Available: https://uvadoc.uva.es/handle/10324/54231. [Último acceso: 26 01 2024].

[15] R. Vidal del Cura, «Célula multi-robot para el montaje de una luminaria, basado en robots ABB controlados mediante un interface de Matlab, empleando el protocolo de comunicaciones OPC,» 2022. [En línea]. Available: https://uvadoc.uva.es/handle/10324/53931. [Último acceso: 22 01 2024].

[16] R. Sancho García, «Diseño con Autodesk Inventor de una célula de trabajo robótica para realizar montajes multitarea apoyados con visión artificial, coordinado mediante protocolo de comunicación OPC UA entre RobotStudio-MATLAB,» 2021. [En línea]. Available: https://uvadoc.uva.es/handle/10324/47254. [Último acceso: 22 01 2024].

[17] J. Ruiz Luengo, «Simulación de un Robot ABB IRB 120 para la detección por rastreo de piezas y posterior emplazamiento llevado a cabo por Robot ABB CRB 15000 GoFa. Implementación de protocolo OPC UA para la comunicación entre ambos y diseño de Interfaz gráfica desde MATLAB,» 2023. [En línea]. Available: https://uvadoc.uva.es/handle/10324/61462. [Último acceso: 15 02 2024].

[18] SATOSHI, «opiron,» 2018. [En línea]. Available: https://www.opiron.com/que-es-opc-ua/. [Último acceso: 24 11 2023].

[19] ABB, «Manual del operador RobotStudio,» [En línea]. Available: https://library.e.abb.com/public/997992016618ffbbc1257b4b0051ca17/3HAC032104-005_revC_es.pdf. [Último acceso: 26 02 2024].

[20] ABB, «Especificaciones del producto IRB 1200,» [En línea]. Available: https://library.e.abb.com/public/22dfa39f7411413e83396b812ac216c8/3HAC046982%20PS%20IRB%201200-es.pdf?x-sign=qFHoeqNciHrmnUJalvWPcZLaMOWxSEKb+20kDpMviKa/Z3V5upAUMp3C9cVTbV6C. [Último acceso: 20 03 2024].

[21] ABB, «GoFa™ CRB 15000,» [En línea]. Available: https://new.abb.com/products/robotics/es/robots/robots-colaborativos/gofa-crb-15000. [Último acceso: 23 03 2024].

# 9 Anexo

Se procede a mostrar los scripts realizados en MATLAB y RobotStudio.

## MATLAB

```
classdef Interfaz < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure              matlab.ui.Figure
        RepetirLabel          matlab.ui.control.Label
        LampRepNo             matlab.ui.control.Lamp
        LampRepSi             matlab.ui.control.Lamp
        RepNoButton           matlab.ui.control.StateButton
        RepSiButton           matlab.ui.control.StateButton
        ComentariosLabel      matlab.ui.control.TextArea
        ComentariosLabel_2    matlab.ui.control.Label
        ArrancarSwitch        matlab.ui.control.ToggleSwitch
        ComerButton           matlab.ui.control.StateButton
        BeberButton           matlab.ui.control.StateButton
        Lamp                  matlab.ui.control.Lamp
        Image3                matlab.ui.control.Image
        Image2                matlab.ui.control.Image
        Image                 matlab.ui.control.Image
        ArrancarSwitchLabel   matlab.ui.control.Label
        ComerLamp             matlab.ui.control.Lamp
        ComerLampLabel        matlab.ui.control.Label
        BeberLamp             matlab.ui.control.Lamp
        BeberLampLabel        matlab.ui.control.Label
        AleatorioButton       matlab.ui.control.Button
        Slider                matlab.ui.control.Slider
    end


    properties (Access = private)
        da
        group_ok

        stop=0
        stop3=0

        leer

        ArrancarIt
        EmpezarIt
        RecogerIt
        ServicioIt
        PosicionIt
        RespuestaIt
    end

    methods (Access = private)

        function  Desconetar(app)
```

```
                app.ComentariosLabel_2.Text = ' Desconectando del
servidor...';
                write(app.ServicioIt,'0');
                write(app.PosicionIt,'0');
                write(app.EmpezarIt,'0');
                app.stop3 = 1;
                app.stop = 1;
                pause(2);
                disconnect(app.da);
                app.ComentariosLabel_2.Text = 'Desconectado';
                delete(app.da);
                app.LampRepSi.Color= 'r';
                app.LampRepNo.Color= 'r';
                app.BeberLamp.Color= 'r';
                app.ComerLamp.Color= 'r';
                app.Lamp.Color = 'r';
        end

        function reiniciar(app)

                app.ComerButton.Value=0;
                app.BeberButton.Value=0;
                write(app.ServicioIt,'0');
                write(app.PosicionIt,'0');
                write(app.EmpezarIt,'0');
                write(app.RespuestaIt,'2');
                app.stop3=1;
                app.ComerLamp.Color= 'r';
                app.BeberLamp.Color= 'r';
                app.LampRepSi.Color= 'r';
                app.LampRepNo.Color= 'r';

        end

        function Repetir2(app)

          app.stop3=0;
          app.ComerButton.Value=0;
          app.BeberButton.Value=0;

          while app.stop3==0
            if app.RepSiButton.Value==1 || app.RepNoButton.Value==1
                if app.RepSiButton.Value==1
                    app.LampRepSi.Color= 'g';
                    app.LampRepNo.Color= 'r';
                    app.ComentariosLabel_2.Text = 'Quiere repetir';
                    write(app.RespuestaIt,'1');
                    pause(0.1);
                    app.reiniciar();
                    pause(0.1);
                    app.RepSiButton.Value=0;

                elseif app.RepNoButton.Value==1
                    app.LampRepSi.Color= 'r';
                    app.RepSiButton.Value=0;
                    app.LampRepNo.Color= 'g';
                    write(app.RespuestaIt,'0');
```

```
                    app.ComentariosLabel_2.Text = ' NO quiere repetir';
                    app.ComerButton.Value=0;
                    app.ComerLamp.Color= 'r';
                    app.BeberButton.Value=0;
                    app.BeberLamp.Color= 'r';
                    pause(0.1);
                    app.Desconetar();
                end
            end
            pause(0.1);
        end
    end



end


% Callbacks that handle component events
methods (Access = private)

    % Value changing function: Slider
    function SliderValueChanging(app, event)

    end

    % Value changed function: ArrancarSwitch
    function ArrancarSwitchValueChanged(app, event)
        value = app.ArrancarSwitch.Value;

        switch value
            case 'On'

                    app.stop=0;

                    app.Lamp.Color = 'r';

app.da=opcda('localhost','ABB.IRC5.OPC.Server.DA');
                    connect(app.da);
                    app.group_ok=addgroup(app.da,'GRP0081E5D8');
                    set(app.group_ok,'UpdateRate',0.001);
                    pause(0.4);
                    app.Lamp.Color = 'y';
                    app.ComentariosLabel_2.Text= 'Conectado al
servidor';



app.ServicioIt=additem(app.group_ok,'Controlador2_Controlador2.RAPID.T_R
OB1.Module1.Movimiento');

app.PosicionIt=additem(app.group_ok,'Controlador2_Controlador2.RAPID.T_R
OB1.Module1.ValorAleRec');

app.EmpezarIt=additem(app.group_ok,'Controlador2_Controlador2.RAPID.T_RO
B1.Module1.Empezar');
```

```
recogerItem=serveritems(app.da,'Controlador2_Controlador2.IOSYSTEM.IOSIG
NALS.Recoger');
                        app.RecogerIt=additem(app.group_ok,recogerItem);

app.RespuestaIt=additem(app.group_ok,'Controlador2_Controlador2.RAPID.T_
ROB1.Module1.respuesta');

app.ArrancarIt=additem(app.group_ok,'Controlador2_Controlador2.IOSYSTEM.
IOSIGNALS.Arrancar');


                        pause(0.1);

                        while app.stop==0
                            app.Lamp.Color = 'g';

                            if app.BeberButton.Value == 1
||app.ComerButton.Value == 1
                                if app.BeberButton.Value == 1
                                    write(app.ServicioIt,'2');
                                    pause(0.1);
                                    value = app.Slider.Value;
                                    valor1=num2str(value);
                                    app.ComentariosLabel_2.Text= ['Gira
:' num2str(valor1)];
                                    write(app.PosicionIt,value);
                                    pause(0.1);
                                    app.leer=read(app.RecogerIt);
                                    if app.leer.Value==1
                                        app.Repetir2();
                                    end

                                end

                                if app.ComerButton.Value == 1
                                    write(app.ServicioIt, '1');
                                    pause(0.1);
                                    value = app.Slider.Value;
                                    valor1=num2str(value);
                                    app.ComentariosLabel_2.Text= ['Gira
:' num2str(valor1)];
                                    write(app.PosicionIt,value);
                                    pause(0.1);
                                    app.leer=read(app.RecogerIt);
                                    if app.leer.Value==1
                                        app.Repetir2();
                                    end
                                end
                            end


                            pause(0.5);
                        end

                case 'Off'
```

```
            app.Desconetar();
        end
    end

    % Value changed function: BeberButton
    function BeberButtonValueChanged(app, event)
        valueB = app.BeberButton.Value;

        if (valueB==0)
            app.BeberLamp.Color = 'r';
        elseif (valueB==1)
            app.ComerButton.Value=0;
            app.ComerLamp.Color = 'r';
            app.BeberLamp.Color = 'g';
        end
    end

    % Value changed function: ComerButton
    function ComerButtonValueChanged(app, event)
       valueC = app.ComerButton.Value;
        if (valueC==0)
            app.ComerLamp.Color = 'r';
        elseif (valueC==1)
            app.BeberButton.Value=0;
            app.BeberLamp.Color= 'r';
            app.ComerLamp.Color = 'g';

        end
    end

    % Button pushed function: AleatorioButton
    function AleatorioButtonPushed(app, event)

        value =randi([0,1]);
        if value ==0

            app.BeberButton.Value=1;
            app.ComerButton.Value=0;
        elseif value == 1

            app.ComerButton.Value=1;
            app.BeberButton.Value=0;
        end
    end

    % Value changed function: Slider
    function SliderValueChanged(app, event)

    end

    % Value changed function: RepSiButton
    function RepSiButtonValueChanged(app, event)
            valueRS= app.RepSiButton.Value;
            if valueRS==1
                app.LampRepSi.Color='g';
            elseif valueRS==0
                app.LampRepSi.Color='r';
```

```
            end

        end

        % Value changed function: RepNoButton
        function RepNoButtonValueChanged(app, event)
            ValueRN= app.RepNoButton.Value;
            if ValueRN==1
                app.LampRepNo.Color='g';
            elseif ValueRN==0
                app.LampRepNo.Color='r';
            end


        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [1 1 1];
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';

            % Create Slider
            app.Slider = uislider(app.UIFigure);
            app.Slider.Limits = [0 80];
            app.Slider.MajorTickLabels = {''};
            app.Slider.ValueChangedFcn = createCallbackFcn(app,
@SliderValueChanged, true);
            app.Slider.ValueChangingFcn = createCallbackFcn(app,
@SliderValueChanging, true);
            app.Slider.Position = [118 79 405 3];

            % Create AleatorioButton
            app.AleatorioButton = uibutton(app.UIFigure, 'push');
            app.AleatorioButton.ButtonPushedFcn = createCallbackFcn(app,
@AleatorioButtonPushed, true);
            app.AleatorioButton.BackgroundColor = [0.4118 0.4118
0.4118];
            app.AleatorioButton.Position = [278 153 86 43];
            app.AleatorioButton.Text = 'Aleatorio';

            % Create BeberLampLabel
            app.BeberLampLabel = uilabel(app.UIFigure);
            app.BeberLampLabel.HorizontalAlignment = 'right';
            app.BeberLampLabel.Position = [129 151 37 22];
            app.BeberLampLabel.Text = 'Beber';

            % Create BeberLamp
            app.BeberLamp = uilamp(app.UIFigure);
            app.BeberLamp.Position = [181 151 20 20];
```

```
            app.BeberLamp.Color = [1 0 0];

            % Create ComerLampLabel
            app.ComerLampLabel = uilabel(app.UIFigure);
            app.ComerLampLabel.HorizontalAlignment = 'right';
            app.ComerLampLabel.Position = [430 151 41 22];
            app.ComerLampLabel.Text = 'Comer';

            % Create ComerLamp
            app.ComerLamp = uilamp(app.UIFigure);
            app.ComerLamp.Position = [486 151 20 20];
            app.ComerLamp.Color = [1 0 0];

            % Create ArrancarSwitchLabel
            app.ArrancarSwitchLabel = uilabel(app.UIFigure);
            app.ArrancarSwitchLabel.HorizontalAlignment = 'center';
            app.ArrancarSwitchLabel.Position = [295 324 51 22];
            app.ArrancarSwitchLabel.Text = 'Arrancar';

            % Create Image
            app.Image = uiimage(app.UIFigure);
            app.Image.Position = [430 334 143 103];
            app.Image.ImageSource = 'Eii.jpg';

            % Create Image2
            app.Image2 = uiimage(app.UIFigure);
            app.Image2.Position = [82 334 148 103];
            app.Image2.ImageSource = 'Uva.jpg';

            % Create Image3
            app.Image3 = uiimage(app.UIFigure);
            app.Image3.Position = [262 195 118 111];
            app.Image3.ImageSource = 'Captura de pantalla 2024-01-05
173014.png';

            % Create Lamp
            app.Lamp = uilamp(app.UIFigure);
            app.Lamp.Position = [354 376 35 35];
            app.Lamp.Color = [1 0 0];

            % Create BeberButton
            app.BeberButton = uibutton(app.UIFigure, 'state');
            app.BeberButton.ValueChangedFcn = createCallbackFcn(app,
@BeberButtonValueChanged, true);
            app.BeberButton.Text = 'Beber';
            app.BeberButton.Position = [118 195 101 54];

            % Create ComerButton
            app.ComerButton = uibutton(app.UIFigure, 'state');
            app.ComerButton.ValueChangedFcn = createCallbackFcn(app,
@ComerButtonValueChanged, true);
            app.ComerButton.Text = 'Comer';
            app.ComerButton.Position = [417 195 106 54];

            % Create ArrancarSwitch
            app.ArrancarSwitch = uiswitch(app.UIFigure, 'toggle');
```

```
            app.ArrancarSwitch.ValueChangedFcn = createCallbackFcn(app,
@ArrancarSwitchValueChanged, true);
            app.ArrancarSwitch.Position = [310 371 20 45];

            % Create ComentariosLabel_2
            app.ComentariosLabel_2 = uilabel(app.UIFigure);
            app.ComentariosLabel_2.BackgroundColor = [1 1 1];
            app.ComentariosLabel_2.HorizontalAlignment = 'center';
            app.ComentariosLabel_2.FontSize = 14;
            app.ComentariosLabel_2.FontWeight = 'bold';
            app.ComentariosLabel_2.Position = [222 16 279 22];
            app.ComentariosLabel_2.Text = ' ';

            % Create ComentariosLabel
            app.ComentariosLabel = uitextarea(app.UIFigure);
            app.ComentariosLabel.HorizontalAlignment = 'center';
            app.ComentariosLabel.FontSize = 14;
            app.ComentariosLabel.FontWeight = 'bold';
            app.ComentariosLabel.Position = [118 14 97 26];
            app.ComentariosLabel.Value = {'Información:'};

            % Create RepSiButton
            app.RepSiButton = uibutton(app.UIFigure, 'state');
            app.RepSiButton.ValueChangedFcn = createCallbackFcn(app,
@RepSiButtonValueChanged, true);
            app.RepSiButton.Text = 'Sí';
            app.RepSiButton.Position = [278 112 40 23];

            % Create RepNoButton
            app.RepNoButton = uibutton(app.UIFigure, 'state');
            app.RepNoButton.ValueChangedFcn = createCallbackFcn(app,
@RepNoButtonValueChanged, true);
            app.RepNoButton.Text = 'No';
            app.RepNoButton.Position = [324 112 40 23];

            % Create LampRepSi
            app.LampRepSi = uilamp(app.UIFigure);
            app.LampRepSi.Position = [290 94 16 16];
            app.LampRepSi.Color = [1 0 0];

            % Create LampRepNo
            app.LampRepNo = uilamp(app.UIFigure);
            app.LampRepNo.Position = [336 94 16 16];
            app.LampRepNo.Color = [1 0 0];

            % Create RepetirLabel
            app.RepetirLabel = uilabel(app.UIFigure);
            app.RepetirLabel.VerticalAlignment = 'top';
            app.RepetirLabel.Position = [229 95 61 40];
            app.RepetirLabel.Text = 'Repetir: ';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
```

```
    methods (Access = public)

        % Construct app
        function app = Interfaz

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

RobotStudio

Controlador2: Module1.

```
MODULE Module1

    CONST robtarget
inicio:=[[597.468565782,0,837.782032303],[0.706638273,0,0.707574979,0],[0,0,
0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_1:=[[-482.782027287,-83.111859452,-
162.012832383],[0.894807681,-0.363774547,0.239762986,-0.097473092],[1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_2:=[[0,0,-87.157438849],[1,0,0,0],[1,-2,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_3:=[[-437.486663888,-0.000035911,-
87.157480153],[1,0.000000132,0.000000014,0.000000124],[1,-2,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_4:=[[-458.722016747,-421.502770332,-
35.075802176],[1,0.000000161,0.000000022,0.000000087],[1,1,-
2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_5:=[[-458.722032314,-847.376628583,-
55.535304701],[1,0.000000125,-0.000000006,0.00000014],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget
Target_boca:=[[712.068900091,0,800],[0.707106781,0,0.707106781,0],[0,0,0,1],[
9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_6:=[[-60.542357209,-
587.764227631,144.424936003],[1,0.000000103,0.00000001,0.000000115],[0,1,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_7:=[[-16.380482628,-
553.045488735,162.514683911],[1,0.0000001,0.000000003,0.000000169],[0,1,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_8:=[[-4.612319746,-
550.857600319,195.890414431],[1,-0.0000001,-0.000000046,-
0.000000036],[0,1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_1_Ten:=[[13.352,115,19.023],[0.999206015,-
0.000697578,0.039808443,0.001463188],[-2,1,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_2_Ten:=[[15,115,91.59],[1,0,0,0],[-2,1,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_3_Ten:=[[-
426.618933014,136.149327583,19.023509367],[0.999206026,-
0.000700185,0.039808139,0.001462981],[-2,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_4_Ten:=[[-
426.61892242,687.487799313,19.023480011],[0.999206023,-
0.00070019,0.039808203,0.001462978],[-1,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_5_Ten:=[[-
207.747670102,747.184404197,19.023478637],[0.999206028,-
0.000700131,0.039808062,0.0014631],[-1,-
2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Target_6_Ten:=[[14.328609601,747.18440781,19.023473894],[0.999845362,0.0
16686716,-0.005337211,0.001523323],[-1,-
2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_7_Ten:=[[14.329,747.184,148.452],[0.999999738,-
0.000724312,0,0],[-1,-2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_8_Ten:=[[14.329,647.18,12.924],[0.9998044,-
0.00072417,0.0197114,0.001448725],[-1,-
2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_9_Ten:=[[14.329,647.18,73.95],[0.999999738,-
0.000724312,0,0],[-1,-2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10_Ten:=[[-
164.619186477,649.973816281,91.900820689],[0.999804387,-
0.000729508,0.01971184,0.00144875],[-1,-
2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_11_Ten:=[[-601.110333631,872.23720546,-
7.961897335],[0.95830516,-0.285062386,0.019309822,-0.004217626],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget Target_Boton_Beb_Pos:=[[575,575,500],[0,0,1,0],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Target_Boton_Beb_I:=[[575,575,470],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9
E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Beb_F:=[[575,575,420],[0,0,1,0],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Com_Pos:=[[575,-425,500],[0,0,1,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Com_I:=[[575,-425,470],[0,0,1,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Boton_Com_F:=[[575,-425,420],[0,0,1,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_3_coger:=[[0,0,12.842561151],[1,0,0,0],[1,-2,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_taza_9_dejar_10:=[[0,-550,233.449],[1,0,0,0],[0,1,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Target_A_10:=[[0,0,0],[1,0,0,0],[0,2,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Target_Alim_10:=[[-0.000031788,0,-0.000058676],[1,0,-
0.000000007,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    local VAR bool nosalir:= false;
    LOCAL VAR bool nosalir2:= FALSE;
    LOCAL VAR num opcion:= 1;
    LOCAL PERS num seed:= 27249;

    PERS num ValorAleRec;
    PERS num ValorNuevo;
    PERS num Movimiento;
    PERS pos altura;
    PERS num Empezar:=0;
    PERS num respuesta;
    PERS num respuesta2;

    VAR intnum iparada;
    VAR intnum irespuesta;
    VAR intnum irespuesta2;
    VAR num contadorComer:= 0;
    VAR num contadorBeber:= 0;



    PROC main()

      VAR num Mov_Aleatorio;
      VAR num Emp;
```

```
VAR bool ok;
SetDO pinza, 0;
SetDO Repetir_Proceso, 0;
SetDO Mov_Be,0;
SetDO Mov_Co,0;
SetDO Mov_Pos,0;
SetGO Posicion,0;
SetDO Sentido_Giro, 0;
SetDO Recoger, 0;
SetDO Arrancar,0;
respuesta:=2;
respuesta2:=0;
ValorAleRec := 0;
Movimiento:=0;
Mov_Aleatorio:=Round(rand());


nosalir:=TRUE;
nosalir2:=TRUE;



WHILE nosalir DO

  IF Repetir_Proceso = 1 THEN
      SetDO pinza, 0;
      SetDO Mov_Be,0;
      SetDO Mov_Co,0;
      SetDO Mov_Pos,0;
      SetGO Posicion,0;
      SetDO Sentido_Giro, 0;
      SetDO Recoger, 0;
      respuesta:=2;
      nosalir2:=TRUE;
    ENDIF

  IF Movimiento=1 OR Movimiento=2 THEN
    TPWrite"movimiento: "\Num:=Movimiento;
    IF Movimiento=1 THEN
      SetDO Arrancar,1;
      WaitTime 1;
      WaitDI Posicionado,1;
      WaitTime 1;
      setdo Mov_Co,1;
      SetDO mov_be,0;
      WaitDI Movim_Correcto,1;
      Accion_Boton_C;
      WaitDI Mover,1;
```

```
        Nueva_Posicion;
        Giro_Cabeza;
        SetDO Mov_Pos,1;
        Mov_Comer;
        rotar;
        Mov_Comer_Recog;
        SetDO Recoger, 1;
        WHILE nosalir2 DO
          IF respuesta=0 THEN
            nosalir:=FALSE;
            nosalir2:=FALSE;
            EXIT;
          ELSEIF respuesta=1 THEN
            SetDO Repetir_Proceso,1;
            WaitDI Posicionado,1;
            nosalir2:=FALSE;
          ENDIF
          WaitTime 0.1;
        ENDWHILE

      ELSEIF Movimiento=2 THEN
        SetDO Arrancar,1;
        WaitTime 1;
        WaitDI Posicionado,1;
        WaitTime 1;
        setdo Mov_Co,0;
        SetDO Mov_Be,1;
        WaitDI Movim_Correcto,1;
        Accion_Boton_B;
        WaitDI mover,1;
        Nueva_Posicion;
        Giro_Cabeza;
        SetDO Mov_Pos,1;
        Mov_Beber;
        rotar;
        Mov_Beber_Recog;
        SetDO Recoger, 1;
        WHILE nosalir2 DO
          IF respuesta=0 THEN
            nosalir:=FALSE;
            nosalir2:=FALSE;
            EXIT;
          ELSEIF respuesta=1 THEN
            SetDO Repetir_Proceso,1;
            WaitDI Posicionado,1;
            nosalir2:=FALSE;
          ENDIF
          WaitTime 0.1;
```

```
        ENDWHILE

      ENDIF
    ENDIF

  WaitTime 0.1;
ENDWHILE



    SetDO pinza, 0;
    SetDO Repetir_Proceso, 0;
    SetDO Mov_Be,0;
    SetDO Mov_Co,0;
    SetDO Mov_Pos,0;
    SetGO Posicion,0;
    SetDO Sentido_Giro, 0;
    SetDO Recoger, 0;
    Stop;

 ENDPROC

PROC interrupciones()
 IDelete iparada;
 CONNECT iparada WITH trap_parada;
 ISignalDO Arrancar,0,iparada;
ENDPROC

LOCAL TRAP trap_parada
   TPWrite "Parada";
ENDTRAP
LOCAL func num valor()
   VAR num opcion;

   IF (Movimiento=1) THEN
      opcion:=1;
   ELSEIF (Movimiento=2) THEN
      opcion:=2;
   ELSEIF (Movimiento=3) THEN
      opcion:=3;
   ENDIF

    RETURN opcion ;

 ENDFUNC

 LOCAL FUNC num rand()
   seed:=(171*seed) MOD 30269;
```

```
    RETURN seed/30269;
  ENDFUNC


     LOCAL TRAP trap_diComer
    TPWrite "Has elegido comer";
    TPWrite "Moviendo robot para acercar tenedor";
    opcion:= 1;
  ENDTRAP

  PROC inicializar()
    SetDO pinza, 0;
    SetDO Repetir_Proceso, 0;
    SetDO Mov_Be,0;
    SetDO Mov_Co,0;
    SetDO Mov_Pos,0;
    SetGO Posicion,0;
    SetDO Sentido_Giro, 0;
    SetDO Recoger, 0;
    SetDO Arrancar,0;
    respuesta:=2;
    respuesta2:=0;
  ENDPROC

PROC Giro_Cabeza()

   VAR num contador:=0;
        IF (ValorALErec>55 AND valorALErec<=80) THEN
            SetDO Sentido_Giro, 1;
            IF contador = 0 THEN
               contador:= contador+1;
            ENDIF

        ELSEIF (valorALErec <= 40 AND valorALErec>=0) THEN
            SetDO Sentido_Giro, 0;
            IF contador = 0 THEN
            contador:= contador+1;
            ENDIF

       ENDIF

endproc

PROC Nueva_Posicion()

   IF (ValorALErec>40 AND valorALErec<=80) THEN
            ValorAleRec:=Trunc(ValorAleRec);
            ValorNuevo:= (valoraleRec-40);
```

```
            SetGO Posicion, ValorNuevo;
  ELSEIF (valorALErec <= 40 AND valorALErec>=0) THEN
            ValorAleRec:=Trunc(ValorAleRec);
            ValorNuevo:=valorALErec;
            SetGO Posicion, ValorNuevo;
   ENDIF
ENDPROC


  PROC rotar()

  VAR robtarget rt_PointB;
  VAR robtarget rt_PointC;
  VAR robtarget rt_PointD;
  VAR pose ps_PointA;
  VAR pose ps_PointB;
  VAR pose ps_PointC;
  VAR pose ps_PointD;
  VAR num valorZ;
  VAR num valorY;
  VAR num valorX;
  VAR num n_RotationAngle ;

  n_RotationAngle:= ValorNuevo;
  ps_PointA := [target_a_10.trans,target_a_10.rot];


  IF (ValorALErec>40 AND valorALErec<=80) THEN

     valorZ:=Sin(180-90-n_RotationAngle);
     valorY:=Sin(n_RotationAngle);
     valorX:= altura.z - 80;

     ps_PointB := PoseMult( [[(800-valorx),-(360*valory),-
(360*valorz)],OrientZYX(0,0,-n_RotationAngle)], ps_PointA);
     ps_PointC := PoseMult( [[(800-valorx),-(340*valory),-
(340*valorz)],OrientZYX(0,0,-n_RotationAngle)], ps_PointA);
     IF Mov_Be=1 AND Mov_Co=0 THEN
        ps_PointD := PoseMult(  [[(800-valorx-20),-(290*valory),-
(290*valorz)],OrientZYX(0,0,-n_RotationAngle)], ps_PointA);
     ELSEIF Mov_Be=0 AND Mov_Co=1 THEN
        ps_PointD := PoseMult( [[(800-valorx-10),-(290*valory),-
(290*valorz)],OrientZYX(0,0,-n_RotationAngle)], ps_PointA);
     ENDIF

  ELSEIF(valorALErec <= 40 AND valorALErec>=0) THEN

     valorZ:=Sin(180-90-n_RotationAngle);
```

```
    valorY:=Sin(n_RotationAngle);
    valorX:= altura.z - 80;

    ps_PointB := PoseMult( [[(800-valorX),390*valory,-
(390*valorz)],OrientZYX(0,0,n_RotationAngle)], ps_PointA);
    ps_PointC := PoseMult( [[(800-valorx),(330*valory),-
(330*valorz)],OrientZYX(0,0,n_RotationAngle)], ps_PointA);
    IF Mov_Be=1 AND Mov_Co=0 THEN
      ps_PointD := PoseMult(  [[(800-valorx-20),(290*valory),-
(290*valorz)],OrientZYX(0,0,n_RotationAngle)], ps_PointA);
    ELSEIF Mov_Be=0 AND Mov_Co=1 THEN
      ps_PointD := PoseMult( [[(800-valorx-10),(290*valory),-
(290*valorz)],OrientZYX(0,0,n_RotationAngle)], ps_PointA);
    ENDIF
  ENDIF

  rt_PointB.trans := ps_pointb.trans;
  rt_PointB.rot := ps_pointb.rot;
  rt_PointC.trans := ps_pointC.trans;
  rt_PointC.rot := ps_pointC.rot;
  rt_PointD.trans := ps_pointd.trans;
  rt_PointD.rot := ps_pointd.rot;

  MoveL inicio,v300,fine,Servo\WObj:=wobj0;
  MoveL rt_PointB,v100,fine, Servo\WObj:=Workobject_Alim;
  MoveL rt_PointC,v100,fine,Servo\WObj:=Workobject_Alim;
  MoveL rt_PointD,v100,fine,Servo\WObj:=Workobject_Alim;

  IF Mov_Be = 1 AND Mov_Co = 0 THEN

    Movel offs (rt_PointD,20,0,0),v100,fine,Servo\WObj:=Workobject_Alim;
    movel RelTool (rt_Pointd,-
10,0,10,\Ry:=30),v100,fine,Servo\WObj:=Workobject_Alim;

  ENDIF

  WaitTime 2;
  MoveL rt_PointC,v100,fine,Servo\WObj:=Workobject_Alim;

  ENDPROC



  PROC Mov_Beber()
    MoveJ Target_taza_1,v500,z50,Servo\WObj:=Workobject_taza_foga;
    MoveL Target_taza_2,v500,z50,Servo\WObj:=Workobject_taza_foga;
```

```
        MoveLdo
Target_taza_3_coger,v300,fine,Servo\WObj:=Workobject_taza_foga, pinza, 1;
        WaitTime 2;
        MoveL Target_taza_2,v500,fine,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_3,v500,z50,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_4,v500,z50,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_5,v500,z50,Servo\WObj:=Workobject_taza_foga;
        MoveJ inicio,v500,fine,Servo\WObj:=wobj0;

    ENDPROC

    PROC Mov_Beber_Recog()
        MoveJ Target_taza_5,v300,z50,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_6,v300,z50,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_7,v300,fine,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_8,v300,fine,Servo\WObj:=Workobject_taza_foga;
        MoveLdo
Target_taza_9_dejar_10,v300,fine,Servo\WObj:=Workobject_taza_foga,pinza,0;
        WaitTime 2;
        MoveL Target_taza_8,v500,z50,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_7,v500,z50,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_6,v500,z50,Servo\WObj:=Workobject_taza_foga;
        MoveL Target_taza_5,v500,z50,Servo\WObj:=Workobject_taza_foga;
        MoveJ inicio,v500,fine,Servo\WObj:=wobj0;
    ENDPROC



    PROC Mov_Comer()
        MoveL Target_1_Ten,v500,z50,Servo\WObj:=Workobject_tenedor;
        MoveLdo
Target_2_Ten,v300,fine,Servo\WObj:=Workobject_tenedor,pinza,1;
        WaitTime 2;
        MoveL Target_1_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
        MoveL Target_3_Ten,v300,z50,Servo\WObj:=Workobject_tenedor;
        MoveL Target_4_Ten,v300,z50,Servo\WObj:=Workobject_tenedor;
        MoveL Target_5_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
        MoveL Target_6_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
        MoveLdo Target_7_Ten,v300,fine,Servo\WObj:=Workobject_tenedor,
pinza,0;
        WaitTime 1;
        MoveL Target_6_Ten,v500,z50,Servo\WObj:=Workobject_tenedor;
        MoveL Target_8_Ten,v500,z50,Servo\WObj:=Workobject_tenedor;
        MoveLdo Target_9_Ten,v300,fine,Servo\WObj:=Workobject_tenedor,
pinza,1;
        WaitTime 1;
        MoveL Target_10_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
```

```
    MoveL Target_11_Ten,v300,z100,Servo\WObj:=Workobject_tenedor;
    MoveL inicio,v300,fine,Servo\WObj:=wobj0;
  ENDPROC

  PROC Mov_Comer_Recog()
    MoveL Target_11_Ten,v300,z100,Servo\WObj:=Workobject_tenedor;
    MoveL Target_10_Ten,v300,fine,Servo\WObj:=Workobject_tenedor;
    MoveLdo Target_9_Ten,v300,fine,Servo\WObj:=Workobject_tenedor,
pinza,0;
    waittime 1;
    MoveL Target_10_Ten,v500,fine,Servo\WObj:=Workobject_tenedor;
    MoveL Target_11_Ten,v500,z100,Servo\WObj:=Workobject_tenedor;
    MoveL inicio,v500,fine,Servo\WObj:=wobj0;
  ENDPROC



  PROC Accion_Boton_B()
    MoveJdo Target_Boton_Beb_Pos,v500,z50,Servo\WObj:=wobj0,pinza,1;
    MoveL Target_Boton_Beb_I,v500,z50,Servo\WObj:=wobj0;
    MoveL Target_Boton_Beb_F,v300,fine,Servo\WObj:=wobj0;
    MoveLdo Target_Boton_Beb_I,v500,z50,Servo\WObj:=wobj0,pinza,0;
    MoveJ inicio,v500,fine,Servo\WObj:=wobj0;
  ENDPROC

  PROC Accion_Boton_C()
    MoveJdo Target_Boton_Com_Pos,v500,z50,Servo\WObj:=wobj0, pinza, 1;
    MoveL Target_Boton_Com_I,v500,z50,Servo\WObj:=wobj0;
    MoveL Target_Boton_Com_F,v400,fine,Servo\WObj:=wobj0;
    MoveLdo Target_Boton_Com_I,v500,z50,Servo\WObj:=wobj0,pinza,0;
    MoveJ inicio,v500,fine,Servo\WObj:=wobj0;
  ENDPROC


ENDMODULE
```

### ControladorDoble: T_ROB1

```
MODULE Module1
  CONST robtarget
Target_inicio_doble_1:=[[275.837808993,0,820.538800818],[0.353553386,-
0.612372438,0.612372438,-
0.353553386],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target_10_CogerTaza:=[[-465.538799369,1164.38419416,-
590],[0.353553359,-0.612372439,-0.353553386,-0.612372454],[0,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget Target_11_CogerTaza:=[[-479.866064719,1053.190698596,-
383.987303164],[0.636049826,-0.289052243,-0.292503046,-0.652940569],[-1,-
1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_12_CogerTaza:=[[-479.866080823,1002.07580806,-
383.987309344],[0.707106763,0.000000018,0.000000051,-0.7071068],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_13_CogerTaza:=[[-357.416869775,1259.422012423,-
383.987309458],[0.707106774,0.000000046,0.000000025,-0.707106789],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_14_CogerTaza:=[[-100.000023092,1259.422008292,-
383.987309948],[0.70710682,0.000000008,-0.000000023,-0.707106743],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_15_CogerTaza:=[[-35,1259.422,-
196.414],[0.707106781,0,0,-0.707106781],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20_ServirTaza:=[[-249.850855243,1259.422009408,-
399.378510608],[0.707106791,-0.000000027,0.000000013,-0.707106771],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_21_ServirTaza:=[[-249.850792854,1355.228000044,-
824.977323389],[0.707106789,-0.000000029,-0.000000026,-0.707106773],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_22_ServirTaza:=[[-249.850813488,1355.227999855,-
824.977298258],[0.687664962,-0.164672111,-0.164672077,-0.687664965],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_25_ServirTaza:=[[-175.511578488,895.073092384,-
3.284439016],[0.707106743,-0.00000002,0.000000021,-0.707106819],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_26_ServirTaza:=[[-175.511575906,41.982344533,-
44.196377305],[0.707106803,-0.000000123,-0.000000161,-0.707106759],[-2,-
3,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_190:=[[-0.417360416,0.293073877,-
84.602897854],[0.707106948,-0.000000129,-0.000000076,-0.707106615],[-2,-
3,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_170:=[[-44.142085862,2.173993946,-
141.008400486],[0.707106715,0.000000082,0.000000089,-0.707106848],[-2,-
3,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_27_ServirTaza:=[[-
331.927017725,457.75270454,41.990365806],[0.707106736,0.00000007,0.00000
0131,-0.707106827],[-2,-3,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_28_ServirTaza:=[[-331.927023568,911.179592264,-
382.14030553],[0.383862783,-0.593842757,-0.107388428,-0.698904765],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_41_CogerTene:=[[-131.419555,1251.295865179,-
940.622381074],[0.000945847,-0.726171375,-0.68750773,0.002712996],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_42_CogerTene:=[[-
274.164794482,1251.295870088,-
```

```
852.170308372],[0.235516233,0.6828238,0.645910103,0.247151574],[0,0,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_43_CogerTene:=[[-
498.710705955,1251.295798226,-
555.129301841],[0.480031927,0.52105303,0.49217591,0.505802295],[-1,0,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_44_CogerTene:=[[-
387.483367114,1251.295801193,-
555.129301412],[0.480031946,0.521053021,0.49217592,0.505802277],[-1,0,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_46_CogerTene:=[[-266.647573,1251.295812719,-
851.265001348],[0.001607114,0.726176336,0.687506615,-0.000016555],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_47_CogerTene:=[[-20.789976863,1251.295857215,-
872.770509862],[0.000945846,-0.72617145,-0.687507651,0.002713027],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_48_CogerTene:=[[-20.790001281,1144.898992874,-
872.770506171],[0.003883308,0.716801589,0.697227247,-0.007387103],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_50_CogerTene:=[[-
132.199153891,1148.821899572,-
1023.099377135],[0.003883322,0.71680152,0.697227317,-0.007387073],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_51_CogerTene:=[[-132.199166994,815.085178582,-
1137.361656534],[0.00388334,0.716801568,0.697227268,-0.007387134],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_52_CogerTene:=[[-131.227280086,8.82606574,-
1026.483834503],[0.000458039,0.708173792,0.706025465,0.004208851],[1,2,-
4,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_54_CogerTene:=[[-60.732845213,8.827281466,-
1040.980568452],[0.000456945,0.708173668,0.706025582,0.004209953],[1,2,-
4,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_55_CogerTene:=[[-170.268663778,168.329019251,-
1106.273008863],[0.000457361,0.708174167,0.706025083,0.004209831],[1,2,-
4,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_57_CogerTene:=[[-205.73024747,356.742732885,-
1195.077510031],[0.00045735,0.708174277,0.706024971,0.004210038],[1,2,-
4,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_58_CogerTene:=[[-294.673280719,529.662991219,-
1213.973420417],[0.000457347,0.708174329,0.70602492,0.004210014],[1,2,-
4,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_59_CogerTene:=[[-294.673270267,727.052392904,-
1213.973417139],[0.000457336,0.708174432,0.706024815,0.004210045],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Target_16_CogerTaza:=[[-5,1259.422,-
141.108],[0.707106781,0,0,-0.707106781],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget Target_23_ServirTaza:=[[-349.387155512,1298.648577922,-
715.557790747],[0.687664967,-0.164672116,-0.164672092,-0.687664954],[0,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_24_ServirTaza:=[[-387.908447831,1419.580872335,-
604.579277372],[0.687664958,-0.164672148,-0.164672056,-0.687664964],[0,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_200:=[[-
0.417467952,0.29294618,12.993342392],[0.707106685,0.000000313,0.00000021
5,-0.707106878],[-2,1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40_CogerTene:=[[-20.789882265,1251.295811098,-
940.622499374],[0.000945606,-0.726171072,-0.68750805,0.002713098],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_49_CogerTene:=[[-20.789974195,1148.821890365,-
1015.069104671],[0.003883299,0.716801565,0.69722727,-0.007387097],[0,1,-
3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_53_CogerTene:=[[-20.058609456,8.826675675,-
1181.589883685],[0.000457452,0.708173657,0.706025598,0.004209344],[1,2,-
4,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Inicio_R2:=[[-436,1320.914184223,-
590],[0.683012702,-0.683012702,0.183012702,-
0.183012702],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


    PROC main()
        ConfJ\Off;
        ConfL\Off;
        SetDO Pos1R2,0;
        SetDO Pos2R2,0;
        SetDO Pos3R2,0;
        SetDO Pos4R2,0;
        SetDO pinza_doble_1,0;

        WaitDI Preparado,1;

        WHILE TRUE DO

            IF Select_Beb=1 AND Preparado=1 THEN
                ServirTaza;
                MoveJ
Target_Inicio_R2,v300,fine,Servo\WObj:=Workobject_taza_doble;
            ENDIF
            IF Select_Com=1 AND Preparado=1 THEN
                ServirTenedor;
                MoveJ
Target_Inicio_R2,v300,fine,Servo\WObj:=Workobject_taza_doble;
            ENDIF

            IF Repetir=1 THEN
```

```
        SetDO Pos1R2,0;
        SetDO Pos2R2,0;
        SetDO Pos3R2,0;
        SetDO Pos4R2,0;
        SetDO pinza_doble_1,0;
      ENDIF

      ConfJ\Off;
      ConfL\Off;
      WaitTime 0.01;
    ENDWHILE

  ENDPROC

  PROC ServirTaza()
    MoveJ Target_inicio_doble_1,v500,z50,Servo\WObj:=wobj0;
    MoveL
Target_10_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL
Target_11_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL
Target_12_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL
Target_13_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL
Target_14_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL
Target_15_CogerTaza,v500,fine,Servo\WObj:=Workobject_taza_doble;
    MoveLdo
Target_16_CogerTaza,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_do
ble_1,1;
    WaitTime 2;
    MoveL
Target_20_ServirTaza,v500,fine,Servo\WObj:=Workobject_taza_doble;
    MoveL
Target_21_ServirTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL
Target_22_ServirTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    WaitDO Pos4R2,1;
    MoveLdo
Target_23_ServirTaza,v300,fine,Servo\WObj:=Workobject_taza_doble,Pos3R2,1
;
    Waitdo Pos4R2,0;
    MoveL
Target_13_CogerTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
    MoveL
Target_25_ServirTaza,v500,z50,Servo\WObj:=Workobject_taza_doble;
```

```
        MoveL
Target_26_ServirTaza,v500,fine,Servo\WObj:=Workobject_taza_doble;
        MoveL Target_190,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveLdo
Target_200,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_doble_1,0;
        WaitTime 1;
        MoveL Target_190,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveL Target_170,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_26_ServirTaza,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_27_ServirTaza,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveJ
Target_28_ServirTaza,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveJ Target_inicio_doble_1,v500,fine,Servo\WObj:=wobj0;
    ENDPROC
    PROC ServirTenedor()
        MoveJ Target_inicio_doble_1,v500,fine,Servo\WObj:=wobj0;
        MoveLdo
Target_40_CogerTene,v300,fine,Servo\WObj:=Workobject_taza_doble,
pinza_doble_1,1;
        WaitTime 1.5;
        MoveL
Target_41_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_42_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
        MoveLdo
Target_43_CogerTene,v500,fine,Servo\WObj:=Workobject_taza_doble,Pos2R2,1
;
        WaitDO Pos1R2,1;
        MoveL
Target_44_CogerTene,v100,fine,Servo\WObj:=Workobject_taza_doble;
        WaitTime 1;
        MoveLdo
Target_43_CogerTene,v500,fine,Servo\WObj:=Workobject_taza_doble,Pos2R2,0
;
        MoveL
Target_46_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_41_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
        MoveLdo
Target_40_CogerTene,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_do
ble_1,0;
        WaitTime 1;
        MoveL
Target_47_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_48_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
```

```
        MoveLdo
Target_49_CogerTene,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_do
ble_1,1;
        WaitTime 1;
        MoveL
Target_50_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_51_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_52_CogerTene,v500,z50,Servo\WObj:=Workobject_taza_doble;
        MoveLdo
Target_53_CogerTene,v300,fine,Servo\WObj:=Workobject_taza_doble,pinza_do
ble_1,0;
        waittime 1;
        MoveL
Target_54_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_55_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_57_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_58_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveL
Target_59_CogerTene,v500,z100,Servo\WObj:=Workobject_taza_doble;
        MoveJ Target_inicio_doble_1,v500,z100,Servo\WObj:=wobj0;
    ENDPROC

ENDMODULE
```

### ControladorDoble: T_ROB2

```
MODULE Module1
    CONST robtarget Target_Inicio_R3:=[[240.204591746,-381.206623507,-
350],[0.683235787,-0.182178067,-0.683235795,-0.182178097],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_3_1:=[[147.059909743,-272.765800824,-
201.390007742],[0.945074737,-0.251994905,-0.201136773,-0.053631215],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_3_2:=[[147.059912019,55.167302479,-
172.2876073],[0.94507474,-0.251994889,-0.20113678,-0.053631222],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_3_3:=[[13.804396275,55.16730724,-
80.738317646],[0.999616331,-0.001388187,-0.024009977,-0.013739875],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_3_5:=[[13.80440231,-232.7798948,-
201.761509368],[0.99961633,-0.00138821,-0.024010042,-0.013739879],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget Target_3_6:=[[-0.423882641,-419.437973805,-
425.396212153],[0.968357819,0.001751342,-0.249183515,-0.01369829],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_3_7:=[[-21.593672568,-419.437978165,-
426.971849535],[0.968357812,0.001751356,-0.249183541,-0.013698287],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_3_8:=[[-21.593656387,-419.437980043,-
426.971824673],[0.813041773,-0.439751472,-0.339033709,-0.175036747],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60:=[[207.554285984,100.028396725,-
912.896464146],[0.604831024,0.36630507,-0.34215955,-0.618810851],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[143.792204529,-52.361094043,-
871.105094797],[0.604831019,0.366305076,-0.342159568,-0.618810844],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_30:=[[82.558595239,-108.730491841,-
768.4856835],[0.604831022,0.366305088,-0.342159542,-0.618810847],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40:=[[-123.732904889,-108.730479816,-
487.746091698],[0.604831021,0.366305048,-0.342159578,-0.618810853],[0,-2,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Target_3_4:=[[14.774594636,55.167319952,10.543981706],[0.999898884,-
0.001671313,-0.003392822,-0.013708316],[-1,-
2,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[143.792205697,100.028386244,-
871.10509325],[0.604831025,0.366305082,-0.342159553,-0.618810842],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50:=[[-123.733,-122.982,-
410.878],[0.604830672,0.366305689,-0.342158943,-0.618811165],[0,-2,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_70:=[[188.239486081,-350.999993557,-
350],[0.686306887,-0.195718448,-0.67948093,-0.170243394],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_Ini_R3:=[[-105.894184223,-351,-
350],[0.612372436,-0.612372436,-
0.353553391,0.353553391],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    PROC main()
        ConfJ\Off;
        ConfL\Off;
        SetDO Pos1R2,0;
        SetDO Pos2R2,0;
        SetDO Pos3R2,0;
        SetDO Pos4R2,0;
        SetDO pinza_doble_2,0;

        WaitDI Preparado,1;
```

```
    WHILE TRUE DO

      IF Select_Beb=1 AND Preparado=1 THEN
        Servir_J;
        MoveJ Target_Ini_R3,v300,fine,Servo\WObj:=Workobject_Servir;

      ENDIF
      IF Select_Com=1 AND Preparado=1 THEN
        Servir_P;
        MoveJ Target_Ini_R3,v300,fine,Servo\WObj:=Workobject_Servir;

      ENDIF

      IF Repetir=1 THEN

        SetDO Pos1R2,0;
        SetDO Pos2R2,0;
        SetDO Pos3R2,0;
        SetDO Pos4R2,0;
        SetDO pinza_doble_2,0;
      ENDIF
      waittime 0.1;
    ENDWHILE


  ENDPROC
  PROC Servir_J()
    MoveL Target_Inicio_R3,v300,fine,Servo\WObj:=Workobject_Servir;
    MoveJ Target_3_1,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveJ Target_3_2,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_3,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveLDO
Target_3_4,v500,fine,Servo\WObj:=Workobject_Servir,pinza_doble_2,1;
    WaitTime 2;
    MoveL Target_3_3,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_5,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveLDo
Target_3_6,v500,fine,Servo\WObj:=Workobject_Servir,Pos4R2,1;
    waitDo Pos3R2,1;
    MoveL Target_3_7,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_8,v20,fine,Servo\WObj:=Workobject_Servir;
    WaitTime 2;
    MoveLdo Target_3_7,v500,fine,Servo\WObj:=Workobject_Servir,Pos4R2,0;
    MoveL Target_3_6,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_5,v500,fine,Servo\WObj:=Workobject_Servir;
    MoveL Target_3_3,v500,fine,Servo\WObj:=Workobject_Servir;
```

```
        MoveLDO
Target_3_4,v500,fine,Servo\WObj:=Workobject_Servir,pinza_doble_2,0;
        WaitTime 2;
        MoveL Target_Inicio_R3,v300,fine,Servo\WObj:=Workobject_Servir;
    ENDPROC
    PROC Servir_P()
        MoveJ Target_Inicio_R3,v500,fine,Servo\WObj:=Workobject_Servir;
        MoveJ Target_60,v500,fine,Servo\WObj:=Workobject_Servir;
        MoveLDo
Target_10,v500,fine,Servo\WObj:=Workobject_Servir,pinza_doble_2,1;
        WaitTime 1.5;
        MoveL Target_20,v500,fine,Servo\WObj:=Workobject_Servir;
        MoveL Target_30,v500,fine,Servo\WObj:=Workobject_Servir;
        MoveL Target_40,v500,fine,Servo\WObj:=Workobject_Servir;
        WaitDO Pos2R2,1;
        MoveLDo Target_50,v500,fine,Servo\WObj:=Workobject_Servir,Pos1R2,1;
        WaitDO Pos2R2,0;
        MoveJ Target_60,v500,fine,Servo\WObj:=Workobject_Servir;
        MoveLDo
Target_10,v500,fine,Servo\WObj:=Workobject_Servir,pinza_doble_2,0;
        WaitTime 1;
        MoveL Target_60,v500,fine,Servo\WObj:=Workobject_Servir;
        MoveL Target_Inicio_R3,v500,fine,Servo\WObj:=Workobject_Servir;
    ENDPROC


ENDMODULE
```