Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

INTEGRACIÓN DE ROBOT COLABORATIVO EN SISTEMA DE FABRICACIÓN FLEXIBLE EN EL CONTEXTO DE LA INDUSTRIA 4.0

Estudiante *Ane Ayerra González*
Directora *Mª Isabel Sarachaga González*
Departamento *Ingeniería de Sistemas y Automática*
Curso académico *2020-2021*

Bilbao, 21 de septiembre del 2021

RESUMEN:

En este Proyecto Fin de Máster se plantea y resuelve el ensamblaje de un producto mediante un robot colaborativo en el ámbito de la Industria 4.0. Se trata de un proceso de fabricación que va a formar parte del demostrador de un proyecto de investigación centrado en la gestión de Sistemas de Fabricación Flexible.

En este proceso concreto, un operario solicita servicios de ensamblado, especificando para ello la cantidad de productos que el brazo robótico debe montar. Durante la realización del proceso, la estación de trabajo debe reportar los tiempos de inicio y finalización de la fabricación de cada producto. Para ello, el proceso será primero simulado en un software de simulación y después se deberán crear dos conexiones a partir de él, una con el PLC *(Controlador Lógico Programable)* y otra con el sistema robótico. De esta manera, podrá asegurarse que la petición del operario llegue al sistema robótico y que al final del proceso reciba la información requerida.

El principal beneficio de este trabajo es la obtención del asset correspondiente al robot que constituye el primer paso hacia la obtención del Componente I4.0 y su integración en el contexto de la Industria 4.0.

Palabras clave: Industria 4.0, robot colaborativo, software de simulación, PLC.

LABURPENA:

Master Amaierako Proiektu honetan, produktu baten mihiztatzea planteatzen eta ebazten da, Industria 4.0 arloan elkarlanean aritzeko robot baten bidez. Fabrikazio malguko sistemen kudeaketan oinarritutako ikerketa-proiektu baten erakuslean parte hartuko duen fabrikazio-prozesu bat da.

Prozesu zehatz horretan, langile batek mihiztatze-zerbitzuak eskatzen ditu, beso robotikoak zenbat produktu muntatu behar dituen zehaztuz. Prozesua egin bitartean, lan-estazioak produktu bakoitzaren fabrikazioaren hasiera- eta amaiera-denborak adierazi behar ditu. Horretarako, lehenik simulazio-software batean simulatuko da prozesua, eta gero bi konexio sortuko dira softwaretik abiatuta, bata PLCarekin *(Kontrolatzaile Logiko Programagarria)* eta beste sistema robotikoarekin. Horrela ziurtatu ahal izango da langilearen eskaera sistema robotikora iristen dela eta prozesuaren amaieran eskatutako informazioa jasotzen duela.

Lan honen onura nagusia robotari dagokion asset bat lortzea da, I4.0 osagaia lortzeko eta 4.0 Industriaren testuinguruan integratzeko lehen urratsa dena.

Hitz gakoak: Industria 4.0, elkarlanean aritzeko robota, simulazio softwarea, PLC.

ABSTRACT:

In this Final Master Project, the assembly of a product is proposed and solved by a collaborative robot in the field of Industry 4.0. It is a manufacturing process that will be part of the demonstration of a research project focused on the management of Flexible Manufacturing Systems.

In this specific process, an operator requests assembly services, specifying the quantity of products that the robotic arm must assemble. During the process, the workstation must report the start and end times of the manufacture of each product. To do this, the process will first be simulated in a simulation software and then two connections must be created from it, one with the PLC *(Programmable Logic Controller)* and another with the robotic system. In this way, it can be confirmed that the operator's request reaches the robotic system and that at the end of the process it receives the required information.

The main benefit of this work is obtaining the asset corresponding to the robot that constitutes the first step towards obtaining Component I4.0 and its integration in the context of Industry 4.0.

Key words: Industry 4.0, collaborative robot, simulation software, PLC.

Contenido

Índice de figuras

Índice de tablas

1. MEMORIA

1.1 Introducción

El trabajo que se presenta en esta memoria está enmarcado dentro de un proyecto de investigación que tiene por objetivo la investigación y desarrollo de soluciones para la gestión de Sistemas de Fabricación Flexible. En el contexto de la Industria 4.0, se define un Sistema de Fabricación Flexible como aquél que tiene la capacidad de adaptarse de forma automática a cambios de contexto no planificados durante el proceso de fabricación, como pueden ser paradas de máquinas no planificadas (por ejemplo, por avería), o la entrada de pedidos de última hora que impliquen una replanificación de la producción.

El desarrollo de Sistemas de Fabricación Flexible conlleva que las máquinas y elementos que forman parte del sistema tengan la capacidad de interactuar e informarse unos a otros ante cambios en su estado. Para ello, la arquitectura de referencia para la Industria 4.0, RAMI 4.0 *(Reference Architectural Model Industrie)*, propone el concepto de Componente I4.0, que consta de dos partes:

- El activo o asset: se trata de cualquier entidad que participa en el proceso de fabricación. Por lo general, los assets suelen ser activos físicos (en este caso será el robot UR3e quién hará de asset).
- El Asset Administration Shell: los activos cuentan con una representación virtual que les permite conectarse a la Industria 4.0 llamada Asset Administration Shell. Esta representación virtualizada permite la comunicación entre diferentes activos de una fábrica, ya sea ofreciendo servicios o solicitándolos.

En el caso de este proyecto de investigación, se ha decidido utilizar la tecnología de sistemas multiagente para implementar el Asset Administration Shell de los activos del Sistema de Fabricación Flexible. Esta tecnología se basa en el uso de agentes, entidades software que cuentan con la capacidad de interactuar entre sí, pero

también tienen la capacidad de tomar decisiones. Gracias a esta toma de decisiones se proporciona flexibilidad al sistema, dado que, si por ejemplo se avería un robot que tenía que hacer ciertas operaciones de un proceso, los agentes serán quienes detecten la incidencia y decidan como actuar frente a ella. Esto es lo que lo convierte en un Sistema de Fabricación Flexible.

Por otra parte, la implementación de los assets que desarrollan los servicios de fabricación sigue siendo una parte fundamental para la puesta en marcha de los Sistemas de Fabricación Flexible. Por ello, este trabajo se centra en la puesta en marcha de una estación de trabajo mediante el uso de herramientas de simulación para el desarrollo de la solución.

1.2 Contexto

A continuación, se tratará el concepto de Industria 4.0 y su arquitectura de referencia RAMI 4.0, con el fin de contextualizar lo anteriormente mencionado.

1.2.1 Industria 4.0

El término "Industria 4.0" aparece por primera vez en Hannover (Alemania), en el año 2011 en una feria dedicada a la tecnología industrial. Dicho concepto propone utilizar la ayuda de la tecnología de la información y comunicación para la interconexión inteligente de máquinas y procesos en la industria. Estas redes inteligentes pueden ser utilizadas por las empresas de diferentes formas, incluyendo las que serán nombradas a continuación [1]:

- Producción flexible: al fabricar un producto, hay muchas empresas que participan en dicha producción paso a paso. Gracias a la red digital, dichos pasos pueden estar mejor coordinados y tener una mejor planificación para la carga de la máquina.
- Fábrica convertible: el mercado actual exige a las empresas la capacidad de producir productos individualizados en pequeñas cantidades a precios asequibles. Para ello, las líneas de producción tienen que construirse en módulos que permitan adaptar rápidamente los procesos de fabricación, mejorando así la productividad.
- Soluciones orientadas al cliente: habrá mayor comunicación entre los consumidores y los clientes. Por una parte, los propios clientes podrán diseñar sus productos, por ejemplo, unas zapatillas diseñadas al gusto del cliente. Al mismo tiempo, los productos inteligentes que ya se han entregado podrán mandar información útil al fabricante, para así mejorar los futuros productos y poder ofrecer servicios novedosos al cliente.
- Logística optimizada: las rutas de entrega pueden calcularse con precisión gracias a los algoritmos, las máquinas pueden informar de diferentes aspectos a los empleados, por ejemplo, cuando necesitan nuevo material.

- Uso de datos: se analizarán los datos sobre el proceso de producción y el estado del producto. Mediante este análisis se sabrá cómo hacer un producto de una manera más eficiente. Además, gracias a la base de modelos y servicios completamente nuevos, se podrá corregir el error en un modelo o servicio antes del fallo total mediante el mantenimiento predictivo.

Para entender mejor la Industria 4.0 y cuáles son en realidad las novedades que conlleva, se hará una comparativa con la Industria 3.0, lo cual ayudará a ver las diferencias entre ambos.

La industria 3.0 fue un gran paso adelante en el que la aparición del ordenador y la automatización gobernaron la escena industrial. Fue durante esa época cuando se utilizaron cada vez más robots en los procesos para realizar las tareas que realizaban los humanos.

En cambio, la cuarta era de la industria es la era de los sistemas ciberfísicos *(Cyber-physical Systems, CPS)*. Los CPS se componen de una parte física que realiza servicios o funcionalidades y otra virtual que se encarga de la representación y las capacidades de comunicación de los elementos del sistema. Los CPS pueden ser máquinas inteligentes, sistemas de almacenamiento e instalaciones de producción, entre otros, capaces de intercambiar información de forma autónoma y desencadenar acciones. Este intercambio de información se realiza mediante el Internet Industrial de las cosas *(Industrial Internet of Things, IIOT)* en el que miles de sensores trabajan en tiempo real y transfieren los datos a un servidor local o a un servidor en la nube donde se realiza el análisis de dichos datos. Dicho análisis ayuda a las industrias a mejorar los procesos de fabricación, el uso de materiales, la cadena de suministro y la gestión del ciclo de vida del producto.

En pocas palabras, la diferencia básica entre las dos eras industriales es que las máquinas funcionan de forma autónoma sin la intervención de un humano en la Industria 4.0. Mientras que en la industria 3.0 las máquinas solo se automatizan en parte mediante la utilización de autómatas que permiten el control de robots y otros equipamientos industriales. En algunos casos, las mismas máquinas de las últimas

etapas de la industria 3.0 se pueden actualizar a la maquinaria I4.0 haciéndoles intercambiar datos con la ayuda de IIOT y varios sensores [2].

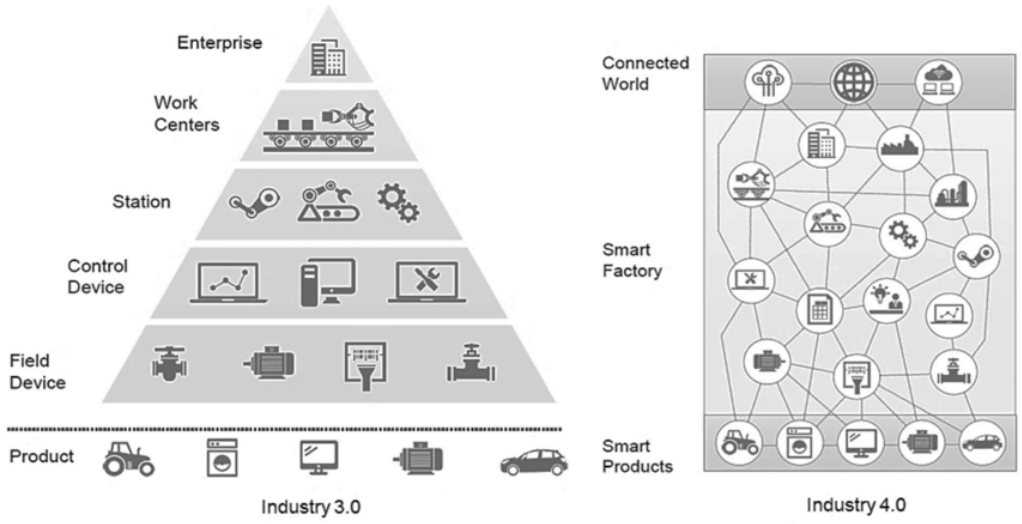

Figura 1.- Esquema comparativo entre Industria 3.0 e Industria 4.0 [3]

La estrategia I4.0 es compatible con la integración de CPS e IIOT para mejorar la productividad, la eficiencia y la flexibilidad de los procesos de producción.

Gracias a la integración digital que supone la Industria 4.0, los negocios pueden llevarse a cabo en un ciclo continuo, ya que el acceso a la información puede hacerse en tiempo real entre los mundos físico y digital. Este intercambio continuo de información se hace a través de una serie de pasos que se conoce como PDP *(physical-to-digital-to-physical)* y se realiza de la siguiente manera:

- Del mundo físico al digital: la información se recoge del mundo físico e inmediatamente se crea un registro digital.
- De digital a digital: la información recogida del mundo físico se interpreta mediante la analítica avanzada, análisis de escenarios e inteligencia artificial para así decidir cuál es la información que se necesitará después.
- Del mundo digital al físico: las decisiones tomadas en el mundo digital se pasan al mundo físico traduciéndolas mediante algoritmos, para después efectuar cambios en el mundo físico.

Este movimiento no solo afectará a los procesos de fabricación. Realmente, su impacto va mucho más allá, afectando incluso a la sociedad. Por esta razón, es importante entender el efecto que tendrá en diferentes niveles, como en grandes ecosistemas, a nivel organizacional o a nivel individual (ya sea en empleados o clientes) [4][5]:

- Ecosistemas: esta revolución industrial afecta a los diferentes agentes del ecosistema (proveedores, clientes, consideraciones regulatorias, inversores...), además de cambiar el modo en el que las empresas operan. Gracias a estas nuevas tecnologías, podrá haber interacciones entre los diferentes puntos de una red.
- Organizaciones: las organizaciones podrán ser más receptivas, ya que pueden acceder a datos en tiempo real y ajustarse a ellos. Gracias a dichos datos, la organización será capaz de reducir los riesgos en materia de productividad.
- Individuos: la Industria 4.0 tendrá un diferente impacto para cada grupo de personas. Los empleados notarán un gran cambio en el trabajo que van a realizar y en el modo de hacer dicho trabajo, mientras que para los clientes supondrá una mejora en los productos y servicios, ya que podrán tener una mayor personalización.

1.2.2 RAMI 4.0

RAMI 4.0 [6] es una arquitectura de referencia que propone cómo abordar la Industria 4.0 de una manera estructurada. Uno de los objetivos principales de RAMI 4.0 es que todos los miembros involucrados en la Industria 4.0 tengan un marco común donde poder entenderse entre sí.

RAMI 4.0 consta de un sistema de coordenadas tridimensional que describe todos los aspectos cruciales de la Industria 4.0. De esta manera, las interrelaciones complejas se dividen en grupos más pequeños y simples.

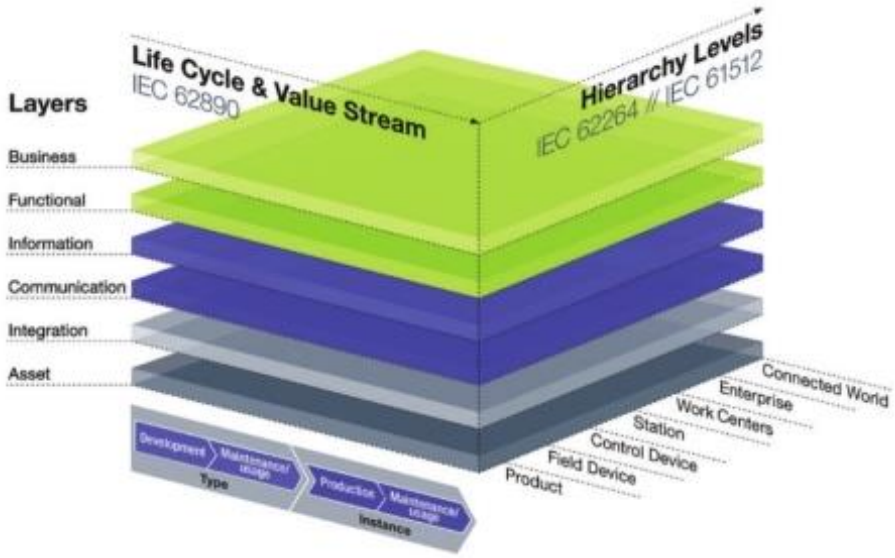

Figura 2.- Sistema tridimensional RAMI 4.0 [6]

1. El eje de ciclo de vida del producto se basa en el estándar IEC 62890, y es utilizado en la medición, control y automatización de procesos industriales. Además, se hace una distinción entre "tipos" e "instancias". Un "tipo" se convierte en una "instancia" cuando se han completado el diseño y la creación de prototipos y se está fabricando el producto real.
2. Las seis capas en el eje vertical describen la descomposición de una máquina en sus propiedades, estructuradas capa por capa, es decir, el mapeo virtual de una máquina. Estas representaciones se originan en la tecnología de la información y las comunicaciones, donde las propiedades de los sistemas complejos se suelen dividir en capas.
3. El eje de los niveles de jerarquía acoge la serie de estándares internacionales IEC 62264 y IEC 61512 para sistemas de control e IT *(Tecnología de la Información)* empresariales. Estos niveles de jerarquía representan las diferentes funcionalidades dentro de las fábricas o instalaciones. No obstante, para representar el entorno de Industria 4.0, estas funcionalidades se han ampliado para incluir piezas de trabajo, etiquetadas como "Producto" y la conexión a Internet de las cosas y servicios, etiquetado como "Mundo conectado".

Este modelo integra diferentes perspectivas de usuario y proporciona una forma común de ver las tecnologías de la Industria 4.0. Por lo tanto, RAMI 4.0 puede considerarse como un mapa de soluciones de Industria 4.0. Es una orientación para trazar los requisitos de los sectores junto con los estándares nacionales e internacionales para definir y desarrollar aún más la Industria 4.0.

1.3 Objetivos y alcance

En este apartado de la memoria se expondrá el objetivo principal del proyecto. Además, también se describirá el alcance del trabajo, tanto en el ámbito espacial como en el temporal.

1.3.1 Objetivos

El objetivo principal de este proyecto es la puesta en marcha del robot colaborativo UR3e y su integración en un sistema de fabricación flexible, todo ello en el contexto de la Industria 4.0. En concreto, este robot deberá completar un proceso de ensamblado en el que el número de unidades a ensamblar podrá variar entre 1 y 6 unidades en cada petición.

Para ello, primero habrá que estudiar el nuevo robot con objeto de ponerlo en marcha y saber cómo programarlo para que pueda ejecutar este servicio y comunicarse con otros dispositivos. Dicho proceso será previamente simulado en un software de simulación.

Esta estación recibirá las peticiones de servicio de ensamblado por parte de un agente. No obstante, con objeto de centrarse en el objetivo principal de este trabajo, en la solución presentada las peticiones son introducidas manualmente por un operario mediante un fichero de texto que permite hacer pruebas enviando la misma información que enviaría el agente. Esta información se procesará mediante un PLC para después mandar la orden al robot.

1.3.2 Alcance

Dentro del ámbito de la Industria 4.0 y los Sistemas de Fabricación flexible, y más concretamente, en el contexto del proyecto de investigación en el que está englobado este trabajo, el objetivo se centra exclusivamente en la puesta en marcha de un activo de fábrica, como es un robot, para enriquecer el demostrador utilizado en este proyecto de investigación. Por este motivo, como se ha explicado en los objetivos,

este trabajo se centra en la implementación de los servicios de fabricación, en este caso de ensamblado, proporcionados por el robot, quedando la tecnología de sistemas multiagente fuera del alcance de este trabajo. Además, cabe destacar que, con objeto de poder hacer pruebas relacionadas con la flexibilidad del proceso de fabricación, los servicios ofrecidos por esta estación deben ser idénticos a los que proporciona otra estación de trabajo disponible en el demostrador, pero que cuenta con un brazo robótico no colaborativo.

Este proyecto ha sido realizado en el Grupo de Control e Integración de Sistemas *(GCIS)* de la Escuela de Ingeniería de Bilbao, en uno de los laboratorios de investigación del departamento de Ingeniería de Sistemas y Automática. Dicho trabajo es parte de un proyecto de investigación, por lo que todo el trabajo realizado está destinado a la investigación.

El cumplimiento de los objetivos se ha realizado en un tiempo de 5 meses, desde abril hasta mediados de septiembre, cumpliendo así con las 600 horas asignadas a dicho trabajo (repartidas entre investigación, desarrollo y puesta en marcha). Como se ha comentado anteriormente, este proyecto está enmarcado en un proyecto de investigación, que a su vez entronca con la línea de investigación que mantiene el grupo GCIS en el ámbito de los Sistemas de Fabricación Flexible. Por ese motivo, cabe destacar que se ha partido de una base de conocimiento generada en el grupo de investigación, y que los desarrollos conseguidos en este trabajo servirán de base para futuros trabajos de otras personas en el grupo de investigación (alumnos, profesores, investigadores...).

1.4 Beneficios que aporta el trabajo

El principal beneficio de este proyecto es la obtención del asset correspondiente al robot UR3e que participa en un Sistema de Fabricación Flexible en el que se realiza un proceso de ensamblado. A partir de dicho asset, se podrá realizar su representación virtual (Asset Administration Shell) en trabajos futuros, convirtiéndolo así en un Componente I4.0 que se comunique con los diferentes activos de la fábrica ofreciendo servicios o solicitándolos. Por lo tanto, dicho asset constituye un primer paso hacia la integración del Sistema de Fabricación Flexible en el contexto de la Industria 4.0 con todos los beneficios que ello reporta [7].

Otro beneficio importante está relacionado con el know-how adquirido por el grupo de investigación en lo que respecta al propio robot UR3e, al software de simulación RoboDK empleado en crear un gemelo digital, en las comunicaciones con el PLC usando el protocolo OPC UA *(Arquitectura Unificada de Comunicaciones de Plataforma Abierta)* y en la ejecución del programa en el robot desde el software de simulación, todos ellos ámbitos en los que el grupo de investigación deseaba ampliar conocimientos poniéndolos en práctica en un caso de estudio.

1.5 Análisis de alternativas

En este apartado de la memoria, se analizarán las diferentes alternativas que se han barajado en este proyecto tanto para el sistema robótico como para el software de simulación, atendiendo a los requisitos que se deben garantizar en cada caso. De esta manera se podrán comparar las diferentes opciones y elegir la más adecuada para el proyecto.

1.5.1 Sistema robótico

Tal y como se recoge en el apartado 1.3.2, los servicios ofertados por esta estación deben ser idénticos a los ya ofrecidos por una estación del demostrador. Por ese motivo, lo ideal es utilizar un brazo robótico que presente unas características similares o superiores en cuanto a alcance y capacidad de carga a las del robot de la otra estación. De esta manera, los siguientes requisitos son obligatorios: un radio de alcance en torno a 541mm, una carga útil de al menos 3Kg, una superficie de colocación similar a 179x179mm y conexión a PLC.

No obstante, hay algunos requisitos no obligatorios que sería interesante que el robot pudiera cumplir pensando en proyectos futuros, como, por ejemplo, que sea un robot colaborativo y que exista la posibilidad de poder controlar los ejes del robot desde un PLC o similar. Teniendo esto en consideración, lo ideal es utilizar un robot colaborativo, también llamados cobots, ya que pueden trabajar en un entorno donde también puede haber un operario.

Además, a la hora de elegir el robot, se ha de tener en cuenta que el laboratorio ya cuenta con un robot KUKA, por lo que se ha decidido elegir uno del fabricante Universal Robots o del fabricante ABB, y buscar en dichos fabricantes el robot que presente las características que más se asemejen a las proporcionadas por el robot KUKA. A continuación, se analizarán dichos fabricantes para después comparar los dos robots que mejor satisfagan los requisitos establecidos.

UNIVERSAL ROBOTS

Los robots colaborativos de Universal Robots ofrecen la capacidad anteriormente mencionada de un entorno compatible entre el robot y el humano, y, además, sus características principales hacen que sea muy atractivo para el comprador. Dichas características son las siguientes [8]:

- Programación sencilla: gracias a la tecnología que utilizan, los cobots pueden ser fácilmente programados por operarios sin una gran experiencia. La tableta que tiene con visualización 3D, hace que moverlo a los puntos deseados sea muy sencillo.
- Configuración rápida: gracias a su sencilla configuración, Universal Robots ha conseguido reducir a unas pocas horas la configuración de sus cobots, que puede llegar a ser incluso de días en el caso de otros fabricantes.
- Automatización flexible: son robots que no se limitan a una única aplicación, además de poder moverlos entre diferentes procesos de forma fácil y rápida, ya que son livianos y no ocupan demasiado espacio. Por todo ello, se pueden utilizar en procesos de pequeños lotes y frecuentes cambios.
- Colaborativo y seguro: está preparado para todos aquellos trabajos peligrosos y aburridos que son muy repetitivos para los humanos. Gracias a su sistema de seguridad, no es necesario trabajar con protección cerca de ellos, siempre y cuando se haga una evaluación de riesgos anteriormente.

Este fabricante danés ofrece robots colaborativos con cuatro opciones de carga útil diferentes: 3, 5, 12.5 y 16 Kg., permitiendo una amplia variedad de aplicaciones. Todos ellos cuentan con seis grados de libertad, una increíble flexibilidad y una fácil integración en los entornos de producción existentes.

El modelo UR3e [9] puede manejar una carga útil de hasta 3Kg y cuenta con un alcance máximo de 500mm. Este modelo es el que más se asemeja al modelo utilizado en el laboratorio, ya que cuenta con la misma carga útil y su alcance es

mínimamente inferior. Además, también cuenta con comunicación Profinet y Ethernet/IP, lo que hace posible la conexión a un PLC.

Cuenta con una rotación ±360 grados en todas las articulaciones y gracias a su rotación infinita en el extremo es ideal para ensamblajes ligeros y aplicaciones de atornillado.

Figura 3.- Brazo robótico UR3e [9]

ABB

El fabricante suizo ABB está abierto a más ámbitos que Universal Robots, y lleva más años en el sector, 130 años para ser exactos, siendo uno de los pioneros tanto en robótica como en automatización [10].

Gracias a contar con la experiencia de personal del laboratorio con este fabricante, se ha podido llegar relativamente rápido a la conclusión de que el robot más parecido al KUKA del laboratorio es el IRB1100. Las principales características de dicho modelo son las siguientes [11][12]:

- Cuenta con tiempo de ciclo hasta un 35% más rápido en comparación con los robots de su clase y una gran repetibilidad que hace posible conseguir un alto rendimiento en fabricación de alta calidad.
- Tiene la mayor carga útil entre los robots de su clase, 4Kg., y un gran alcance, 580mm.

- Gracias a su compacto y ligero diseño, su implementación es más fácil en diferentes entornos. Es adecuado para aplicaciones rápidas de montaje, recogida y colocación de materiales, ya que cuenta con un nuevo controlador con capacidades avanzadas para ello.

Todas estas características hacen que el IRB1100 sea ideal para las siguientes aplicaciones:

- Montaje y prueba.
- Carga y descarga.
- Atornillado.
- Inserción de goma.
- Pulido, trituración, desbardo y lijado.

Figura 4.- Brazo robótico IRB 1100 [11]

ANÁLISIS COMPARATIVO

La Tabla 1 recoge las principales características de los robots analizados, así como las proporcionadas por el robot KUKA existente en el laboratorio.

En este proyecto se ha decidido utilizar el UR3e por diferentes motivos que se explicarán a continuación. Por un lado, se puede observar que el robot del fabricante ABB dispone de una mayor carga útil, mayor que la de los demás, siendo ésta de 4Kg frente a los 3Kg que tienen los robots KR3 540 y URe3. Esta característica no tiene gran repercusión para este proyecto, ya que las cargas que el robot tiene que coger son relativamente pequeñas, por lo que la ventaja que da el robot ABB con su mayor carga útil no será necesaria.

En cuanto al alcance y la conectividad se puede observar que no hay grandes diferencias, por lo que será interesante el bloque de requisitos no obligatorios para poder tomar una decisión. La ventaja más significante del URe3 es la de ser un robot colaborativo, lo que lo hace claramente diferente al KUKA y al ABB, y muy interesante para el proyecto.

Tabla 1.- Tabla comparativa del sistema robótico

ROBOTS	KR3 540 (KUKA)	UR3e (Universal Robots)	IRB 1100 (ABB)
REQUISITOS OBLIGATORIOS			
Radio de alcance	541 mm	500mm	580mm
Capacidad de carga	3Kg	3Kg	4Kg
Superficie de colocación	179mm x 179mm	180mm (diam.)	160mm x 172mm
Soporte de protocolos de comunicación en modo esclavo/dispositivo (respecto al PLC)	SÍ	Si, Profinet, Ethernet/IP	Si, Profinet, Ethernet/IP
REQUISITOS NO OBLIGATORIOS			
Capacidad de control del robot externalizada (control de ejes desde el PLC o similar)	SÍ	NO	SÍ
Robot colaborativo	NO	SÍ	NO

1.5.2 Software de simulación

Antes de empezar a trabajar con el robot, siempre es mejor simular el proceso en un gemelo digital, ya que así se podrán identificar los posibles problemas y buscarles una solución. De esta manera se podrán evitar daños materiales provocados por colisiones y ahorrar tiempos de prueba con el robot real que siempre son más lentas que en un software de simulación.

Aunque Universal Robots dispone de su propio simulador, éste no permite simular el robot en su entorno de trabajo haciendo así muy difícil fijar las diferentes posiciones del proceso. Por esta razón, es conveniente elegir un software que permita la simulación del robot en el entorno donde trabajará después.

Antes de elegir el software de simulación que se utilizará, es conveniente fijar los requisitos que éste debe cumplir. Como se ha comentado, es imprescindible que se pueda simular el robot en el mismo entorno de trabajo que tendrá en realidad, para que así todos los movimientos que se hagan sean los mismos que hará después el robot real.

También es importante que tenga una interfaz gráfica intuitiva, ya que así se ahorrará el tiempo de aprendizaje. Además, sería aconsejable que los programas generados por el software de simulación se pudieran después convertir en scripts para que el robot pudiera leerlos y ejecutarlos.

Para terminar con los requisitos obligatorios, es necesario que cuente con una biblioteca con diferentes robots, y que puedan importarse modelos 3D para así crear el entorno al gusto de cada usuario.

En cuanto a los requisitos no obligatorios, será interesante que permita el flujo de material, ya que así se podrá observar de una manera más clara el proceso. También es aconsejable que sea posible crear una conexión entre el software de simulación y el sistema robótico para así poder ejecutar el proceso desde el simulador sin la necesidad de convertirlo en scripts y pasarlos al robot. Por último, será interesante poder utilizar lenguajes de programación más avanzados como Python o C++ entre otros.

En este caso las dos opciones que se han barajado son RoboDK y Tecnomatix, los cuales se analizarán a continuación para elegir la mejor opción para este proyecto.

ROBODK

El software de simulación RoboDK permite programar cualquier robot industrial con el mismo entorno de simulación que tendrá en la realidad. Gracias a la programación fuera de línea, se puede simular cualquier brazo robótico sin conexión, para después implementar dichos programas en el sistema robótico.

Las principales características y beneficios que aporta dicho software son las siguientes [13]:

- Gracias a su interfaz gráfica intuitiva es posible programar el robot sin necesidad de codificación, para después obtener programas .SCRIPT partiendo de ellos. Dichos programas en .SCRIPT podrán ser utilizados en el robot bien conectando el RoboDK con el robot o bien copiándolo en una memoria USB.
- Cuenta con una amplia gama de robots, herramientas, actuadores y sensores para poder utilizar en el proyecto.
- Es compatible con modelos 3D de archivos STL, STEP e IGES, pudiendo utilizarlos para crear nuevas herramientas. De esta manera se podrá crear el entorno de trabajo exactamente igual que el real.
- Permite evitar errores del robot como pueden ser las singularidades, límites de articulaciones, límites de alcance y colisiones. Se evitará así posibles futuros gastos provocados por colisiones o posibles fracturas del robot provocadas por límites de articulaciones.
- También cuenta con un uso avanzado para aquellos que quieran programar utilizando un lenguaje de programación (Python, C, Visual Basic, C++ y Matlab).
- En el entorno de trabajo que se cree, pueden ser simulados varios robots a la vez, pudiendo sincronizarlos para que trabajen en paralelo.

TECNOMATIX

El software de simulación Tecnomatix permite la simulación, visualización, análisis y optimización de sistemas de producción gracias a la Simulación en Planta. No obstante, para este proyecto dicha característica no proporciona ninguna ventaja, ya que no se trata de un proceso en cadena, sino de un solo robot que llevará a cabo el ensamblaje de un producto. Las características de Tecnomatix son las siguientes [14]:

- Cuenta con una Interfaz Gráfica que hace que la programación sea sencilla e intuitiva, haciendo así que cualquiera pueda utilizarla sin mayores complicaciones. Dichos programas se pueden convertir después en scripts

para así poder pasarlos al robot mediante una memoria USB, aunque no permite la conexión entre el robot y el software.

- No posee una biblioteca con los diferentes modelos de robots para poder utilizarlos. Es necesario que el fabricante del robot que se vaya a utilizar facilite el archivo JT (archivo CAD) del robot para poder utilizarlo en el simulador. Dicho archivo JT es un formato de modelo 3D desarrollado por Siemens PLM Software que fue diseñado como un formato de almacenamiento abierto.

- Los objetos deben de importarse en formato 3D para poder crear el entorno de trabajo.

- Gracias a las diferentes gráficas que se pueden crear, se podrá hacer un análisis detallado del proceso para así poder detectar cuellos de botella o poder optimizar el uso de energía.

- Se puede programar mediante scripts directamente, aunque se debe hacer con el lenguaje desarrollado por el fabricante del robot que se utilice o con el lenguaje universal para robots.

- Se puede simular el flujo de material que se tendrá en la planta real, haciendo que sea más visual el proceso que se está simulando.

ANÁLISIS COMPARATIVO

En la Tabla 2 se pueden observar los requisitos obligatorios y no obligatorios que el software de simulación debe cumplir. En el caso de RoboDK se puede comprobar que cumple todos los requisitos a excepción del flujo de material. No obstante, se trata de un requisito no obligatorio ya que el proceso puede ser simulado de la misma manera, aunque no será igual de visual que si permitiera el flujo de material.

En cambio, Tecnomatix no cumple uno de los requisitos obligatorios, que es la biblioteca de robots. Eso hace que el proceso se ralentice ya que habrá que pedir el archivo JT del robot a su fabricante y éstos pueden tardar un tiempo en contestar.

Además, tampoco cumple dos de los requisitos no obligatorios, que son utilizar lenguajes de programación, como puede ser Phyton, y poder crear una conexión entre el software de simulación y el robot. Esta última característica facilita en gran medida el envío del script al robot, evitando depender de una memoria USB que siempre puede crear más problemas.

Por lo tanto, se utilizará RoboDK dado que presenta las características básicas que un software de simulación debe de tener para este proyecto.

Tabla 2.- Tabla comparativa del software de simulación

Software de simulación	RoboDK	Tecnomatix
Requisitos obligatorios		
Interfaz gráfica intuitiva	Sí	Sí
Entorno de trabajo	Sí	Sí
Obtener scripts de los programas creados	Sí	Sí
Biblioteca de robots	Sí	No
Requisitos no obligatorios		
Lenguajes de programación (Python, C++ ...)	Sí	No
Conexión con el robot	Sí	No
Flujo de material	No	Sí

1.6 Análisis de riesgos

En este apartado se analizan los posibles riesgos que pueden surgir durante el desarrollo del proyecto, identificando la probabilidad de ocurrencia de dichos riesgos y los daños que pueden causar en el proyecto, con objeto de establecer un plan de actuación para poder prevenirlos.

1.6.1 Definición de los riesgos

Los posibles riesgos identificados son los siguientes:

1. **Problemas en el simulador:** problemas para desarrollar el gemelo virtual con el software de simulación. Dicho gemelo debe simular el proceso que en un futuro realizará el robot real.
2. **Problemas con el montaje del entorno:** una vez simulado el proceso es necesario ejecutar el proceso con el robot real. Para ello debe de estar montado el entorno real, con las diferentes piezas que se utilizarán en el ensamblaje del producto.
3. **Problemas con la pinza:** problemas de puesta en marcha del robot y en la simulación debido a que la herramienta no es del mismo fabricante que el sistema robótico.
4. **Problemas en la conexión entre el software de simulación y el PLC:** problemas al crear la conexión entre el software de simulación y el PLC, ya que es una conexión que no se ha realizado previamente por el personal del laboratorio.
5. **Problemas en la conexión entre el software de simulación y el robot:** problemas al crear la conexión entre el software de simulación y el sistema robótico. Normalmente la conexión del sistema robótico se crea con el PLC, pero en este proyecto se ha decidido crearla con el software de simulación dado que RoboDK permite realizar dicha conexión.

1.6.2 Probabilidad e impacto de los riesgos

En este apartado se asocia a cada riesgo identificado una probabilidad de ocurrencia y el correspondiente impacto sobre el proyecto. En la Tabla 3 se presenta la matriz probabilidad-impacto de los riesgos mencionados.

1. **Problemas en el simulador:** simular el proceso en el software de simulación es indispensable para empezar este proyecto, por lo que tener problemas con dicho software tendrá un alto impacto. No obstante, la probabilidad de que esto ocurra es baja, dado que el simulador cuenta con una guía y varios ejemplos para aprender a utilizarlo.
2. **Problemas con el montaje del entorno:** las diferentes piezas que se utilizarán en el ensamblado del producto tienen que ser impresas en una impresora 3D. Dicha función recae en un técnico de la Escuela encargado también de imprimir piezas para otros proyectos, por lo que la probabilidad de que haya algún retraso en dicha impresión será media. Sin embargo, eso no impide seguir con el proyecto, por lo que el impacto será también medio.
3. **Problemas con la pinza:** se trata de una pinza neumática que necesitará tubos para que el aire comprimido accione la pinza cuando sea necesario. Dichos tubos junto a los cables de los sensores que indican cuando la pinza está abierta y cuando cerrada irán atados alrededor del robot. Al tratarse de un brazo robótico de 6 grados de libertad, estos tubos y cables pueden enroscarse con los movimientos del robot haciendo que se active la parada de emergencia. La probabilidad de que esto ocurra será baja, dado que los tubos y cables irán bien sujetos al robot, pero si se enroscan en algún movimiento su impacto será medio.
4. **Problemas en la conexión entre el software de simulación y el PLC:** la conexión entre el simulador y el PLC se hará mediante OPC UA, que es un protocolo desconocido hasta el momento por el personal del laboratorio, por ello la probabilidad de tener problemas en dicha conexión es media, y tiene un impacto alto al ser una conexión imprescindible para el proyecto.

5. **Problemas en la conexión entre el software de simulación y el robot:** al ser un robot nuevo y un software de simulación nunca utilizado en el laboratorio, habrá una alta probabilidad de tener problemas al crear la conexión entre ambos. Dicha conexión depende de la red de la Escuela, por lo que, si ésta se cae, el impacto que tendrá en el proyecto será alto, siendo algo que no se puede controlar.

Tabla 3.- Tabla de influencia y probabilidad de los riesgos

		Probabilidad		
		Baja	Media	Alta
Impacto	Baja			
	Media	3	2	
	Alta	1	4	5

1.6.3 Prevención de los riesgos

A continuación, se describen las soluciones propuestas para afrontar los posibles riesgos mencionados, teniendo en cuenta el análisis realizado en la matriz probabilidad-impacto.

1. **Problemas en el simulador:** para evitar problemas en el software de simulación, es conveniente primero mirar su guía básica acompañándolo de algún vídeo. Esto es necesario ya que es un software de simulación que no se había utilizado anteriormente en el laboratorio, por lo que esta formación básica de inicio será fundamental para después poder programar el proceso sin tener que detenerse demasiado.
2. **Problemas con el montaje del entorno:** será necesario avisar con la suficiente antelación al técnico encargado de imprimir las piezas 3D para que estén listas a la hora de hacer pruebas con el robot real, una vez terminada la simulación.

3. **Problemas con la pinza:** para evitar que los tubos y cables que deben ir alrededor del robot se enrosquen al hacer diferentes movimientos, a la hora de colocarlos se irá moviendo el robot articulación por articulación y calculando las dimensiones.
4. **Problemas en la conexión entre el software de simulación y el PLC:** es conveniente encontrar la documentación adecuada que facilite dicha conexión (pdf explicativos, vídeo tutoriales...) para poder ahorrar tiempo a la hora de hacerla.
5. **Problemas en la conexión entre el software de simulación y el robot:** será necesario que el robot y el software de simulación puedan verse mutuamente, y para ello deberán estar en la misma subred. Esta condición será la primera a comprobar antes de crear la conexión, ya que si el ordenador no está conectado a la misma subred que tiene el sistema robótico será imposible que éstos se conecten.

1.7 Desarrollo de la solución

Este apartado recoge una detallada explicación de las diferentes fases que se han seguido para el desarrollo del proyecto, el cual comienza con la puesta en marcha del robot UR3e recientemente adquirido. Dado que se utiliza por primera vez para este proyecto, es necesario tanto su montaje como su programación.

A continuación, se utiliza el software de simulación RoboDK para simular el proceso de ensamblado que debe desempeñar el robot real. El número de productos a ensamblar puede variar entre 1 y 6 unidades en cada petición.

Dado que el operario es el encargado de indicar el número de unidades, es necesario establecer una comunicación OPC UA entre el software de simulación y el PLC para que este último le aporte dicho número. Asimismo, una vez finalizado el proceso, el operario debe recibir información de cuándo empieza y acaba el proceso completo, así como de cuándo empieza y acaba el ensamblado de cada unidad.

Además, también es preciso establecer la conexión entre el software de simulación y el sistema robótico, con objeto de ejecutar el programa generado con RoboDK en el robot. La ejecución de dicho programa se puede realizar tanto desde el software de simulación como desde el controlador del robot.

Por último, se presentan las pruebas realizadas, los problemas encontrados en dichas pruebas y las soluciones propuestas.

1.7.1 Puesta en marcha del robot

El proceso de puesta en marcha se explicará con más detalle en los siguientes subapartados. En primer lugar, se abordará el montaje del robot y la herramienta utilizada, ya que es una parte importante en la puesta en marcha del robot. A continuación, se describirá la interfaz gráfica de usuario a través de la pantalla táctil del TeachPendant, denominada PolyScope, y las utilidades que éste ofrece, ya que puede servir de gran ayuda a la hora de hacer pequeñas pruebas o de programar

pequeños procesos siendo necesario su uso a la hora de poner en marcha el robot. Por último, se presentará una pequeña introducción a la programación mediante scripts.

Los robots colaborativos de Universal Robots pueden ser programados mediante el PolyScope o mediante scripts. No obstante, para este proyecto se utilizará la programación mediante scripts, ya que se hará mediante el software de simulación.

1.7.1.1 Montaje del robot y herramienta

El montaje del sistema robótico es relativamente sencillo ya que el brazo viene en una única pieza y solo hay que sujetarlo a la mesa adquirida previamente.

Una vez el sistema robótico está bien sujetado a la mesa, es necesario colocarle una herramienta adecuada para el proyecto. En este caso, se trata del montaje de un producto que contiene diferentes piezas, por lo que la herramienta más adecuada será una que contenga dos pinzas: una que pueda coger piezas como el palet o el eje, y otra para que pueda coger tanto los rodamientos como la tapa, haciendo así posible que pueda coger todos los tipos de piezas que se utilizarán.

Una vez puesta la herramienta, será necesario añadir los tubos para el aire comprimido, ya que se trata de dos pinzas neumáticas. Además, cada pinza tendrá dos sensores, uno para cada una de las posibles posiciones, por lo que habrá que añadir los 4 cables que tienen los sensores. En la Figura 5 se puede observar la apariencia final del brazo robótico, con la herramienta, y los tubos y cables necesarios alrededor del robot.

Las dos pinzas neumáticas utilizan aire comprimido tanto para abrirlas como para cerrarlas. Funcionan de manera opuesta, teniendo una el comportamiento de normalmente abierta y la otra de normalmente cerrada, por lo que sus sensores trabajarán de forma complementaria.

Figura 5.- Robot y herramienta en entorno real

- **Pinza normalmente abierta** (Figura 6)

Se trata de una pinza que está abierta en su posición natural, por lo que se cerrará cuando la salida digital que tiene asignada pase de su valor inicial 0 al valor 1, esto es, cuando la salida se active. Esta pinza será la que se utilizará para poder mover de sitio el palet y los ejes, dado que la pinza agarrará dichos objetos por la parte exterior de éstos.

Figura 6.- Pinza neumática normalmente abierta [15]

- **Pinza normalmente cerrada** (Figura 7)

La posición natural de esta pinza será cuando está cerrada, y dicha pinza se abrirá cuando se active la salida digital que tiene asignada para que la pinza pueda agarrar piezas como los rodamientos y la tapa. En este caso, se trata de las piezas que tienen que ser agarradas por su interior, ya que la otra pinza no se abre lo suficiente para poder agarrarlas por su exterior.

Figura 7.- Pinza neumática normalmente cerrada [16]

1.7.1.2 Polyscope

PolyScope [18] es una Tablet de 12 pulgadas (Figura 8) utilizada para realizar la programación del robot. En ella se encuentra la interfaz gráfica de usuario desarrollada por Universal Robots que permite manejar el brazo robótico y la caja de control, ejecutar programas del robot y crear nuevos programas fácilmente.

Figura 8.- PolyScope [17]

A continuación, se presentarán las pantallas indispensables del PolyScope, las que será necesario conocer antes de empezar a utilizarlo, ya que son las básicas a la hora de querer ejecutar o programar un proceso.

- **Interfaz de usuario**

La Figura 9 muestra la pantalla de bienvenida del PolyScope, en la que se puede observar que hay diferentes opciones: ejecutar un programa ya existente, programar el robot con un nuevo programa o haciendo cambios en un programa existente, hacer diferentes configuraciones en él o apagarlo.

Figura 9.-Interfaz de usuario del PolyScope [18]

- **Pantalla de inicialización**

Mediante esta pantalla se podrá inicializar fácilmente el brazo robótico, simplemente dando al botón de "iniciar" lo que permitirá que se liberen los frenos y finalmente se encienda el robot. Se puede observar un LED verde en la Figura 10, lo que indica que el robot está listo para ser utilizado. Ese LED puede tener también los colores amarillo o rojo, que indican respectivamente que el robot está encendido, pero no listo para ser utilizado y que el robot está apagado.

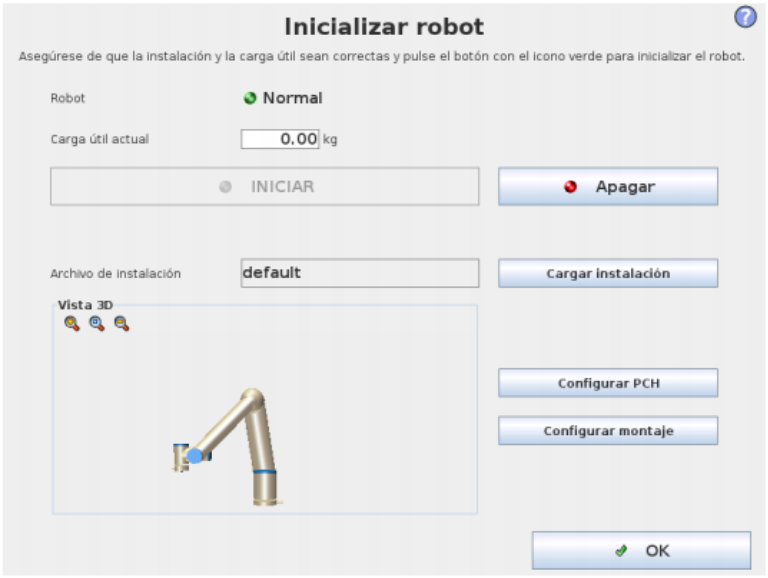

Figura 10.-Pantalla de inicialización del PolyScope [18]

- **Nuevo programa**

La interfaz de inicio para un nuevo programa (Figura 11) consta de dos opciones: se puede cargar un programa guardado anteriormente en un dispositivo externo como puede ser una memoria USB, o iniciar un programa utilizando una plantilla. Estas plantillas contienen programas sencillos, como puede ser mover un objeto de sitio, que se deben personalizar para el proyecto.

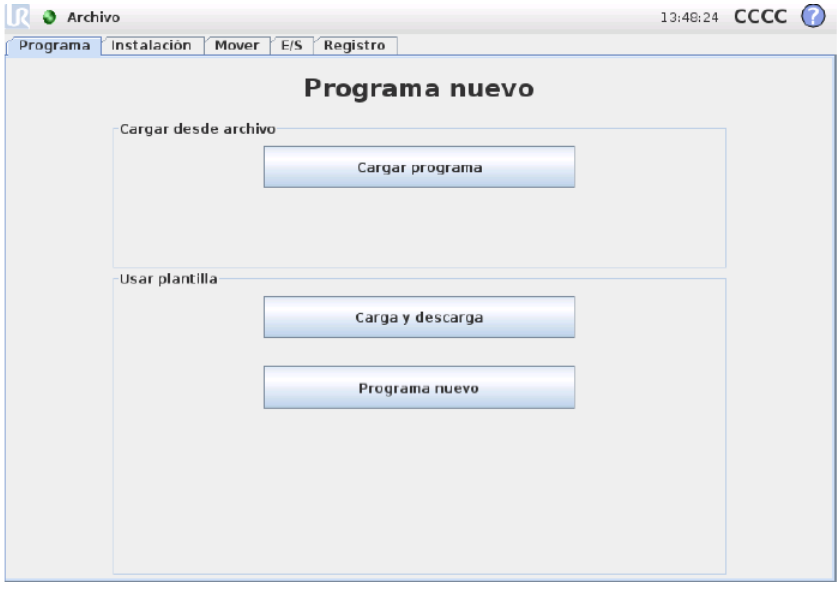

Figura 11.-Pantalla de nuevo programa del PolyScope [18]

- **Ficha programa**

En la Figura 12 se puede observar la ventana donde irá apareciendo el programa del robot. Concretamente, irá apareciendo en la parte blanca de la izquierda, donde se podrá observar un diagrama de árbol con los diferentes comandos del programa.

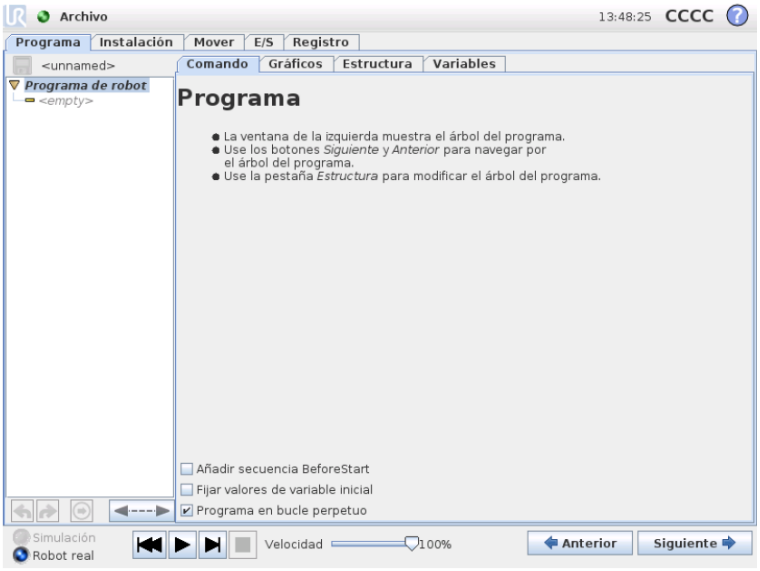

Figura 12.- Pantalla programa del PolyScope [18]

- **Comandos de programación**

Los comandos con los que programar el robot pueden ser de nivel básico o avanzado. Como el objetivo de este proyecto no será programar utilizando el TeachPendant, sino que solo se utilizará para hacer unas pequeñas pruebas en la puesta en marcha del robot, no se ve necesaria la explicación de todos los comandos.

En el nivel básico, se pueden utilizar comandos como movimiento, para que el brazo robótico se mueva entre dos puntos, o el comando esperar, para que éste espere antes de ejecutar el siguiente comando.

Por otra parte, en el nivel avanzado, se puede hacer uso de funciones como el bucle, para que un programa se repita siempre o se repita mientras se cumpla una determinada condición. También se podrá utilizar la función *if...else*, para que el programa ejecute una parte del código si se cumple la condición dentro del *if*.

Para terminar con los ejemplos del nivel avanzado, cabe comentar la función *palé*. Mediante esta función, se facilita mucho la operación de paletizado, que puede ser muy útil para muchos procesos. Solo es necesario fijar los puntos de paso y acercamiento, los de entrada y salida del producto, y el proceso será llevado a cabo por el brazo robótico.

1.7.1.3 Programación mediante scripts

Otra forma de programar el robot es mediante scripts [19]. Estos scripts pueden ser escritos con los lenguajes de programación C++, Java o Python entre otros, y pueden ser enviados al robot mediante un dispositivo externo, o bien añadirlos en el PolyScope. En el caso de usar dispositivos externos, pueden conectarse en los puertos 30001, 30002 o 30003 de los que dispone el robot, aunque si se opta por crear una conexión servidor-cliente es aconsejable no utilizar los puertos anteriormente nombrados.

La programación mediante scripts ofrece varias ventajas si se compara con la programación mediante comandos en el PolyScope.

- El programa puede ser editado mientras el robot realiza sus tareas.
- Es posible introducir y crear nuevas funciones ampliando así las capacidades del robot.
- Permite controlar varios robots a la vez, utilizando un solo ordenador.

1.7.2 Simulación del proceso

El proceso que se llevará a cabo para el ensamblado del producto está compuesto por las siguientes 6 fases:

- Fase 1: Se mueve el palet con las 6 bases al punto de montaje.
- Fase 2: Se coloca un rodamiento en un agujero del que dispone la base.
- Fase 3: Se coloca el eje en el interior del rodamiento anteriormente integrado.
- Fase 4: Se añade otro rodamiento encima del eje que se ha colocado en la fase anterior.

- Fase 5: Se añade la tapa quedado así finalizado el producto.
- Fase 6: Para finalizar el proceso, se mueve el palet con los productos terminados al punto de salida.

Las Fases 2-5 de repiten para cada una de las unidades que se ha solicitado ensamblar. En la figura 13 se pueden observar las piezas que se utilizarán en el ensamblado del producto.

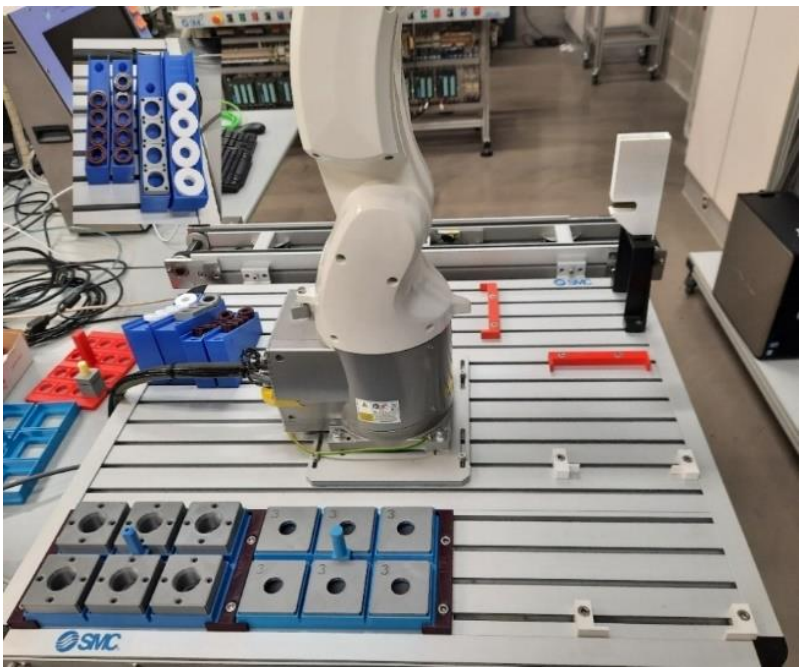

Figura 13.-Piezas para montaje en entorno real

Estas 6 fases son las que tienen que ser simuladas mediante el software de simulación RoboDK [20]. Lo primero será cargar el modelo del robot que se utilizará en el proyecto (UR3e) disponible en la biblioteca de dicho software, e importar los diferentes objetos 3D creados anteriormente para otro proyecto que se utilizarán para el ensamblaje del producto. También se añadirá como objeto 3D la herramienta que se utilizará en el proyecto, y se fijará como TCP *(Punto Central de la Herramienta)* del robot. A continuación, se podrá observar en la pantalla del ordenador el mismo escenario del que dispone el laboratorio, tal y como ilustra la Figura 14.

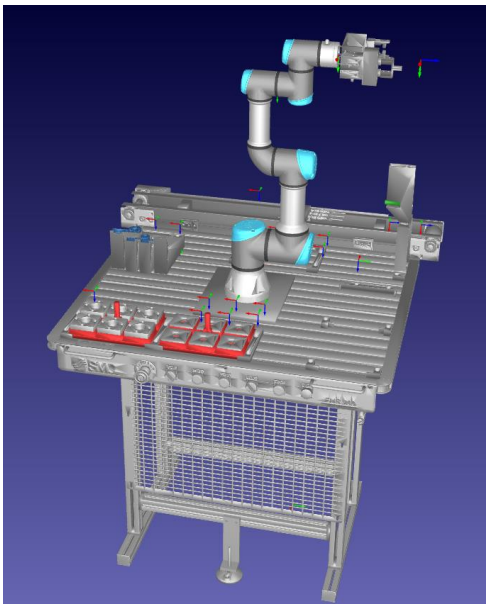

Figura 14.-Entorno creado en el software de simulación

Una vez la mesa está montada, se procederá a la simulación del proceso, que consistirá en simular las 6 fases explicadas. Para ello, será necesario fijar primero los diferentes puntos de paso que tendrá que seguir el brazo robótico durante el montaje del producto, para posteriormente crear los movimientos entre dichos puntos.

En la Figura 15 se pueden observar las diferentes secciones disponibles en el software RoboDK. En la parte de la izquierda, se puede observar el árbol de estación, donde irán apareciendo todos los cambios que se vayan haciendo en el proyecto.

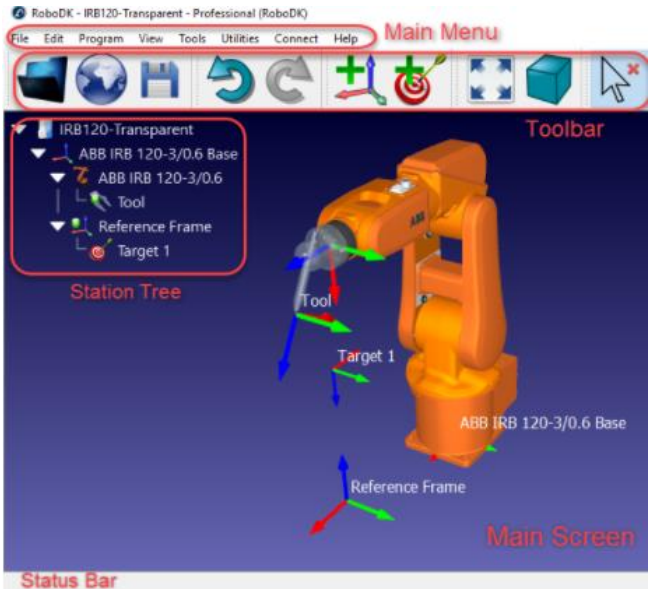

Figura 15.-Interfaz gráfica de RoboDK [20]

Gracias a la barra de herramientas se podrán fijar los puntos de paso haciendo uso del icono de la Figura 16:

Figura 16.- Icono de nuevo objetivo [20]

Una vez fijados todos los puntos de paso necesarios, se crearán nuevos programas con cada una de las fases. Las fases 1 y 6 necesitarán un único programa por fase. En cambio, las fases 2, 3, 4 y 5 tendrán que ser repetidas 6 veces cada una, ya que los puntos de paso serán diferentes para cada producto que se esté montando en el palet. Los nuevos programas se añadirán mediante el icono que se puede observar en la Figura 17.

Figura 17.- Icono de nuevo programa [20]

Los programas se basarán en los movimientos que servirán de unión entre los puntos de paso. Esos movimientos podrán ser axiales o lineales, dependiendo si el sistema robótico tiene que moverse en línea recta o al contrario es necesario que el movimiento sea axial. En la Figura 18 se pueden observar los dos iconos que se deberán utilizar para el movimiento axial y para el lineal, respectivamente.

Figura 18.- Icono de nuevo movimiento [20]

El software de simulación RoboDK no cuenta con la opción de poder abrir y cerrar la herramienta para poder coger los diferentes objetos, pero sí que permite configurar entradas y salidas digitales, que servirán para que al exportar la simulación al robot, se

activen y desactiven las salidas que se deseen para que la herramienta pueda abrirse y cerrarse. La Figura 19 presenta cómo pueden activarse dichas salidas digitales:

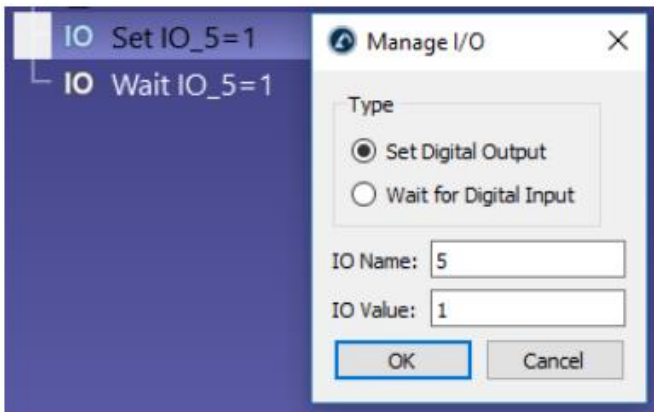

Figura 19.- Pantalla de configuración de entradas y salidas digitales [20]

Como se ha comentado en la puesta en marcha del robot, la herramienta que se utilizará para este proyecto tiene dos pinzas, cada una de ella con una salida digital, la 0 y la 1. Dependiendo de que pinza sea necesaria para cada fase, se activará una salida u otra.

Una vez simuladas todas las fases con sus respectivas repeticiones, se crearán nuevos programas que tengan como subprogramas las fases necesarias. Estos nuevos programas dependerán de la cantidad de productos que sean necesarios montar, por lo que serán 6 programas diferentes llamados *1Productos, 2Productos...6Productos* que consistirán en el montaje de las diferentes cantidades posibles. Para ello, será preciso hacer llamadas a los subprogramas necesarios en cada programa, tal y como se dispone en la Figura 20.

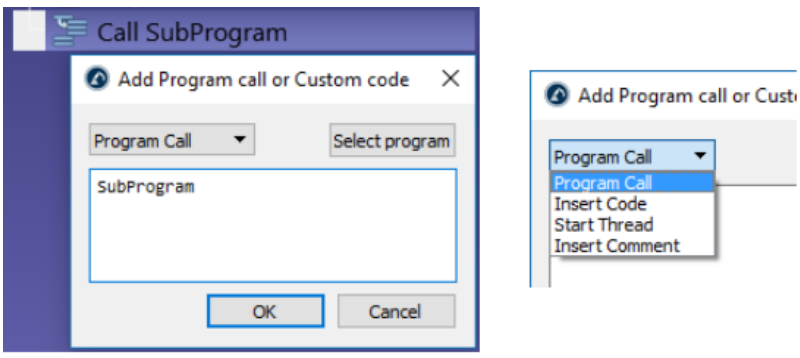

Figura 20.- Pantalla de llamada a subprograma [20]

Cuando todos los programas con el diferente número de productos estén listos, se procederá a hacer el programa principal, que tendrá que decidir qué programa de los 6 hechos anteriormente *(1Productos, 2Productos...6Productos)* debe ejecutar en función del número de productos que pida hacer el operario. Para ello, es necesario escribir un script en Python en el que mediante un comando *if...else* sea capaz de ejecutar el programa con el número de piezas que se deben montar en cada momento. Dicho programa principal codificado con el lenguaje de programación Python se podrá escribir utilizando el icono de la Figura 21.

Figura 21.- Icono de nuevo programa Python [20]

A continuación, se presentará el código escrito para que el sistema robótico pueda hacer el ensamblaje del producto. Para ello, será conveniente utilizar la API *(Application Programming Interfaces)* de RoboDK, ya que se introducirán diferentes subprogramas creados anteriormente que son los correspondientes al número de productos que se deben ensamblar en cada caso. La API de RoboDK está formada por un conjunto de rutinas y comandos que dicho software proporciona mediante un único lenguaje de programación universal.

Si se compara con la programación de robots específica del proveedor, se puede comprobar que la API de RoboDK ofrece la gran ventaja de poder simular y programar un brazo robótico usando un lenguaje de programación universal, como puede ser el caso de los lenguajes de alto nivel Python, C++ o Matlab. Esto permite que se pueda crear un programa con el lenguaje de programación Python, pero que dicho programa pueda hacer llamadas a otros subprogramas anteriormente programados utilizando la interfaz gráfica de usuario de RoboDK, como se puede observar en la Figura 22.

*ProgramaPincipal.py - C:\Users\aayerra003\AppData\Local\Temp\ProgramaPincipal.py (3.7....

File Edit Format Run Options Window Help

```
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox

RDK = Robolink()

#get the robot item
robot = RDK.Item('UR3e')

#ProgramaPincipal
NumeroProductos = mbox ('Introduze el numero de productos', entry=True)
NumeroProductos = int(NumeroProductos)

if NumeroProductos == 1:
    prog1 = RDK.Item ('1Productos', ITEM_TYPE_PROGRAM)
    prog1.RunProgram()
elif NumeroProductos == 2:
    prog2 = RDK.Item ('2Productos', ITEM_TYPE_PROGRAM)
    prog2.RunProgram()
elif NumeroProductos == 3:
    prog3 = RDK.Item ('3Productos', ITEM_TYPE_PROGRAM)
    prog3.RunProgram()
elif NumeroProductos == 4:
    prog4 = RDK.Item ('4Productos', ITEM_TYPE_PROGRAM)
    prog4.RunProgram()
elif NumeroProductos == 5:
    prog5 = RDK.Item ('5Productos', ITEM_TYPE_PROGRAM)
    prog5.RunProgram()
else:
    prog6 = RDK.Item ('6Productos', ITEM_TYPE_PROGRAM)
    prog6.RunProgram()
```

Figura 22.- Código del programa principal

En las dos primeras líneas del código se puede observar cómo se ha importado la API de RoboDK. Una vez hecho eso, cada vez que se quiera utilizar algo programado antes o utilizado anteriormente será necesario utilizar la extensión "RDK.", que ha sido el nombre utilizado para nombrarlo en la línea 3.

El programa pedirá al operario el número de productos que deben ser montados, y dependiendo de ese número, se ejecutará uno de los 6 programas programados anteriormente accesibles de una forma sencilla gracias a la API.

La elección del programa que se debe ejecutar en cada momento se ha programado mediante un *if...elif...else*, dependiendo simplemente del valor guardado en cada momento en la variable "NumeroProductos".

Cuando se cumpla la primera condición del *if*, esto es, cuando se deba montar un solo producto, se ejecutará el programa anteriormente programado con el nombre *1Productos*. Así será consecutivamente con las siguientes 4 condiciones escritas cada una de ellas en un *elif*. Para terminar, cuando el número de productos a ensamblar sea 6, no se cumplirá ninguna de las condiciones anteriores, por lo que se ejecutará el programa que aparece después del *else*.

El número de productos que se deberá montar en cada momento también será enviado mediante el PLC, una vez que la conexión entre el PLC y el software de simulación esté creada.

1.7.3 Código PLC

El código utilizado en el PLC reutiliza parte del código desarrollado para otro proyecto llevado a cabo en el laboratorio, en el que se realiza el ensamblado del mismo producto, pero usando el robot KUKA.

Por esa razón, este apartado se centrará en aquellos aspectos del código que han sido añadidos y/o modificados para el desarrollo de la solución. Concretamente, el objetivo del código es leer un fichero .txt donde se define el número de productos que el robot debe ensamblar en cada proceso y procesar dicha información para que se pueda enviar al sistema robótico. Además, también deberá crear una función que sea capaz de convertir el modo en el que leerá la hora de comienzo y final tanto de cada producto como del total de unidades a ensamblar (palet), de forma que sea posible calcular la durabilidad.

1.7.4 Conexión software de simulación-PLC

Como ya se ha comentado anteriormente, el objetivo de este proyecto se basa en que el brazo robótico UR3e monte un número de productos que será especificado por un operario, y durante el proceso se mande la información de cuándo se empieza y acaba el proceso y cuándo se empieza y acaba cada pieza, para después poder calcular la duración de cada proceso.

Para ello, es indispensable crear una conexión entre el software de simulación y el PLC, que en este caso será llevado a cabo mediante OPC UA [22].

OPC UA permite intercambiar datos e información en dispositivos dentro de máquinas, entre máquinas, y de máquinas y sistemas de forma gratuita (Figura 23). Esta arquitectura hace que empresas con pequeños presupuestos sean capaces de tener acceso a datos generados anteriormente por otras tecnologías, como puede ser la visión artificial.

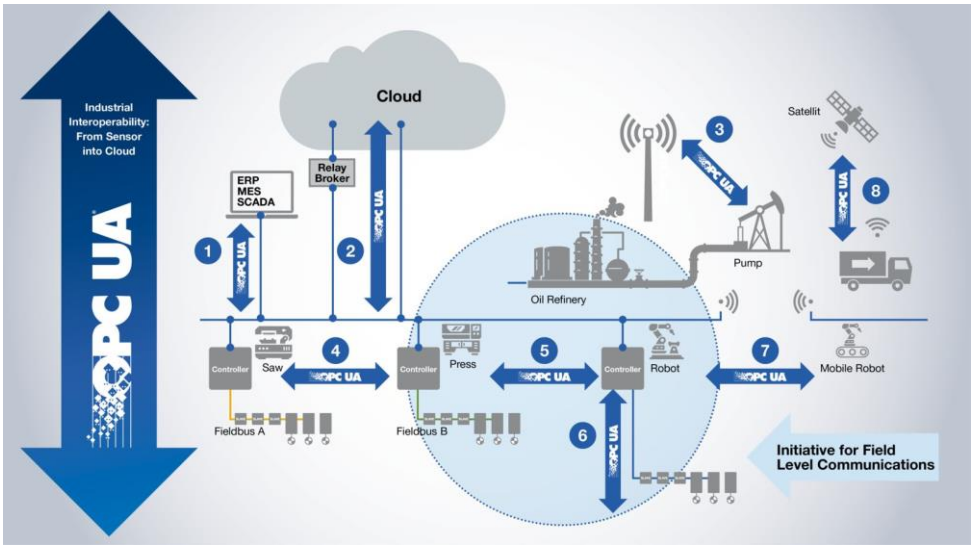

Figura 23.-Esquema OPC UA [21]

Por supuesto, este no es el único método de comunicación existente en la automatización industrial, pero la mayoría de ellos son de acceso limitado ya que son específicos del vendedor. Ejemplo de ello pueden ser PROFINET, usado en los PLCs de Siemens, o iQSS para PLCs de Mitsubishi. Sin embargo, al ser OPC UA un protocolo de comunicación abierto, puede ser utilizado por cualquier sistema operativo, incluyendo Linux, Windows, Android o iOS. Esta es una de las grandes ventajas que presenta OPC UA respecto al estándar original OPC, ya que este último solo funcionaba en dispositivos con Windows de Microsoft.

OPC UA fue creado para poder conectar diferentes máquinas, dispositivos y sistemas dentro de un establecimiento de fabricación moderno y así aprovechar los beneficios de la tecnología moderna para una instalación inteligente.

Esta conexión está basada en el principio cliente-servidor y en este caso, tanto el software de simulación RoboDK como el PLC pueden actuar de las dos maneras. Por lo tanto, es conveniente entender primero las diferencias entre un cliente y un servidor para así poder decidir cuál es el rol más adecuado en cada caso.

En el mundo de las tecnologías de la información, un servidor es un ordenador que contiene los archivos, y los clientes son los dispositivos que se conectan al servidor con el fin de utilizar los datos en aplicaciones. En dicho entorno, siempre hay un servidor y muchos clientes (Figura 24).

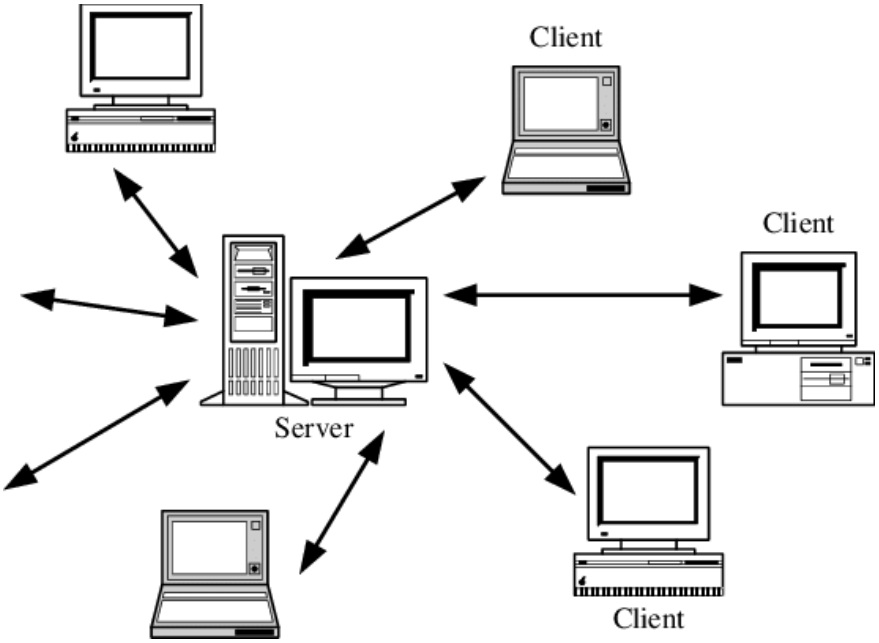

Figura 24.- Esquema cliente y servidor en IT [23]

En cambio, en el mundo de la automatización es totalmente diferente, ya que los servidores son dispositivos de punto final. En redes como EtherNet/IP y PROFINET IO, hay un cliente y muchos servidores, justo lo contrario a lo que pasa en el mundo de IT. El controlador PROFINET IO o el escáner EtherNet / IP es en este caso el dispositivo cliente. Un cliente atiende una sección de un sistema de fabricación y se conecta a varios dispositivos finales, los servidores.

En todas las aplicaciones de fabricación con comunicación OPC UA, no importa cómo se llame el dispositivo (controlador, escáner o cliente), el dispositivo configurado por

el usuario para abrir conexiones con uno o más dispositivos finales, enviar salidas a dichos dispositivos finales y recibir entradas de esos dispositivos es el dispositivo cliente.

Al convertir un dispositivo en un servidor OPC UA, los dispositivos convertidos en cliente OPC UA son capaces de acceder a cualquier dato del mundo físico disponible a través del espacio de direcciones del servidor. Hay al menos dos formas en que los clientes pueden lograr esto. Por una parte, leyendo y escribiendo directamente los atributos del espacio de direcciones. Por otro lado, utilizando los elementos supervisados y los servicios de publicador o bien de suscriptor para que esos atributos se publiquen en un momento determinado.

En cambio, al convertir un dispositivo en un cliente OPC UA, puede tomar cualquier valor del mundo físico que tenga y escribirlo en algún otro servidor OPC UA siempre que lleguen nuevos datos.

Pero a diferencia de lo que pasa en el mundo de IT, en el mundo de la automatización se permite que un dispositivo se comporte tanto como cliente como servidor, haciendo que su implementación sea muy flexible. Gracias a ello, los clientes podrán recibir los datos mediante los dos mecanismos anteriormente mencionados: bien leyendo datos de atributos de otros servidores o bien escribiendo datos de atributos en otros servidores [24].

En el caso de este proyecto, el PLC será el primero en mandar información al software de simulación, pero una vez finalizado el proceso, será justo al revés. El software de simulación mandará información para que el PLC pueda recibirla y guardarla. Por lo tanto, ambos pueden tener la función de cliente y de servidor indistintamente.

Al mirar las distintas opciones, configurar el software de simulación como cliente tiene más complicaciones que hacerlo en el PLC, por lo que se ha optado en configurar el software RoboDK como servidor y hacer lo propio con el PLC, pero configurándolo como cliente (Figura 25).

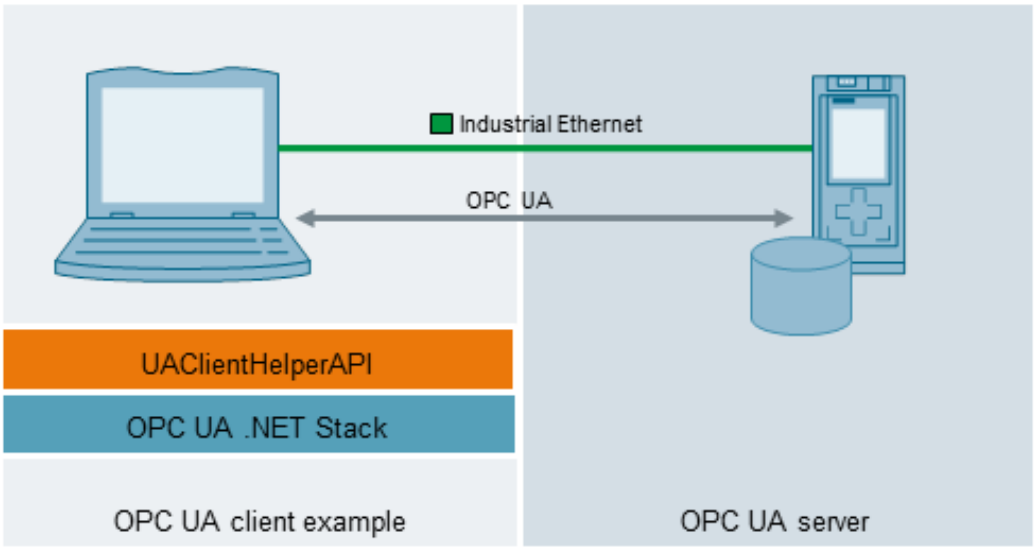

Figura 25.-Esquema conexión OPC UA en PLC [25]

1.7.4.1 Configuración del software de simulación como servidor OPC UA

Para activar la comunicación OPC UA en RoboDK, es suficiente con seguir 3 pasos:

1. Seleccionar *Tools-Plug-Ins.*
2. Seleccionar *Load Plug-In.*
3. Seleccionar *OPC-UA.*

Una vez hecho esto, aparecerá un menú adicional con las opciones de OPC UA, como se puede apreciar en la Figura 26.

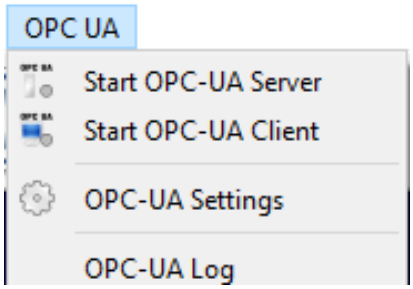

Figura 26.- Pantalla OPC UA en RoboDK

En dicho menú, se pueden observar las opciones para configurar RoboDK bien como servidor o bien como cliente, además de los ajustes necesarios para conectarlo de las dos maneras mencionadas (Figura 27).

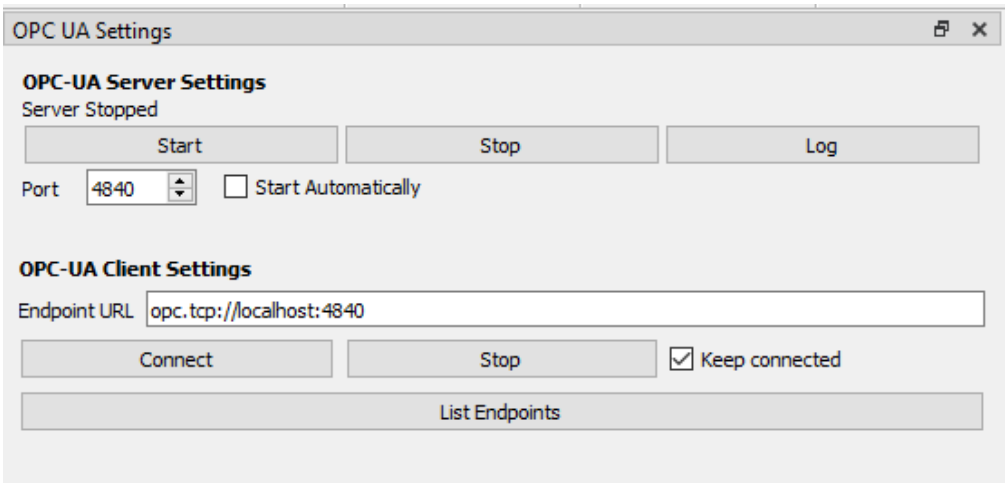

Figura 27.-Pantalla configuración OPC UA en RoboDK

En este proyecto, el software de simulación RoboDK será configurado como servidor, por lo que será preciso configurar las variables para que el cliente pueda enviar y recibir las que sean precisas. Estos parámetros son accesibles desde la estación creada en el software de simulación, mediante el uso de los nodos *StationParameter* y *StationValue*. Así, se debe establecer el parámetro de estación que se desee ver o modificar en el campo *StationParameter,* y ver o modificar su correspondiente valor usando *StationValue*. Éstos serán los parámetros que se deberán leer o mandar al cliente.

En este proyecto serán 3 los parámetros que habrá que configurar después en el cliente:

- *NumeroProducto*, para que el programa principal creado en el software de simulación pueda decidir qué programa debe ejecutar, en función de cuantos productos debe ensamblar.
- *ProcessStarted,* será una salida digital que se activará cuando el sistema robótico empiece a ejecutar el proceso, por lo tanto, la orden de activarla se dará en la fase 1, cuando se mueve el palet a la zona de montaje. Dicha salida digital se desactivará cuando la fase 6 termine, esto es, cuando el palet sea movido a la zona de salida del producto.

- *ItemStarted,* también se trata de una salida digital, que en este caso será activada cuando empiece el montaje de cada producto. De esta manera, se podrá saber el tiempo necesario para ensamblar un solo producto. En este caso, la salida digital se activará al empezar la fase 2 y se desactivará al terminar la fase 5, que son las fases de comienzo y final de cada producto. Al contrario de lo que pasaba con la variable *ProcessStarted*, la variable *ItemStarted* podrá ser activada varias veces en un mismo proceso, tantas como la cantidad de productos que se deben ensamblar.

1.7.4.2 Configuración del PLC como cliente OPC UA

Una vez configurado el software de simulación RoboDK como servidor OPC UA, será necesario configurar el PLC como cliente para después crear una conexión entre ambos.

Los clientes OPC UA no sólo se configuran, sino que se programan usando bloques de función del sistema, a diferencia de lo que pasa cuando se desea que el PLC trabaje como servidor.

Siemens da acceso a un ejemplo de bloque de usuario S7 para un cliente OPC UA disponible para Tia Portal tanto en la versión 15 como en la 16, aunque para este proyecto será suficiente con la versión 16, ya que es la versión instalada en el ordenador utilizado. El bloque de usuario *OpcUaClient* acelera la implementación y simplifica la programación del PLC como cliente.

En la Figura 28 se puede observar cómo se creará dicha conexión, partiendo de los bloques disponibles en la página de Siemens, configurando el bloque *OpcUaClient*, y consiguiendo así que el PLC se comporte como cliente y cree la conexión con el servidor RoboDK. La explicación detallada para configurar el PLC como cliente OPC UA se describe en el documento [26] disponible en la página web de Siemens junto al ejemplo disponible para Tia Portal.

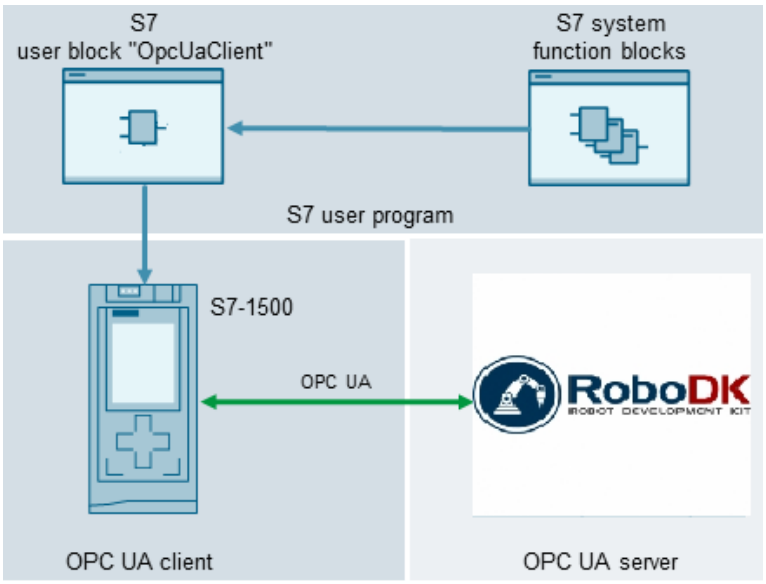

Figura 28.-Esquema conexión OPC UA entre PLC y RoboDK [26]

Al abrir en Tia Portal el ejemplo descargado de la página Siemens, se podrá observar que está preparado para poder hacer la configuración de servidor o de cliente, por lo que en este caso se eliminará la parte perteneciente al servidor, ya que la función de servidor la hará el software de simulación.

Lo primero que se debe configurar es la dirección IP de Ethernet que utilizará el cliente en la red 192.168.0.100, que es la red utilizada por Tia Portal cuando hace la función de cliente OPC UA.

A continuación, se procederá a la *configuración de dispositivo* de la CPU *(Unidad Central de Procesamiento)*. Se debe comprobar que el acceso de servidor está desactivado y el acceso del cliente activado; si no es así, se deberá activar cuanto antes, como se muestra en la Figura 29.

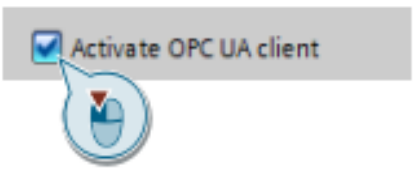

Figura 29.-Pantalla activación cliente OPC UA en Tia Portal [26]

La licencia requerida por OPC UA debe ser la misma que la licencia que se esté utilizando, y puede ser *small, medium o large*, siendo en este caso la *medium* la requerida por la CPU, por lo que se configurará para que en este proyecto Tia Portal trabaje con la licencia *medium.*

Una vez configurada la CPU, se procederá a la creación de una interfaz de cliente OPC UA para utilizar las funciones OPC UA de los bloques de función S7. La interfaz contiene toda la información relevante requerida por el cliente en el módulo "OpcUaClient". TIA Portal almacena esta información en dos bloques de datos generados automáticamente:

- *OpcUaClientInterface_Configuration*: esta base de datos contiene la información de conexión, los ID de nodo y los tipos de datos.
- *OpcUaClientInterface _Data*: los valores leídos o por escribir se almacenan en esta base de datos que también contiene una lista de estado para las variables individuales.

Para configurar la interfaz de cliente OPC UA, será necesario añadir los datos del servidor, tanto la dirección IP como el puerto. En el caso de este proyecto, como ambos softwares están en el mismo ordenador, será conveniente configurarlo con la dirección *localhost.*

El siguiente paso será configurar las listas de lectura y escritura. En la Figura 30 se muestra la pantalla disponible para configurar esas dos listas. En la parte derecha, se puede ver la interfaz del servidor, que es donde se creará la conexión con el software de simulación. Cuando la conexión es satisfactoriamente creada, se observará una carpeta con los datos del servidor que contendrá la información proveniente de RoboDK. Las variables que se deben leer o escribir en el PLC, habrá que ponerlas en la ventana de la izquierda, que es la que se utilizará para configurar las listas de lectura y escritura del cliente.

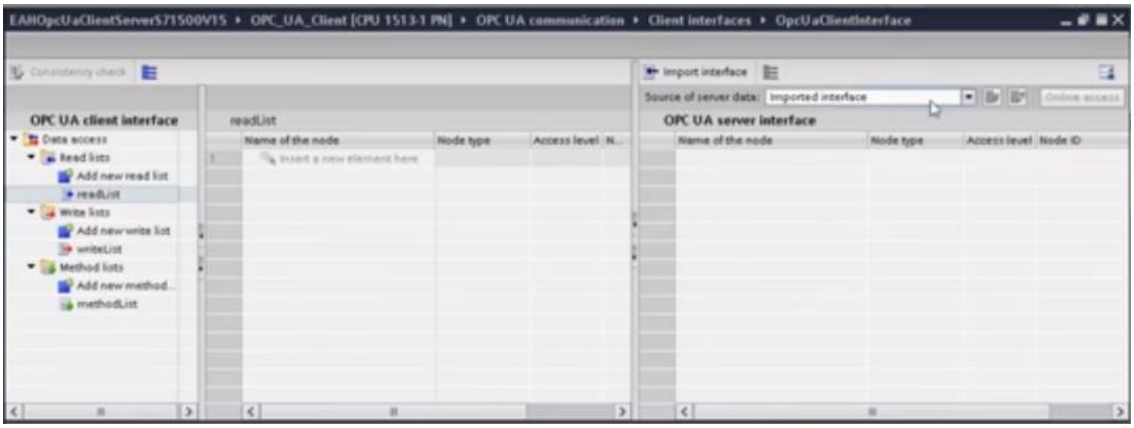

Figura 30.- Pantalla para configuración de las listas de lectura y escritura en Tia Portal [26]

En este proyecto, son 3 los parámetros que se deberán configurar.

- En el caso de la lista de escritura, el PLC deberá tener acceso al parámetro *NumeroProductos*, De esta manera, el servidor podrá recibir la petición de cuantos productos deben montarse y ejecutar el proceso correspondiente.
- En cambio, en la lista de lectura deberán configurarse 2 parámetros: *ItemStarted* y *ProcessStarted*. Dichos parámetros se modificarán en el software de simulación, y el PLC tendrá que leerlos para que el operario pueda recibir después la información de la durabilidad del montaje de cada producto y del proceso completo.

También cabe la posibilidad de configurar una lista de métodos, pero en este proyecto no será necesaria dado que los parámetros que deben ser transferidos entre los dos dispositivos son simplemente de escritura o lectura.

De esta manera quedará configurada la interfaz del cliente OPC UA, y se deberá compilar el proyecto para que Tia Portal cree los bloques *Configuración* y *Data* que serán los que contengan todos los parámetros configurados para el cliente OPC UA.

El bloque *OpcUaClient* estará listo para copiarlo en el proyecto de automatización creado para este trabajo. A continuación, será necesario conectar los parámetros de control y diagnóstico del bloque creando variables adecuadas para ello. Después se deberán conectar los parámetros de la interfaz de cliente del bloque *OpcUaClient*. Las

variables que se deben interconectar se pueden encontrar en los bloques de datos que se generan a través de la interfaz del cliente. Hecho esto, el proyecto estará preparado para cargarlo en la CPU mediante la opción de *Descargar a Dispositivo*.

Para la puesta en servicio, se hará uso de la tabla de observación que permite comprobar que funciona correctamente. En la Figura 31 se pueden observar cuatro secciones:

1. El área *Inputs* contiene las variables para controlar el bloque.
2. El área *Outputs* contiene las variables para el diagnóstico del bloque.
3. El área *Data Read* contiene las variables leídas por el servidor OPC UA.
4. El área *Data Write* contiene las variables que se escriben en el servidor OPC UA.

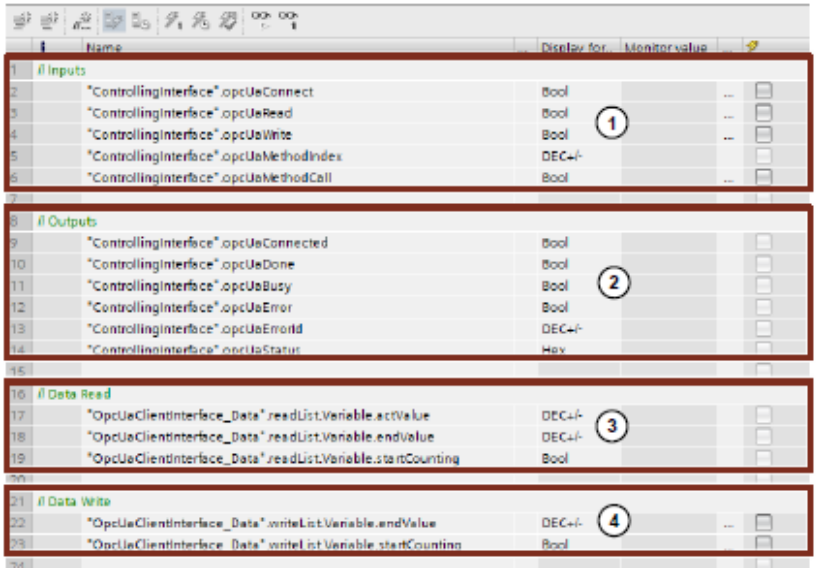

Figura 31.-Pantalla de la tabla de observación en Tia Portal [26]

Cuando en dicha tabla la variable que muestra que la conexión está creada se encuentra a *True*, se podrán leer las variables del servidor o escribirlas.

Para escribir variables en el servidor, será necesario cambiar el valor de la variable mediante la opción *Modify* y después cambiar el valor a *True* para que la acción pueda ser ejecutada. Para poder comprobar que la acción ha sido correctamente ejecutada, se observarán las variables *opcUaDone*, *opcUaError*, *opcUaErrorId* y *opcUaStatus* de la sección de *Outputs*.

En cambio, para leer parámetros del servidor, primero habrá que modificar el valor a *True*, y una vez comprobado que la acción se ha ejecutado correctamente, el valor enviado por el servidor aparecerá en la sección de *Data Read*.

1.7.5 Conexión software de simulación-sistema robótico

A la hora de ejecutar un programa creado en el software de simulación RoboDK en un sistema robótico, hay dos opciones:

- Ejecutar el programa desde el ordenador creando antes una conexión entre el software de simulación y el sistema robótico.
- Ejecutar el programa desde el controlador del robot. Para ello, será necesario convertir el programa creado en un archivo .URP, para que el sistema robótico pueda entenderlo y ejecutarlo. RoboDK cuenta con diferentes post-procesadores para los diferentes robots que hay en su librería. En el caso del robot UR3e, será necesario elegir el post-procesador Universal Robots URP, para que pueda hacer correctamente la conversión del archivo. Una vez esté listo el archivo que se ejecutará después en el robot, se copiará en una memoria USB y se pasará al TeachPendant para que el robot real lleve a cabo el proceso.

La segunda opción requiere crear otra conexión entre el PLC y el sistema robótico para que puedan transferir información ente ambos. Como anteriormente ya se ha creado una conexión entre el PLC y el software de simulación para esa transferencia de información, en este caso será suficiente con conectar el software de simulación al sistema robótico y de esta manera se ahorrará una nueva conexión con el PLC.

1.7.5.1 Ejecutar programa desde el ordenador

Los programas del robot pueden ser ejecutados directamente desde el software de simulación RoboDK siempre que se haya establecido una conexión de red. Para ello, se deben seguir los siguientes pasos:

1. Hacer clic derecho en el robot en RoboDK.
2. Seleccionar *Conectar al robot*.

3. Introducir la IP del robot.
4. Seleccionar *conectar.*

Una vez la conexión haya sido establecida con éxito, deberá aparecer una barra verde que indica "Listo" (Figura 32).

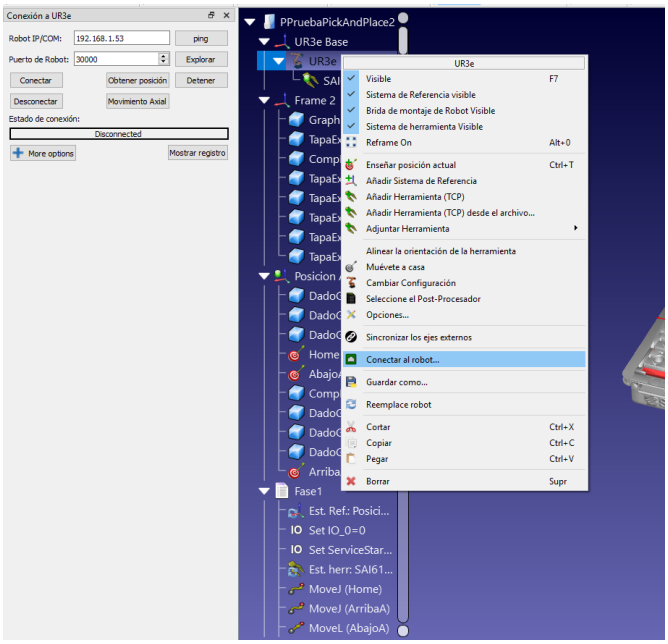

Figura 32.- Pantalla para conectar al robot en RoboDK

La IP del robot puede ser recuperada desde el menú *Acerca de* disponible en la pantalla principal del TeachPendant (Figura 33).

Figura 33.- Pantalla "Acerca de" en el TeachPendant [26]

Con objeto de crear la conexión, es necesario que el sistema robótico y el ordenador estén en la misma subred. Para ello, primero se creará la conexión entre ambos mediante un cable de red. A continuación, se mirará cuál es la IP del robot, cuáles son las redes disponibles en el ordenador y se seleccionará la red conectada al cable que

va hasta el robot. Si ambas subredes no coinciden, habrá que cambiar la dirección IP del ordenador haciendo que coincida la subred con la del robot. De esta manera, al crear la conexión como se ha explicado previamente y hacer *ping,* se podrá confirmar que ambos dispositivos pueden verse y crear correctamente la conexión.

Hay dos métodos diferentes para poder ejecutar el programa desde el ordenador: el primero se basa en la programación en línea y el segundo en la programación fuera de línea [27].

- **Programación en línea**

Este método permite ejecutar paso a paso el programa en el robot. El robot funciona como un servidor, y cada instrucción es enviada al robot paso a paso, ya que el programa se ejecuta en el simulador. Para ello, hace falta elegir la opción de *Ejecutar en el robot* (Figura 34), y a partir de ese momento cada vez que se haga doble clic en un programa, se ejecutará en el robot real.

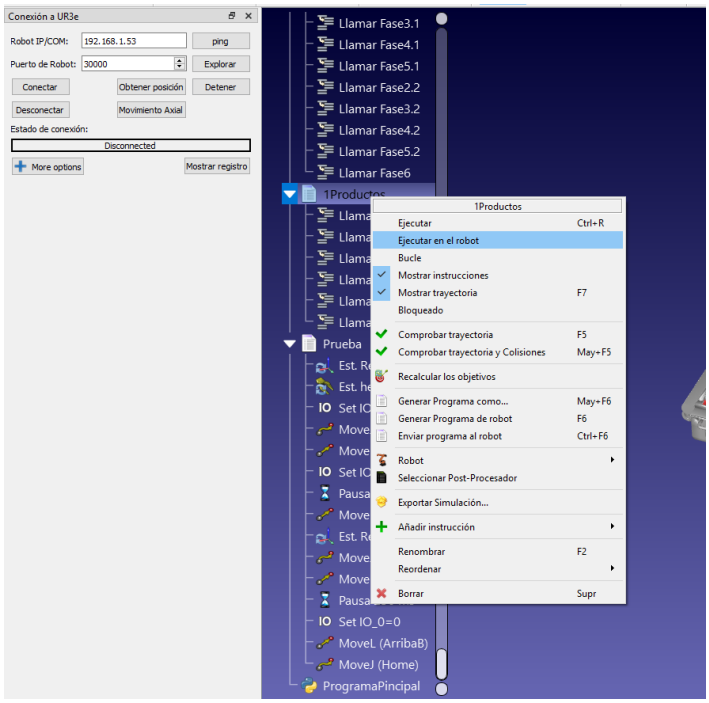

Figura 34.- Pantalla para ejecutar programa en el robot en RoboDK

De esta manera, no será necesario convertir el script entero para que el sistema robótico pueda entenderlo. Al tener una conexión creada y estar haciendo uso de ella,

los comandos que el robot debe ir ejecutando se irán convirtiendo paso a paso. En caso de parar la ejecución en medio del proceso, no será necesario convertir los comandos que faltan por ejecutar.

Por otra parte, también puede ser un método contraproducente. La red puede fallar en cualquier momento, por lo tanto, si el fallo de red se produce en medio de una ejecución de un proceso, éste se parará dado que el sistema robótico no será capaz de recibir las órdenes desde RoboDK.

- **Programación fuera de línea**

Este método genera el programa completo, lo transfiere al robot y el programa se ejecuta en el robot real, siendo una programación fuera de línea. Para ello, se elegirá la opción *Enviar programa al robot* (Figura 35) y de esta manera el sistema robótico comenzará la ejecución del proceso.

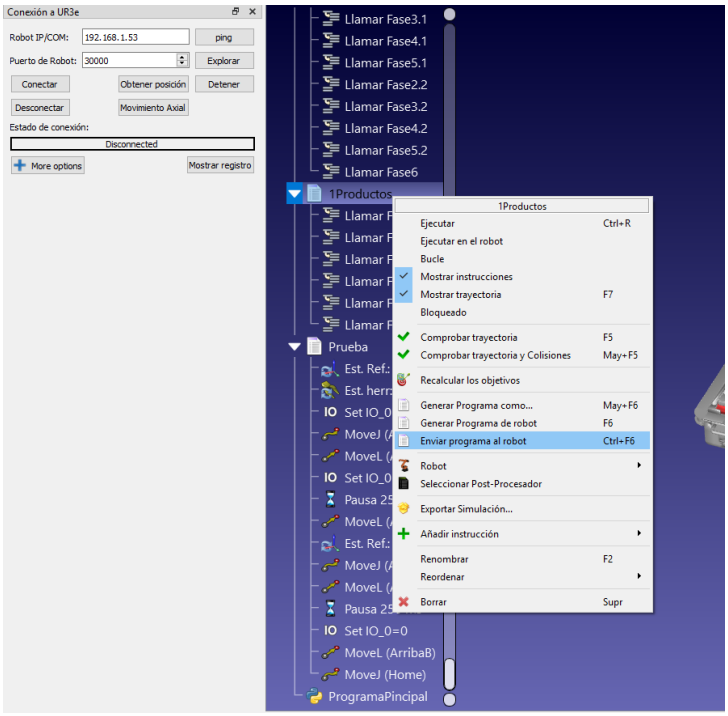

Figura 35.- Pantalla para enviar programa al robot en RoboDK

En este caso, aunque la red padezca un fallo en medio del proceso, el robot seguirá con dicho proceso, dado que recibe el programa completo desde un principio en el lenguaje apropiado para poder ejecutarlo.

1.7.5.2 Ejecutar el programa desde el controlador UR

Este método es más sencillo que el anterior, por lo que se utilizará simplemente para hacer algunas pruebas. Una vez terminado el proceso en el software de simulación, será necesario comprobar que los diferentes puntos fijados en RoboDK coinciden con el entorno real. Para ello, no es necesario que el sistema robótico tenga ninguna conexión con el PLC o con el software de simulación.

Así, se crearán simples programas de prueba en el software de simulación que muevan el brazo robótico entre las diferentes posiciones fijadas durante el proceso. De este modo, se podrá comprobar si, en el entorno real, las pinzas son capaces de coger las piezas que se necesitan para cada fase del proceso o es necesario ajustar las posiciones.

Generalmente, suele ser necesario ajustar algunas de las posiciones, dado que al crear el entorno en el software de simulación se usan las mediciones realizadas en el escenario real para después colocarlo todo en el simulador, pero esas mediciones es posible que tengan pequeños errores que pueden ser determinantes a la hora de coger un objeto.

1.7.6 Pruebas realizadas y resultados obtenidos

En este apartado se resumirán las diferentes pruebas hechas durante el proceso y los problemas que han surgido.

Al hacer pruebas en el software de comunicación, en el programa principal han surgido algunos errores de código. Cada fase del proceso se puede programar mediante la interfaz gráfica de RoboDK, pero a la hora de programar el programa principal, es necesario hacerlo mediante un script en el lenguaje de programación Python. Como la llamada a los programas que deben ejecutarse no están en scripts

de Python, el programa principal no puede acceder a ellos. Para poder solucionarlo, se ha utilizado la API de RoboDK, que permite acceder a todo lo anteriormente programado mediante la interfaz gráfica. Una vez hecho el cambio, el software de simulación ha sido capaz de ejecutar el programa sin mayores complicaciones. De esta manera, se han realizado correctamente las pruebas para asegurar que el programa ejecuta el proceso adecuado dependiendo de la cantidad de productos que debe montar en cada momento.

Por otro lado, será necesario comprobar que la conexión entre el software de simulación y el robot funciona correctamente, y que los programas creados en RoboDK pueden ejecutarse en el robot coincidiendo los puntos fijados en el simulador con los del entorno real.

Como el entorno no está completamente montado, será imposible probar el proceso real al completo, pero para comprobar si los puntos fijados en el software de simulación coinciden con los de la mesa de montaje, se han creado unos programas de prueba en RoboDK haciendo que el brazo robótico se mueva entre los diferentes objetivos. Al hacer estas pruebas, se ha podido comprobar que los puntos no coinciden con total exactitud, por lo que ha sido necesario calibrarlos en el software de simulación.

Al comprobar la conexión entre RoboDK y el sistema robótico, se ha presentado un pequeño problema y la conexión no se podía crear. La razón ha sido que no estaban conectados a la misma subred, por lo que no podían verse entre ellos. Una vez cambiada la dirección IP del ordenador y puesta la misma subred del robot, la conexión ha sido creada correctamente.

Para finalizar con las pruebas, hay que comprobar que la conexión entre el software de simulación y el PLC creada mediante OPC UA funciona correctamente. Una vez creada la conexión desde el PLC (cliente) y comprobado que dicha conexión funcionaba correctamente, al configurar los parámetros de lectura y escritura que deberá intercambiar con el software de simulación (servidor), se ha presentado un pequeño problema. Los diferentes parámetros que se han configurado en el servidor

no pueden verse en el PLC, sino que se puede ver un parámetro general llamado *StationParameter*. Para poder comprobar si dentro de dicho parámetro se encuentran todos los configurados en el software de simulación, se ha decidido descargar el programa UA Expert. Como este programa funciona como cliente OPC UA, creando una conexión con el servidor es posible ver todos los parámetros que se tienen en cuenta en el parámetro general que se ve en Tia Portal *(StationParameter)* y se podrán configurar dándoles el valor adecuado en cada momento.

2. METODOLOGÍA

En este apartado se explicará la metodología que se ha seguido para llevar a cabo este proyecto. Para ello, primero se hará una descripción de las tareas, a continuación, se separarán dichas tareas en 4 fases generales, y se terminará con un diagrama Gantt que recoja toda esa información.

Las diferentes tareas realizadas durante el proyecto son las siguientes:

1. Gestión del Proyecto
2. Preparación
3. Ubicación del tema
4. Puesta en marcha del robot
5. Simulación
6. Código para el PLC
7. Conexión simulador-PLC
8. Conexión simulador-sistema robótico
9. Verificación y obtención de resultados
10. Documentación
11. Presentación del proyecto

2.1 Descripción de tareas

2.1.1 Gestión del proyecto

Mientras se va haciendo el proyecto, es importante tener una gestión continua del mismo. Para ello, será importante marcar unos objetivos con fecha límite, y será imprescindible ir cumpliendo los hitos marcados para poder ir avanzando en el proyecto.

2.1.2 Preparación

En esta fase de preparación, se definirán las tareas necesarias junto con la directora del proyecto y el tiempo aproximado para cada tarea. Utilizando el diagrama Gantt para poder visualizarlo de una manera más clara y práctica.

2.1.3 Ubicación del tema

Lo primero que se debe hacer en un proyecto de este tipo es buscar información sobre el tema principal. Para ello, es conveniente leer diferentes artículos relacionados con el tema, para poder enriquecer el conocimiento que se tiene del mismo.

2.1.4 Puesta en marcha del robot

Al ser un robot nuevo y nunca antes utilizado en el laboratorio, es importante leerse el manual para saber cómo usarlo. Una vez leídas las partes importantes del manual, se debe proceder al montaje del robot, seguido de su puesta en marcha. Para ello se probará con un programa sencillo que simplemente mueva el brazo robótico entre dos posiciones anteriormente marcadas.

2.1.5 Simulación

Para la simulación del proceso se utilizará el software de simulación RoboDK, con el que se programará el proceso que ha de seguir el robot en el mismo ámbito del laboratorio, esto es, utilizando la mesa de montaje y las diferentes piezas para poder llevar a cabo el proceso de montaje del producto.

2.1.6 Código para el PLC

El código para el PLC no será necesario escribirlo partiendo de cero, ya que se aprovechará el código utilizado en un proyecto paralelo se harán los cambios pertinentes para poder adecuarlo a este proyecto.

2.1.7 Conexión software de simulación-PLC

Es necesario crear una conexión entre el software de simulación y el PLC para que pueda haber una transferencia de datos entre ambos. Para ello, se utilizará OPC UA, y será necesario configurar uno de ellos para que tenga el rol de servidor y el otro para que tenga el rol de cliente.

2.1.8 Conexión software de simulación-sistema robótico

Una vez hecha la conexión entre el software de simulación y el PLC, será necesario conectar este primero con el sistema robótico. Dicha conexión se creará conectando el RoboDK al robot con objeto de ejecutar el script en dicho robot.

2.1.9 Verificación y obtención de resultados

Se harán las pruebas necesarias para poder comprobar que todo funciona como se esperaba, haciendo pruebas tanto en el software de simulación como en el sistema robótico.

2.1.10 Documentación

A medida que se vaya realizando el proyecto, se deberá ir recabando información. En un principio se buscará documentación más teórica con el fin de ir completando la parte teórica del documento. Una vez finalizado el proyecto y realizadas todas las pruebas, se podrán cumplimentar los apartados prácticos del documento completándolo en su totalidad.

2.1.11 Presentación del proyecto

Aunque terminar el proyecto, comprobar que todo funciona bien y redactar el documento son apartados muy importantes, la presentación final del trabajo realizado también tiene mucha importancia. Para ello, se preparará una presentación clara que no contenga mucho texto, en la que la información aparezca de una manera adecuada, dando importancia al trabajo realizado.

2.2 Fases

En este apartado se explicarán las diferentes fases seguidas durante el proyecto y las tareas realizadas en cada una de ellas. Para ello, se tendrá en cuenta que el laboratorio está cerrado en agosto.

Tabla 4.- Fase de la gestión del proyecto

Fase 1 – Gestión del proyecto				
Nº de la tarea	Nombre de la tarea	Fecha de inicio	Fecha final	Duración
1	Gestión del proyecto	05/04/2021	05/10/2021	117 días

Tabla 5.- Fase de inicio del proyecto

Fase 2 – Inicio del proyecto				
Nº de la tarea	Nombre de la tarea	Fecha de inicio	Fecha final	Duración
2	Preparación	05/04/2021	07/04/2021	2 días
3	Ubicación del tema	05/04/2021	18/04/2021	10 días

Tabla 6.-Fase del desarrollo del proyecto

Fase 3 – Desarrollo del proyecto				
Nº de la tarea	Nombre de la tarea	Fecha de inicio	Fecha final	Duración
4	Puesta en marcha del robot	19/04/2021	16/05/2021	20 días
5	Simulación	17/05/2021	04/07/2021	35 días
6	Código para el PLC	05/07/2021	18/07/2021	10 días
7	Conexión simulador-PLC	19/07/2021	04/09/2021	13 días
8	Conexión simulador-robot	05/09/2021	08/09/2021	3 días
9	Verificación y obtención de resultados	09/09/2021	17/09/2021	7 días

Tabla 7.-Fase de la presentación del proyecto

Fase 4 – Presentación del proyecto				
Nº de la tarea	Nombre de la tarea	Fecha de inicio	Fecha final	Duración
10	Documentación	05/04/2021	21/09/2021	60 días
11	Presentación del proyecto	22/09/2021	05/10/2021	10 días

Cabe destacar, que aunque las fechas de inicio y final de documentación son desde el comienzo del proyecto hasta la entrega del documento, la duración es bastante más corta, dado que la intensidad ha ido variando a lo largo de la ejecución del proyecto.

2.3 Diagrama de Gantt

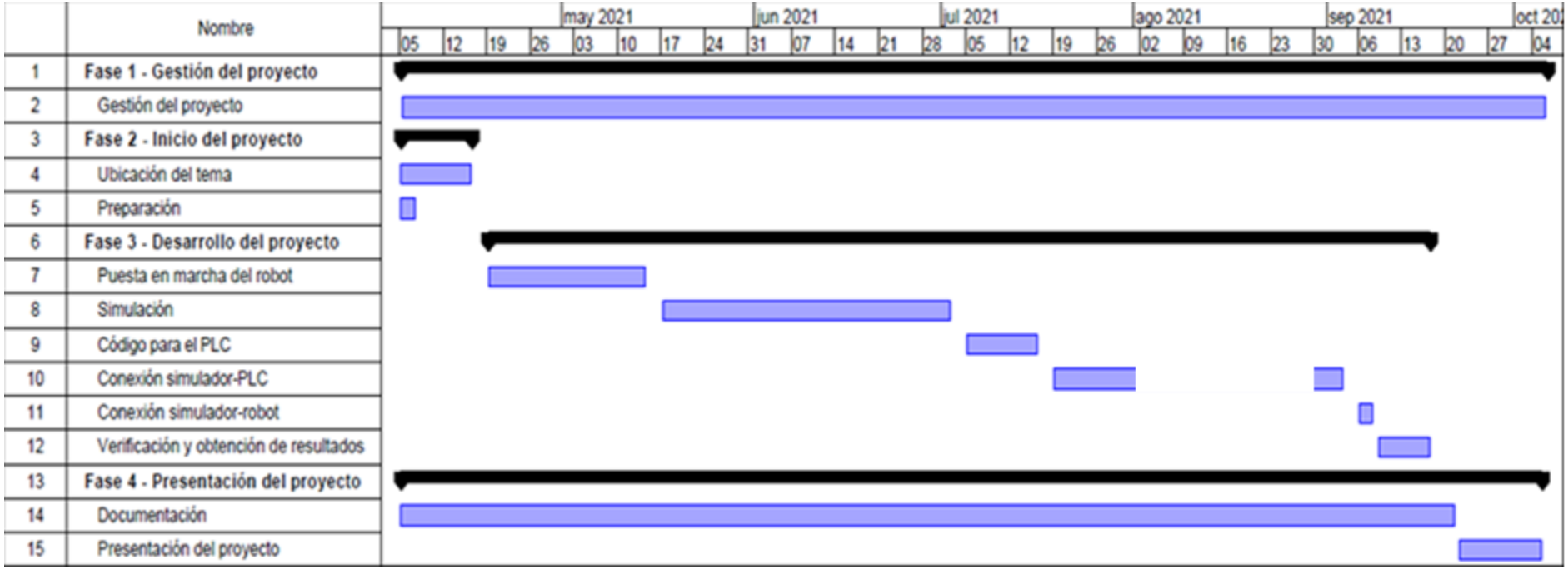

Figura 36.- Diagrama Gantt

3. ASPECTOS ECONÓMICOS

3.1 Presupuesto

En este apartado del documento se presenta el presupuesto del proyecto, para el que se han tenido en cuenta los siguientes factores:

- **Horas internas:** horas totales que han dedicado los trabajadores al desarrollo del proyecto. Los costos por hora han sido fijados en función de la experiencia y responsabilidad de cada trabajador.
- **Amortizaciones:** parte proporcional del coste de la maquinaria utilizada en el proyecto que también será utilizada para futuros proyectos, que se calcula teniendo en cuenta su coste de compra, su vida útil, y las horas utilizadas. En este caso, el cálculo de amortizaciones se ha aplicado al sistema robótico, a la herramienta y al ordenador, ya que todos ellos serán utilizados en futuros proyectos.
- **Gastos:** en este apartado se debería de tener en cuenta el material comprado exclusivamente para este proyecto. En este proyecto, el único material comprado será el material para imprimir las piezas 3D que se utilizarán en el ensamblado del producto.
- **Costes indirectos:** se tendrá en cuenta la luz utilizada para el proyecto, el uso de las instalaciones... siendo un 7% del subtotal.

Tabla 8.- Horas internas

Horas internas			
	Horas de trabajo	**Coste**	**Coste**
Ingeniero Junior	600 h	8 €/h	4800,00 €
Director	25 h	20 €/h	500,00 €
		Total	5300,00 €

Tabla 9.- Amortizaciones

Amortizaciones				
Concepto	**Coste de compra**	**Vida útil**	**Horas de uso**	**Coste**
Ordenador	1200 €	5 años	600 h	16,44 €
Sistema robótico	15000 €	15 años	80 h	0,61 €
Herramienta	1000 €	15 años	80 h	9,13 €
			Total	26,18 €

Tabla 10.- Gastos

Gastos			
Concepto	**Precio**	**Cantidad**	**Coste**
Material para 3D	20 €/Kg	650g	13,00 €
		Total	13,00 €

Tabla 11.- Total de gastos

Total de gastos	
Concepto	**Coste**
Horas internas	5300,00 €
Amortizaciones	26,18 €
Gastos	13,00 €
Subtotal 1	5339,18 €
Costes indirectos (7%)	373,74 €
Subtotal 2	5712,92 €
Imprevistos (5%)	285,65 €
TOTAL	5998,56 €

4. CONCLUSIONES

En este proyecto se ha llevado a cabo la integración del robot colaborativo UR3e en un sistema de fabricación flexible. Son los primeros pasos que se dan con este robot, por lo que será un proyecto con continuación en el futuro.

Teniendo en cuenta que se trata de un robot recientemente adquirido por el laboratorio, se puede decir que se ha logrado su puesta en marcha de manera satisfactoria. Se ha utilizado un software de simulación diferente al anteriormente utilizado en el laboratorio (Tecnomatix), generando así know-how para el grupo de investigación en el uso de esta herramienta. Además, se ha podido ver que RoboDK tiene grandes ventajas. La más significativa puede ser que permite crear una conexión con el sistema robótico y ejecutar los programas directamente desde el ordenador, ahorrando así la conexión entre el PLC y el robot.

Por otra parte, también ha sido enriquecedora la conexión creada entre el software de simulación y el PLC, dado que se ha podido conocer el protocolo de comunicación OPC UA.

En definitiva, este Trabajo Fin de Máster ha contribuido al desarrollo del proyecto de investigación en el que ha tenido lugar, dando los primeros pasos para el aprovechamiento de un nuevo recurso de fabricación como es el robot colaborativo UR3e. En el futuro, se espera que se continúen los trabajos para integrar este robot con un agente que lo conecte al Sistema de Fabricación Flexible que conforma el demostrador del proyecto. Además, se deberá trabajar en ampliar la variedad de tipos de servicio de fabricación que el robot pueda ofertar, programando diferentes tipos de ensamblados.

REFERENCIAS BIBLIOGRÁFICAS

[1] What is Industrie 4.0? (2021). Plattform Industrie 4.0. https://www.plattformi40.de/IP/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html

[2] El-Askalany, A. C. (2019, 21 febrero). The difference between #Industry 3.0 and #Industry 4.0. LinkedIn. https://www.linkedin.com/pulse/difference-between-industry-30-40-ahmed/

[3] Interaction between hierarchies Levels in Industry 3.0 and Industry 4.0. (2021). [Figura]. https://www.researchgate.net/figure/Interaction-between-hierarchies-Levels-in-Industry-30-and-Industry-40-source_fig2_327130483

[4] ¿Qué es la Industria 4.0? (2021). Deloitte. https://www2.deloitte.com/es/es/pages/manufacturing/articles/que-es-la-industria-4.0.html

[5] Germany: Industrie 4.0. (2017).

[6] Lydon, B. (2021, 13 enero). RAMI 4.0 Reference Architectural Model for Industrie 4.0. Isa.Org. https://www.isa.org/intech-home/2019/march-april/features/rami-4-0-reference-architectural-model-for-industr

[7] Advantages & Disadvantages of Flexible Manufacturing System. (2018, 24 julio). CPV Manufacturing. https://www.cpvmfg.com/news/advantages-disadvantages-flexible-manufacturing-system/

[8] Why Cobots? | All the Benefits of Collaborative Robots. (2021). Universal Robots. https://www.universal-robots.com/products/collaborative-robots-cobots-benefits/

[9] UR3e Ficha técnica. (2021). Universal Robots. https://seiki-robotics.com/wp-content/uploads/2019/09/ur3e-tech-spec-es.pdf

[10] ABB Robótica. (2021). Robotics. https://new.abb.com/products/robotics/es

[11] IRB 1100. (2021). Robotics. https://new.abb.com/products/robotics/es/robots-industriales/irb-1100

[12] IRB 1100. (2021b). ABB Robotics. https://search.abb.com/library/Download.aspx?DocumentID=9AKK107046A7014&LanguageCode=en&DocumentPartId=&Action=Launch

[13] RoboDK. (2021). Simulator for industrial robots and offline programming. https://robodk.com/

[14] Avantek. (2021, 27 julio). Tecnomatix: la herramienta que ofrece funcionalidad y ventajas. https://avantek.es/productos/tecnomatix/

[15] SMC GRIPPER MHKL2-16D. (2021b). [Figura]. https://nl.rubix.com/nl/grijpers-separatoren-mhk2/p-G1324050463

[16] Pinza neumática SMC MHKL2-25D. (2021). [Figura]. https://es.rubix.com/es/g1324050467/p-G1324050467

[17] Why was this man smiling at AUTOMATICA? (2021). [Figura]. https://www.therobotreport.com/why-was-this-man-smiling-at-automatica/

[18] Manual de PolyScope. (2021). Universal Robots. https://s3-eu-west-1.amazonaws.com/ur-support-site/16267/Software_Manual_es_Global.pdf

[19] Desarrollo de aplicaciones mediante robots colaborativos basadas en interfaces naturales hombre-máquina. (2018).

[20] Guía Básica - Documentación RoboDK. (2021). RoboDK. https://robodk.com/doc/es/Basic-Guide.html#Start

[21] OPC UA from sensor to cloud. (2021b). [Figura]. https://www.br-automation.com/es-es/tecnologias/opc-ua/preguntas-frecuentes-acerca-de-opc-ua-sobre-tsn/

[22] Qué es la OPC UA y por qué es fundamental para la automatización industrial. (2021). Cognex. https://www.cognex.com/es-es/blogs/machine-vision/why-opc-ua-is-essential-for-factory-automation

[23] Illustration of a client-server network. (2021). [Figura]. https://www.researchgate.net/figure/Illustration-of-a-client-server-network_fig2_2605250

[24] Rinaldi, J. S. (2018, 27 noviembre). OPC UA Client vs. Server. Real Time Automation, Inc. https://www.rtautomation.com/rtas-blog/opc-ua-client-vs-server/

[25] Creating of OPC UA clients with .NET and helper class. (2021). [Figura]. https://support.industry.siemens.com/cs/document/109737901/creating-of-opc-ua-clients-with-net-and-helper-class?dti=0&lc=en-PG

[26] S7 user block for the OPC UA client of a SIMATIC S7-1500. (2021).

[27] Universal Robots - Documentación RoboDK. (2021). RoboDK. https://robodk.com/doc/es/Robots-Universal-Robots.html#UR-RoboDK