UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA.

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

AUTOR: DAYRON RODRÍGUEZ DÍAZ

TUTOR: D. SERGIO GARCÍA-NIETO RODRÍGUEZ

COTUTOR: D. RAÚL SIMARRO FERNÁNDEZ

Curso Académico: 2020-2021

AGRADECIMIENTOS

A todas las personas que han intervenido en mi proceso de formación, mencionando el inicio de los estudios de grado en la universidad en La Habana [*CUJAE]*, como el personal docente de la *UPV*, en especial a mi tutor el D. Sergio García-Nieto Rodríguez por retarme a entregar siempre un poco más en el desarrollo de la presente investigación.

A toda la familia: la de sangre, empezando por mi madre que supo criar un hombre honesto y tenaz, mi abuela que sentó las bases de mi educación y a la familia que uno escoge, mis padres en España, mis suegros que me alentaron a concluir los estudios universitarios, a Concha por apoyarme en los momentos difíciles en este largo periplo, y a mi marido, sin él nunca habría podido conseguir todo lo que actualmente he logrado en la vida……

RESUMEN

El presente trabajo describe el método para implementar una plataforma virtual que permita crear un *Digital Twin* de una celda automatizada con dos robots colaborativos que intercambian una pieza empleando visión artificial (Figura 1). La plataforma es completamente *gratuita* y permite a futuros alumnos e interesados entrenarse o desarrollar proyectos de investigación rompiendo las barreras de acceso a estas tecnologías involucradas en la *Industria 4.0.*

El Digital Twin habilita la migración de todos los programas implementados a los dispositivos físicos en cualquier instante tras la depuración de posibles colisiones y errores en el código.

Para materializar los módulos, se utilizan simuladores de programación fuera de línea como *RoboDK*, para la representación gráfica del proceso y la secuencia de trabajo, *URSim* como modelo fisicomatemático de los robot *UR3e*, *CODESYS* permite emular un *softPLC* con la lógica de control, este componente además tiene embebido un *WebServer* para representar todos los datos y el control de la celda, el cuál presenta un concepto de diseño gráfico desarrollado desde los cimientos del proyecto por el autor, al mismo tiempo y con el mismo propósito, una pantalla *HMI* de *Omron* actúa como interfaz hombre-máquina. La visión artificial se programa empleando lenguaje *Python*, el cual aprovecha las *APIs* de *RoboDK* para simular una cámara 2D.

Todos los elementos se interrelacionan entre sí en tiempo real y se comunican con dos servicios de *cloud computing,* comenzando por *UBIDOTS* para monitorizar parámetros de la celda y establecer la posición en la cual el primer robot entrega la pieza en el proceso de intercambio a través de variables que se manipulan por medio de protocolos como *OPC UA, MQTT, JSON* y nodos implementados en *Node-Red*, la segunda plataforma está desarrollada por *CODESYS Automation Server,* un servicio web donde el usuario puede consultar el estado del controlador, distintas versiones del código entre las cuales se puede elegir cual ejecutar, o programar directamente el autómata desde un navegador web rompiendo la barrera de la programación anclada a un *PC* convencional, pudiendo realizar dicha tarea en una *Tablet* o un *Smartphone* en cualquier punto geográfico.

- ***Palabras Claves:*** *Digital Twin, Industria 4.0., RoboDK, URSim, Python, CODESYS, Robots colaborativos, UR3e, WebServer, Visión Artificial, Diseño gráfico.*

Figura 1. Diagrama de conexiones de todas las herramientas empleadas en el proyecto.

ABSTRACT

This project describes the method to implement a virtual platform to create a *Digital Twin* of an automated cell with two collaborative robots exchanging a part using artificial vision (Figure 1). The platform is completely free of charge and allows future students and interested parties to train or develop research projects by breaking down the barriers of access to these technologies involved in *Industry 4.0.*

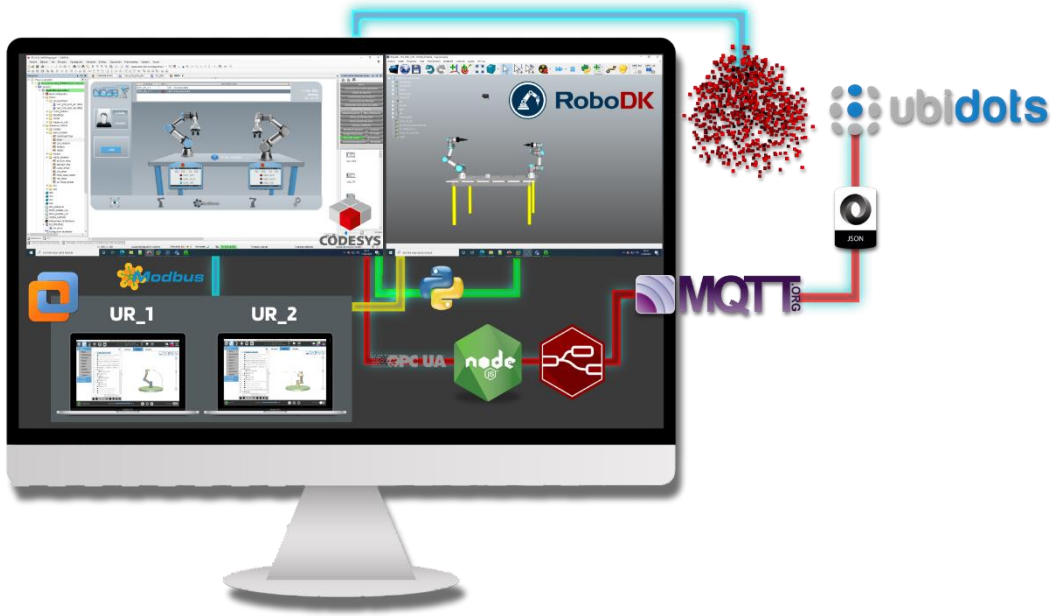
The *Digital Twin* enables the migration of all implemented programs to the physical devices at any time after debugging possible collisions and errors in the code.

To realise the modules, offline programming simulators such as *RoboDK* are used for the graphical representation of the process and the work sequence, *URSim* as a physicomathematical model of the *UR3e* robots, *CODESYS* allows the emulation of a *softPLC* with the control logic, this component also has an embedded *WebServer* to represent all the data and control of the cell, which presents a graphic design concept developed from the foundations of the project by the author, at the same time and for the same purpose, an *Omron HMI* screen acts as a human-machine interface. The machine vision is programmed using *Python* language, which takes advantage of the *RoboDK APIs* to simulate a 2D camera.

All the elements interrelate with each other in real time and communicate with two cloud computing services, starting with UBIDOTS to monitor cell parameters and establish the position in which the first robot delivers the part in the exchange process through variables that are manipulated by means of protocols such as OPC UA, MQTT, JSON and nodes implemented in Node-Red, the second platform is developed by CODESYS Automation Server, a web service where the user can check the status of the controller, different versions of the code from which you can choose which one to run, or program the automaton directly from a web browser, breaking the barrier of programming anchored to a conventional PC, being able to perform this task on a Tablet or Smartphone at any geographical location.

- Keywords: *Digital Twin, Industry 4.0., RoboDK, URSim, Python, CODESYS, Collaborative Robots, UR3e, WebServer, Artificial Vision, Graphic Design.*

Figure 1. Wiring diagram of all tools used in the project.



DOCUMENTOS CONTENIDOS EN EL TFG

- Memoria
- Pliego de Condiciones
- Presupuesto

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

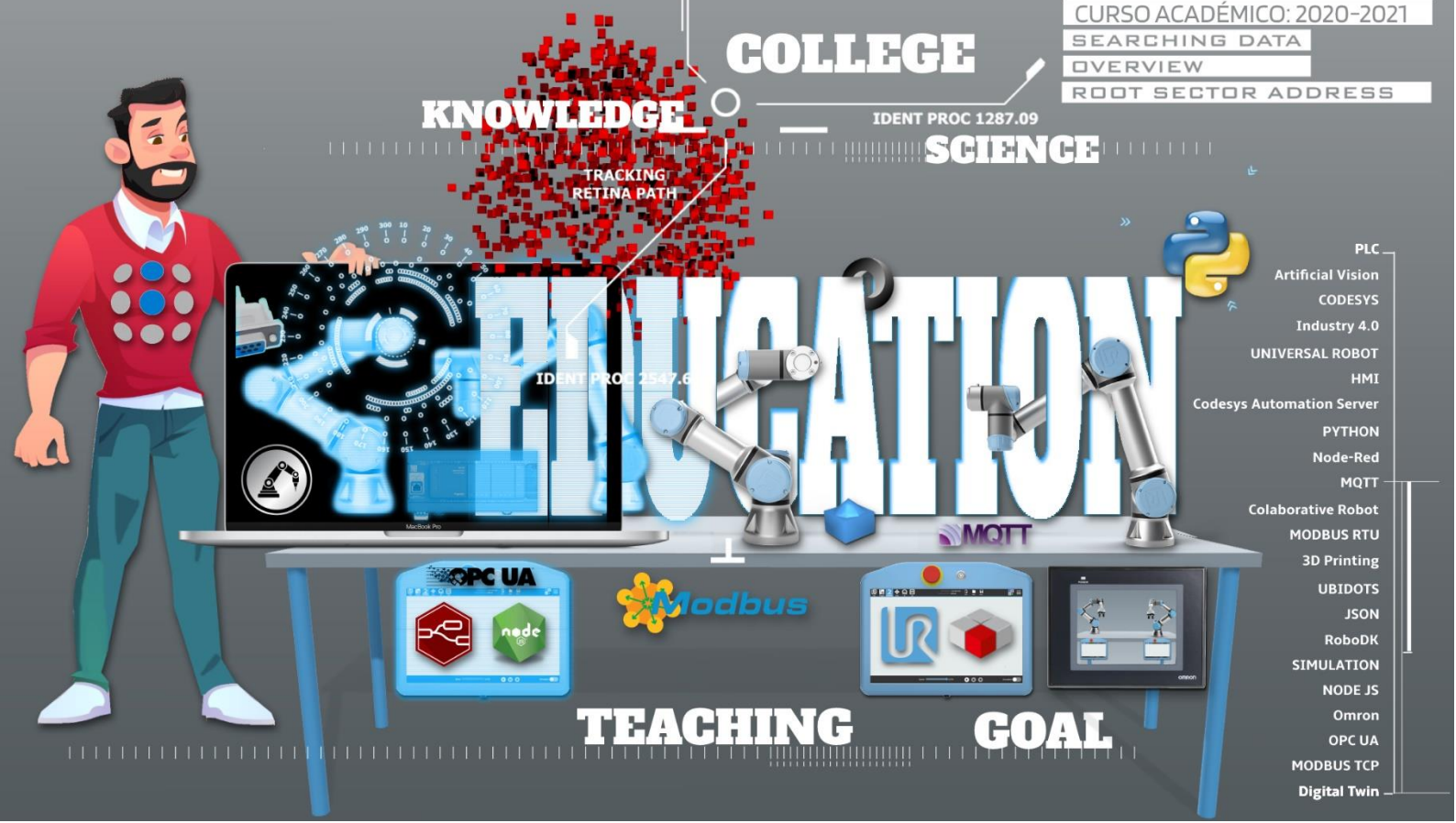
Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA

MEMORIA TÉCNICA DESCRIPTIVA

Dayron Rodríguez Díaz



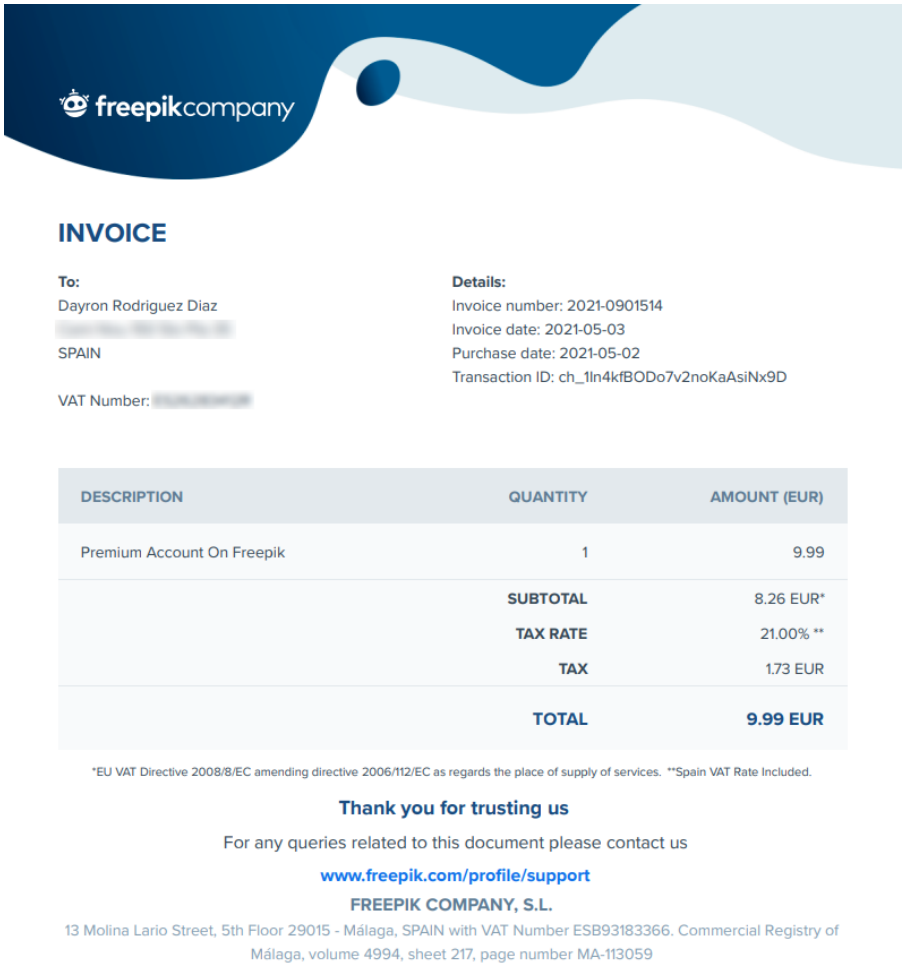
Contenido

NOTA ACLARATORIA:

Según el **Real Decreto 1/1996, del 12 de abril**, por el que se aprueba el texto refundido de la **Ley de Propiedad Intelectual,** donde se regulan, aclaran y armonizan las disposiciones legales vigentes en la materia relativa a la propiedad intelectual de obras; la disposición del público por terceros de cualquier imagen, obra fotográfica o mera fotografía, divulgada periódicamente o en sitio web, estará sujeta a la autorización de su autor.

Existe un supuesto donde se autoriza el **uso de las imágenes sin que medie la autorización de su autor**: **con fines docentes o de investigación**, siempre y cuando se refieran a obras ya divulgadas y se establezca su inclusión a través de citas, comentarios o un juicio crítico. Se deberá indicar la fuente y el nombre del autor de la obra.

Por estos supuestos, el autor incluye debajo de cada imagen la fuente de la cual se extrajo, en caso de imágenes propias generadas en la investigación no se incorpora el mencionado pie de figura. Las imágenes conformadas con partes de otras, o la totalidad de estas, han sido descargadas con licencia *Premium* del sitio www.freepick.es, del cual se tienen guardadas para su consulta todas las licencias. Se adjunta una factura donde aparece el **nombre y apellidos del autor** como prueba de buena voluntad y respeto a la propiedad intelectual de terceros.



freepikcompany

INVOICE

To:
Dayron Rodriguez Diaz
[redacted]
SPAIN

VAT Number: [redacted]

Details:
Invoice number: 2021-0901514
Invoice date: 2021-05-03
Purchase date: 2021-05-02
Transaction ID: ch_1In4kfBODo7v2noKaAsiNx9D

DESCRIPTION	QUANTITY	AMOUNT (EUR)
Premium Account On Freepik	1	9.99
	SUBTOTAL	8.26 EUR*
	TAX RATE	21.00% **
	TAX	1.73 EUR
	TOTAL	**9.99 EUR**

*EU VAT Directive 2008/8/EC amending directive 2006/112/EC as regards the place of supply of services. **Spain VAT Rate Included.

Thank you for trusting us

For any queries related to this document please contact us

www.freepik.com/profile/support

FREEPIK COMPANY, S.L.

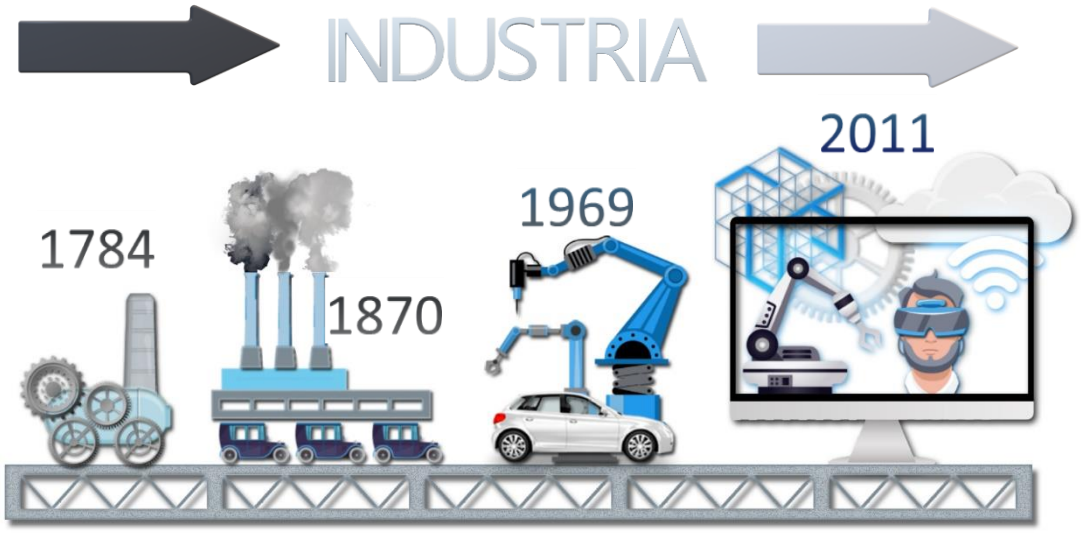
13 Molina Lario Street, 5th Floor 29015 - Málaga, SPAIN with VAT Number ESB93183366. Commercial Registry of Málaga, volume 4994, sheet 217, page number MA-113059

1. INTRODUCCIÓN

Una década ha transcurrido desde que se bautizó, empleando el término "**Industria 4.0**", al nuevo modelo de producción y manufactura que se encuentra en pleno proceso de implantación en el sector industrial. La *Feria de Hannover* [Alemania, 2011], fue el punto de partida de esta nueva era a la cual estamos asistiendo.

En el transcurso del desarrollo de los medios de producción, la humanidad ha cursado tres grandes cambios en el modelo de fabricación, el primero se remonta al siglo XVIII, cuando *James Watt* mejora la máquina de vapor de *Newcomen* e impulsa la primera revolución industrial llevada a cabo en *Reino Unido* en el año 1784, siendo este, el modelo social, económico y de transformación tecnológico más revolucionario en la historia hasta dicha fecha (Figura 2). Los cambios de mayor importancia se vivieron en la industria textil y en concreto en la producción de lana. Un siglo tardó en extenderse por todo el mundo y desde ese momento "*la riqueza y la renta per cápita se multiplicó como no lo había hecho nunca en la historia*" (Lucas, 2004).

Figura 2. Distintas etapas del desarrollo industrial.



La segunda mitad del siglo XIX marca el inicio de la siguiente revolución industrial (entre 1850 -1914): la explotación de los recursos naturales como el gas y el petróleo para producir energía eléctrica, el empleo del acero, las comunicaciones, los nuevos medios de transporte (vehículos a motor, aviones, etc.), la producción en masa y en grandes volúmenes: siendo el caso más representativo el revolucionario concepto de organización de la producción ideado por *Henry Ford* en sus plantas de fabricación del *Ford T* que populariza la producción en cadena; entre otros, son los factores que cambian radicalmente el tejido empresarial mundial y preparan a la industria, no sin transcurrir 100 años, para una nueva revolución.

Tras dos Guerras Mundiales acaecidas principalmente en territorio europeo, la industria necesita resurgir y poner en práctica nuevos modelos y adelantos técnicos. A finales de los 50 del pasado siglo,

se distribuye el primer transistor de silicio comercial de la mano de *Texas Instruments*, constituyendo la unidad básica para el desarrollo de la electrónica en la tercera revolución industrial, además, se produce el auge de la automatización, las tecnologías de la información, la aparición de los robots, los primeros controladores lógicos programables (*PLC*), la electrónica digital, entre otras.

El desarrollo de la red de redes y el uso y la democratización de *Internet,* la implementación del modelo *OSI,* los protocolos de comunicaciones estándares como *TCP/IP*, entre otros, abren un nuevo horizonte a la industria mundial para traspasar la barrera anterior y avanzar hacia la nueva era: ***Industria 4.0***, denominada así por *Klaus Schwab*[1].

El horizonte de esta nueva revolución industrial se ve difuminado y sin fronteras, las barreras entre las personas y la tecnología se entremezclan, campos de la ciencia totalmente excluyentes tiempo atrás, actualmente coexisten y se comunican entre ellos.

Dicho de otro modo: "Las tecnologías digitales permiten la vinculación del mundo físico (dispositivos, materiales, productos, maquinaria e instalaciones) con el digital (sistemas). Esta conexión habilita que dispositivos y sistemas colaboren entre ellos y con otros sistemas para crear una industria inteligente, con producción descentralizada y que se adapta a los cambios en tiempo real". (Blanco et al., 2017)

Dejando a un lado por el momento las áreas que enmarcan esta nueva revolución, caben destacar tres grandes paradigmas asociados al término *Industria 4.0*:

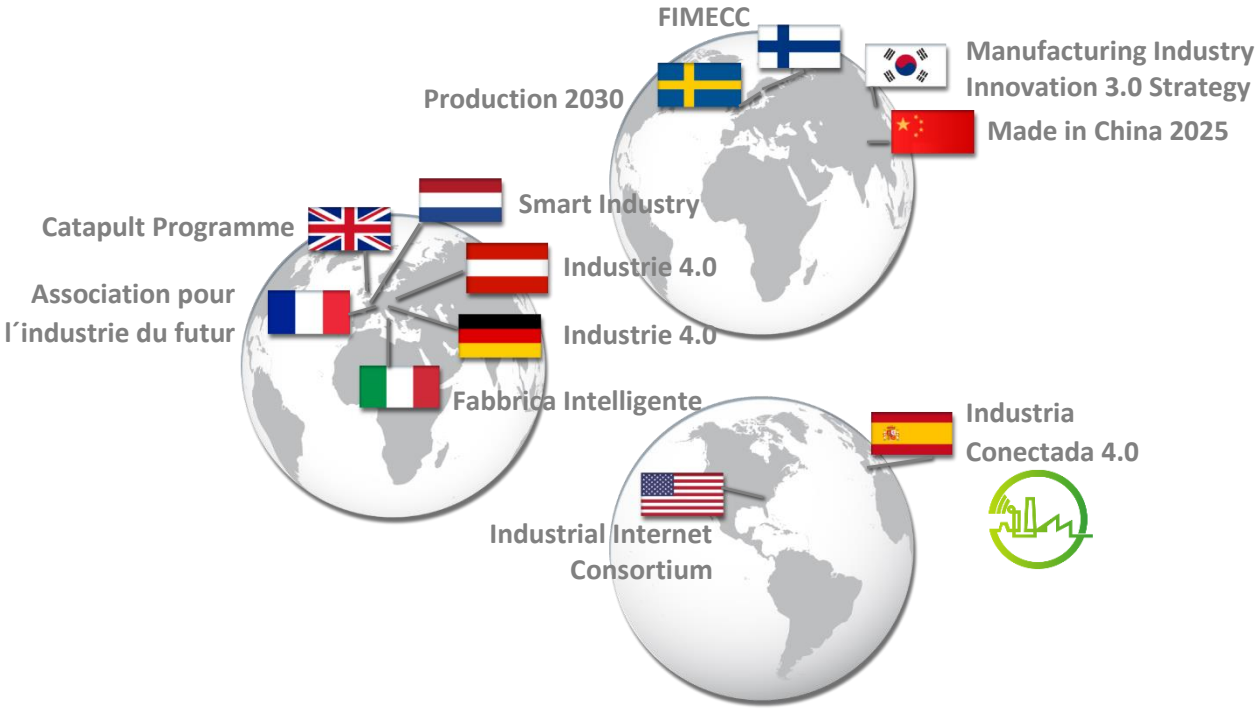
1. Del Hardware al Software y el Contenido: El software adquiere una mayor relevancia y permite que dispositivos antes diseñados y ubicados con un emplazamiento fijo y un número limitado de funcionalidades, puedan actualizarse y se abandone la idea de crear un diseño ideal desde el inicio. El software, que en el mundo de la automatización se denomina *Firmware*, dota de un carácter versátil y adaptativo a cualquier máquina y de esta forma las fábricas pueden responder con inmediatez a cualquier variación de la producción, defectos de diseño, introducción de nuevas gamas, etc.
2. De Usuario a Diseñador: En etapas anteriores, los departamentos de diseño de las empresas eran los responsables de conceptualizar una necesidad o de generar la misma en clientes a través de bienes o servicios. Este sistema de producción tiene la desventaja de no poder cuantificar las respuestas de los consumidores o sus opiniones hasta avanzados períodos de la cadena de producción. En la actualidad, todos disponemos de un teléfono, Tablet, PC, o cualquier otro dispositivo electrónico que permita la interacción y la recogida de datos para evaluar si el producto en cuestión debe ser actualizado, se puede aumentar su producción o directamente debe ser sustituido por otro con mejores prestaciones. En resumen, la nueva industria incorpora a los clientes en el proceso de producción desde fases tempranas del ciclo de fabricación, aumentando los beneficios y garantizando un mayor desarrollo del *PLM (Product Lifecycle Management o Gestión del Ciclo de Vida del Producto, en español).*
3. Creación de Valor a través de Nuevos Modelos de Negocio: Todas las áreas que rodean este nuevo concepto de manufactura son imposibles de ser dominadas por una única empresa, lo que obliga a relacionarse con un amplio tejido de nuevas compañías, Startups o negocios que aporten valor a la fábrica. Con el surgimiento de las nuevas relaciones es muy posible que se detecten vertientes de negocio desconocidas o ignoradas previamente y permitan poner en

[1] Economista y empresario, conocido principalmente por ser el fundador del Foro Económico Mundial; *1938, Rasvensburg, Alemania.*

funcionamiento otras fuentes de ingreso para la empresa. Es por este motivo que las relaciones se deben ampliar todo lo posible, incluso tal y como plantea David Pozo (Dr. Técnico Factory & Process Automation SIEMENS S.A.): "*La empresa debe ser Promiscua"*(PIXEL SISTEMAS, 2016).

Alemania es uno de los motores en cuanto a tecnología e innovación se refiere, es por ello por lo que lidera la oferta y la demanda de tecnologías de producción, empresas de renombre como *Tesla*, *Ford*, *Samsung*, etc., recurren a proveedores alemanes para implantar su tecnología en las fábricas. Desarrollar las nuevas técnicas que se plantean en el ambiente de la *Industria 4.0* es un proceso con elevado coste monetario, esto limita la entrada y el acceso de países en vías de desarrollo a los nuevos recursos, abriendo aún más si cabe, la brecha entre estos y los países del llamado primer mundo. En España, el *Ministerio de Industria, Comercio y Turismo*, crea una iniciativa denominada Industria Conectada 4.0 para fomentar la transición de los actuales sistemas de producción a la nueva era digital con una intervención de capital público y privado, ya que la industria en nuestro país ocupa "*el 13% de valor añadido y mantiene ocupado al 11% de la población"* (Ministerio de Industria, Energía y Turismo, 2016), de esta forma el gobierno central proporciona las herramientas necesarias para que nuestro país compita con el resto de territorios industrializados que ya han propuestos sus respectivos planes de migración industrial (Figura 3).

Figura 3. Diferentes iniciativas planteadas por los países industrializados del primer mundo.



Los nueves pilares sobre los que se sostiene el mencionado modelo de producción constituyen un organismo vivo que con el paso del tiempo pueden convivir con otras tecnologías disruptivas y crear nuevas formas de relacionarse entre ellos (Figura 4). A continuación, se describen cada uno de estos soportes tecnológicos para una mejor comprensión de los objetivos del presente proyecto que se definirán en el siguiente apartado.

- **Sistemas de integración horizontal y vertical**

  Con el intercambio fluido de datos y la interconexión de los medios de producción, existe una integración de todos los departamentos de la empresa ocasionando un gran impacto en la gestión interna. Además, tal y como se comentó anteriormente, el cliente, los proveedores y los fabricantes, se ven envueltos de forma activa y enlazados por los sistemas al ciclo de producción.

- **Robótica colaborativa**

  La incorporación de novedosos sensores para el cálculo de fuerzas, momentos e inercias, han dotado a los robots de un nuevo entorno de trabajo totalmente libre de barreras que les permite estar codo con codo con los humanos. Asimismo, la capacidad de convertirse en elementos cada vez más autónomos, flexibles y adaptativos en la producción, son algunas de las características principales con las cuales se han dotado a los robots, antropomórficos o no, que se encuentran en la *Industria 4.0*. Por otro lado, existe un mayor número de *AVG (Vehículos de Guiado Automático)* en las fábricas que permite la agilización y automatización de un considerable conjunto de procesos industriales como son el transporte autónomo, optimización de entrega de materiales y reducción del volumen de stock en los puestos de trabajo. Se tienen en cuenta también los drones en esta tecnología, según varios autores.

- **Internet de las cosas (IoT)**

  Todos los dispositivos en la industria actualmente tienen una vía de comunicación con el exterior, ya no solo miden, sino también interactúan. La estandarización de los protocolos de comunicación así como el desarrollo de nuevas plataformas como la *Red 5G*, abren una nueva dimensión donde todos los dispositivos del entramado tecnológico se comunican entre ellos a velocidades que admitan una "descentralización del análisis y la toma de decisiones en un lapso de tiempo real" (Blanco et al., 2017).

- **Manufactura aditiva**

  El desarrollo de las herramientas *CAD/CAM* y la posibilidad de imprimir piezas en 3D abre un inmenso horizonte de oportunidades a nuevos procesos de fabricación que eliminan la necesidad de crear moldes partiendo directamente del modelo virtual. Las piezas se fabrican emplean-do nuevas capas y con distintos mate-

Figura 4. Pilares o tecnologías que sustentan el concepto de Industria 4.0.

riales que se disponen unos encima de otros en geometrías tan diversas como la forma hexagonal de los panales de abejas, la proporción aurea de una concha o la red de tejido óseo que se encuentra en el interior de los huesos. Un ejemplo del uso de esta tecnología se encuentra en el sector aeronáutico que permite obtener piezas más resistentes y livianas que reducen el consumo de combustible y el ruido de las aeronaves.

- **Big Data**

  Todos los dispositivos conectados a la red generan un gran volumen de datos y a una elevada velocidad, este conjunto de información debe ser tratada, filtrada, organizada y presentada para extraer las pesquisas necesarias que redunden en una mejora del ciclo de producción, proceso de diseño, nivel de satisfacción de un producto o servicio, o el estado físico y la salud de un paciente en un hospital en tiempo real. En estos momentos es una tecnología con una gran demanda en el sector corporativo, según un estudio realizado por *LinkedIn* el pasado año, "*el Ingeniero de Datos ocupa el puesto 9 entre las 15 profesiones emergentes en España, detectándose un 47% de incremento anual de la demanda de profesionales pertenecientes a este sector*" (*Emerging_Jobs_Report_112119_SP.pdf*, s. f.).

- **Cloud Computing**

  La recolección de los datos requiere de un espacio para su almacenamiento y procesado, en la *Nube* se llevan a cabo todas estas tareas que se sustentan en tres servicios diferenciados: la infraestructura, la plataforma y el software. Esta tecnología de cómputo virtual permite el acceso desde distintos dispositivos y departamentos de la empresa a la información que se presenta en tiempo real con un mínimo esfuerzo administrativo.

- **Simulación de entornos virtuales**

  El desarrollo de la capacidad de cómputo respaldada por la *Ley de Moore*[2], entre otras, hacen posible la simulación del comportamiento de máquinas, sistemas y personas en tiempo real antes de ser materializados, lo que evita la aparición de posibles defectos o errores y redunda en un ahorro en costes y maximización de los beneficios. Además, está ganando cada vez más empuje el fenómeno que describe perfectamente esta metodología conocida como *Digital Twin*[3]. En resumen, la simulación elimina el conocido procedimiento de *Prueba-Error* que tantos inconvenientes presenta.

- **Inteligencia artificial**

  Consiste en la creación de algoritmos de cómputo que puedan analizar grandes volúmenes de información como por ejemplo: comportamientos, lenguajes, visión artificial, lectura de sensores, etc. y permitan, en tiempo real, habilitar a la máquina con características cognitivas y de aprendizaje similar a un ser humano (Boden, 2017). Tiene aplicación en todas las ramas de la ciencia, ayuda a determinar por ejemplo desde la predicción de comportamientos futuros de una población de individuos o conocer el futuro consumo eléctrico de una empresa que permita idear una nueva política de ahorro energético.

- **Ciberseguridad**

  Todas las tecnologías anteriores se ven relacionadas con la *Nube* por ser el soporte físico donde los datos se encuentran almacenados y son procesados, este fenómeno trae aparejado consigo

[2] Aproximadamente cada dos años el número de transistores en un microprocesador se duplica.

[3] Gemelo Virtual: Réplicas virtuales que objetos o procesos que simulan el comportamiento de los sistemas reales.

el ataque y robo de la información por criminales informáticos. Son ampliamente conocidos los ataques cibernéticos a los que se enfrentan organismos mundiales y empresas multinacionales en todo el mundo, es por ello por lo que actualmente la seguridad de todo este volumen de información es una tarea que las empresas cada vez se toman más en serio, evitando la vulnerabilidad de la propiedad intelectual, el *know-how*, los datos personales y la privacidad.

- **Realidad aumentada**

  De todas las columnas sobre la que se apoya la *Industria 4.0*, la realidad aumentada es la menos desarrollada hasta la actualidad, pero según la opinión del autor de este proyecto, será en un futuro una de las más empleadas. Consiste en incorporar al mundo real elementos virtuales que permitan ampliar la información que se tiene de un producto en un instante determinado, por ejemplo, un operario utilizando un soporte físico como unas gafas, podría ver un procedimiento de reparación, o un conjunto de características de una pieza específica. Será, además, un elemento revolucionario en el sector de la docencia, la medicina y la construcción entre otros campos en los próximos años.

La *Industria 4.0* solo se impondrá si se adoptan un grupo de acciones correctivas necesarias para su desarrollo, cabe mencionar: la estandarización de los sistemas, plataformas y protocolos, la disponibilidad de trabajadores con formación suficiente en cada una de las áreas, mayor inversión pública y privada, acceso de todos los países a dichas tecnologías, impulso de las energías renovables y cuidado del medio ambiente y de forma geográfica, una iniciativa europea que englobe a todos los países comunitarios bajo un único modelo de implantación que propaguen la migración a la era digital.

El presente proyecto se apoya en 4 pilares de la *Industria 4.0* para llevar a cabo una simulación de una celda automatizada con robótica colaborativa, estos son: la simulación de entornos virtuales, el empleo de robot colaborativos, *IoT* y *cloud computing.*

2. OBJETO Y ALCANCE DEL PROYECTO

El objetivo general que persigue el proyecto es crear una plataforma multiherramienta que permita diseñar un *Digital Twin* de una celda automatizada con robótica colaborativa. **El proceso industrial consistirá en que dos robots colaborativos logren intercambiar una pieza entre ellos, utilizando visión artificial para detectar la posición de entrega, siendo la misma configurable desde un servicio en la *Nube***. La estación virtual pasa a constituir un recurso formativo con amplia mutabilidad para el aprendizaje y formación de futuros alumnos o personal del sector, en el campo de la automatización y la *Industria 4.0*.

Los objetivos específicos siguientes permiten alcanzar el propósito mencionado de forma previa:

- Introducir en el proyecto un modelo fisicomatemático exacto de dos robots colaborativos que permita obtener los parámetros necesarios para su control y manejo.
- La lógica de control de todos los elementos de la plataforma debe estar centralizada en un *PLC* virtual.
- El *PLC* debe tener un servicio *WebServer* desde el cual se podrá visualizar toda la información relativa a la celda, como son el diagnóstico de todos los dispositivos, estado de las comunicaciones, alarmas, parámetros de configuración y trabajo, permitir el accionamiento de determinadas señales, así como la representación exacta del punto de funcionamiento real de la celda.
- Diseñar un algoritmo para la visión artificial que permita el intercambio de la pieza.
- Crear un dispositivo de interfaz humana o *HMI* sobre el cual el usuario pueda interactuar de forma tangible con la instalación.
- Idear un plan que abarque una línea de diseño de la interfaz y la experiencia de usuario con una presentación depurada, funcional y equiparable a las actuales tendencias de diseño de aplicaciones digitales.
- Programar un *Dashboard* en una plataforma de *cloud computing* para monitorizar parámetros de la celda y configurar en qué posición el primer robot debe entregar la pieza.
- Alojar toda la programación del *PLC* en un servicio web donde el usuario pueda consultar el estado del controlador, distintas versiones del código entre las cuales se puede elegir cual ejecutar, o programar directamente el autómata desde un navegador web rompiendo la barrera de la programación anclada a un *PC* convencional, pudiendo realizar dicha tarea en una *Tablet* o un *Smartphone* en cualquier punto geográfico.
- Conectar, empleando diversos *Protocolos de Comunicación Industrial,* así como varios lenguajes de programación, los distintos elementos para su coexistencia en la plataforma.
- Simular la aplicación en un entorno gráfico *CAM* que permita observar de forma “real” el funcionamiento de la celda.
- Las herramientas que no sean de uso libre deben contar con un período de prueba de al menos un mes.
- Obtener los programas de cada dispositivo simulado para ser cargados en los equipos físicos reales tras realizar una depuración del código en el entorno virtual.

El autor quiere hacer un paréntesis en la relación de los objetivos específicos del proyecto para incluir, de forma paralela, el compromiso propio de que este trabajo sirva como contribución al desarrollo de al menos cinco de los ***Objetivos de Desarrollo Sostenibles*** marcados por la ***ONU***: (Gamez, s. f.)

- ***Objetivo 4***: Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos.
- ***Objetivo 8***: Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos.
- ***Objetivo 9***: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación.
- ***Objetivo 10***: Reducir la desigualdad en y entre los países.
- ***Objetivo 17***: Revitalizar la Alianza Mundial para el Desarrollo Sostenible.

Figura 5. Objetivos de desarrollo sostenible definidos por la ONU.

Fuente: [Objetivos y metas de desarrollo sostenible – Desarrollo Sostenible (un.org)]

Ofreciendo todo el contenido que encontrará en el presente documento como una guía de aprendizaje y adquisición de competencias que eleven el nivel cultural y la capacidad de desarrollo del lector en cualquier sitio donde se encuentre.

Queda excluida de este proyecto, toda la lógica de control superior de la celda que se puede implementar siguiendo la guía *GEMMA* o cualquier método similar. El inicio del ciclo se realizará en la aplicación gráfica y se ejecuta en bucle hasta que el usuario lo decida.

3. ESTUDIO DE NECESIDADES

3.1. ANTECEDENTES

El sector industrial demanda una ingente cantidad de programadores de robótica que es imposible satisfacer por el escaso número de especialistas en la rama, gran parte de este problema reside en el alto coste económico de los robots, este hecho limita la posibilidad de crear centros de formación con una infraestructura adecuada.

De forma general, el precio de un robot colaborativo de la empresa *Universal Robot* gama *e-Serie UR3e* es de *19.250 €*, dato suficiente para comprender que un estudiante recién graduado no tenga formación específica ni experiencia con robots de forma directa actualmente. En el departamento de *Ingeniería de Sistemas y Automática* de la *UPV* solo existen 2 unidades *UR3e*, limitando el número de investigaciones, proyectos y prácticas de laboratorio que se puedan realizar empleando este recurso.

El *Grado en Ingeniería Electrónica Industrial y Automática* cuenta con una asignatura denominada *Sistemas Robotizados* donde se imparten los conocimientos relacionados con el control y la programación de robots, la representación de la posición y orientación, los conceptos de espacios de trabajo, matrices de transformación, entre otros, que sientan las bases en el alumnado para la posterior comprensión y manejo de robots industriales. Sin embargo, el número de prácticas donde se tiene contacto real con un robot físico es “0”, obedeciendo a todo lo mencionado anteriormente.

Cabe resaltar que existen laboratorios donde se hace uso de la herramienta *RobotStudio* de *ABB* para controlar un robot *IRB120* y familiarizar a los estudiantes con el control de un brazo antropomórfico y el lenguaje *RAPID*, por medio de instrucciones de movimiento, control de tiempo, procesamiento de señales *I/O*, flujo y el manejo de diferentes tipos de datos. La limitación radica en que dicho software solo puede controlar los robots de la marca y no es libre, aunque cuenta con un período de evaluación de un mes.

Existen proyectos e investigaciones anteriores que intentan dar una solución similar a la que se persigue en el presente proyecto, como es el caso del *TFM* realizado por Eduardo Pérez Belzuz en la *UPC* (*Belzuz - Convocatoria de entrega del TFM.pdf*, s. f.), el cual sienta las bases para crear un sistema híbrido entre elementos físicos como es un *PLC Siemens* con un módulo *IoT 2040* y la máquina virtual de *Universal Robots URSim*, sin embargo, a juicio del autor de este proyecto, dicha publicación no presenta una solución real al problema planteado, si todos los elementos no pueden ser simulados, el número de trabajos que se pueden realizar con los dispositivos que se tienen se ve automáticamente reducido.

3.2. NECESIDADES Y LIMITACIONES

Tras cinco años como *Programador de Robótica Industrial,* el autor del proyecto conoce perfectamente las barreras de entrada a este sector industrial, el secretismo profesional, así como el difícil acceso a cursos de formación específicos de cada marca de robots, tanto por el precio, como por el limitado número de horas de cada training, llegándose a impartir conocimientos muy básicos que tras finalizar el temario no se pueden poner en práctica ni desarrollar.

Todo lo anterior plantea al autor el reto de democratizar el acceso a la formación y acercar a los alumnos e interesados en la materia, una solución que les permita familiarizarse, aprender, simular, ensayar o validar sus conocimientos en esta área de la ingeniería en pleno crecimiento con un coste nulo, necesitando únicamente un *PC*.

En resumen, existe la necesidad de crear toda una herramienta de simulación que englobe un modelo de un robot real que sea capaz de comunicarse, empleando algún *Protocolo de Comunicación Industrial,* con un elemento de control superior (*PLC*), dispositivo que también debe ser simulado, además, el *PLC* debe ser capaz de atender solicitudes de acceso a datos desde un servidor Web, una pantalla de usuario, una plataforma de *Cloud Computing* un programa de simulación de elementos *CAD*.

Este conjunto de necesidades planteadas se justifica a través de los siguientes factores:

- La plataforma debe ser implementada por cualquier usuario con unos conocimientos medios en los procedimientos aquí descritos (alumno del grado en 4to año).
- El método debe ser un *Digital Twin* de un sistema físico existente que permita planificar, desarrollar e implementar posteriormente todos los programas diseñados en los dispositivos reales.
- Este planteamiento elimina de un plumazo todos los aspectos relativos a la seguridad y la posibilidad de dañar algún elemento de control del sistema, pudiendo detectar previamente posibles colisiones, activación de elementos de control no deseados o fallos en el diseño.
- Desde el punto de vista de mantenimiento, la plataforma no exige ninguno, salvo las modificaciones necesarias que el usuario requiera para poner en evaluación sus necesidades.
- El tráfico de información y datos debe llevarse a cabo en tiempo real entre todos los elementos de la plataforma.
- El coste del diseño y la utilización de la plataforma debe ser "*Gratuita",* es el compromiso que el autor hace con el lector, asegurando que al menos se tiene un mes de plazo para desarrollar su investigación haciendo uso del fruto de este proyecto.

Cabe mencionar que, como todo trabajo de investigación, el presente cuenta con ciertas limitaciones, destacando entre ellas:

- El autor no diseña una secuencia de control superior basado en algún procedimiento o protocolo.
- No existe un manejo prioritario de señales de emergencia de la celda.
- No se incluye en todos los elementos de monitorización la totalidad de los datos a los que se tiene acceso, contando solamente el *WebServer* del *PLC* con esta funcionalidad, tanto el *HMI*

como el *Dashboard* de *UBIDOTS* presentan una muestra de dichos datos para ejemplificar como se establece la comunicación entre estos elementos y el *PLC*.

3.4. NORMATIVA

Para la realización y diseño del proceso de creación de la plataforma que en el presente proyecto se describe, es necesario tener en cuenta la siguiente normativa vigente:

- UNE-EN 61131-3:2013 (Ratificada): Autómatas programables. Parte 3: Lenguajes de programación.
- UNE-EN 60848:2013: Lenguaje de especificación *GRAFCET* para diagramas funcionales secuenciales.
- UNE-EN 60870-5-101:2003: Norma para la monitorización de sistemas de energía, control y las comunicaciones asociadas a los mismos.
- UNE-EN 61508:2011: Seguridad funcional de los sistemas eléctricos/electrónicos/electrónicos programables relacionados con la seguridad.
- UNE-EN ISO 13850:2016: Redes de comunicación industrial. Redes de automatización de alta disponibilidad.
- UNE-EN IEC 61158-4-24:2019 (Ratificada): Redes de comunicaciones industriales. Especificación de *Fieldbus*. Parte 4-24: Especificación del protocolo de la capa de enlace de datos.
- UNE-EN IEC 62769-115-2:2020: Integración de dispositivos de campo (*FDI*). Parte 115-2: Perfiles. *Modbus-RTU*.
- ISO 15745-4:2003/Amd 2:2007: Profiles for *Modbus TCP*.
- BS EN IEC 62453-71. Field device tool (*FDT*) interface specification. Part 71. *OPC UA* Information Model for *FDT*.
- UNE-EN ISO 10218-1: Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 1: Robots.
- UNE-EN ISO 10218-2: Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 2: Sistemas robots e integración.
- ISO/TS 15066:2016: Robots y dispositivos robóticos. Robots colaborativos.
- UNE 0060: *Industria 4.0*. Sistema de gestión para la digitalización. Requisitos.
- UNE 0061: *Industria 4.0*. Sistemas de gestión para la digitalización. Criterios para la evaluación de requisitos.
- ISO/IEC 30161:2020: Internet of Things (*IoT*) -- Requirements of *IoT* data exchange platform for various *IoT* services
- ISO/IEC TR 30166:2020: Internet of things (*IoT*) -- Industrial *IoT*.
- ISO/IEC 20933:2019: Information technology—Distributed application platforms and services (DAPS) -- Framework for distributed real-time access systems.
- ISO/IEC 19944-1:2020: Cloud computing and distributed platforms - Data flow, data categories and data use—Part 1: Fundamentals.
- ANSI/ISA-101.01-2015: Interfaces Humano-Máquina para Sistemas de Automatización de Procesos.

- ISO/TR 24464:2020: Automation systems and integration—Industrial data—Visualization elements of digital twins.
- UNE-EN ISO/ASTM 52900:2017: Fabricación aditiva. Principios generales. Terminología.
- UNE-EN ISO/ASTM 52921:2017: Terminología normalizada para la fabricación aditiva. Sistemas de coordenadas y métodos de ensayo.

3.5. DESCRIPCIÓN DE LAS PRINCIPALES CARACTERÍSTICAS DESEABLES A OBTENER

Con los objetivos, las necesidades y limitaciones del proyecto plasmados previamente, a continuación, se detallan todos los elementos necesarios, así como su interrelación para concretar el desarrollo de la plataforma que simula el control de una celda automatizada con robótica colaborativa.

Todos los elementos deben ser virtuales para brindar la posibilidad de acceso a los usuarios interesados y desarrollar futuros proyectos de investigación en la rama de la *Industria 4.0.* Además, la intercomunicación entre todos los elementos debe desarrollarse a través de protocolos de comunicación industriales estudiados previamente en el grado, fundamentalmente los que están basados en tecnología *TCP/IP*, debido a su distribuida presencia en la industria. El *PLC* será el elemento que gobierne la lógica de control de la celda, con el cual el resto de los dispositivos y servicios se comunicarán para el intercambio de señales y datos.

Dos robots colaborativos, modelo *UR3e,* serán los principales actuadores en la celda, los cuales, a través de un sistema de visión artificial que se conecta al autómata, permitirán que exista el intercambio de una pieza entre ellos.

El usuario debe ser capaz de interactuar con la celda, así como poder visualizar y monitorizar los procesos desarrollados en la secuencia de trabajo, para ello, se diseñará un *WebServer,* alojado en el *PLC* y un *HMI* que permitirán el control y la representación de los parámetros de la estación.

Figura 6. Descripción gráfica de los elementos necesarios para implementar la plataforma.

Por último, se debe incluir un servicio de computación en la *Nube* que permita ajustar la posición en la cual el primer robot debe entregar la pieza, se valorará la posibilidad de acceder desde un *Smartphone* o una *Tablet* en cualquier punto geográfico, así como la viabilidad de comunicar el *PLC* con una plataforma que permita consultar el estado del controlador, distintas versiones del código entre las cuales se puede elegir cual ejecutar, o programar directamente el autómata desde un navegador web rompiendo la barrera de la programación anclada a un *PC* convencional.

4. PLANTEAMIENTO DE SOLUCIONES ALTERNATIVAS

Para construir la plataforma de simulación de una celda automatizada con robótica colaborativa se tienen en cuenta las siguientes soluciones a los dispositivos necesarios que a continuación se detallan:

4.1. PLC

El *PLC* es órgano central de control del sistema, por ello su elección debe analizarse con detenimiento, teniendo en consideración, además, el amplio número de requisitos que se establecen en este proyecto. El autor del presente trabajo analiza 4 marcas de *PLC* que podrían solventar la implementación de toda la lógica de control y las comunicaciones:

- ***CP2E***: Es un micro *PLC* de *Omron* diseñado para equipos compactos compatibles con recopilación de datos y comunicación máquina-máquina. Cuenta con comunicación *CompoBus/S, DeviceNet, EtherNet, ModBus* y *PROFIBUS-DP Serie*, se programa por medio del software *CX-ONE* empleando bloques de función o lenguaje de programación *Ladder*. Precio medio ***300.08 €*** (*p145_cp2e_datasheet_en.pdf*, s. f.) (Figura 7).

Figura 7. PLC CP2E de la marca Omron, 8 entradas tipo Digital, 6 salidas tipo Relé, comunicación Ethernet.

Fuente: CP2E | OMRON, España

- ***SIMATIC S7-1200***: Es un *PLC* de la marca *Siemens* que da soluciones automatizadas compactas con opciones de comunicación ampliadas, como por ejemplo *OPC – UA,* conectividad con una nube nativa de la casa *Siemens MindSphere*, admite módulos de expansión para diferentes protocolos de comunicación, así como tarjetas de adquisición de datos. Se programa a través del software *TiaPortal* empleando lenguajes de programación *Ladder*, diagrama de funciones y *SCL*. Precio medio ***280.00 €*** (*SIMATIC S7-1200 -Take Control of Communication*, s. f.) (Figura 8).

Figura 8. PLC SIMATIC S7-1200 de Siemens, E/S INTEGRADAS: 6 DI 24 V DC; 4 DO 24 V DC; 2 AI 0-10V DC.

Fuente: SIMATIC S7-1200 | SIMATIC Controllers | Siemens Global

- ***Modicon M241***: Es un controlador de la marca *Schneider Electric* con 14 y 10 *DI/DO* respectivamente, con un puerto *Ethernet* sobre el cual se pueden implementar los siguientes protocolos de comunicación, *Cliente DHCP*, *Cliente/servidor FTP*, *Modbus TCP cliente E/S escáner*, *WebServer*, *OPC UA server*, entre otros. Se programa con el software *SoMachine* y soporta lenguaje *Ladder*, lista de instrucciones y *SFC*. Precio medio ***351.33 €*** (*TM241CE24T.pdf*, s. f.) (Figura 9)

Figura 9. PLC Modicon TM241 de Schneider Electric.

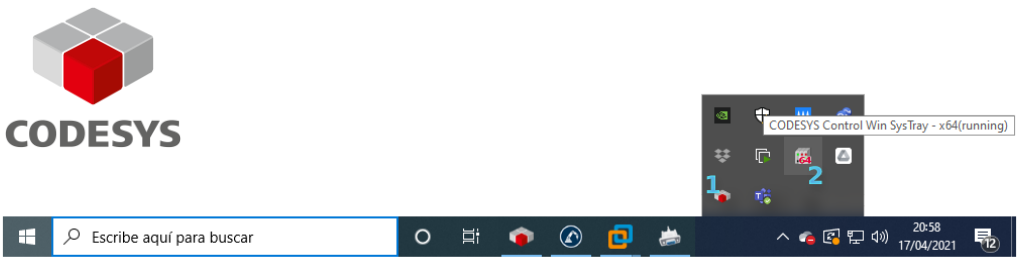
Fuente: Modicon M241 | Schneider Electric Global (se.com)

- ***softPLC***: El software ***CODESYS*** (Figura 10) de la empresa alemana *3S-Smart Software Solutions GMBH* es un entorno de desarrollo para la programación de controladores conforme con el estándar industrial internacional IEC 61131-3, abarca 6 lenguajes de programación diferentes:

1. Lista de instrucciones. (*IL*)
2. Texto estructurado. (*ST*)
3. Diagrama de escalera o de contacto. (*LD*)
4. Diagrama de bloques de funciones. (*FBD*)
5. Bloques de funciones secuenciales. (*SFC*)
6. Cuadro de funciones continuas. (*CFC*)

Se utiliza para programar *PLCs* de *Bosch Rexroth*, *FESTO*, *Beckhoff*, *Schneider Electric*, *IFM* o *Mitsubishi*. *Codesys* tiene dos *Servicios* que se ejecutan en *Windows* una vez instalado el programa en un ordenador con las siguientes denominaciones, *Codesys Gateway SysTray* y *Codesys Control Win SysTray* estos programas son necesarios para simular un controlador y de esta forma llevar a cabo al virtualización del componente, existe un paquete instalable desde la tienda de *Codesys denominado CONTROL WIN SL* (*CODESYS Store International - CODESYS Control Win SL*, s. f.) el cual permite programar un *PLC* virtual sobre *Windows* sin tiempo límite de ejecución con una licencia de ***420.00 €***, sin embargo, empleando los servicios mencionados, el *SoftPLC* permite emular un controlador con todas las característica necesarias por un tiempo superior a una hora y media, con un coste nulo.

Figura 10. Logo de Codesys. Servicios que permiten simular un controlador en el PC.
***1** - Codesys Gateway SysTray.* ***2** - Codesys Control Win SysTray V3.*



4.2. ROBOT COLABORATIVO

Un robot colaborativo, o *COBOT*, es un robot industrial que se caracteriza por ser de menores dimensiones que sus predecesores, han sido creados para automatizar procesos de producción repetitivos compartiendo el entorno de trabajo codo con codo con operarios humanos. Poseen un grupo de ventajas sobre los robots convencionales como, por ejemplo: tienen un precio más económico que se amortiza con mayor rapidez, la instalación y puesta en marcha de estos dispositivos es sumamente sencilla y rápida debido a la ligereza de todos sus componentes, tienen una gran capacidad de adaptación a diferentes tareas y sobre todo ofrecen total seguridad al contar con sensores de detección de fuerza en sus ejes que evitan golpes con una mayor inercia.

Este el componente sobre el cual recae el mayor grado de exigencia del proyecto es en sí mismo el determinante para lograr con éxito el objetivo del trabajo, por ello se analizan varias propuestas:

- ***YuMi***: Robot colaborativo de la marca sueca *ABB* con una carga máxima de trabajo por brazo de 0.5 *kg*, un alcance 559 *mm* y una exactitud de 0.02 *mm*, es un robot ideal con una alta precisión y velocidad de trabajo colaborativo, control de movimiento de última generación y cuenta con dos brazos independientes de 7 ejes cada uno (Figura 11).

Figura 11. Robot colaborativo Yumi con dos brazos articulados de 7 ejes de la casa ABB.

Fuente: IRB 14000 YUMI - ROBOT COLABORATIVO - Robots colaborativos | ABB

Todos los robots de *ABB* se programan utilizando el software *RobotStudio* de la misma marca, el mismo cuenta con un período de evaluación gratuito de un mes. El impedimento fundamental con esta marca y en general con el *IDE* son las barreras para comunicar las tarjetas de *I/O* configuradas en el interior de la controladora del robot con un s*oftPLC*, hecho que hace descartar inmediatamente esta opción. (*Datos IRB 14000 YuMi - IRB 14000 YUMI - ROBOT COLABORATIVO (Robots colaborativos) | ABB*, s. f.).

- ***LBR iiwa***: "*LBR es la abreviatura utilizada para robot de estructura liviana, mientras que iiwa es la abreviatura de intelligent industrial work assistant*" (*LBR iiwa*, s. f.). Es el cobot por excelencia de la marca alemana *KUKA* con dos gamas bien definidas en cuanto al payload que soportan, 7 y 14 *Kg* con un alcance de 800 y 820 *mm* respectivamente. Posee un tiempo de

reacción mínimo gracias a sus sensores de esfuerzo articulado, incorporan una gran capacidad de aprendizaje, es muy intuitivo y brinda una autonomía que facilita las tareas de puesta en servicio incluso en caso de casos complejos (Figura 12).

Figura 12. Cobot LBR iiwa de la marca KUKA.

Fuente: LBR iiwa | KUKA AG

Todos los sistemas colaborativos de *KUKA* se programan bajo la licencia del software *KUKA Sunrise.OS* junto con otros productos *Mobility KUKA*, el cual facilita la totalidad funcional necesarias para el servicio de los robots de estructura liviana. *"Junto con el software KUKA Sunrise.Workbench, que es un nuevo sistema de ingeniería offline, el software KUKA Sunrise.OS ofrece innovadoras funciones para la programación, la planificación y el diseño de avanzadas aplicaciones para robots."* (*KUKA Sunrise.OS*, s. f.).

KUKA Sunrise.OS es un modelo de programación gráfica soportado por una consola virtual (*smartPAD*) que en conjunto con código *JAVA* en los llamados bloques también permite a los planificadores representar los procesos sin tener conocimientos de programación. La programación gráfica ofrece una serie de ventajas, como son un lenguaje estructurado y estandarizado que puede reutilizarse en todo el proceso de programación, planificación, mantenimiento, etc., siendo además, eficiente y escalable. El paquete admite las siguientes extensiones:

- ***KUKA Sunrise.Profinet M/S***: Ofrece al robot la posibilidad de comunicarse a través de *PROFINET* tanto con los equipos conectados como con el *PLC*, o mejor dicho, el *PLC* protegido que incorporan todos los robots de la marca en el armario de control.

- **KUKA Sunrise.Servoing** es una interfaz de software en tiempo real que facilita la integración de sensores a nivel de la aplicación del controlador Sunrise en Java.

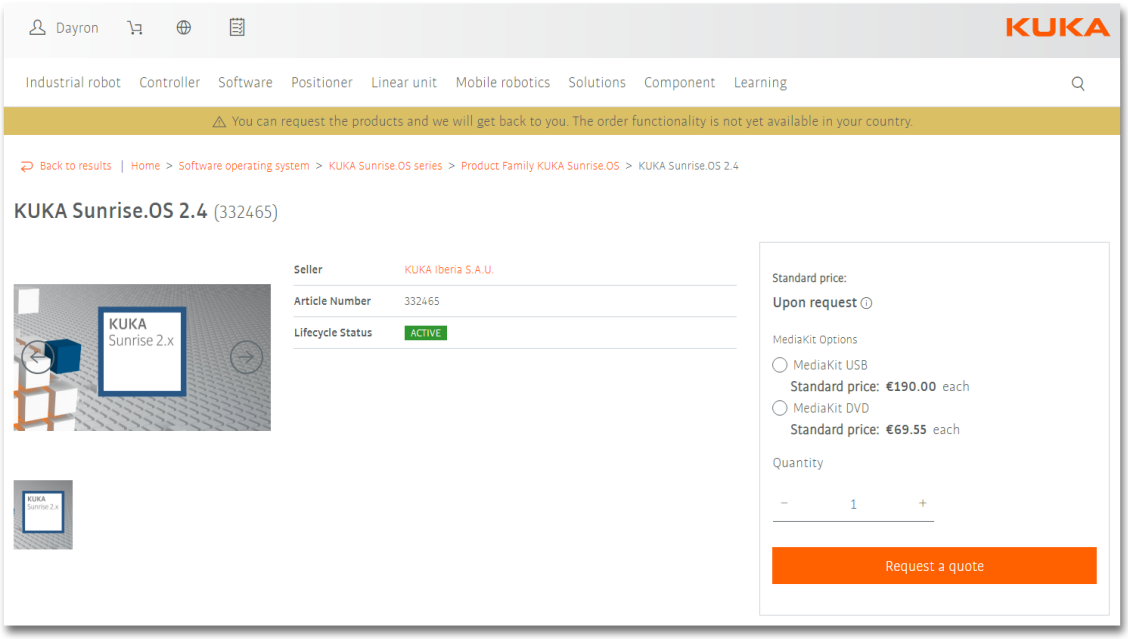
- **KUKA Sunrise.FRI** es una interfaz de robot rápido que permite el acceso al controlador del robot desde un ordenador externo en tiempo real.

- Con **KUKA Sunrise.SafeOperation** también dispondrá de las funciones "*Safe velocity monitoring*" y "*Safe workspaces and protected spaces*" en el nivel de rendimiento d y en la categoría 3 según **EN ISO 13849-1:2008**.

- **KUKA Sunrise.HRC** activa las siguientes funciones que permiten una colaboración entre hombre y robot en el nivel de rendimiento d y la categoría 3 según **EN ISO 13849-1:2008**:
  - Supervisión de fuerza y de par segura.
  - Detección de colisiones segura.
  - Control de velocidad seguro.

La desventaja de todos los componentes de la marca es el precio de acceso al software que permite la programación de robots, la (Figura 13) muestra el valor de una licencia para *Kuka Sunrise.OS versión 2.4* por *69.55 €*, no existiendo la posibilidad de disponer de un período de evaluación para nuestro país o el tipo de usuario.

Figura 13. Adquisición de una licencia para el software de programación de un cobot de Kuka desde la página oficial.



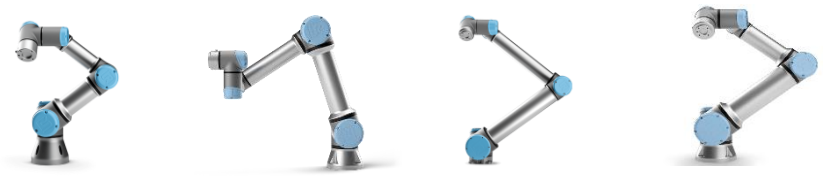
Fuente: KUKA | Marketplace

- ***URs***: Son los robots colaborativos más difundidos en el tejido industrial mundial. La empresa *Universal Robots* se funda en el año 2005 en *Dinamarca* con un objetivo en mente, cambiar la estructura y tendencia de robots pesados, costosos, grandes y poco adaptables por un nuevo modelo más versátil y accesible a nuevas empresas y negocios. En el año 2008 la compañía lanza el primer dispositivo al mercado denominado *UR5,* en 2012 se comercializa el modelo *UR10* y dos años más tarde en la feria de automática de *Munich* se presenta en benjamín de la casa, el *UR3*. En el mismo espacio (2018) se presenta al mundo la gama de robots ***eSerie*** (“*la ‘e’ de e-Series significa empowering (empoderamiento), ease of use (fácil de usar), everyone (para todos) y evolution (evolución)”)* (*es-e-series-brochure.pdf*, s. f.).

El último modelo en presentarse llegó en septiembre del 2019, el *UR16* capaz de manejar cargas de hasta 16 *kg* en un espacio reducido.

La (Tabla 1) muestra una comparativa entre los distintos modelos de la marca y sus principales características.

Tabla 1. Comparativa de los modelos de la gama eSerie de Universal Robots.



Especificación	UR3e	UR5e	UR10e	UR16e
Carga útil	3 *kg*	5 *kg*	10 *kg*	16 *kg*
Alcance	500 *mm*	850 *mm*	1300 *mm*	900 *mm*
Huella	Ø 128 *mm*	Ø 149 *mm*	Ø 190 *mm*	Ø 190 *mm*
Resolución	0.02 *mm*	0.04 *mm*	0.02 *mm*	0.20 *mm*
Precisión	0.10 *mm*	0.30 *mm*	0.60 *mm*	0.50 *mm*
GdL	6	6	6	6
Velocidad - TCP	1 *m/s*	1 *m/s*	1 *m/s*	1 *m/s*
DI – DO – AI – AO	2 – 2 – 2 – 0	2 – 2 – 2 – 0	16 – 16 – 2 – 2	16 – 16 – 2 – 2
Peso	11.2 *kg*	20.6 *kg*	33.5 *kg*	33.1 *kg*

La marca ofrece todo un conjunto de herramientas (Figura 14) para que un usuario sin ninguna experiencia previa adquiera los conocimientos necesarios en aras de programar sus robots utilizando la propia página web, dotándola de un valor añadido que no posee ninguna compañía fabricante de robots colaborativos del mundo.

*Figura 14. Formación ONLINE gratuita de Universal Robot disponible en el apartado **Academy** en su página web.*

Fuente: Formación en línea de e-Series (universal-robots.com)

La compañía pone a disposición de cualquier usuario, de forma gratuita y sin tiempo de evaluación, una máquina virtual bajo el entorno *Linux* con denominación ***URSim***, es un software de simulación que se utiliza para la programación offline y la simulación de programas de los cuatro robots de la marca. Es necesario instalar *Oracle VM* para cargar el disco virtual y ejecutar el software de programación llamado ***Polyscope***.
(ursim_vmoracle_installation_guide_v3_es.pdf, s. f.)

4.3. CLOUD COMPUTING

Actualmente, la posibilidad de poder conectar la mayor parte de dispositivos a la red a través de *IoT* hace posible la recolección de una infinita cantidad de datos para ser procesados y tratados posteriormente con el objetivo de tener resultados que permitan obtener un ahorro en costes, consumo de materiales, energía o la identificación de nuevas necesidades en los clientes.

Los servicios *cloud* o *cloud computing* permiten la obtención de infraestructuras de computación de datos de forma deslocalizada mediante *Internet*, además la posibilidad de procesar dicho volumen de información, elaborar paneles de visualización de estos e interactuar conectando elementos accionadores virtuales con señales físicas en el mundo real. Debido a la gran cantidad de datos la potencia para analizarlos de un usuario o negocio no puede ser suficiente y de este modo se liberan recursos internos contratando capacidad de computación en la *Nube*, permitiendo a sus empleados poder acceder a ellas desde cualquier dispositivo en cualquier lugar y a cualquier horario.

"Tradicionalmente, las pequeñas y medianas empresas (PYME) tenían que realizar una alta inversión de capital por adelantado para la adquisición de infraestructura de TI, desarrolladores cualificados y administradores de sistemas, lo que se traduce en un alto costo de propiedad. La computación en la Nube tiene como objetivo ofrecer una red de servicios virtuales para que los usuarios puedan acceder a ellos desde cualquier parte del mundo por suscripción a costos competitivos dependiendo de sus requisitos de calidad de servicio (QoS)" (Garg et al., 2013).

4.3.1. SERVICIOS

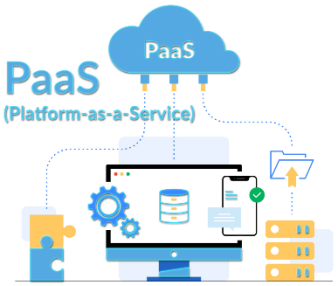
Existen tres servicios principales proporcionados por la arquitectura de computación en la nube de acuerdo con las necesidades de los clientes:

- ***SaaS***: es un modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación (*TIC*), a los que se accede vía *Internet* desde un cliente. (Figura 15).

Figura 15. Servicio SaaS en la nube.

— ***PaaS***: Un proveedor de servicios ofrece acceso a un entorno basado en cloud en el cual los usuarios pueden crear y distribuir aplicaciones. El proveedor proporciona la infraestructura subyacente. (Figura 16).

Figura 16. Servicio PaaS en la nube.



— ***IaaS***: Se plantean las infraestructuras como servicio. Las empresas contratan la infraestructura de hardware a un tercero a cambio de una cuota o alquiler. La contratación de este hardware permite elegir la capacidad de proceso (procesadores), la memoria a utilizar (memoria *RAM*) y el espacio de almacenamiento (disco duro). *IaaS* ofrece también servicios de virtualización como máquinas virtuales, cortafuegos, sistemas de backups o balanceadores de carga. (Figura 17)

Figura 17. Servicio IaaS en la nube.

4.3.2. PROVEEDORES

Según los tres tipos analizados de servicios en la nube, se describen a continuación varios proveedores que ofrecen sus productos para dar solución a la creciente demanda de almacenamiento y procesado de datos:

- ***Amazon Web Services:*** *AWS IoT* es una plataforma en la nube administrada que permite que los dispositivos conectados interactúen de manera fácil y segura con aplicaciones en la *Nube* y otros dispositivos. *AWS IoT* puede admitir miles de millones de dispositivos y billones de mensajes, y puede procesar y enrutar esos mensajes a *AWS Endpoints* y a otros dispositivos de forma fiable y segura. Con *AWS IoT*, las aplicaciones de las empresas pueden hacer un seguimiento y comunicarse con todos sus dispositivos, todo el tiempo, incluso cuando no están conectados. *AWS IoT Device Gateway* permite que los dispositivos se comuniquen de forma segura y eficiente con *AWS IoT*. *Device Gateway* puede intercambiar mensajes utilizando un modelo de publicación/suscripción, que permite comunicaciones de uno a uno y de uno a varios. Con este patrón de comunicación de uno a varios, *AWS IoT* hace posible que un dispositivo conectado difunda datos a varios suscriptores para un tema determinado. *Device Gateway* admite los protocolos *MQTT* y *HTTP 1.1* y puede implementar fácilmente la compatibilidad con protocolos propietarios o heredados. *Device Gateway,* además, se escala automáticamente para soportar más de mil millones de dispositivos sin necesidad de aprovisionar infraestructura. (Figura 18).

Figura 18. Proveedor de servicios cloud computing ***Amazon Web Services.***

Fuente: Capa gratuita de AWS | Cloud computing gratis |AWS (amazon.com)

- ***Microsoft Azure (Microsoft):*** *Azure* es compatible con la más amplia selección de sistemas operativos, lenguajes de programación, marcos de trabajo, herramientas, bases de datos y dispositivos. Permite ejecutar contenedores *Linux* con integración *Docker*; crea aplicaciones con *JavaScript*, *Python*, *.NET*, *PHP*, *Java* y *Node.js*; además incluye back-ends para dispositivos *iOS*, *Android* y *Windows*. El servicio en la *Nube* de *Azure* es compatible con las mismas tecnologías en las que ya confían millones de desarrolladores y profesionales de *TI*. Algunos proveedores de *Nube* le hacen elegir entre su centro de datos y la *Nube*. No es el caso de *Azure*, que se integra fácilmente con su entorno de *TI* existente a través de la mayor red de conexiones privadas seguras, soluciones híbridas de bases de datos y almacenamiento, y funciones de residencia y cifrado de datos, para que sus activos permanezcan justo donde los necesita. Con *Azure Stack*, se puede llevar el modelo *Azure* de desarrollo y despliegue de aplicaciones a su centro de datos. Las soluciones de *Nube* híbrida de *Microsoft* ofrecen lo mejor de ambos mundos: más opciones de *TI* y menos complejidad y costes. Por eso es uno de los mejores servicios de *Cloud Computing* disponibles. (Figura 19).

Figura 19. Proveedor de servicios cloud computing ***Microsoft Azure.***

Fuente: Servicios de informática en la nube | Microsoft Azure

- ***UBIDOTS***: Es una plataforma de desarrollo de aplicaciones *IoT* que automatiza el proceso de creación de aplicaciones basadas en internet para que las empresas y los particulares puedan desplegar cualquier solución *IoT* a escala, y hacerlo rápidamente. La plataforma es un constructor de aplicaciones *IoT* de enlace de nodos centrado en el usuario, con herramientas de análisis de datos y funciones en la *Nube*, visualizaciones de cuadros de mando, herramientas de gestión de dispositivos, eventos de *BI* y motor de alarmas, y autenticación/acceso del usuario final para dar a los usuarios y operadores los datos que necesitan y nada más. Con la plataforma *UBIDOTS*, los usuarios recopilan, mejoran y entregan datos de sensores, actuadores y balizas que son importantes para las empresas y los usuarios para tomar decisiones basadas en datos que mejoran la eficiencia y la eficacia (Figura 20). La plataforma está enfocada principalmente para dar solución a los siguientes campos:

  - *Supervisión y optimización de activos.*
  - *Seguimiento de activos.*
  - *Monitorización de la cadena de frío y alarmas.*
  - *Supervisión del medio ambiente.*
  - *Monitorización y gestión de la energía.*
  - *Automatización industrial.*
  - *Control y supervisión de la irrigación.*
  - *Gestión y supervisión de inventarios.*
  - *Control y seguimiento del ganado.*
  - *Agricultura de precisión y AgTech.*
  - *Monitorización de la contaminación y control de la calidad del aire.*
  - *Control y gestión del agua.*
  - *Sistemas meteorológicos y alarma.*

*Figura 20. Proveedor de servicios de cloud computing **UBIDOTS**.*

Fuente: IoT platform | Internet of Things | Ubidots

UBIDOTS ofrece un *mes gratuito* para implementar proyectos de integración *IoT* con fines de investigación, educación o uso personal; eligiendo un tipo de cuenta ***FOR BUSSINES*** (Figura 21.a), solo es necesario un correo electrónico, pudiéndose utilizar el de la universidad para obtener una cuenta en la plataforma (Figura 21.b).

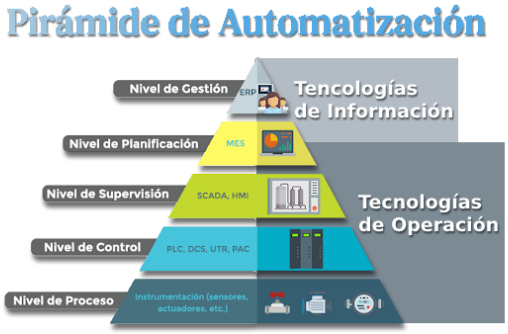
Figura 21. a) Selección del tipo de cuenta en la plataforma. b) Acceso al servicio y proceso de alta.

Fuente: IoT platform | Internet of Things | Ubidots

4.4. DISPOSITIVOS DE INTERFAZ GRÁFICA

Todos los datos procesados deben ser analizados y posteriormente representados en algún soporte físico o virtual que permita al operario interactuar con el proceso. Estos dispositivos se encuentran fundamentalmente en el tercer nivel de la pirámide de automatización industrial (Figura 22), aunque en el presente proyecto se abordan métodos de implementación de representación de datos asociados al cuarto y quinto escalón correspondientes con los niveles de planificación y gestión de la producción respectivamente.

Figura 22. Pirámide de automatización industrial con representación de los distintos niveles del proceso de producción basados en los dispositivos con los que se relacionan.



Fuente: >> Pirámide CIM de Automatización Industrial - ATEC ENERGY BLOG (atec-energy.com)

4.4.1. WEBSERVER

Los controladores lógicos programables se utilizan ampliamente en control industrial hoy en día porque son baratos, fáciles de instalar y son muy flexibles en sus aplicaciones. Un *PLC* interactúa con el mundo exterior a través de sus entradas y salidas. En el pasado, la supervisión y el control remotos de sistemas y procesos industriales tenían una aplicación limitada porque el sistema de control no era accesible desde varios lugares. Los módems permitían acceder al sistema de control desde a los sistemas de control desde distintos lugares, pero se limitan a la descarga y carga de archivos de datos y requieren una interfaz personalizada para acceder al sistema de control. La incorporación del servidor *Web* en el *PLC* garantiza el flujo oportuno de información desde la planta de producción. Al mismo tiempo, la interfaz *Ethernet* integrada permite que esta información se comparta fácilmente entre las empresas para una toma de decisiones más rápida.

"*Los PLC habilitados para la Web (Figura 23) pueden cambiar significativamente la forma en que las plantas reducen el tiempo de inactividad, se comunican de forma proactiva y aumentan la productividad*." (Mahato et al., 2015).

Figura 23. WebServer alojado dentro de un PLC ***Phoenix Contact.***

Fuente: How to access the PLCnext web server (HTTP) remotely using IXON Cloud (plcnext-community.net)

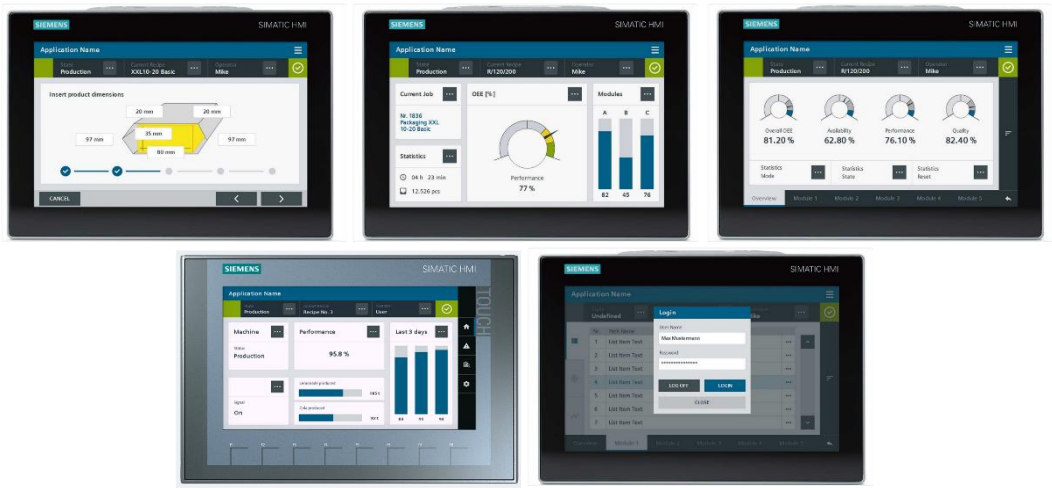
Este servicio agregado a la cadena de valor del *PLC* se corresponde con los niveles 4 y 5 de la pirámide de automatización, ya que permite, con el análisis y manejo de datos recogidos en todo el proceso productivo, tomar decisiones que afecten de forma directa la planificación y gestión de la empresa o la forma de obtener un mayor beneficio de los productos o servicios generados.

4.4.2. HMI

Los dispositivos de interfaz humana (*HMI)* son por excelencia los componentes de interacción fundamentales con los cuales los operarios y supervisores se relacionan con el proceso controlado y coordinan los procesos industriales y de fabricación.

La función principal de los *HMI* es mostrar información en tiempo real, proporcionar gráficos visuales y digeribles que aporten significado y contexto sobre el estado de algún dispositivo, sensores, alarmas y demás parámetros de un determinado proceso (Figura 24).

Figura 24. Distintas pantallas de un proceso automatizado empleando HMI Siemens.



Fuente: HMI Template Suite | Operator Control and Monitoring Systems | Siemens Global

La mayoría de los fabricantes de *PLC* tienen en su catálogo un número interesante de estos dispositivos para dar soporte a la visualización del proceso y a la interacción hombre-máquina antes mencionada, se quieren destacar algunos fabricantes de pantallas, como son:

- ***Siemens***
- ***Allen Bradley***
- ***Omron***
- ***Schneider Electric***
- ***Delta***
- ***Hytech***

4.5. SIMULADORES GRÁFICOS

El desarrollo de la capacidad de cómputo permite que las nuevas tarjetas gráficas procesen elementos digitales cada vez con mayor eficiencia y rapidez. Uno de los conceptos que más se ha desarrollado en la industria en la actualidad es la capacidad de simular componentes para detectar, programar, analizar o predecir futuros comportamientos de estos elementos, en el mundo real.

Los robots son dispositivos de automatización que no se han quedado rezagados en esta carrera, más bien todo lo contrario, han sido motor impulsor de esta rama de la ingeniería. Actualmente es imprescindible poder simular cualquier robot empleando *Robot Operating System* (ROS)[4] y una *Graphic User Interface* (*GUI*)[5].

Todos los fabricantes tienen en catálogo estas *GUI* como se vió anteriormente en el apartado *ROBOT COLABORATIVO*, sin embargo, como se comentó, el acceso a dichas herramientas es costoso, es por ello por lo que cada vez es más frecuente que empresas dedicadas a ingeniería de software desarrollen aplicaciones que aglutinen varios componentes de distintos fabricantes bajo un mismo entorno y emplean lenguajes de programación como *JAVA¸ C++, Python*, entre otros, para simular el comportamiento de los robot, incluso llegando a obtener los programas totalmente disponibles para cargar en la controladora real.

Precisamente el autor del presente proyecto quiere comentar una muestra de estas herramientas con sus respectivas características por si el lector considera que modificando parámetros de su proyecto puedan ser de mayor utilidad personal que el simulador finalmente seleccionado.

4.5.1. GAZEBO

Un simulador bien diseñado permite probar rápidamente algoritmos, diseñar robots, realizar pruebas de regresión y entrenar sistemas de *IA* (inteligencia artificial) utilizando escenarios realistas. *Gazebo* ofrece la posibilidad de simular de forma precisa y eficaz poblaciones de robots en complejos entornos interiores y exteriores. Tiene a su disposición un robusto motor de física, gráficos de alta calidad y cómodas interfaces gráficas y de programación. Lo mejor de todo es que *Gazebo es gratuito* y cuenta con una vibrante comunidad.

Gazebo proporciona un renderizado realista de los entornos, incluyendo iluminación, sombras y texturas de alta calidad, genera datos de sensores, opcionalmente con ruido, desde telémetros láser, cámaras 2D/3D, sensores estilo *Kinect*, sensores de contacto, fuerza-par, y más. Permite desarrollar *plugins* personalizados para el control del robot, los sensores y el entorno, estos proporcionan acceso directo a la *API* de la aplicación. Se encuentran disponibles muchos robots y en caso de que no esté disponible en la biblioteca de componentes el robot deseado, siempre se puede construir uno propio utilizando SDF (Figura 25).(*Gazebo*, s. f.)

Figura 25. GUI para la simulación de ROS. GAZEBO

[4] Se basa en una arquitectura gráfica dentro de los nodos. Estos reciben mensajes y multiplexan mensajes de sensores, control, estado, planificación y actuadores.

[5] Programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

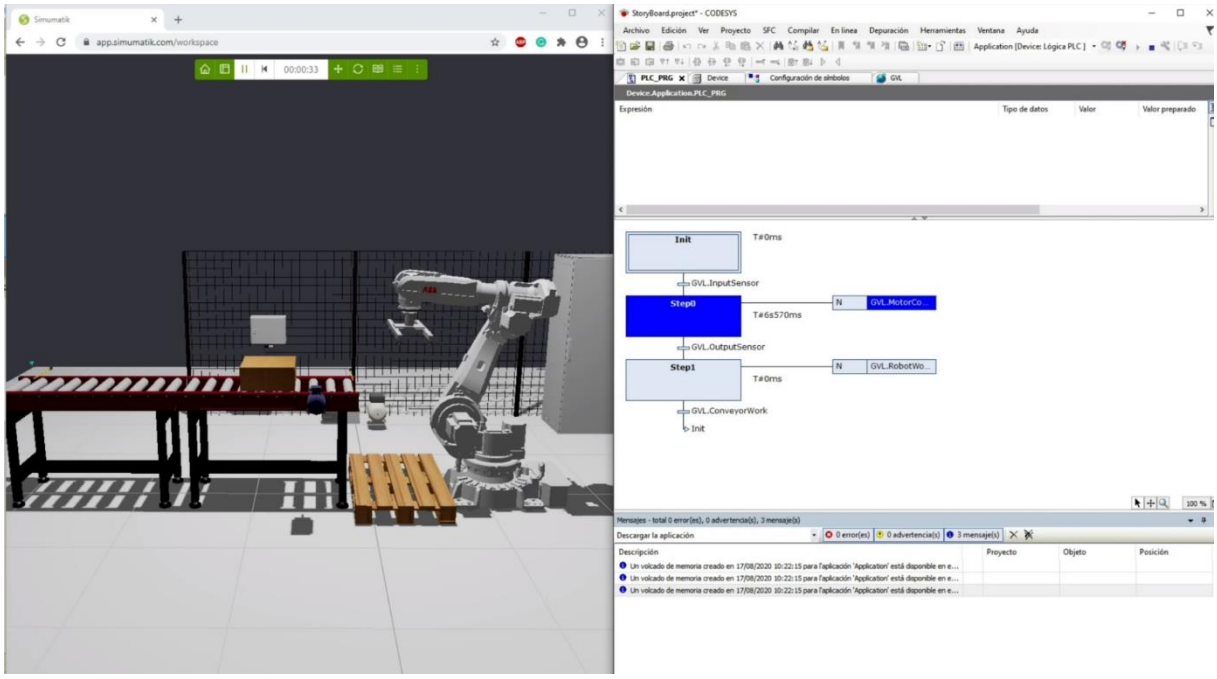
4.5.2. SIMUMATIK

Simumatik ofrece una plataforma de emulación basada en la *Nube*, dándole el poder de crear con precisión cualquier sistema o proceso que necesite. Las instituciones educativas ayudan a hacer posible esta transición, formando a los futuros técnicos e ingenieros para que dominen la nueva tecnología digital.

Por ello, *Simumatik* le ofrece una potente plataforma para ayudar a dar forma a la sostenibilidad futura en la industria y la educación. El editor de componentes de *Simumatik* ofrece a los usuarios la posibilidad de crear cualquier componente que deseen. Utilizando un modelo de datos abierto, puede modelar sin problemas los aspectos clave de un componente, incluyendo su geometría y su física empleando código *Python* para establecer el comportamiento físico y dinámico del mecanismo.

Diseñado por ingenieros, sigue un "enfoque basado en componentes" naturales, se pueden crear modelos digitales de componentes como sensores, actuadores, controladores e incluso robots, conectándolos entre sí para construir un sistema, tal y como se hace en el mundo real. («Learn Engineering Online With Simumatik's Cloud-Based Platform», s. f.).

Figura 26. Conexión de Simumatik con el IDE de Codesys para el control de una estación virtual.



Fuente: Simumatik - The Cloud Emulation Platform for Professionals & Education.

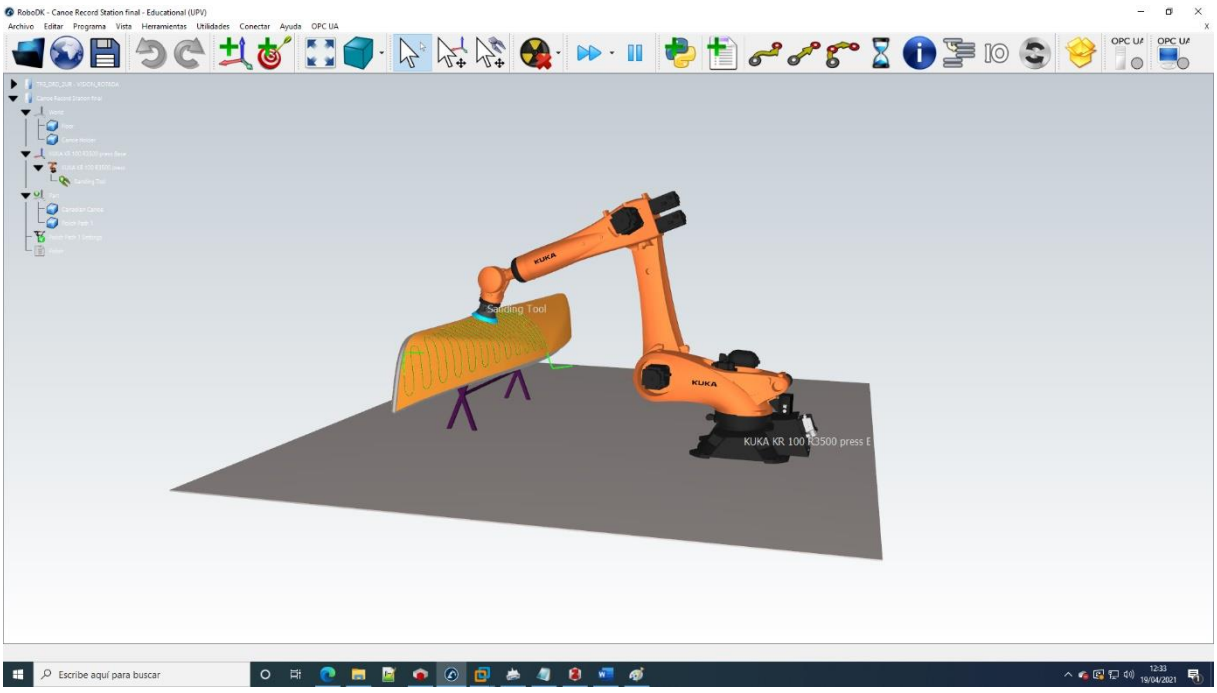
Simumatik se puede conectar a softwares de terceros y controlar mediante la programación de un *PLC*, *Controladora de Robot*, *Python* y sistemas *IoT* (Figura 26). Existe un período de prueba bajo petición vía *e-mail* de un mes y dispone de dos versiones, una dedicada al sector industrial y otra al nivel educativo, con la cual los estudiantes pueden acceder a laboratorios virtuales, crear sus propios proyectos y acceder a recursos online de forma gratuita.

4.5.3. RoboDK

Por ahora, el mejor método en la investigación en el campo de la robótica para garantizar que los algoritmos de control avanzados funcionan correctamente antes de trasladarlos a un robot real, es la llamada *programación offline* (o simulación), en la cual se implementa toda la lógica de control del mecanismo y se programan las rutinas de movimiento principales, corrigiendo cualquier colisión o desplazamiento no deseado.

RoboDK es un software desarrollado por una empresa de *Canadá* y su principal objetivo es aportar potentes capacidades de simulación y programación de robótica a todas las empresas. En la actualidad, es compatible con más de 200 robots de 50 fabricantes, como *ABB*, *Fanuc*, *KUKA*, *Yaskawa*, *Stäubli* y *Universal Robots*. El simulador es una herramienta de programación universal fuera de línea, que facilita la programación de todo tipo de robots, así como la generación de programas de robot específicos de la marca. Se utiliza habitualmente en aplicaciones industriales como el fresado con robot, la soldadura con robot, impresión 3D de pintura y calibración de robots. (Figura 27). La versión gratuita de *RobotDK* de un mes es suficiente para formar al programador con las capacidades del sistema y desarrollar soluciones de automatización complejas.

Figura 27. Interfaz gráfica de RoboDK simulando un proceso de pulido de una canoa empleando un robot Kuka.



Fuente: Archivo de ejemplo "Example-03.d-Polishing Canadian Canoe – KUKA" de RoboDK

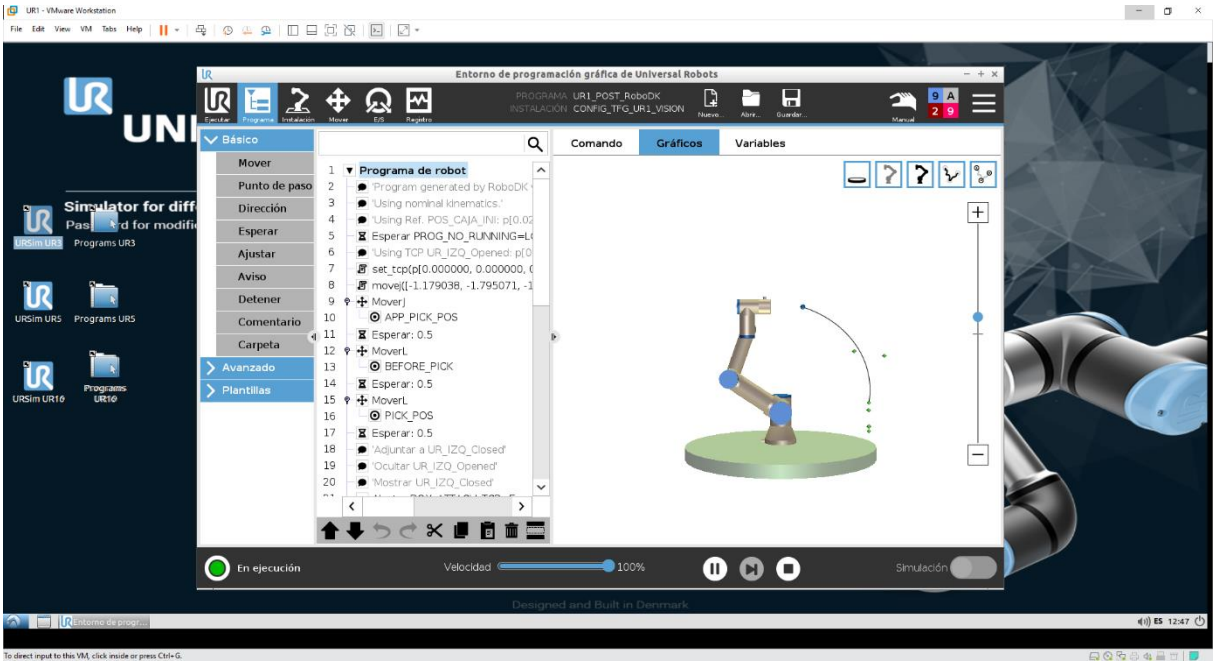
RoboDK proporciona una interfaz gráfica de usuario fácil y cómoda donde se pueden aplicar las restricciones necesarias para llevar a la realidad lo programado previamente, esta interfaz hace que la experiencia en el campo no sea necesaria necesitando solamente conocimientos básicos de elementos *CAD* y acciones de *Drag-and-Drop.* Además, al utilizar la *API* de *RoboDK* no hay límites para la simulación y la programación offline.

4.5.4. URSim

URSim es un software de simulación que se utiliza para la programación y simulación offline de programas de robots de la empresa *Universal Robots.* El simulador tiene algunas limitaciones, ya que no se conecta a un brazo robótico real lo que lleva a que el control de fuerza estará limitado en su uso.

Si se selecciona el modo de simulación en la esquina inferior derecha de la consola de programación, es posible simular las entradas digitales en la página de *E/S*. *URSim* es un software de simulación destinado a la programación fuera de línea y a la simulación tanto de los programas del robot como del movimiento manual del mismo. (Figura 28).

*Figura 28. Máquina virtual VMware Workstation ejecutando **URSim**.*



Universal Robots pone a disposición de los usuarios una imagen de disco virtual de un sistema operativo *Linux* que contiene el software que simula el *Polyscope* y los robots físicos, llamado *URSim*.

4.6. CRITERIOS DE VALORACIÓN

Todo lo mencionado con anterioridad brinda un amplio abanico de posibilidades para llevar a cabo el objetivo del presente proyecto: el *Diseño de una Celda Automatizada con Robótica Colaborativa*, por este motivo se establecen una serie de parámetros para evaluar cada familia de componentes y esgrimir aquel que mejor se adapte a los objetivos que se persiguen.

- ***Sencillez***: El elemento analizado debe ser capaz de abordarse con rapidez y cumplir con los parámetros fijados en los objetivos.

- ***Versatilidad***: El servicio o dispositivo debe ser capaz de adaptarse y ser flexible ante los requerimientos del proyecto, permitiendo incluso futuras mejoras no contempladas en el momento en el que se redacta este informe**.**

- ***Coste:*** Uno de los principales objetivos que se establecen al inicio del proyecto es no invertir económicamente un céntimo de euro para obtener una plataforma multidisciplinar y con carácter mutable que permita desarrollar investigaciones y futuros proyecto a alumnos e interesados en general en la materia. En caso de no poder acceder a herramientas gratuitas, estas deben brindar un período de evaluación mínimo de un mes con la mayoría de las prestaciones disponibles.

- ***Plan futuro:*** Se deben incluir en la plataforma herramientas que estén en pleno desarrollo y explotación en el sector industrial y formen parte de la actual revolución industrial, de esta forma tiene sentido el proyecto y se asegura que es útil durante un plazo no inferior a 3 años.

- ***Disponibilidad:*** Las herramientas deben ser fáciles de conseguir, en la medida de lo posible todo debería ser descargable de internet, en formato digital y con carácter inmediato.

Con estos parámetros, el autor presenta el resultado del conjunto de herramientas que conformarán la solución al proyecto empleando una tabla comparativa (Tabla 2).

Tabla 2. Comparativa mediante criterios de selección de la solución a adoptar.

		Sencillez	Versatilidad	Coste	Plan futuro	Disponibilidad
PLC	CP2E	●●●●○	●●●○○	○○○●●	●●●●○	●●●○●
	S7-1200	●●●○○	●●●●●	○○○○●	●●●●●	●○○○○
	M241	●●●●○	●●●●●	○○●●●	●●●●○	●●●●○
	CODESYS	●●●●●	●●●●●	○○○○○	●●●●●	●●●●●
Robot	Yumi	●●○○○	●●●●●	○○○●●	●●●○○	●●●○○
	LBR iiwa	●●●○○	●●●●●	○○●●●	●●●●○	●○○○○
	UR	●●●●○	●●●●●	○○○○○	●●●●●	●●●●●
Cloud	Amazon WS	●●●●○	●●●●●	○○○●●	●●●●●	●●●●○
	Microsoft Azure	●●●●○	●●●●●	○○○●●	●●●●○	●●●●○
	UBIDOTS	●●●●○	●●●●●	○○○○●	●●●●○	●●●●●
HMI	Siemens	●●●●○	●●●●●	○○●●●	●●●●●	●○○○○
	Allen Bradley	●●●○○	●●●●●	○○●●●	●●●●●	●○○○○
	Omron	●●●●○	●●●○○	○○○○○	●●●○○	●●●●●
	Schneider Electric	●●●●○	●●●●○	○○○○●	●●●●○	●●●●○
WebServer	CODESYS	●●●●●	●●●●●	○○○○○	●●●●●	●●●●●
Simulador	Gazebo	●●●○○	●●●●●	○○○○○	●●●●●	●●●●●
	Simumatik	●●●●○	●●●●○	○○○●●	●●●○○	●●●○○
	RoboDK	●●●●●	●●●●●	○○○○●	●●●●●	●●●●●
	URSim	●●●●●	●●●●○	○○○○○	●●●●○	●●●●●

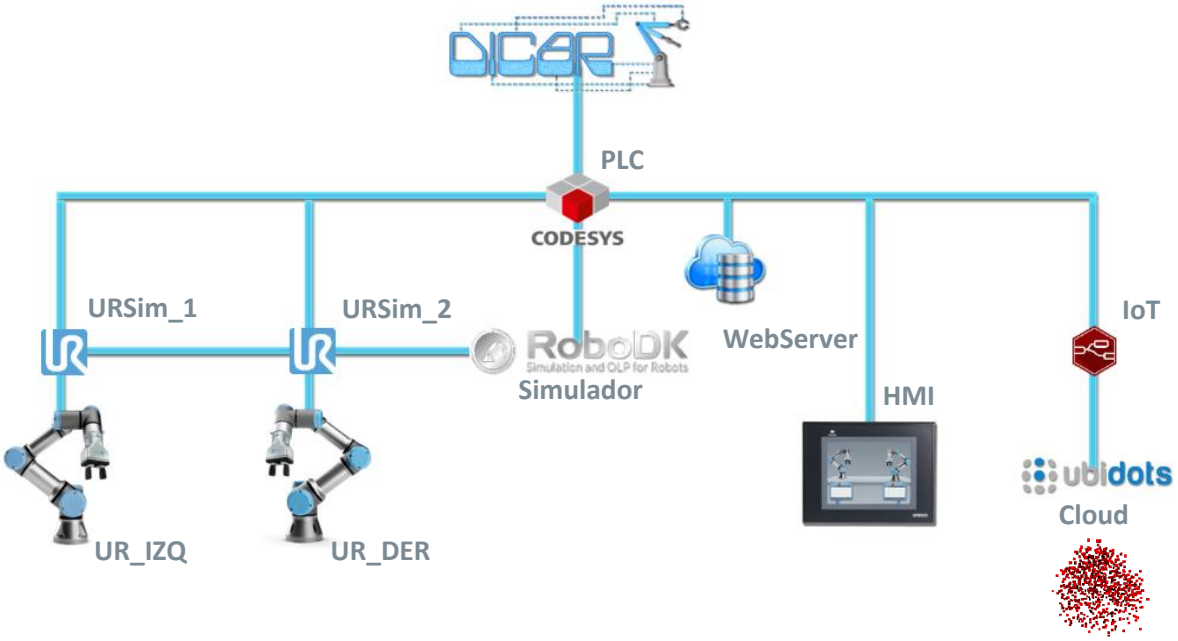
Tras lo analizado, el paquete de herramientas con las cuales se dará solución al objetivo del proyecto será:

- PLC: *SoftPLC CODESYS*
- Robot: *UNIVERSAL ROBOT UR3e*
- Cloud: *UBIDOTS*
- HMI: *OMRON*
- WebServer: *CODESYS*
- Simulador: *RoboDK*

5. DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN ADOPTADA

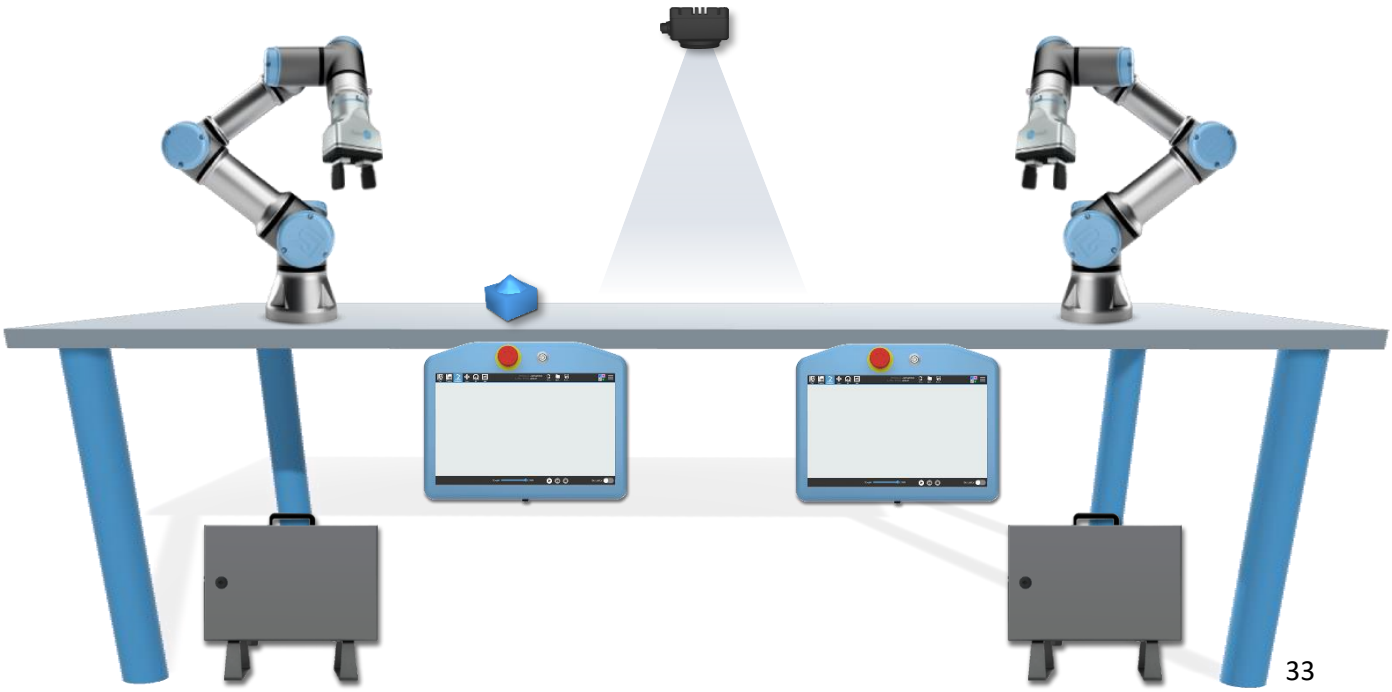
La descripción detallada de la alternativa elegida se elabora jerarquizando el proyecto a través de un organigrama (Figura 29), a su vez representa gráficamente la estructura o el conjunto de elementos o secuencias necesarios para construir un *Digital Twin* de una celda automatizada con robótica colaborativa.

Figura 29. Organigrama que da solución al Diseño e Implementación de una Celda Automatizada con Robótica Colaborativa.



En un inicio, el planteamiento de los objetivos del proyecto era implementar una celda con dos robots *UR3e* que se intercambiasen una pieza empleando visión artificial y haciendo uso de los dos robots de los que dispone el *Departamento de Ingeniería de Sistemas y Automática* de la *UPV*. (Figura 30).

Figura 30. Planteamiento original del proyecto con dispositivos físicos.



En la actual situación de pandemia sobrevenida por el virus *SARS COVID-19*, es necesario encontrar nuevos recursos que suplan la carencia del factor de presencialidad en todas las áreas de nuestra vida, la universidad no se vió exonerada de tal golpe, la mayoría de las clases pasan a ser virtuales al igual que las prácticas, exámenes, *TFG* y *TFM.*

Por este motivo el autor del presente proyecto se encuentra con un cambio radical de los objetivos y todas las exigencias que acarrea virtualizar el conjunto de herramientas físicas reales y que, hasta donde se tiene constancia, nunca se habían aglutinado en un entorno de simulación.

Después de analizar las ventajas e inconvenientes de las herramientas que se podrían disponer, surge la idea de redirigir el *TFG* enfocado en las tecnologías que conforman la actual *Industria 4.0* y diseñar un *Digital Twin* del proyecto inicial, la idea de convertir toda la plataforma en una herramienta educativa para suplir las carencias de componentes sobrevino en cuanto se plasmaron todos los bloques del diagrama presentado. En la (Tabla 3) se relacionan el número de elementos dispuestos, así como su denominación y la relación entre ellos materializada con protocolos de comunicación y códigos de programación.

Tabla 3. Relación de dispositivos y enlaces utilizados.

		Enlace	Aplicación	Tecnología
1	**PLC**	OPC-UA	RoboDK	CLIENT - SERVER
		MODBUS-TCP	URSim	CLIENT - SERVER
		ETHERNET	WebServer	TCP/IP
		OPC-UA	Node-RED	IoT
2	**UR**[6]	ETHERNET	URSim	MÁQUINA VIRTUAL
		ETHERNET	RoboDK	RTDE
1	**HMI**	MODBUS-RTU	CODESYS	RS485
1	**Cloud**	ETHERNET	CODESYS	MQQT, IoT, Node-Red, Node.js, Json

Durante el curso 2019-2020, el autor cursa la asignatura *Instalaciones de Control Industrial* del *Grado en Ingeniería Electrónica Industrial y Automática*, en la cual se reciben los conocimientos necesarios sobre protocolos de comunicaciones industriales, programación de *PLC*, redes, diseño de *HMI*, obtención de modelos matemáticos de sistemas físicos, simulación de estos y posterior control, la asignatura sirve de base para entrar en contacto con gran parte de los dispositivos que se emplean y que a continuación, se describen siguiendo la estructura jerarquizada superior.

[6] Para una mejor identificación de los dispositivos en el proyecto se ha denominado *UR_IZQ ó UR1* al robot que se encuentra en el lado izquierdo de la mesa y *UR_DER ó UR2* al robot posicionado a la derecha en la mesa de trabajo.

5.1. PLC softPLC - CODESYS

CODESYS es el software elegido para implementar la lógica de control, es además el núcleo del proyecto pues del *PLC* parten o arriban todas las comunicaciones necesarias para desarrollar la plataforma, se emplea la versión ***V3.5 SP16 Patch 4+ (64-bit)*** (Figura 31).

Figura 31. Versión del software utilizada para programar el PLC.

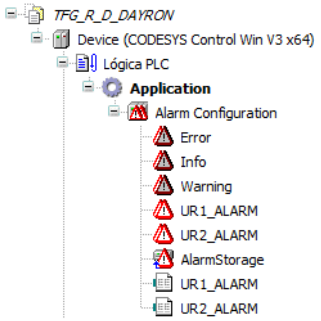
La organización del proyecto está enlazada a un *PLC virtual* denominado *Device (CODESYS Control WIN V3 x64)* el cual está ligado al servicio instalado en el *PC,* mencionado en el apartado *4.1. softPLC*, dicho servicio será el encargado de simular un controlador lógico virtual.

Toda la lógica de control en el *PLC* se implementa en programas denominados *POU*, un objeto del tipo *POU* es una *Unidad de Organización de Programa* en un proyecto *CODESYS*. Para agregar un *POU* se debe indicar que tipo de objeto será (*Programa, Función o Bloque de Función*) y que lenguaje se utilizará en su definición.

Al *PLC* se enlazan un conjunto de elementos que se pasan a describir a continuación para mayor comprensión y justificación:

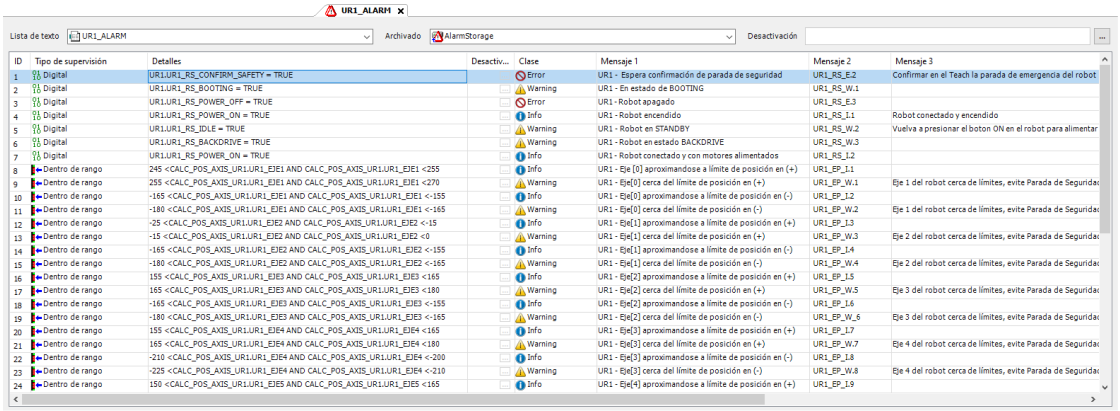
- ***Alarm Configuration***: Objeto preparado para la configuración de todas las alarmas necesarias en el proyecto, se incluyen tres tipos de eventos en la programación, ***Error*** (son eventos que generan paros de emergencia por superar parámetros críticos en el proceso, paradas de emergencia de algún dispositivo, así como cualquier otra señal que se desee incorporar y tenga el efecto de prioridad máxima sobre el sistema), ***Warning*** (eventos que sin llegar a provocar una parada de emergencia, advierten al usuario que se está aproximando a una situación crítica o existe un apartado que necesita revisión o comprobación), ***Info*** (eventos con carácter meramente informativos, no provocan ninguna acción que resulte correctiva ni requieren atención inmediata) (Figura 32).

Figura 32. Relación de objetos adjuntos a una configuración de alarmas en CODESYS.



Los elementos *UR1_ALARM* y *UR2_ALARM* son tablas donde se programan las distintas alarmas antes mencionadas relativas a cada robot (Figura 33); será un algoritmo general en la solución del proyecto, discriminar cada robot o pantalla en su tratamiento y control para obtener una programación segmentada y organizada.

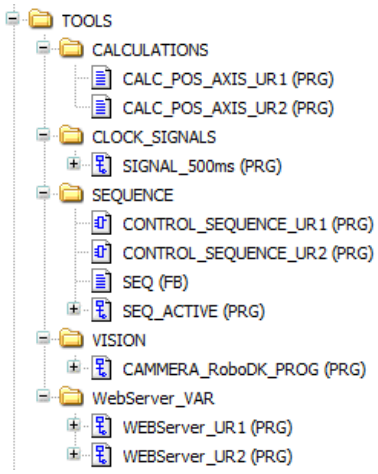
Figura 33. Tabla de configuración de todas las Alarmas, Warnings e Info relativos a UR_IZQ.



ID	Tipo de supervisión	Detalles	Desactiv...	Clase	Mensaje 1	Mensaje 2	Mensaje 3
1	Digital	UR1.UR1_RS_CONFIRM_SAFETY = TRUE		Error	UR1 - Espera confirmación de parada de seguridad	UR1_RS_E.2	Confirmar en el Teach la parada de emergencia del robot
2	Digital	UR1.UR1_RS_BOOTING = TRUE		Warning	UR1 - En estado de BOOTING	UR1_RS_W.1	
3	Digital	UR1.UR1_RS_POWER_OFF = TRUE		Error	UR1 - Robot apagado	UR1_RS_E.3	
4	Digital	UR1.UR1_RS_POWER_ON = TRUE		Info	UR1 - Robot encendido	UR1_RS_I.1	Robot conectado y encendido
5	Digital	UR1.UR1_RS_IDLE = TRUE		Warning	UR1 - Robot en STANDBY	UR1_RS_W.2	Vuelva a presionar el boton ON en el robot para alimentar
6	Digital	UR1.UR1_RS_BACKDRIVE = TRUE		Warning	UR1 - Robot en estado BACKDRIVE	UR1_RS_W.3	
7	Digital	UR1.UR1_RS_POWER_ON = TRUE		Info	UR1 - Robot conectado y con motores alimentados	UR1_RS_I.2	
8	Dentro de rango	245 <CALC_POS_AXIS_UR1.UR1_EJE1 AND CALC_POS_AXIS_UR1:UR1_EJE1 <255		Info	UR1 - Eje [0] aproximandose a límite de posición en (+)	UR1_EP_I.1	
9	Dentro de rango	255 <CALC_POS_AXIS_UR1.UR1_EJE1 AND CALC_POS_AXIS_UR1.UR1_EJE1 <270		Warning	UR1 - Eje[0] cerca del límite de posición en (+)	UR1_EP_W.1	Eje 1 del robot cerca de límites, evite Parada de Seguridad
10	Dentro de rango	-165 <CALC_POS_AXIS_UR1.UR1_EJE1 AND CALC_POS_AXIS_UR1.UR1_EJE1 <-155		Info	UR1 - Eje[0] aproximandose a límite de posición en (-)	UR1_EP_I.2	
11	Dentro de rango	-180 <CALC_POS_AXIS_UR1.UR1_EJE1 AND CALC_POS_AXIS_UR1.UR1_EJE1 <-165		Warning	UR1 - Eje[0] cerca del límite de posición en (-)	UR1_EP_W.2	Eje 1 del robot cerca de límites, evite Parada de Seguridad
12	Dentro de rango	-25 <CALC_POS_AXIS_UR1.UR1_EJE2 AND CALC_POS_AXIS_UR1.UR1_EJE2 <-15		Info	UR1 - Eje[1] aproximandose a límite de posición en (+)	UR1_EP_I.3	
13	Dentro de rango	-15 <CALC_POS_AXIS_UR1.UR1_EJE2 AND CALC_POS_AXIS_UR1.UR1_EJE2 <0		Warning	UR1 - Eje[1] cerca del límite de posición en (+)	UR1_EP_W.3	Eje 2 del robot cerca de límites, evite Parada de Seguridad
14	Dentro de rango	-165 <CALC_POS_AXIS_UR1.UR1_EJE2 AND CALC_POS_AXIS_UR1.UR1_EJE2 <-155		Info	UR1 - Eje[1] aproximandose a límite de posición en (-)	UR1_EP_I.4	
15	Dentro de rango	-180 <CALC_POS_AXIS_UR1.UR1_EJE2 AND CALC_POS_AXIS_UR1.UR1_EJE2 <-165		Warning	UR1 - Eje[1] cerca del límite de posición en (-)	UR1_EP_W.4	Eje 2 del robot cerca de límites, evite Parada de Seguridad
16	Dentro de rango	155 <CALC_POS_AXIS_UR1.UR1_EJE3 AND CALC_POS_AXIS_UR1.UR1_EJE3 <165		Info	UR1 - Eje[2] aproximandose a límite de posición en (+)	UR1_EP_I.5	
17	Dentro de rango	165 <CALC_POS_AXIS_UR1.UR1_EJE3 AND CALC_POS_AXIS_UR1.UR1_EJE3 <180		Warning	UR1 - Eje[2] cerca del límite de posición en (+)	UR1_EP_W.5	Eje 3 del robot cerca de límites, evite Parada de Seguridad
18	Dentro de rango	-165 <CALC_POS_AXIS_UR1.UR1_EJE3 AND CALC_POS_AXIS_UR1.UR1_EJE3 <-155		Info	UR1 - Eje[2] aproximandose a límite de posición en (-)	UR1_EP_I.6	
19	Dentro de rango	-180 <CALC_POS_AXIS_UR1.UR1_EJE3 AND CALC_POS_AXIS_UR1.UR1_EJE3 <-165		Warning	UR1 - Eje[2] cerca del límite de posición en (-)	UR1_EP_W_6	Eje 3 del robot cerca de límites, evite Parada de Seguridad
20	Dentro de rango	155 <CALC_POS_AXIS_UR1.UR1_EJE4 AND CALC_POS_AXIS_UR1.UR1_EJE4 <165		Info	UR1 - Eje[3] aproximandose a límite de posición en (+)	UR1_EP_I.7	
21	Dentro de rango	165 <CALC_POS_AXIS_UR1.UR1_EJE4 AND CALC_POS_AXIS_UR1.UR1_EJE4 <180		Warning	UR1 - Eje[3] cerca del límite de posición en (+)	UR1_EP_W.7	Eje 4 del robot cerca de límites, evite Parada de Seguridad
22	Dentro de rango	-210 <CALC_POS_AXIS_UR1.UR1_EJE4 AND CALC_POS_AXIS_UR1.UR1_EJE4 <-200		Info	UR1 - Eje[3] aproximandose a límite de posición en (-)	UR1_EP_I.8	
23	Dentro de rango	-225 <CALC_POS_AXIS_UR1.UR1_EJE4 AND CALC_POS_AXIS_UR1.UR1_EJE4 <-210		Warning	UR1 - Eje[3] cerca del límite de posición en (-)	UR1_EP_W.8	Eje 4 del robot cerca de límites, evite Parada de Seguridad
24	Dentro de rango	150 <CALC_POS_AXIS_UR1.UR1_EJE5 AND CALC_POS_AXIS_UR1.UR1_EJE5 <165		Info	UR1 - Eje[4] aproximandose a límite de posición en (+)	UR1_EP_I.9	

- ***TOOLS***: Correlativo a la configuración de alarmas se crea una carpeta que almacena toda la lógica de control del proyecto, a su vez en su interior, como se comentó anteriormente, existen subcarpetas que dividen las tareas llevadas a cabo según su función y el orden en que se desarrollaron. (Figura 34).

Figura 34. Organización de las carpetas que contienen la lógica de control del proyecto.



La denominación de cada carpeta, así como de todos los programas que a continuación se detallan se encuentran en *inglés* para facilitar la comprensión de los distintos apartados a los alumnos, o cualquier interesado que no domine el castellano.

NOTA: Todo el código contenido dentro de los programas se publica en el apartado *Manual de Programación*.

- ***CALCULATIONS*** contiene la primera de las herramientas desarrolladas, corresponde a un programa definido con lenguaje estructurado (*ST*), *CALC_POS_AXIS_UR1(PRG)*, en el cual se implementa el código necesario para obtener el valor de cada eje de *UR1* enviado por el servidor *Modbus* alojado dentro de cada robot en la máquina virtual correspondiente. Dentro de dicho programa se crean las variables oportunas para cada eje siguiendo la lógica:

  UR1_EJE1, ***UR1_EJE2***, ***UR1_EJE3***, ***UR1_EJE4***, ***UR1_EJE5*** y ***UR1_EJE6*** (recepcionan la posición)
  UR1_REV1, ***UR1_ REV2***, ***UR1_ REV3***, ***UR1_ REV4***, ***UR1_ REV5*** y ***UR1_ REV6*** (guardan el número de revoluciones que presenta el eje en ese momento)

  Cada variable de posición es de tipo ***REAL*** para acumular valores flotantes con signo, este tipo de datos abarca una longitud de 32 bits y se almacena bajo el estándar *IEEE 754-2008.* Para determinar el resultado de la posición del eje, dependiendo de cada uno se emplea la siguiente fórmula:

Ecuación 1. Fórmula para el cálculo de la posición del eje 1 de UR1 teniendo en cuenta si existe revolución de la articulación y expresando el resultado en grados.

```
IF UR1_REV1=0 THEN
        UR1_EJE1:=(WORD_TO_REAL(UR1.UR1_POS_EJE1))*((180)/(1000*PI));
ELSIF
        UR1.UR1_POS_EJE1=0 THEN
        UR1_EJE1:=0;
ELSE
        UR1_EJE1:=(WORD_TO_REAL(UR1.UR1_POS_EJE1))*((180)/(1000*PI))-360;
END_IF
```

El mismo proceder se ejecuta para los 6 ejes del robot, de forma análoga el programa *CALC_POS_AXIS_UR2 (PRG)* contiene iguales instrucciones para determinar la posición de todos los ejes del robot *UR2 o UR_DER.*

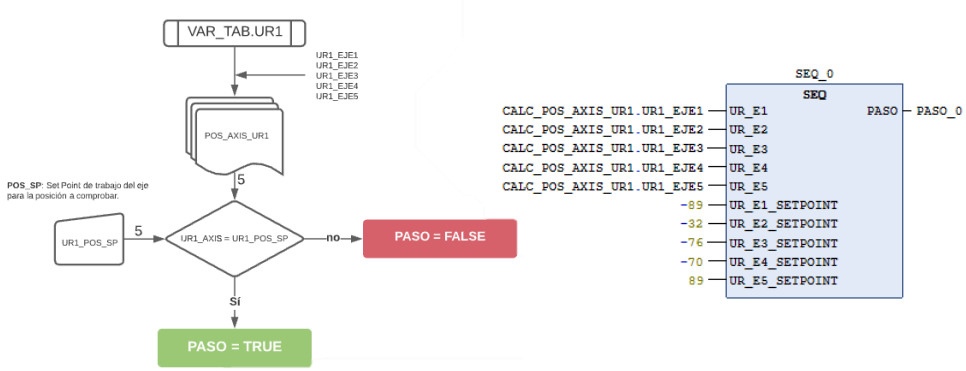
- ***CLOCK_SIGNALS*** alberga un programa en lenguaje *GRAFCET* secuencial para generar una señal alterna con un *T = 0.5 s* que se emplea para hacer parpadear determinados elementos en las pantallas, tanto del *WebServer* como en el *HMI.*

- ***SEQUENCE*** es la carpeta que contiene todos los programas necesarios para determinar en qué punto se encuentra la lógica del proyecto. La idea de control para comprobar en que posiciones se encuentran los robots es anotar el valor de cada eje de ambos robots y crear una tabla (*Adjunta en el Anexo I: Tabla de control de posiciones*) que contenga dichos valores.

— *SEQ (FB)* está diseñado en lenguaje de texto y conforma un bloque que posteriormente se instancia para comprobar, suministrando determinados valores de referencia para cada eje, si el robot se encuentra en una posición de interés.

La (Figura 35.a) representa un diagrama de flujo que describe el principio de funcionamiento del bloque y en la (Figura 35.b) se observa el bloque instanciado con valores de referencia para comprobar si el robot *UR1* se encuentra en *HOME*.

Figura 35.a) Diagrama de flujo del bloque de comprobación de una posición de interés para UR1.
b) Instancia del bloque para detectar la posición HOME.

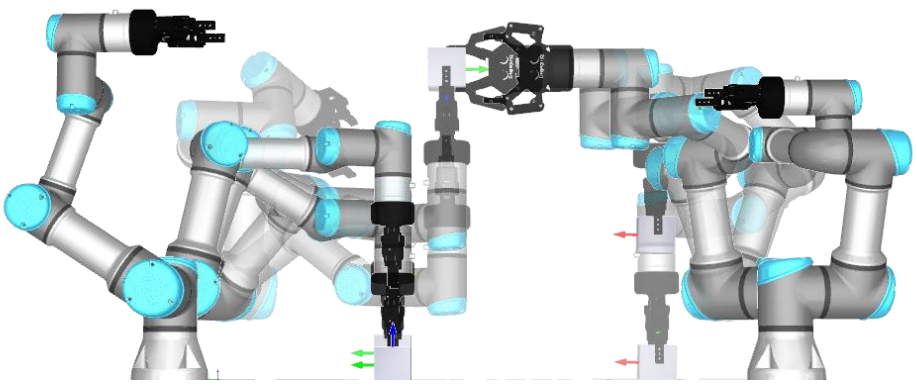


La (Tabla 4) describe las posiciones de interés para cada robot y la (Figura 36) representa dichos estados de forma superpuesta.

Tabla 4. Relación de posiciones de control para UR1 y UR2.

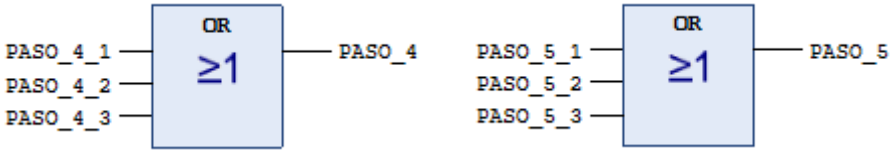
Paso	UR1	UR2
PASO_0	UR1_HOME_POS	UR2_HOME_POS
PASO_1	UR1_BEBORE_PICK_POS	UR2_APPROX_PICK_POS
PASO_2	UR1_PICK_POS	UR2_PICK_NO_BOX_POS
PASO_3	UR1_AFTER_PICK_POS	UR2_PICK_WITH_BOX_POS
PASO_4	UR1_DELIVERY_POS	UR2_AFTER_PICK_POS
PASO_5	UR1_AFTER_DELIVERY_POS	UR2_APPROX_DROP_POS
PASO_6	UR1_APPROX_HOME	UR2_DROP_POS
PASO_7	UR1_HOME_END_POS	UR2_AFTER_DROP_POS
PASO_8		UR2_HOME_END_POS

Figura 36. Representación superpuesta de las posiciones de control de ambos UR en la secuencia de trabajo.



— *CONTROL_SEQUENCE_UR1 (PRG)* es un programa en lenguaje de diagrama de bloques funcionales (*FBD*), contiene 11 instancias del bloque *SEQ*, a pesar que solamente existen 7 posiciones de interés para UR1, como la posición de entrega de la pieza puede variar en 3 posibles valores preconfigurados y definidos posteriormente en el servicio de cloud de *UBIDOTS,* es necesario tener en cuenta que el *PASO_4* y el *PASO_5* pueden tomar cada uno tres posiciones diferentes, lo que deriva en crear pasos intermedios tal como se muestran en la (Figura 37), al final de la secuencia se consulta si alguna de las tres posiciones de cada paso se encuentra activa para detectar si *UR1* se encuentra entregando la pieza o en el paso posterior.

Figura 37. Comprobación de las diferentes posiciones que activan los pasos 4 y 5 de UR1.



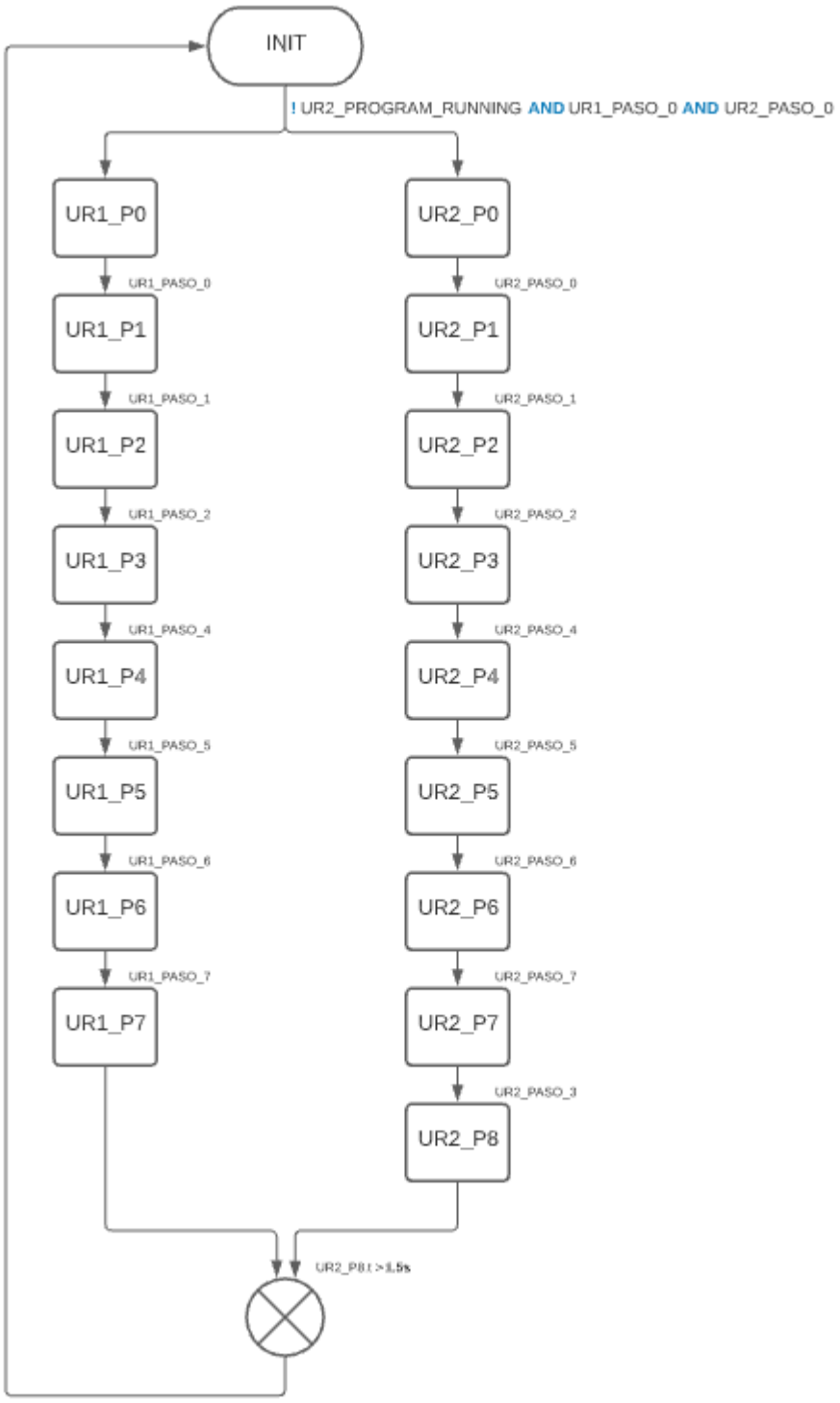
De forma análoga *CONTROL_SEQUENCE_UR2(PRG)* instancia a *SEQ* en 17 ocasiones, a pesar de tener *UR2* únicamente 8 posiciones de control, *PASO_1, PASO_2, PASO_3* y *PASO_4* se pueden activar cada uno proveniente de tres conjuntos de posiciones diferentes asociadas al punto en el cual *UR1* entrega de la pieza. Igualmente se comprueba haciendo uso de funciones *OR* si alguna de las posiciones que activa un paso se cumple para conocer si *UR2* se encuentra en algún punto de control de la secuencia.

— *SEQ_ACTIVE (PRG)* es el último programa de control de la secuencia y en el cual se implementan una serie de tareas relativas a la representación gráfica del estado del proceso, mencionando, por ejemplo, animaciones de las transiciones y etapas, la asignación de valores a determinadas variables para la carga de datos relativos a las posiciones en las que se encuentran en todo momento ambos *UR,* así como la comprobación de la recepción de las señales enlazadas al bus de comunicaciones *MODBUS versión Client,* sobre la cual las máquinas virtuales *URSim* envían *I/*O para indicar el punto de trabajo de cada robot por separado. Dichas animaciones se presentan en el *WebServer* y el *HMI*, como se describe posteriormente, ubicadas dentro de las ventanas *PROGRAM* relativas a cada *UR.*

El ciclo de trabajo de la celda es continuo, cabe recordar nuevamente que no es objetivo del proyecto diseñar una lógica de control de marchas, paradas de emergencias o de fin de ciclo, arranques, maniobras de cierre, tareas de prueba o ejecución de ciclo paso a paso.

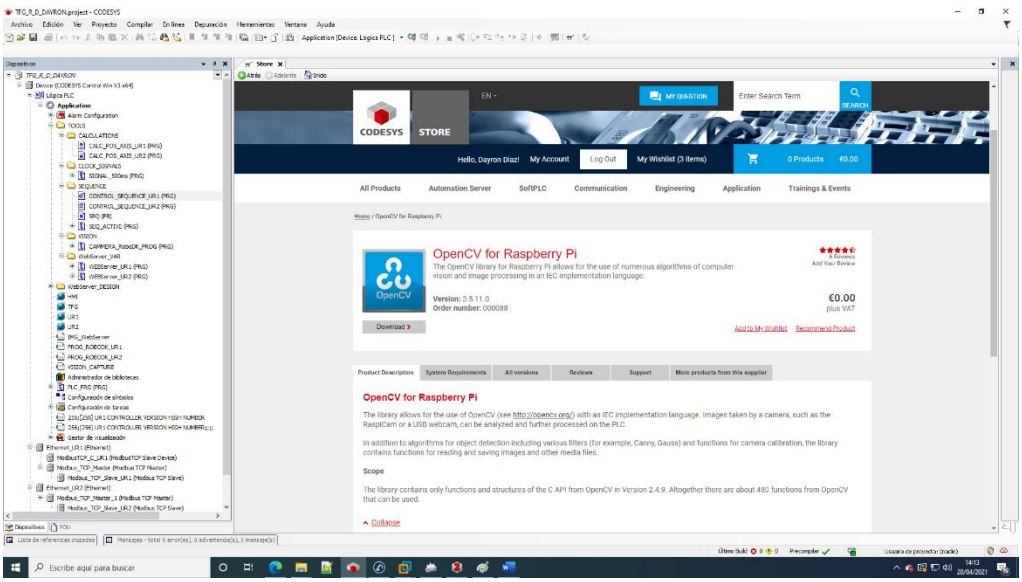
La (Figura 38) presenta un diagrama de flujo en el cual se puede comprobar, empleando un *GRAFCET* secuencial, la fase cíclica de la celda, gobernada exclusivamente por el usuario empleando la función "*Ejecutar en el Robot*" que se verá en el apartado relativo al simulador *RoboDK.*

Figura 38. Diagrama de flujo que representa el ciclo de trabajo de la estación, gobernado por los cambios de posición de los ejes de los UR.



— ***VISION*** contiene toda la lógica detrás de las animaciones relativas al proceso de medición *"simulado en RoboDK"*. El autor del proyecto hizo todo lo posible por implementar un sistema de visión real empleando la librería *OpenCV*[7] en *CODESYS*, para ello se consulta un *Paquete* disponible en *CODESYS STORE* (Figura 39) en el cual se conecta una cámara *Raspberry PI* o *USB*. Además de los algoritmos para la detección de objetos, que incluyen varios filtros (por ejemplo, *Canny*, *Gauss*) y funciones para la calibración de la cámara, la biblioteca contiene funciones para leer y guardar imágenes y otros archivos multimedia.

Figura 39. Paquete disponible en CODESYS STORE para analizar mediante la librería OpenCV, imágenes provenientes de una cámara Raspberry PI.



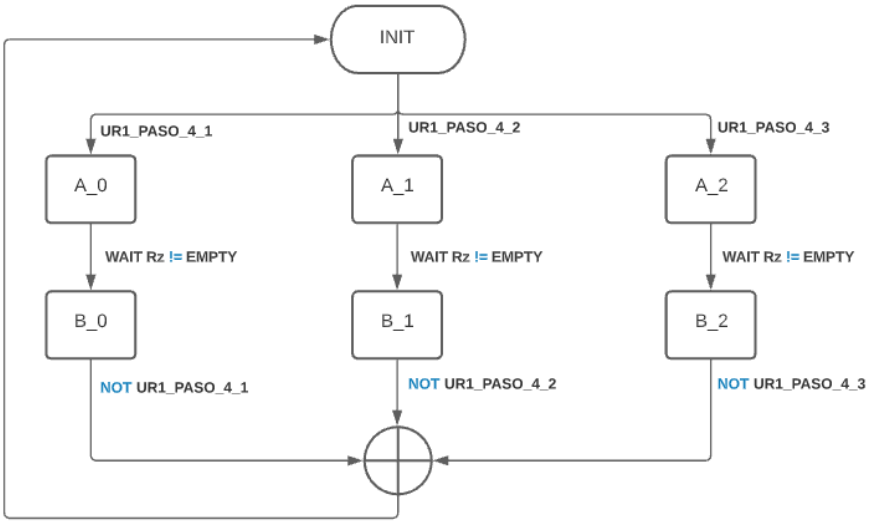
Tras analizar el código suministrado en el *Paquete* se concluye que las librerías disponibles necesitan de forma inexcusable conectar físicamente el dispositivo de adquisición de imágenes. Es por ello por lo que se llega a un compromiso intermedio, empleando *APIs* de *RoboDK* se activa una cámara en la cual se pueden configurar varios parámetros como son la *Profundidad de Campo,* la *Distancia Focal, Longitud de la Lente* entre otros y se procesa la imagen retornada por la cámara empleando un *script* programado en *Python* para detectar en que posición se encuentra la pieza que intercambian *UR1* y *UR2.*

— *CAMMERA_RoboDK_PROG (PRG)* constituye el repositorio del código asociado a todo el procesamiento de los elementos que intervienen en la visión artificial, las animaciones presentadas en la ventana *VISION* en el *WebServer* y el *HMI*, así como el manejo de los datos que se reciben desde *RoboDK* relativos a la posición de la pieza en el espacio de trabajo, en este

[7] **OpenCV** es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Véase https://opencv.org

programa además se lleva un control del número de piezas procesadas, en que posición se entrega, así como la representación del ciclo de medición que se encuentre activo y el anterior. La (Figura 40) representa un diagrama de flujo del proceso de control de la visión artificial.

Figura 40. Diagrama de flujo de la lógica para tratar la visión artificial desarrollada en el proyecto.



La transición entre *A_#* y *B_#* comprueba que se reciben todos los valores de las coordenadas donde *UR1* ha entregado la pieza, la lógica consiste en permanecer en la etapa *A_#* en tanto no se tiene el valor de *Rz* que corresponde a la rotación de la pieza relativa al eje *Z*, dicho valor es el último que envía *RoboDK*. En las etapas *A* y *B* se procesan señales de activación de animaciones, conteo de piezas, recepción y muestra de datos, así como el tratamiento del ciclo anterior. El diagrama describe tres posibles secuencias y están relacionadas con la posición en la que *UR1* se encuentra entregando la pieza.

- ***WebServer_VAR*** es la última subcarpeta que contiene código de control del proceso, dentro se han diseñado dos programas que manejan y procesan todos los datos necesarios para representar el ciclo de trabajo de la estación en el *WebServer* embebido dentro del *softPLC de CODESYS*.

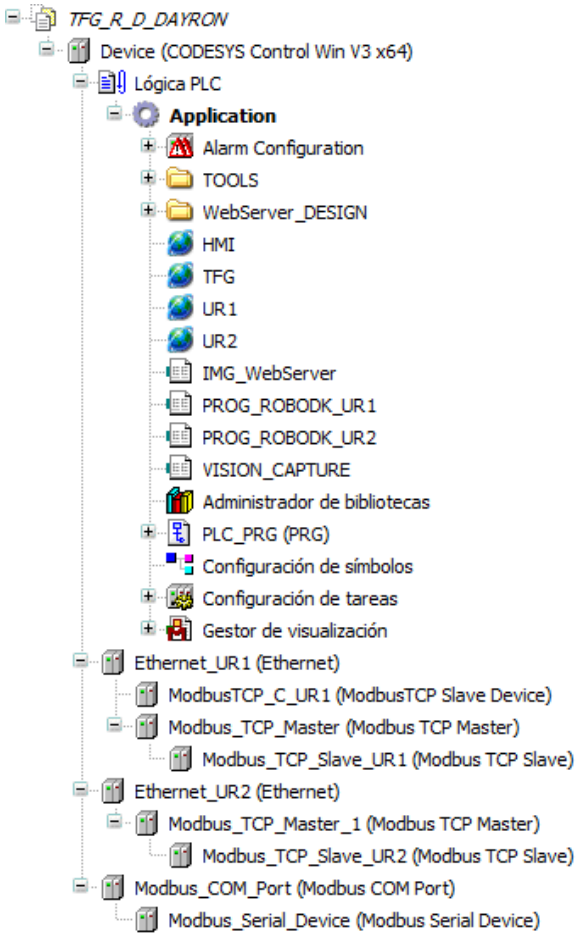
  - *WebServer_UR1 (PRG)* y su homólogo *WebServer_UR2(PRG)*, analizan los datos enviados por el *Modbus Server* alojado dentro de cada máquina virtual de *UR*, dicho servidor envía los siguientes datos internos de cada robot a través de registros de 16 *bits* en formato ***WORD*** como se analizará posteriormente: *estado de funcionamiento*, *velocidad de cada eje*, *corriente consumida*, *temperatura y diagnóstico del estado de cada eje*, *corriente consumida por el robot y el módulo de I/O*, *versión del controlador conectado*, *posición y velocidad de la brida del robot*, *así como del TCP*, *diagnóstico del*

estado y temperatura de la herramienta y por último, la corriente consumida por el TCP instalado en la brida.

Todos estos parámetros son representados en el *WebServer*, el *HMI* y el *Dashboard* que se creará en *UBIDOTS.*

- ***Tablas de Variables Globales (GVL)***: La lógica de control necesita de variables para almacenar, procesar, calcular o representar valores o señales del proceso, tal como se comentó con anterioridad, se diseña el control de forma segmentada para una mayor organización de la programación y posterior depuración del código resultante. Los objetos *HMI*, *TFG*, *UR1* y *UR2* (Figura 41), son tablas de variables globales que recopilan los datos descritos, cada una de ellas contiene parámetros, señales o variables asociadas al dispositivo que describe su denominación, con excepción de *TFG* donde las variables existentes son de carácter general del proyecto, mencionando por ejemplo las variables tipo ***BOOL*** *UBIDOTS_POS_1*, *UBIDOTS_POS_2* y *UBIDOTS_POS_3* que activan respectivamente desde el *Cloud* la posición en la que *UR1* debe entregar la pieza. Cada tabla de variables está organizada con todas las señales debidamente comentadas por el autor de forma que el lector pueda conocer la función de cada parámetro, ya que muchas se han definido siguiendo un protocolo propio.

Figura 41. Árbol del proyecto donde se representan las tablas de variables globales HMI, TFG, UR1 y UR2.

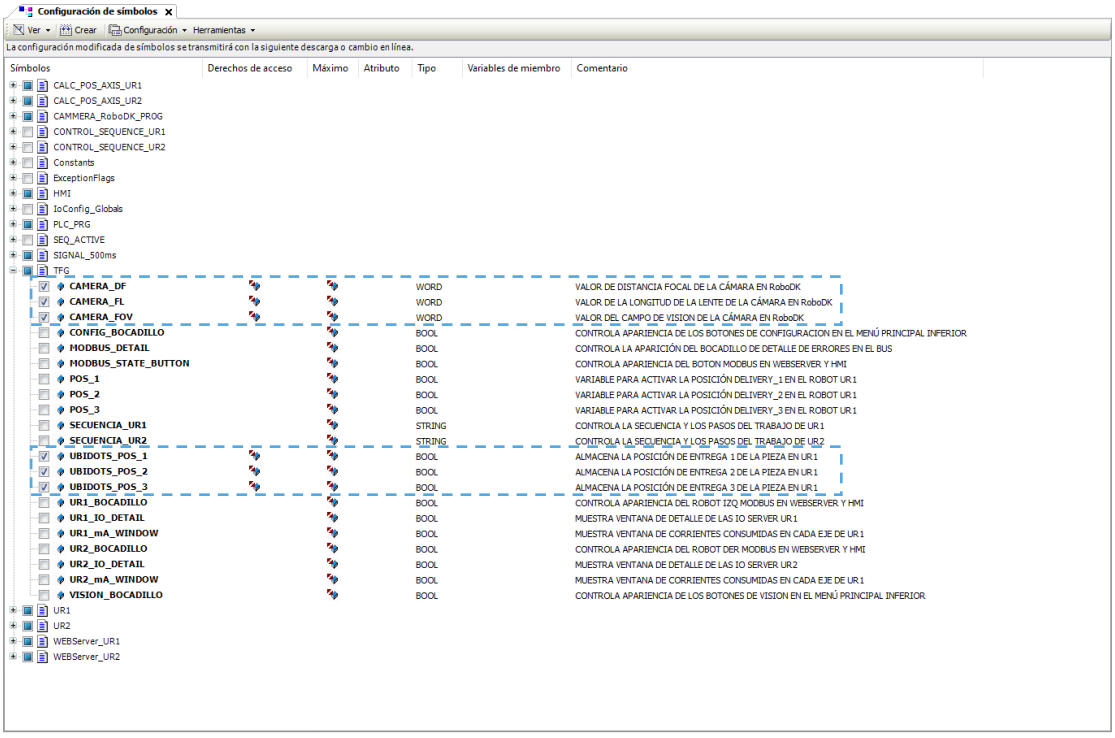


- ***Colección de Imágenes***: Seguido de las tablas de variables en el árbol del proyecto mostrado en la página anterior se encuentran 4 repositorios de imágenes que almacenan los gráficos empleados en el diseño del *WebServer*, todas las imágenes son de formato "**.png*", para garantizar transparencia de la capa y poder superponer unas con otras.

- ***PLC_PRG(PRG)***: Es el programa principal instaurado por defecto en *CODESYS* al crear un nuevo proyecto y ejecuta las instrucciones siguiendo el lenguaje definido al inicio del proyecto. En el caso particular del presente trabajo, *PLC_PRG* está configurado en lenguaje *GRAFCET* o diagrama funcional secuencial. Se ejecuta de forma cíclica y dentro se analiza el estado del bus de comunicaciones *Modbus_TCP* para *UR1* y *UR2*, se actualizan los valores de los parámetros de la cámara en *RoboDK* y se envían la mayor parte de los datos al *HMI*.

- ***Configuración de símbolos***: Objeto para la configuración de variables con acceso remoto, es esencial para establecer intercambio de información utilizando *OPC UA*[8] entre distintos dispositivos. Es equivalente a una tabla de variables que abarca todas las ***GVLs*** disponibles en el árbol del proyecto, los programas creados y sus variables internas, así como parámetros internos de configuración como el estado de los buses de comunicación, etc. Para intercambiar los datos se debe compilar previamente el proyecto y no deben existir errores, posteriormente se activa que objeto se desea compartir y de este, que variables. La (Figura 42) muestra como de la ***GLV*** *TFG* se comparten las variables asociadas a los parámetros de la cámara para poder ajustarla en *CODESYS,* así como las tres variables booleanas para activar cada una de las tres posiciones de entrega de la pieza.

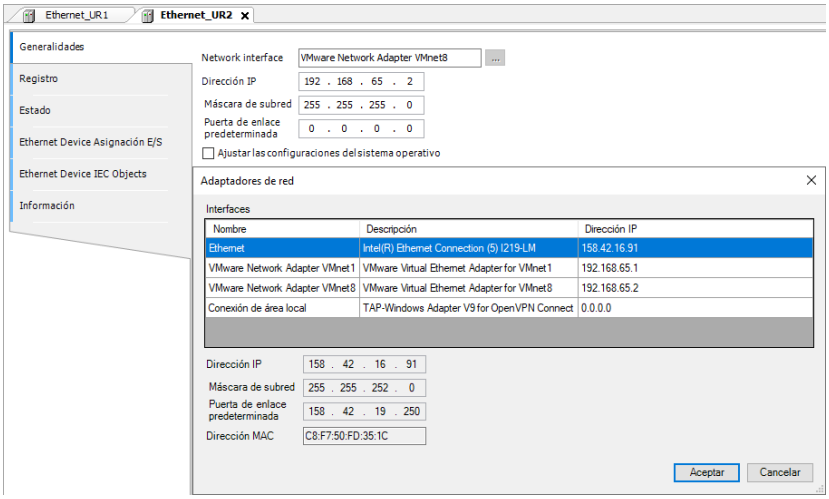
Figura 42. Configuración de símbolos para compartir variables a través del protocolo OPC UA.



[8] Es un protocolo de comunicación independiente del proveedor para aplicaciones de automatización industrial. Se basa en el principio cliente-servidor y permite una comunicación continua desde los sensores y actuadores individuales hasta el sistema ERP o la nube. Véase OPC UA | B&R Industrial Automation (br-automation.com) (*OPC UA | B&R Industrial Automation*, s. f.)

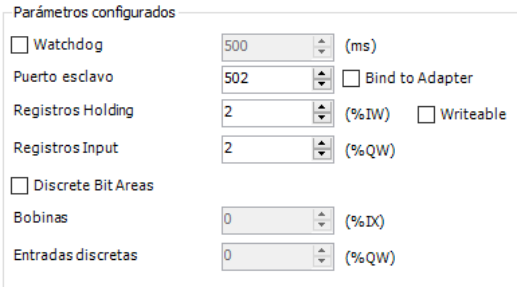
- ***COMUNICACIONES***: Como se puede apreciar en el árbol del proyecto de la (Figura 41, página 43), en el *PLC* existen dos interfaces *Ethernet* denominadas *UR1* y *UR2* para conectar mediante protocolo *Modbus_TCP* el controlador y las tarjetas de *Modbus* configuradas en los robots, así como el servidor *Modbus* alojado por defecto en cada *URSim* para enviar parámetros del robot como se explicó anteriormente. Cada interfaz está mapeada sobre uno de los puertos virtuales de *Ethernet Adapter* asociados a cada máquina virtual, para el *UR1* corresponde la dirección 192.168.65.1 y para *UR2* 192.168.65.2, como se puede apreciar la dirección se relaciona con una red local que se crea en el *PC Host*[9]. (Figura 43).

Figura 43. Configuración de las dos interfaces de Ethernet del proyecto desde las cuales se establece la comunicación Modbus con los UR.



- En la interfaz *Ethernet_UR1* se inserta un dispositivo *Modbus_TCP_Slave* que se corresponde con el servidor *Modbus* disponible dentro de *CODESYS*, este servidor será el único encargado de comunicarse con las dos tarjetas *Modbus Client* configuradas dentro de cada *UR*, las cuales se utilizan para mapear señales *I/O* o registros para la comunicación *PLC-UR.* Los únicos parámetros que se configuran son el número de registros de entrada y salidas que se compartirán con los dispositivos externos (Figura 44), en el caso particular del proyecto, con el mínimo establecido de 2 es suficiente.

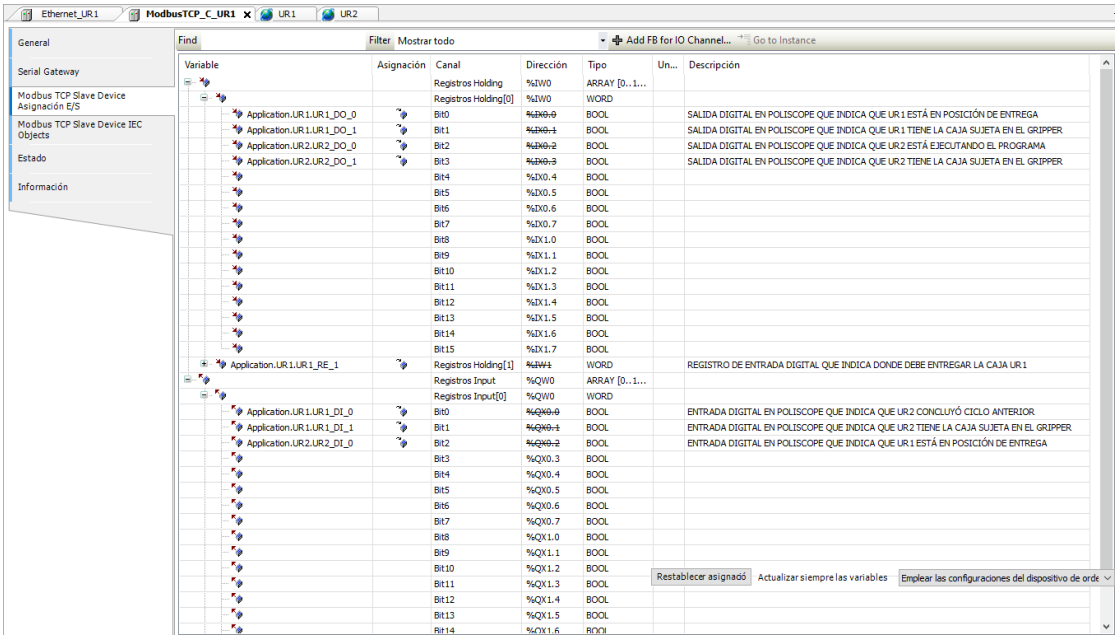
Figura 44. Configuración del dispositivo Modbus Server alojado en el PLC de CODESYS.



[9] El proyecto se desarrolla en un *PC* ubicado en el *Departamento de Ingeniería de Sistemas y Automática* de la *UPV* contra el cual se establece un *Acceso a Escritorio Remoto* desde el domicilio del autor del proyecto, en dicho *PC* se encuentran instaladas todas las herramientas necesarias para solventar los objetivos del *TFG*, a su vez, se instaló una licencia de VMware Workstation para ejecutar las máquinas virtuales de *URSim* descritas.

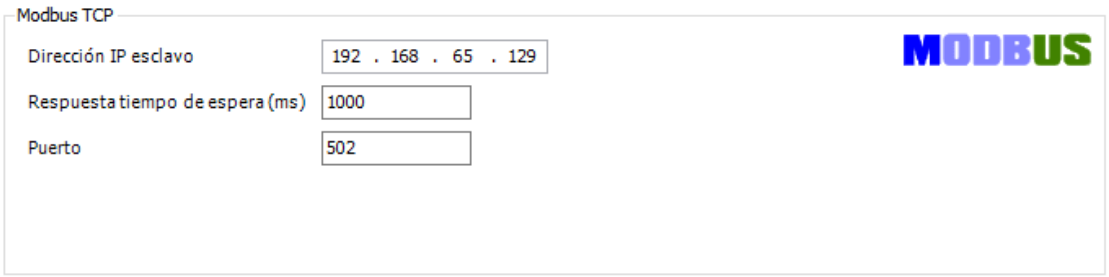
Una vez configurados los registros de entrada-salida, solo resta mapear las señales en las direcciones correctas para que cada *UR* puede intercambiar información con el *PLC* sin sobrescribir ningún dato en el bus. La (Figura 45) representa como se diseña el mapeado de las señales para la comunicación con las *URSim*.

Figura 45. Mapeado de las señales que intercambia el PLC con las dos URSim a través de Modbus_TCP.



El próximo dispositivo que se adjunta a la interfaz *Ethernet* es un terminal *Modbus_TCP_Master,* no requiere configuración alguna, sobre este se agrega otro módulo denominado *Modbus_TCP_Slave_UR1* (existe su homólogo *UR2* para el robot de la derecha), en el cual se configuran los registros que se van a leer del *Modbus Server* alojado dentro de cada *UR* (el *Anexo II: UR MODBUS DATA SERVER)* numera cada registro accesible dentro del robot). En cada uno de estos módulos solo debe configurarse la *dirección IP* de los robots, tal como se muestra en la (Figura 46), como se comentará posteriormente *UR1* ocupa la dirección 192.168.65.129 y *UR2* la dirección 192.168.65.130.

Figura 46. Configuración de la dirección IP correspondiente a cada robot que equivale al Modbus Server interno.



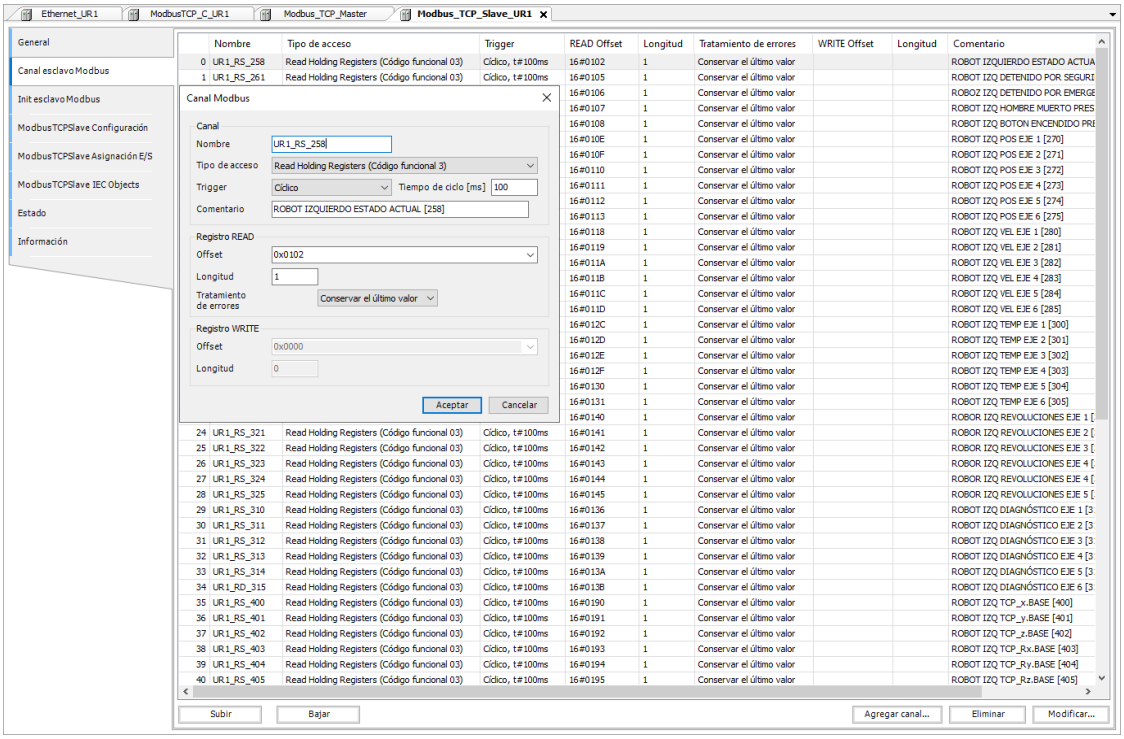
El acceso a los registros de *Modbus* se realiza teniendo en cuenta las siguientes direcciones que se muestran en la (Tabla 5):

Tabla 5. Modelo de Direccionamiento Modbus_TCP.

Modbus Command	Description	Slave Address	Data Address
01	Read Coil Status	1 - 255	00001 – 09999
02	Read Input Status	1 – 255	10001 – 19999
03	Read Holding Register	1 – 255	40001 – 49999
04	Read Input Register	1 - 255	30001 - 39999

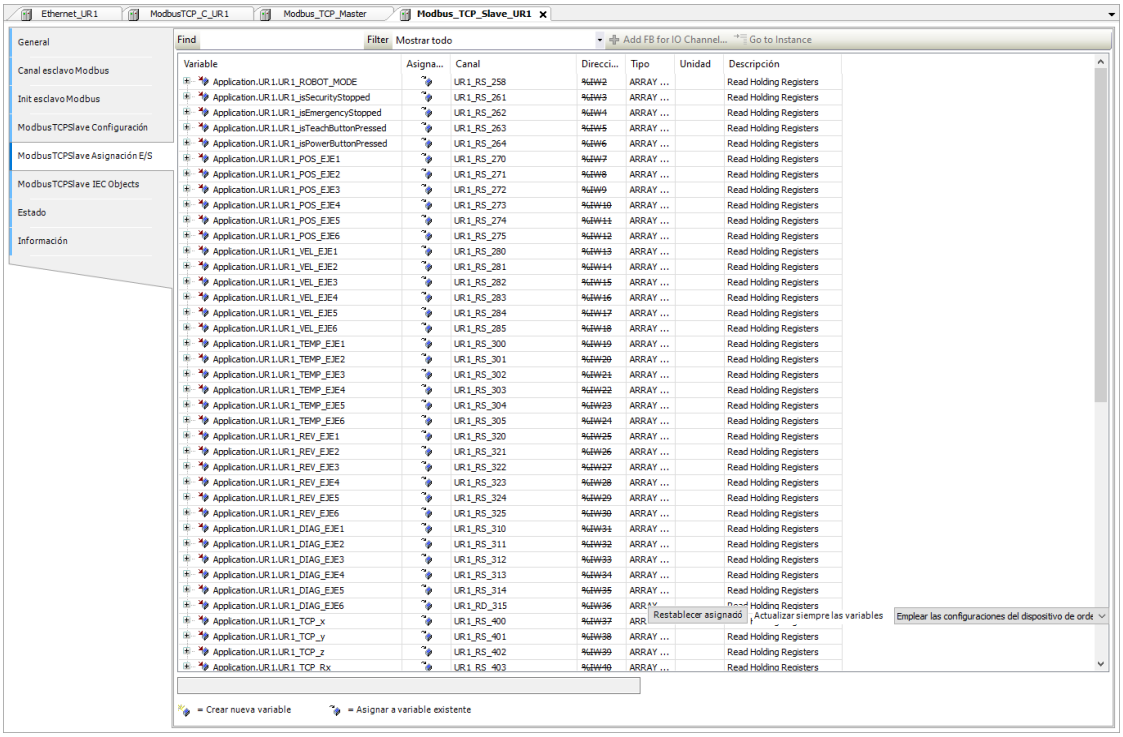
Tras asignar la dirección *IP* a cada módulo correspondiente con cada *UR* y teniendo en cuenta el acceso a los registros de *Modbus*, solo queda configurar en el dispositivo *Modbus_TCP_SLAVE_UR1* aquellos registros que interesan examinar, en este proyecto se consultan todos y cada uno de ellos, desde la dirección ***256*** hasta el registro alojado en la dirección ***770***. Para consultar o leer la información de un registro de *UR* hay que apoyarse en la tabla de direccionamiento de registros que *Universal Robot* proporciona (*Anexo IV: UR MODBUS DATA SERVER*), en la (Figura 47) se describe el procedimiento para acceder al registro ubicado en la dirección ***258*** que corresponde al estado actual del robot *UR1* o de forma equivalente al modo de funcionamiento en el que se encuentra. Para crear un nuevo registro de consulta se presiona sobre el botón inferior izquierdo *Agregar Canal* de la ventana *Canal esclavo Modbus* y se parametriza el registro con la información necesaria, el tipo de acceso corresponde con el comando ***03*** de *Modbus* (*Read Holding Register*) y el *offset* no es más que la dirección del registro que se desea consultar.

Figura 47. Configuración de los registros que se desean leer en el Modbus Server del robot UR1.



Tras definir todos los canales de acceso a los registros de los robots, el último paso que se debe realizar es el mapeado con las señales o variables creadas en cada ***GLV*** anterior como se muestra en la (Figura 48), todos los pasos descritos garantizan una monitorización absoluta y real de cada robot, tal y como se hace en una programación industrial totalmente operativa, uno de los elementos fundamentales en una celda automatizada es el diagnóstico de todos los dispositivos y procesos, en ello reside el éxito de un buen funcionamiento, plan de mantenimiento o planificación de la producción.

Figura 48. Mapeado de las señales predefinidas en las tablas de variables sobre los canales configurados.



- ***Modbus_RTU***: Para comunicar el *softPLC* de *CODESYS* con la pantalla *HMI* de *Omron* se implementará una conexión virtual entre ambos dispositivos emulando un enlace a través de los *puertos series* ***COM1*** y ***COM2*** del ordenador. Para ello es necesario instalar algún programa que permita simular la conexión interna de los puertos mencionados, el autor quiere dejar a disposición del lector 3 herramientas gratuitas diferentes que cuentan con período de validez de 15 días respectivamente, de esta forma se tienen 45 días en total o en su defecto si alguno no funciona correctamente, se cumple el objetivo de facilitar herramientas para el desarrollo de la plataforma con al menos un mes de prueba con la mayoría de las funcionalidades activas.

1- *Free Virtual Serial Por Tool:* Virtual Serial Port Driver - create and emulate virtual COM port (eltima.com)

2- *Virtual Serial Port Kit:* Download Virtual Serial Port Kit (fabulatech.com)

3- *Com Port Data Emulator:* Free download | COM Port Data Emulator - Freeware (aggsoft.com)

Instalado el programa, es necesario incluir en el árbol del proyecto del *PLC* un dispositivo *Modbus_COM_Port* en el cual se configura el puerto serie a través del cual se va a comunicar

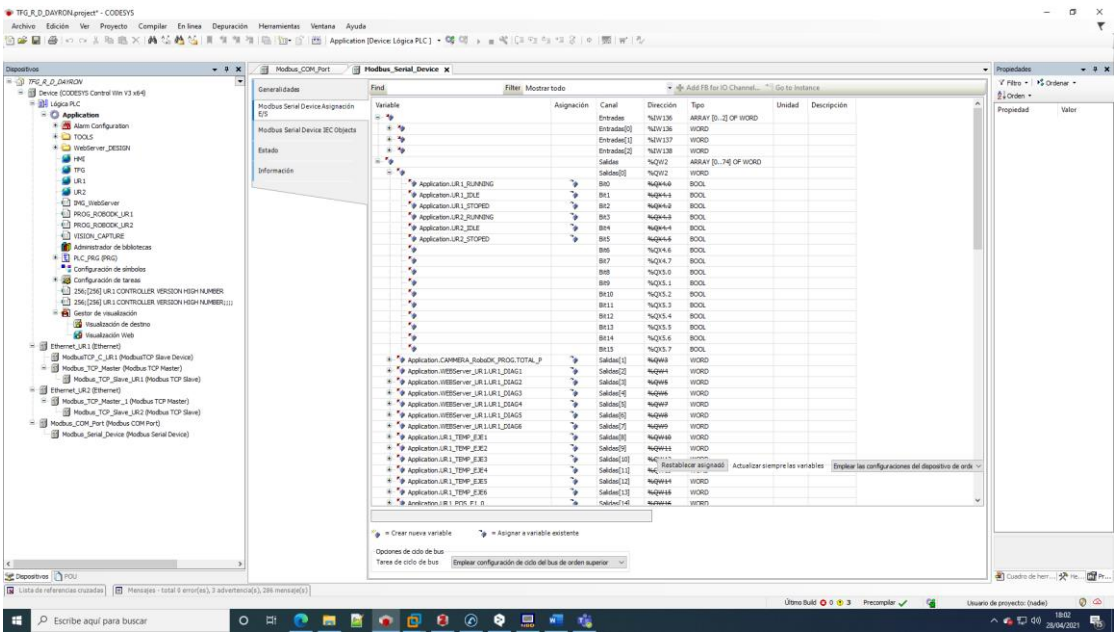
CODESYS, la *velocidad* de la comunicación, en *baudios*, la *paridad*, el *Bits de datos* y *Bits de parada*, de forma respectiva la configuración queda, ***115000***, ***EVEN***, ***8*** y ***1***.

En el dispositivo se adjunta un elemento *Modbus_Serial_Device* que permite la configuración de las señales que se intercambiarán con la pantalla. En la pestaña *Generalidades* del módulo, se configuran los *Registros Holding (%IW)* en ***3 registros*** y los *Registros Inputs (%QW)* en ***75 registros***.

Se informa al lector que el HMI solo mostrará la información relativa a la pantalla INDEX y UR1_EJES, no es objetivo volver a reproducir todas las pantallas que se desarrollaran en el WebServer nuevamente en otro dispositivo. El concepto consiste en crear el par de ventanas mencionadas y así el lector tiene una base para realizar un diseño propio.

Con los registro para el intercambio de información declarados en la pestaña anterior, es momento de realizar el mapeo de las señales necesarias para desarrollar posteriormente el diseño del *HMI* en *NB Designer.* La (Figura 49) muestra cómo se establece el direccionamiento del bus serie.

Figura 49. Direccionamiento del bus Modbus_RTU para la comunicación con el HMI.



5.2. ROBOT UR - *URSim*

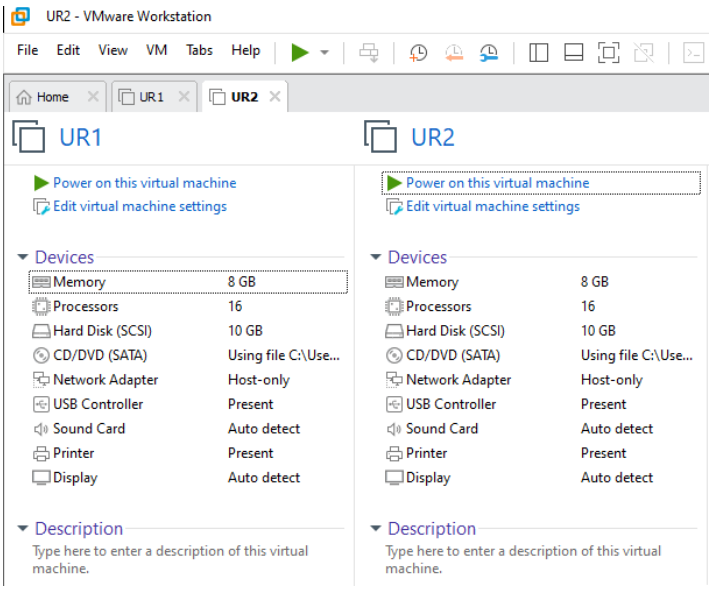
"Fabricantes de todo el mundo están recurriendo a la automatización para ayudar a resolver la escasez de mano de obra, aumentar la productividad y mejorar la calidad del producto. Los robots colaborativos proporcionan una solución de automatización rentable, flexible y segura para una amplia gama de tareas de producción. Universal Robots vendió más de 50.000 Cobots que se utilizan en diferentes entornos productivos cada día en todo el mundo" (Fabricante Robots colaborativos – Cobots de Universal Robots, s. f.).

Esta empresa danesa lidera el segmento de mercado de los *Cobots*, creando una filosofía de compartir libremente sus conocimientos y romper de una vez las barreras de acceso a la tecnología de los robots industriales. *Universal Robot* es la *única* empresa de producción de robots que ofrece de forma gratuita un *IDE* para la simulación y programación de sus dispositivos a todo aquel que esté interesado. Es posible descargar la versión más reciente del simulador *offline* desde el siguiente enlace, solo es necesario darse de alta:

UR Download | Support Site | Universal Robots (universal-robots.com)[10]

Junto con el disco duro creado en *Linux Ubuntu-64 bits*, se dispone de un documento, traducido a varios idiomas incluido el castellano, donde se describe paso a paso como instalar *URSim*[11], en el caso personal, la configuración con *VM Oracle* no permitía poder maximizar la ventana de *Polyscope*, por lo cual, utilizando una licencia de la *UPV* del software *VMware Workstation PRO,* se implementan dos máquinas virtuales idénticas denominadas *UR1* y *UR2* respectivamente (Figura 50), constituyendo el modelo fisicomatemático que virtualiza el robot real.

Figura 50. Máquinas virtuales idénticas de URSim para la virtualización de los dos robots UR de la celda.



[10] Puede que el lector encuentre una versión más actual a la mencionada. Consultado con fecha 21/04/2021.

[11] Consultar ursim_vmoracle_installation_guide_v3_es.pdf (universal-robots.com)

La (Tabla 6) muestra los parámetros de configuración de la máquina virtual, por si el lector desea realizar una parametrización similar a la tratada en este proyecto:

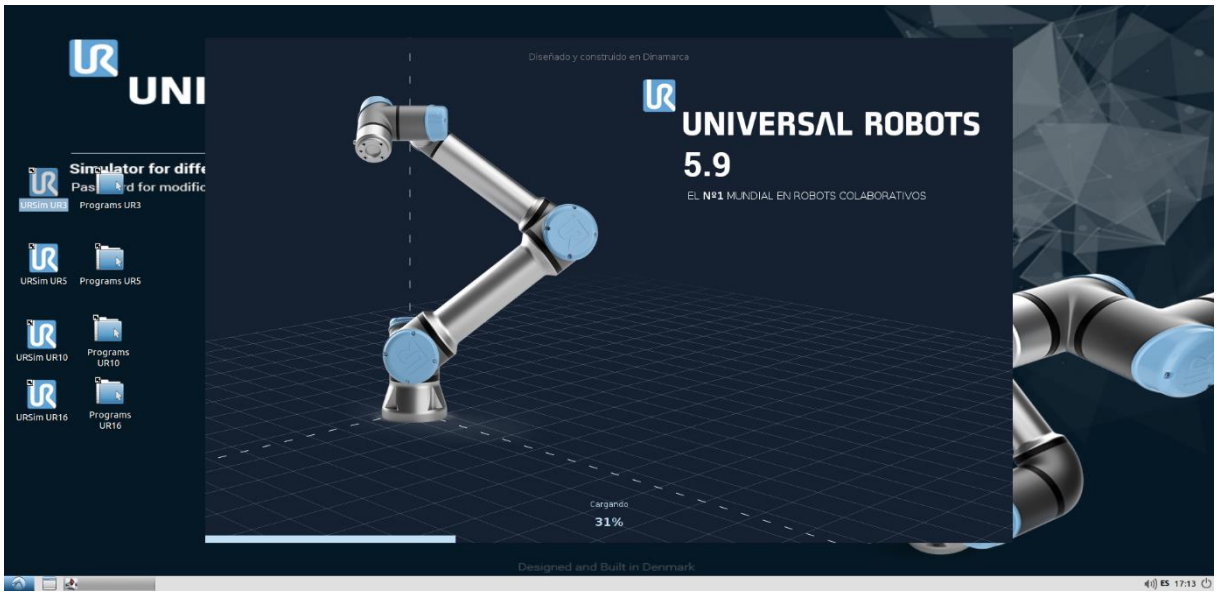
Tabla 6. Parámetros de configuración de la máquina virtual de URSim en VMware Workstation PRO.

Dispositivo	Descripción
Memory	8 GB
Processors	16 (4 x 4)
Hard Disk	10 GB
CD/DVD (SATA)	*** Cargar el disco virtual de URSim ***
Network Adapter	Host – only
Virtual Machine Name	UR1 o UR2
Guest Operating System	Linux
Versión	Ubuntu 64-bits
Share Folders	Enable (Always Enable) ** Se comparte el escritorio *PC HOST* **

Configuradas las dos máquinas virtuales y ejecutándose al unísono, es necesario establecer algunos parámetros en los robots para implementar la comunicación con el simulador gráfico *RoboDK*.

URSim por defecto trae instalado en el escritorio los 4 *IDE* relacionados con todos los modelos de robots de la marca (Figura 51), la celda original contempla dos *UR3e*, por lo que será este el software que se emplea en la resolución del proyecto.

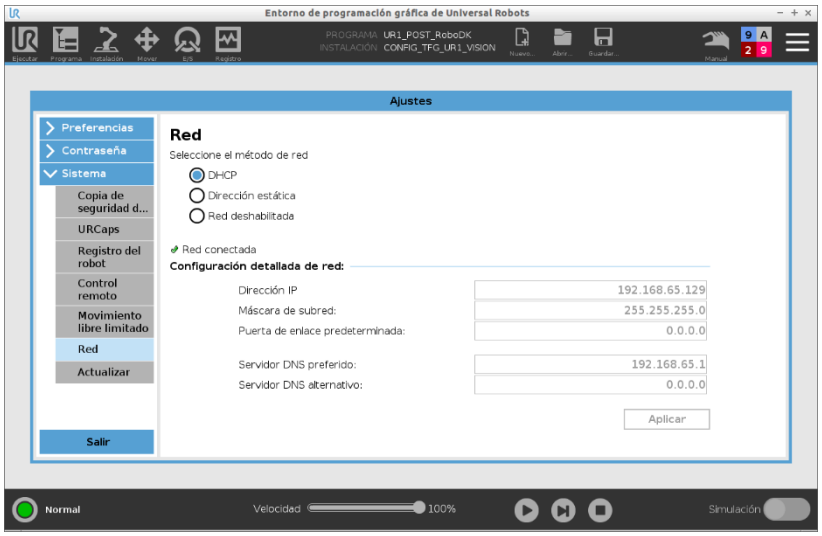
Figura 51. Se emplea la versión del software relativa a URSim UR3 para desarrollar la virtualización del robot.



El primer paso es conocer que dirección *IP* tiene cada robot, en un sistema robotizado real la dirección es configurable por el programador que desarrolla la celda, en el caso de *URSim*, el robot toma la dirección de la máquina virtual y no es posible modificarla pues se establece empleando *DHCP*, para conocer la dirección de cada robot, una vez abierto *URSim UR3*, en la esquina superior derecha existe un icono con tres rayas horizontales (≡ *Menú*), al presionar aparece una ventana de *Ajustes*, navegar

hasta *Sistema/Red* (Figura 52), dicho valor debe ser tenido en cuenta para configurar los módulos *Modbus_TCP_Slave* mencionados en el apartado anterior.

Figura 52. Ventana de Ajustes de UR1 donde aparece la dirección IP del robot.



En la misma ventana, constituye una buena praxis accediendo a la pestaña *Contraseña*, crear una contraseña de *Modo* (necesaria para el cambio de modos de trabajo: *Control Remoto-Automático-Manual*) y otra de *Seguridad* (en este caso es un nivel de protección para autenticarse ante posibles modificaciones de los parámetros de seguridad del robot). La pestaña *Preferencias* permite elegir el idioma, cambiar la hora y fecha y la activación de la barra deslizante de velocidad.

No es objetivo del proyecto describir cómo se programa un robot de *Universal Robot,* en caso necesario el lector dispone de clases gratuitas en la página *Web* de la empresa, apartado *Academy*[12] (Figura 53).

Figura 53. Certificados obtenidos por el autor que garantizan poseer los conocimientos básicos para programar un robot.

Fuente: Formación en línea gratuita (universal-robots.com)

[12] Véase Formación de Manejo de Robots Gratuita | Universal Robots Academy (universal-robots.com)

En cada robot se monta una pinza electrónica de la marca *Robotik* modelo *2F-140*[13], las características de este elemento se muestran en la (Tabla 7):

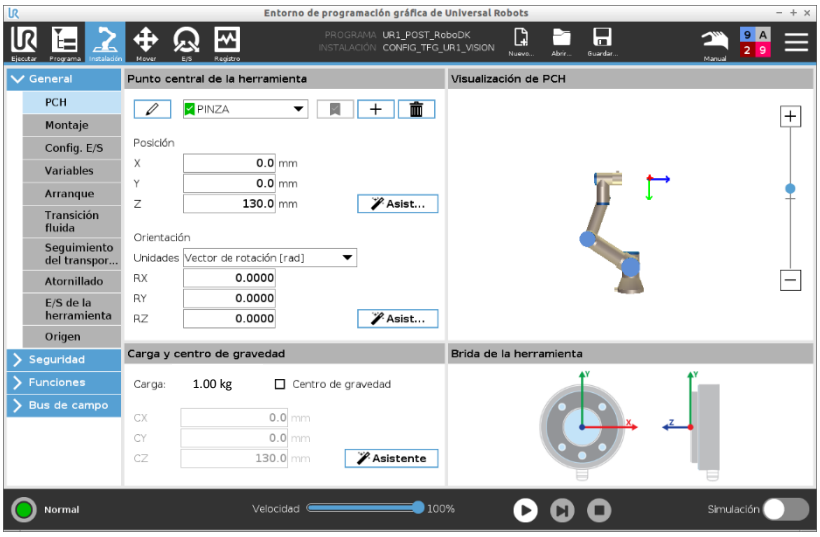
Tabla 7. Especificaciones de los modelos de pinzas 2F-85 y 2F-140 de la marca Robotik.

SPECIFICATIONS	2F-85	2F-140
Stroke (adjustable)	85 mm	140 mm
Grip force (adjustable)	20 to 235 N	10 to 125 N
Form-fit grip payload	5 kg	2.5 kg
Friction grip payload	5 kg	2.5 kg
Gripper mass	0.9 kg	1 kg
Position resolution (fingertip)	0.4 mm	0.6 mm
Closing speed (adjustable)	20 to 150 mm/s	30 to 250 mm/s
Communication protocol	Modbus RTU (RS-485)	
Ingress protection (IP) rating	IP40	IP40

Fuente: Product-sheet-Adaptive-Grippers-EN.pdf (robotiq.com)

El *PCH*[14] se estable con una distancia de 130 *mm* (Figura 54) en el menú *Instalación/PCH*, y se define la posición *Home* del robot *UR1* en la pestaña *Origen* con los valores relativos a dicho punto definidos en el (*Anexo I: Tabla de control de posiciones*).

Figura 54. Ventana de configuración del PCH.



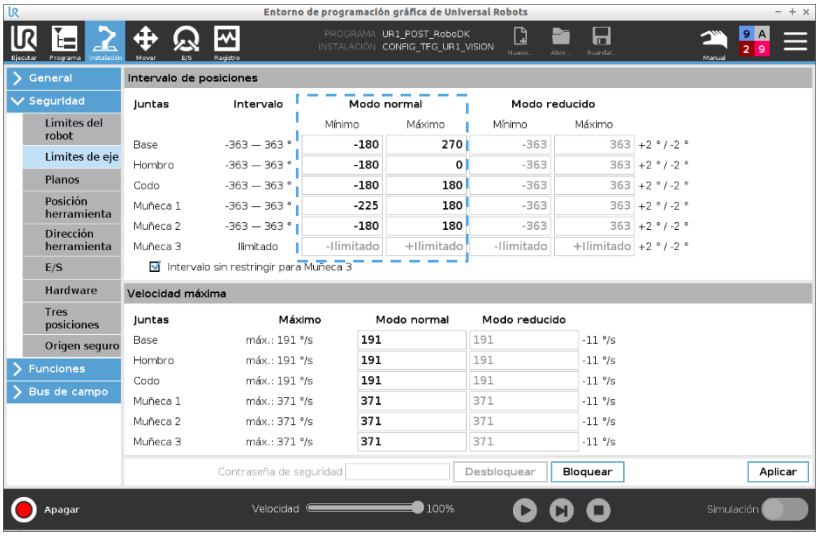
Según la posición en la que quedan montados los dos robots en la mesa, enfrentados uno al otro y sobre una línea imaginaria paralela a los bordes de la mesa, el autor plantea una restricción de

[13] Véase manual en Robotiq 2F-85 & 2F-140 Instruction Manual

[14] *PCH*. Punto central de la herramienta, término más conocido en el sector como *TCP*.

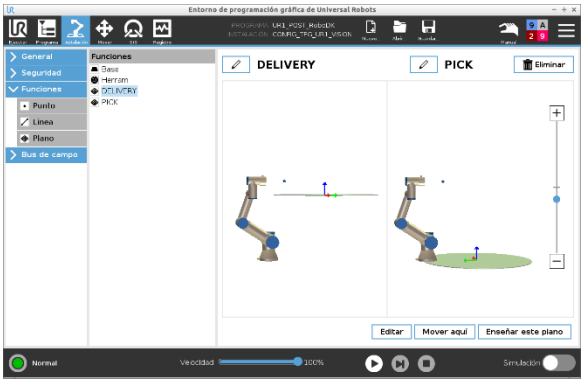
movimiento de los ejes de cada robot según se muestra en la (Figura 55), sirviendo estos parámetros de trabajo para configurar elementos *id* de alarmas en el *PLC*, tal y como se comentó anteriormente.

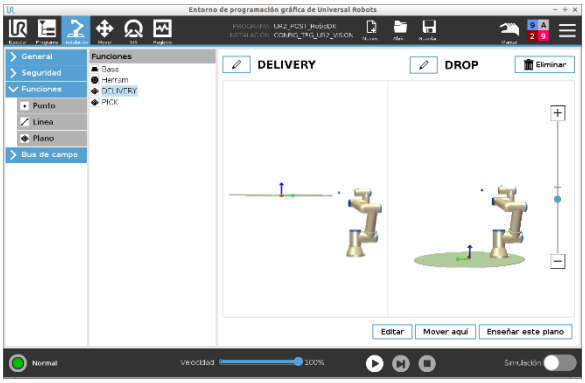
Figura 55. Parametrización de los límites de los ejes de cada robot.



Los próximos elementos que se crean son los sistemas de referencias empleados en *RoboDK* para definir las posiciones de trabajo, denominados *Delivery* y *Pick* para *UR1* y *Delivery* y *Drop* para *UR2* (Figura 56), para su implementación en necesario acceder desde la pestaña *Funciones* y se emplean funciones de tipo *Plano.*

Figura 56. Definición de los sistemas de referencias necesarios para los puntos de trabajo de la estación.

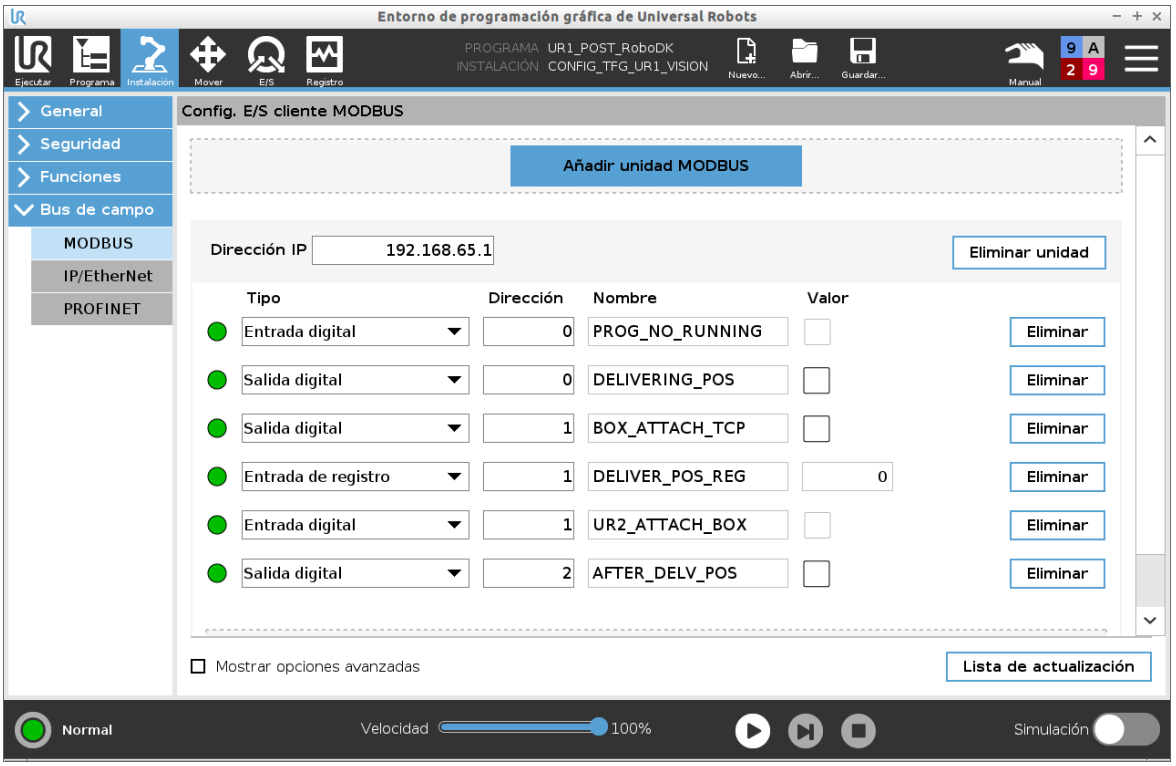


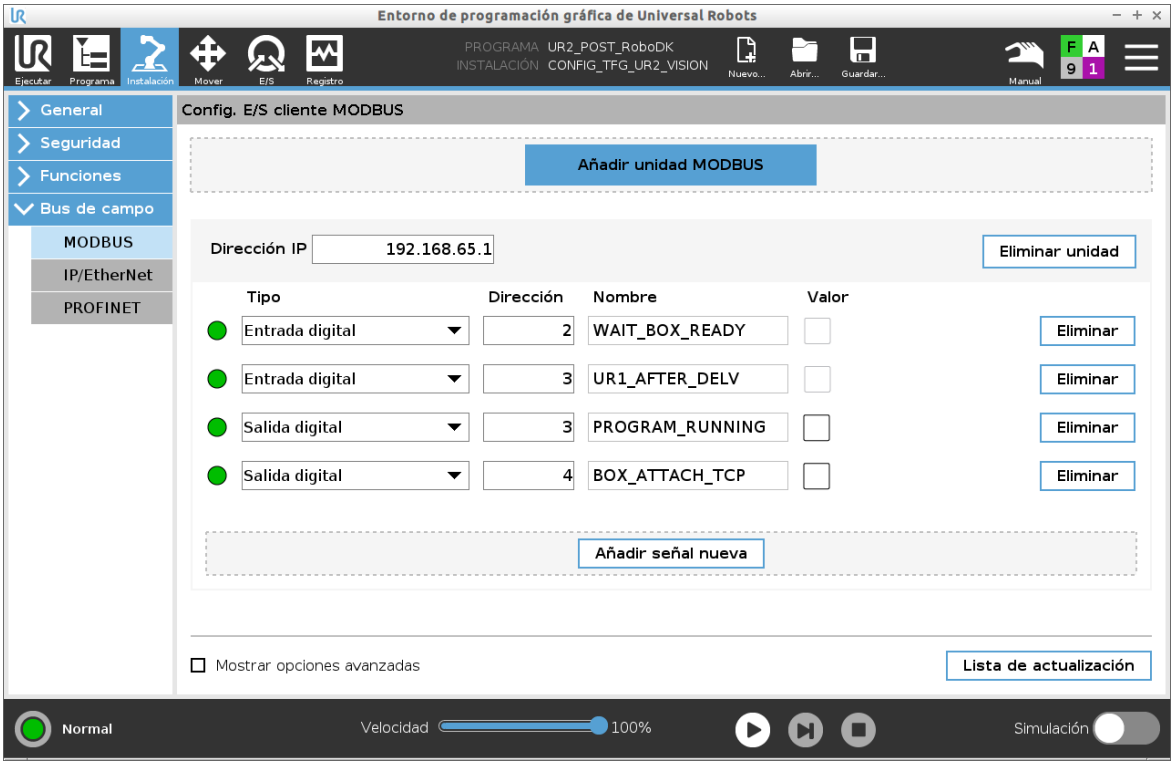


El último elemento que necesita ser declarado es la tarjeta de comunicaciones *Modbus Client* en cada *URSim* para el envío de las señales que se comentaron en el apartado *Comunicaciones* (página 45), como se puede consultar, el dispositivo *Modbus_TCP Slave Device* del controlador está enlazado a la interfaz *Ethernet_UR1* con *dirección IP 192.168.65.1*, este módulo constituye el elemento que actúa como *Server* dentro del *PLC* de *CODESYS*, por este motivo, en las dos tarjetas de *Modbus* que se crean en cada robot, la *dirección IP* que se indica será la de la tarjeta *Modbus Server* de *CODESYS*, como es fácil de entender, *CODESYS* solo puede contener un dispositivo *Server* sobre el cual se reciben peticiones de datos provenientes de todos los periféricos conectados a dicha unidad, por lo que es importante

prestar especial atención al direccionamiento con el cual se programa en cada tarjeta de los *UR*, de esta forma se evita el solapamiento de señales[15]. (Figura 57.a y Figura 57.b).

Figura 57.a. Configuración y direccionamiento de la tarjeta Modbus Client de UR1. b. Configuración y direccionamiento de la tarjeta Modbus Client de UR2.





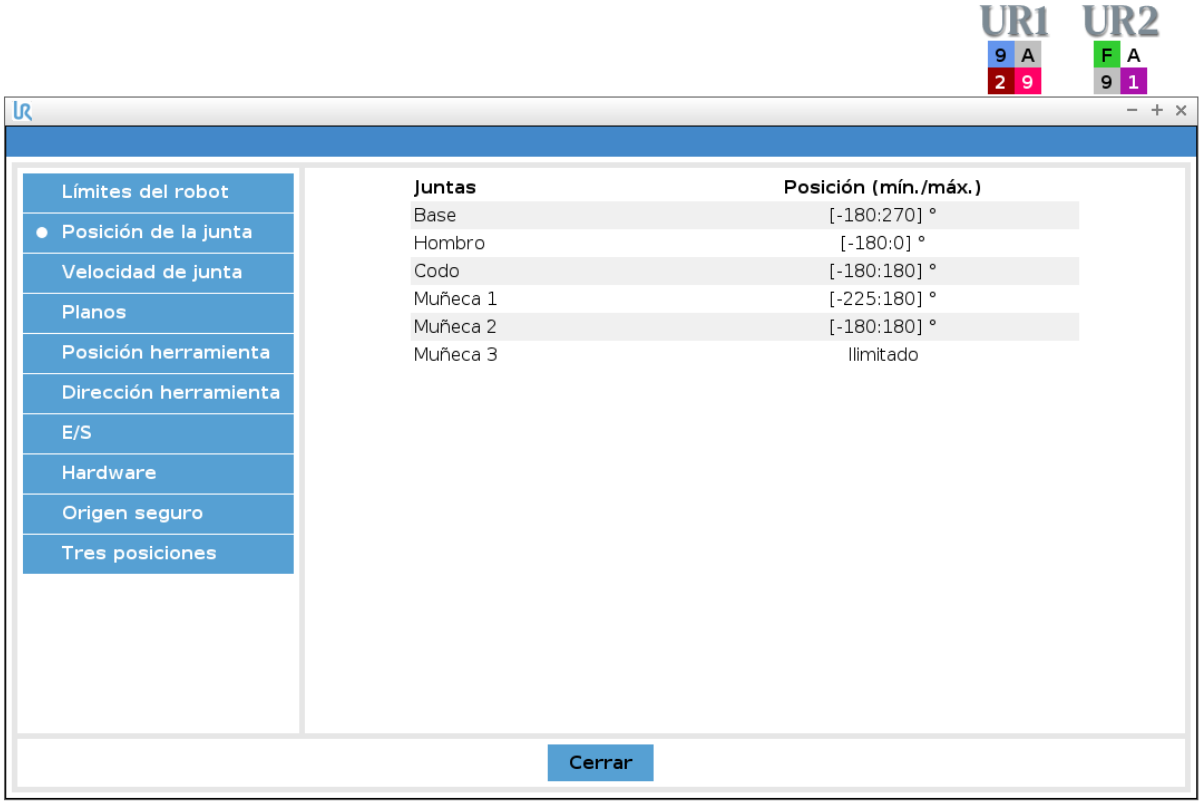
[15] Véase Figura 45. Mapeado de las señales que intercambia el *PLC* con los dos *URSim* a través de *Modbus_TCP*. Página 46

Cada robot, según la configuración realizada, ha actualizado el valor de la ***suma de comprobación de seguridad*** (Figura 58):

El icono *suma de comprobación de seguridad* muestra la configuración de seguridad de robot aplicada, de izquierda a derecha, por ejemplo: **BF4B**. El texto o colores diferentes indican cambios a la configuración de seguridad aplicada. (*99454_UR3e_User_Manual_en_US.pdf*, s. f.)

- La *suma de comprobación de seguridad* cambia si se modifican los ajustes de las funciones de seguridad, dado que la *suma de comprobación de seguridad* solo se genera mediante los ajustes de seguridad.
- Debe aplicar sus cambios en la configuración de seguridad para que la suma de comprobación de seguridad refleje sus cambios.
- Constituye el único método de comprobación de que ningún parámetro de configuración del robot ha sido alterado desde la entrega de la celda por el técnico de desarrollo de la programación del robot.

*Figura 58. identificación de la **suma de comprobación de seguridad** para cada robot.*



5.2.1. RTDE

*"La comunicación entre el simulador gráfico RoboDK y los robots virtuales implementados a través de dos máquinas virtuales URSim se materializa a través del protocolo de intercambio de datos **RTDE**. La interfaz de intercambio de datos en tiempo real RTDE proporciona una forma de sincronizar aplicaciones externas con el controlador UR a través de una conexión TCP/IP estándar, sin romper ninguna de las propiedades en tiempo real del controlador del robot. Esta funcionalidad es útil, entre otras cosas, para interactuar con los controladores de bus de campo (por ejemplo, Ethernet/IP), manipular las E/S del robot y trazar el estado de este (por ejemplo, las trayectorias del robot). La interfaz RTDE está disponible por defecto cuando el controlador UR está en funcionamiento.*

La sincronización es configurable y puede incluir, por ejemplo, los siguientes datos:

- *Salida: estado del robot, de la junta, de la herramienta y de la seguridad, E/S analógicas y digitales y registros de salida de propósito general.*

- *Entrada: salidas digitales y analógicas y registros de entrada de propósito general*

La funcionalidad de RTDE se divide en dos etapas: un procedimiento de configuración y un bucle de sincronización. Al conectarse a la interfaz RTDE, el cliente se encarga de configurar las variables a sincronizar. Se puede especificar cualquier combinación de registros de entrada y salida que el cliente necesite escribir y leer, respectivamente. Para ello, el cliente envía una lista de configuración de los campos de entrada y salida con nombre que deben contener los paquetes de datos de sincronización reales. La definición de un formato de paquete de datos de sincronización se denomina receta. Existe un límite máximo de 2048 bytes para representar la lista de nombres de campos de entrada/salida a los que un cliente desea suscribirse. A cambio, la interfaz RTDE responde con una lista de los tipos de variables o especifica que no se ha encontrado una variable concreta. Cada receta de entrada que se haya configurado correctamente obtendrá un id de receta único. Una vez completada la configuración, la sincronización de datos puede iniciarse y pausarse.

Cuando se inicia el bucle de sincronización, la interfaz RTDE envía al cliente los datos solicitados en el mismo orden en que fueron solicitados por el cliente. Además, se espera que el cliente envíe entradas actualizadas a la interfaz RTDE en caso de cambio de valor. La sincronización de datos utiliza datos binarios serializados.

Todos los paquetes comparten la misma estructura general con una cabecera y una carga útil, si procede. Los paquetes utilizados para el procedimiento de configuración generan un mensaje de retorno. Los paquetes del bucle de sincronización no lo hacen. Tanto el cliente como el servidor pueden enviar en cualquier momento un mensaje de texto, cuyo nivel de advertencia especifica la gravedad del problema. El RTDE está disponible en el puerto número 30004". (*Real-Time Data Exchange (RTDE) Guide - 22229*, s. f.)

La lista de nombres de campos admitidos y sus tipos de datos asociados se encuentra en el documento facilitado por Universal Robot del cual se ha obtenido toda la información anterior relativa a este protocolo:

Real-Time Data Exchange (RTDE) Guide - 22229 (universal-robots.com)

Tal como se ha definido *RTDE* emplea *TCP/IP* para el intercambio de datos entre la controladora y el periférico que se desea conectar a ella. Como se pudo constatar en el desarrollo de la plataforma, el ***proxy*** de la máquina virtual de *URSim* se encuentra desactivado, lo que imposibilita establecer ninguna comunicación entre la controladora y *RoboDK.* Este aspecto es de vital importancia, la mayoría de la bibliografía relacionada con el simulador de *URSim* no menciona nada al respecto, por lo cual, es común encontrar, incluso en bibliografía elaborada por expertos en la materia, que *URSim* ***no se puede conectar físicamente con ningún dispositivo externo y solo es válido para la programación offline de los robots de Universal Robot***, hecho que como se demostrará en este proyecto, no es correcto.

5.2.2. URSim PROXY

Además de emplear los puertos 30001, 30002 y 30004 para la comunicación *RTDE* entre la controladora y *RoboDK*, los robots necesitan implementar una conexión *Modbus* con el *Server* alojado en *CODESYS*, el cual establece el enlace por medio del puerto 502. Como no se sabe si en un futuro se necesite comunicar las máquinas virtuales mediante *http* o *https*, estos puertos serán igualmente incluidos en la excepción del *proxy.*

El procedimiento es sencillo y requiere escasos 5 minutos, para ello se debe abrir un terminal *UXTERM* en la máquina virtual (Figura 59)

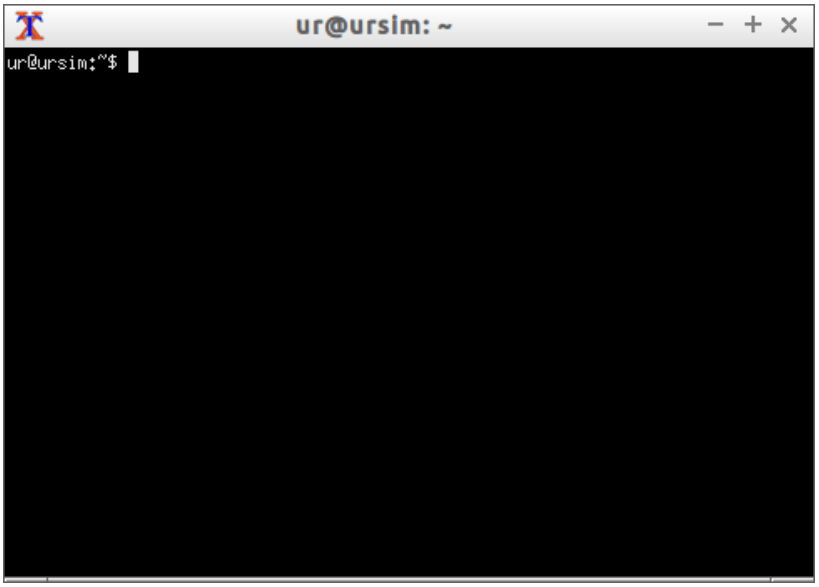
- **Activar el proxy:** El código a introducir es el siguiente (texto subrayado):

  ur@ursim:~$ sudo ufw enable

  el sistema devuelve un mensaje confirmando la activación

  Firewall is active and enabled on system startup

Figura 59. Apertura del terminal para la activación del proxy en Linux así como la apertura de puertos.



- **Apertura de puertos:**

```
ur@ursim:~$ sudo ufw allow 502/tcp
ur@ursim:~$ sudo ufw allow 80/http
ur@ursim:~$ sudo ufw allow 443/https
ur@ursim:~$ sudo ufw allow 21/ftp
ur@ursim:~$ sudo ufw allow 30000
ur@ursim:~$ sudo ufw allow 30001
ur@ursim:~$ sudo ufw allow 30002
ur@ursim:~$ sudo ufw allow 30003
ur@ursim:~$ sudo ufw allow 30004
```

el sistema devuelve un mensaje confirmando la apertura

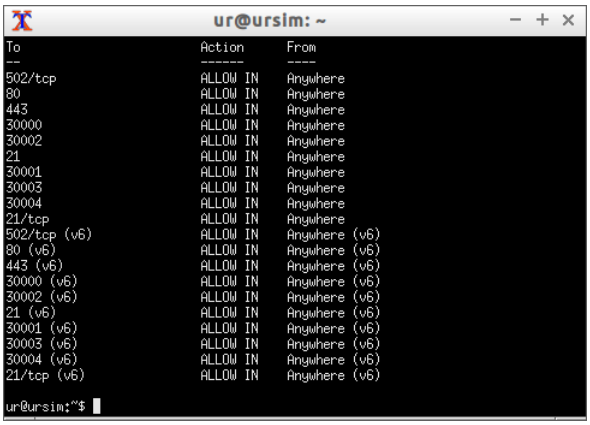
```
Rule updated
Rule updated (v6)
```

- Comprobar la activación de los puertos anteriores:

```
ur@ursim:~$ sudo ufw status verbose
```

el sistema devuelve la siguiente tabla resumen (Figura 60)

Figura 60. Conjunto de puertos activos en el proxy de URSim.



- Desactivar un puerto:

```
ur@ursim:~$ sudo ufw deny 502/tcp
```

el sistema devuelve un mensaje confirmando el cierre

```
Rule updated
Rule updated (v6)
```

- Desactivar el *proxy*:

```
ur@ursim:~$ sudo ufw disable
```

5.3. RoboDK

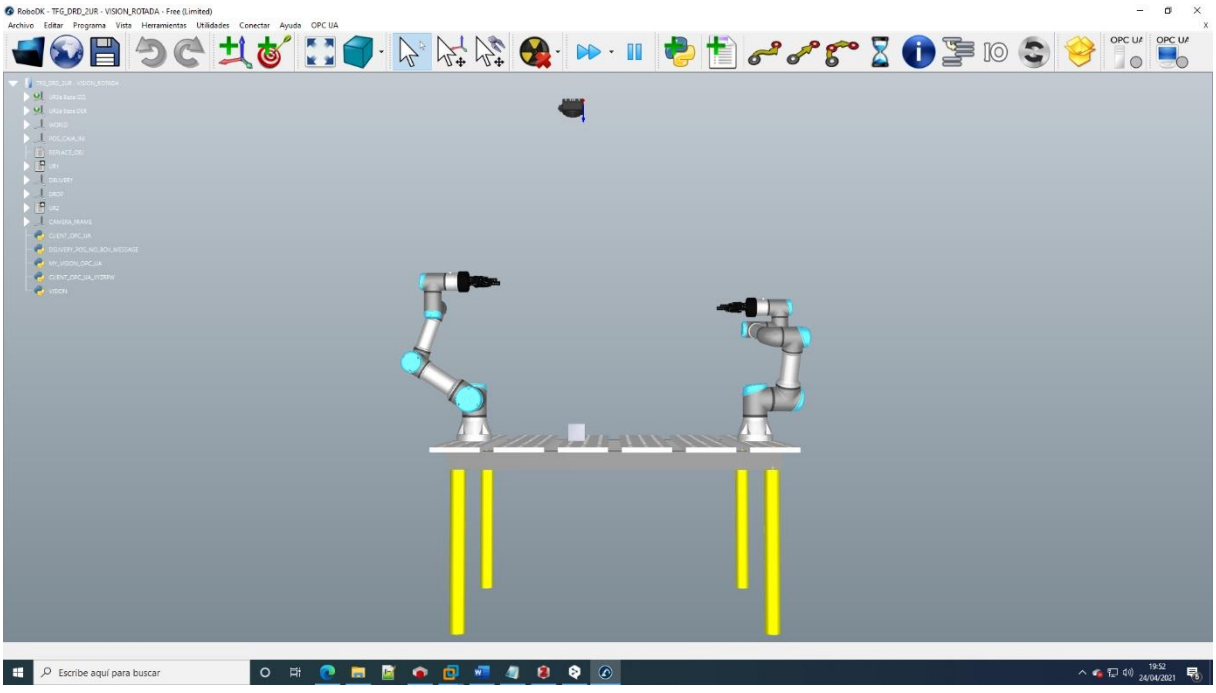
La programación fuera de línea elimina el tiempo de inactividad de la producción causado por la programación en el taller. La simulación y la programación fuera de línea permiten estudiar múltiples escenarios de una célula de trabajo robotizada antes de configurar la célula de producción. Los errores que se suelen cometer al diseñar una célula de trabajo pueden predecirse a tiempo.

Dichos errores se solventan en este proyecto empleando *RoboDK* como simulador gráfico que permitirá crear toda la secuencia descrita en los objetivos del proyecto y una vez completada, el mismo programa generará los archivos necesarios para que los robots reales puedan ejecutar las instrucciones creadas de forma *offline*.

La estación virtual en *RoboDK* se construye empleando objetos almacenados en la biblioteca del mismo programa, la cual incluye (Figura 61):

- Mesa (1).
- Robot *UR3e* (2).
- Herramienta *Robotik 2F-140*, *UR_Opened* (2), *UR_Closed* (2).
- Cámara *Matrox Iris GTR* (1).
- Pieza *Cubo* (1).

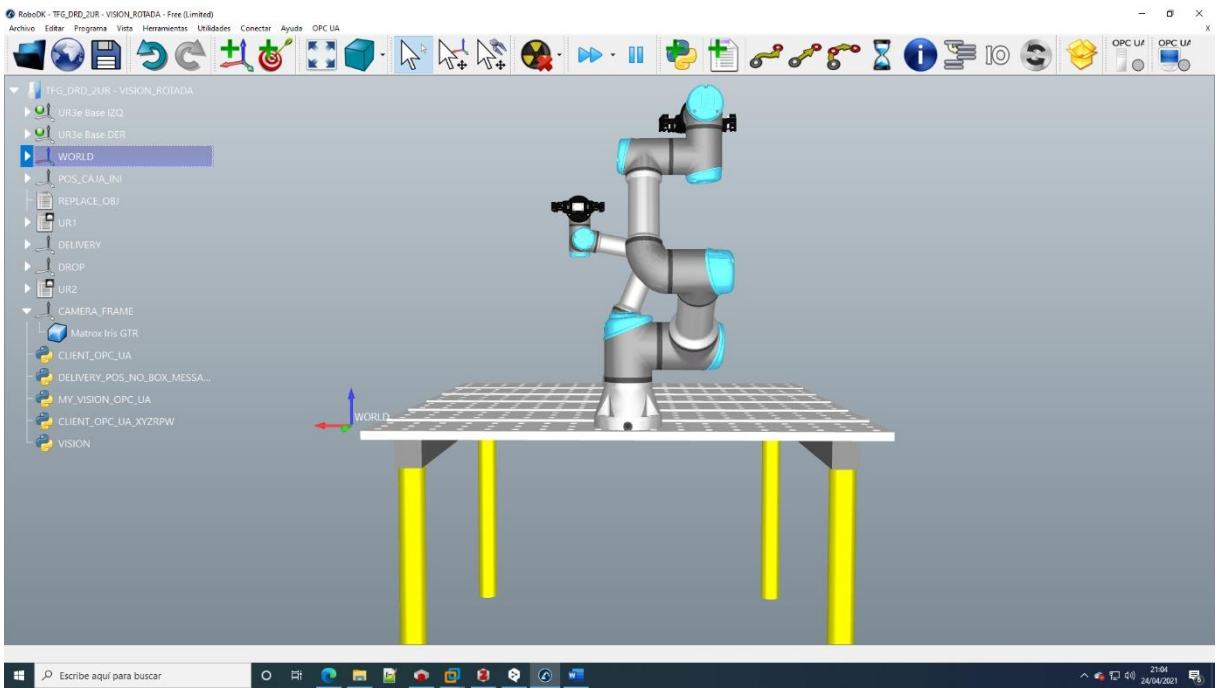
Figura 61. Estación creada en RoboDk con todos los elementos de la celda real para su simulación.



Los elementos dentro de *RoboDK* se emplazan utilizando los distintos sistemas de referencias, por defecto, al crear una nueva estación como la mostrada en la (Figura 61), existe el sistema de referencia denominado *WORD*, con respecto al cual se pueden representar las posiciones de todos los elementos

de la celda. Sirve como punto de partida para el cálculo de las posiciones y todo lo que se describirá relativo a la visión artificial implementada (Figura 62).

*Figura 62. Representación del sistema de referencia **WORD** sobre el cual se emplazan todos los elementos de la celda.*



Cada robot posee además su propio sistema de coordenadas en la base, además, en la (Tabla 8) se muestra la relación de sistemas de coordenadas creados para dar solución a la programación de la secuencia, así como los elementos asociados de forma inicial a cada uno.

Tabla 8. Relación de los sistemas de coordenadas existentes en la celda y los elementos asociados inicialmente a estos.

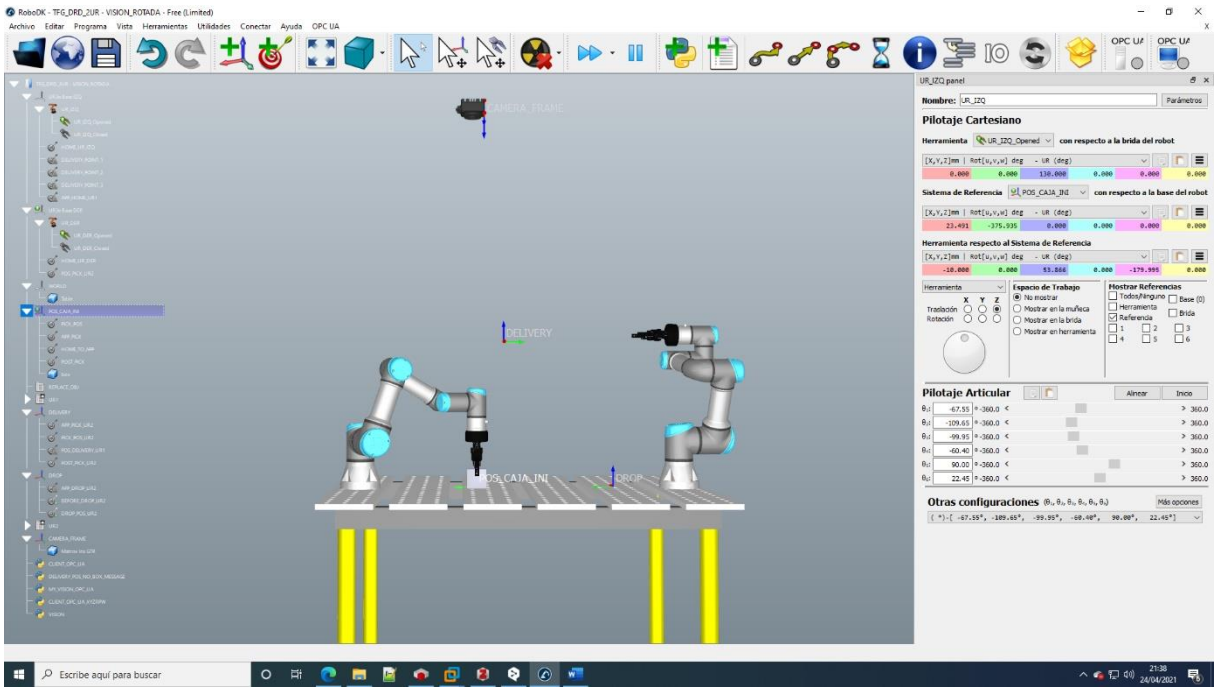
Sistema de Referencia	Elementos Asociados
UR3e Base IZQ	**UR IZQ**, TCP (UR_IZQ_Opened, UR_IZQ_Closed) y [Puntos]
UR3e Base DER	**UR DER**, TCP (UR_DER_Opened, UR_DER_Closed) y [Puntos]
WORD	**Table** (Mesa de Trabajo)
POS_CAJA_INI	**box (**Pieza que intercambian los UR) y [Puntos]
DELIVERY	[Puntos] (Asociados a la posición de entrega de la caja)
DROP	[Puntos] (Asociados a la posición de dejada de la caja por UR2)
CAMERA FRAME	**Matrox Iris GTR** (Cámara)

Como se puede observar en la (Figura 62) *RoboDK* dispone, a la izquierda de la interfaz, de un árbol de proyecto que permite ir agregando instrucciones, secuencias, programas, señales, dispositivos, entre otros elementos.

La lógica de trabajo es ir moviendo cada robot hasta los puntos necesarios y almacenar en una tabla las posiciones de cada eje para realizar el control de la secuencia en el *PLC* como se comentó anteriormente (*Anexo I: Tabla de control de posiciones*), para llevar a cabo esta tarea, se presiona *doble*

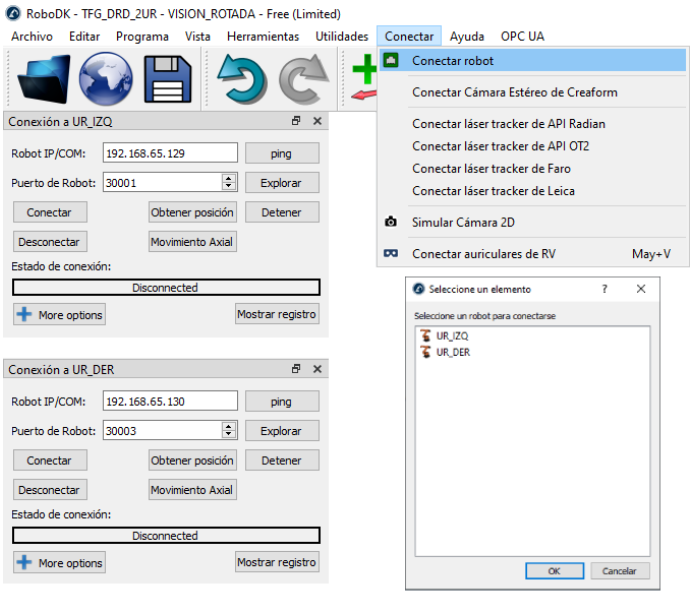
clic sobre *UR1* o *UR2* para desplegar la ventana de control de movimiento cartesiano asociada (Figura 63) y se copian los valores de los ejes de cada punto.

*Figura 63. Representación del árbol del proyecto con todos los elementos de la celda y la ventana de control de posición cartesiana de **UR1.***



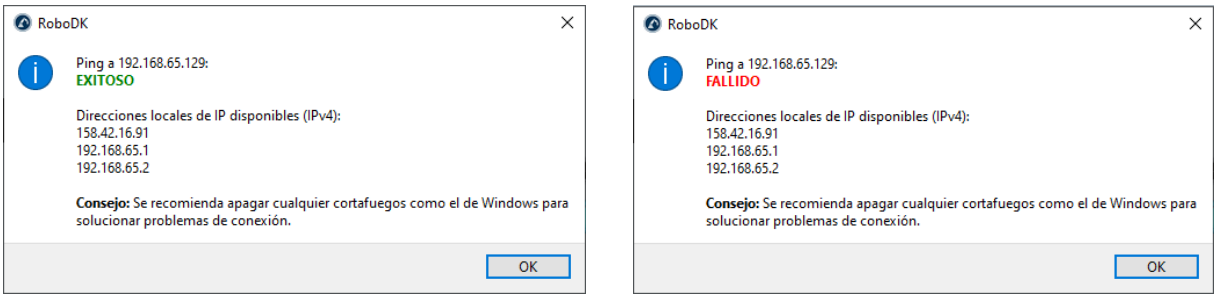
Primeramente, antes de incluir ningún programa o instrucción, el reto consistió en la comunicación de *RoboDK* con los robots que se están simulando mediante *URSim*, para desarrollar este objetivo, en el menú principal del programa, opción *Conectar*, muestra una ventana en la que se selecciona el robot que se quiere vincular (Figura 64):

*Figura 64. Secuencia para establecer la conexión de los dos robots de la estación con **URSim.***



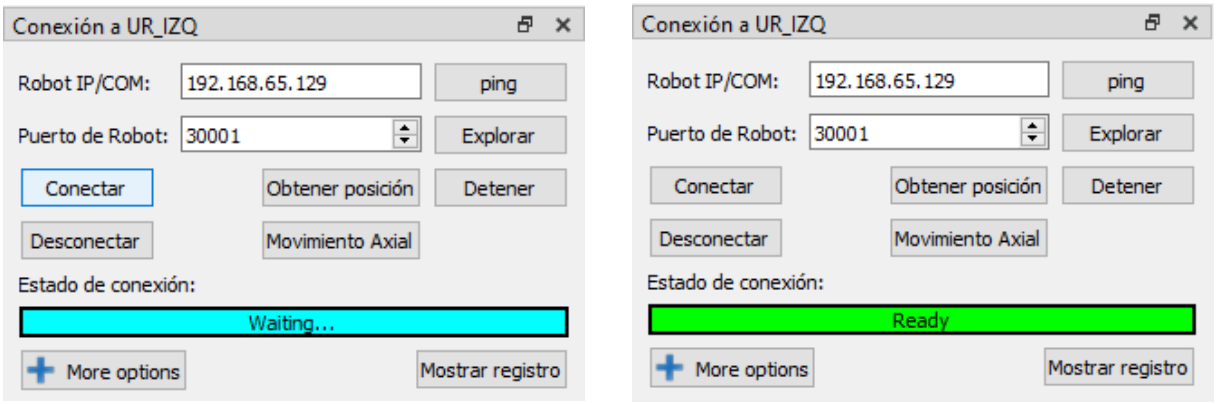
Para conectar los *UR* a través de uno de los puertos ***RTDE*** antes mencionados, de forma arbitraria, *RoboDK* utiliza el *30001* para comunicarse con *UR1* y el puerto *30003* para el enlace con *UR2*. El parámetro imprescindible en la ventana de configuración de comunicación es la *dirección IP* del robot, definida anteriormente por *DHCP*, para comprobar si el *proxy* en *URSim* está activo con las excepciones de los puertos correctamente configurada, se puede presionar el botón ***ping*** en la ventana obteniendo un resultado *EXITOSO* o *FALLIDO* (Figura 65).

Figura 65. Comprobación de la comunicación de cada robot en RoboDK con la máquina virtual de URSim correspondiente.



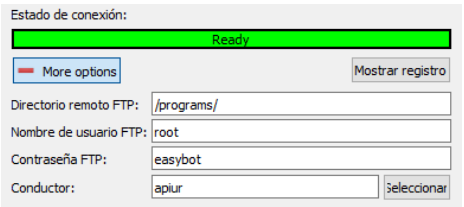
Si el resultado indica que se ha realizado un *ping EXITOSO* a la máquina virtual con la *dirección IP* indicada, solo resta establecer la conexión presionando el botón *Conectar*, ante lo cual, la barra de estado muestra un color azul con la palabra *Waiting…* y posteriormente, si la conexión queda establecida, se actualiza a *Ready* (Figura 66).

Figura 66. Proceso de conexión entre el robot UR1 de la estación y el robot UR1 de URSim.



Con los dos robots conectados, todos los movimientos que realicen los robots en *RoboDK*¸ serán reproducidos en *URSim* por los robots virtuales, de esta forma se solventa el objetivo: ***Simular la aplicación en un entorno gráfico CAM que permita observar de forma "real" el funcionamiento de la celda.*** Existe una opción disponible en la ventana de *Conexión a UR* que permite enviar los programas de los robots mediante protocolo *ftp* a los robots "reales" conectados a *RoboDK (*Figura 67), sin embargo, queda como posible punto de mejora del proyecto.

*Figura 67. Configuración para el envío de los programas vía **FTP** a los robots.*



5.3.1. RoboDK API

"La API de RoboDK[16] (Application Program Interface) son un conjunto de rutinas y comandos que RoboDK expone a través de un lenguaje de programación. La API de RoboDK le permite programar cualquier robot utilizando uno de los lenguajes de programación compatibles, como C#, Python o C++. También puede automatizar tareas repetitivas utilizando la API de RoboDK. En comparación con la programación de robots específicos de un proveedor, con la API de RoboDK es posible simular y programar cualquier robot utilizando un lenguaje de programación único/universal, como ***Python****.*

Una vez instalado RoboDK, utilizando la configuración por defecto, ya tiene instalado Python y está listo para su uso. Python es un lenguaje de programación de alto nivel ampliamente utilizado para la programación de propósito general. La mayoría de los ejemplos y documentación del RoboDK están basados en programas de Python.

La API consiste en un protocolo TCP/IP implementado en el RoboDK. Cuando el programa se está ejecutando se comporta como un servidor y expone un conjunto de comandos a través del protocolo TCP/IP.

La API del RoboDK puede utilizarse para las siguientes tareas:

1. *Automatizar la simulación: Crear macros para automatizar tareas específicas dentro del simulador RoboDK, como mover objetos, marcos de referencia o robots.*

2. *Programación fuera de línea: Programar robots fuera de línea a partir de un lenguaje de programación universal. RoboDK generará programas específicos para un controlador de robot concreto cuando se utilice la API (como un programa en Python o C#). El programa del robot se genera según el post-procesador seleccionado para un robot específico.*

3. *Programación en línea: Programación de robots en línea mediante un lenguaje de programación universal: Es posible mover los robots y recuperar su posición actual desde la API de RoboDK. El programa manejará los robots utilizando los controladores de robot.*

En otras palabras, el mismo programa que se utiliza para la simulación (1) puede utilizarse para generar programas de robot (2, Programación fuera de línea) y para mover el robot en tiempo real (3, Programación en línea).

La API de RoboDK para Python se divide en los siguientes dos módulos:

- ***Módulo Robolink*** *(robolink.py): Es la interfaz entre RoboDK y Python. Cualquier objeto en el árbol de la estación de RoboDK puede ser recuperado mediante un objeto Robolink y está representado por el elemento objeto. Es posible realizar diferentes operaciones en ese elemento según la clase Robolink.Item.*

[16] Véase RoboDK API - RoboDK Documentation

— ***Módulo RoboDK*** *(robodk.py): Es una caja de herramientas robóticas para Python que permiten operar con transformaciones Pose y obtener ángulos de Euler para diferentes proveedores de robot. Todos los post-procesadores dependen de este módulo RoboDK."*

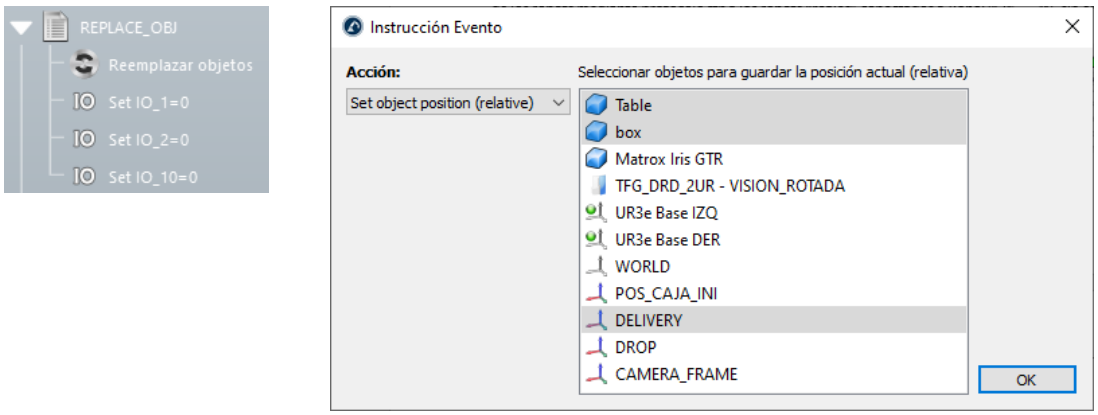
5.3.2. RoboDK SECUENCIA

El conjunto de elementos que forman parte de la secuencia creada está constituido fundamentalmente por los puntos relativos a cada robot y que permiten el *Pick and Drop* de la caja, así como el intercambio de esta. En segundo orden, la próxima unidad en importancia, lo constituyen los programas creados y que incluyen a su vez, las instrucciones de movimiento así como subrutinas.

A continuación se describen los programas existentes en el árbol del proyecto, ver (Figura 63, página 62):

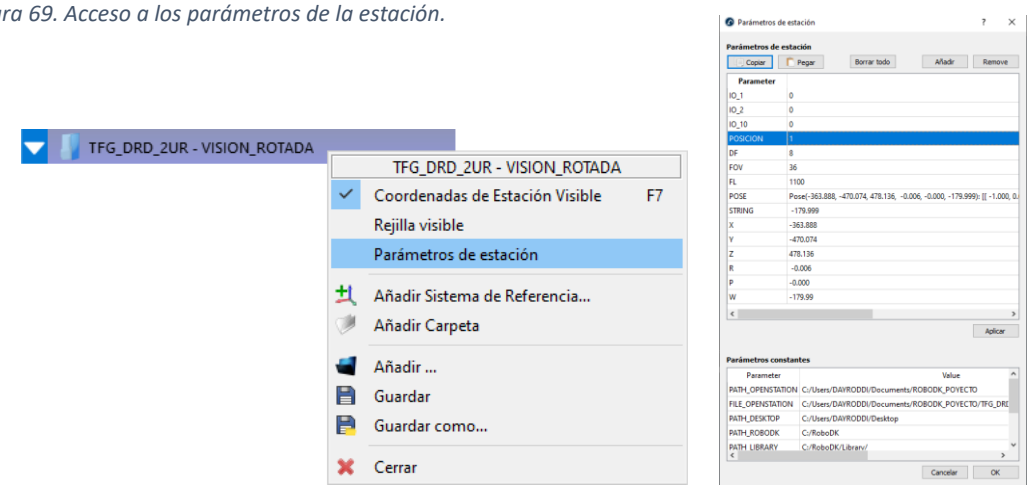
- ***REPLACE_OBJ***: Programa que incluye una llamada a un evento de tipo [*Set object position (relative)*], es el encargado de guardar la posición del sistema de coordenadas *DELIVERY*, y los objetos *BOX* y la *TABLE*, este programa se ejecuta al final de la secuencia para reestablecer las posiciones iniciales de los objetos mencionados y de esta forma queda la estación configurada para un nuevo ciclo. (Figura 68).

Figura 68. Programación del evento que inicializa las posiciones de los objetos en la celda.



- ***PROGRAMAS Python***: La estación debe ser capaz de intercambiar señales y datos con *CODESYS* para el control del proceso de visión artificial implementado, así como el flujo de datos relativos a la configuración de la cámara y los resultados del ciclo de medición. Para desarrollar estas tareas es necesario declarar en la celda los parámetros requeridos, dichas variables son accesibles presionando *clic derecho* sobre la estación y seleccionando *Parámetros de la estación* (Figura 69).

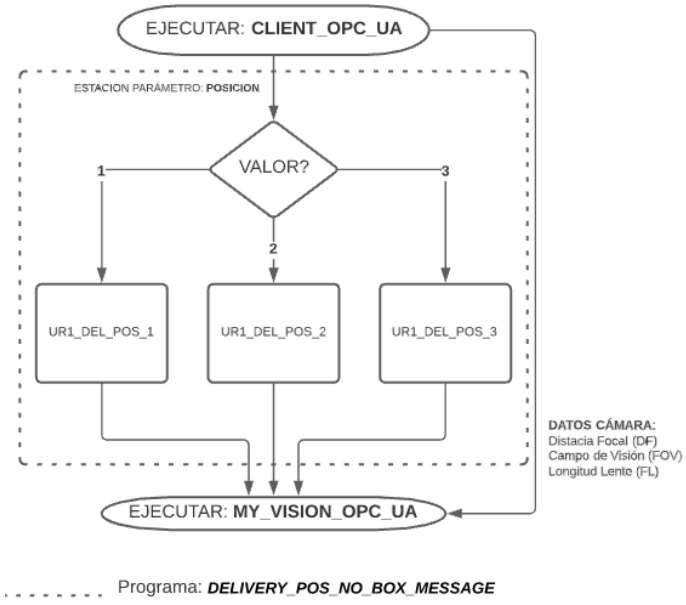
Figura 69. Acceso a los parámetros de la estación.



— ***CLIENT_OPC_UA***: Programa escrito en lenguaje *Python* para establecer la comunicación *OPC_UA Client* entre *RoboDK* y *CODESYS,* en el cual se leen las variables relacionadas con los tres parámetros de la cámara así como la posición en la cual *UR1* debe entregar la caja a *UR2*, el código desarrollado está basado en el repositorio extraído de Python-opcua/client-minimal.py at master · FreeOpcUa/Python-opcua · GitHub (*FreeOpcUa/Python-Opcua*, s. f.). (*Apartador Manual de programación. A_2.3.1. CLIENT_OPC_UA.py*)

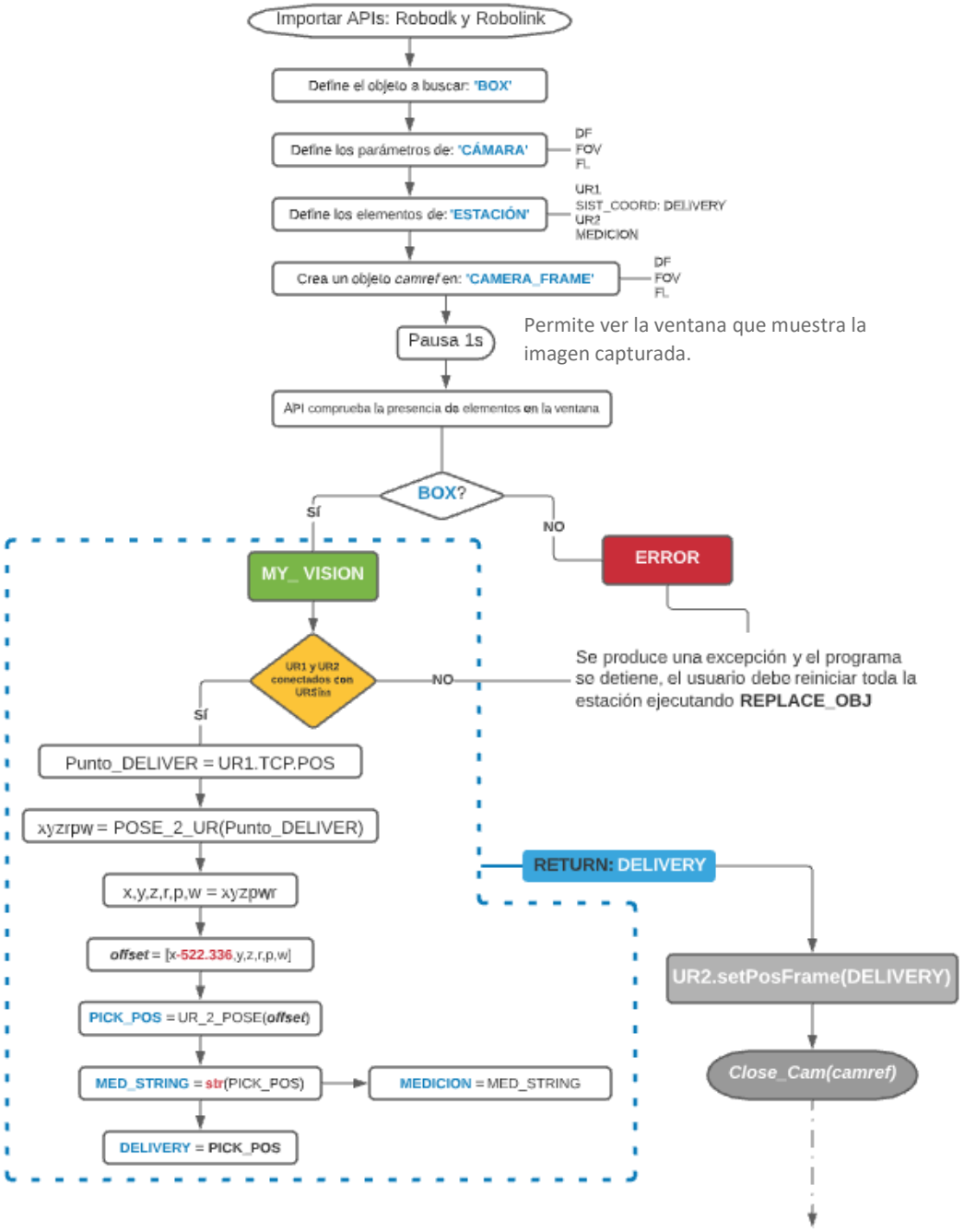
— ***DELIVERY_POS_NO_BOX_MESSAGE***: En un inicio se implementó una solución que involucraba la aparición de una ventana donde se preguntaba al usuario en que posición debía *UR1* entregar la pieza en la estación, al cambiar el procedimiento y recibir dicho dato a través de *OPC UA*, este programa solo analiza el valor almacenado en el parámetro *POSICION* de la estación y de acuerdo con el número contenido (1, 2 ó 3) ejecuta una sencilla lógica para decidir si *UR1* entrega la pieza en la posición asociada a cada número. La (Figura 70) muestra un diagrama de flujo con el principio de funcionamiento de este programa.

Figura 70. Diagrama de flujo de la lógica implementada para la discriminación de la posición de entrega de la pieza.



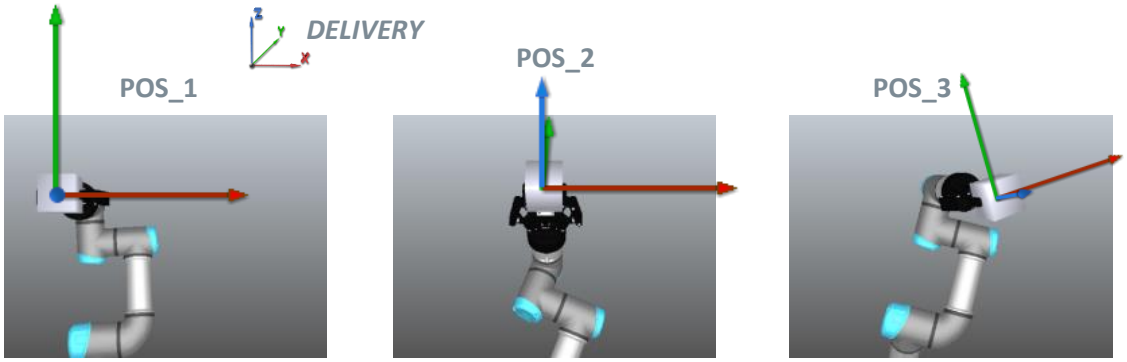
— ***MY_VISION_OPC_UA***: Es el programa encargado de desarrollar toda la lógica de visión artificial. Con ayuda de las *API* de *RoboDK* se activa una cámara interna en la estación enlazada a un sistema de coordenadas para ajustar la posición de origen de la visión, en el caso de este proyecto, el sistema de coordenadas es *CAMERA_FRAME* en el cual se fijó la cámara *MATROX Iris GTR*, realmente solo representa un modelo gráfico del dispositivo real del cual toma su geometría y denominación. Con los parámetros recibidos por *OPC UA* para la configuración de la visión y los pasos que se presentan en el siguiente diagrama de flujo, el autor da por cumplido el objetivo: ***Diseñar un algoritmo para la visión artificial que permita el intercambio de la pieza.*** (Figura 71).

Figura 71. Diagrama de flujo que describe la lógica del programa de visión artificial.



El principio de funcionamiento del código es sencillo, a través de las funciones implementadas en las *APIs* se detecta si hay elementos dentro de la ventana capturada, en caso afirmativo debe existir la caja (***BOX***), si también se cumple dicha condición y ambos robots se encuentran enlazados a robots reales, en el caso del proyecto, las máquinas virtuales de *URSim*, se almacena en una variable (*Punto_DELIVERY*) el valor del *PCH* de *UR1*, posteriormente se utiliza la función (*POSE_2_UR*) para convertir el valor de las coordenadas del *PCH* en un tipo de datos de posición necesario para que el post-procesador pueda interpretar la posición como un punto con los parámetros de los robots *UR,* los parámetros se dividen en las respectivas coordenadas *X, Y, Z, R, P y W* para aplicar un *offset* en *X* debido al desplazamiento del sistema de coordenadas ***WORD***, respecto al sistema de coordenadas de la estación, la variable *PICK_POS* contiene el valor real del punto que coincide con el punto central de la caja (***BOX***) en la estación, a su vez, el sistema de coordenadas ***DELIVERY*** de la estación coincide con el punto de entrega de *UR1*, resumiendo, la lógica de visión lo que permite es actualizar la posición del sistema de coordenadas ***DELIVERY*** según la posición de entrega de *UR1***,** ya que los puntos de aproximación, recogida y retirada de *UR2* están programados en base a este sistema de coordenadas, de esta forma, *UR2* siempre acude a la posición de *UR1*. La función ***MY_VISION*** devuelve el valor de la posición del sistema de coordenadas para pasarlo como parámetro a la estación. En la (Figura 72) se muestra una recreación del proceso de actualización de ***DELIVERY*** según la posición de entrega de *UR1.*

*Figura 72. Actualización de la posición del sistema de coordenadas **DELIVERY** según la posición de entrega de UR1.*



POS_1 es una posición de entrega "básica" (no contiene rotación en ninguno de los ejes), sin embargo, *POS_2* está desplazada en *X* con respecto a *POS_1* y además, tiene una rotación sobre el mismo eje, en el caso de *POS_3* existe rotación en los tres ejes y un desplazamiento relativo a *X* referido a *POS_2.*

Las posiciones de entrega de *UR1* están prefijadas y solo dependen de la elección del usuario, se ha intentado simular una planta donde un robot entrega una pieza atendiendo a las exigencias del proceso y gracias a la visión, el segundo puede acceder al punto de recogida y producirse el intercambio.

La cámara implementada en *RoboDK* a través de las *API* es una cámara 2D, lo que imposibilita detectar rotaciones de piezas en el espacio, es por ello por lo que el autor recurre al recurso de utilizar la posición del *PCH* como base de la visión artificial.

- ***CLIENT_OPC_UA_XYZRPW***: Es el último de los programas desarrollados para obtener la secuencia planteada al inicio, al igual que su homólogo *CLIENT_OPC_UA,* establece una comunicación donde intercambia el valor de las variables que contienen los datos de la medición para representarlos en el *WebServer* del *PLC*.

Como se puede apreciar en la (Figura 71, página 67) se hace una conversión del dato ***PICK_POS*** a una cadena de texto denominada ***MED_STRING***, dicha cadena almacena en formato de texto, los parámetros que componen las coordenadas del sistema de referencia ***DELIVERY*** como se muestra en la (Figura 72).

*Figura 72. Variable **MED_STRING** que almacena en formato texto la posición del sistema de referencia **DELIVERY.***

Parameter	Value
IO_1	0
IO_2	0
IO_10	0
RoboDK	RoboDK 64 bit v5.2.1.19954
time	03/15/2021 18:35:34.686.000.000
SimulationSpeed	10
Station	TFG_DRD_2UR - VISION
POSICION	3
DF	6
FOV	30
FL	1000
MED_STRING	Pose(-605.773, -470.039, 480.993, -9.313, -24.131, -163.836): [[-0.877, 0.254, -0.409, -605.773], [-0.338, -0.929, 0.148, -470.039], [-0.342, 0.268, 0.901, 480.993], [0, 0, 0, 1]]
STRING	-163.836
X	-605.773
Y	-470.039

El programa recorre la cadena para extraer los primeros seis datos relativos a las coordenadas *X, Y, Z, R, P* y *W* del sistema de coordenadas y enviar dichas variables secuencialmente en el mismo orden que se ha descrito al *PLC*, como se definió en el apartado ***VISION*** del autómata una vez se detecta la recepción de *W* que corresponde con la rotación del punto respecto al eje *Z (Rz)* se concluye el proceso de visión y se desconecta el enlace *OPC UA.*

5.3.3. PROGRAMAS UR1 – UR2

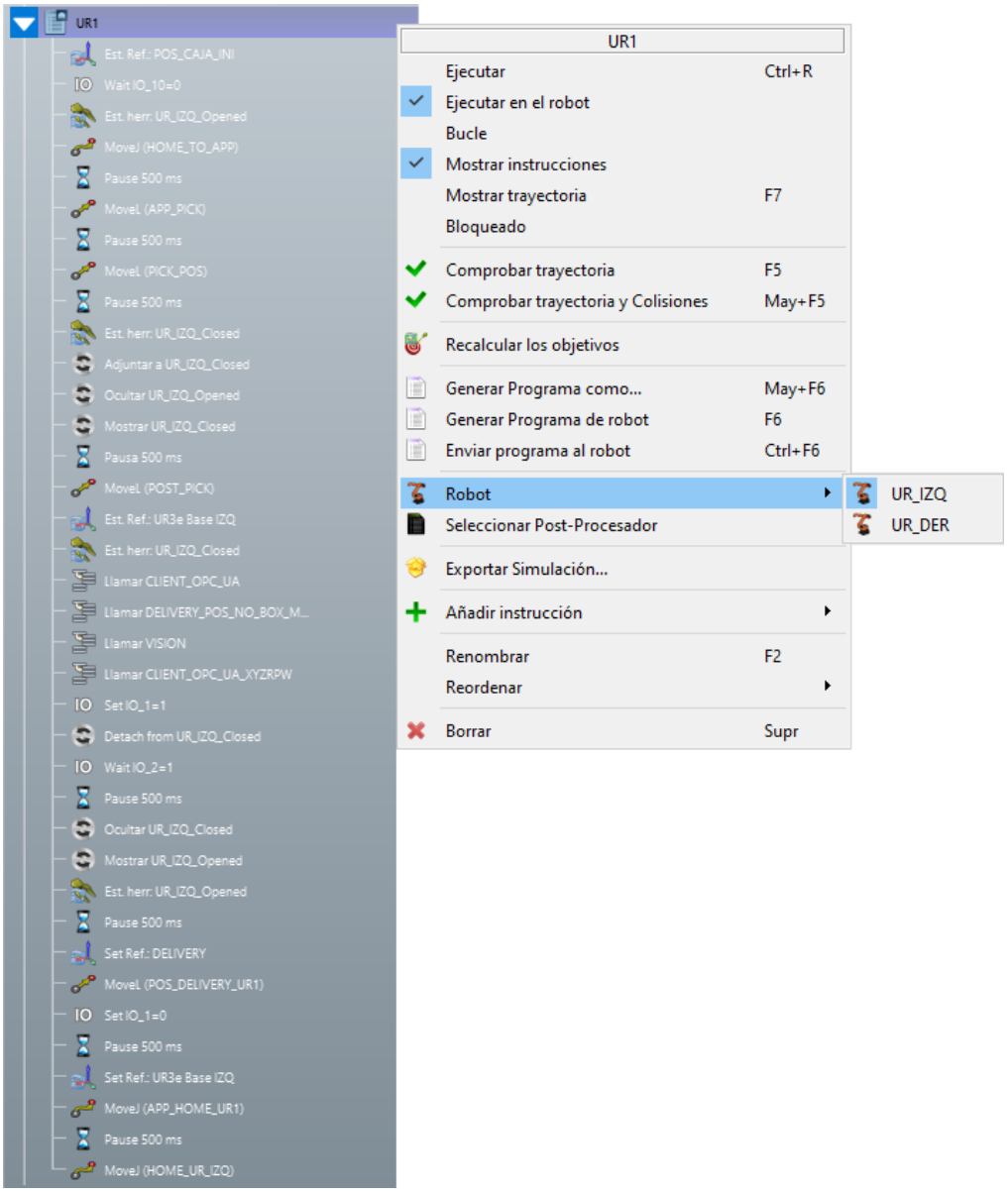
Cada robot ejecuta un conjunto de instrucciones, movimientos, activación de señales, entre otros elementos propios a la tarea que deben desempeñar, es por ello por lo que en la estación existen dos programas denominados *UR1* y *UR2*.

Cuando el programa contiene todos los elementos necesarios para que el robot desempeñe el trabajo necesario, es requisito indispensable indicar al programa cuál será el robot que ejecutará dicho código, o lo que es lo mismo pero tiene más sentido, realmente se está enlazando el programa en concreto a un robot de la estación. El procedimiento para llevar a cabo esta tarea consiste en dar *clic derecho* sobre *UR1* (*UR2*) y seleccionar en la ventana emergente *Robot -> UR_IZQ* para el caso de *UR1* y *Robot -> UR_DER* para *UR2* respectivamente.

- Existen un conjunto de opciones extras relativas a los programas y que son de vital importancia en el presente proyecto; para el control del flujo de ejecución y un funcionamiento cíclico, se disponen de dos opciones descritas como ***Ejecutar*** y ***Bucle*** respectivamente (Figura 73) en el

menú secundario del objeto programa. La opción ***Ejecutar en el robot*** garantiza la ejecución paso a paso con retroalimentación en *RoboDK*, este método utiliza el robot como un servidor y cada instrucción se envía al robot contenido en *URSim* paso a paso, ya que se ejecuta en el simulador, lo que garantiza el cumplimiento de uno de los objetivos fundamentales del proyecto: ***Simular la aplicación en un entorno gráfico CAM que permita observar de forma "real" el funcionamiento de la celda.***

Figura 73. Opciones relativas al menú secundario del programa ***UR1.***



La opción marcada ***Mostrar instrucciones*** permite visualizar el conjunto de elementos que conforman el programa *UR1* y que se pueden observar en el árbol del proyecto a la izquierda, si se desactiva dicha casilla, el programa *UR1* oculta todas las instrucciones y de esta forma se "protege" el código programado.

5.3.4. GENERAR PROGRAMAS DE ROBOT UR1 – UR2

Uno de los objetivos del proyecto consiste en: ***Obtener los programas de cada dispositivo simulado para ser cargados en los equipos físicos reales tras realizar una depuración del código en el entorno virtual.***

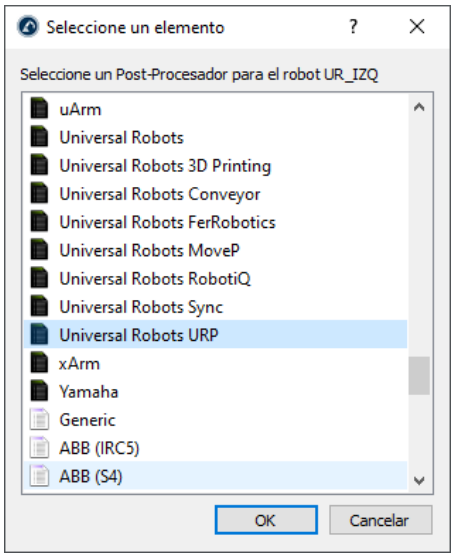
Este es el verdadero sentido de un *Digital Twin*, *RoboDK* tiene un conjunto de herramientas y opciones ideales para completar esta meta marcada en el proyecto. Como se puede observar en la (Figura 73, página 70), debajo de la opción ***Robot***, existe una alternativa para acceder al post-procesador, los cuales "*son un paso clave en la programación fuera de línea porque pueden generar programas de robot para un controlador de robot específico. La programación del robot debe seguir las reglas de programación específicas del proveedor, estas reglas se implementan en el postprocesador. Un postprocesador de robot define cómo deben generarse los programas de robot para un controlador de robot específico.*

La conversión de la simulación del RoboDK a un programa de robot específico la realiza un postprocesador. Cada robot está vinculado a un postprocesador que definirá un estilo de programación de robot específico. El postprocesador se utiliza cuando el programa se genera fuera de línea…" (*Post Processors - RoboDK Documentation*, s. f.)

Para seleccionar el post-procesador que será el encargado de generar los programas de los robots para ser cargados por la controladora existen dos caminos alternativos:

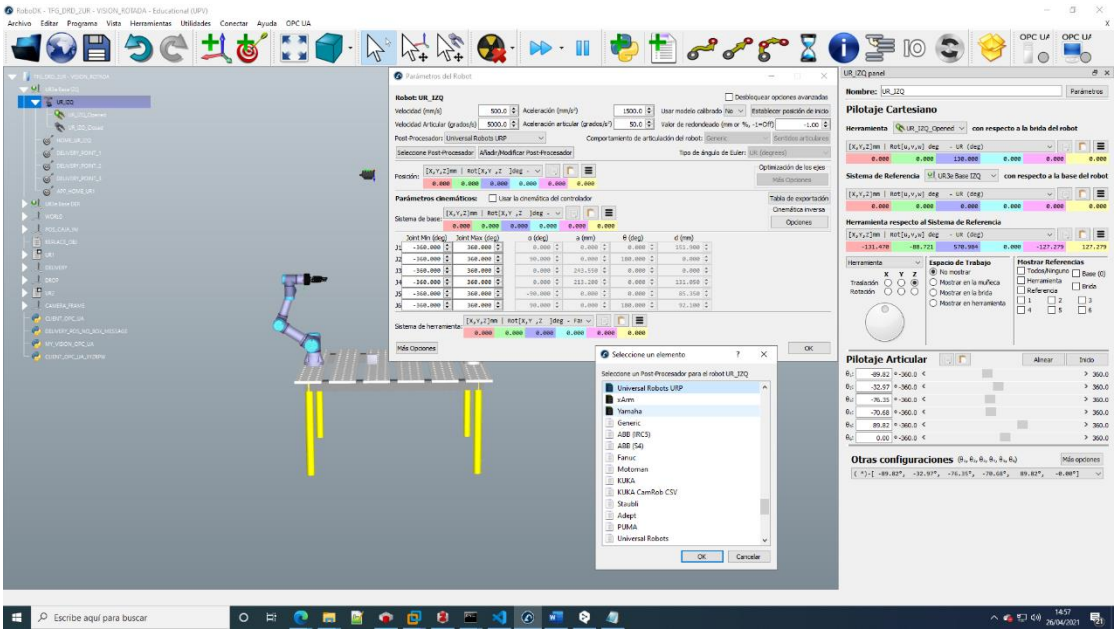
- *Clic derecho* (*Programa **UR1***) -> ***Seleccionar post-procesador***: Esta opción muestra una ventana emergente (Figura 74) donde por defecto se encuentra activo el post-procesador ***Universal Robot***. "*El post-procesador por defecto genera un programa SCRIPT y también un programa URP ejecutando el programa como un archivo script. Este post-procesador está optimizado para funcionar bien con grandes programas (este no es el caso si usamos el post-procesador Universal_Robots_URP)*".

*Figura 74. Selección del post-procesador por defecto para los robots de **Universal Robot.***



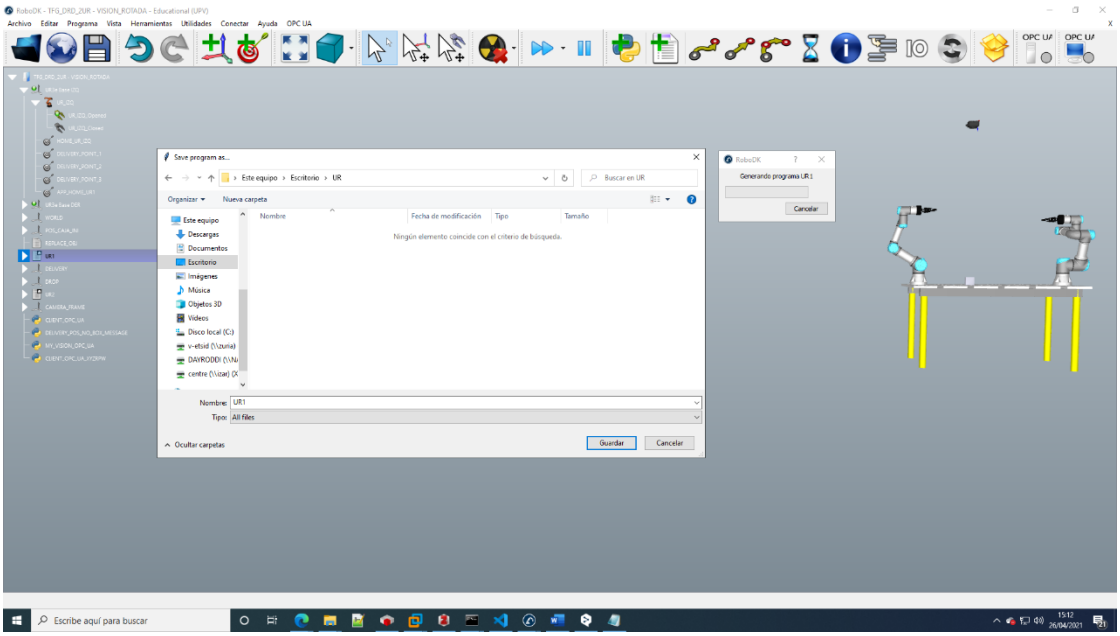
- *Clic derecho* (*UR_IZQ*) -> ***Opciones*** -> ***Parámetros*** (*ventana UR_IZQ panel*) -> ***Seleccione post-procesador*** (*ventana parámetros del robot*) [*esta ventana solo se puede visualizar con una licencia activa del programa*] -> ***Universal Robot URP*** (ventana *Seleccione un elemento*). *"Se debe usar el post-procesador llamado **Universal_Robots_URP** para generar un archivo URP que se pueda modificar fácilmente en el controlador del robot."* (Figura 75) (*Universal Robots - Documentación RoboDK*, s. f.)

*Figura 75. Secuencia de pasos para seleccionar el post-procesador **Universal Robot URP.***



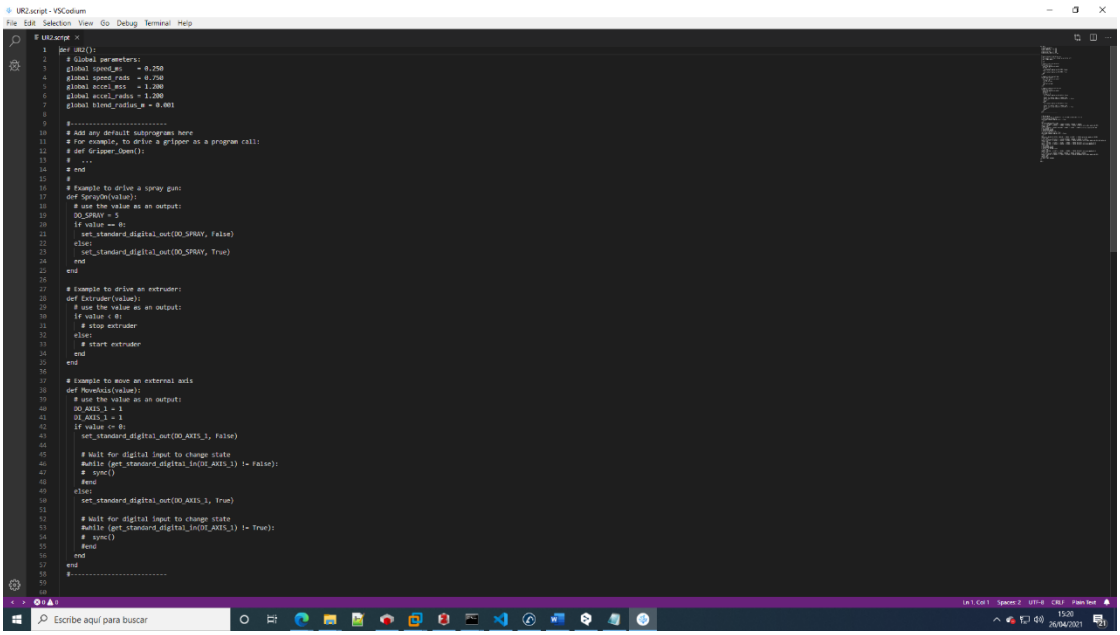
Con el post-procesador seleccionado, solo es necesario un paso más para obtener los programas con formato *URP* necesarios para cargar en la controladora de los robots implementados por las máquinas virtuales *URSim*. Presionando nuevamente sobre el programa *UR1 clic derecho*, seleccionar la opción ***Generar Programa como…*** (Figura 76)

Figura 76. Generando el programa UR1.urp para cargar en la controladora del robot.



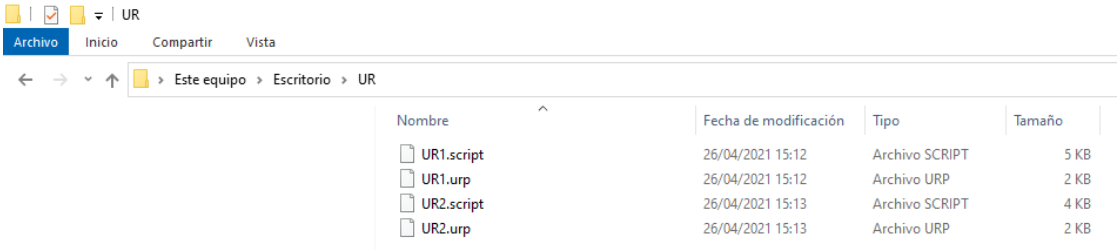
Una vez indicada la ruta para guardar los programas, *RoboDK* muestra el código generado empleando *VSCode*, que es un editor de código abierto multiplataforma creado por Microsoft. Tiene versiones para Windows, Mac y Linux y fue clasificado como el mejor IDE entre todos los tipos de desarrolladores. (Figura 77).

Figura 77. Código relativo al programa UR1.urp mostrado empleando el IDE VSCode.



Como se puede apreciar en la figura anterior, el programa generado es del tipo *.script*, pero en la carpeta indicada previamente para generar los programas, se guardan ambas versiones (Figura 78).

Figura 78. Programas generados por el post-procesador ***UNIVERSAL ROBOT URP.***

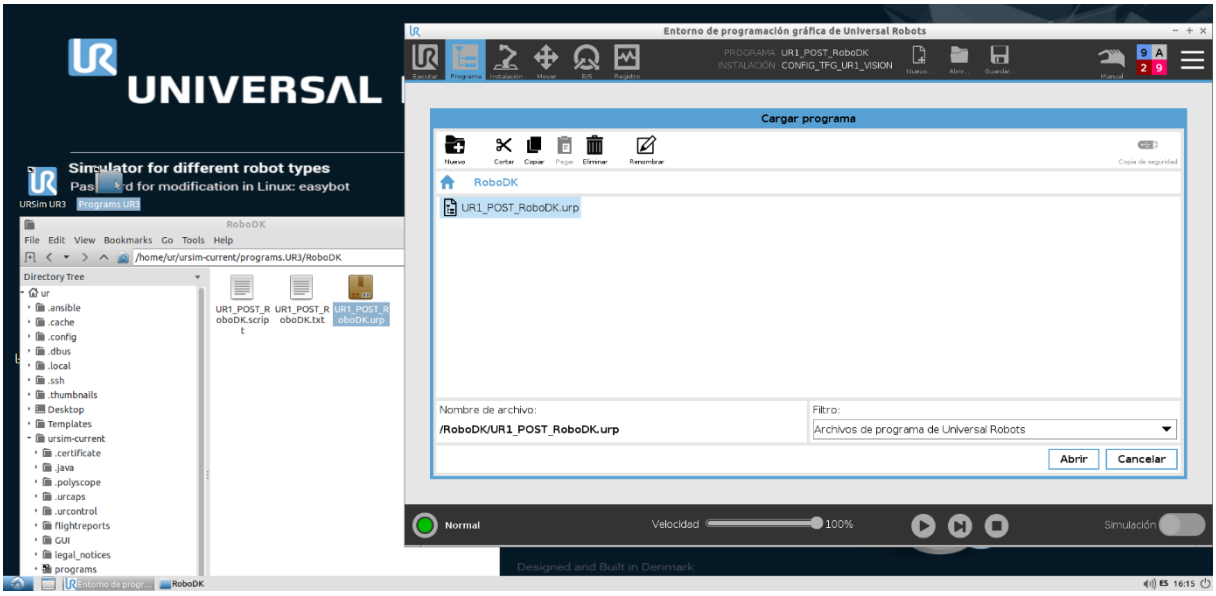


En el apartado ***5.2. ROBOT UR – URSIM*** se describió como estaba configurada la máquina virtual y en la (Tabla 6) se especifica que el *Escritorio* del ordenador *HOST* sería la carpeta compartida entre la cual *URSim* y el *PC* físico compartirían archivos, por este motivo los programas *UR1* y *UR2* se almacenan en la carpeta *UR* en el escritorio, de esta forma, accediendo desde un explorador de archivos en la máquina virtual hasta dicha carpeta, cada controladora puede cargar los programas generados por los post-procesadores tal y como se comentó anteriormente.

Dentro de *URSim* (Figura 51, página 51), en el escritorio, existe un programa *IDE* para cada modelo de robot así como una carpeta para contener los programas generados en la controladora asociado a cada robot. Dentro de la carpeta ***Programs UR3*** se crea una subcarpeta con el nombre *RoboDK* donde se

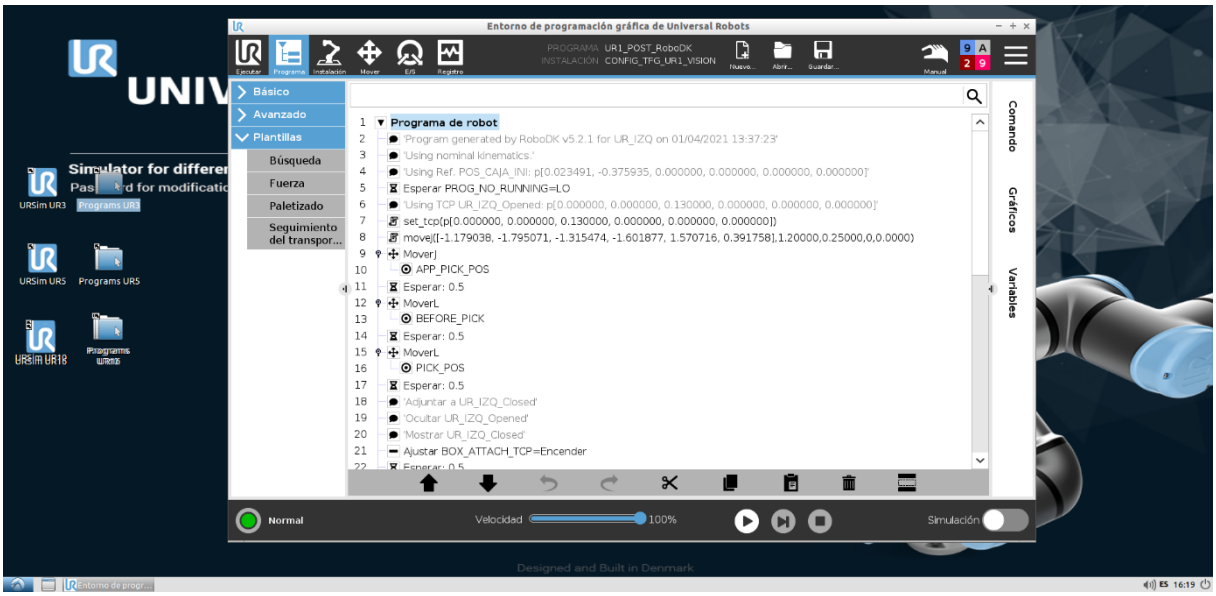
guardará el programa obtenido *UR1.urp* (en el momento de desarrollo de la investigación del proyecto el autor nombró al programa *UR1_POST_RoboDK.urp*) (Figura 79)

Figura 79. Carga del programa UR1.urp en Polyscope.



Si el post-procesador en *RoboDK* pudo generar el programa sin ninguna excepción u error, accediendo al menú ***Programa*** de *Polyscope*, se puede observar el programa cargado, que, como es de esperar coincide en secuencia y orden, con las instrucciones contenidas en el programa *UR1* en *RoboDK.* (Figura 80).

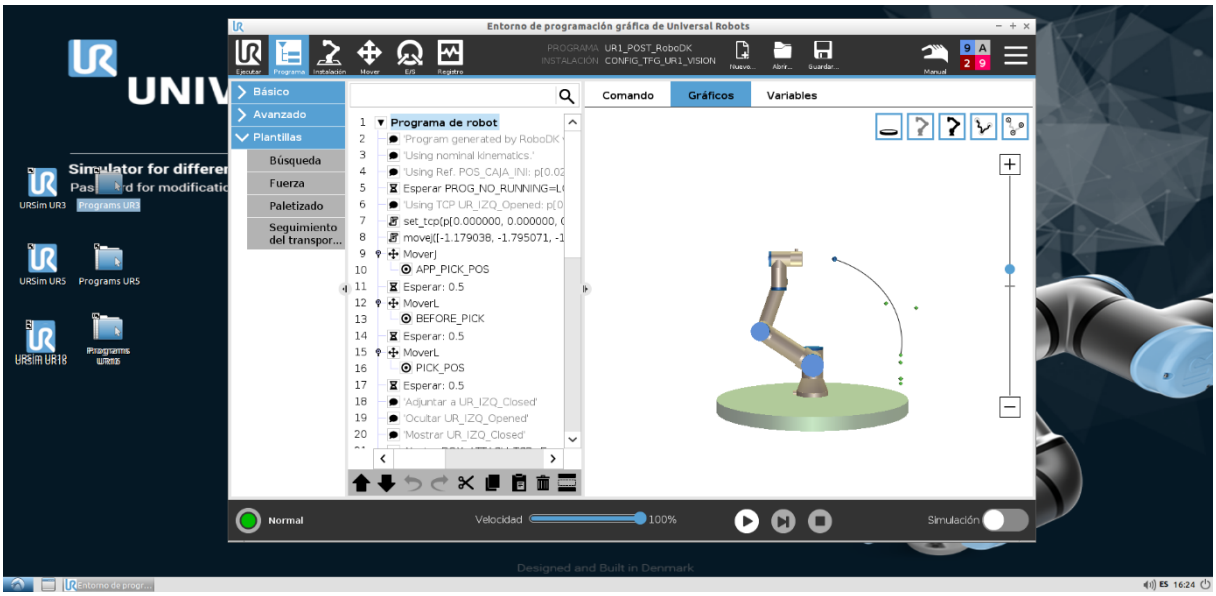
Figura 80. Programa cargado en Polyscope en formato URP.



A la derecha de la ventana donde se observa el programa generado, existen tres sub-ventanas que permiten visualizar, en conjunto con el código, elementos extras como *Comando* (permite insertar secuencias previas al programa principal, fijar valores a una variable inicial o declarar la ejecución del

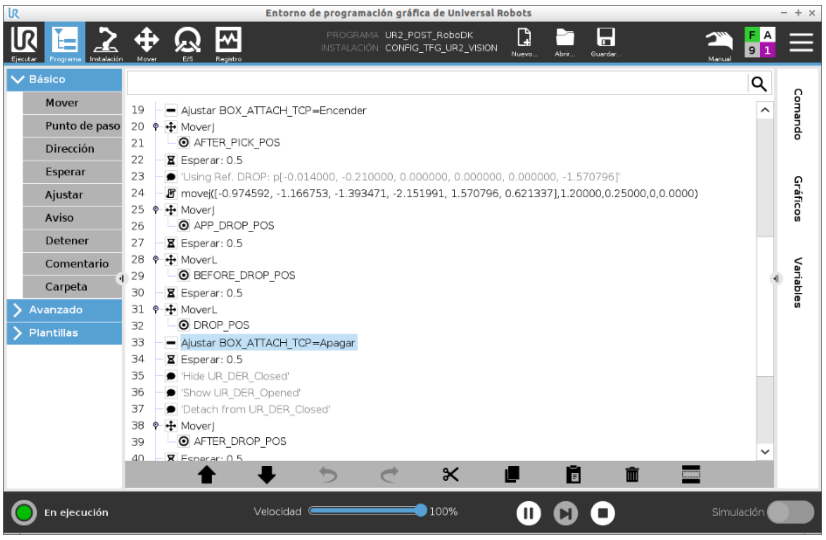
programa de forma cíclica), *Gráficos* (genera una ventana para observar la posición actual del robot mientras se ejecuta un programa, los puntos de paso, trayectorias así como planos) (Figura 81) y por último *Variable* (donde se muestran los valores de las variables contenidas en el programa principal). Como se puede apreciar, sin tener siquiera idea de cómo programar un robot *UR* y utilizando un sencillo lenguaje gráfico en una *IDE* independiente como *RoboDK* se ha podido desarrollar una secuencia compleja de funcionamiento de una celda automatizada con robótica colaborativa empleando visión artificial para el intercambio de una pieza.

Figura 81. Ventana Gráfico para la visualización de los puntos, trayectorias y planos contenidos en el programa.



El único elemento que tuvo que ser modificado en el programa cargado en la máquina virtual, fue la activación de las señales digitales implementadas en la tarjeta *Modbus_TCP Client* para la comunicación con *CODESYS*, en un inicio, dichas instrucciones aparecen en el programa *UR1.urp* como si fuesen un comentario, deben ser sustituidos por instrucciones de manejo de señales digitales tal y como se aprecia en la (Figura 82) relativa a la activación de la señal *BOX_ATTACH_TCP* correspondiente a *UR2* una vez que la herramienta se ha cerrado para capturar la caja.

Figura 82. Sustitución de comentarios por instrucciones de activación/desactivación se señales digitales del robot.



5.4. DISEÑO GRÁFICO

El presente apartado se quiere iniciar con una frase del cofundador de *Apple Steve Jobs*, que justifica la importancia del objetivo desarrollado relativo a: ***Idear un plan que abarque una línea de diseño de la interfaz[17] y la experiencia de usuario[18] con una presentación depurada, funcional y equiparable a las actuales tendencias de diseño de aplicaciones digitales.***

> ***"Some people think design means how it looks. But of course, if you dig deeper, it´s really how it works".***
>
> *Steve Jobs*

Actualmente el diseño de interfaces de software es un campo de la tecnología con un crecimiento exponencial, empresas de renombre invierten sumas millonarias por entender el valor de un producto con una interfaz que se ajuste a los deseos de los clientes y permitan la correcta utilización de las herramientas desarrolladas por parte de los usuarios.

El sector industrial aún se encuentra en desventaja en relación con el mundo de la tecnología doméstica o de consumo general. Tal vez sea un factor sustancial lo poco accesible que es una fábrica de ensamblado de coches, la industria cementera o una empresa farmacéutica a personas ajenas a estas instalaciones, incluso dentro de la misma empresa, un sistema puede ser solamente conocido por un puñado de operarios.

En pleno 2021, el diseño de pantallas *HMI* y elementos de representación de la información de los procesos en la industria se encuentra desactualizado, en la mayoría de los casos las empresas no cuentan con un departamento que atienda estas necesidades o elaboren un protocolo o algoritmo de diseño que cumpla determinados estándares aprobados para unificar bajo un mismo concepto, toda la imagen gráfica de la empresa de carácter interno.

Uno de los objetivos marcados en la confección del presente proyecto, tal como se comentó al inicio del apartado, es la presentación de un plan de diseño tanto de la experiencia de usuario como la interfaz de usuario que asegure una representación, interacción y relación de todos los datos y

[17] La **interfaz de usuario** (**UI**) es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo. Véase Interfaz de usuario - Wikipedia, la enciclopedia libre

[18] La **experiencia de usuario** (**UX**) es el conjunto de factores y elementos relativos a la interacción del usuario con un entorno o dispositivo concretos, dando como resultado una percepción positiva o negativa de dicho servicio, producto o dispositivo. Dicha percepción depende no solo de los factores relativos al diseño (*hardware*, *software*, usabilidad, diseño de interacción, accesibilidad, diseño gráfico y visual, calidad de los contenidos, buscabilidad o encontrabilidad, utilidad, etcétera), sino de aspectos relativos a las emociones, sentimientos, construcción y transmisión de la marca, confiabilidad del producto, entre otros. Véase Experiencia de usuario - Wikipedia, la enciclopedia libre.

procesos incluidos en la plataforma con una presentación depurada, funcional y equiparable a las actuales tendencias de diseño de aplicaciones digitales empleando una pantalla *HMI* o el servicio alojado en el *PLC WebServer.*

Para lograr el desarrollo de este objetivo, el autor consultó una extensa fuente de material asociada al tema, incluyendo la normativa actual, sin embargo, se quiere hacer hincapié en la documentación aportada por *SIEMENS* que ofrece un ***Cuaderno de Trabajo de Diseño de HMI*** (descarga gratuita) que "*permite un desarrollo más rápido, mayor calidad y una interfaz de usuario que inspira a los futuros usuarios de las aplicaciones. El exclusivo cuaderno de diseño de HMI le guía paso a paso hacia la HMI óptima."* (*HMI Design Workbook*, s. f.)

Como se puede comprobar, fabricantes de pantallas como *Siemens* están elaborando paquetes de componentes propios con nuevos estándares que se acerquen a un diseño más contemporáneo y funcional en un intento de abandonar la práctica de emplear pocos recursos para el diseño de pantallas; el 98 % de las ocasiones es el propio programador de *PLC* quien diseña la interfaz.

A través de los siguientes apartados el autor quiere demostrar como con un poco de interés y buena voluntad, se puede construir desde cero un concepto de diseño con los elementos generales que se necesitan asegurando un aspecto sencillo, elegante y utilitario sin tener formación en diseño industrial ni diseño de aplicaciones.

5.4.1. DISEÑO UX

Todo producto empieza con una necesidad o una idea constituyendo el inicio del proceso; en este apartado se justificará como darle forma a este problema, para ello se definirán funcionalidades, estructura y objetivos para elaborar un diseño que cumpla con las expectativas esperadas y que se encuentre al nivel de la plataforma desarrollada.

El inicio de todo trabajo creativo está constituido por la búsqueda de inspiración, crear un ***Moodboard*** [19] fue el primer paso antes de decidir qué aspecto tendría el diseño seleccionado, para ello se consultaron *Web*s dedicadas al diseño como son:

- Behance: Buscar projects | Fotos, vídeos, logotipos, ilustraciones y marcas en Behance
- Dribbble: Dribbble - Discover the World's Top Designers & Creative Professionals
- Codrops: Codrops | Creative front-end resources and inspiration for web professionals (tympanus.net)
- The Interaction design fundation: UX Design Courses & Global UX Community | Interaction Design Foundation (IxDF) (interaction-design.org)

entre otros recursos, como el libro: ***No me hagas pensar*** de *Steve Krug* (*no_me_hagas_pensar.pdf*, s. f.)**.**

[19] El Moodboard es una herramienta creativa que consiste en una visualización rápida de imágenes y palabras en un mismo soporte, a modo de lluvia de inputs que nos ayuden a preparar el cerebro para la fase de ideación de un proyecto, de ahí lo de inspiración.

Algunos de los elementos tenidos en cuenta en el proceso de creación del *Moodboard* se presentan en la (Figura 83):

Figura 83. Confección de un Moodboard como punto de inspiración para el diseño.

Fuente: Buscar projects | Fotos, vídeos, logotipos, ilustraciones y marcas en Behance

El desarrollo y diseño de un *HMI* es un proceso complejo y de mejora continua que abarca 7 aspectos fundamentales:

- Replanteamiento del diseño
- Comprensión del contexto
- Dibujo de un esquema general
- Diseño de los aspectos visuales
- Diseño de las interacciones
- Probar prototipos
- Estilo y finalización

Todo el proceso desarrollado se ha planteado no como una tarea artística, sino más bien un proceso de artesanía digital, donde poco a poco se va moldeando una idea inicial. El *HMI,* o de igual forma el *WebServer,* es una herramienta más de la celda y se realiza para enlazar el proceso productivo o producto con el operario, por lo tanto, si se mejora el diseño y la funcionalidad del *HMI* se está mejorando la plataforma en su conjunto.

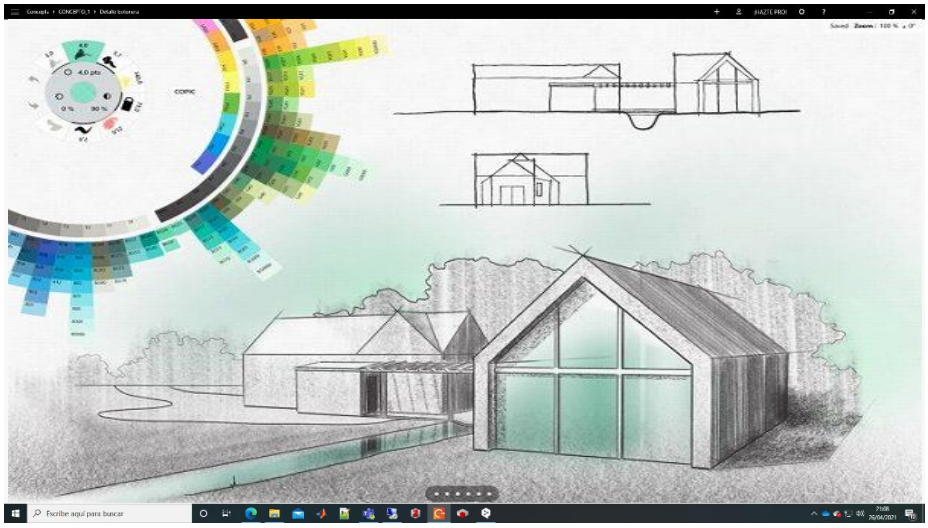
Actualmente es imposible crear algo nuevo, más aún en el sector industrial, por lo tanto es una práctica muy aconsejable adoptar e imitar buenas ideas que se puedan circunscribir al contexto del proyecto. El trabajo de diseño de las pantallas debe ir aparejado de la programación del *PLC*, de esta forma se evitarán posibles errores de funcionamiento, así como una buena interrelación entre la lógica y la representación de los datos. Es necesario que un diseñador sea capaz de ver un paso más allá, o lo que es lo mismo, pueda anticiparse a futuras necesidades de la celda o de los operarios y dote al sistema del carácter flexible necesario para abordar este reto.

Los ***pain points***[20] deben estar presentes en todo el proceso de diseño, si se vigila el cumplimiento de estos factores negativos y la idea o concepto propuesta, solventa dichos puntos, se puede concluir que el diseño propuesto es un éxito.

A modo de ejemplo, si una persona entra en una habitación oscura y desconocida, a medida que pasa la mano por las paredes va encontrando interruptores de luz, enchufes, muebles o cualquier otro elemento, que cada vez que entre en dicha estancia le ayudarán a orientarse y entrenar su posicionamiento espacial. En diseño, estos elementos son conocidos como ***Migas de Pan*** y sirven a los operarios para navegar a través de la interfaz, en este conjunto se encuentran los *menús, botones* ubicados en la misma posición en las distintas ventanas, *paneles* con diseños y morfologías estructurados, entre otros.

Para crear el esquema general de diseño de la interfaz se utilizó el software ***Concepts*** (Figura 84), *Concepts* es una aplicación construida para el diseño, es una versión avanzada de bocetos de papel, donde las herramientas naturales cumplen con la manipulación de vectores para que las ideas puedan cambiar y crecer a medida que lo hace el croquis.

*Figura 84. Herramienta para el bocetado de ideas **Concepts**.*



Fuente: Concepts App • Infinite, Flexible Sketching

Con ayuda de esta aplicación gratuita y un lápiz digital se realiza un ***Wireframe***[21] con las principales pantallas que conformarán la propuesta de diseño final. Como todo proceso de construcción, primero se diseña un modelo y luego se construye la "casa", el objetivo de este boceto es comprobar la funcionalidad y la interacción de los componentes y no el aspecto o el diseño gráfico de estos.

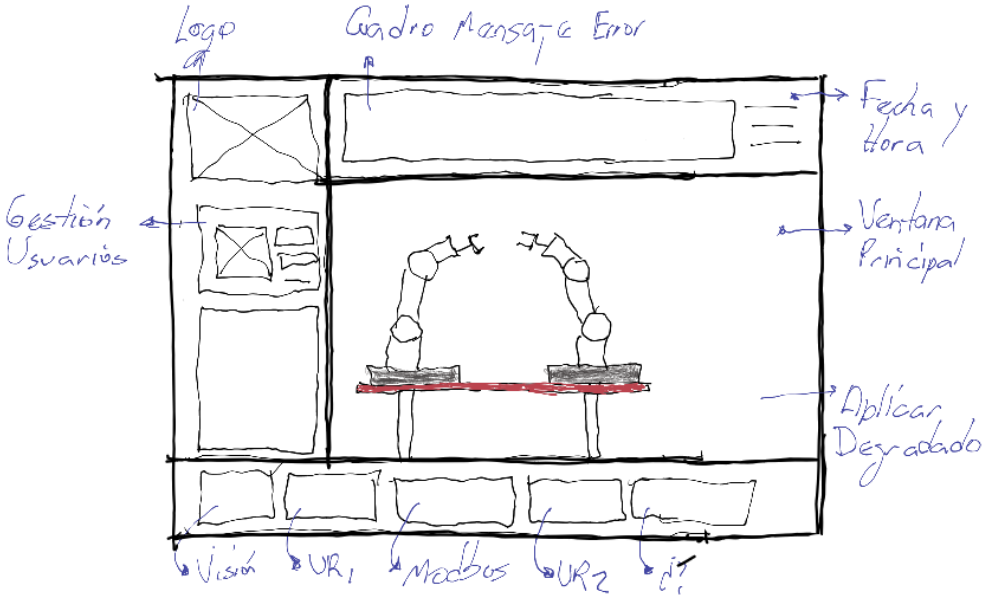
Como se puede observar en la (Figura 85), se desarrolla el primer esquema relativo a la pantalla principal de la interfaz o ***landing page***, donde se muestran los robots en la mesa para indicar que se corresponde

[20] **Pain points** (Puntos de dolor) son necesidades de los clientes o aquellos aspectos que causan preocupación o problemas diversos en el potencial cliente y que en este momento, no sabe cómo superar. Eso no quiere decir que no tenga ya una solución para ello, puede que la tenga pero quizás no sea la más apropiada, la más económica o la mejor solución.

[21] Guía visual que representa el esqueleto o estructura visual de un sitio web. El ***Wireframe*** esquematiza el diseño de página u ordenamiento del contenido del sitio web, incluyendo elementos de la interfaz y sistemas de navegación, y cómo funcionan en conjunto. Véase Website wireframe - Wikipedia, la enciclopedia libre

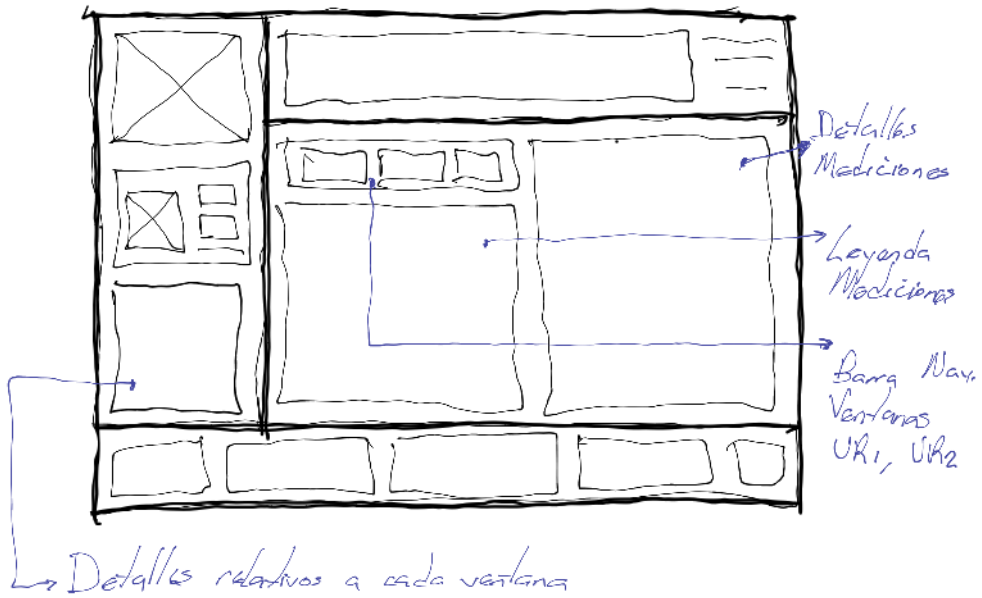
con dicha pantalla, además, se disponen algunos elementos para navegar, como son el menú inferior, el logo del proyecto en la esquina superior izquierda (constituye el botón ***HOME***), una ventana para la gestión de permisos de usuarios y logging, un cuadro de mensajes de errores de la aplicación así como la disposición general de la ventana que constituirá la unidad básica del proyecto.

Figura 85. Boceto de la página principal del diseño del HMI y el WebServer.



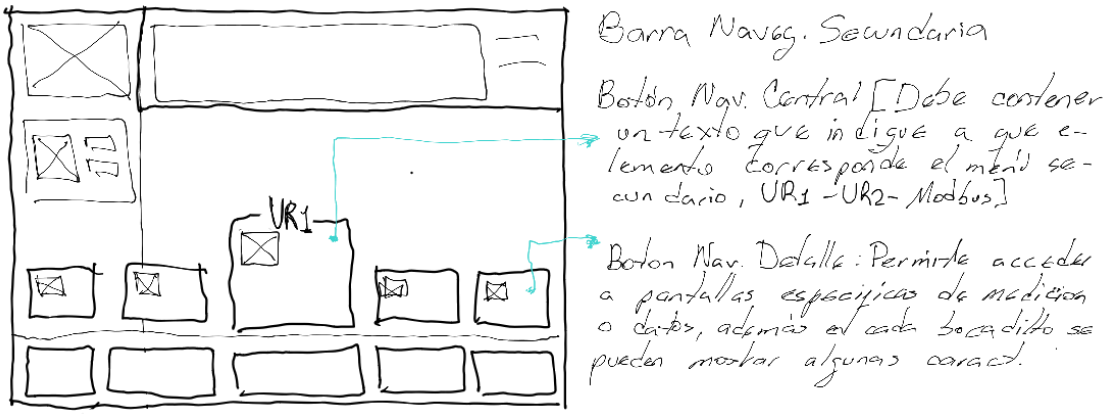
El segundo concepto de pantalla concreta el núcleo del diseño, en esta ventana se representará el mayor conjunto de datos de la aplicación (Figura 86), en ella existen tres paneles en el centro de la vista, correspondientes a una barra de navegación que permite el salto entre ventanas relativas a un mismo robot, mencionando por ejemplo las pantallas de *Programa*, *I/O*, *Ejes* y *TCP*, un cuadro con elementos descriptivos de los parámetros mostrados (*leyenda*) y un panel de representación de datos o *Dashboard.*

Figura 86. Ventana para la representación general de los datos asociados a cada elemento de la celda.



Por último se realiza una nueva vista para diseñar un submenú que permita expandir las opciones asociadas a cada robot y será accesible desde cada pantalla (Figura 87), ampliando de esta forma la accesibilidad de la interfaz y una rápida interoperabilidad del sistema.

Figura 87. Boceto para el diseño de la barra de navegación secundaria de la interfaz.



5.4.2. DISEÑO UI

En diseño de interfaz de usuario, se tiene al cliente en el centro de todo el proceso, pues es para quién el diseñador está elaborando la propuesta de solución gráfica. El actual diseño *UI* se rige por cinco factores que permitirán el desarrollo de una interfaz que cumpla con la normativa vigente y además, permite acercar al lector las nuevas tendencias de esbozo digital:

- ***Claridad***: Elaborar una composición gráfica para un proyecto digital debe dejar siempre claro al usuario algunos aspectos relativos la experiencia en el momento del uso de la aplicación, por ejemplo, siempre debe quedar manifiesto en qué parte del flujo se encuentra, qué puede hacer en la pantalla en la cual se ubica y qué consecuencias tienen dichas acciones. "*La claridad va de la mano con la simplicidad"* (autor desconocido).

- ***Flexibilidad***: El diseño debe ser flexible y se deben tener en cuenta las necesidades específicas de los usuarios que van a utilizar el sistema, condiciones en las que se instalará la pantalla, luminosidad, tamaño del dispositivo, resolución de las imágenes, compartimentado de la información, entre otros factores. Un usuario estará más inclinado a realizar tareas complejas si se dividen en pasos más pequeños.

- ***Familiaridad y Patrones***: Las personas se sienten más cómodas cuando encuentran algo que les resulte familiar en una situación desconocida. Es por ello la importancia de las *Migas de Pan* y el uso de elementos conocidos, o que se repitan a lo largo de todo el diseño, de esta forma el usuario conforma en su mente un mapa de la aplicación.

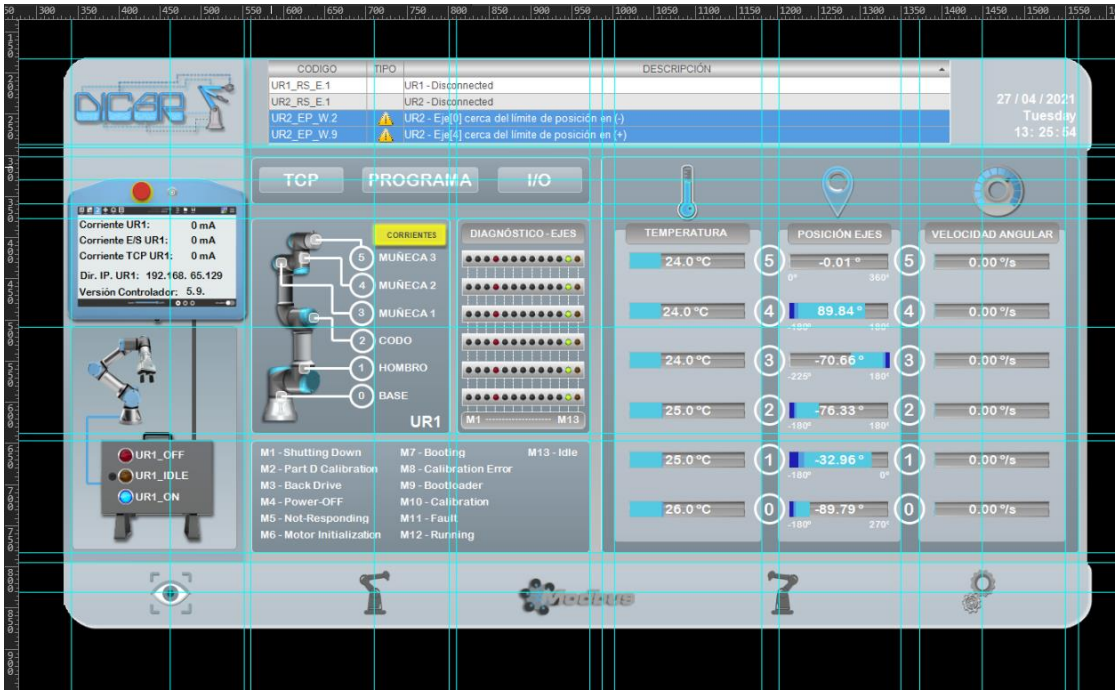
- ***Feedback***: Toda acción debe estar acompañada de una respuesta, el sistema debe indicar al usuario cuando una tarea ha sido ejecutada, un paso realizado o un proceso completado.

- ***Consistencia***: Una propuesta gráfica debe ser consistente, este concepto se desdobla en repetir los elementos empleados con un mismo patrón en todo el conjunto de pantallas, de igual forma la posición y el espaciado debe guardar relación en las distintas presentaciones, así existe una sensación de continuidad.

Teniendo en mente los 5 factores asociados al *UI* se presentan los siguientes apartados que permiten desarrollar la propuesta de solución al diseño de las pantallas del *HMI* y el *WebServer.*

— ***Composición***: El sistema de escritura occidental establece una lectura de izquierda a derecha y de arriba abajo, esta práctica influencia muchas áreas de la vida cotidiana, en diseño gráfico se tiene muy en cuenta este fenómeno y por ello la presentación de la información en el proyecto comienza guardando esta relación, a la izquierda de la ventana principal se detallan los elementos sobre los que se presentan los datos y a la derecha, en la misma ventana, se encuentran los contenedores con los datos recopilados y procesados para su visualización. Además, el concepto debe tener una estructura donde los elementos se presenten de forma organizada (Figura 88), con una jerarquía o estructura que facilite la comprensión de la información y una organización que redunde en una comodidad visual de los usuarios.

Figura 88. Empleo de líneas guías para la colocación de los elementos en las pantallas de forma organizada.

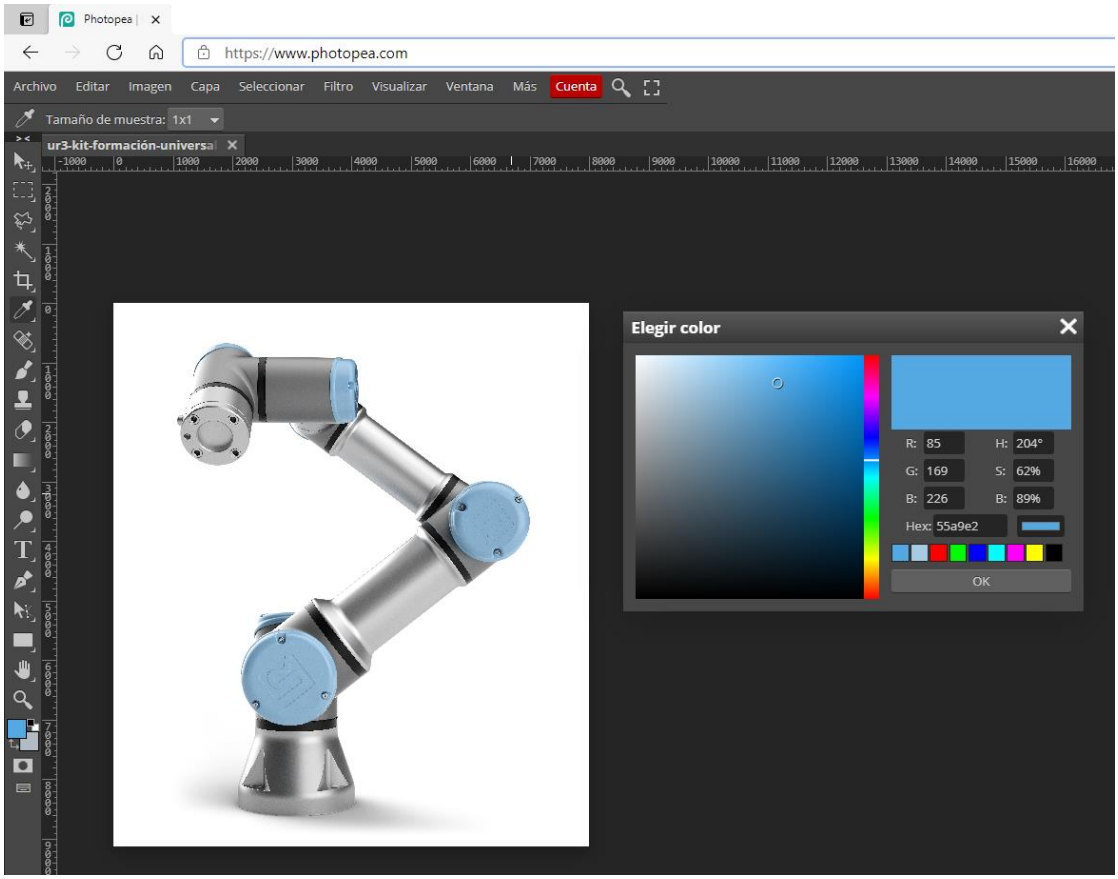


El autor consultó la teoría del espacio de ***Gestalt***[22] con sus leyes y principios generales para comprender mejor el origen de las percepciones a través de los estímulos, mencionando por ejemplo la *ley de figura y fondo* o la relativa al *contraste* y *jerarquización.*

[22] La psicología de la Gestalt (también psicología de la forma o psicología de la configuración) es una corriente de la psicología moderna, surgida en Alemania a principios del siglo XX. Véase Psicología de la Gestalt - Wikipedia, la enciclopedia libre

— ***Colores***: La teoría del color cuenta con múltiples estudios basados en la influencia que los diferentes matices de color tienen sobre el estado de ánimo, carácter y sobre el modo en que se aprecian los objetos que rodean a una persona. La vista es el sentido más desarrollado del cuerpo humano a través del cual llega hasta el *80 %* de la información que procesa el cerebro, es por ello por lo que la elección de la gama de colores empleada en el diseño gráfico del *WebServer* y el *HMI* fue un proceso realizado con detenimiento y rigor. El concepto para elaborar una paleta de colores en sí es sencillo, haciendo uso de un software de edición de imágenes[23] y con una foto de un robot *UR3e* (Figura 89), se selecciona un punto del robot para conocer el valor alfanumérico del color, de esta forma, una sola imagen brinda una paleta cromática que evita convertir el proyecto en un cuaderno de colores.

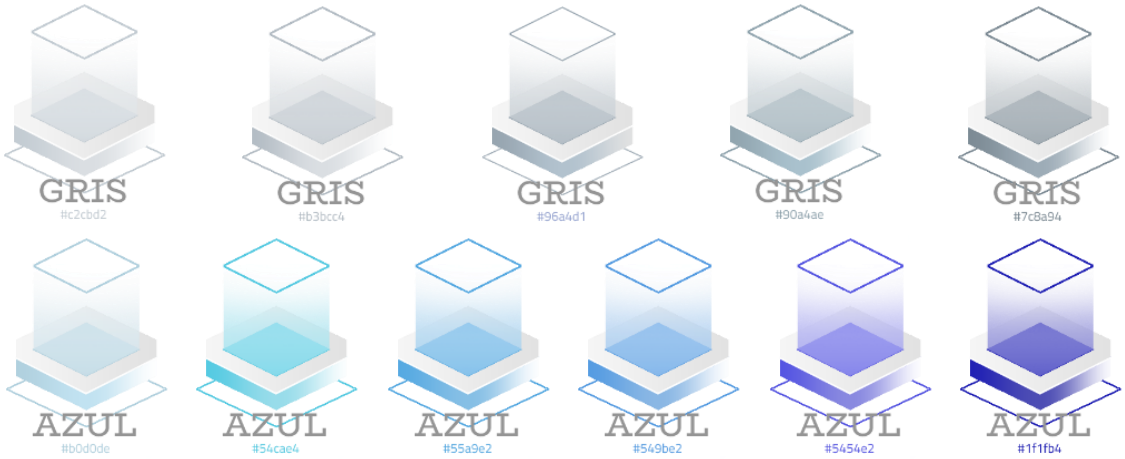
Figura 89.Configuración de la gama de colores azules del proyecto utilizando el selector de color.



Los colores en un proyecto deben estructurar visualmente lo que se ve, guiar al ojo y hacer que las interrelaciones sean más rápidamente identificables. Como se podrá apreciar a continuación, las gamas de colores elegidas coinciden completamente con los colores empleados por la empresa ***Universal Robot*** para la fabricación de sus robots (Figura 90), de esta forma se consigue una consistencia y homogeneidad visual del proyecto.

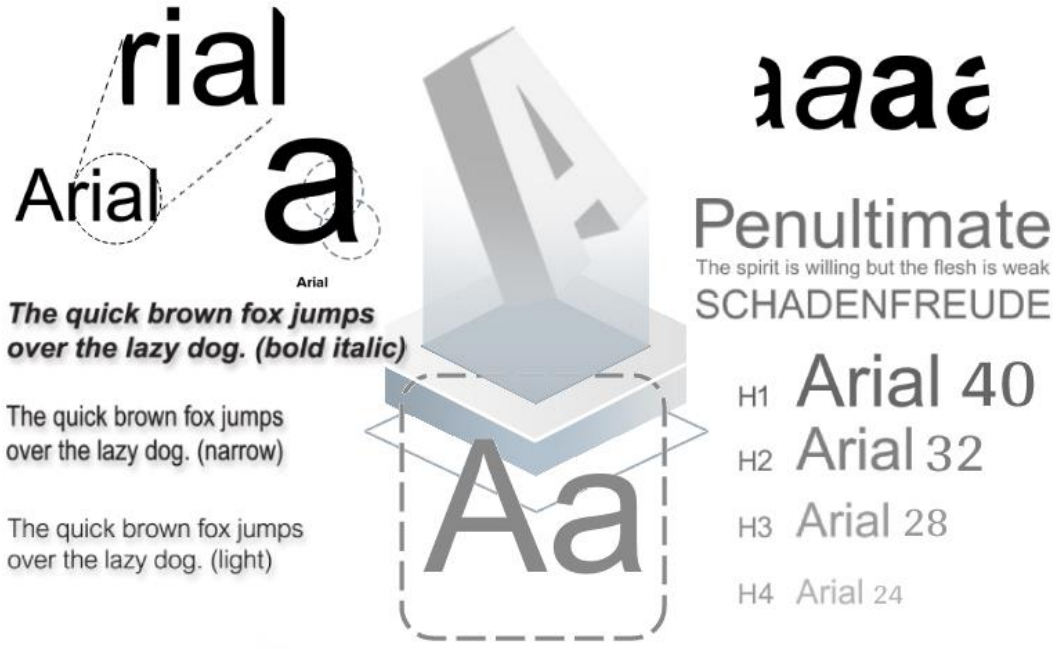
[23] Todas las imágenes del proyecto fueron tratadas con **Photopea**. Véase Photopea | Online Photo Editor

Figura 90. Infografía o mapa de colores empleados en el diseño.



— ***Tipografía***: "***Arial*** en ocasiones expresada o mostrada como ***ArialMT*** en determinado software es un tipo de letra *Sans Serif* presente en varias aplicaciones de *Microsoft*. Debido a que la *Arial* viene incluida en el sistema operativo *Windows*, se ha convertido en una de las tipografías más populares del mundo y tiene un marcado carácter legible." («Arial», 2020). Es por ello por lo que ha sido la tipografía seleccionada en todo el diseño , tanto del *WebServer* como el *HMI* (Figura 91).

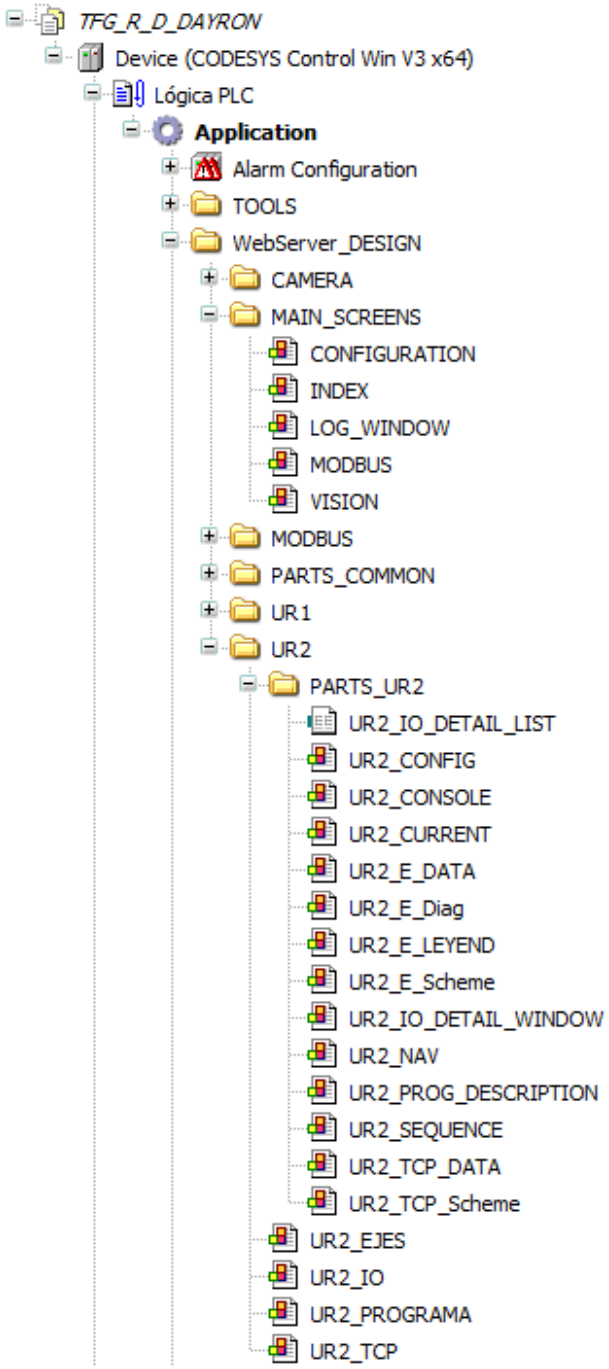
*Figura 91. Presentación de la tipografía **Arial** seleccionada como fuente en el proyecto.*



5.5. WEBSERVER

El concepto de diseño está desarrollado y solo queda poner en práctica los criterios analizados dentro de *CODESYS* para implementar el *WebServer*. Regresando al *PLC* (Figura 92), debajo de la carpeta *TOOLS* que contiene toda la lógica de tratamiento de secuencias, datos, representación de la información, animaciones e implementación de las comunicaciones, se crea una nueva carpeta denominada ***WebServer_DESIGN***, como ya se ha comentado, la jerarquía se presenta a través de subcarpetas que dividen el trabajo en fragmentos más pequeños y fáciles de depurar y modificar.

*Figura 92. Organización de la estructura del **WebServer** en el árbol del proyecto en CODESYS.*



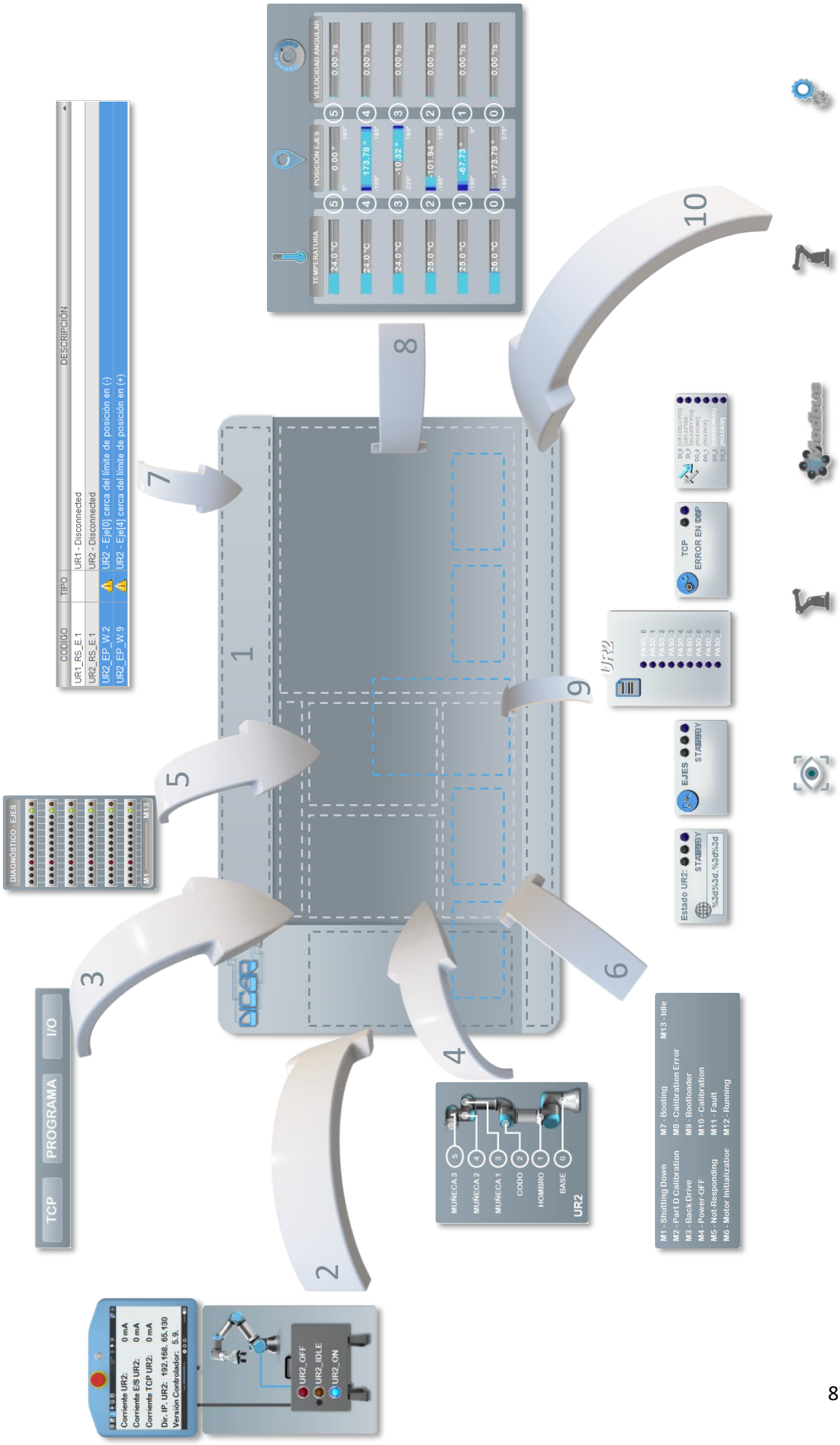
Para explicar el proceso de desarrollo del *WebServer* se escoge como muestra el objeto visualización (***UR2_EJES***), de esta forma el lector puede comprender de una forma sencilla y gráfica como la segmentación de la información tiene un sinfín de ventajas que se mencionan a continuación:

- Permite desmigar el trabajo convirtiéndolo en un proceso más sencillo.
- Se crean módulos que pueden ser instanciados tantas veces se deseen liberando de esta forma la memoria del *PLC.*
- En caso de detectar algún error, este se soluciona en el panel principal correspondiente y automáticamente todas las instancias se actualizan, ahorrando un considerable volumen de trabajo.
- El código tiene una estructura jerárquica y más accesible.
- Los objetos (paneles) diseñados para un componente inicial pueden ser copiados y reutilizados, reestableciendo los valores de las variables asociadas a sus propiedades, dotando al diseño del carácter flexible antes mencionado.

La (Figura 93) de la página siguiente muestra un esquema de cómo se conforma la ventana donde se visualizan todos los datos recibidos por *Modbus* desde la máquina virtual *URSim UR2* relacionados con los ejes del robot, dicha pantalla se divide en los siguientes módulos:

1. **Ventana principal**: Está constituida únicamente por una imagen el formato "*.png"* que actúa como elemento contenedor de los distintos módulos, tiene una resolución de 1280x720 *pixeles*, suficiente para poder ser visualizada en la mayoría de los navegadores actuales.
2. **Consola UR2**: Corresponde a una simulación gráfica de la controladora del robot, en ella se representa el estado de funcionamiento de este, así como una animación donde se puede observar de forma inmediata si se encuentra encendido, apagado o en *standby*. Al mismo tiempo se localiza la dirección *IP*, la corriente consumida por el robot, el módulo de entradas/salidas, la herramienta y por último la versión del controlador.
3. **Menú navegación**: Permite el desplazamiento entre las distintas pantallas relativas a *UR2,* (*TCP, EJES, I/O* y *Programa*).
4. **Leyenda**: Gráfico que describe los distintos ejes del robot así como su posición.
5. **Diagnóstico de ejes**: Tabla resumen para el diagnóstico de cada eje del robot, en ella se puede identificar inmediatamente si algún eje en concreto presenta error.
6. **Leyenda de diagnóstico**: Constituye una tabla para la identificación de las distintas posiciones de los leds del panel anterior.
7. **Tabla de alarmas**: Elemento que visualiza de forma inmediata en todas las ventanas las *alarmas*, *warnings* o *info* activas. Es el principal elemento de diagnóstico de la celda.
8. **Panel de datos**: En este módulo se representan todos los datos recibidos relativos a los ejes, mencionando la posición, temperatura y velocidad angular de cada uno.
9. **Menú secundario**: Representa información general de cada apartado del robot así como permite una mayor interoperabilidad del sistema.
10. **Menú principal**: Diseñado para la navegación a través de todo el *WebServer*, es el elemento que facilita el acceso a las ventanas de *visión artificial, diagnóstico de las comunicaciones*, etc.

*Figura 93. Conformación de la pantalla **UR2_EJES** empleando los distintos módulos*

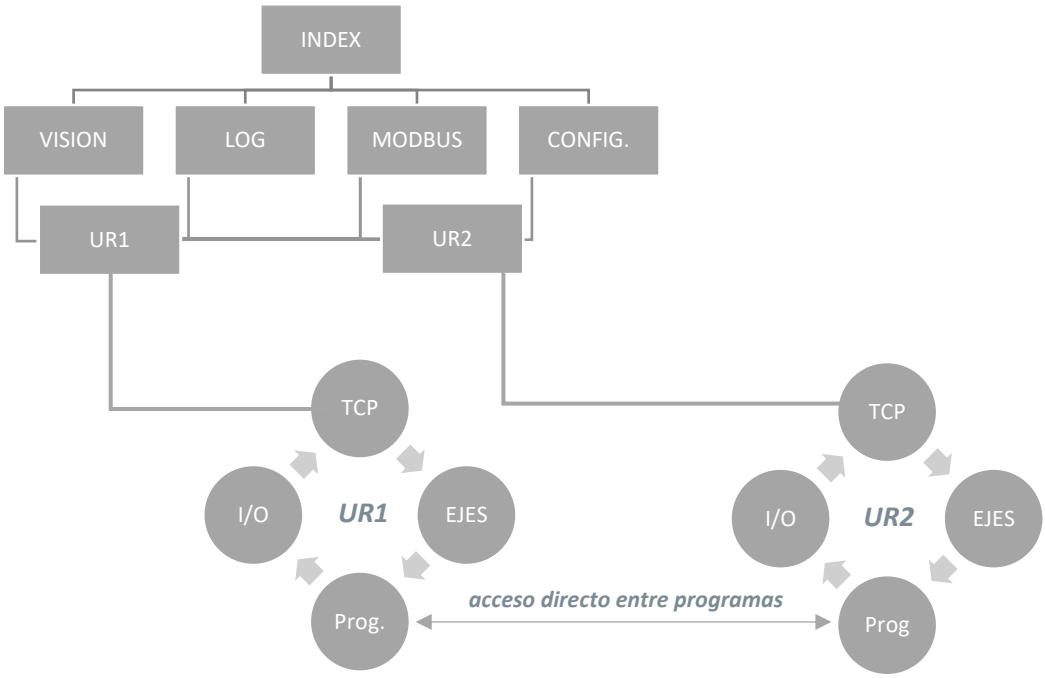


5.5.1. ORGANIGRAMA

La carpeta ***Main_Screens*** guarda las pantallas principales del *WebServer* (Figura 94), contando con 5 ventanas:

- **INDEX**: Página principal del proyecto, muestra una ventana donde se representa la celda.
- **VISIÓN**: Como se comentó previamente, contiene las animaciones relacionadas con la visión artificial así como la representación de los valores obtenidos en cada ciclo por la cámara en *RoboDK* y datos de cantidades de piezas procesadas, señales de la cámara, etc.
- **LOG_WINDOW**: Se define la configuración del archivo de la base de datos donde CODESYS guarda las alarmas que se producen. Permite el manejo de todas las alarmas, almacenar un *log* para consultas posteriores de los eventos generados así como la obtención de posibles soluciones a alarmas activas.
- **MODBUS**: Contiene un sistema de diagnóstico en tiempo real de las comunicaciones (se amplía información en próximas páginas).
- **CONFIGURATION**: Todo sistema cuenta con parámetros que deben ser establecidos en determinados rangos o valores fijos, estos permiten el buen funcionamiento y consecución de los procesos en una celda. El proyecto no es ajeno a esta situación, en la ventana *Configuration* el usuario puede modificar los parámetros de la cámara y cambiar la dirección *IP* de los robots (con la versión de los simuladores esta opción carece de relevancia pues *URSim* tiene direccionamiento *DHCP*, pero la plataforma se encuentra configurada para conectar los *UR* reales en cualquier momento que se desee), por otra parte existe un módulo que indica que posición se encuentra activa para que *UR1* entregue la pieza. Todos los parámetros son accesibles solo si se tienen permisos de usuarios con derechos de acceso.

Figura 94. Organigrama del WebServer.

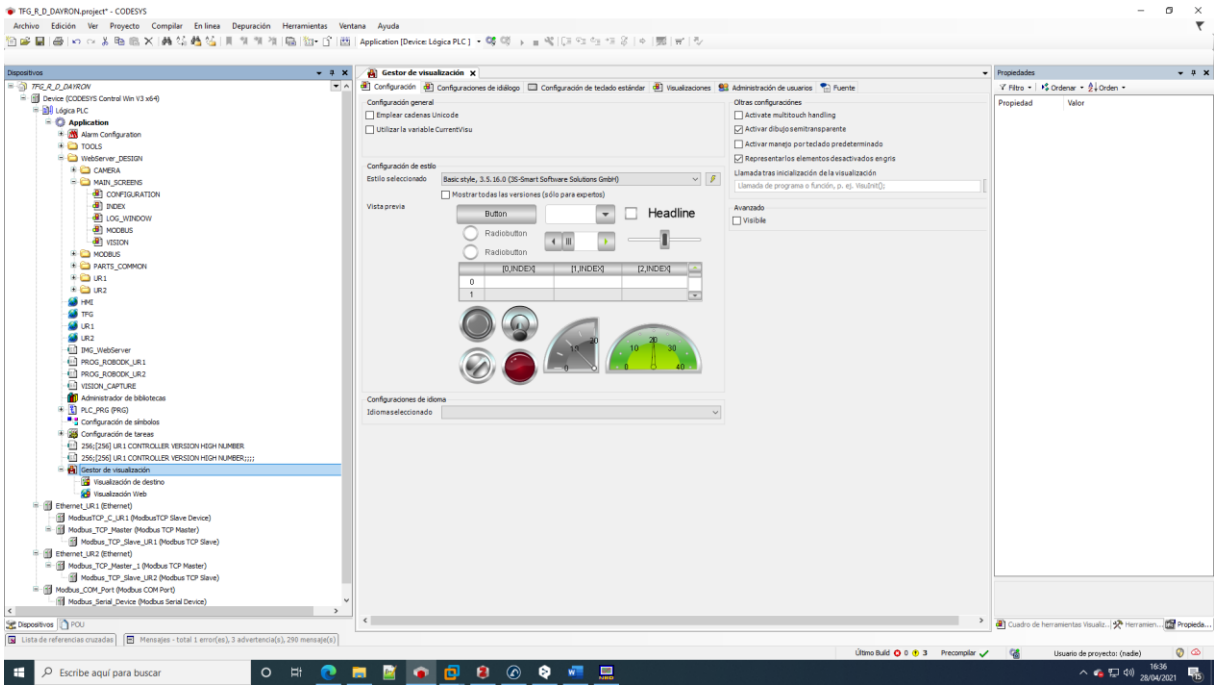


5.5.2. GESTOR VISUALIZACIÓN

Cuando se crea una ventana de visualización en un proyecto de *CODESYS*, automáticamente se genera un objeto denominado *Gestor de Visualización* (Figura 95), del cual depende la *Visualización de destino* y *la Visualización Web.*

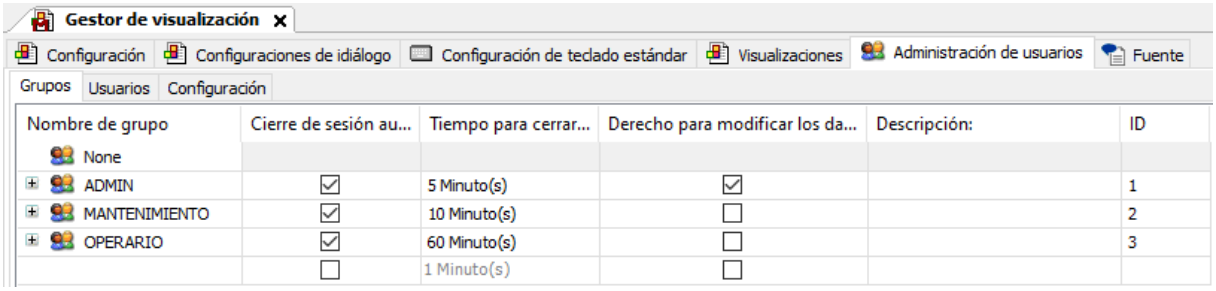
Dentro del gestor existen un grupo de configuraciones interesantes de comentar, en la pestaña *Configuración* se selecciona el estilo que presentaran los componentes nativos de *CODESYS* para la visualización de los datos en las pantallas a diseñar, en el caso del presente proyecto dicho estilo es *Basic style, 3. 5. 16. 0 (3S-Smart Solutions GmbH)*, la siguiente pestaña corresponde a la *configuración de diálogo* y es la seleccionada por defecto, dentro de *visualizaciones* se encuentran todas las ventanas creadas, en el caso del proyecto se han implementado ***59 pantallas*** diferentes.

Figura 95. Gestor de visualización del proyecto.



Sin embargo, la pestaña que más interés tiene está relacionada con la *Administración de usuarios*, en la cual se definen los permisos, el tiempo de cierre de sesión automático asignado a cada perfil, el nombre y la contraseña para cada uno, así como el grupo al cual pertenece. En la (Figura 96) se observa los tres usuarios presentes en el proyecto organizados en orden prioritario.

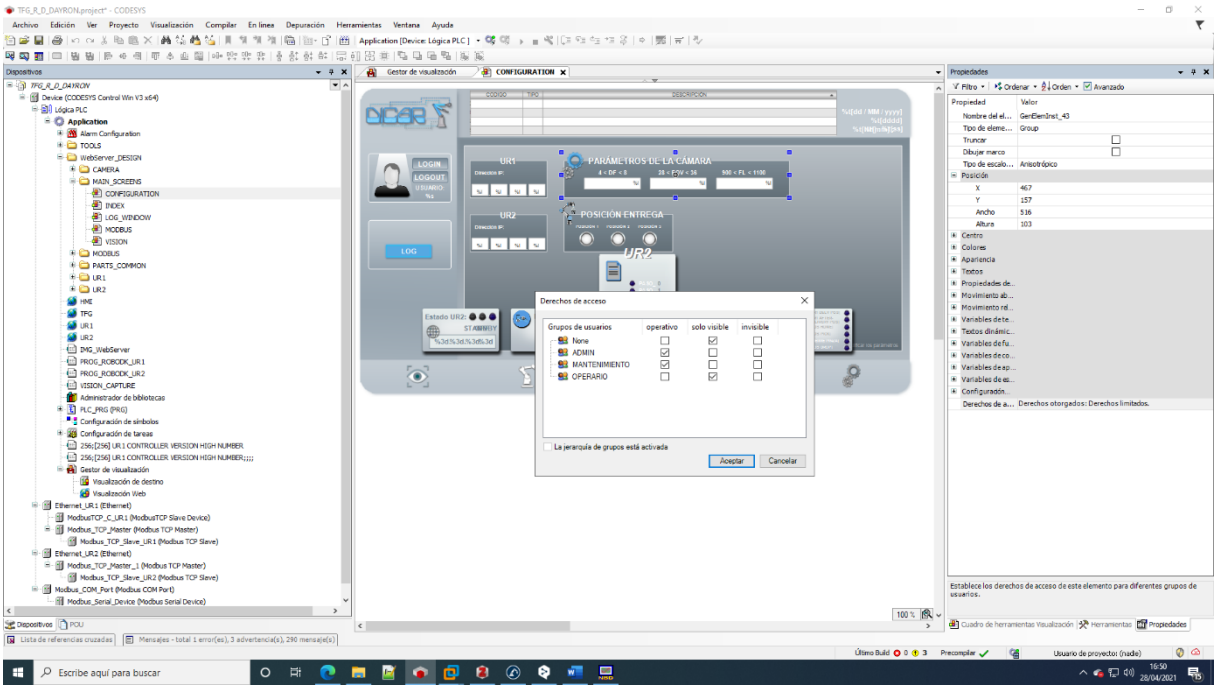
Figura 96. Creación de los usuarios con permisos de acceso en el proyecto.



Nombre de grupo	Cierre de sesión au...	Tiempo para cerrar...	Derecho para modificar los da...	Descripción:	ID
None					
ADMIN	☑	5 Minuto(s)	☑		1
MANTENIMIENTO	☑	10 Minuto(s)	☐		2
OPERARIO	☑	60 Minuto(s)	☐		3
	☐	1 Minuto(s)	☐		

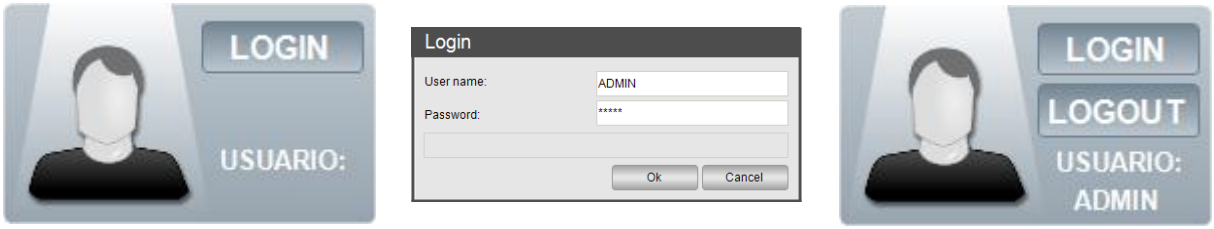
El componente que necesite permisos de acceso a través de usuarios debe ser configurado como tal, en la ventana *Configuration* se toma como ejemplo el módulo *Parámetros de la Cámara*, al ser una ventana instanciada, directamente en la ventana *Propiedades* (Figura 97), a la derecha del espacio de trabajo de *CODESYS,* la última propiedad está relacionada con el derecho de acceso, si se presiona *doble clic* el programa muestra una ventana emergente donde se relacionan los usuarios declarados y se conceden permisos de operación o de visibilidad al componente en cuestión.

Figura 97. Configuración de los derechos de acceso de un componente.



En la ventana principal del proyecto, *INDEX*, se encuentra disponible un acceso al gestor de usuarios de la aplicación (Figura 98), en la cual, al presionar *LOGIN* se muestra una ventana emergente que permite introducir el usuario y la contraseña.

Figura 98. Proceso para el acceso con usuario y contraseña al sistema.

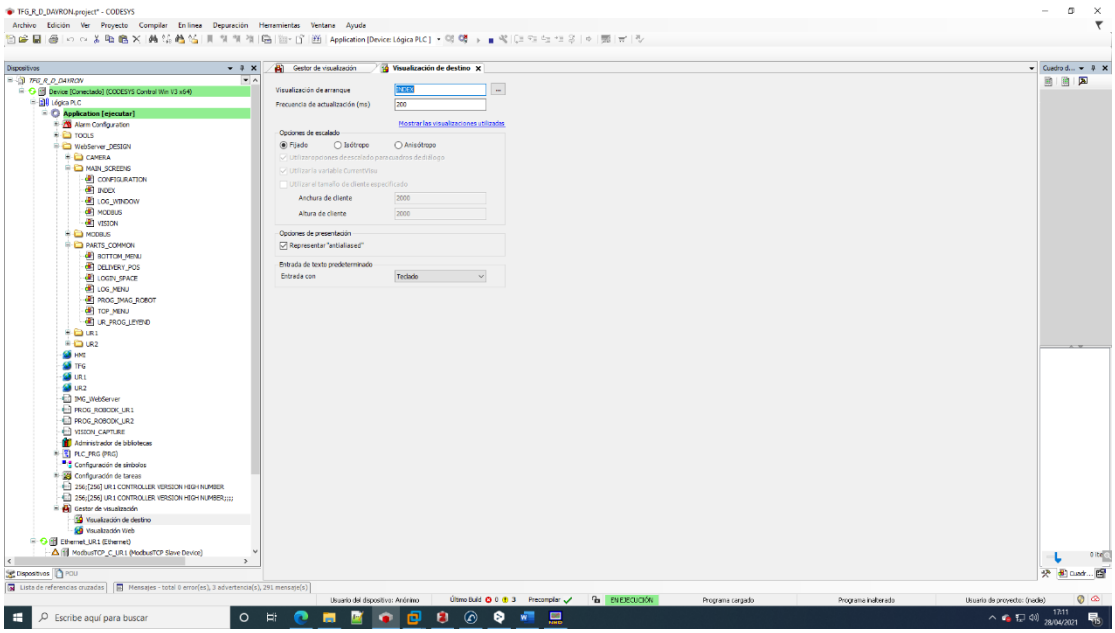


Los tres usuarios dados de alta en el sistema son:

- **ADMIN:** Posee los mayores niveles de acceso en el sistema. *Password*: **1234**
- **MANTENIMIENTO:** Permisos de acceso relacionados al trabajo desempeñado. *Password:* ***1234***
- **OPERARIO:** Usuario por defecto dado de alta, no posee permisos de acceso en el sistema.

- **Visualización de destino**: Es un objeto dependiente del *gestor de visualización*, en él se configura cuál es la ventana principal del proyecto, o lo que es similar, que pantalla mostrará el navegador cuando se haga una petición de acceso al *WebServer* (Figura 99), en el caso que ocupa al lector, la ventana definida como principal, como ya se comentó anteriormente, corresponde a *INDEX.* Además se definen las propiedades de representación y escala de la ventana en el navegador, contando con tres posibles opciones, el proyecto se deja configurado con una representación *fija* para que todos los dispositivos apliquen la resolución de 1280x720 *pixeles* definida.

Figura 99. Configuración de la visualización de destino.

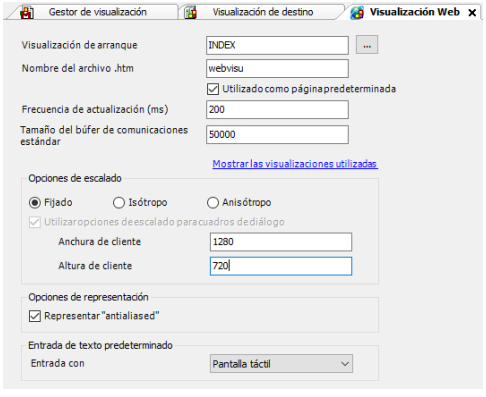


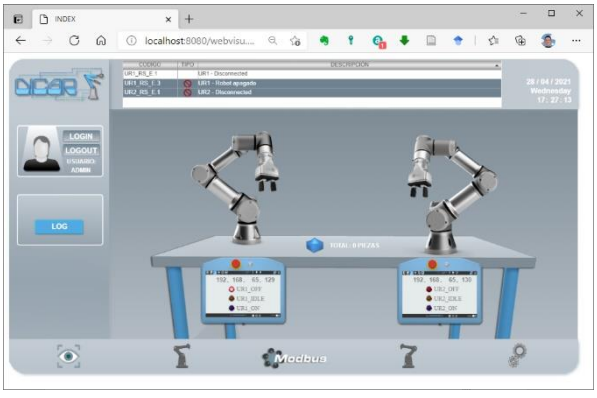
- **Visualización Web**: Es el último objeto enlazado al *Gestor de Visualización*, los parámetros de interés en esta ventana son la ventana de V*isualización de arranque* que como se indicó arriba es *INDEX* y el *Nombre del archivo .htm*, este coincide con el nombre que se debe incluir en la barra de navegación del explorador para invocar la página principal del *WebServer*. (Figura 100).

La dirección para acceder al *WebServer* es:

http://localhost:8080/webvisu.htm

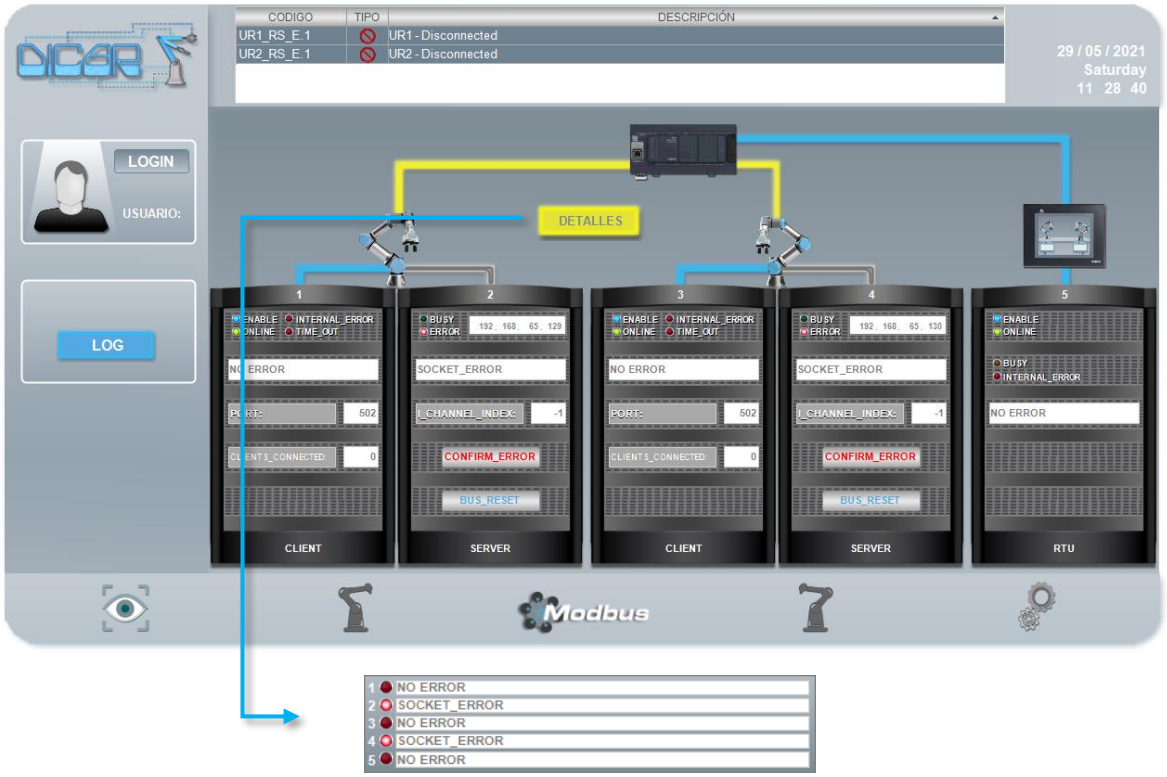
Figura 100. a).Configuración de la visualización web. b). Navegador web con mostrando la ventana principal del WebServer.





- **Ventana *Modbus***: Dentro del conjunto de pantallas diseñadas en el *WebServer* se quiere hacer hincapié en la ventana dedicada al diagnóstico de las comunicaciones entre el *PLC* y los dispositivos periféricos (*UR1*, *UR2* y el *HMI*). Para desarrollar una tarea cien por ciento funcional, se recurren a los ***IEC Object***[24], cada módulo de *Modbus* posee dichos elementos que se denominan *atributos*, los servidores y clientes proporcionan elementos para controlar los dispositivos y recuperar la información de estado de estos. La (Figura 101) muestra el diagnóstico del bus de comunicación en estado de *error* provocado por la desconexión de los dos *UR*, para provocar dicho fallo, se cerraron los *IDE* que simulan los *UR3* en cada máquina virtual. Es posible realizar un *reset* del bus en caso de recuperar la conexión con los dispositivos, así como un chequeo en tiempo real de la consulta de todos los registros consultados en el *Modbus Server* de cada robot. Dicha pantalla cumpliría con el estándar de mayor exigencia existente en cualquier fábrica del sector de la automoción o la biomedicina, por mencionar algunos de los sectores con notable nivel de requerimientos en cuanto al tratamiento de las comunicaciones, la seguridad y el diagnóstico.

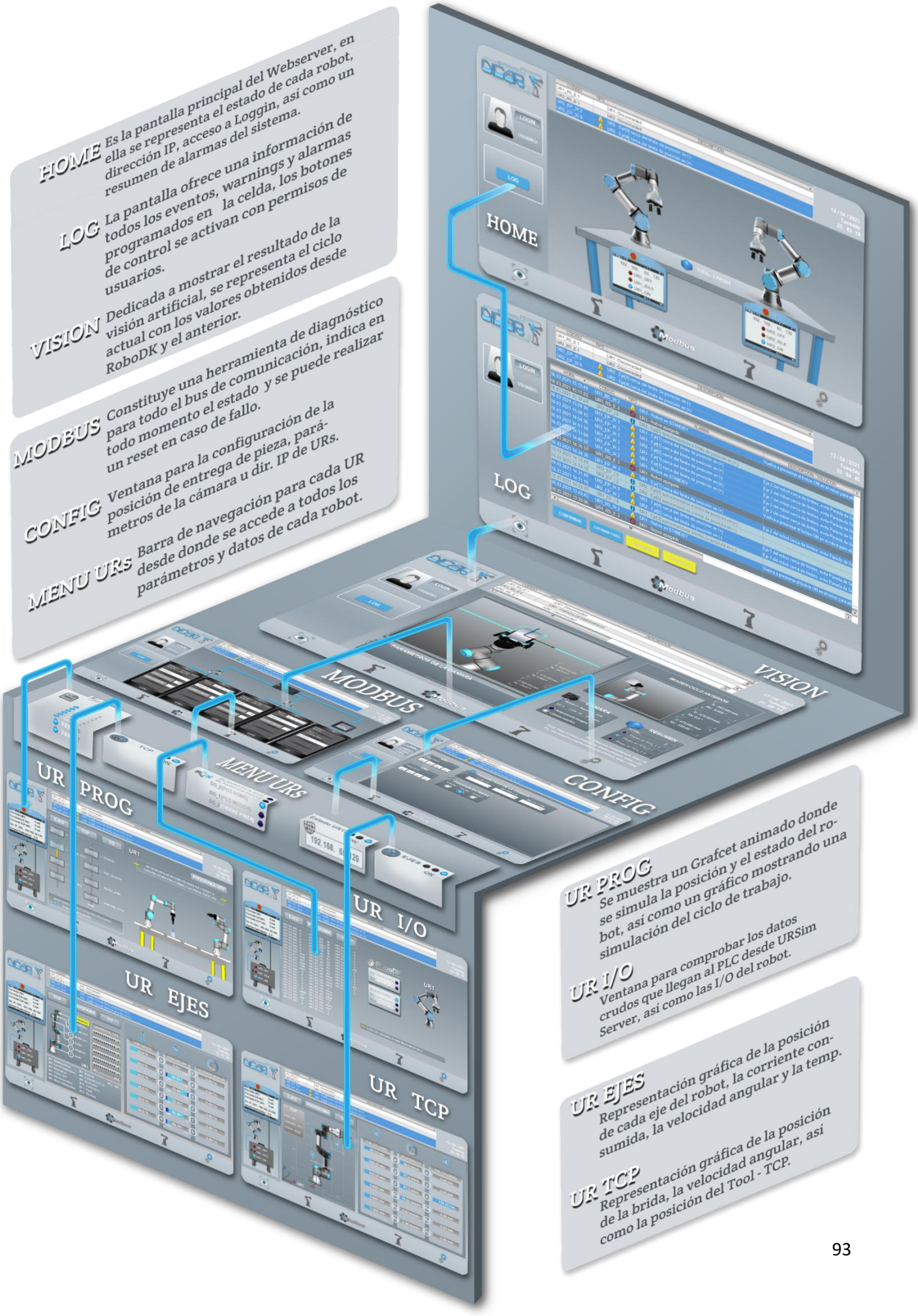
*Figura 101. Ventana para el diagnóstico del bus de comunicación **Modbus** entre el PLC y la preriferia.*



Una vez comentado todo el proceso de diseño, organización, segmentación del trabajo y justificación de los elementos presentes en el desarrollo del *WebServer*, solo resta mostrar el flujo entre las distintas pantallas que conforman el primer dispositivo de interfaz gráfica de la celda: (Figura 102). En el *manual de programación* se detallan todas las pantallas y módulos programados.

[24] Estándar para la automatización de subestaciones en el que se abordan aspectos relacionados con requerimientos generales del sistema, gestión de los proyectos de ingeniería y requerimientos de comunicaciones.

*Figura 102. Diagrama de flujo para la navegación entre las distintas pantallas diseñadas para el **WebServer.***

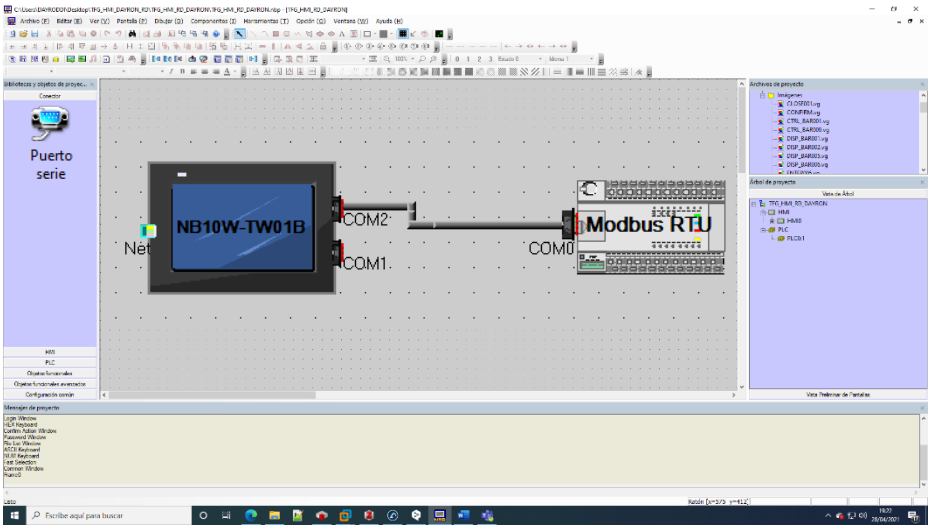


5.6. HMI

La comunicación con el *HMI* será igualmente virtual y simulada empleando el software ***NB Designer de Omron,*** de forma específica se selecciona una pantalla *NB10W-TW01B* que cuenta con un panel LCD TFT de 10.1" y permite, al ser un elemento virtual, mayor resolución y espacio para acomodar los distintos módulos de representación de datos. (*v412_nb_series_hmi_datasheet_es.pdf*, s. f.).

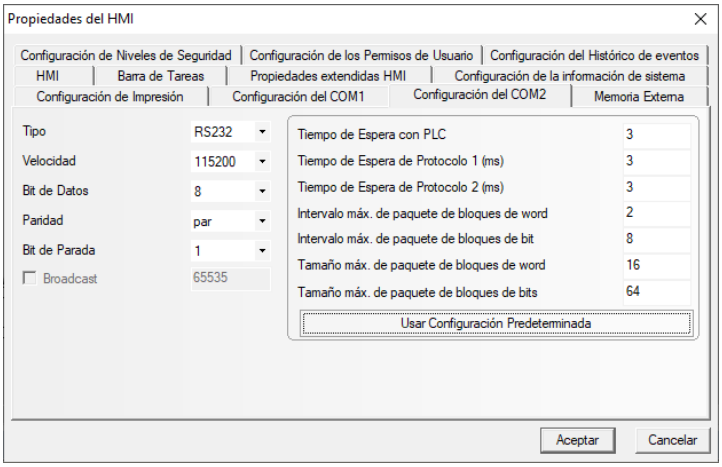
La configuración de los dispositivos es sencilla, se emplean tres elementos, siendo uno de ellos la pantalla, (Figura 103), como se definió en el apartado *Comunicaciones* en el *PLC* y de forma más concisa *Modbus_RTU*, el autómata emplea el puerto ***COM1*** para establecer el enlace, por lo que el *HMI* debe estar conectado a través del ***COM2***.

Figura 13. Configuración de la conexión entre el HMI y el PLC.



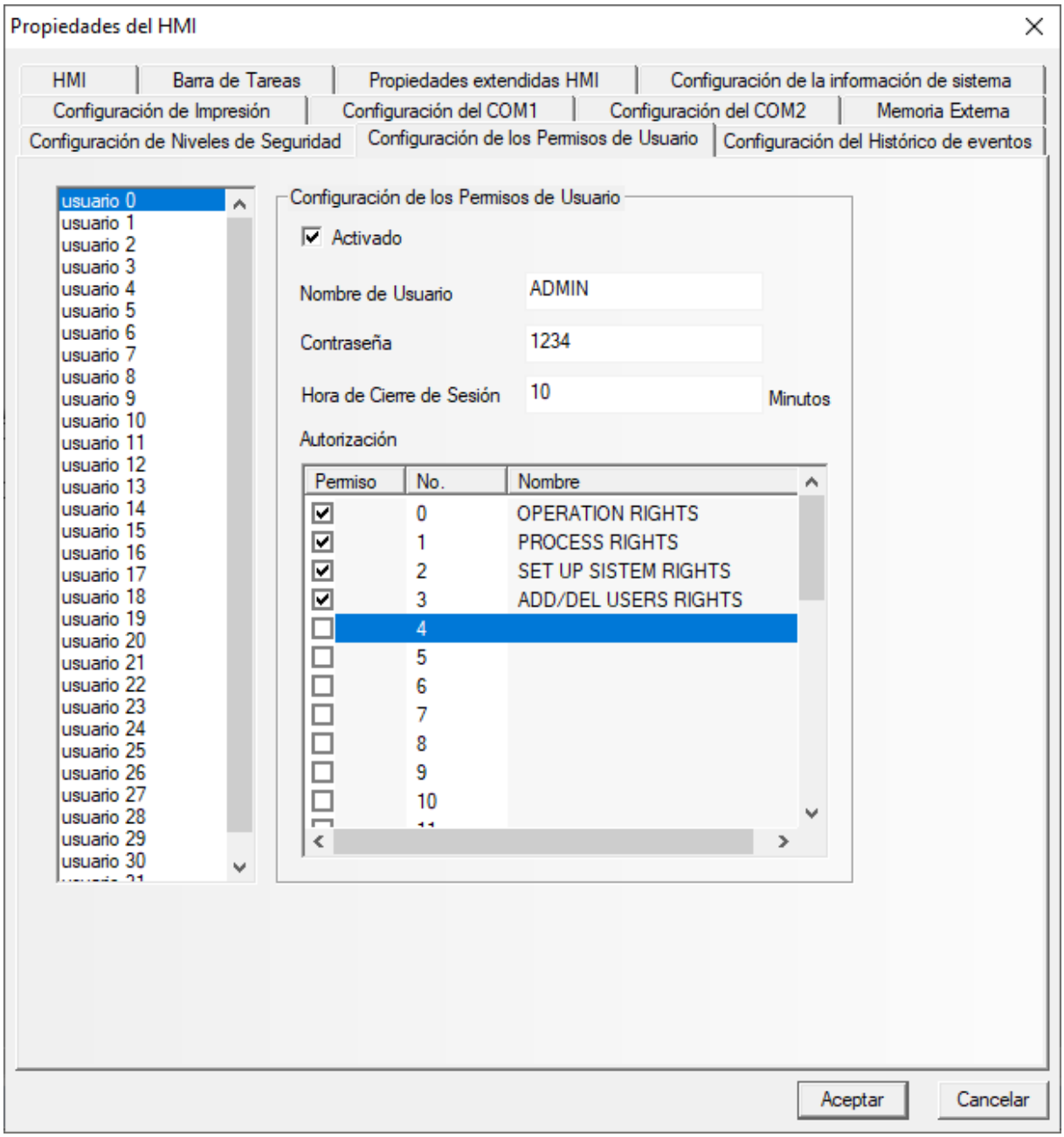
Presionando *doble clic* sobre la pantalla se accede a la ventana de configuración, en ella existe una pestaña denominada *Configuración del COM2* donde se deben definir los mismos parámetros que en su momento se establecieron en el módulo *Modbus_Serial_Device* del *PLC* en *CODESYS*, en la (Figura 104) se aprecia dicha configuración.

*Figura 104. Configuración del puerto serie **COM2** de la pantalla.*



En la misma ventana existe una pestaña relacionada con la configuración de los *permisos de usuarios*, *NB Designer*, permite generar una lista de hasta 32 usuarios diferentes, y asignarle a cada uno hasta 32 permisos o niveles de acceso diferentes. El procedimiento consiste en seleccionar al ***usuario 0*** y realizar una equivalencia con el ***ADMIN*** del sistema, tal y como se configuró en el *Gestor de Visualización* de *CODESYS* (Figura 105), posteriormente se asigna el mismo valor para la *Contraseña* y por último, se asignan los distintos permisos que necesite la estación.

Figura 105. Configuración del grupo de usuarios y los permisos definidos para cada uno de ellos.



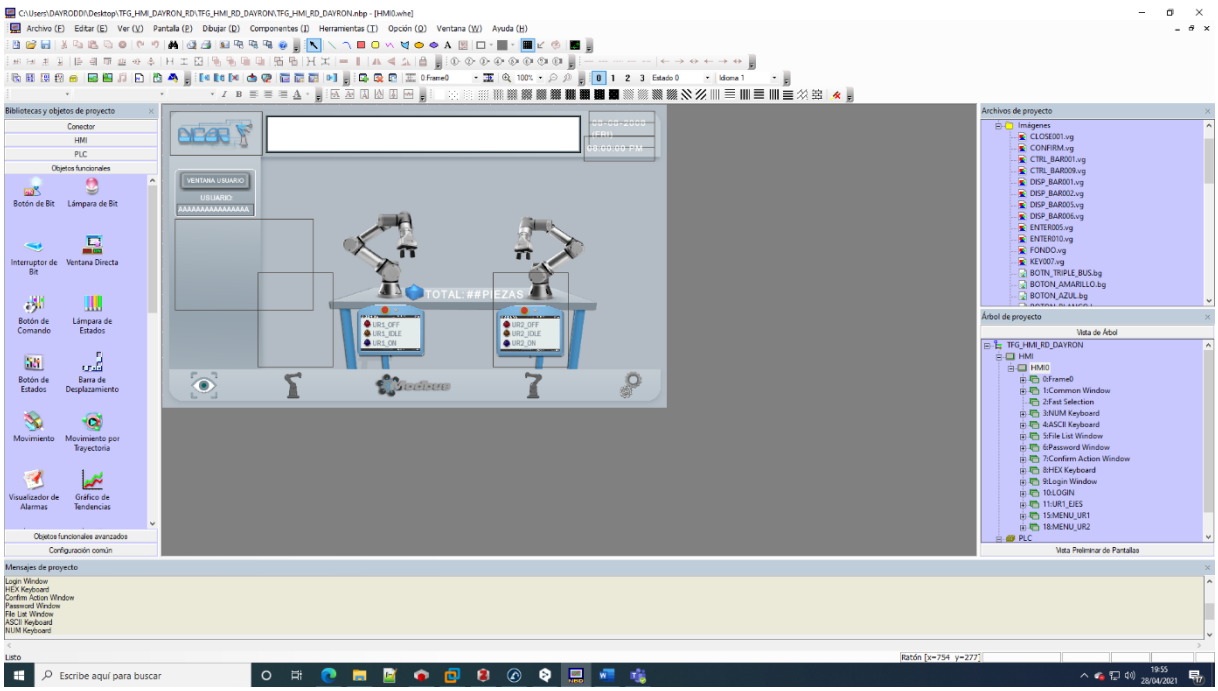
A modo de ejemplo, se definen cuatro permisos entre los cuales los usuarios pueden tener optar, definir nuevos, o no conceder ninguna autorización, como es el caso de *Operario.*

Una vez el panel *HMI* se encuentra configurado, se arrastra al área de trabajo un icono *Puerto Serie* (Figura 103, página 94, panel izquierdo), para enlazar la pantalla con el *PLC* previamente insertado en la misma región, el dispositivo en un elemento genérico de tipo *Modbus_RTU* (Figura 103).

Tras concluir la configuración de la pantalla y la comunicación con el autómata, se inicia el proceso de diseño de las distintas ventanas, con el único requisito de que deben ser similares a las desarrolladas previamente en el *WebServer* de *CODESYS*, con las salvedades impuestas por elementos que tienen una apariencia diferente en *NB Designer.* Conviene recordar uno de los principios de *UI* definidos anteriormente, el proceso de diseño de las interfaces debe ser *Consistente.*

El software de desarrollo de interfaces gráficas de *Omron* posee un grupo de componentes predefinidos que permiten, tras la configuración, intercambiar información con los dispositivos con los cuales esté conectada la pantalla (Figura 106), como se puede apreciar en la imagen, la pantalla principal del proyecto es casi idéntica a *INDEX* en el *PLC,* para la representación de los estados de cada *UR* se utilizaron 6 componentes *Lámpara de Bit*, corresponden a los *leds* incluidos en los *TeachPendant* que están apoyados en la mesa.

Figura 106. Interfaz de NB Designer con la primera de las pantallas en desarrollo. INDEX.



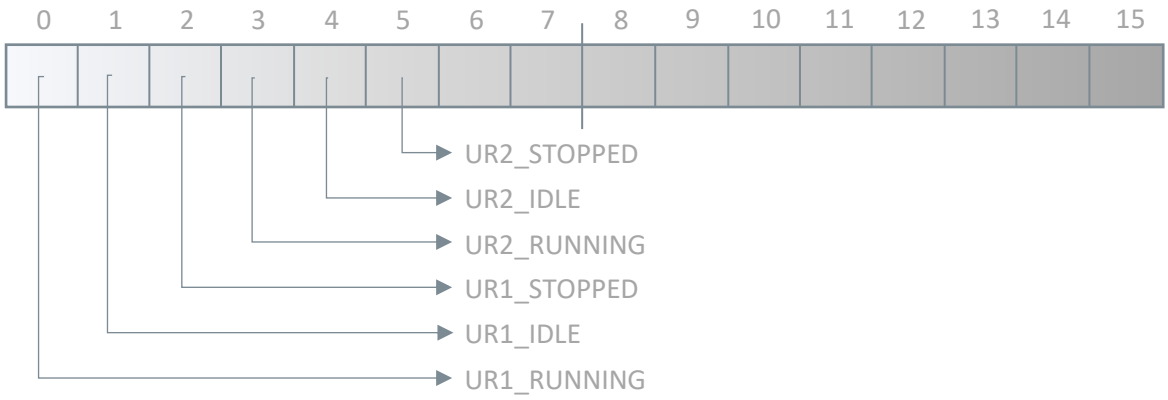
Estos componentes leen direcciones del bus *Modbus_RTU*, por lo que el acceso a estos datos se realiza a través de las direcciones ***1x*** , para ejemplificar la configuración de un elemento de esta naturaleza, se escoge el elemento que representa el robot *UR1* en la mesa, para que el componente tenga un comportamiento dinámico de dos estados, debe estar enlazado a una variable de tipo *BOOL* en el *PLC* que indique si el robot se encuentra conectado o no.

Consultando la (Figura 49, página 49) se observa que el primer bit del registro de salidas del *PLC* es justo el dato necesario para indicar si *UR1* está operativo o no, sin embargo, si en la ventana de configuración del componente *Lámpara de Bit* se configura el *Área/Etiqueta* en las direcciones ***1x*** y la *Dirección* en ***1***

tal y como aparece en la tabla de direccionamiento del bus de datos *Modbus_RTU* del *PLC*, no se recibe ningún dato independientemente del estado del robot.

La dificultad radica en el formato que emplea la tarjeta configurada en el autómata para enviar la información, para una mejor comprensión, el autor recurre a dibujar un dato tipo *WORD* con todos los bits accesibles (Figura 107).

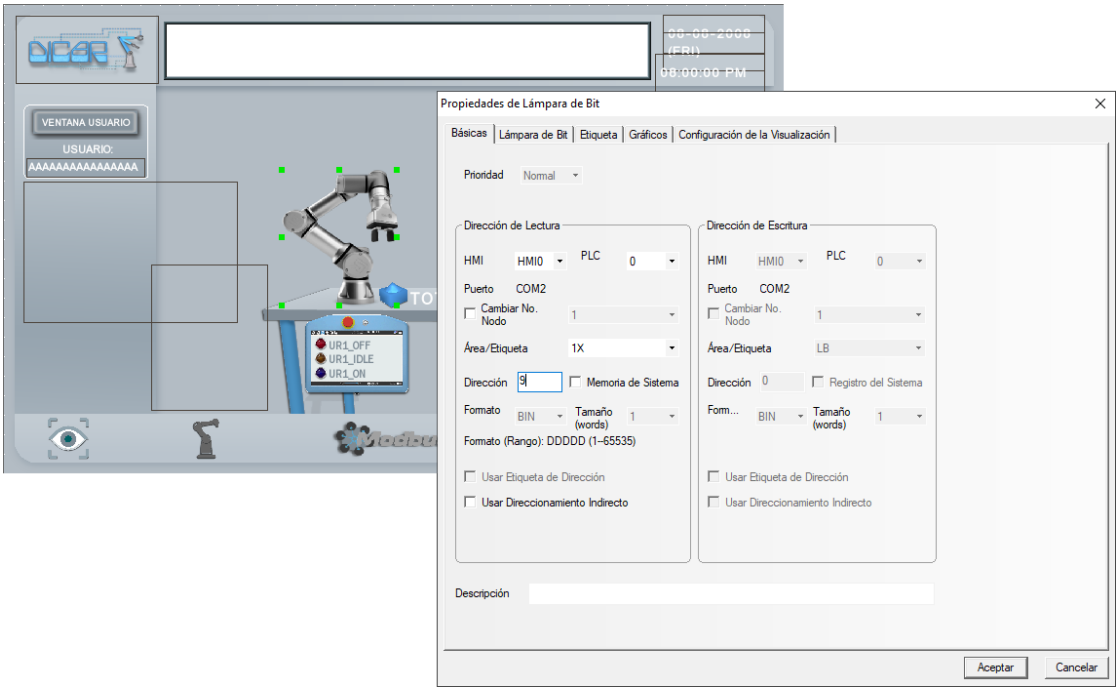
Figura 107. Representación gráfica del primer registro de salida del PLC a través de la interfaz Modbus_RTU.



La (Figura 49) muestra un direccionamiento del primer registro de salida del *PLC* tal como se ha representado en la imagen superior, sin embargo, si se comprueba la configuración del dispositivo *Modbus_COM_Port*, uno de los parámetros era *Bits de datos* que estaba fijado en ***8***, esto quiere decir que el autómata no es capaz de enviar en una única operación datos de 16 bits, en su lugar descompone la *palabra* en dos bytes independientes, enviando primeramente la parte alta, o de mayor ponderación, y posteriormente la más baja, o lo que es lo mismo, primero se envían los bits relativos a las posiciones 8-15 y luego de 0-7.

Siguiendo este razonamiento, si *UR1_RUNNING* se encuentra direccionado en el bit 0 del primer registro, el componente *Lámpara del bit* debe apuntar al *Área/Etiqueta* ***1x*** pero a la *Dirección* ***9*** (Figura 108), pues el bit ***0*** en la secuencia de envío se recibe en el orden número 9.

Figura 108. Direccionamiento del componente que representa UR según el razonamiento explicado.



La (Tabla 9) muestra el direccionamiento para acceder a los registros del bus de datos *Modbus_RTU*:

Tabla 9. Direcciones Modbus para el acceso de los distintos datos enviados.

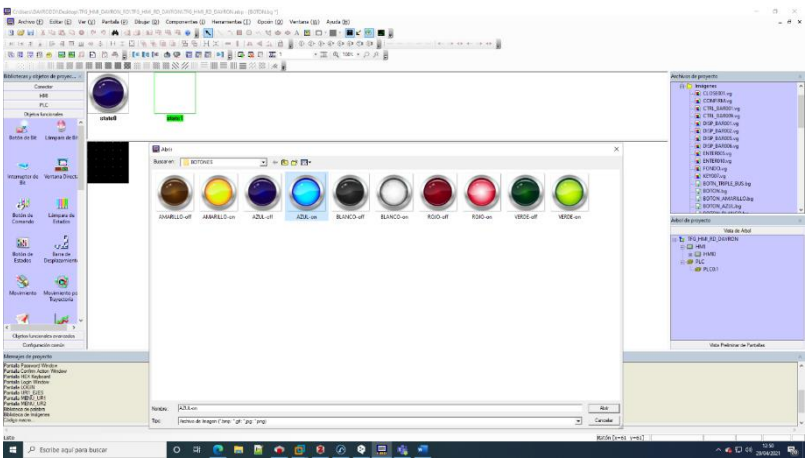
Dirección	Descripción
0xxxx	Lectura/Escritura de salidas discretas o *Coils*. Una dirección de referencia 0x se utiliza para conducir los datos de salida a un canal de salida digital.
1xxxx	Lectura de entradas discretas. El estado ON/OFF de una dirección de referencia 1x es controlado por el canal de entrada digital correspondiente.
2xxxx	Lectura de registros de entrada. Un registro de referencia 3x contiene un número de 16 bits recibido de una fuente externa, por ejemplo, una señal analógica.
3xxxx	Lectura/Escritura registros de salida o *Holding*. Un registro 4x se utiliza para almacenar 16 bits de datos numéricos (binarios o decimales), o para enviar los datos de la CPU a un canal de salida.

5.6.1. BIBLIOTECA DE IMÁGENES

A pesar de que *NB Designer* no brinda una gama de colores tan amplia (*16 bits*) como *CODESYS*, el autor del proyecto intenta que las pantallas que se van a diseñar en el *HMI* mantengan la misma apariencia de las creadas previamente en el *WebServer*, *concepto de Consistencia*, para ello es necesario importar al proyecto todas las imágenes que se utilizaron.

Omron permite crear una *biblioteca de imágenes* donde cada elemento puede tener una apariencia relacionada con diferentes estados, para ejemplificar el proceso se creará un *led azul*, (Figura 109), en el menú principal del programa, seleccionar *Dibujar/Añadir imágenes*, posteriormente, en la ventana emergente asignar un nombre y seleccionar la opción *Bitmap*, al presionar *Aceptar* el espacio de trabajo es similar al mostrado, en el cual el ***state0*** aparece por defecto, es necesario crear un nuevo estado con ayuda de un *clic derecho/Añadir estado*, solo resta asociar una imagen a cada estado del componente, seleccionando *state0* se presiona *clic derecho* sobre el recuadro negro que se encuentra debajo, *Cargar imagen*, en el navegador se accede a la ubicación del archivo que se desea incluir y al finalizar tendremos un elemento en la biblioteca que se adjuntará como imagen de un componente tipo *Lámpara, Botón o Interruptor de bit*.

Figura 109. Proceso para crear un elemento con dos estados en la biblioteca de imágenes de NB Designer.

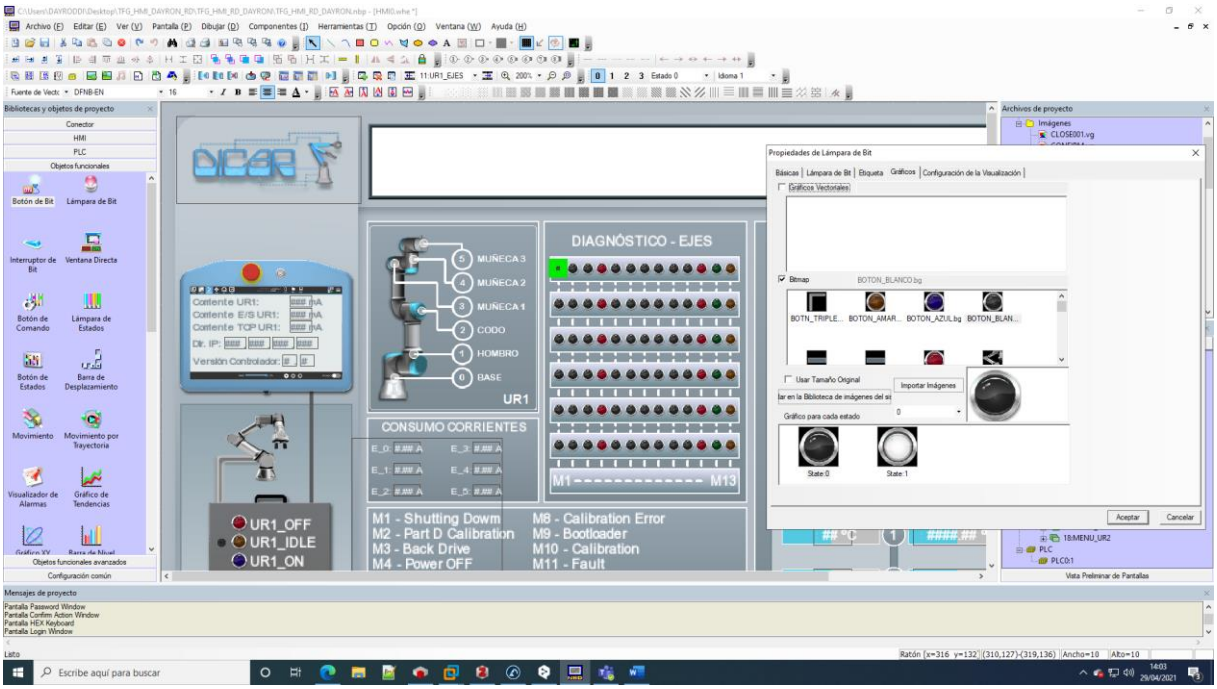


Con todos los archivos incorporados a la biblioteca de imágenes se puede personalizar todo el diseño de los elementos que conformarán los distintos módulos incluidos en cada pantalla, tal y como se llevó a cabo en el *WebServer*.

> **No es objetivo del proyecto replicar todas y cada una de las pantallas del *WebServer*, a modo de ejemplo solo se diseñan aquellas homólogas de *INDEX* y *UR1_EJES*. En ellas se utilizan todos los componentes esenciales para desarrollar una futura interfaz empleando una pantalla *HMI* plenamente funcional y con los estándares marcados en la actual *Industria 4.0.***

Completando la metodología de introducir en el diseño todas las imágenes reutilizadas, se toma como muestra un componente *Lámpara de bit* de la pantalla *UR1_EJES* relacionado con el primer bit de diagnóstico del *EJE_1* (Figura 110), accediendo a la ventana de propiedades, pestaña *Gráfico* y seleccionando la opción *Bitmap*, se puede asignar cualquiera de los bloques gráficos previamente creados.

*Figura 110. Asignación gráfica a una **Lámpara de Bit** utilizando un elemento de la biblioteca de imágenes creada previamente.*



El resto de los elementos para la representación de los datos relacionados a *UR1*, están compuestos por registros de 16 *bits*, como se mostró en la (Figura 49, página 49), en este caso no es necesario tener en cuenta el proceso de intercambio de *bytes* antes descrito para el acceso a *bits* independientes dentro del registro, *NB Designer* interpreta de forma similar a *CODESYS* el protocolo de transmisión y recepción de información.

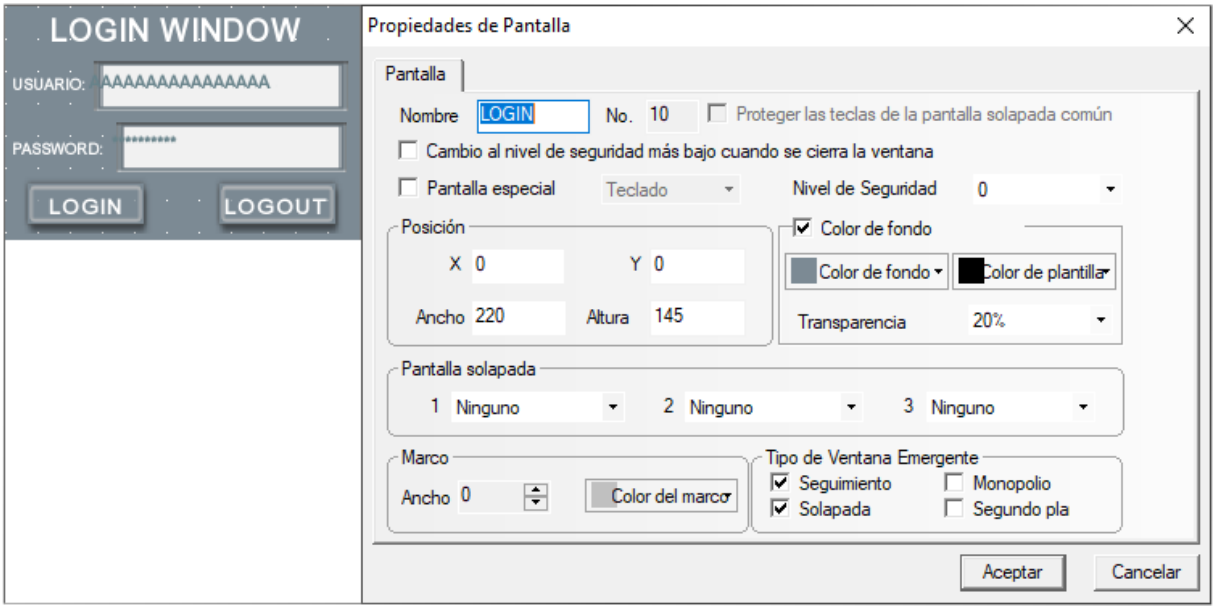
La primera pantalla del *HMI* se denomina por defecto ***HMI0*** y es accesible desde el panel inferior derecho del programa, en esta primera vista, de forma idéntica al *WebServer,* está representada la celda a través de los elementos que la conforman, la disposición de estos es la misma así como los

paneles que la componen, diferenciando el recuadro superior donde se representan las alarmas del sistema que no admite una configuración gráfica parecida al elemento equivalente en *CODESYS.*

5.6.2. GESTIÓN DE USUARIOS

Para imitar la ventana donde los usuarios pueden llevar a cabo el proceso de autenticación para activar los permisos que le conceden acceso a determinadas tareas, se crea una nueva pantalla en el proyecto denominada ***LOGIN***, con una dimensión de 220 *pixeles* de Ancho y 145 *pixeles* de Alto. Esto garantizará, a continuación, crear un elemento en la pantalla principal que realice una llamada a *LOGIN* al desencadenar una acción determinada (Figura 111).

Figura 111. Configuración de la pantalla para la autenticación de usuarios en la aplicación.



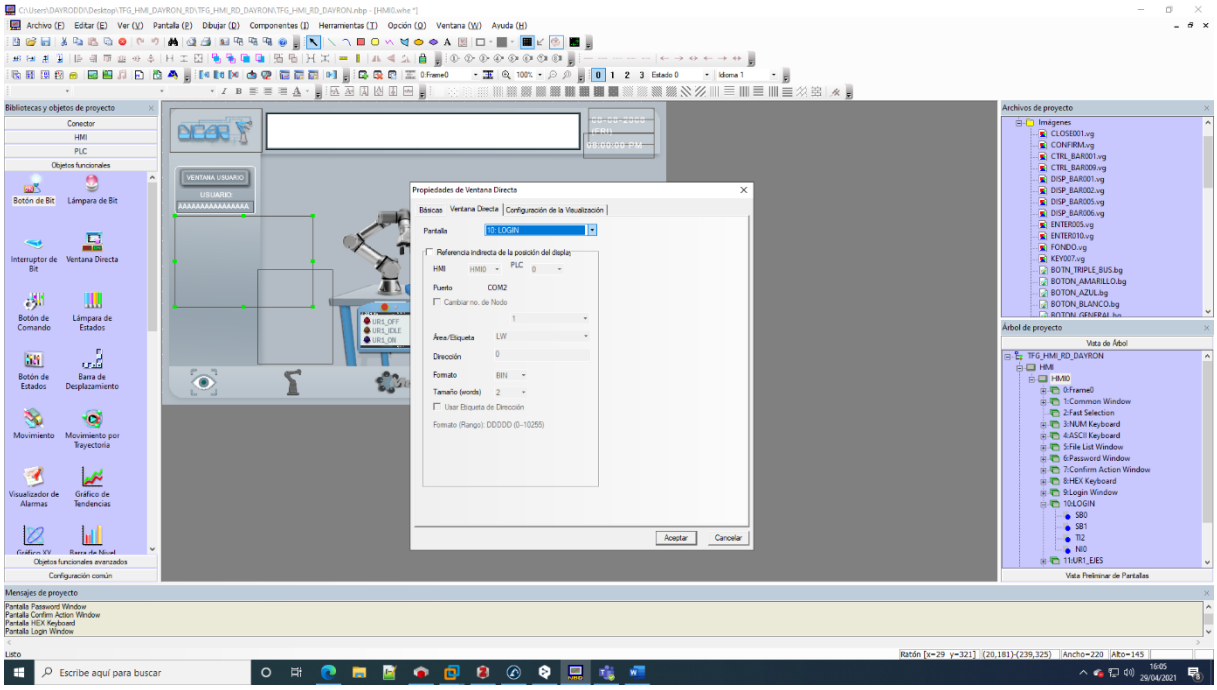
El *cuadro de texto* relativo al campo *USUARIO* es un componente de *entrada de texto* y la dirección a la que apunta está relacionada con los registros internos que tiene el *HMI* para la gestión de permisos. La (Tabla 10) muestra todas las direcciones necesarias para completar la tarea de gestión de usuarios en el *HMI.*(*NB-Series Programmable Terminals NB-Designer Operation Manual*, s. f.)

Tabla 10. Relación de los registros internos para la gestión de usuarios.

Dirección	Componente	Función	Descripción
LW9486	Campo Texto (usuario)	Entrada USUARIO	Máximo 32 caracteres
LW9502	Campo Texto (Password)	Entrada CONTRASEÑA	Double-Word
LB9165	Botón (LOGIN)	Ejecuta LOGIN	Tras su activación, se realiza la operación de inicio de sesión.
LB9166	Botón (LOGOUT)	Ejecuta LOGOUT	Tras su activación, se realiza la operación de cierre de sesión.

Completados todos los pasos necesarios para que la ventana sea operativa, se regresa a la pantalla principal del proyecto y se inserta un objeto *Ventana Directa* el cual, dentro de la ventana propiedades y la pestaña *Ventana Directa*, se configura que la pantalla que mostrará dicho elemento es *LOGIN* (Figura 112), solo queda definir el desencadenante para que el objeto sea visible, para ello en el panel izquierdo de *HMI0*, existe un botón denominado *Ventana Usuario*, el cual se comporta como un botón alternador que escribe sobre la dirección interna del *HMI* ***LB 0***, de esta forma cuando se presiona por una vez *LB0 = 1* y la ventana *LOGIN* se encuentra visible, tras autenticarse se muestra debajo el usuario dado de alta y el botón se torna azul con la denominación *Cerrar Ventana*, al volver a presionarlo *LB0 = 0* y la ventana *LOGIN* queda oculta.

Figura 112. Configuración del objeto ***Ventana Directa****.*



5.6.3. MENÚ

En el proyecto se definen dos menús para navegar entre las distintas pantallas, el primero coincide plenamente con el menú principal del *WebServer* localizado en la zona inferior de todas las pantallas, el segundo se diseña de forma diferente al menú secundario que se superpone a cada pantalla en *CODESYS* pues *NB Designer* no es una aplicación ideal para presentar imágenes con transparencias (Figura 113), sin embargo, al utilizar los mismos colores, tipografías y elementos principales, se guarda la *consistencia* del diseño.

Figura 113. Menú principal y secundario para navegar por el HMI.

La composición del menú principal se implementa con dos *Botones de Bit y* tres *Botones de Función*, estos últimos se les pueden asociar distintas acciones (*Cambiar Pantalla*, *Tecla [alfanumérica]*, *Ejecutar una Macro,* etc.), el comportamiento programado será *Cambiar Pantalla*, de esta forma cada uno de los tres botones apunta a una pantalla relativa a la función que describe el icono que lo representa.

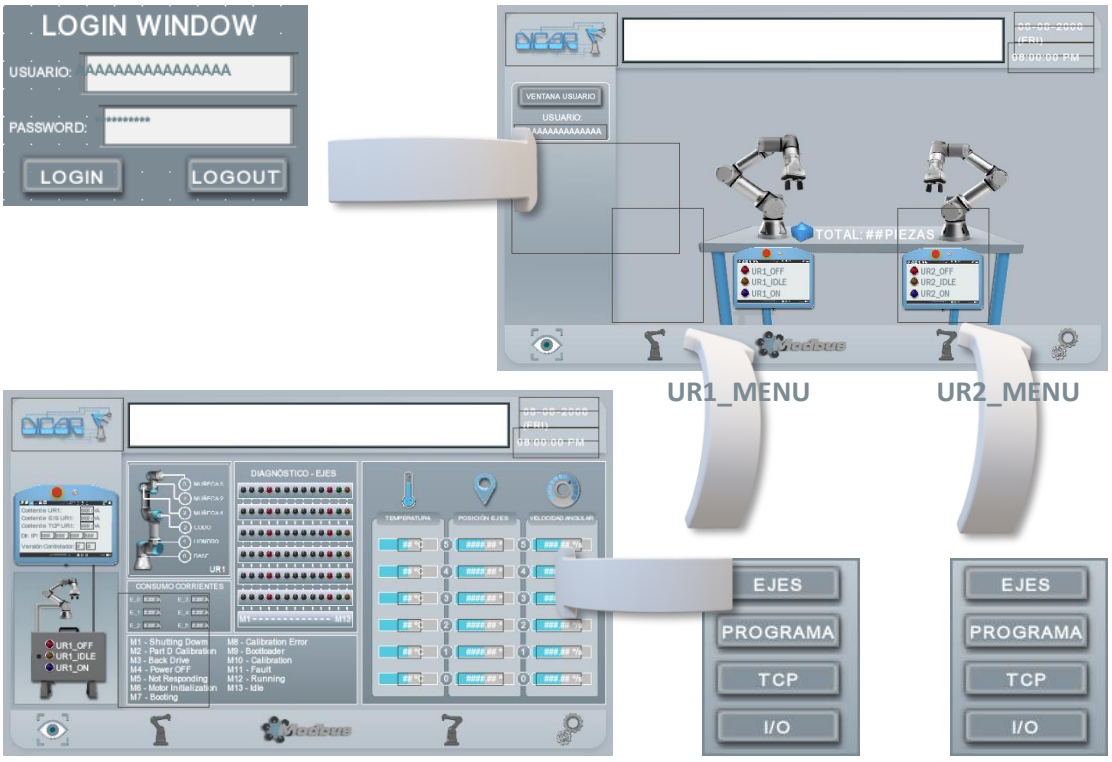
Los robots *UR1* y *UR2* necesitan de cuatro pantallas diferentes que permitan el acceso a la totalidad de los datos que se reciben desde cada máquina virtual, es por ello que los botones que representan a *UR_IZQ* y *UR_DER* en el menú principal son *Botones de Bit*, con el comentado comportamiento *Alterno* y el direccionamiento *LB1* y *LB2*, que serán las direcciones de memoria que se consultarán por sendas *Ventanas Directas*, son los encargados de desencadenar la visualización de la ventana *MENU_UR1* y *MENU_UR2* respectivamente.

En el caso del menú secundario, la botonera con los cuatro accesos a *EJES*, *PROGRAMA*, *I/O* y *TCP* se compone igualmente de los respectivos botones de función que estarán enlazados a las pantallas con la misma denominación que la etiqueta que describe el funcionamiento de los pulsadores.

5.6.4. PANTALLAS

Con todos los elementos definidos previamente solo resta presentar como quedan conformada las dos pantallas diseñadas en el *HMI* (Figura 114), tal como se representa, con solo cinco ventanas se muestra al lector como puede implementar por sí mismo una interfaz gráfica empleando *NB Designer* como software de programación de pantallas *HMI*. La ventana *UR1_EJES* es plenamente operativa y muestra el mismo volumen de datos que se representan en *UR1_EJES* en el *WebServer*.

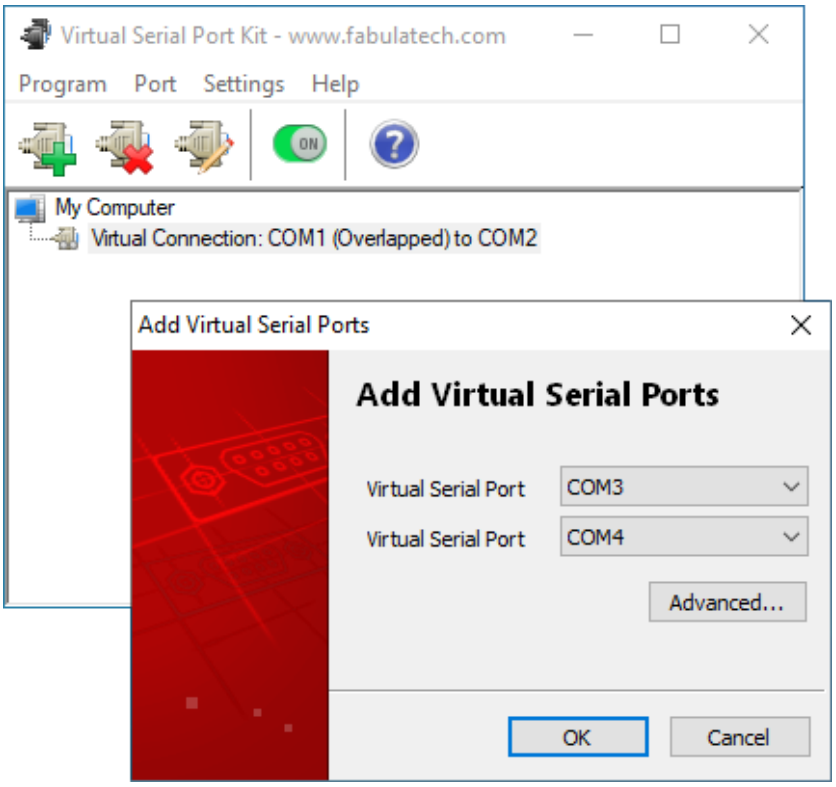
Figura 114. Presentación de las pantallas diseñadas en NB Designer con su interrelación.



5.6.5. VIRTUAL SERIAL PORT KIT

Para enlazar el *HMI* con *CODESYS*, tal como se mencionó en el apartado *Modbus_RTU* (página 48), se emplea un software que virtualiza la comunicación y establece un enlace entre los puertos ***COM1*** y ***COM2*** en el *PC* (Figura 115), *Virtual Serial Port Kit posee una* configuración simple que solo requiere crear un enlace nuevo entre dos puertos a través del icono que simula un conector *RS232* con un símbolo **+**, en la ventana emergente se seleccionan que puertos se necesitan enlazar (en la imagen aparecen *COM3* y *COM4* porque ya existe la conexión entre *COM1* y *COM2*) y por último se presiona ***OK***, la ventana contempla un botón *Advance* que permite una configuración detallada del patillaje del conector pero como será una simulación, no es necesario modificar ningún campo extra.

*Figura 115. Creación de una comunicación virtual entre el puerto **COM1** y **COM2** empleando un software de simulación de puerto serie.*



5.6.6. SIMULACIÓN

Antes de lanzar la simulación de la pantalla quedan un par de tareas por realizar, *NB Designer* necesita compilar todo el programa diseñado y que no exista ningún error, a través del menú principal *Herramientas* se presiona sobre *Compilar Todo*, tras los cual el compilador del *IDE* de *Omron* comprueba que el código implementado es correcto.

NB Designer contempla tres formas distintas de simular una pantalla, cada una de ellas se encuentra en el menú principal opción *Herramientas*, en la (Tabla 11) se amplía la información de cada modo de

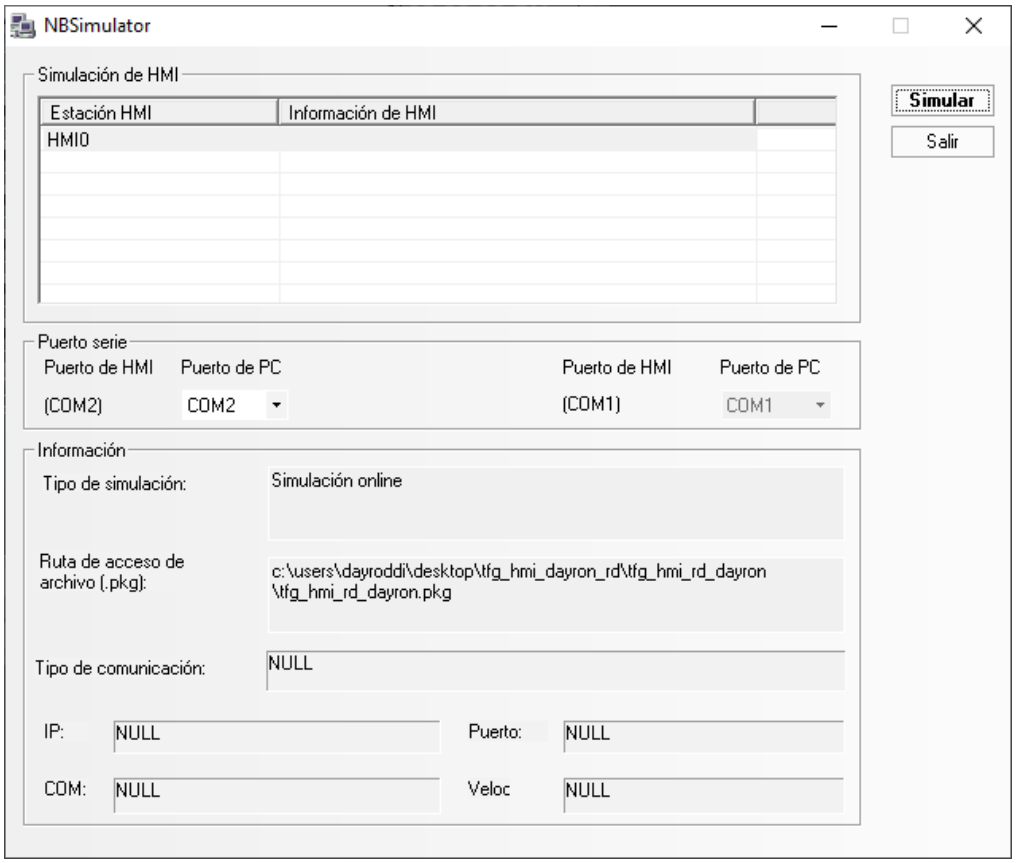
simulación de las pantallas, pero se adelanta al lector que el que se emplea es *Simulación Online Directa*.

Tabla 11. Descripción de los tres métodos de simulación de una pantalla en NB Designer.

Método	Descripción
Simulación Offline	Se emplea como paso previo a la descarga del programa a la pantalla real, no requiere conexión ni intercambia datos con ningún dispositivo, su uso se basa más bien para comprobar funcionalidad e interoperabilidad del diseño.
Simulación Online Indirecta	Método para comprobar una correcta comunicación de todos los parámetros con el *PLC*, sin embargo, en este caso la simulación de la interfaz se realiza en el *PC* pero la adquisición de los datos y la comunicación real con el autómata la implementa un *HMI* físico.
Simulación Online Directa	Es un método equivalente a la descarga del proyecto en la pantalla física y la interconexión de esta con el *PLC*, con la salvedad de que se evita la descarga del código en el *HMI*, es una simulación perfecta con un tiempo de ejecución máximo de 15 minutos.

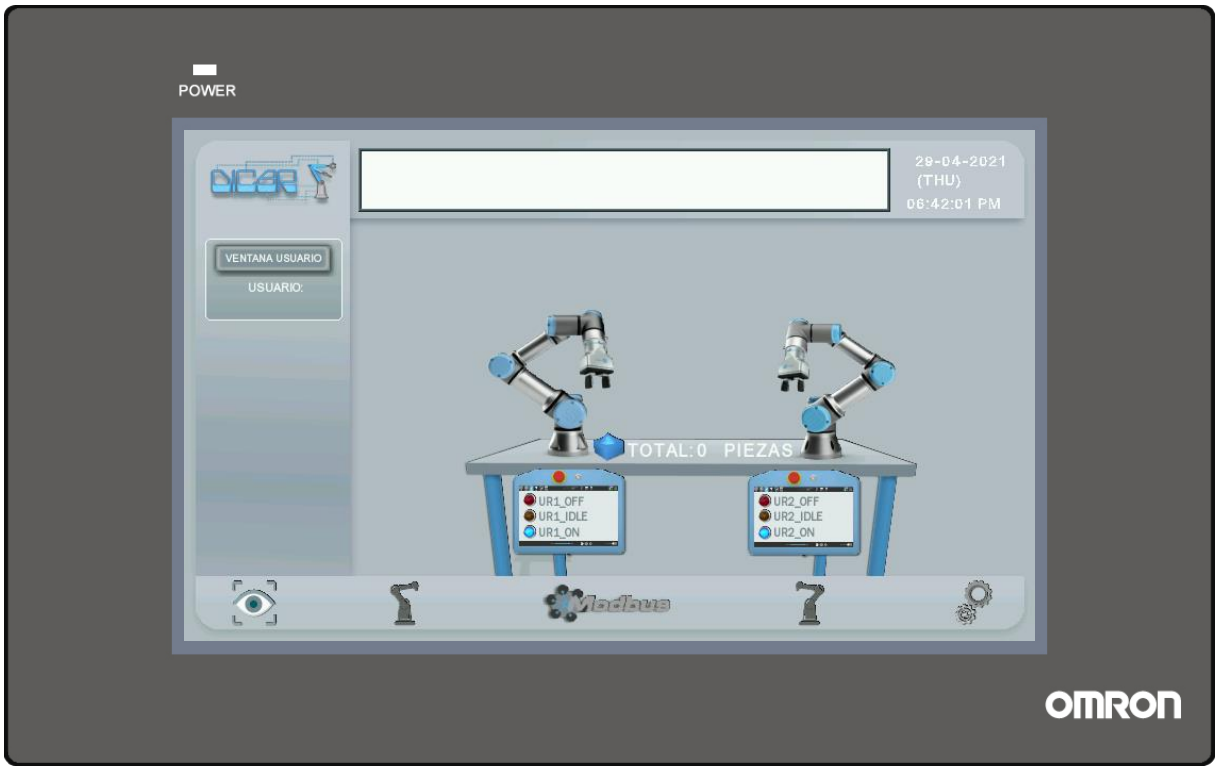
Si el compilador arroja un resultado ***OK***, se presiona sobre la opción *Simulación Online Directa*, apareciendo una ventana emergente que indica el dispositivo *HMI* que se va a simular así como el puerto a través del cual realizará la conexión con el *PLC* (Figura 116).

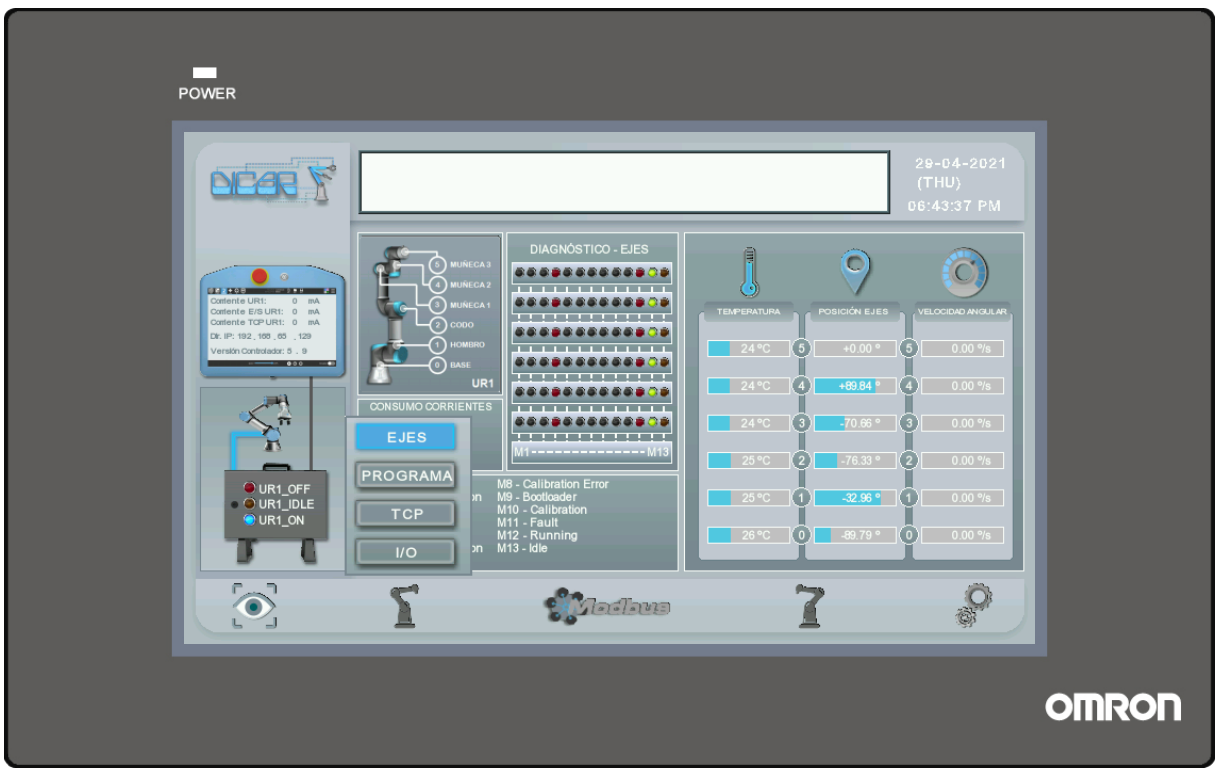
Figura 116. Ventana emergente previa a la simulación del proyecto.



Lanzado el simulador aparece una ventana flotante en el *PC* con la apariencia física de la pantalla *NB10W-TW01B* (Figura 117), permitiendo interactuar con el dispositivo tal y como se llevaría a cabo con el modelo físico real. El sistema es plenamente funcional y representa los mismos datos que muestra la pantalla equivalente en *WebServer* en tiempo real, garantizando el comportamiento del dispositivo como un verdadero *Digital Twin* de la celda física.

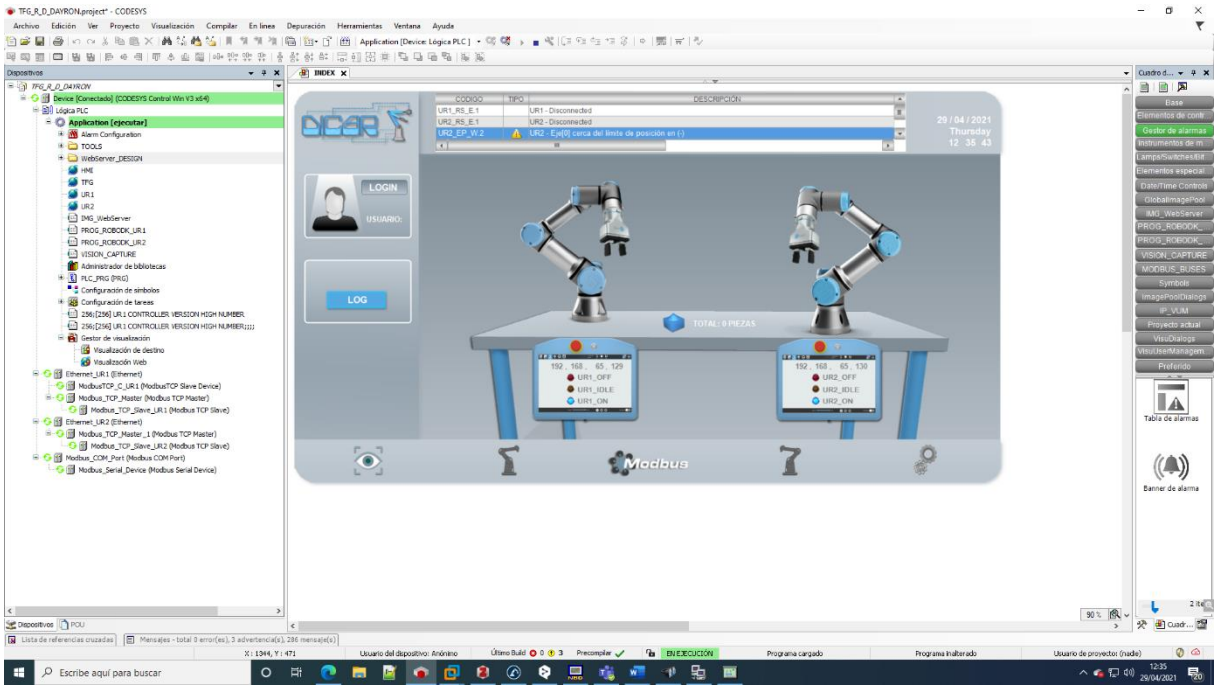
Figura 117. Capturas de pantalla del simulador de NB Designer ejecutando el diseño programado equivalente al WebServer.





La simulación de la pantalla de *Omron* era el último componente vinculado físicamente a las interfaces de comunicación configuradas en *CODESYS*, una vez ejecutándose la simulación del *HMI*, previamente activando el enlace de los puertos ***COM1*** y ***COM2***, se puede observar como todos los módulos del *PLC* se encuentran en un estado de *Ejecución sin errores* (Figura 118).

Figura 118. PLC en CODESYS ejecutándose con todos los módulos funcionando sin errores.



De esta forma queda planteada toda la lógica de conexión y configuración de los elementos fundamentales del proyecto, restando únicamente, el análisis y descripción de la conexión del *PLC* con la *Nube* de *UBIDOTS* y la plataforma *CODESYS Automation Server.*

5.7. CLOUD COMPUTING TOOLs

"La computación en nube es una evolución de la tecnología de la información y un modelo de negocio dominante para el suministro de recursos informáticos. Con la computación en nube, los individuos y las organizaciones pueden obtener acceso a la red bajo demanda a un conjunto compartido de recursos informáticos gestionados y escalables, como servidores, almacenamiento y aplicaciones. Recientemente, tanto los académicos como los profesionales han prestado mucha atención a la computación en nube. Dependemos en gran medida de los servicios en la nube en nuestra vida cotidiana, por ejemplo, para almacenar datos, escribir documentos, gestionar negocios y jugar en línea. La computación en la nube también proporciona la infraestructura que ha impulsado tendencias digitales clave como la computación móvil, el Internet de las cosas, el big data y la inteligencia artificial, acelerando así la dinámica de la industria, perturbando los modelos de negocio existentes e impulsando la transformación digital. Sin embargo, la computación en nube no solo ofrece un gran número de ventajas y oportunidades, sino que también conlleva varios retos y preocupaciones, por ejemplo, en lo que respecta a la protección de los datos de los clientes." (Sunyaev, 2020).

Es imprescindible para un ingeniero en la rama industrial tener nociones de computación volcada en la *Nube*, por ello, el autor consideró que el proyecto debía ser capaz de intercambiar datos con algún servicio de cómputo remoto, para cumplir con el objetivo: ***Programar un Dashboard en una plataforma de cloud computing para monitorizar parámetros de la celda y configurar en qué posición el primer robot debe entregar la pieza.*** Es por ello por lo que a continuación se describen los pasos fundamentales para solventar la meta de incluir el *cloud computing* en la celda.

5.7.1. OPC UA

"Es un estándar de tipo "plug and play" para el control y la automatización de procesos. Permite crear sistemas abiertos, sin depender de "marcas", ofreciendo mayor posibilidad de integrar sistemas en cualquier nivel (campo, control, planta…). Los fabricantes de HW solo tendrán que preparar un conjunto de componentes software para que los clientes utilicen en sus aplicaciones.

El protocolo OPC Clásico es un estándar industrial creado en colaboración por grandes fabricantes de software y hardware para control y Microsoft. Basado en las tecnologías de Microsoft OLE (Object Linking and Embedding, Active X) y DCOM (Distributed Component Object Model) para facilitar la integración y compatibilidad entre aplicaciones de forma local o remota. OPC consiste en un conjunto de reglas, interfaces, propiedades y métodos para usar en aplicaciones de control de procesos y automatización. Se soporta en el principio Cliente/Servidor donde cada cliente interroga al servidor que contiene los datos que necesita.

Dentro del protocolo están definidos los siguientes servicios o módulos:

- *OPC DA (Data Access): Define mecanismos estándar para leer y/o escribir datos de proceso en tiempo real.*
- *OPC DA Server: contiene información sobre la configuración del servidor y sirve de contenedor para los objetos de tipo "grupo":*

- ***El objeto grupo**: sirve para organizar los datos que leen y escriben los clientes (ej.: valores en una pantalla HMI o en un informe de producción). Un grupo puede ser público, es decir, compartido por varios clientes OPC o local (privado) para un solo cliente.*
- ***El objeto item**: representa conexiones a fuentes de datos en el servidor (no son las fuentes de datos en sí). Tiene asociados los atributos:*
  - *Value: es la representación numérica del dato.*
  - *Quality: indicador de la fiabilidad del dato.*
  - *Time Stamp: momento exacto en el que fue capturado el dato.*

- *OPC DA Client: accede a los items OPC a través de los grupos que ofrece el servidor. Pueden definir el ritmo al cual el servidor les informa sobre cambios en los datos. Existen dos tipos de acceso a un servidor OPC:*
  - ***Síncrono**: el cliente lee/escribe los datos de/en la caché del servidor a una cadencia determinada (para pequeñas cantidades de datos).*
  - ***Asíncrono** (subscripción): el cliente indica al servidor qué datos quiere leer y el servidor le informa cada vez que éstos cambian (más eficiente).*

*En resumen **OPC UA** (Unified Architecture) se crea en 2008 como un protocolo libre, independiente de la plataforma hardware que se emplee, está disponible para Windows, Mac, Linux, etc. (desde un microcontrolador ARM hasta un servidor cloud).Incorpora mecanismos de seguridad (autenticación, encriptación, etc.…).”* (Grado en Ingeniería Electrónica et al., s. f.).

5.7.2. UaExpert

El *UaExpert* es un cliente *OPC UA* completo que demuestra las capacidades de SDK/Toolkit de cliente *C++ OPC UA*. El programa está diseñado como un cliente de prueba de propósito general que soporta características de *OPC UA* como *Data Access*, *Alarmas & Condiciones*, *Acceso Histórico* y llamada a *Métodos UA*. *UaExpert* es un cliente de prueba *OPC UA* multiplataforma programado en *C++* y utiliza la sofisticada biblioteca *GUI QT* de *Nokia* (anteriormente *Trolltech*) formando el marco básico que es extensible por *Plugins*, la versión gratuita de *UaExpert* contiene activados los siguientes:

- Vista de acceso a datos *OPC UA*
- Vista de Alarmas y Condiciones *OPC UA*
- Vista de tendencias históricas de *OPC UA*
- Vista de Diagnóstico del Servidor
- Plugin Simple *Datalogger CSV*
- Plugin de rendimiento de *OPC UA*
- Plugin *GDS Push-Model*
- Vista *XMLNodeSet-Export* (requiere licencia)
- *UaExpert* está disponible para *Windows* y *Linux*.

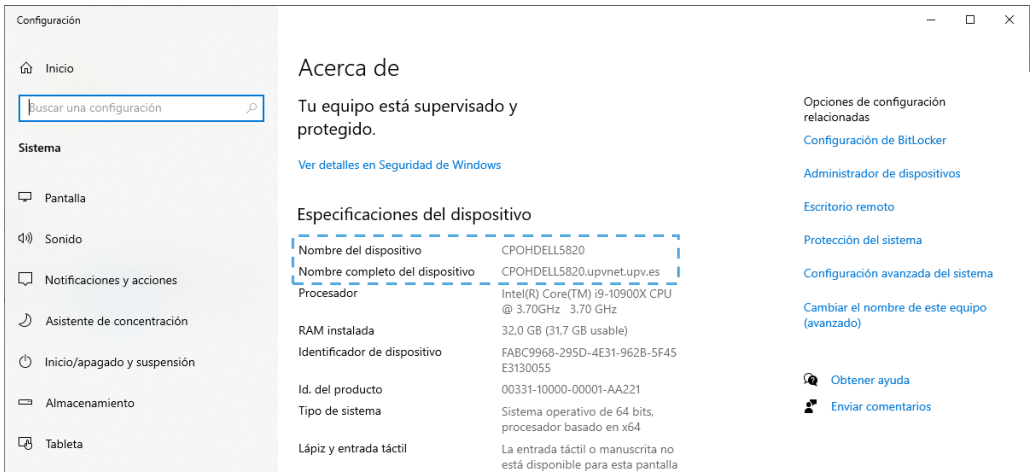
El programa no es necesario para el intercambio de datos entre el *PLC* y la plataforma de *cloud computing*, pero constituye una herramienta de diagnóstico que el autor empleó durante la fase de

investigación con bastante frecuencia, es por ello por lo que se incluye en el presente informe, pues se considera que el lector podría sacar provecho de una herramienta tan funcional y gratuita.

Si el *softPLC* de *CODESYS* se está ejecutando, al agregar un *Configurador de símbolos* que comparte variables por *OPC UA*, este componente crea un *Servidor OPC UA* que devolverá el valor de una variable consultada por un cliente, para visualizar estos datos se ejecuta *UaExpert*[25] que actúa como *OPC UA Client*, posteriormente hay que enlazar el *Server* que se corresponde al *softPLC*, presionando *clic derecho/ + Add,* (Figura 119.b), sobre *Servers*, la ventana emergente muestra los servidores de *OPC UA* que se encuentran activos en el *PC*, en el caso del proyecto, *CODESYS* crea el servidor con la misma denominación que tiene el *PC* (Figura 119.a) sobre el cual se está ejecutando [***CPOHDELL5820***], la ventana detecta el servidor con el formato:

OPCUAServer@nombre_del_PC → *OPCUAServer@CPOHDELL5820*

Figura 29. a. Ventana propiedades de PC (Windows 10). b. Configuración del servidor OPC UA en UaExpert.



[25] Véase UaExpert "UA Reference Client" - Unified Automation (unified-automation.com)

Enlazado el *Servidor* a *UaExpert* solo queda conectarlo para escanear las variables disponibles, para ello, estando seleccionado OPCUAServer*@CPOHDELL5820*, en el menú principal opción *Server /Connect* es posible llevar a cabo el vínculo (Figura 120). Cuando se logra conectar con *CODESYS* en el panel izquierdo se dispone de un árbol de navegación relativo al *Servidor* conectado, como se puede ver en la imagen presentada, el objeto vinculado a *CPOHDELL5820* es *CODESYS Control Win V3 x64* y coincide con el nombre del *PLC* configurado en *CODESYS* (Figura 32, página 35).

Dentro del *PLC* están contenidos todos los programas que comparten variables a través de *OPC UA*, así como las *tablas de variables globales* del proyecto, cada una de ellas con sus respectivas variables. Desplegando cualquiera de estos elementos aparecen las variables independientes, arrastrándolas a la ventana central del programa se muestran las características que se necesitarán posteriormente, entre ellas están:

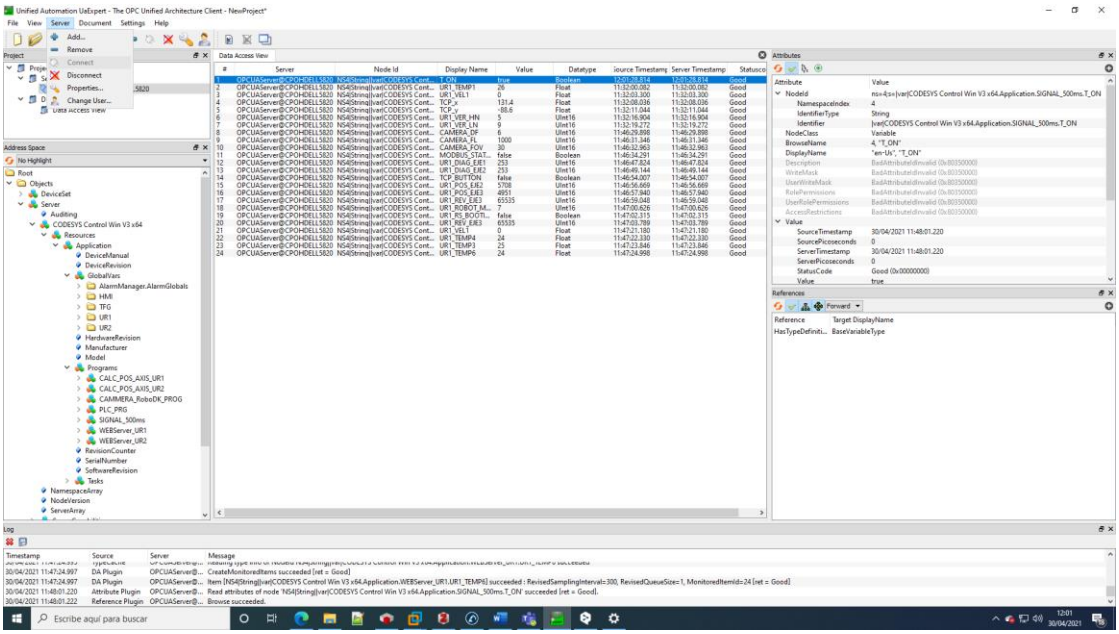
- **#:** Lista numéricamente los elementos en la ventana.
- **Server:** Indica a que servidor pertenece la variable relativa al *id* en cuestión.
- **NodeID:** Es el campo de mayor importancia, las variables en el contexto *OPC UA* se denominan *nodos*, a su vez se componen de tres atributos:
    1. *NamespaceIndex (ns):* Espacio de nombres en el que se encuentra el nodo, en el caso del proyecto este campo se corresponde a *ns = 4*.
    2. *Identifier:* Identificador único del nodo dentro de su espacio de nombres.
    3. IdentifierType: Tipo de nodo: *String*, *Guid* y *Numeric*.

Para el caso de la variable ***T_ON*** del programa *SIGNAL_500ms* de *CODESYS*, el ***NodeID*** es:

ns=4;s=|var|CODESYS Control Win V3 x64.Application.SIGNAL_500ms.T_ON

Estos atributos representan el llamado ***NodeID*** y son requeridos por algunos bloques de función. Con la ayuda de *UaExpert* es posible determinar simplemente los atributos de un nodo estableciendo una conexión con el *Servidor OPC UA* y navegando hasta el nodo deseado, los atributos son visibles en el panel de Atributos.

Figura 120. Visualización de varias variables disponibles a través de OPC UA en el PLC de CODESYS empleando UaExpert.



5.7.3. INSTALACIÓN NODE-RED

*"Node-RED es una herramienta de programación visual que permite al usuario programar sin tener que escribir una línea. **E**s un editor de flujo basado en un navegador Web donde se puede añadir o eliminar nodos y conectarlos entre sí con el fin de hacer que se comuniquen entre ellos**.** Además, conectar los dispositivos de hardware, APIs y servicios en línea es una tarea simple. La herramienta se ha convertido en el estándar open-source en entornos de Industria 4.0, IoT, Marketing digital o sistemas de Inteligencia Artificial, entre otros, que gestiona y procesa los datos en tiempo real, logrando simplificar los procesos entre productores y consumidores de información.*

La sencillez de aprendizaje y uso, que no requiere de conocimientos de programación, su robustez y la necesidad de bajos recursos de cómputo ha permitido que hoy en día se encuentre integrado en prácticamente la mayoría de dispositivos IoT e IIoT del mercado, así como equipos Raspberry, sistemas cloud o equipos locales.

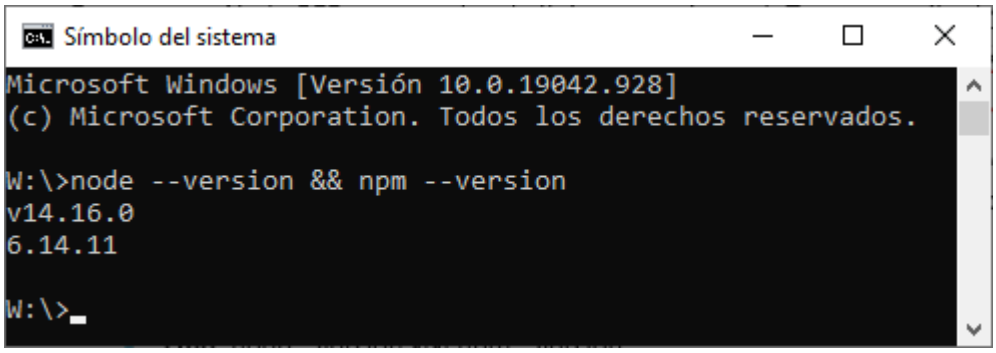
Node-RED está creado a partir de NodeJS[26] y la librería de JavaScript D3.js. NodeJS proporciona la potencia suficiente para que Node-RED sea fiable y escalable. NodeJS es un software muy potente que permite la programación en JavaScript del lado del servidor.

En resumen, Node-RED es un motor de flujos con enfoque IoT, que permite definir gráficamente flujos de servicios, a través de protocolos estándares como REST, MQTT, Websocket, AMQP… además de ofrecer integración con APIs de terceros, tales como Twitter, Facebook, Yahoo!…" (jecrespom, 2020).

Una vez instalado *NodeJS* y *npm*[27], empleando una ventana *Símbolo de sistema* o el *PowerShell* de *Windows* se comprueba que la instalación se desarrolló de forma correcta a través del siguiente código (Figura 121):

- **Powershell**: node --version; npm –version
- **cmd**: node --version && npm --version

*Figura 121. Comprobación de las versiones instaladas de **NodeJS** y **npm.***



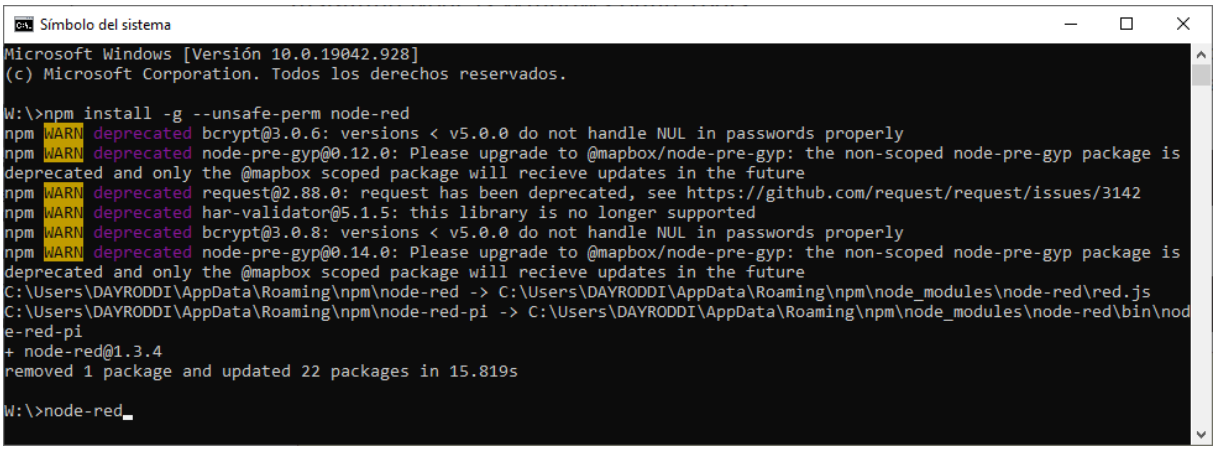
[26] Es necesario instalar NodeJS previo a la instalación de Node-Red. Se puede descargar el programa de forma gratuita desde: **Node.js (nodejs.org)**

[27] **npm (Node Package Manager):** es un gestor de paquetes desarrollado en su totalidad bajo el lenguaje JavaScript a través del cual se puede obtener cualquier librería con tan solo una sencilla línea de código, lo cual permite agregar dependencias de forma simple, distribuir paquetes y administrar eficazmente tanto los módulos como el proyecto a desarrollar en general.

Para instalar *Node-Red* es necesario abrir nuevamente una consola *Símbolo de Sistema* o *PowerShell* en *Windows* y ejecutar el siguiente código (Figura 122):

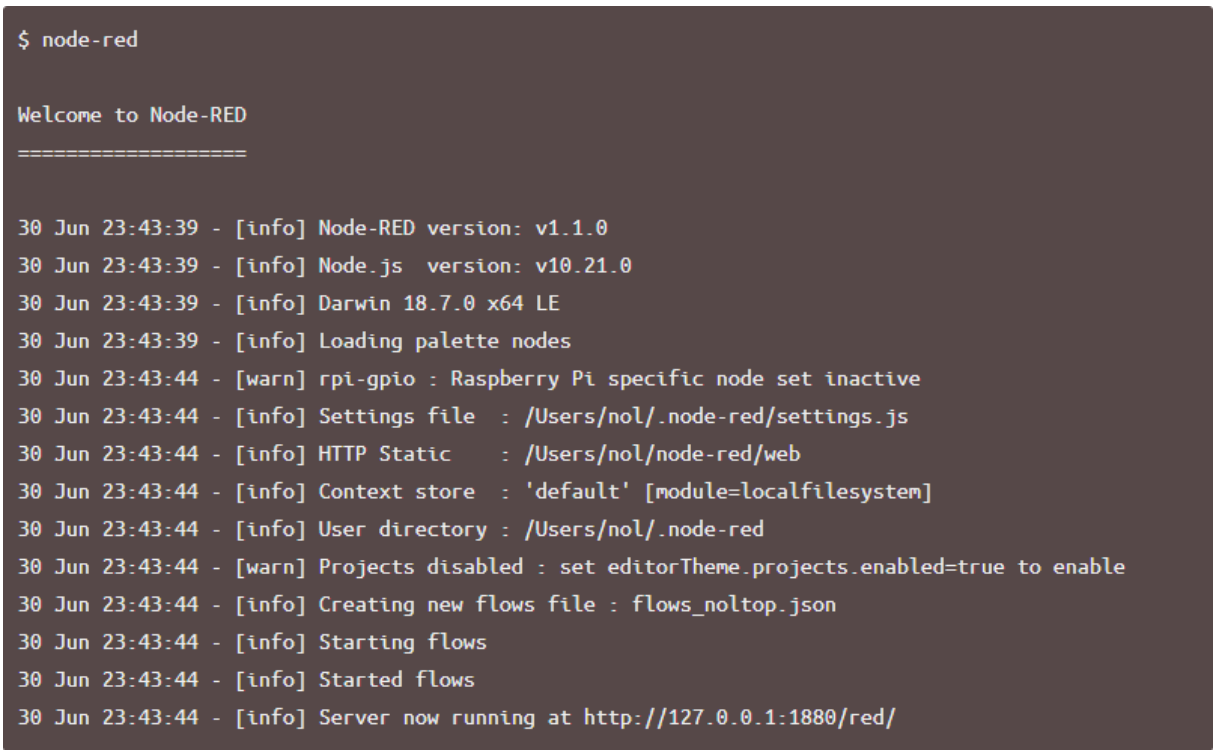
- **cmd o Powershell:** npm install -g --unsafe-perm node-red

Figura 122. Código para instalar Node-Red y ejecución de la herramienta.



Una vez completada la instalación se ejecuta en la misma ventana ***cmd*** el código [**node-red**], tal como se muestra en la (Figura 123) la herramienta entrará en funcionamiento ***siempre que la ventana Símbolo del sistema no se cierre***.

Figura 123. Node-Red en ejecución.

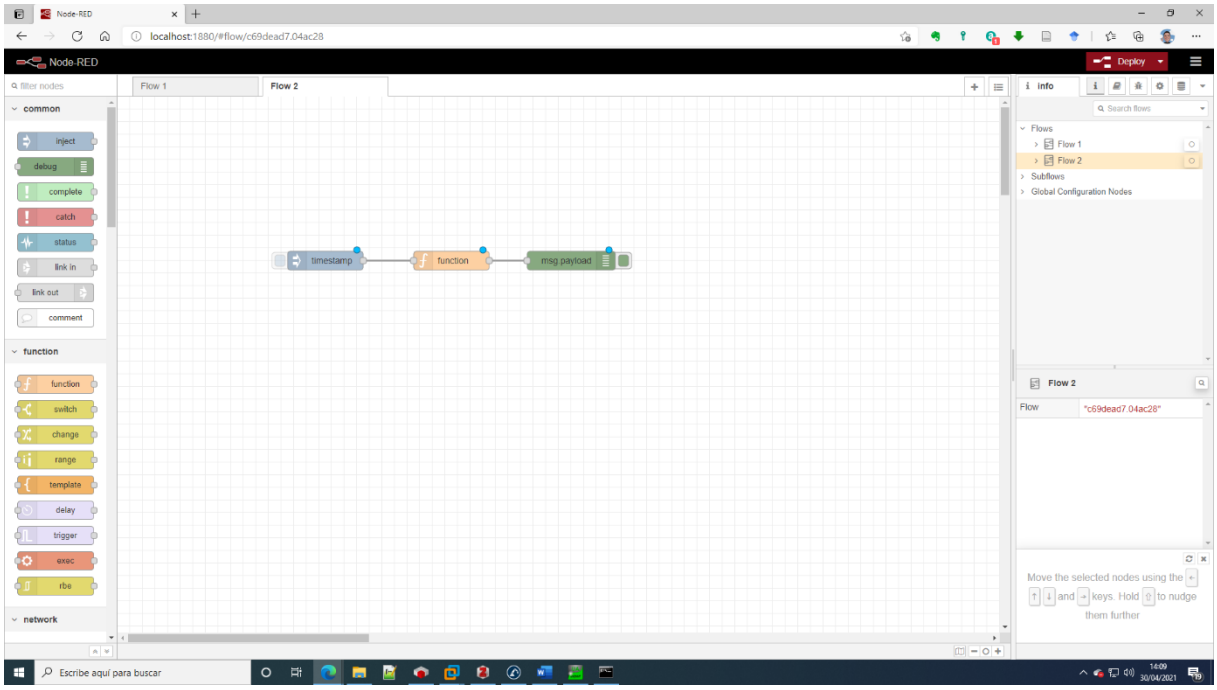


Fuente: Running Node-RED locally : Node-RED (nodered.org)

A continuación, se puede acceder al editor de *Node-RED* escribiendo en la barra de dirección de un navegador: ***http://localhost:1880***.

El editor de *Node-Red* (Figura 124) se compone de un espacio de trabajo, en el centro del navegador, un panel con *nodos* a la izquierda y una ventana multifuncional a la derecha, en la cual se puede visualizar *información* de los *Flow*[28], mostrar la *ayuda* relativa a un nodo específico, una barra lateral de *depuración* que muestra los mensajes pasados a los nodos de depuración dentro del flujo, así como ciertos mensajes de registro del tiempo de ejecución, opciones de *configuración* y una última pestaña con *datos de contexto*.

*Figura 124. Vista general del editor de Node-Red en un navegador **Edge** de Windows.*



Los nodos son la unidad básica de programación de *Node-Red*, a través de ellos se implementa toda la lógica de programación gráfica, este procedimiento libera al usuario de conocer el lenguaje *JavaScript* subyacente tras los elementos interconectados y permite el desarrollo de múltiples tareas de forma ágil.

El objetivo de utilizar esta herramienta en el proyecto es poder comunicar la plataforma *UBIDOTS* con el *PLC* de *CODESYS*, donde *Node-Red* actúa como un intermediario entre ambos módulos, capturando datos de las variables compartidas por el autómata, procesándolas, enrutándolas hacia la nube y viceversa.

5.7.4. MQTT

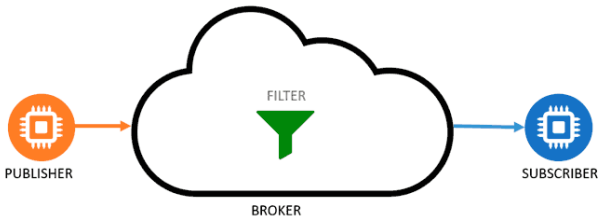
*"MQTT (Message Queue Telemetry Transport) es un protocolo de comunicación Machine to Machine (M2M) y un protocolo de transporte de mensajes Cliente/Servidor basado en publicaciones y subscripciones a los denominados "TOKEN". Se ha convertido en **uno de los principales pilares del IoT** por su sencillez y ligereza.*

[28] Un *Flow* se representa como una pestaña dentro del espacio de trabajo del editor y es la principal forma de organizar los nodos.

MQTT implementa enlaces entre dispositivos con conexión IoT, lo cual simplifica y hace más sencilla la recogida de datos de sensores, la publicación de los diferentes valores obtenidos y la configuración remota de los nodos. El protocolo funciona sobre TCP/IP o sobre otros protocolos de red con soporte bidireccional y sin pérdidas de datos. En el caso de MQTT cada conexión se mantiene abierta y se "reutiliza" en cada comunicación. Es una diferencia, por ejemplo, a una petición HTTP 1.0 donde cada transmisión se realiza a través de conexión.

Es un servicio de mensajería push con patrón publicador/suscriptor (pub-sub). Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el bróker le hará llegar los mensajes suscritos (Figura 125).

Figura 125. Diagrama de funcionamiento del protocolo MQTT.



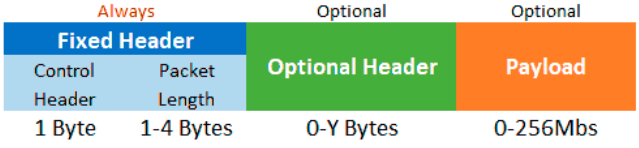
Fuente: ¿Qué es MQTT? Su importancia como protocolo IoT (luisllamas.es)

Los clientes inician una conexión TCP/IP con el bróker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS. Para ello el cliente envía un mensaje CONNECT que contiene información necesaria (nombre de usuario, contraseña, client-id…). El bróker responde con un mensaje CONNACK, que contiene el resultado de la conexión (aceptada, rechazada, etc.).

Uno de los componentes más importantes del protocolo MQTT es la definición y tipología de los mensajes, ya que son una de las bases de la agilidad en la que radica su fortaleza. Cada mensaje consta de 3 partes (Figura 126):

- ***Cabecera fija.*** *Ocupa 2 a 5 bytes, obligatorio. Consta de un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje. La longitud se codifica en 1 a 4 bytes, de los cuales se emplean los 7 primeros bits, y el último es un bit de continuidad.*
- ***Cabecera variable.*** *Opcional, contiene información adicional que es necesaria en ciertos mensajes o situaciones.*
- ***Contenido (payload).*** *Es el contenido real del mensaje. Puede tener un máximo de 256 Mb aunque en implementaciones reales el máximo es de 2 a 4 kB."* («¿Qué es MQTT?», s. f.).

Figura 126. Estructura de un mensaje MQTT.



Fuente: ¿Qué es MQTT? Su importancia como protocolo IoT (luisllamas.es)

5.7.5. JSON

*"JavaScript Object Notation (JSON) es un formato ligero basado en los tipos de datos del lenguaje de programación JavaScript. En su esencia, los documentos JSON son diccionarios formados por pares clave-valor, donde el valor puede ser de nuevo un documento JSON, lo que permite un nivel arbitrario de anidamiento. La (*Figura 127*) muestra un ejemplo de dato JSON.*

Figura 127. Ejemplo de dato empleando el formato JSON.

```
{
  "name": {
     "first": "John",
     "last": "Doe"
     },
  "age": 32,
  "hobbies": ["fishing","yoga"]
}
```

Fuente: JSON: Data model and query languages - ScienceDirect

Debido a su simplicidad y al hecho de que es fácilmente legible tanto por humanos como por máquinas, JSON se está convirtiendo rápidamente en uno de los formatos más populares para el intercambio de datos en la Web. Esto es especialmente evidente en los servicios web que se comunican con sus usuarios a través de una interfaz de programación de aplicaciones (API), ya que JSON es actualmente el formato predominante para enviar solicitudes y respuestas de API a través del protocolo HTTP." (Bourhis et al., 2020).

"La sintaxis de JSON es bastante sencilla, existen dos tipos de elementos:

- ***Matrices:** Las matrices son listas de valores separados por comas. Las matrices se escriben entre corchetes [ ].*

  ```
  [1, "prueba", 37.38, "Prueba Cadena"]
  ```

- ***Objetos** (objects): Los objetos son listas de parejas clave / valor. El nombre y el valor están separados por dos puntos : y las parejas están separadas por comas. Los objetos se escriben entre llaves { } y los nombres de las parejas se escriben siempre entre comillas dobles.*

  ```
  {"nombre": "Loren Ipsum", "valor": 47, "Parámetro": verdadero}
  ```

Los valores (tanto en los objetos como en las matrices) pueden ser:

- *Números: enteros, decimales o en notación exponencial. El separador decimal es el punto ., un número negativo empieza por el signo menos - y el indicador de la notación exponencial es e o E. Los números positivos no pueden empezar por el signo +, los números no pueden empezar por varios ceros o por un cero seguido de otra cifra.*
- *Cadenas: Las cadenas se escriben entre comillas dobles. Los caracteres especiales y los valores Unicode se escriben con una contrabarra **** delante. Los caracteres que deben escribirse siempre como caracteres especiales son **\"** (comillas), **** (contrabarra), **\b** (retroceso), **\f** (salto de página), **\n** (salto de línea), **\r** (retorno de carro), **\t** (tabulador) y los caracteres Unicode (**\u**).*

El carácter / (barra) puede escribirse como carácter / o como carácter especial \/ (suele ser necesario cuando el contenido es código HTML y la barra indica un cierre de etiquetas).

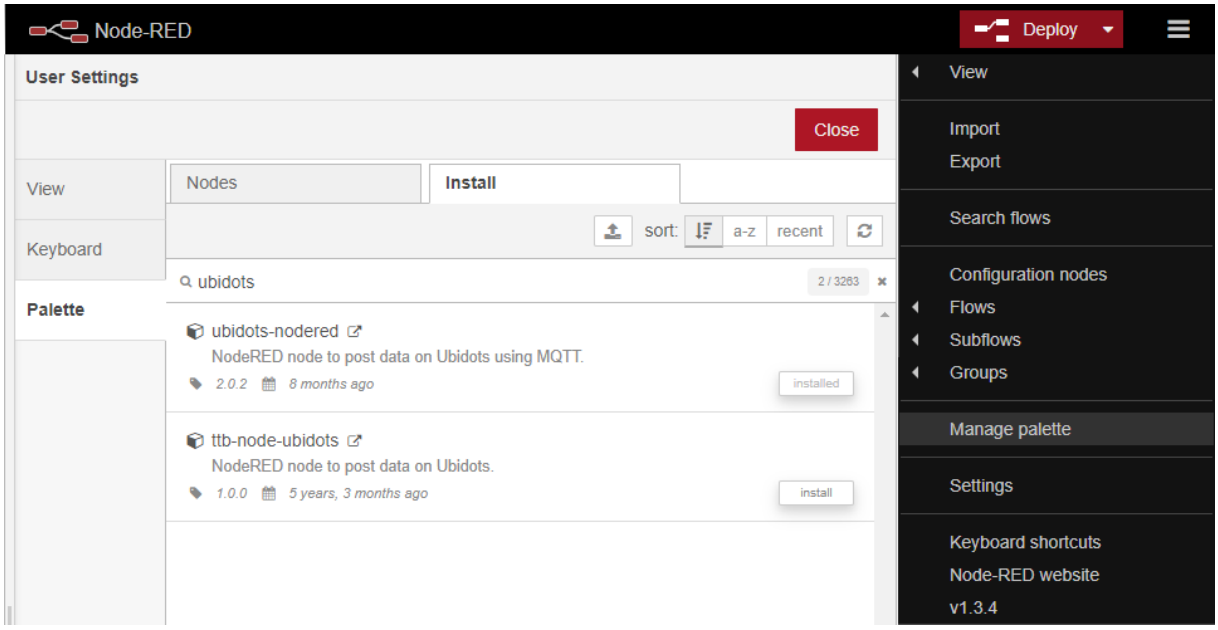
- *Los valores true, false y null se escriben sin comillas.*
- *Pueden existir objetos y matrices dentro de objetos y de matrices, sin límite de anidamiento.*
- *Los ficheros JSON no pueden contener comentarios."*

(*JSON. Formatos. Informática. Bartolomé Sintes Marco. www.mclibre.org*, s. f.)

5.7.6. EDITOR NODE-RED

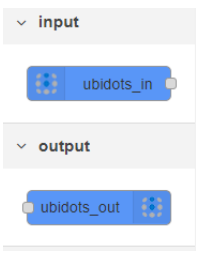
La información arriba citada relativa a *MQTT* y *JSON* es de vital importancia para entender cómo se va a programar en el editor de *Node-Red* a continuación. Para establecer la comunicación entre la plataforma de *UBIDOTS* y la herramienta que ocupa este apartado, es necesario descargar en el espacio de trabajo una biblioteca que permite transmitir y recibir datos empleando *MQTT* desde y hacia la *Nube*, dicha biblioteca se denomina *ubidots-nodered* (Figura 128), el procedimiento consiste en acceder al menú (≡) opción *Manage Palette*, en la ventana flotante escribir el nombre de la biblioteca antes mencionado en la pestaña *Install* y finalmente cuando encuentre el resultado presionar *Install.*

Figura 128. Proceso de instalación de la biblioteca UBIDOTS-NODERED.



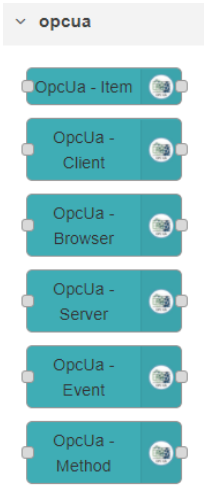
Al concluir el proceso de instalación, *Node-Red* agrega en el panel de nodos a la izquierda del navegador dos nuevos elementos (Figura 129).

Figura 129. Nodos creados tras la instalación.



Para la gestión de los datos entre *Node-Red* y *CODESYS* se utilizarán las variables enviadas a través del protocolo *OPC UA*, lo que hace necesario instalar en *Node-Red* otra biblioteca, repitiendo el mismo proceder anterior pero con la nueva denominación: *node-red-contrib-opcua.* Tras concluir, se encuentra un nuevo conjunto de nodos disponibles en el panel izquierdo (Figura 130), con estos pasos finalizados la herramienta se encuentra a punto para comenzar la programación.

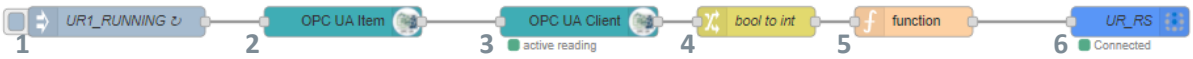
Figura 130. Nodos de Node-Red para establecer intercambio de datos utilizando OPC UA.



Tal y como se comentó en el apartado relativo al diseño del *HMI*, no es objetivo del proyecto crear un *Dashboard* en *UBIDOTS* con todos los datos representados previamente en el *WebServer*, sin embargo, si se mostraran datos de tipo ***BOOL*** y ***REAL***, además, de un tráfico de información bidireccional, de esta forma el lector tiene todo el espectro de posibilidades explicadas para implementar sus propios diseños con un soporte previo.

El primer conjunto de datos que se enviará desde *Node-Red* a *UBIDOTS* será el estado de los dos *UR*, para ello se emplean tres variables para cada elemento, *UR_RUNNING*, *UR_IDLE* y *UR_STOPPED*, se describirá el conjunto de nodos necesarios para transmitir a la *Nube* solo la primera de las variables, el resto (5), se programan de igual forma con la salvedad del *ID* correspondiente a la variable (Figura 131).

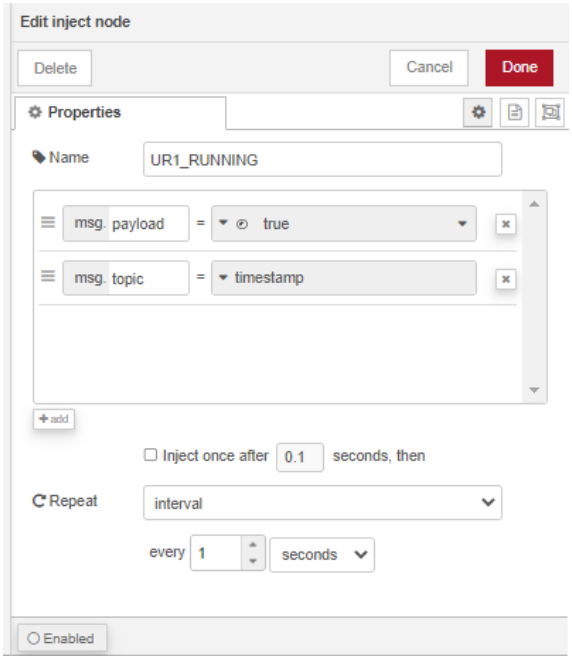
Figura 131. Nodos empleados para enviar el valor de UR1_RUNNING a UBIDOTS.



Los elementos presentados en la imagen anterior se configuran del siguiente modo.

1. ***UR_RUNNING***: Es un nodo de tipo *Inject* que se encuentra en la biblioteca *commons*, en el panel izquierdo del navegador, presionando *doble clic* se abre la ventana de edición del elemento, en la cual se puede asignar un nombre al nodo, se precisa el tipo de dato que recibe el campo *payload* y el *topic*¸ se define que será un objeto que realizará una petición de lectura cada cierto tiempo y de forma cíclica. Los valores de la configuración específica para el elemento *Inject UR1_RUNNING* se muestran en la (Figura 132).

*Figura 132. Edición de los parámetros del objeto **Inject** para la lectura de la variable UR1_RUNNING.*

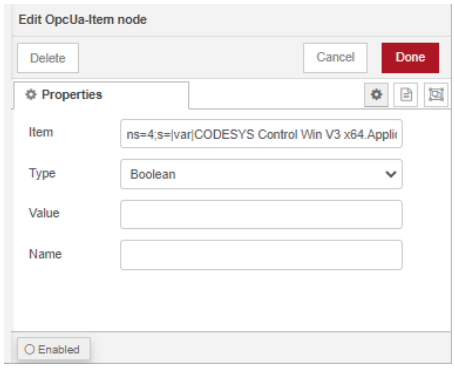


INJECT (help): Inyecta un mensaje en un flujo de forma manual o a intervalos regulares. El *payload* del mensaje puede ser de varios tipos, como cadenas, objetos *JavaScript* o la hora actual. El nodo *Inject* puede iniciar un flujo con un valor específico. El *payload* por defecto es una marca de tiempo (*timestamp*) de la hora actual en milisegundos desde el 1 de enero de 1970. El nodo también admite la inyección de cadenas, números, BOOLeanos, objetos *JavaScript* o valores de flujo/contexto global. Por defecto, el nodo se activa manualmente haciendo clic en su botón dentro del editor. También puede configurarse para que se inyecte a intervalos regulares o según una programación o para que inyecte una vez cada vez que se inicien los flujos. El intervalo máximo que puede especificarse es de unas 596 horas / 24 días. Sin embargo, si se trata de intervalos superiores a un día, se debe considerar el uso de un nodo programador que pueda hacer frente a cortes de energía y reinicios.

2. ***OPC UA Item***: Nodo donde se especifica el ***NodeID*** con el cual *CODESYS* está compartiendo la variable que se intenta leer para enviarla a *UBIDOTS*, tal y como se analizó en el apartado *UaExpert*, para *UR1_RUNNING* el tipo de datos es *BOOLean y* su correspondiente ***NodeID*** es: (Figura 133)

ns=4;s=|var|CODESYS Control Win V3 x64.Application.HMI.UR1_RUNNING

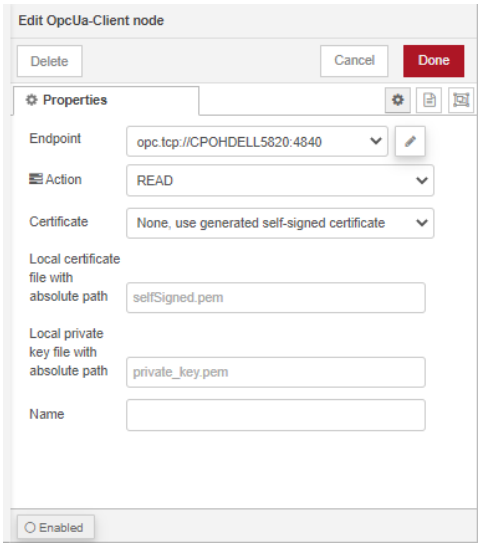
Figura 133. Parámetros del nodo OPCUA Item.



OPC UA Item (help): Define el elemento, el tipo y el valor de *OPC UA*. El elemento contiene una dirección *OPC UA* válida como ns=2;i=4 ó ns=3;s=*MiVariable.* No todos los tipos posibles se encuentran disponibles por ahora, pero pueden seleccionarse algunos. El valor es necesario solo si el elemento será escrito en el servidor *OPC UA*. Si el valor no se llena, *OpcUa-Item* envía el *payload* de entrada. Los valores de la matriz están delimitados por una coma y deben ser dados con la siguiente sintaxis: 5,4,3,2,1.

3. ***OPC UA Client***: La comunicación con el servidor *CODESYS OPC UA* es realizada por este nodo. *OPC UA Client* define como ***Endpoint*** la dirección y el puerto del servidor *OPC UA* (de forma general dicho campo es *opc.tcp: // dirección_IP: 4840*) pero en el caso del proyecto coincide con el nombre del *PC*, tal como se mencionó en el apartado relativo a *UaExpert* (*opc.tcp: // CPOHDELL5820: 4840*). (Figura 134).

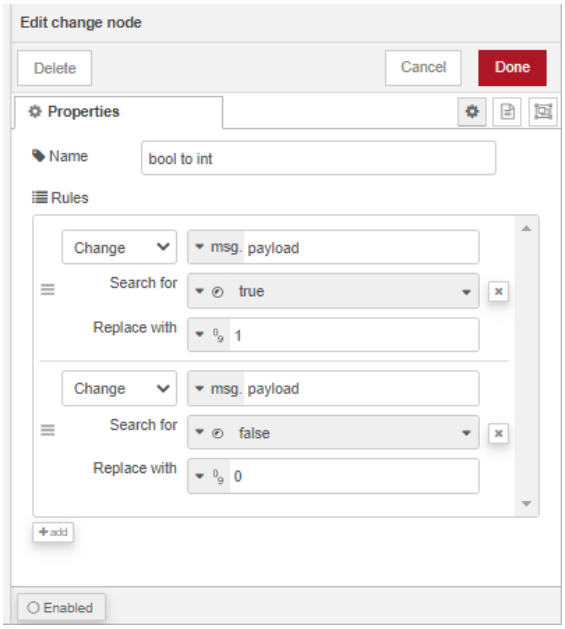
Figura 134. Configuración del nodo OPC UA Client.



OPC UA Client (help): Componente que actúa como cliente para el request de información al servidor de *CODESYS* a través del protocolo *OPC UA*, permite realizar un conjunto de acciones como: *lectura, escritura, suscripción, navegación, información, monitorización, etc.* El nodo puede realizar una lectura múltiple que se almacenará primero todos los valores de los *NodeID* de los mensajes de entrada en un array, siendo necesario solo un nodo de este tipo, sin embargo, para no complicar el procesamiento de los datos, teniendo que recorrer un array posteriormente, se coloca un nodo por cada variable.

4. ***CHANGE***: Posteriormente se verá que *UBIDOTS* procesa las variables *BOOL* como datos con el formato *JSON* que deben tener en el campo *payload* el valor ***1*** ó ***0***, es por ello por lo que se hace necesario preprocesar el dato leído por el nodo *2* y convertirlo de ***BOOL*** a ***INT*** para posteriormente, por medio de una *function* formatear el valor a *JSON*. La configuración de esta función se muestra en la (Figura 135).

*Figura 135. Función **CHANGE** para convertir el dato **BOOL a int**.*

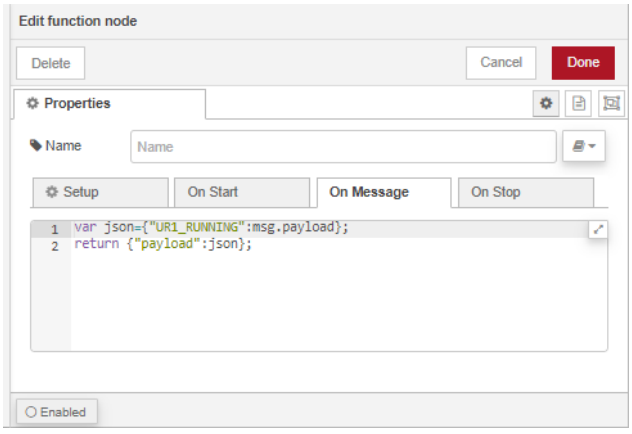


CHANGE (help): Permite establecer, cambiar, eliminar o mover las propiedades de un mensaje, contexto de flujo o contexto global. El nodo puede especificar múltiples reglas que se aplicarán en el orden en que se definan. Las operaciones disponibles son:

- Establecer: establece una propiedad. El valor puede ser de diferentes tipos, o puede tomarse de una propiedad de mensaje o contexto existente.
- Cambiar: Buscar y reemplazar partes de la propiedad. Si las expresiones regulares están habilitadas, la propiedad "reemplazar con" puede incluir grupos de captura, por ejemplo $1. Reemplazar solo cambiará el tipo si hay una coincidencia completa.
- Borrar: Eliminar una propiedad.
- Mover: Mover o renombrar una propiedad.

5. ***Function***: Toda función cuenta con entradas y salidas, la finalidad de este componente es modificar el valor de la variable de entrada almacenado en el campo *payload* y devolver dicho *payload* en formato *JSON* (Figura 136), para el correcto procesamiento del dato en *UBIDOTS*.

Figura 136. Función para convertir el valor del dato en formato JSON.

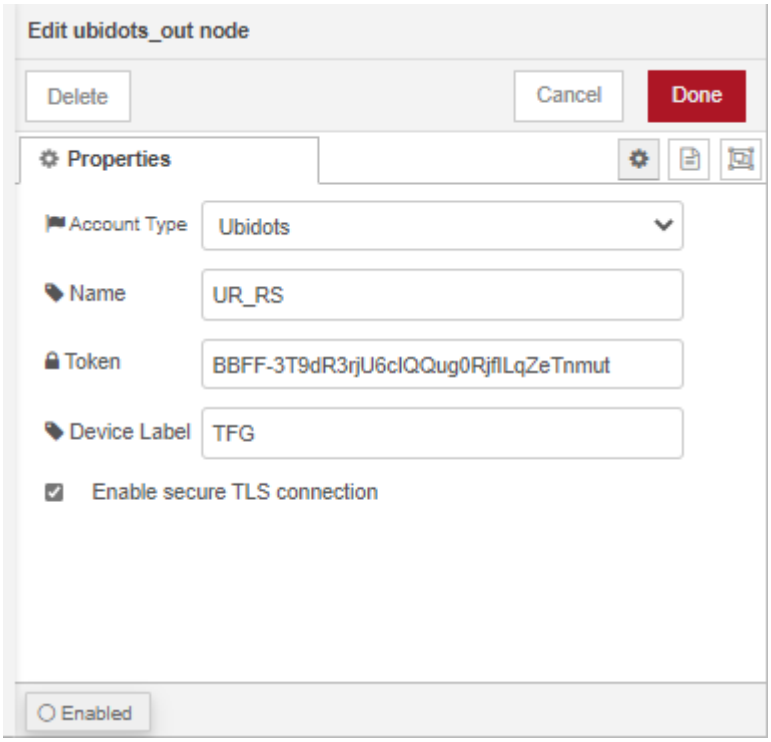


Function (help): Es una función JavaScript que se ejecuta contra los mensajes que recibe el nodo. Los mensajes se pasan como un objeto JavaScript llamado *msg*. Por convención tendrá una propiedad *msg.payload* que contiene el cuerpo del mensaje. Se espera que la función devuelva un objeto de mensaje (o varios objetos de mensaje), pero puede optar por no devolver nada para detener un flujo. La pestaña *Setup* contiene el código que se ejecutará cada vez que se inicie el nodo. La pestaña *Cerrar* contiene el código que se ejecutará cuando el nodo se detenga. Para una mejor visualización de la función se copia a continuación el código implementado:

```
var json={"UR1_RUNNING":msg.payload};
return {"payload":json};
```

6. ***UR_RS***: Componente que representa la plataforma de *UBIDOTS,* los parámetros necesarios para la configuración son el tipo de cuenta contra la cual se enviarán los datos, existiendo dos variantes, *Business* y *Education*. En el apartado relativo a *UBIDOTS* se detallan estos campos, además se asigna al componente un nombre, se copia el *Token* de la aplicación de la *Nube* y por último se indica cuál de los dispositivos creados en el *cloud computing* recibirá el dato. (Figura 137).

Figura 137. Configuración del nodo ***Ubidots_Out****.*



UBIDOTS_Out (help): Recibe un mensaje y utiliza sus valores para publicar una solicitud a la *API* de *UBIDOTS* a través de *MQTT*. Los campos para configurar son:

- Nombre: Etiqueta del nodo en el espacio de trabajo de *Node-Red*. Si está vacía, el valor predeterminado es "*UBIDOTS_Out*".

- Token (Obligatorio): Token necesario para autenticar la conexión con su cuenta. Para obtener el token, es necesario iniciar sesión en *UBIDOTS.com*, en "Mi perfil" haga clic en "Credenciales de la API".
- Etiqueta del dispositivo (Obligatorio): La etiqueta del dispositivo en el que se publicarán los datos. Si no existe ningún dispositivo con esta etiqueta, se creará automáticamente. Se puede enviar dinámicamente en el objeto de mensaje *JSON* de entrada con la clave: "*UBIDOTSDeviceLabel*". Si no se envía ninguna etiqueta de dispositivo en el mensaje, se devuelve por defecto el valor del campo *Device Label*.
- SSL: Por defecto todos los datos se envían encriptados vía TLS. Desmarque la casilla si los datos deben enviarse sin cifrar.

La configuración de los 6 nodos necesarios para enviar la variable *UR1_RUNNING,* se repite 5 veces más para mostrar en la *Nube* el estado de funcionamiento de los dos robot, la (Figura 138) muestra el código gráfico que materializa el objetivo mencionado.

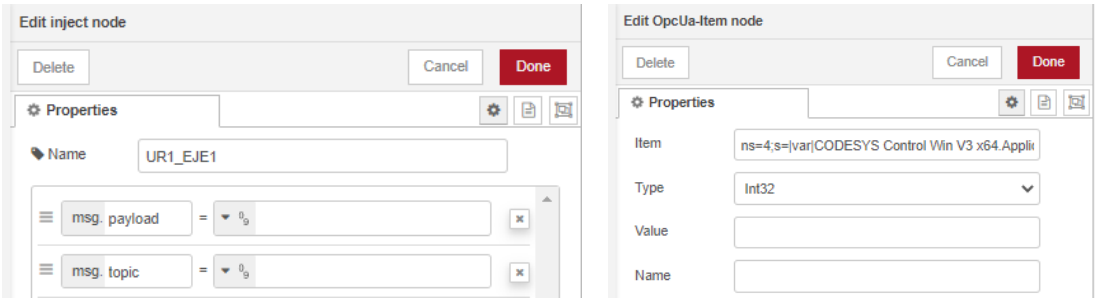
Figura 138. Programación gráfica para mostrar en el Dashboard de UBIDOTS el estado de funcionamiento de ambos robot UR.

El próximo conjunto de datos que se representa es el valor de cada eje del robot *UR1*¸ de forma similar al ejemplo anterior se muestra el procedimiento para un solo campo y el lector puede extrapolar el procedimiento (Figura 139).

Figura 139. Código gráfico para el envío del valor del EJE_1 de UR1 a UBIDOTS.

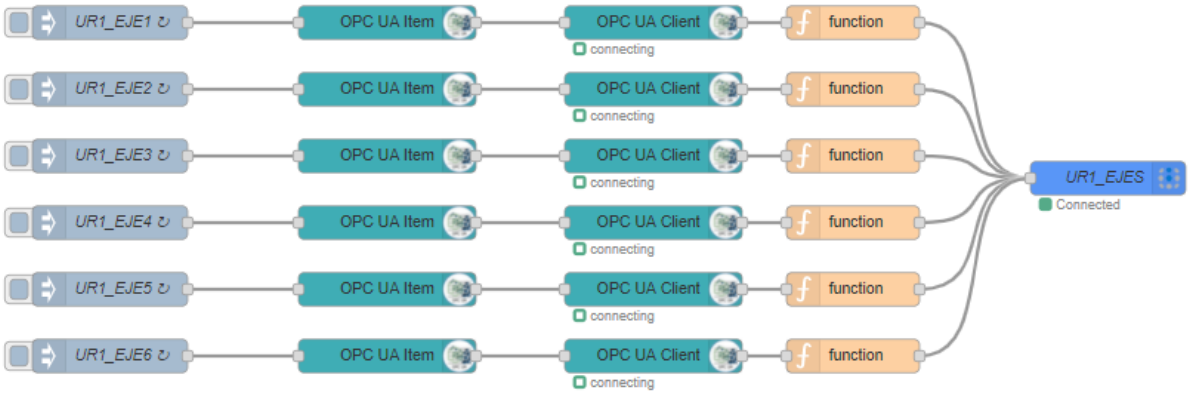
Las configuraciones que varían para poder enviar datos tipo *real* son: el nodo *Inject* debe cambiar el *msg.payload* al tipo ***INT*** y el nodo *OPC UA Item*, el tipo de dato debe ser ***INT32*** para recibir datos con signo, (Figura 140).

Figura 140. Configuración de los nodos que se deben modificar en relación con el proceso anterior para enviar datos reales con signo.



La configuración de los 6 nodos necesarios para enviar el valor de los ejes de *UR1* se repite 5 veces más para mostrar en la *Nube* campos numéricos que representen la posición del robot, la (Figura 141) muestra el código gráfico que materializa el objetivo mencionado.

Figura 141.Programación gráfica para mostrar en el Dashboard de UBIDOTS la posición de UR.

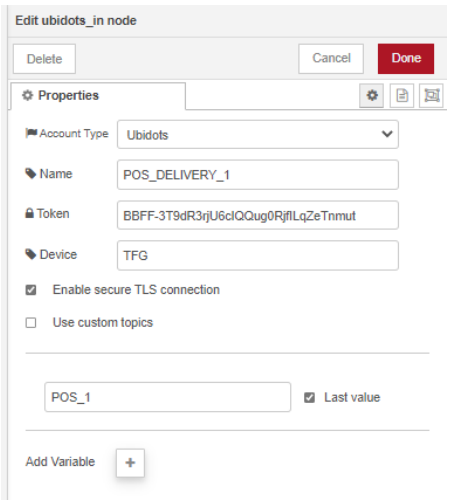


El último grupo de datos que se comentará corresponde a las variables que *UBIDOTS* permite modificar en el *Dashboard* y que se conectan al *PLC* para indicar en qué posición *UR1* debe entregar la pieza (dichas variables son de tipo *BOOL)*. De forma semejante se describe el procedimiento para una posición y el lector puede desarrollar tantas modificaciones como necesite. (Figura 142).

Figura 142. Código gráfico para el envío de la posición de entrega 1 activa a UR1.

1. ***POS_DELIVERY_1***: Constituye un nodo de extracción de datos (*UBIDOTS_IN*) de la *Nube* para enviarlo a *CODESYS* y de esta forma activar la *POS_1* de entrega de *UR1*. Los parámetros necesarios de configurar se muestran en la (Figura 143), *Token* está referido a la cuenta específica de *UBIDOTS*, *Device* al dispositivo configurado en la plataforma y *POS_1* es la variable extraída del *Dashboard* que se crearé en el siguiente apartado.

Figura 143. Configuración del Nodo UBIBOTS_IN.

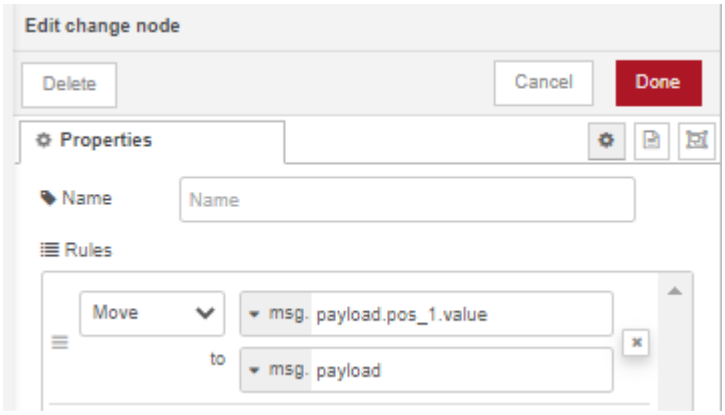


UBIDOTS_IN (help): Se suscribe a un máximo de 10 variables o de *UBIDOTS* y los envía como mensaje a otros nodos. Utiliza la biblioteca *MQTT* para establecer la conexión y suscribirse a la variable o tema. Descripción de los parámetros:

- Tipo de cuenta: Por defecto es "*UBIDOTS*", que es válida tanto para cuentas industriales como para cuentas *STEM* (*Cuentas Educativas*). Si tiene una cuenta de *UBIDOTS for Education*, elija "*UBIDOTS for Education*".
- Nombre: Etiqueta del nodo en el espacio de trabajo de *Node-Red*. Si está vacía, el valor predeterminado es "*UBIDOTS en*".
- Token (Obligatorio): Token necesario para autenticar la conexión con su cuenta de *UBIDOTS*. Para obtener su token, inicie sesión en *UBIDOTS.com*, en "*Mi perfil*" haga clic en "*Credenciales de la API*".
- Etiqueta del dispositivo: La etiqueta del dispositivo al que se suscribe el nodo. Este campo se descuida en caso de que se utilicen Temas personalizados.
- SSL: Por defecto, todos los datos se envían encriptados a través de *TLS*. Desmarque si los datos deben enviarse sin encriptar.

2. ***move msg.payload.pos_1.value:*** En un nodo del tipo antes descrito *CHANGE* en el cual se extrae únicamente del objeto *JSON* el campo *Value* que corresponde con el estado de la variable *BOOL POS_1* de esta forma el componente a su salida solo devuelve un ***0*** ó ***1*** si la variable se encuentra ***False*** o ***True*** respectivamente, de esta forma *CODESYS* vincula directamente este valor con la variable *POS_1* de la tabla de variables globales *TFG.* El nodo directamente toma el campo *Value* del *payload* del dato entrante y lo entrega a la salida (Figura 144).

*Figura 144. Configuración del nodo **CHANGE** para extraer el valor de la variable POS_1 de UBIDOTS.*



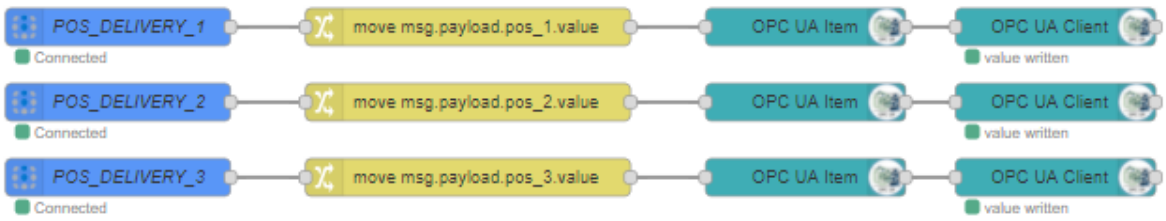
3. ***OPC UA Item:*** El bloque número 3 es similar al descrito previamente, se debe configurar el valor del *Item*:

ns=4;s=|var|CODESYS Control Win V3 x64.Application.TFG.UBIDOTS_POS_1

4. ***OPC UA Client:*** El campo *Endpoint* es igual al configurado en la (Figura 134, página 119), aunque la *Action* que realiza el nodo en este caso es de escritura y debe parametrizarse como ***WRITE***.

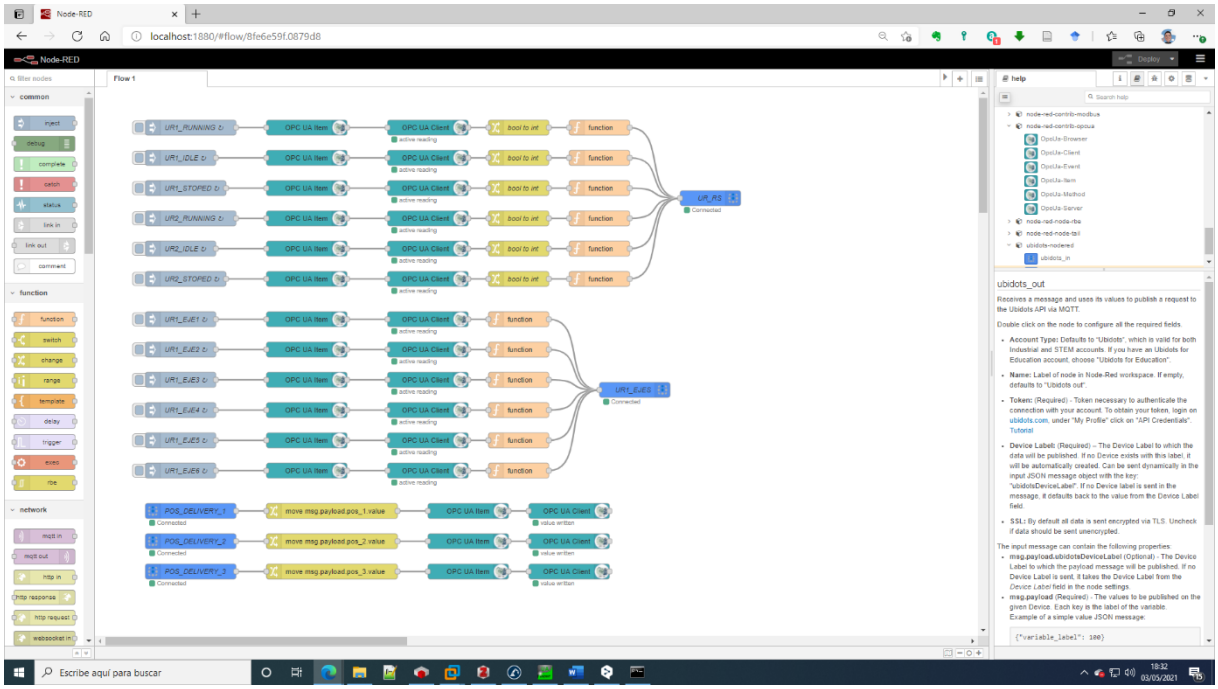
El conjunto de nodos necesarios para extraer la posición en la que debe entregar la pieza el robot *UR* se representan en la (Figura 145):

Figura 145. Nodos necesarios para conocer qué posición de entrega se encuentra activa para que UR1 y UR2 intercambien la pieza.



La (Figura 146) presenta todos los nodos programados y que se encuentran en activo transfiriendo los datos programados entre *CODESYS* y *UBIDOTS* en tiempo real:

Figura 146. Conjunto de nodos para el intercambio de señales entre CODESYS y UBIDOTS.



Node-Red facilita *Importar/Exportar* nodos al espacio de trabajo empleando el formato *JSON*, por ejemplo: un nodo *Inject* en un *Flow* es equivalente a:

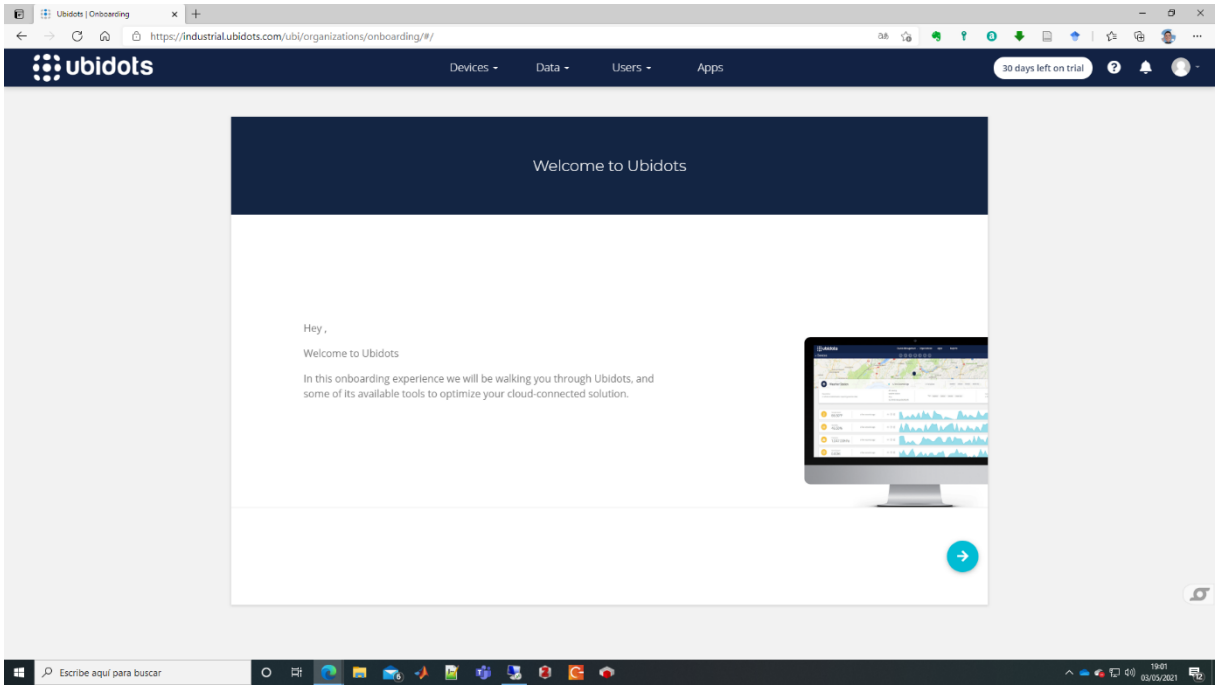
[{"id":"5d96b5bd.eba46c","type":"inject","z":"53f5c755.c3d0f8","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"date","x":230,"y":120,"wires":[[]]}]

Por lo que se empleará esta propiedad para incluir un *ANEXO* (*Anexo III*) con el código *JSON* de los elementos incluidos en *Node-Red* configurados para si el lector desea copiar-pegar y ahorrar tiempo de implementación en su propio proyecto, tan solo debería copiarlo e ir a (≡ → ***Import***) pegar el código.

5.8. UBIDOTS

En la descripción de los proveedores de servicio de *cloud computing* se comentó la posibilidad de crear una cuenta en la plataforma con dos variantes o modalidades, una cuenta *For Educational or Personal Use* y otra *For Business*, en un inicio se escoge la primera opción, tras implementar varias variables y recibir unos 4000 *dots*[29] la plataforma no permite continuar utilizando el servicio hasta el día siguiente, a pesar de no desarrollar un *Dashboard* complejo, en menos de 10 minutos aproximadamente, se alcanzaba esta cuota si los robots estaban moviéndose y se modificaba el valor de los ejes de *UR1*, es por ello por lo que se crea una nueva cuenta versión *For Business* que incluye un período de prueba de 1 mes (Figura 147), tal y como se específica en los objetivos del proyecto, esta nueva cuenta será la que se emplea para describir el proceso que cumple el objetivo: ***Programar un Dashboard en una plataforma de cloud computing para monitorizar parámetros de la celda y configurar en qué posición el primer robot debe entregar la pieza.***

Figura 147. Pantalla de bienvenida a la plataforma tras crear una nueva cuenta tipo ***For Business****.*



En esta primera pantalla, la plataforma incluye un pequeño tutorial que permite al usuario familiarizarse de forma rápida y simple con el concepto de trabajo, en este entrenamiento existe un dispositivo denominado *Machine_A* que contiene 3 variables relacionadas con la *Temperatura, Humedad y la Presión,* en la primera pestaña el usuario encuentra un control deslizante que permite modificar el valor de la temperatura y se presenta en automáticamente en un gráfico de puntos, a continuación *UBIDOTS* permite generar 4 tipos de eventos diferentes (*SMS, e-mail, Telegram o Webhook*), para llevar a cabo una acción si la temperatura excede o no sobrepasa un determinado

[29] ***Dots***: Tasas de ingestión de datos y capacidad mensual de los usuarios de Ubidots del inglés Data Ingestion Rates and Monthly Capacity for Ubidots STEM.

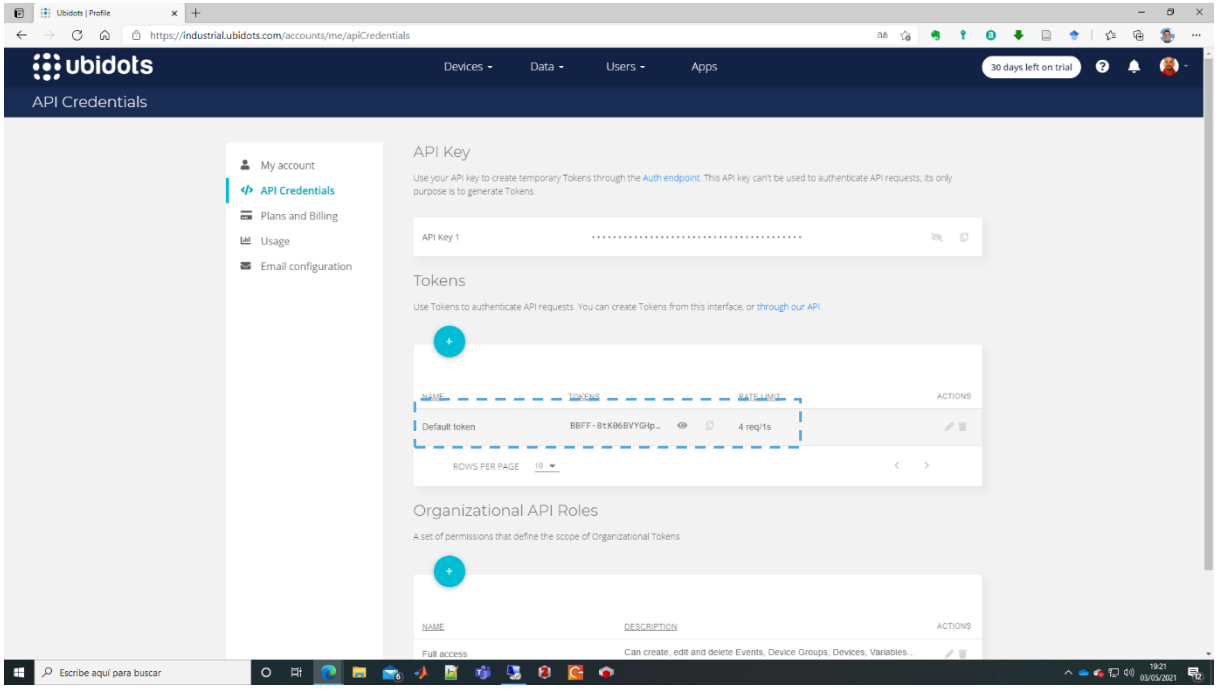
valor, por último el usuario accede a un *Dashboard* con dos componentes que muestra el comportamiento de la temperatura anterior.

5.8.1. API CREDENTIALS

En la descripción de la configuración de los nodos de *Node-Red* y específicamente los relacionados con el intercambio de datos con *UBIDOTS*, se pudo comprobar como algunos parámetros quedaron pendientes de comentar o explicar de dónde se obtenían, en el punto en que se encuentra el proyecto es posible dar respuestas a dichas incógnitas.

Dentro de *UBIDOTS → MyProfile → API Credentials* (Figura 148) se muestran algunos parámetros que se emplearán para comunicar la plataforma con el resto de elementos descritos.

*Figura 148. Pestaña **API Credentials** donde está definido el **Token** para comunicarse con **Node-Red**.*



En el apartado *Tokens*[30] por defecto *UBIDOTS* crea uno denominado *Default Token* con un valor con el siguiente formato:

BBFF-3T9dR3rjU6cIQQug0RjfILqZeTnmut

Este campo debe ser copiado por el usuario y pegado en cada nodo que incluya el parámetro *Token en* su configuración, en el caso comentado en *Node-Red*, estos nodos son *UBIDOTS_In* y *UBIDOTS_Out.*

[30] *Token: Es una clave única que autoriza a su dispositivo a interactuar con la API de Ubidots, está vinculada a un único dispositivo dentro de la base de datos de Ubidots, con uno o ambos de los siguientes permisos:*

- *Enviar datos: Publicar en, o hacer peticiones POST para enviar datos al dispositivo.*
- *Recuperar datos: Suscribirse a, o hacer peticiones GET para recuperar datos del dispositivo.*

Los tokens de dispositivo no pueden utilizarse para crear, editar o eliminar dispositivos o variables. Para realizar estas operaciones, utilice un token de organización o un token de cuenta. Véase Security: Managing Device Tokens | Ubidots Help Center

5.8.2. DEVICE - VARIABLES

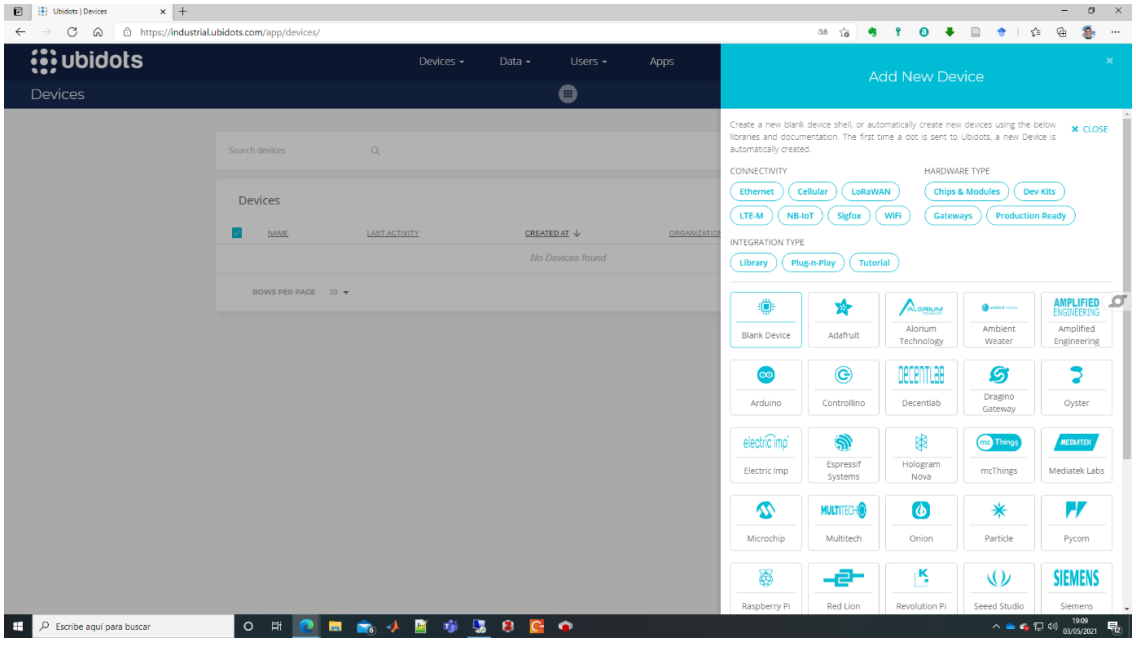
En *Node-Red* se comentó el campo *Device* y *Device Label* relativos a los nodos *UBIDOTS_In* y *UBIDOTS_Out* respectivamente, "*un dispositivo en UBIDOTS es una representación virtual de una fuente de datos o simplemente, un activo que toma datos de sensores y transmite dichos datos a través de un protocolo de conexión a la nube de UBIDOTS. Haga clic aquí para ver ejemplos de firmware y tutoriales actuales para conectar su dispositivo a UBIDOTS*.

Todos los dispositivos son diferentes, pero la configuración estándar para cualquier dispositivo conlleva las mismas características a la hora de conectarse a UBIDOTS:

- *Una biblioteca que debe ser instalada en el IDE del dispositivo. (no es necesario)*
- *Un Webhook o función que debe ser utilizado para comunicarse con las nubes o plataformas de hardware de terceros.*
- *Rellenar los parámetros para la autenticación y la conexión, como un TOKEN (es decir, el ID único para cada cuenta o usuario), la etiqueta del dispositivo, la etiqueta de la variable (es decir, los identificadores o nombres de los dispositivos y las variables en UBIDOTS), el SSID de Wi-Fi o la contraseña, dependiendo del dispositivo y los requisitos.*
- *Realización de la solicitud de la API (es decir, una llamada de un dispositivo al servidor web). Las bibliotecas de UBIDOTS evitan la necesidad de realizar manualmente estas peticiones. Sin embargo, si se trabaja con un dispositivo que no está actualmente en la lista de hardware soportado, es posible elegir un protocolo de conectividad como MQTT o HTTP y hacer una petición usando la API en consecuencia."(Creating Devices in UBIDOTS, s. f.)*

La plataforma ofrece un amplio listado de librerías preconfiguradas para establecer la comunicación entre la *Nube* y los dispositivos de forma más inmediata y sencilla, sin embargo, como no existe un *Device* para *CODESYS*, se muestra el procedimiento para crear un dispositivo vacío (no asociado con una marca específica), en el menú de navegación de *UBIDOTS* presionar *Device* → *Add New Device* (Figura 149), posteriormente se selecciona la primera opción *Blank Device* y se asigna un nombre y etiqueta.

Figura 149. Creación de un nuevo dispositivo vacío en Ubidots.



Se nombra al dispositivo creado como *TFG*, en los nodos *UBIDOTS_In* y *UBIDOTS_Out* existen dos campos relacionados al dispositivo desde/hacia el cual se desea extraer/enviar la información respectivamente, dicho parámetro toma la misma denominación, *Device*, en todos los casos se debe configurar con el mismo dispositivo "*TFG".*

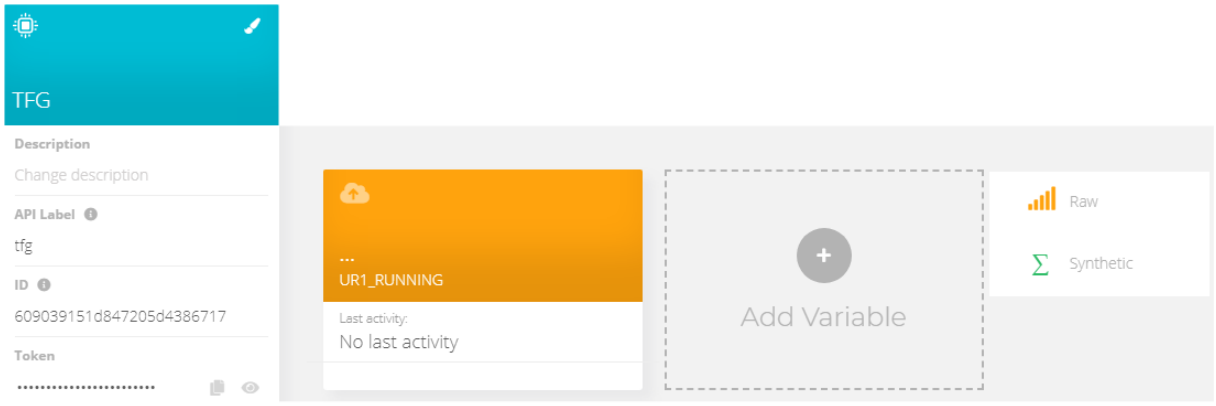
Una vez configurado el dispositivo con el nombre y la etiqueta (generalmente coinciden) es momento de declarar las variables que se desean recibir o enviar, como se comentó previamente no es objetivo del proyecto crear un *Dashboard* que incluya todas las variables representadas en el *WebServer*, en su lugar el autor escoge las siguientes variables para solventar el objetivo concerniente a este apartado (Tabla 12):

Tabla 12. Variables intercambiadas entre CODESYS y UBIDOTS.

Nodo	Tipo	Variable
UBIDOTS_Out	BOOL	POS_DELIVERY_1
	BOOL	POS_DELIVERY_2
	BOOL	POS_DELIVERY_3
UBIDOTS_In	BOOL	UR1_RUNNING
	BOOL	UR1_IDLE
	BOOL	UR1_STOPPED
	BOOL	UR2_RUNNING
	BOOL	UR2_IDLE
	BOOL	UR2_STOPPED
	Real	UR1_EJE_1
	Real	UR1_EJE_2
	Real	UR1_EJE_3
	Real	UR1_EJE_4
	Real	UR1_EJE_5
	Real	UR1_EJE_6

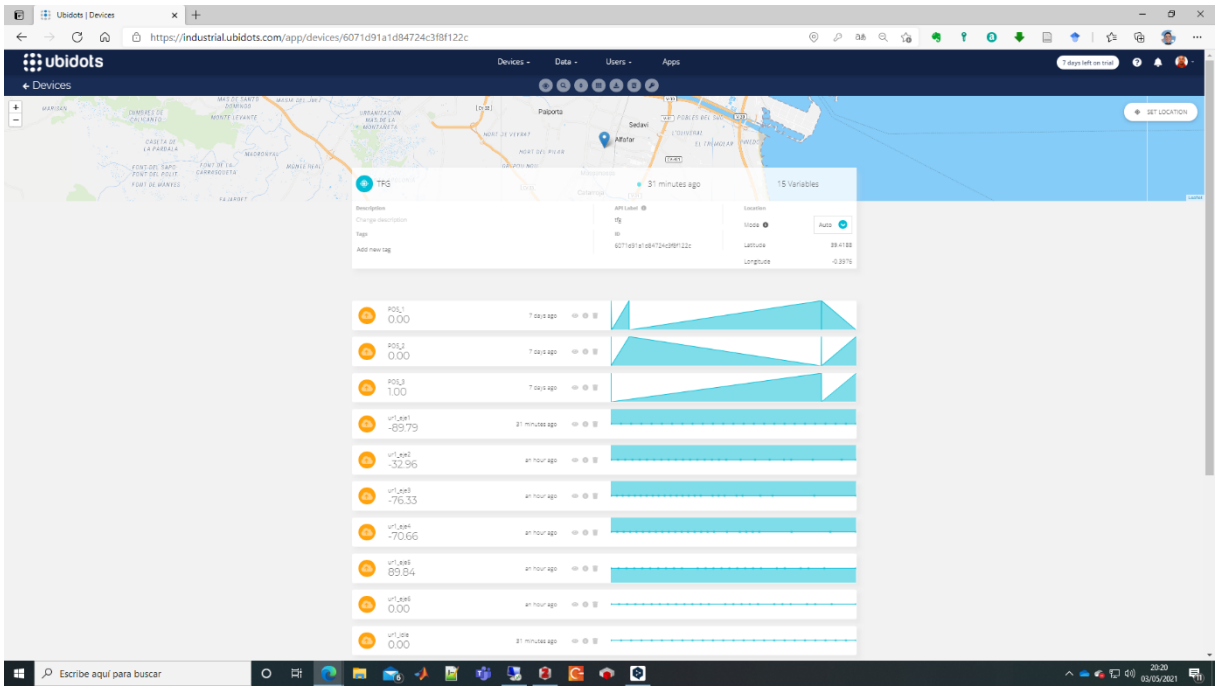
Para crear una variable en *UBIDOTS*, estando dentro de *Device* y presionando *Add Variable* (Figura 150), se debe escoger tipo *Raw* y de esta forma, renombrando cada una con la denominación planteada en la (Tabla 12), se tienen todas las variables necesarias para crear el *Dashboard.*

Figura 150. Proceso de creación de una variable en Ubidots.



La (Figura 151) muestra el total de variables configuradas en el proyecto así como un gráfico asociado a cada una donde se puede visualizar el comportamiento de cada valor asociado en el tiempo, el bloque superior del navegador *UBIDOTS* muestra un mapa con las coordenadas donde se encuentra el dispositivo creado y las variables asociadas a dicho elemento, en caso de tener programados varios *Devices* en una cuenta, el mapa dispondrá de un icono geolocalizado vinculado al dispositivo, esta propiedad permitirá crear eventos desencadenados al movimiento o la entrada/salida de un módulo en una determinada zona geográfica.

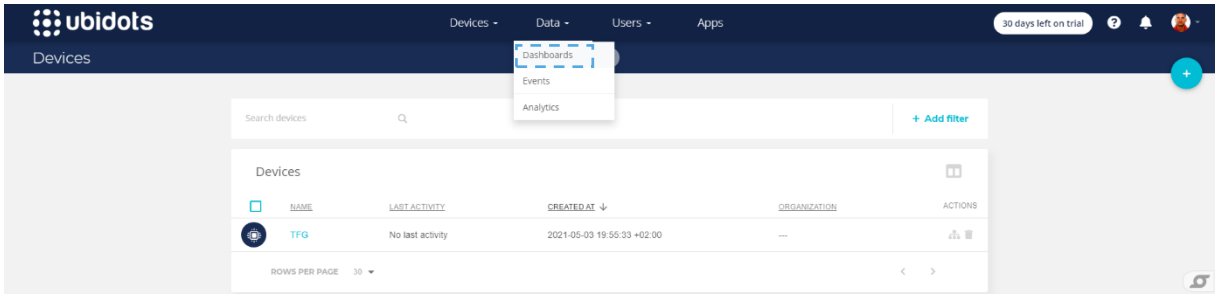
Figura 151. Variables configuradas en el proyecto, así como la representación geográfica de las coordenadas del Device.



5.8.3. DASHBOARD

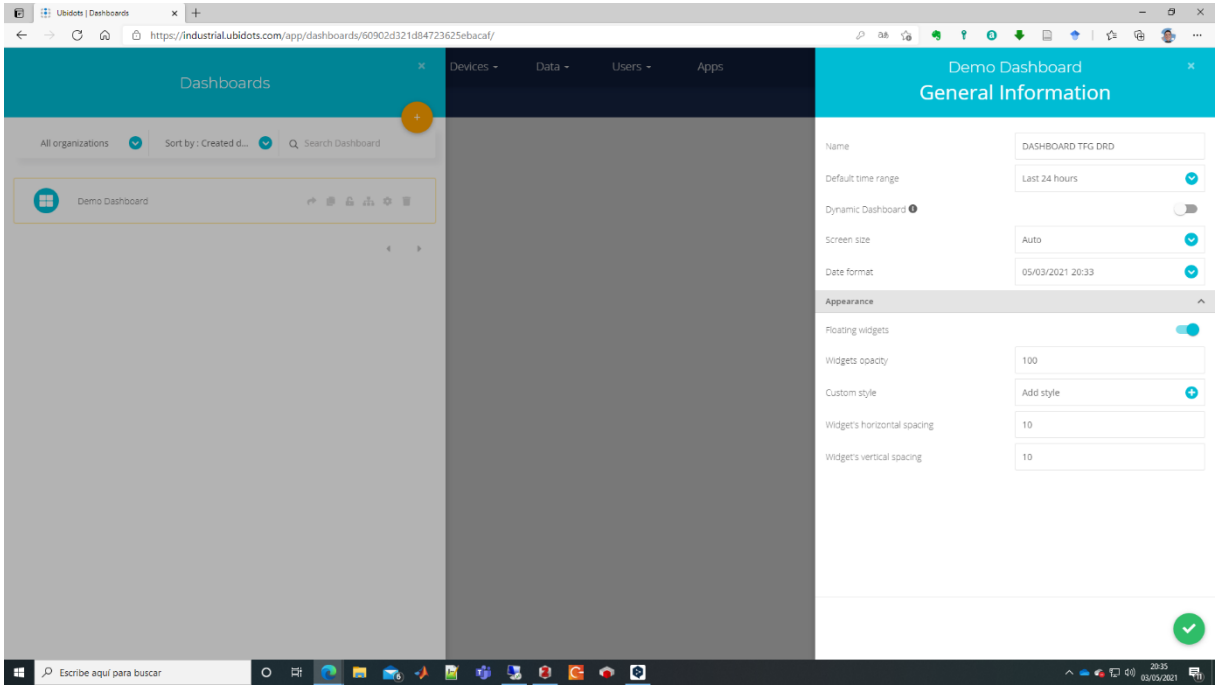
Las variables configuradas permiten su interconexión con elementos de representación gráficos en un *Dashboard* (*Cuadro de Mando*). Para crear uno nuevo, se presiona *Data* en el menú del navegador y se selecciona *Dashboard.* (Figura 152).

Figura 152. Crear un Dashboard en Ubidots.



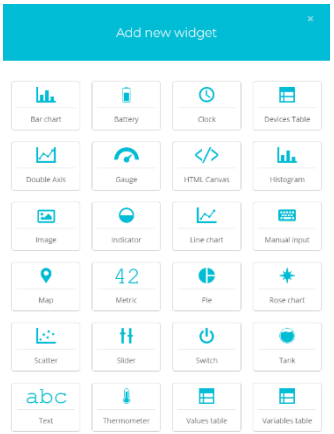
En la nueva ventana es posible modificar algunos parámetros del cuadro de mando como son el nombre, en el caso que se muestra se asigna *DASHBOARD TFG DRD*, y se permite que los *Widgets*[31] se puedan mover dentro del panel. (Figura 153).

Figura 153. Parametrización del Dashboard.



UBIDOTS brinda una amplia variedad de *Widgets* disponible para representar los datos en un tablero de mando (Figura 154), el autor ha seleccionado fundamentalmente tres (*Metric, Indicator y Switch*) para representar el valor de los ejes de *UR1*, el estado de cada robot y seleccionar que posición estará activa para que se realice el intercambio de la pieza, respectivamente.

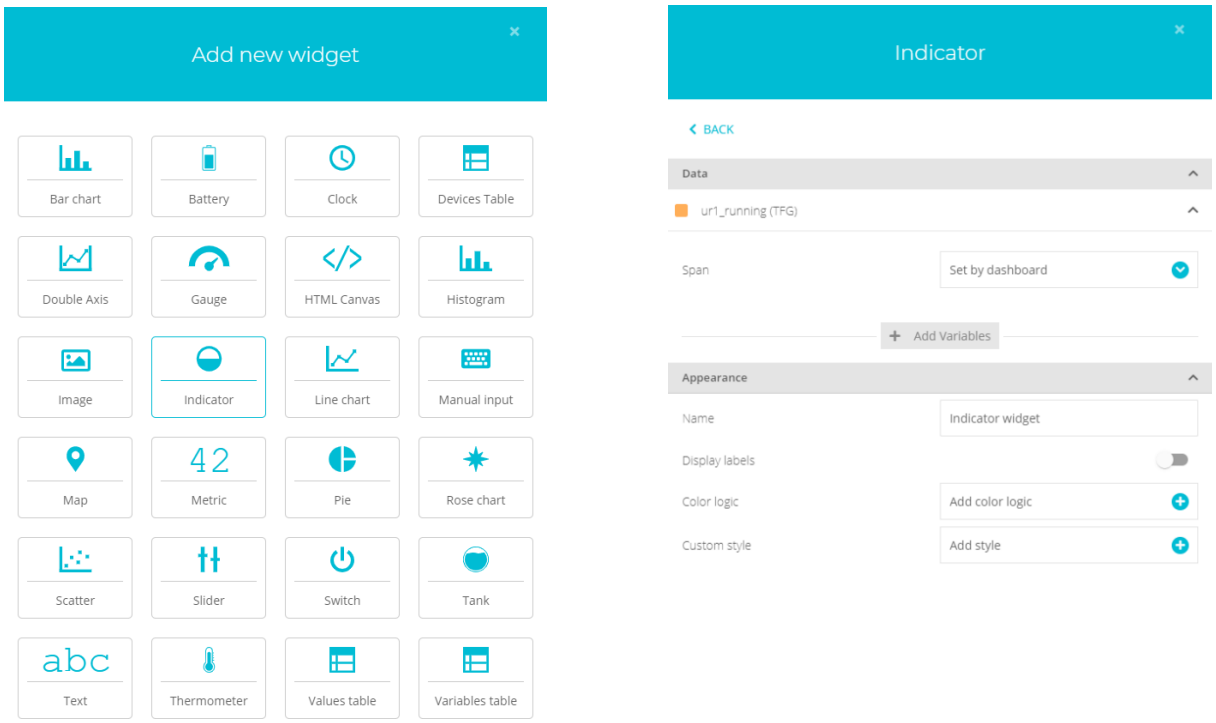
Figura 154. Distintos Widgets disponibles en Ubidots para visualizar datos.



[31] Es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets* o *Widget Engine*. Entre sus objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual. Aunque no es condición indispensable, los widgets suelen ser utilizados para ser "empotrados" en otra página web, copiando el código que el mismo widget pone a disposición del usuario. Dado que son pequeñas aplicaciones, los *widgets* pueden hacer todo lo que la imaginación desee e interactuar con servicios e información distribuida en Internet; pueden ser vistosos relojes en pantalla, notas, calculadoras, calendarios, agendas, juegos, ventanas con información del tiempo en su ciudad, incluso sistemas de tiendas de comercio, etcétera.

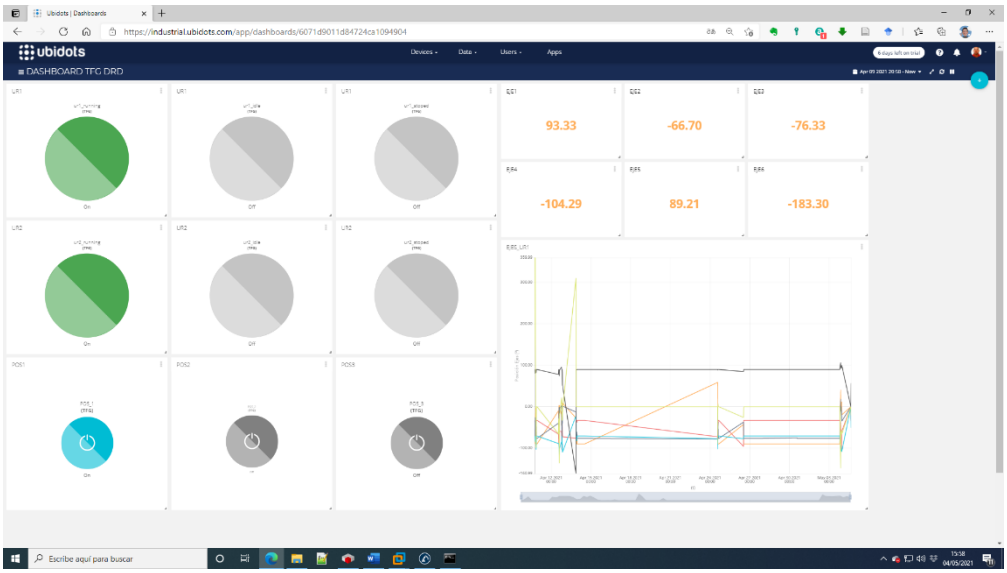
Configurar un *Widgets* es una tarea que se desarrolla de forma inmediata, se requieren escasos 4 pasos que comienzan con el proceso de agregar un nuevo elemento al *Dashboard*, se muestra en la (Figura 155.a), este procedimiento permite representar el estado de una variable, específicamente *UR1_RUNNING* (Figura 155.b).

*Figura 155. Creación de un Widget del tipo **Indicator** par la variable **UR1_RUNNING**.*



El mismo recurso se emplea para insertar los seis *Indicators* para los estados de *UR1* y *UR2*, otros seis *Widgets Metrics* para el valor de cada eje de *UR_IZQ* y tres *Switchs* para activar las posiciones, la (Figura 156) muestra el conjuntos de elementos que conforman el *Dashboard*, el autor ha incluido un gráfico de puntos para juntar en un mismo componente todas las variables relativas a los ejes de *UR1.*

Figura 156. Dashboard confeccionado con las 15 variables configuradas en la Tabla 12.



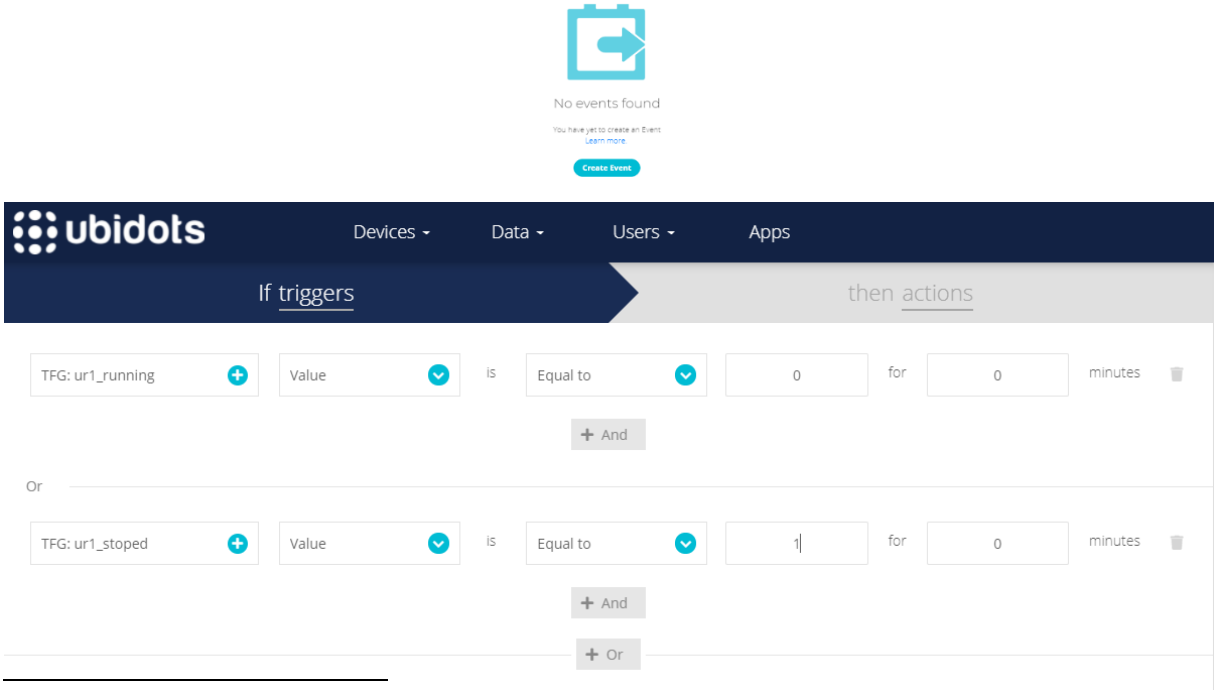
5.8.4. EVENTOS

UBIDOTS soporta eventos integrados que permiten enviar alertas y notificaciones a quienes necesitan tener conocimiento de un determinado incidente, las integraciones preconstruidas de *UBIDOTS* incluyen:

- Notificaciones por correo electrónico
- Notificaciones por *SMS*
- Eventos *Webhook*[32]
- Notificaciones de *Telegram*
- Notificaciones de *Slack*[33]
- Notificaciones de llamadas de voz
- Notificación de vuelta a la normalidad
- Notificaciones de *Geofence*

Para acceder al espacio de configuración de *Eventos* en *UBIDOTS*, en el menú de navegación presionar *Data* → *Events*, posteriormente en la ventana donde se indica que no existe ningún *Evento*, presionar *Crear Evento* (Figura 157.a), una nueva ventana de configuración se muestra donde se debe escoger que variable desencadenará el evento, indicar que parámetro de la variable se tiene en cuenta (*Valor*) y establecer la condición a evaluar (Figura 157.b), al terminar se debe avanzar a la ventana de configuración del desencadenador.

*Figura 157. a. Ventana para crear un Evento. b. Ventana para configurar un **Evento** en Ubidots.*

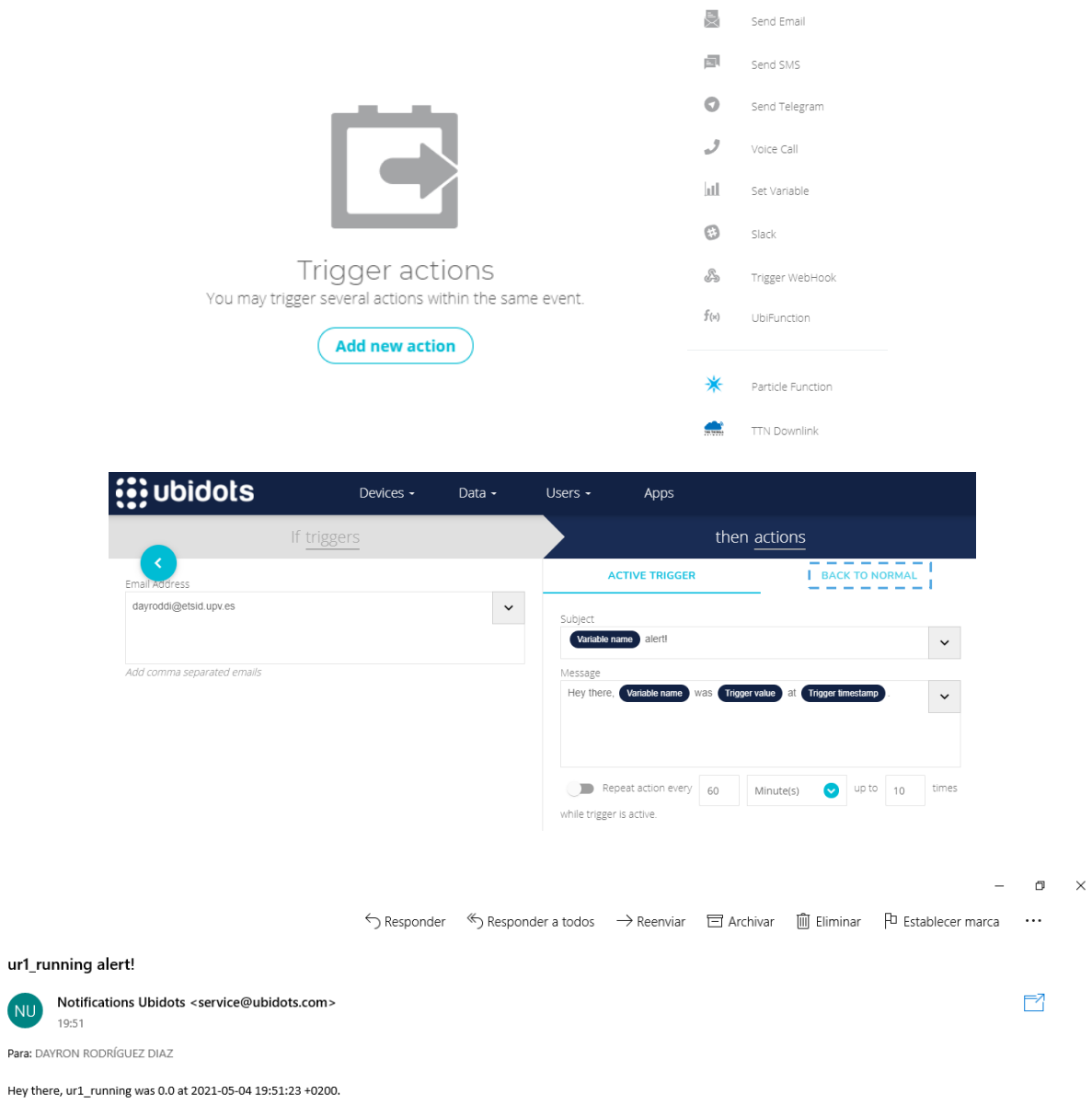


[32] Método de alteración del funcionamiento de una página o aplicación web, con callbacks personalizados. Estos se pueden mantener, modificar y gestionar por terceros; desarrolladores que no tienen por qué estar afiliados a la web o aplicación. Los webhooks son retrollamadas HTTP de usuario. Estos se registran en algunas ocasiones, como al publicar un comentario en un blog. Cuando esto ocurre, la web envía una solicitud HTTP a la URL configurada para el webhook. Los usuarios pueden configurarlos para que la web se comporte de una forma u otra. Como usan HTTP, pueden integrarse en servicios web sin añadir una nueva infraestructura. Véase Webhook - Wikipedia, la enciclopedia libre

[33] Slack es una plataforma de mensajes basada en canales. Gracias a Slack, las personas pueden trabajar juntas de forma más eficaz, conectar todos sus servicios y herramientas de software, y encontrar la información que necesitan para dar lo mejor de sí mismas. Véase Centro de Ayuda de Slack | Slack

En un inicio no existen acciones asociadas, tal como ocurrió con los eventos (Figura 158.a), por ello se debe presionar *Crear acción (Trigger actions)*, en este apartado se comenta como crear un *Evento* que envíe un *e-mail* a la dirección de correo de la Universidad cuando *UR1* se encuentre apagado, seleccionando *Send Email* en el panel derecho de la (Figura 158.a), la ventana del navegador se actualiza (Figura 158.b) para mostrar los parámetros necesarios, la dirección de correo se inserta en el panel izquierdo y se pueden agregar tantas como se deseen separadas por (“,”); a la derecha se aprecia el cuerpo del mensaje que enviará *UBIDOTS* cuando la condición antes programada se cumpla, se puede especificar si se desea repetir esta acción fijando un intervalo de tiempo. Cuando la condición programada se cumple, en el correo se recibe un *e-mail* como el mostrado en la (Figura 158.c), en el apartado *BACK TO NORMAL*, (Figura 158.b), es posible generar otro mensaje para que *UBIDOTS* lo envíe una vez la variable ha recuperado su estado previo y abandone la condición que disparo el *Evento*.

Figura 158. a. Agregar nueva acción tras generarse un evento. b. Configuración de los parámetros para enviar el e-mail. c. Correo recibido tras apagar ***UR1.***



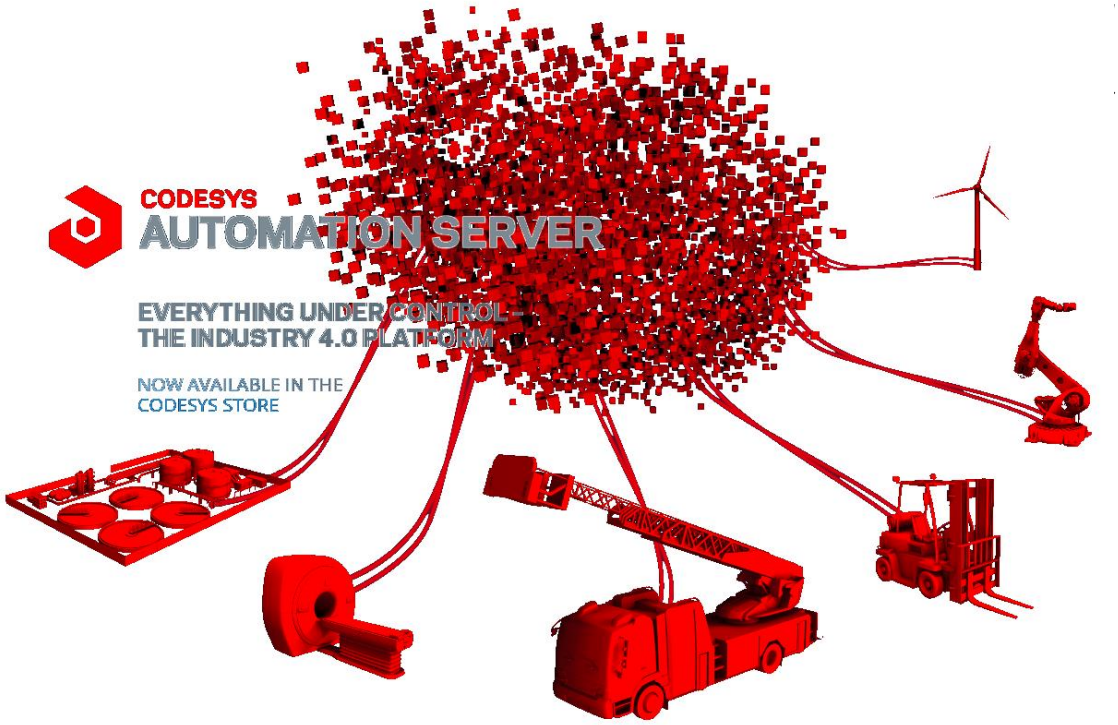
5.9. CODESYS AUTOMATION SERVER

"*CODESYS Automation Server* (Figura 159) *es la plataforma Industry 4.0 para más de 100,000 usuarios de CODESYS en todo el mundo. El servidor les ayuda a administrar sus controladores y aplicaciones relacionadas y a implementar actualizaciones. Al mismo tiempo, la plataforma ofrece una infraestructura segura para acceso remoto, depuración o visualización web.*

CODESYS Automation Server ahora ofrece los llamados proyectos Data Analyzer, los usuarios ahora pueden monitorear, operar y analizar sus controles de máquinas y plantas de forma remota desde cualquier lugar del mundo, gracias a la tecnología en la nube.

Los enfoques existentes generalmente se centran en la ruta de transmisión: las palabras de moda son MQTT u OPC UA. El nuevo paquete de funciones, por el contrario, se concentra en las tareas principales: procesamiento conveniente, visualización, análisis y modificación de datos. Los usuarios utilizan automáticamente la comunicación segura del servidor de automatización CODESYS como una plataforma independiente del fabricante." (García, s. f.).

*Figura 159. Plataforma **CODESYS Automation Server**.*



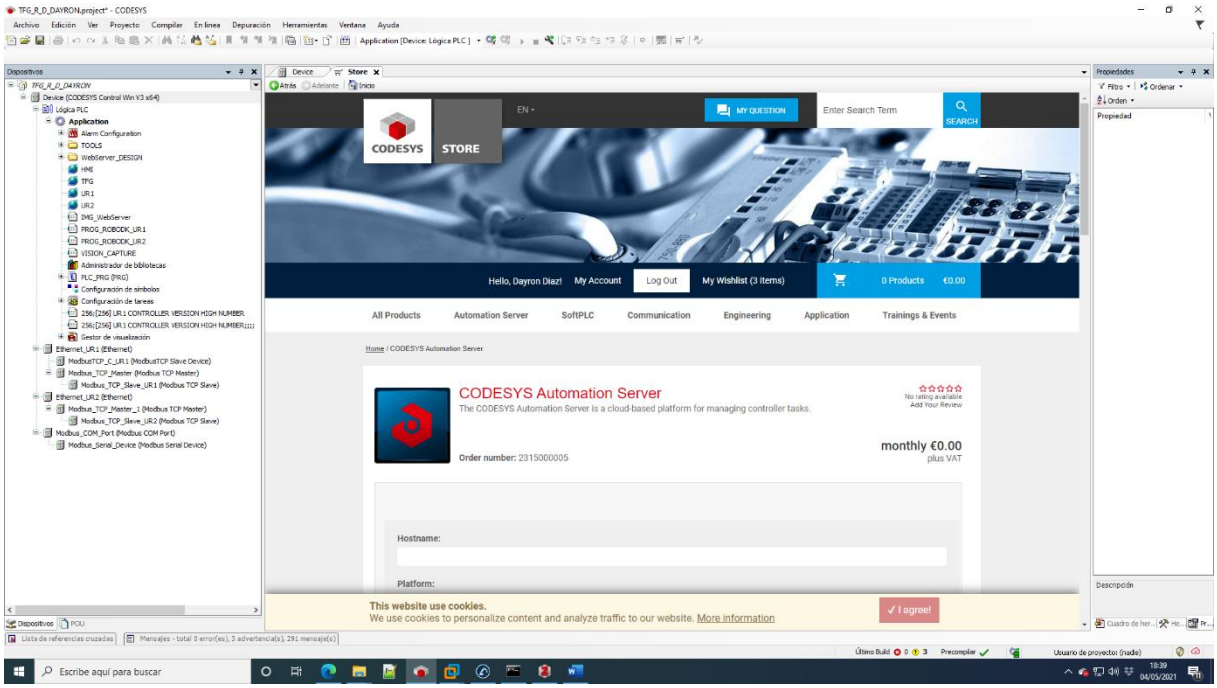
Fuente: Home - CODESYS Automation Server (automation-server.com)

*"La interface web **permite la gestión y monitorización de todas las estaciones**, ya sea en modo local o remoto, así como ejecutar las actualizaciones de los programas. La conexión puede ser directa desde el controlador al servidor y, en los casos en los que la seguridad requerida sea mayor, se puede utilizar un Gateway de CODESYS con seguridad TLS integrada."* (automaticaeinstrumentacion.com, s. f.).

5.9.1. INSTALACIÓN

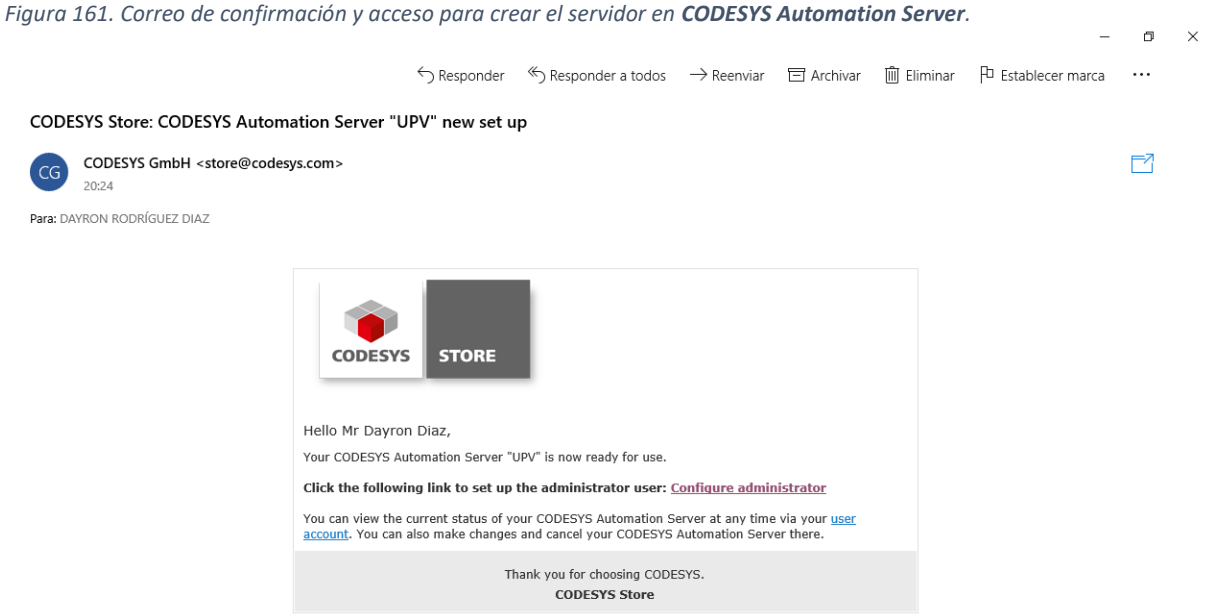
CODESYS Automation Server debe ser descargado desde la *Store* accediendo desde el menú principal del programa *Herramientas → Administrador de Paquetes → Codesys Store* (Figura 160).

Figura 160. Instalación del servicio desde la plataforma CODESYS Store.



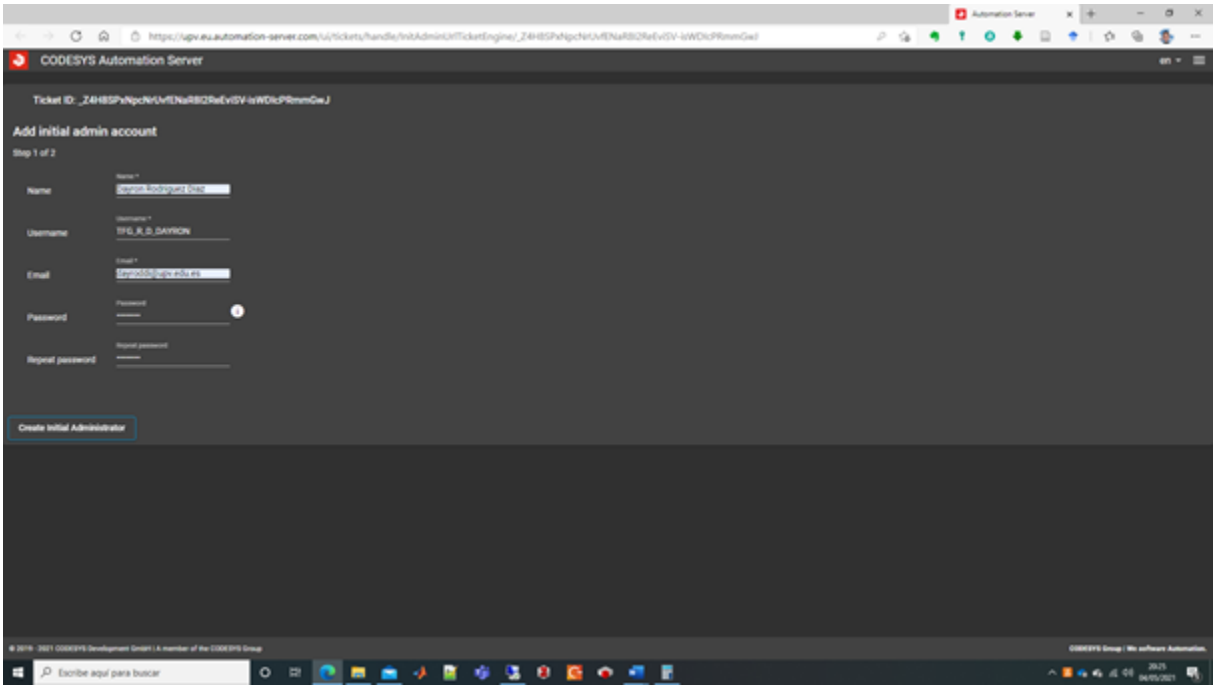
Es un servicio gratuito, aunque apunta *pago con tarjeta*, se debe indicar una dirección *email* que debe coincidir con la misma que se empleó para darse de alta en *CODESYS*, al instalar el paquete se recibe un correo electrónico donde se muestra que el servidor creado está listo para su uso y configuración, para ello se facilita un link con permisos de administrador (Figura 161) a través del cual es posible darse de alta en el servicio con un usuario y una contraseña.

*Figura 161. Correo de confirmación y acceso para crear el servidor en **CODESYS Automation Server**.*



En el enlace se configuran los parámetros de acceso a la plataforma (Figura 162), tras completar todos los campos, se debe presionar el botón *Create Initial Administrator*, la ventana se actualiza y muestra

*Figura 162. Parámetros para crear el **Servidor** en Codesys Automation Server.*

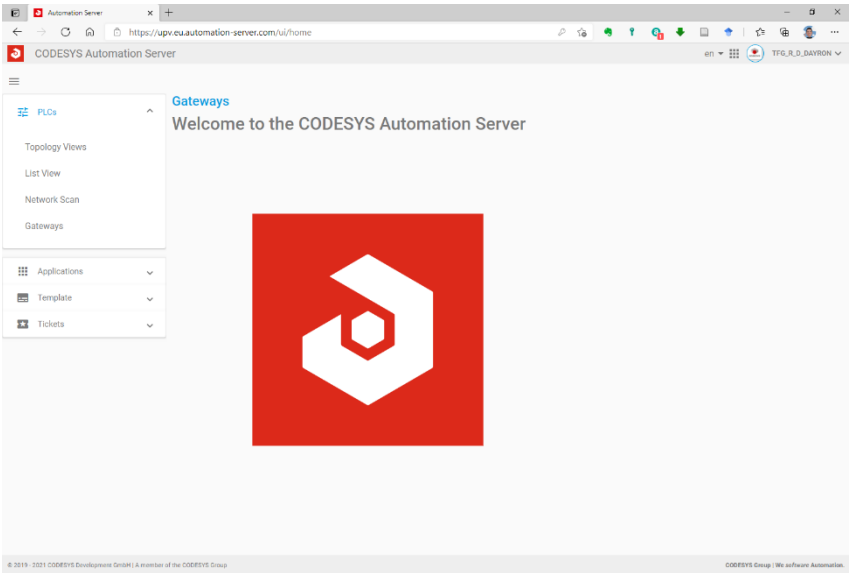


el enlace donde se ha generado el servidor, el sistema emplea el *HOST* que se introdujo al descargar *CODESYS Automation Server* en *CODESYS Store*, en el caso que se describe el autor del proyecto escribió *UPV*, por lo cual el servidor sobre el que se comentará todo el proceso de conexión es:

https://upv.eu.automation-server.com/

Una vez dado de alta, la página solicita al usuario que acceda empleando el *Nick* y la contraseña previos; el diseño de la plataforma es sencillo, en el centro del navegador se muestra la información que es accesible desde varios elementos agrupados en un menú a la derecha. (Figura 163).

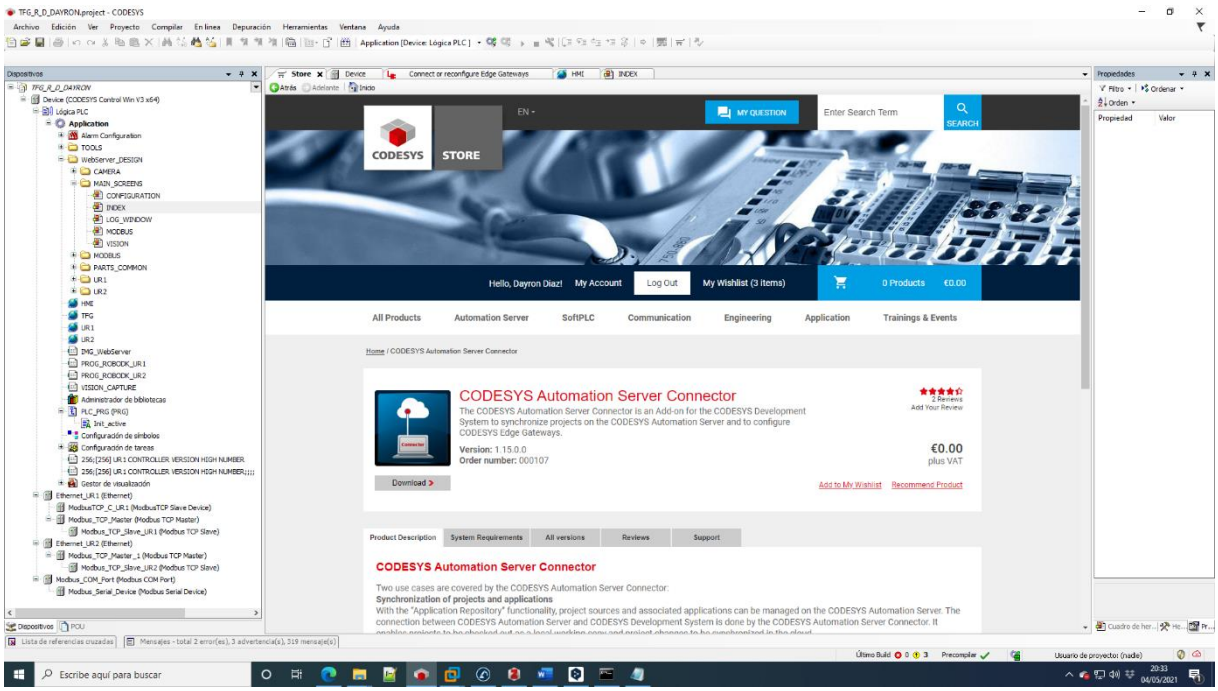
Figura 163. Ventana principal de la plataforma CODESYS Automation Server.

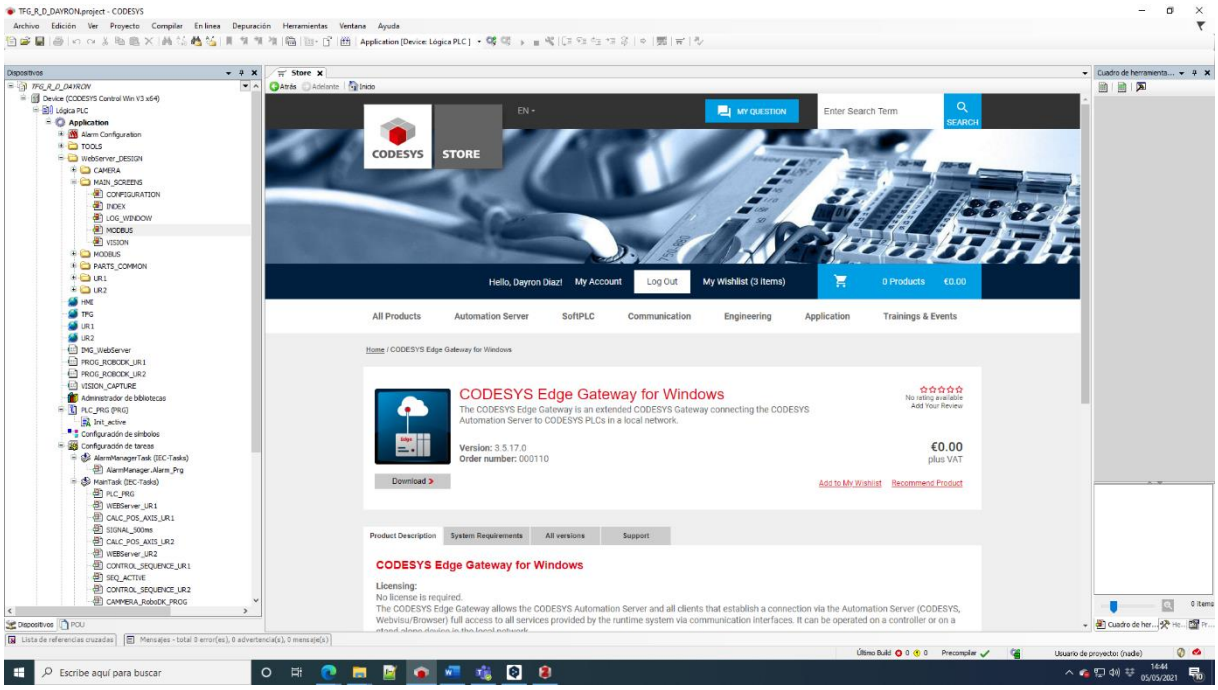


Existen dos componentes extras que deben ser instalados desde *CODESYS Store:*

- *CODESYS Automation Server Conector:* "*Es un complemento para el sistema de desarrollo de CODESYS para sincronizar proyectos en el Servidor de Automatización de CODESYS y para configurar los Gateways de CODESYS Edge.*" (*CODESYS Store International - CODESYS Automation Server Connector*, s. f.). (Figura 164.a).
- *CODESYS Edge Gateway for WINDOWS: "Es un CODESYS Gateway ampliado que conecta el CODESYS Automation Server con los PLCs de CODESYS en una red local."* (*CODESYS Store International - CODESYS Edge Gateway for Windows*, s. f.). (Figura 164.b).

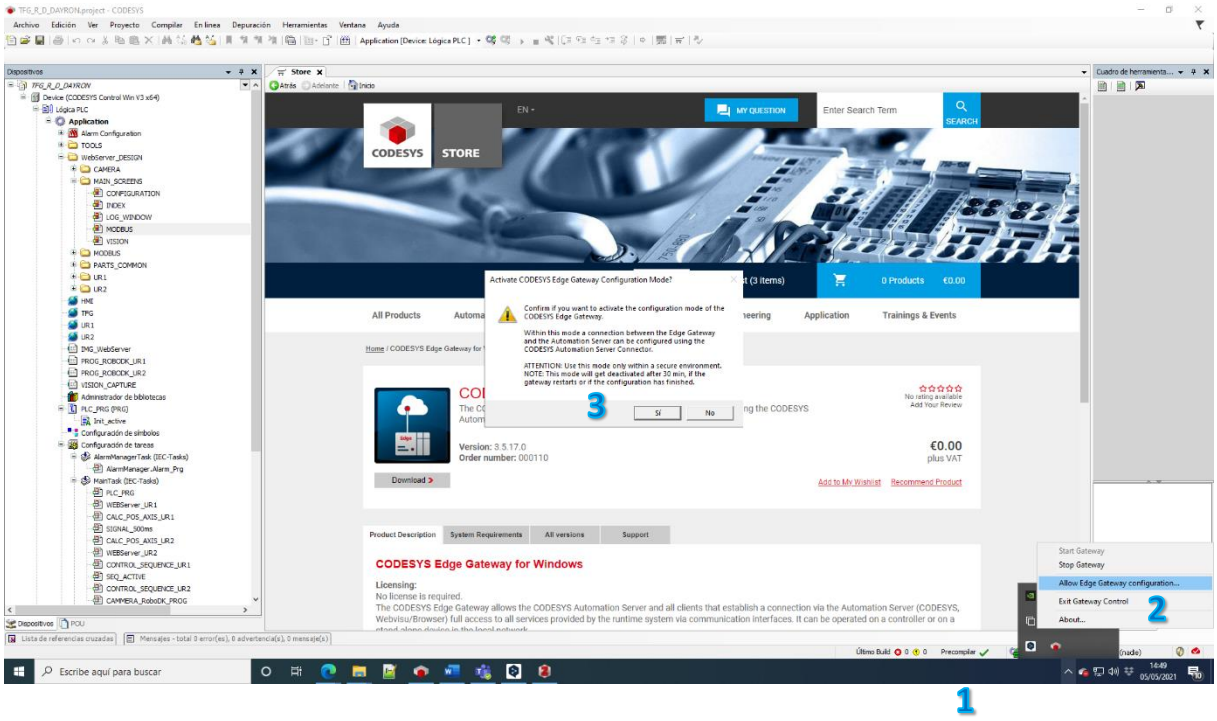
Figura 164. a. Descarga del paquete CODESYS Automation Server Connector. b. Descarga del paquete CODESYS Edge Gateway for Windows.





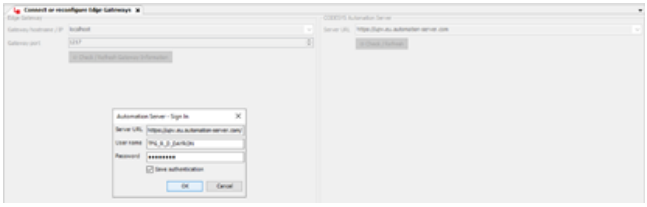
Tras instalar los servicios anteriores, *CODESYS* debe reiniciarse para que cargue los nuevos componentes, posteriormente el usuario debe desplazarse a la barra de tareas de *Windows* (Figura 165.1), en el apartado donde se encuentran los iconoco ocultos, *clic derecho* sobre *CODESYS Gateway SysTray* y seleccionar la opción *Allow Edge Gateway Configuration* (Figura 165.2) este paso establece la conexión entre el *softPLC* y la plataforma en la *Nube.* Un ventana emergente (Figura 165.3) pregunta al usuario si desea activar el modo de configuración del *CODESYS Edge Gateway*. Dentro de este modo se puede configurar una conexión entre el *Edge Gateway* y el *Automation Server* mediante el *CODESYS Automation Server Connector.*

Figura 165. Activación del servicio ***CODESYS Edge Gateway.***



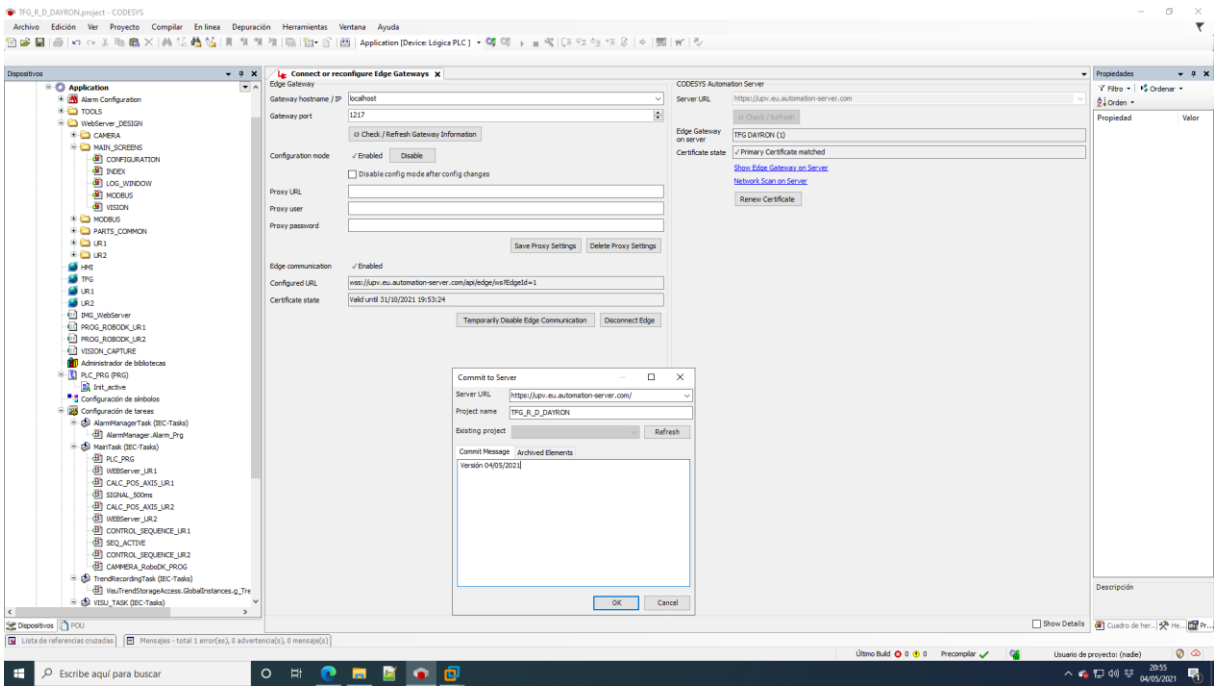
La pasarela está activada, solo resta conectar *CODESYS* con la *Nube*, en la barra de menú principal del programa, opción *Herramientas → Automation Server → Connect Edge Gateway*, se abre una nueva ventana en el programa para configurar la comunicación con el servidor, el primer parámetro que se debe establecer es la dirección *URL* del servidor, comentada antes (https://upv.eu.automation-server.com/), posteriormente se presiona *Check/Refresh* y una ventana solicita los datos de usuario y contraseña para conectar con la plataforma, estos coinciden con los introducidos en la plataforma.

Figura 166. Autenticación para conectar con CODESYS Automation Server.



Al conectar aparece una ventana que apunta al servidor y al proyecto creado (*usuario*), en ella existe una pestaña con la denominación *Current Message* donde se debe escribir algo, normalmente los programadores emplean este recurso para diferenciar las distintas versiones del código que se desean subir a la plataforma, el autor sigue esta práctica y asigna el valor *Versión_(1)_04/05/2021.* (Figura 167).

Figura 167. Estableciendo la primera versión del código para subirla a la nube.



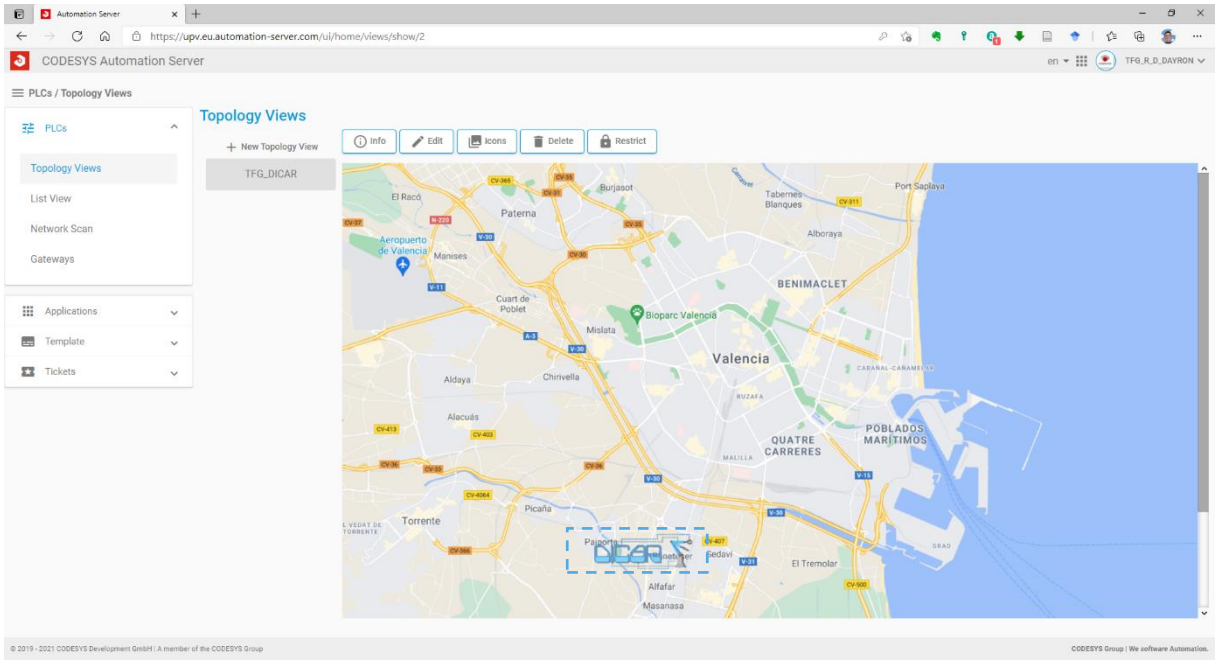
Cuando se presiona *OK* en la ventana anterior *CODESYS* comienza a subir todo el proyecto a la *Nube,* al concluir en la barra de estados del programa aparece un icono con una *nube roja* marcada con una palometa que indica que la conexión se ha establecido satisfactoriamente. (Figura 168).

Figura 168. Barra de estado de CODESYS donde se aprecia si la comunicación con la plataforma ha sido completada.

5.9.2. LA PLATAFORMA

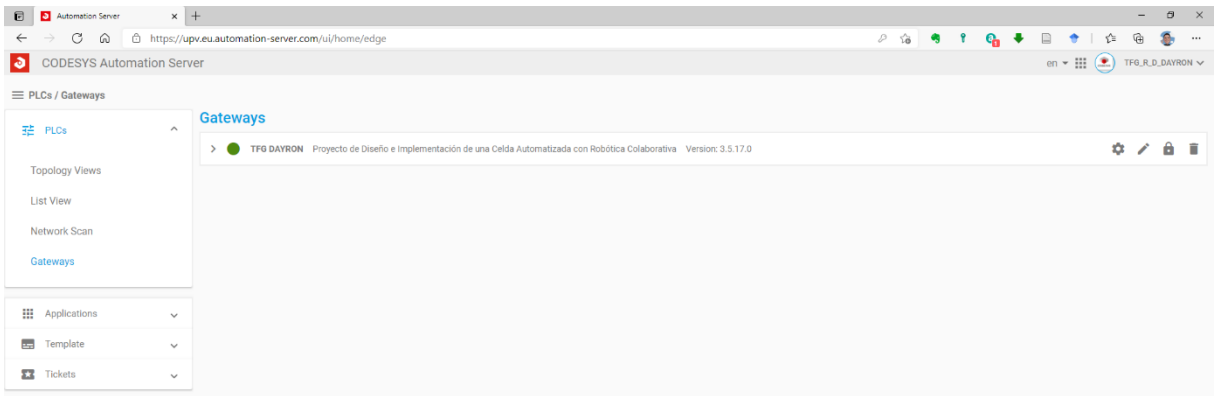
En el presente apartado se describirán las principales herramientas con las que cuenta *CODESYS Automation Server*. En la vista de topología, puede visualizar los *PLCs* introducidos en el *Automation Server* y su topología de red en una pantalla de libre elección. Con el uso de *Google Maps* se capturó un plano de *Valenci*a para subirlo como imagen de fondo de la vista de tipología, posteriormente en el apartado de edición se arrastra el componente *PLC* y se asigna otra imagen a su representación , para ello se emplea el logo del *TFG* y se sitúa en el área urbana de *Benetússer*. (Figura 169).

Figura 169. Configuración de la ventana de tipología de la plataforma.



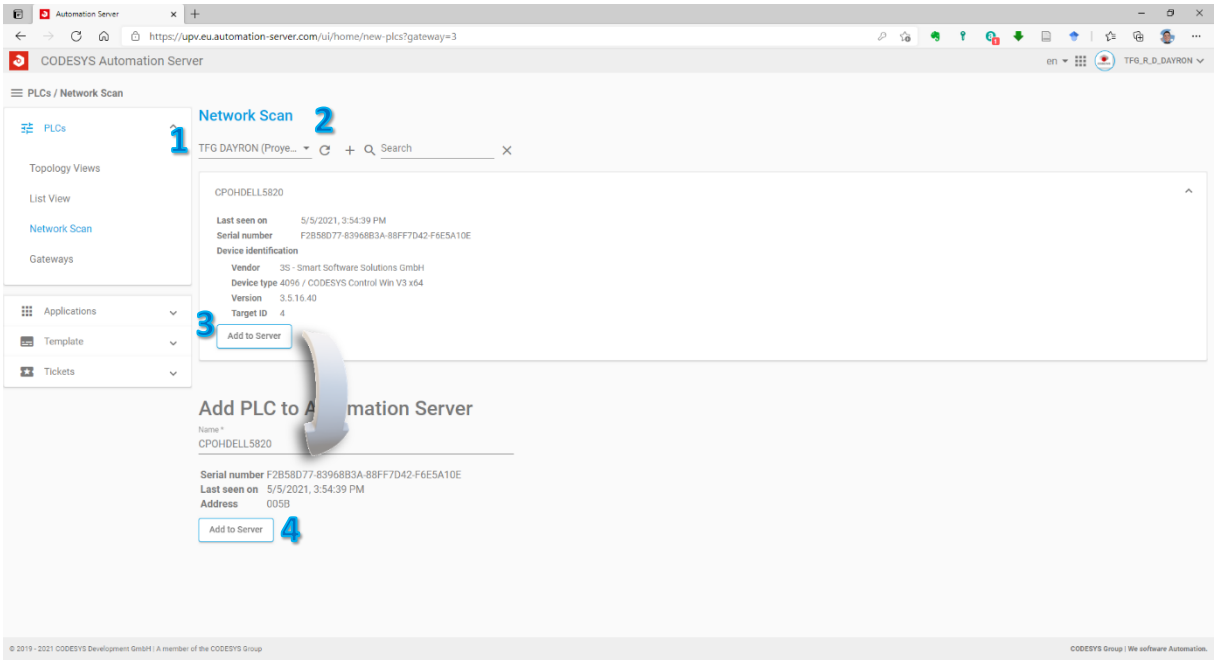
Esta ventana permite al usuario visualizar geográficamente la localización de todos los *PLCs* que haya subido a la plataforma, en el caso del proyecto no brinda mucha información pero para una empresa de desarrollo de soluciones de automatización le permite tener un posicionamiento rápido de todos sus autómatas. Dentro del menú lateral, opción *PLCs*, se encuentra la opción *Gateway* donde se visualizan todas las conexiones configuradas en el *Server*, si se encuentra activa aparece un *led* verde (Figura 170), esto permitirá escanear los autómatas conectados a esta pasarela para acceder a determinados parámetros.

Figura 170. Gateway activa entre CODESYS y Automation Server.



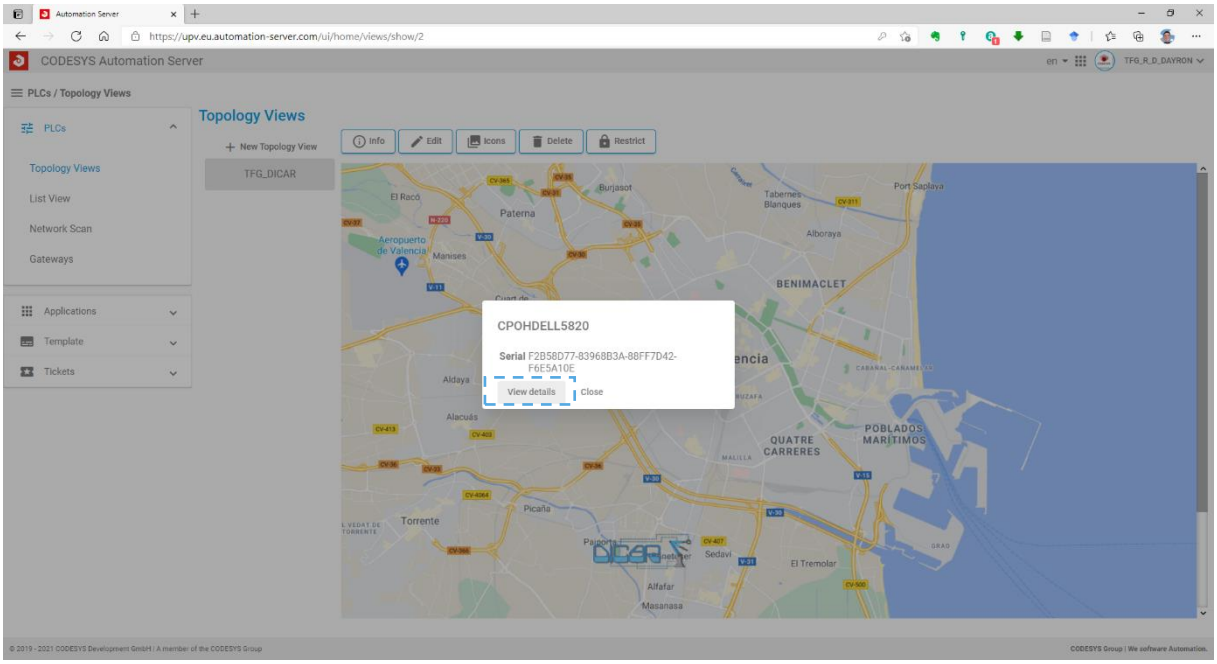
En la opción *Network Scan* se debe seleccionar el *Gateway* con el cual el *PLC* esta enlazado, haciendo *clic izquierdo* sobre él en el desplegable (Figura 171.1), posteriormente presionando sobre el icono de actualización (Figura 171.2) se muestra el *softPLC* alojado en el *HOST*, como se ha visto en otros apartados, coincide con el nombre del *PC "CPOHDELL5820",* el botón *Add to Server* permite crear en la plataforma un dispositivo digital idéntico al *PLC* alojado en *CODESYS*. Para concluir el procedimiento es necesario volver a presionar *Add to Server* en la nueva ventana mostrada tras el paso anterior. (Figura 171.4).

*Figura 171. Procedimiento para crear un **Digital Twin** del PLC alojado en CODESYS.*



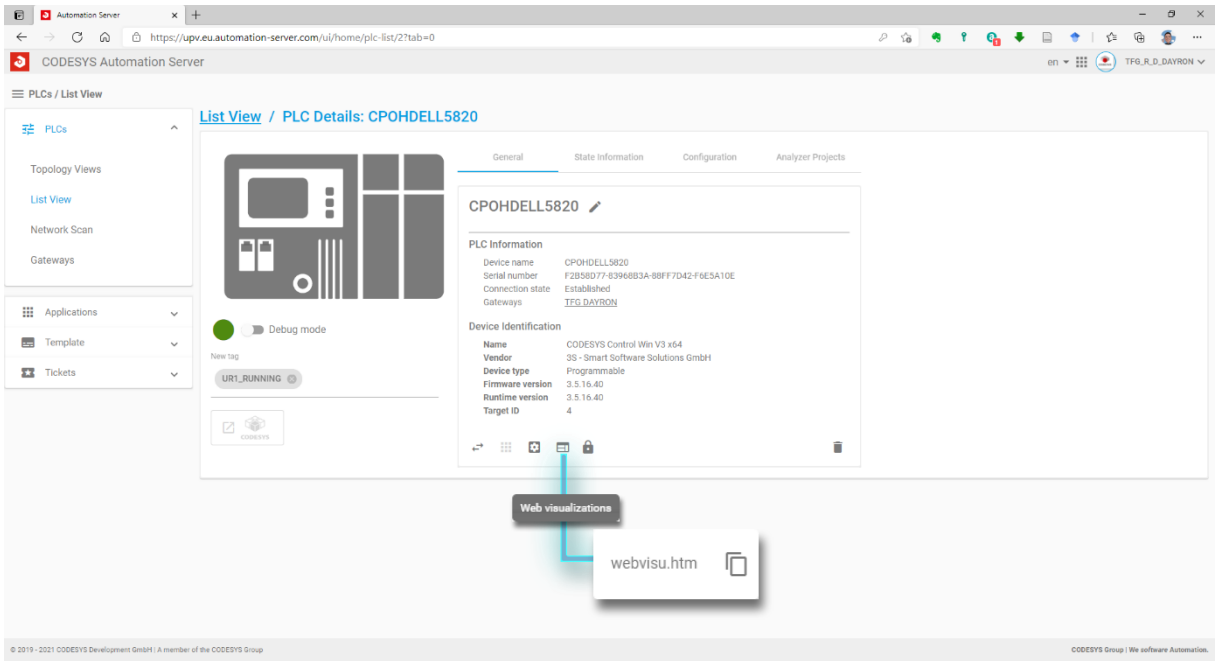
Regresando a la ventana de tipología, botón *Edit*, se puede enlazar el *Digital Twin* recién creado a la imagen del *PLC* representada con el logo del proyecto, de esta forma el elemento dentro del mapa actúa como un acceso directo a las características del autómata (Figura 172).

*Figura 172. Enlace entre el **softPLC** y el objeto en el visor de topología.*



Presionando el botón *View Details* en la imagen anterior, la plataforma muestra una representación del *PLC* conectado (Figura 173) donde están caracterizados todos los parámetros del autómata, en el fondo de la ventana aparecen un grupo de iconos relacionados, entre los cuales se encuentra uno que permite acceder al *WebServer* previamente diseñado, presionando sobre el botón *Web visualization* y posteriormente *webvisu.htm* una nueva pestaña en el navegador aparece donde se puede observar la ventana *INDEX* del diseño antes descrito (Figura 174).

Figura 173. Detalles del PLC. CODESYS Automation Server.

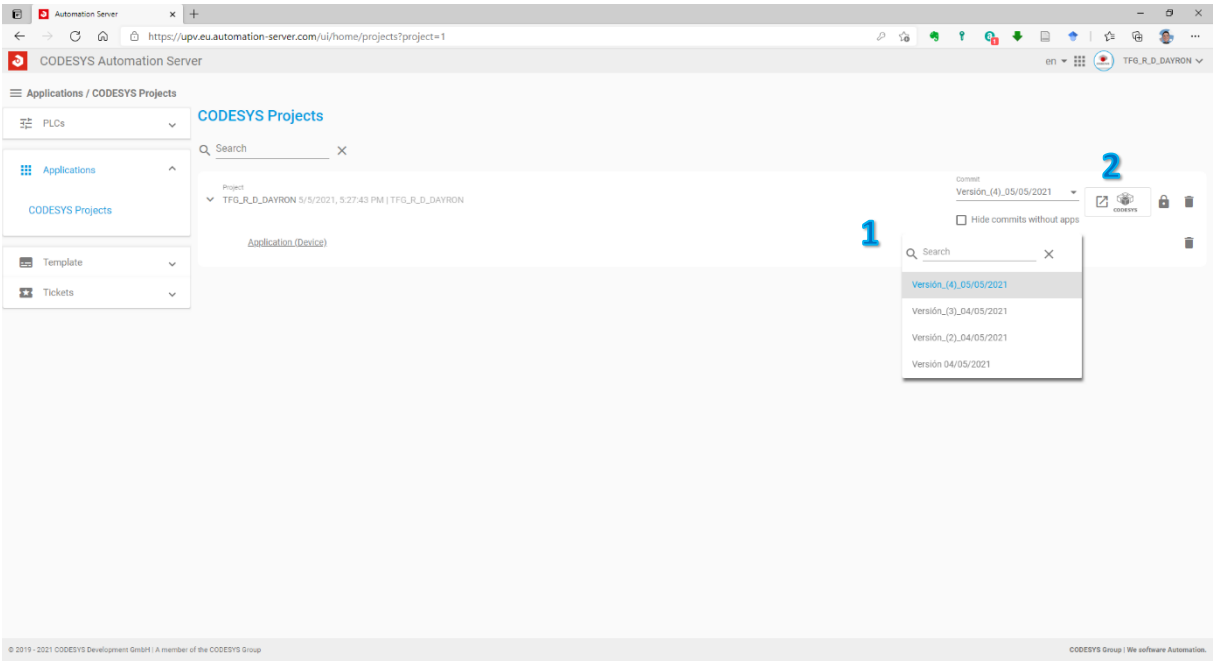


De esta forma se tiene acceso al *WebServer* del proyecto independientemente del sitio donde se encuentre ejecutándose el *softPLC*, rompiendo la barrera de la programación anclada a un *PC* convencional, pudiendo realizar dicha tarea en una *Tablet* o un *Smartphone* en cualquier punto geográfico. La imagen debajo muestra el *WebServer* ejecutándose en *CODESYS Automation Server* con la ventana de *RoboDK* a la derecha donde se puede comprobar el sincronismo de ambos elementos, dando soporte una vez más a la idea inicial de crear un *Digital Twin* de una celda completamente automatizada.

*Figura 174. Visualización del **WebServer** desde CODESYS Automation Server.*

En el menú principal de *CODESYS Automation Server* (Figura 175.1), opción *Applications → CODESYS Projects* se encuentra un repositorio con todas las versiones que se han subido a la plataforma, pudiendo escoger cuál de ellas se estará ejecutando en el *PLC,* además, es posible descargar el proyecto al ordenador, presionando el botón *Open with Codesys* (Figura 175.2).

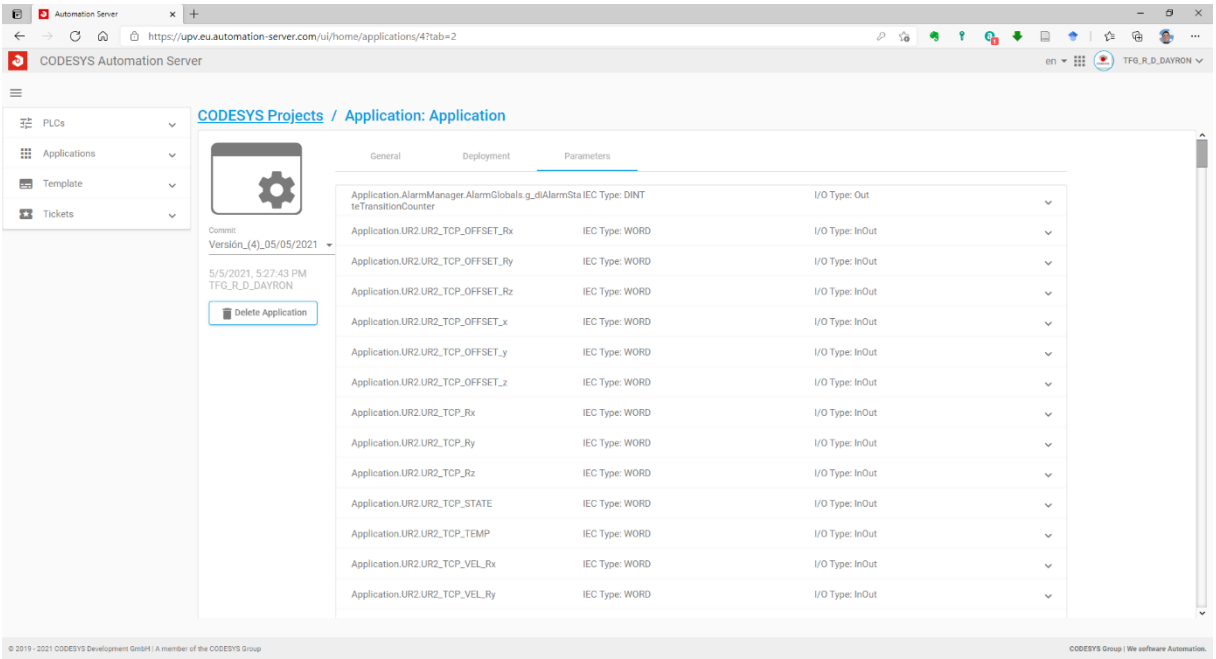
Figura 175. Visualización de las versiones subidas a la plataforma y acceso a la descarga.



Por último, (Figura 176) se puede visualizar el conjunto de variables existentes en el *PLC* en la pestaña *Parameters*, aunque el autor del proyecto considera que en el listado podría aparecer el valor de cada variable.

Con este apartado se dan por cumplidos los objetivos marcados en el *TFG*.

Figura 176. Tabla con todas las variables del PLC.



6. JUSTIFICACIÓN DE LA SOLUCION ADOPTADA

6.1. PROGRAMA DE EJECUCIÓN

El desarrollo de la plataforma ha sido un proceso continuo y largo en el tiempo, el autor ha dedicado más de 6 horas diarias la mayor parte de los días de la semana, concentrando la investigación, el grueso de las horas invertidas para su solución.

Por ello se presenta un diagrama de *Gantt* (Figura 177) como herramienta útil para planificar el proyecto, proporciona una vista general de las tareas programadas, todos los apartados desarrollados en el tema 5 están implicados para de esta forma conocer qué tareas han sido completadas y en qué fecha**.**

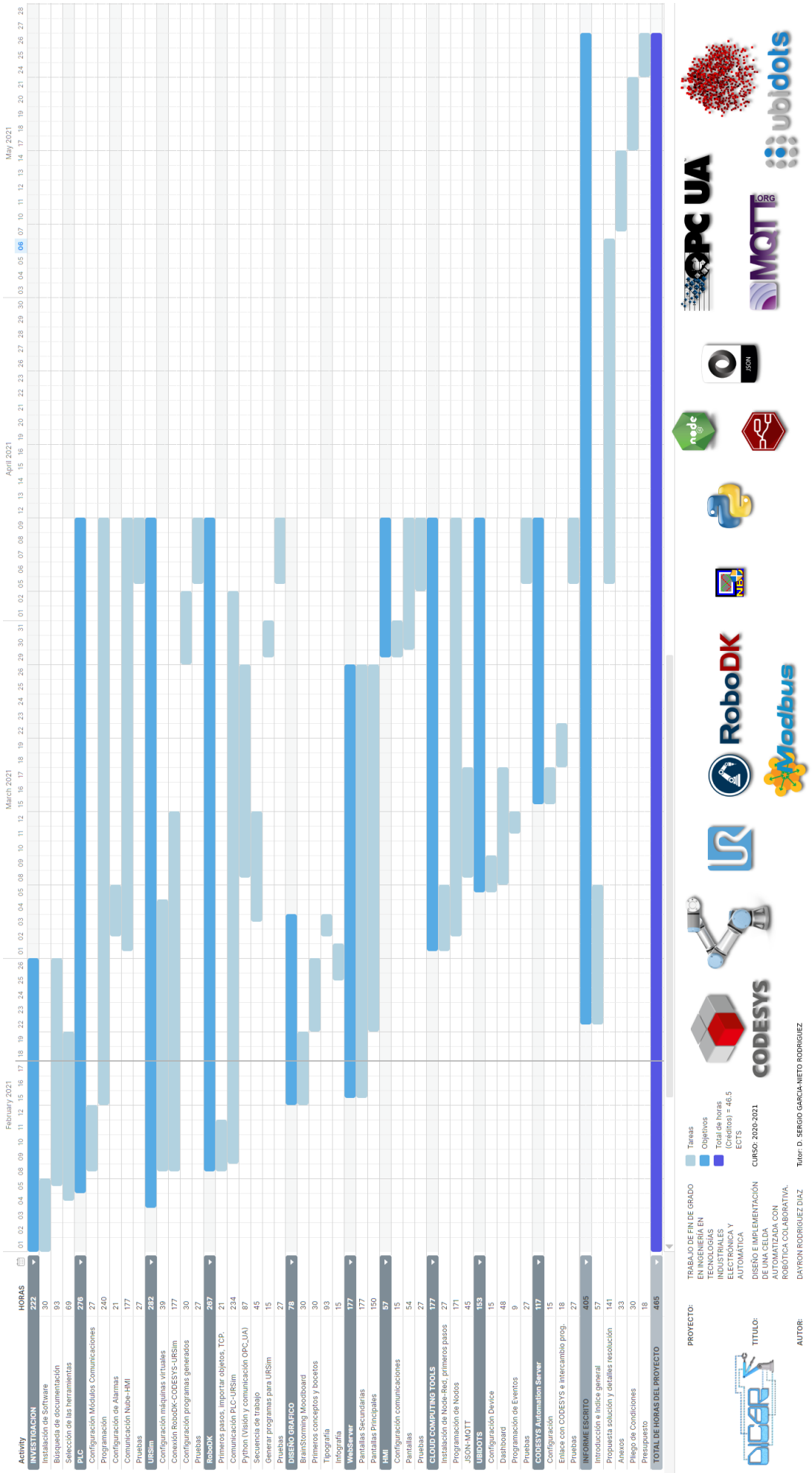
El diagrama de *Gantt* desglosa múltiples tareas y líneas temporales en una vista general única. De esta manera, es posible conocer dónde está concentrada la fuerza de trabajo en cada momento. Además, el diagrama se utiliza para mostrar al lector cómo están organizadas las tareas y qué recursos se destinan a cada una de ellas**.**

Al crear una vista general clara, el autor es consciente de su rendimiento general y puede adaptar su rutina de trabajo según las necesidades del proyecto. Además, se es más consciente de las tareas interdependientes, y, por lo tanto, se comprende mejor el impacto de los retrasos dentro del proyecto. Este tipo de planificación fomenta la gestión del tiempo de investigación y una mejor organización de las tareas.

Como es lógico el proyecto ha sufrido modificaciones inesperadas, pero al tener una vista general de los cambios inesperados dentro de los objetivos o los plazos de tiempo se pudo ajustar las tareas y recursos como corresponde**.**

6.2. DIAGRAMA DE GANTT

Figura 177. Diagrama de Gantt del proyecto TFG_DICAR.



6.3. PRUEBAS FINALES

Tal como muestra el diagrama de *Gantt* del apartado anterior, la semana comprendida entre el 5 y el 9 de *abril* se dedicó íntegramente al desarrollo de pruebas entre todos los elementos que conforman la solución a los objetivos del proyecto.

Para ello *CODESYS* se encuentra conectado al *softPLC*, al mismo tiempo *UR1* y *UR2*, que se localizan dentro de *RoboDK* (Figura 178.1), se enlazan a través del protocolo *RTDE* a las máquinas virtuales *URSim_UR1* y *URSim_UR2* respectivamente (Figura 178.2 y Figura 178.3), se lanza la simulación de la pantalla *HMI* (Figura 178.4), por medio de la ventana *símbolo de sistema* de inicia *Node-Red* (Figura 178.5), en un navegador de carga el *WebServer* (Figura 178.6) y al mismo tiempo se lanza desde *CODESYS Automation Server* (Figura 178.7), el último elemento está formado por la *Nube* de *UBIDOTS,* para liberar la carga del *PC HOST*, el autor instala la aplicación del mismo nombre en su teléfono (Figura 178.8). Con todos los elementos en funcionamiento, desde *RoboDK* se inicia la secuencia declarando que se ejecute en *Bucle.*

Figura 178. Conjunto de aplicaciones ejecutándose en el proceso de pruebas finales.

6.4. RESULTADOS OBTENIDOS

Como se ha explicado en el desarrollo del presente informe, la investigación ha creado una plataforma de entrenamiento para alumnos e interesados que simula el control, comunicación y representación de los datos de una celda automatizada empleando robots colaborativos y un sistema de visión artificial que permite el intercambio de una pieza entre dos unidades virtuales *UR3.*

Para enlazar todos los elementos, se diseñó un sistema de intercambio de parámetros y datos empleando fundamentalmente *Modbus* en sus versiones *TCP* para la comunicación entre *CODESYS* y las máquinas virtuales de *UR*, y *RTU* para la conexión con el *HMI* de *Omron.* Además, el autor tuvo que entrar en contacto por primera vez con el lenguaje de programación *Phyton,* con el cual se implementa el algoritmo de procesamiento de las coordenadas para la visión artificial y el correspondiente intercambio de datos desde *RoboDK* hacia el *PLC* virtual. Por último, se incorporan dos sistemas de *cloud computing* que permiten configurar la posición de entrega de la pieza desde la plataforma *UBIDOTS* y una segunda desarrollada por *CODESYS* donde se puede almacenar todo el proyecto programado para el *PLC* en su *Nube CODESYS Automation Server*, mencionando la posibilidad de gestionar y visualizar el *WebServer* alojado dentro del *PLC* desde el mismo servicio. Una vez todos los elementos entran en funcionamiento, se desarrolla un ciclo de depuración de errores y cálculo del tiempo de ciclo del proceso (Tabla 13), permitiendo afianzar el planteamiento de crear un *Digital Twin* funcional de una celda física, permitiendo la migración de todos los programas y periféricos, a sus homólogos reales en caso de que se contase con ellos. (Figura 179).

Por otra parte, se quiere hacer mención del proceso de diseño de toda la interfaz gráfica de la plataforma, demostrando que un ingeniero con un mínimo de formación en técnicas de diseño digital, puede simplemente buscando información y un grupo de imágenes, crear un concepto que abarque todo el proceso de creación de una solución gráfica para una plataforma virtual.

*Figura 179. Materialización de la hipótesis inicial de crear un **Digital Twin** relacionada con una celda automatizada con robótica colaborativa.*

En el proceso de depuración del código se realizaron diferentes pruebas que permitieron la validación de la lógica de control implementada en el *PLC*, así como toda la secuencia desarrollada en *RoboDK*, esto permitió calcular el tiempo de ciclo de la celda, segmentando cada tarea asociada a los robots, en

la (Tabla 13) se recopilan los tiempos medidos hasta completar todo el ciclo de trabajo, cabe mencionar que los datos corresponden a una secuencia donde *UR1* entrega la pieza en la posición 3, lo cual requiere invertir más tiempo en el movimiento de desplazamiento.

Tabla 13. Cálculo del tiempo de ciclo de la secuencia a través de los distintos pasos.

Paso - Secuencia	Tiempo (s)
Procedimiento de recogida de pieza	00:15.37
Comunicación con *CODESYS* (Parámetros cámara y Posición de entrega)	00:11.05
Movimiento de *UR1* hasta la posición de intercambio	00:26.45
Captura de Imagen	00:01.43
Procesamiento de los datos de visión artificial	00:09.94
Procedimiento de intercambio de pieza entre *UR1-UR2*	00:13.13
Transporte de pieza y dejada por *UR2*	00:41.43
TOTAL	**01:58.81**

Como se puede observar, el resultado obtenido no es aceptable en ningún sector industrial, de forma general una estación consume un tiempo de ciclo aproximado de *50 a 58 segundos*.

No es objetivo del proyecto desarrollar una secuencia que cumpla con esta especificación, para ello, el usuario debe ajustar las velocidades a las cuales los robot se desplazan de un punto a otro, de esta forma es posible cumplir con un tiempo de ciclo dentro de los parámetros de funcionamiento en una cadena de producción real.

7. CONCLUSIONES

El presente trabajo es la desenlace de más de 19 años de estudio del autor, tras iniciar la carrera en *Cuba* en el año 2002, así como la puesta en práctica de los conocimientos adquiridos en asignaturas del grado cursadas en la *UPV*, mencionando distintas disciplinas como *Visión Artificial*, *Instalaciones de Control Industrial* y *Sistemas Robóticos*, por otro lado, la experiencia adquirida como ***Programador de Robótica Industrial durante 6 años*** en los cuales se tiene contacto con los principales fabricantes de robots mundiales, permitió abarcar áreas tan heterogéneas en aras de crear una plataforma virtual que satisfaga cada uno de los objetivos marcados.

El proyecto permite acercar las nuevas tecnologías que dominan el paradigma del concepto *Industria 4.0.* a futuros alumnos de grado o máster, o poner en funcionamiento proyectos que podían verse afectados por la escasez de componentes físicos como *PLC, robots, HMI,* etc.

Con el desarrollo de la investigación, se fueron incorporando nuevos objetivos al proyecto, de forma que enriquecieran el resultado final y ofrecieran al lector una herramienta de trabajo operativa, funcional y con un diseño gráfico que resultara atractivo.

Según se pudo comprobar, la máquina virtual de *Universal Robots* se puede comunicar con elementos externos y de esta forma ofrecer una alternativa tangible a un robot real con el correspondiente ahorro en coste económico, por consiguiente se introduce en la plataforma el algoritmo fisicomatemático que permite simular el comportamiento de un sistema robotizado, obteniendo todos los parámetros necesarios para su control y programación.

Por medio del *IDE* de *CODESYS* se pudo implementar toda la lógica para el control y la supervisión de los elementos de la plataforma a través de un *PLC* virtualizado haciendo uso de dos servicios instalados en *Windows* que permiten emular un *softPLC.*

Al mismo tiempo se pudo conceptualizar, diseñar y programar toda una interfaz gráfica para la representación de la información de la celda, así como su manejo por medio de dos elementos diferentes, un servidor embebido dentro del *PLC* conocido como *WebServer* y una interfaz gráfica materializada con una pantalla *HMI* de la marca *Omron.*

Todo sistema *Digital Twin* debe ser capaz de ser observado en un software con un entorno *CAM* para comprobar el funcionamiento del proceso, *RoboDK* permite realizar este hito en el proyecto, permitiendo además, la interconexión de los robot programados con el *PLC* empleando señales *RTDE* que proporciona una forma de sincronizar aplicaciones externas con el controlador *UR* a través de una conexión *TCP/IP* estándar sin romper ninguna de las propiedades en tiempo real del controlador del robot. Añadiendo la posibilidad de incorporar programas escritos en lenguaje *Python* para articular los elementos gráficos contenidos en el programa con sistemas externos, lo que permitió crear un algoritmo para la visión artificial que permitió el intercambio de la pieza entre los robots.

En la resolución de los objetivos del proyecto se incluyen dos servicios de *cloud computing* independientes, el primero implementado con *UBIDOTS,* el cual permite programar un *Dashboard* para monitorizar parámetros de la celda y configurar la posición en la cual *UR1* entrega la pieza, ya que dicha plataforma tiene componentes que la hacen ideal para procesos de automatización industrial y

en segundo caso, se utiliza la novedosa *Nube* de *CODESYS* [2019], denominada *CODESYS Automation Server* que permite alojar toda la programación del *PLC* en un servicio web donde el usuario pueda consultar el estado del controlador, distintas versiones del código entre las cuales se puede elegir cual ejecutar, o programar directamente el autómata desde un navegador web rompiendo la barrera de la programación anclada a un *PC* convencional, pudiendo realizar dicha tarea en una *Tablet* o un *Smartphone* en cualquier punto geográfico.

Sin embargo, como todo proyecto de investigación, existen limitaciones que se traducen en futuras mejoras de la plataforma atendiendo a las necesidades del usuario. Una de ellas consiste en controlar el proceso a través del *HMI* o el *WebServer*, en vez de iniciar el ciclo en *RoboDK* y establecer que la secuencia se ejecute en bucle, para ello *RoboDK* tiene una *API* denominada ***UR_ActiveMonitoring.py*** accesible desde *C:/RoboDK/Library/Macros/*, la cual permite monitorizar el estado de un robot *UR* y actualizar la posición de la versión digital en *RoboDK*.

Existe además la posibilidad de enviar los programas creados a las controladoras de los robots vía *FTP*, utilizando *FileZilla* uno de los clientes FTP más populares, gratis, de código abierto y disponible tanto para Windows como para Mac y Linux.

Análogamente, queda a disposición del lector la posibilidad de diseñar una secuencia de control basada en los bloques que componen el método de automatización *GEMMA*, o cualquier otro equivalente, en el cual se entremezclen señales de marcha-paro, tratamiento de emergencias, solicitud de paros de fin de ciclos entre otras, de esta forma la celda se asemeja más a una estación automatizada con los estándares actuales de diseño y control en el sector de la *Industria 4.0.*

Queda igualmente pendiente el diseño de todas las pantallas del *HMI* para que albergue todas las funcionalidades que posee el *WebServer*, así como la implementación de un tiempo de ciclo que cumpla con los estándares marcados en el sector industrial.

Por último, el autor quiere ratificar su compromiso de diseñar la plataforma donde las herramientas que no sean de uso libre, cuentan con un período de prueba de al menos un mes, lo que posibilita al futuro usuario poder llevar a cabo proyectos de investigación que actualmente eran difíciles de desarrollar debido al escaso número de componentes físicos, consolidando el pensamiento de una enseñanza libre y democrática.

8. BIBLIOGRAFÍA

99454_UR3e_User_Manual_en_US.pdf. (s. f.). Recuperado 21 de abril de 2021, de https://s3-eu-west-1.amazonaws.com/ur-support-site/68217/99454_UR3e_User_Manual_en_US.pdf

Arial. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Arial&oldid=131657861

automaticaeinstrumentacion.com. (s. f.). ***CODESYS lanza la nueva plataforma 4.0 Automation Server***. Automática e Instrumentación - La revista de la Industria 4.0. Recuperado 4 de mayo de 2021, de https://www.automaticaeinstrumentacion.com/texto-diario/mostrar/2735205/codesys-lanza-nueva-plataforma-40-automation-server

Belzuz—**Estudio e implementación de un sistema IoT en un brazo robot y control en TIA Portal*.pdf***. (s. f.). Recuperado 17 de abril de 2021, de https://upcommons.upc.edu/bitstream/handle/2117/335646/Report.pdf?sequence=3&isAllowed=y

Blanco, R., Fontrodona, J., & Poveda, C. (2017). ***LA INDUSTRIA 4.0: EL ESTADO DE LA CUESTIÓN.*** *Nº406*, 14.

Boden, M. A. (2017). ***Inteligencia Artificial***. Turner.

Bourhis, P., Reutter, J. L., & Vrgoč, D. (2020). **JSON: Data model and query languages**. *Information Systems*, *89*, 101478. https://doi.org/10.1016/j.is.2019.101478

*CODESYS Store International — **CODESYS Automation Server Connector***. (s. f.). Recuperado 5 de mayo de 2021, de https://store.codesys.com/codesys-automation-server-connector.html?___store=en

*CODESYS Store International — **CODESYS Control Win SL***. (s. f.). Recuperado 17 de abril de 2021, de https://store.codesys.com/codesys-control-win-sl.html?___store=en

*CODESYS Store International — **CODESYS Edge Gateway for Windows***. (s. f.). Recuperado 5 de mayo de 2021, de https://store.codesys.com/cas/codesys-edge-gateway-for-windows.html?___store=en&___from_store=default

Creating Devices in Ubidots. (s. f.). Recuperado 3 de mayo de 2021, de http://help.ubidots.com/en/articles/2226697-creating-devices-in-ubidots

*Datos IRB 14000 YuMi — **IRB 14000 YUMI - ROBOT COLABORATIVO** (Robots colaborativos) | ABB*. (s. f.). Recuperado 17 de abril de 2021, de https://new.abb.com/products/robotics/es/robots-colaborativos/yumi/datos-irb-14000-yumi

***Emerging_Jobs_Report_112119_SP.pdf*.** (s. f.). Recuperado 4 de marzo de 2021, de https://business.linkedin.com/content/dam/me/business/en-us/talent-solutions/emerging-jobs-report/Emerging_Jobs_Report_112119_SP.pdf

Es-e-series-brochure.pdf. (s. f.). Recuperado 18 de abril de 2021, de https://www.universal-robots.com/media/1802610/es-e-series-brochure.pdf

Fabricante Robots colaborativos – ***Cobots de Universal Robots*.** (s. f.). Recuperado 21 de abril de 2021, de https://www.universal-robots.com/es/

***FreeOpcUa/python-opcua*.** (s. f.). GitHub. Recuperado 25 de abril de 2021, de https://github.com/FreeOpcUa/python-opcua

Gamez, M. J. (s. f.). **Objetivos y metas de desarrollo sostenible**. *Desarrollo Sostenible*. Recuperado 24 de abril de 2021, de https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

García, D. (s. f.). ***CODESYS Automation Server proporciona un análisis de datos intuitivo*—*InfoPLC***. Recuperado 4 de mayo de 2021, de https://www.infoplc.net/noticias/item/108268-codesys-automation-server-proporciona-un-analisis-de-datos-intuitivo

Garg, S. K., Versteeg, S., & Buyya, R. (2013). **A framework for ranking of cloud computing services**. *Future Generation Computer Systems*, *29*(4), 1012-1023. https://doi.org/10.1016/j.future.2012.06.006

***Gazebo*.** (s. f.). Recuperado 18 de abril de 2021, de http://gazebosim.org/

Grado en Ingeniería Electrónica, Industrial y Automática, & Grado en Ingeniería Electrónica Industrial y Automática. (s. f.). *INSTALACIONES DE CONTROL INDUSTRIAL.* ***HMI-SCADAS. PROTOCOLO OPC*** [Power Point].

***HMI Design Workbook*.** (s. f.). Siemens.Com Global Website. Recuperado 26 de abril de 2021, de https://new.siemens.com/global/en/products/automation/simatic-hmi/hmi-design-workbook.html

jecrespom. (2020, marzo 5). **Qué es Node-RED.** *Aprendiendo Arduino*. https://aprendiendoarduino.wordpress.com/2020/03/05/que-es-node-red/

JSON. Formatos. Informática. *Bartolomé Sintes Marco. Www.mclibre.org*. (s. f.). Recuperado 30 de abril de 2021, de https://www.mclibre.org/consultar/informatica/lecciones/formato-json.html

***KUKA Sunrise.OS*.** (s. f.). KUKA AG. Recuperado 18 de abril de 2021, de https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/software/software-de-sistema/sunriseos

***LBR iiwa*.** (s. f.). KUKA AG. Recuperado 17 de abril de 2021, de https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industrial/lbr-iiwa

Learn Engineering Online With Simumatik's Cloud-Based Platform. (s. f.). *Simumatik*. Recuperado 19 de abril de 2021, de https://3.121.44.205/education/

Mahato, B., Maity, T., & Antony, J. (2015). **Embedded Web PLC: A New Advances in Industrial Control and Automation.** *2015 Second International Conference on Advances in Computing and Communication Engineering*, 156-160. https://doi.org/10.1109/ICACCE.2015.141

Ministerio de Industria, Energía y Turismo. (2016). ***Industria Conectada 4.0. La industria del futuro ha llegado*.** https://www.youtube.com/watch?v=eUDEJpBqZhA&t=5s

NB-series Programmable Terminals ***NB-Designer Operation Manual***. (s. f.). 560.

***No_me_hagas_pensar.pdf*.** (s. f.). Recuperado 26 de abril de 2021, de http://www.nerviooptico.com/no_me_hagas_pensar.pdf

***OPC UA | B&R Industrial Automation*.** (s. f.). Recuperado 20 de abril de 2021, de https://www.br-automation.com/es-es/tecnologias/opc-ua/

P145_cp2e_datasheet_en.pdf. (s. f.). Recuperado 17 de abril de 2021, de https://assets.omron.eu/downloads/datasheet/en/v2/p145_cp2e_datasheet_en.pdf

PIXEL SISTEMAS. (2016, septiembre 30). ***Future of Manufacturing—Visión de Siemens sobre Industria 4.0*.** https://www.youtube.com/watch?v=GF1MYUdFcqo

Post Processors *— RoboDK Documentation*. (s. f.). Recuperado 26 de abril de 2021, de https://robodk.com/doc/en/Post-Processors.html#PostProcessor

¿Qué es MQTT? Su importancia como protocolo IoT. (s. f.). *Luis Llamas*. Recuperado 30 de abril de 2021, de https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

Real-Time Data Exchange (RTDE) Guide*—22229*. (s. f.). Recuperado 21 de abril de 2021, de https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/

***SIMATIC S7-1200 -Take control of communication*.** (s. f.). [Newton_ps-detail]. Siemens.Com Global Website. Recuperado 17 de abril de 2021, de https://new.siemens.com/global/en/products/automation/systems/industrial/plc/s7-1200.html

Sunyaev, A. (2020). ***Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*.** Springer International Publishing. https://doi.org/10.1007/978-3-030-34957-8

TM241CE24T.pdf. (s. f.). Recuperado 17 de abril de 2021, de https://download.schneider-electric.com/files?p_enDocType=Catalog&p_File_Name=Cat%C3%A1logo+-+Controladores+l%C3%B3gicos+Modicon+M241.pdf&p_Doc_Ref=DIA3ED2140107ES

Universal Robots — *Documentación RoboDK*. (s. f.). Recuperado 26 de abril de 2021, de https://robodk.com/doc/es/Robots-Universal-Robots.html#UR-RoboDK

Ursim_vmoracle_installation_guide_v3_es.pdf. (s. f.). Recuperado 18 de abril de 2021, de https://academy.universal-robots.com/media/r3xlna5e/ursim_vmoracle_installation_guide_v3_es.pdf

***V412_nb_series_hmi_datasheet_es.pdf*.** (s. f.). Recuperado 28 de abril de 2021, de https://assets.omron.eu/downloads/datasheet/es/v17/v412_nb_series_hmi_datasheet_es.pdf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

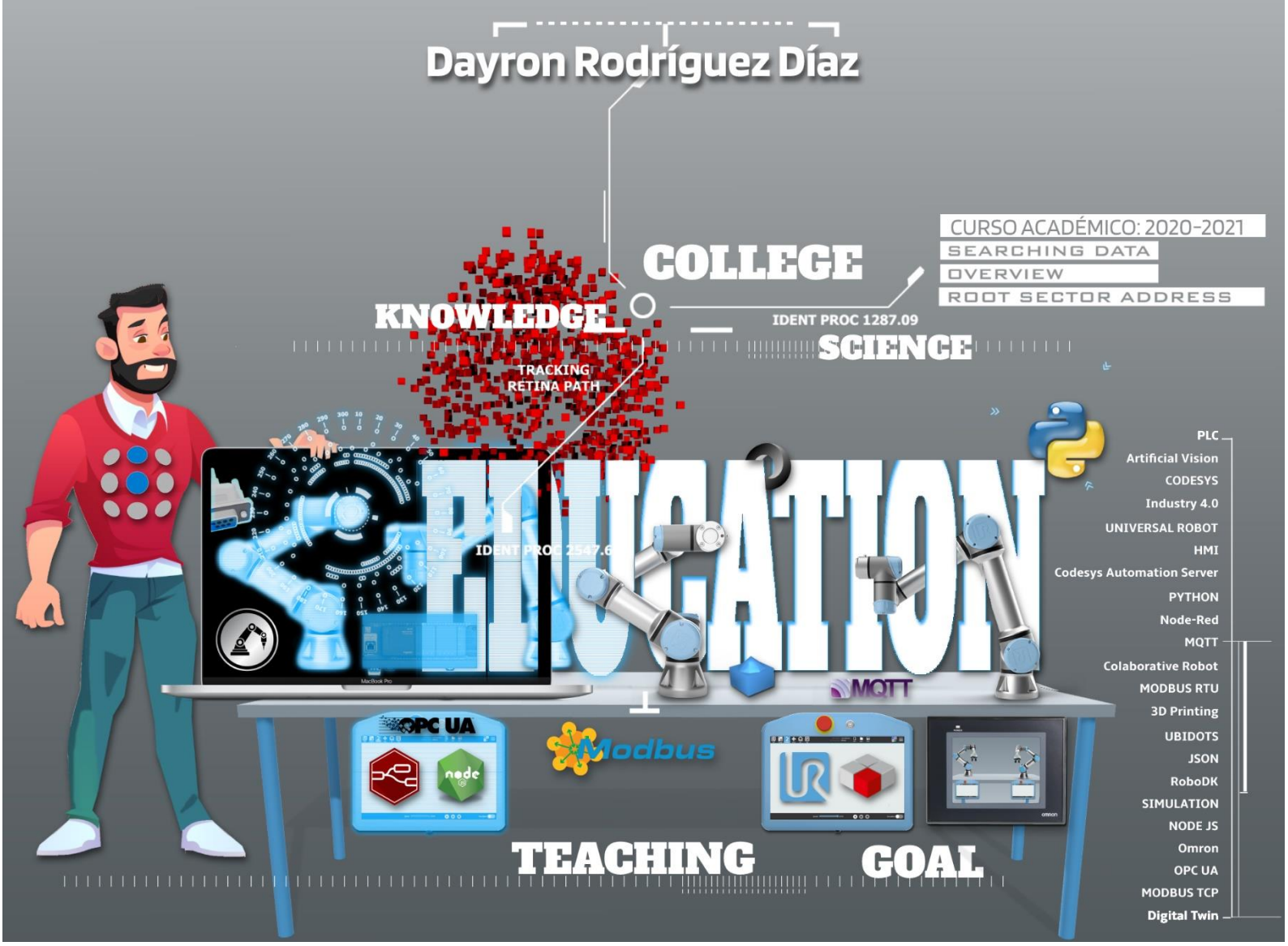
Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA

ANEXO I: TABLA DE CONTROL DE POSICIONES

Dayron Rodríguez Díaz



POSICIONES	ROBOT	EJE 1	EJE 2	EJE 3	EJE 4	EJE 5	EJE 6	DESCRIPCIÓN	SEQUENCE	PASO
POS_0	UR1	-89,82	-32,97	-76,35	-70,68	89,82	0	UR1 EN HOME	SQ0	P0
POS_1	UR1	-67,55	-102,85	-75,37	-91,78	90	22,45	UR1 EN POS ANTES PICK	SQ1	P1
POS_2	UR1	-67,55	-109,65	-99,95	-60,4	90	22,45	UR1 EN POS PICK	SQ2	P2
POS_3	UR1	-67,55	-103,15	-83,31	-83,31	90	22,45	UR1 EN POS AFTER PICK	SQ3	P3
POS_4	UR1	93,31	-40,03	54,3	-104,27	90	-183,31	UR1 EN POS DELIVERY 1	SQ4	P4_1
	UR1	67,27	-80,83	94,37	-71,33	76,83	-161,3	UR1 EN POS DELIVERY 2	SQ5	P4_2
	UR1	75,72	-39,11	46,8	-100,39	115,63	-148,18	UR1 EN POS DELIVERY 3	SQ6	P4_3
POS_5	UR1	92,85	-28,75	61,98	-123,2	89,33	-182,85	UR1 EN POS AFTER DELIVERY 1	SQ7	P5_1
	UR1	61,27	-85,49	126,48	-100,42	73,07	-155,93	UR1 EN POS AFTER DELIVERY 2	SQ8	P5_2
	UR1	80,57	-24,78	45,7	-116,27	114,7	-153,68	UR1 EN POS AFTER DELIVERY 3	SQ9	P5_3
POS_6	UR1	15,55	-68,06	61,44	-204,02	46,38	-80,94	UR1 EN POS BEFORE HOME	SQ10	P6
POS_7	UR1	-89,82	-32,97	-76,35	-70,68	89,82	0	UR1 EN POS HOME	SQ11	P7
POS_0	UR2	-173,76	-67,74	-101,95	-10,31	173,76	0	UR2 EN HOME	SQ12	P0
POS_1	UR2	-95,23	-106,12	-65,26	-8,26	95,24	90,66	UR2 EN APPROX_PICK DELIVERY 1	SQ13	P1_1
	UR2	-64,21	-123,66	-80,75	62,18	68,98	74,24	UR2 EN APPROX_PICK DELIVERY 2	SQ14	P1_2
	UR2	-59,75	-104,78	-42,63	-48,83	75,7	73,68	UR2 EN APPROX_PICK DELIVERY 3	SQ15	P1_3
POS_2	UR2	-94,53	-119,37	-46,04	-14,57	94,53	90,67	UR2 EN POS_PICK DELIVERY 1	SQ16	P2_1
	UR2	-67,52	-132	-58,6	47,29	71,52	76,53	UR2 EN POS_PICK DELIVERY 2	SQ17	P2_2
	UR2	-62,07	-119,68	-24,34	-52,41	77,83	73,13	UR2 EN POS_PICK DELIVERY 3	SQ18	P2_3
POS_3	UR2	-94,53	-119,37	-46,04	-14,57	94,53	90,67	UR2 EN POS_PICK NO NOX DELIVERY 1	SQ19	P3-1
	UR2	-67,52	-132	-58,6	47,29	71,52	76,53	UR2 EN POS_PICK NO NOX DELIVERY 2	SQ20	P3-2
	UR2	-62,07	-119,68	-24,34	-52,41	77,83	73,13	UR2 EN POS_PICK NO NOX DELIVERY 3	SQ21	P3-3
POS_4	UR2	-97,46	-87,12	-86,78	-6,08	97,46	90,68	UR2 AFTER PICK DELIVERY 1	SQ22	P4_1
	UR2	-55,84	-114,95	-110,44	85,17	62,33	68,68	UR2 AFTER PICK DELIVERY 2	SQ23	P4_2
	UR2	-53,13	-85,4	-56,7	-55,09	69,26	75,81	UR2 AFTER PICK DELIVERY 3	SQ24	P4_3
POS_5	UR2	-55,84	-66,85	-79,84	-123,3	90	35,6	UR2 EN APPROX_DROP	SQ25	P5
POS_6	UR2	-55,84	-63,38	-142,64	-63,98	90	35,6	UR2 EN DROP	SQ26	P6
POS_7	UR2	-55,84	-58,4	-131,44	-80,16	90	35,6	UR2 EN AFTER_DROP	SQ27	P7
POS_8	UR2	-173,76	-67,74	-101,95	-10,31	173,76	0	UR2 EN HOME CON BOX	SQ28	P8

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

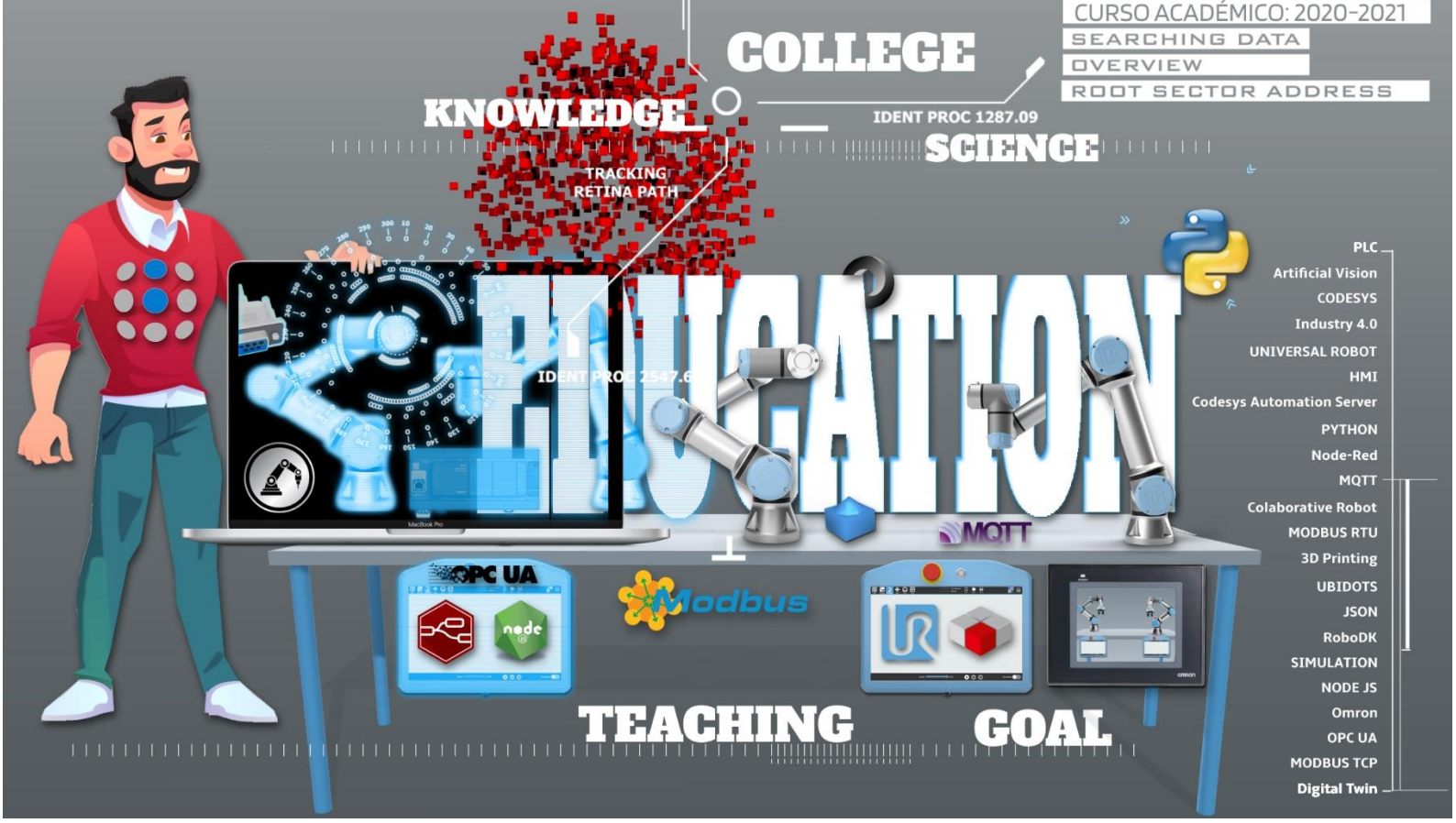
Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA

ANEXO II: UR MODBUS DATA SERVER

Dayron Rodríguez Díaz



REGISTER ADDRESS

R=Read, W=Write

Address	Pre 3.0	3.0	3.1	R	W	
0	x	x	x	*		Inputs, bits 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
1	x	x	x	*	*	Outputs, bits 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
2	x	x	x		*	SetOutputsBitsMask 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
3	x	x	x		*	ClearOutputsBitsMask 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
4	x	x	x	*		Analog input 0 (0-65535)
5	x	x	x	*	*	Analog input 0 domain e {0=current[mA], 1=voltage[mV]}
6	x	x	x	*		Analog input 1 (0-65535)
7	x	x	x	*	*	Analog input 1 domain e {0=current[mA], 1=voltage[mV]}
8	x	x	x	*		Analog input 2 (tool) (0-65535)
9	x	x	x	*	*	Analog input 2 (tool) domain e {0=current[mA], 1=voltage[mV]}
10	x	x	x	*		Analog input 3 (tool) (0-65535)
11	x	x	x	*	*	Analog input 3 (tool) range e {0=current[mA], 1=voltage[mV]}
16	x	x	x	*	*	Analog output 0 output (0-65535)
17	x	x	x	*	*	Analog output 0 output domain e {0=current[mA], 1=voltage[mV]}
18	x	x	x	*	*	Analog output 1 output (0-65535)
19	x	x	x	*	*	Analog output 1 output domain {0=current[mA], 1=voltage[mV]}
20	x	x	x	*	*	Tool output voltage (V) e {0V, 12V, 24V}
21	x	x	x	*		Tool digital input bits
22	x	x	x	*	*	Tool digital output bits
24	x	x	x	*		Euromap67 input bits (0-15)
25	x	x	x	*		Euromap67 input bits (16-32)
26	x	x	x	*		Euromap67 output bits (0-15) (read only!)
27	x	x	x	*		Euromap67 output bits (16-32) (read only!)
28	x	x	x	*		Euromap 24V voltage
29	x	x	x	*		Euromap 24V current
30	-	x	x	*		Configurable inputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
31	-	x	x	*	*	Configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
32	-	x	x		*	Bit mask configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
33	-	x	x		*	Clear configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
34-127	x	x	x			Reserved for future system variables
128-255	x	x	x	*	*	General purpose 16 bit registers
256-	x	x	x	*		Robot state
512-	x	x	x	*		Reserved
768-	x	x	x	*		Tool states
1024-	x	x	x	*		Reserved
2048-	x	x	-	*		RT Machine control
Robot state						
256	x	x	x			Controller version high nuber
257	x	x	x			Controller version low number
258	x	x	x			Robot mode:
						CB3 and 3.1: Disconnected=0, Confirm_safety=1, Booting=2, Power_off=3, Power_on=4, Idle=5, Backdrive=6, Running=7
						CB2: No_controller=-1, Running=0, Freedrive=1,Ready=2, Initializing=3, Security_stoppde=4, Emergency_stopped=5, Fault=6, Not_connected=8, Shoutdown=9
260	x	x	x			isPowerOnRobot
261	x	x	x			isSecurityStopped

262	x	x	x				isEmergencyStopped
263	x	x	x				isTeachButtonPressed
264	x	x	x				isPowerPuttonPressed
265	x	x	x				isSafetySignalSuchThatWeShouldStop
270	x	x	x				Base joint angle (in mrad)
271	x	x	x				Shoulder joint angle (in mrad)
272	x	x	x				Elbow joint angle (in mrad)
273	x	x	x				Wrist1 joint angle (in mrad)
274	x	x	x				Wrist2 joint angle (in mrad)
275	x	x	x				Wrist3 joint angle (in mrad)
280	x	x	x				Base joint angle velocity (in mrad/s)
281	x	x	x				Shoulder joint angle velocity (in mrad/s)
282	x	x	x				Elbow joint angle velocity (in mrad/s)
283	x	x	x				Wrist1 joint angle velocity (in mrad/s)
284	x	x	x				Wrist2 joint angle velocity (in mrad/s)
285	x	x					Wrist3 joint angle velocity (in mrad/s)
290	x	x	x				Base joint current (in mA)
291	x	x	x				Shoulder joint current (in mA)
292	x	x	x				Elbow joint current (in mA)
293	x	x	x				Wrist1 joint current (in mA)
294	x	x	x				Wrist2 joint current (in mA)
295	x	x	x				Wrist3 joint current (in mA)
300	x	x	x				Base joint temperature (in C)
301	x	x	x				Shoulder joint temperature (in C)
302	x	x	x				Elbow joint temperature (in C)
303	x	x	x				Wrist1 joint temperature (in C)
304	x	x	x				Wrist2 joint temperature (in C)
305	x	x	x				Wrist3 joint temperature (in C)
Joint modes From version 1.7							
310	x	x	x				Base joint mode
311	x	x	x				Shoulder joint mode
312	x	x	x				Elbow joint mode
313	x	x	x				Wrist1 joint mode
314	x	x	x				Wrist2 joint mode
315	x	x	x				Wrist3 joint mode
							List of Joint Modes:
							JOINT_SHUTTING_DOWN_MODE = 236;
							JOINT_PART_D_CALIBRATION_MODE = 237;
							JOINT_BACKDRIVE_MODE = 238;
							JOINT_POWER_OFF_MODE = 239;
							JOINT_NOT_RESPONDING_MODE = 245;
							JOINT_MOTOR_INITIALISATION_MODE = 246;
							JOINT_BOOTING_MODE = 247;
							JOINT_PART_D_CALIBRATION_ERROR_MODE = 248;
							JOINT_BOOTLOADER_MODE = 249;
							JOINT_CALIBRATION_MODE = 250;

						JOINT_FAULT_MODE = 252;
						JOINT_RUNNING_MODE = 253;
						JOINT_IDLE_MODE = 255;
320	-	-	x			Base joint revolution count (number of full turns, typically 0 or 1)
321	-	-	x			Shoulder joint revolution count
322	-	-	x			Elbow joint revolution count
323	-	-	x			Wrist1 joint revolution count
324	-	-	x			Wrist2 joint revolution count
325	-	-	x			Wrist3 joint revolution count
TCP						
400	x	x	x			TCP-x in tenth of mm (in base frame)
401	x	x	x			TCP-y in tenth of mm (in base frame)
402	x	x	x			TCP-z in tenth of mm (in base frame)
403	x	x	x			TCP-rx in mrad (in base frame)
404	x	x	x			TCP-ry in mrad (in base frame)
405	x	x	x			TCP-rz in mrad (in base frame)
410	x	x	x			TCP-x speed in mm/s (in base frame)
411	x	x	x			TCP-y speed in mm/s (in base frame)
412	x	x	x			TCP-z speed in mm/s (in base frame)
413	x	x	x			TCP-rx speed in mrad/s (in base frame)
414	x	x	x			TCP-ry speed in mrad/s (in base frame)
415	x	x	x			TCP-rz speed in mrad/s (in base frame)
420	x	x	x			TCP-x offset in mm (in tool frame)
421	x	x	x			TCP-y offset in mm (in tool frame)
422	x	x	x			TCP-z offset in mm (in tool frame)
423	x	x	x			TCP-rx offset in mrad (in tool frame)
424	x	x	x			TCP-ry offset in mrad (in tool frame)
425	x	x	x			TCP-rz offset in mrad (in tool frame)
450	x	x	x			Robot current (in mA)
451	x	x	x			I/O current (in mA)
768	x	x	x			Tool state
769	x	x	x			Tool temperature (in C)
770	x	x	x			Tool current (in mA)
GUI state						
1024	x	x	x			Reserved
RTMachine						
2048	x	x	-		*	Split time (latches the actual time to the registers 2049-2053) calculates the time from last time the controller was restarted
2049	x	x	-	*		Milliseconds
2050	x	x	-	*		Seconds
2051	x	x	-	*		Minutes
2052	x	x	-	*		Hours
2053	x	x	-	*		Days

COIL ADDRESS					
Address	Pre 3.0	3.0	3.1	R	W	
0-15	x	x	x	*		Inputs, bits 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
16-31	x	x	x	*	*	Outputs, bits 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
32-47	x	x	x		*	SetOutputsBitsMask 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
48-63	x	x	x		*	ClearOutputsBitsMask 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
64-79	x	x	x	*		Euromap67 input bits (0-15)
80-95	x	x	x	*		Euromap67 input bits (16-32)
96-111	x	x	x	*		Euromap67 output bits (0-15) (read only!)
112-127	x	x	x	*		Euromap67 output bits (16-32) (read only!)
128-135	x	x	x	*		Configurable inputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
136-143	x	x	x	*	*	Configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
144-151	x	x	x		*	Bit mask configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
152-159	x	x	x		*	Clear configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
260	x	x	x	*		isPowerOnRobot
261	x	x	x	*		isProtectiveStopped
262	x	x	x	*		isEmergencyStopped
263	x	x	x	*		isTeachButtonPressed
264	x	x	x	*		isPowerPuttonPressed
265	x	x	x	*		isSafetySignalSuchThatWeShouldStop

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

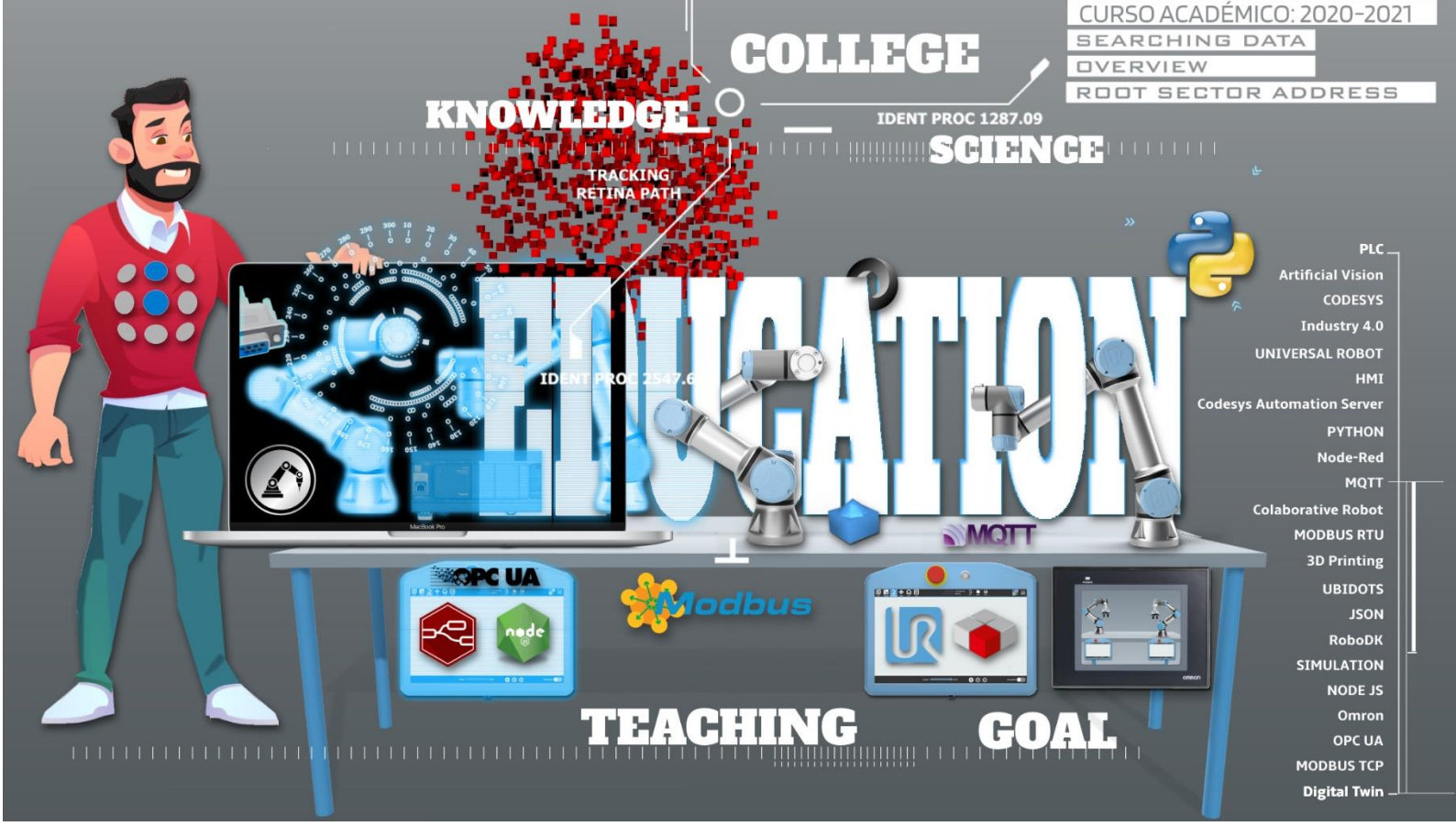
Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA

ANEXO III: CÓDIGO FUENTE NODE-RED

Dayron Rodríguez Díaz



[{"id":"8fe6e59f.0879d8","type":"tab","label":"Flow
1","disabled":false,"info":""},{"id":"a204c408.b829a8","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_RUNNING","props":[{"p":"payl
oad"},{"p":"topic","v":"","vt":"date"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"true","payloadType":"b
ool","x":190,"y":80,"wires":[["8f8adb45.424d38"]]},{"id":"8f8adb45.424d38","type":"OpcUa-
Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3
x64.Application.HMI.UR1_RUNNING","datatype":"Boolean","value":"","name":"","x":420,"y":80,"wires":[["d8eac3ea.f0a9c"]]},{"id":"d8eac
3ea.f0a9c","type":"OpcUa-
Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit"
:"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":80,"wires":[["de2
a8712.b73de8"]]},{"id":"de2a8712.b73de8","type":"change","z":"8fe6e59f.0879d8","name":"bool to
int","rules":[{"t":"change","p":"payload","pt":"msg","from":"true","fromt":"bool","to":"1","tot":"num"},{"t":"change","p":"payload","pt":"
msg","from":"false","fromt":"bool","to":"0","tot":"num"}],"action":"","property":"","from":"","to":"","reg":false,"x":840,"y":80,"wires":[["6
2a782c9.6ce33c"]]},{"id":"62a782c9.6ce33c","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var
json={\"UR1_RUNNING\":msg.payload};\nreturn
{\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1000,"y":80,"wires":[["773cfb97.ec0324","b9a4f503.80677
8"]]},{"id":"773cfb97.ec0324","type":"ubidots_out","z":"8fe6e59f.0879d8","name":"UR_RS","token":"BBFF-
3T9dR3rjU6cIQQug0RjfILqZeTnmut","label_device":"","device_label":"TFG","tier":"business","tls_checkbox":true,"x":1220,"y":220,"wires":
[]},{"id":"2a45ea80.92edf6","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_IDLE","props":[{"p":"payload"},{"p":"topic","v":"","vt":"da
te"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"true","payloadType":"bool","x":170,"y":140,"wires":[["7
e532c46.d5d304"]]},{"id":"7e532c46.d5d304","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3
x64.Application.HMI.UR1_IDLE","datatype":"Boolean","value":"","name":"","x":420,"y":140,"wires":[["53fe8810.7cc9b8"]]},{"id":"53fe881
0.7cc9b8","type":"OpcUa-
Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit"
:"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":140,"wires":[["1e
f7659.839a29a"]]},{"id":"1ef7659.839a29a","type":"change","z":"8fe6e59f.0879d8","name":"bool to
int","rules":[{"t":"change","p":"payload","pt":"msg","from":"true","fromt":"bool","to":"1","tot":"num"},{"t":"change","p":"payload","pt":"
msg","from":"false","fromt":"bool","to":"0","tot":"num"}],"action":"","property":"","from":"","to":"","reg":false,"x":840,"y":140,"wires":[["
18230e95.4d9081"]]},{"id":"18230e95.4d9081","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var
json={\"UR1_IDLE\":msg.payload};\nreturn
{\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":1000,"y":140,"wires":[["773cfb97.ec0324"]]},{"id":"f0fd03d2.71f2f
","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_STOPED","props":[{"p":"payload"},{"p":"topic","v":"","vt":"date"}],"repeat":"1","cro
ntab":"","once":false,"onceDelay":0.1,"topic":"","payload":"true","payloadType":"bool","x":180,"y":200,"wires":[["e560b21b.98356"]]},{"id
":"e560b21b.98356","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3
x64.Application.HMI.UR1_STOPED","datatype":"Boolean","value":"","name":"","x":420,"y":200,"wires":[["5ade2c.8e6f41d4"]]},{"id":"5ade
2c.8e6f41d4","type":"OpcUa-
Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit"
:"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":200,"wires":[["e1
968009.77a6c"]]},{"id":"e1968009.77a6c","type":"change","z":"8fe6e59f.0879d8","name":"bool to
int","rules":[{"t":"change","p":"payload","pt":"msg","from":"true","fromt":"bool","to":"1","tot":"num"},{"t":"change","p":"payload","pt":"
msg","from":"false","fromt":"bool","to":"0","tot":"num"}],"action":"","property":"","from":"","to":"","reg":false,"x":840,"y":200,"wires":[["
d37dfa65.902b38"]]},{"id":"d37dfa65.902b38","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var
json={\"UR1_STOPED\":msg.payload};\nreturn
{\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":1000,"y":200,"wires":[["773cfb97.ec0324"]]},{"id":"b6225d09.dc59
e","type":"inject","z":"8fe6e59f.0879d8","name":"UR2_RUNNING","props":[{"p":"payload"},{"p":"topic","v":"","vt":"date"}],"repeat":"1","c
rontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"true","payloadType":"bool","x":190,"y":260,"wires":[["ca698f5e.a1cbd"]]},{"i
d":"ca698f5e.a1cbd","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3
x64.Application.HMI.UR2_RUNNING","datatype":"Boolean","value":"","name":"","x":420,"y":260,"wires":[["f1a9a91b.d2e108"]]},{"id":"f1a
9a91b.d2e108","type":"OpcUa-
Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit"
:"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":260,"wires":[["a8
b47727.3fe4f8"]]},{"id":"a8b47727.3fe4f8","type":"change","z":"8fe6e59f.0879d8","name":"bool to
int","rules":[{"t":"change","p":"payload","pt":"msg","from":"true","fromt":"bool","to":"1","tot":"num"},{"t":"change","p":"payload","pt":"
msg","from":"false","fromt":"bool","to":"0","tot":"num"}],"action":"","property":"","from":"","to":"","reg":false,"x":840,"y":260,"wires":[["
aab793e7.aa0b6"]]},{"id":"aab793e7.aa0b6","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var
json={\"UR2_RUNNING\":msg.payload};\nreturn
{\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":1000,"y":260,"wires":[["773cfb97.ec0324"]]},{"id":"75a3c0ef.caa64
","type":"inject","z":"8fe6e59f.0879d8","name":"UR2_IDLE","props":[{"p":"payload"},{"p":"topic","v":"","vt":"date"}],"repeat":"1","crontab
":"","once":false,"onceDelay":0.1,"topic":"","payload":"true","payloadType":"bool","x":170,"y":320,"wires":[["4f902721.b8a748"]]},{"id":"4
f902721.b8a748","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3

x64.Application.HMI.UR2_IDLE","datatype":"Boolean","value":"","name":"","x":420,"y":320,"wires":[["fdcdb18f.fec8e"]]},{"id":"fdcdb18f.fec8e","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":320,"wires":[["e3f386a3.789ea8"]]},{"id":"e3f386a3.789ea8","type":"change","z":"8fe6e59f.0879d8","name":"bool to int","rules":[{"t":"change","p":"payload","pt":"msg","from":"true","fromt":"bool","to":"1","tot":"num"},{"t":"change","p":"payload","pt":"msg","from":"false","fromt":"bool","to":"0","tot":"num"}],"action":"","property":"","from":"","to":"","reg":false,"x":840,"y":320,"wires":[["327f0859.4fe198"]]},{"id":"327f0859.4fe198","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var json={\"UR2_IDLE\":msg.payload};\nreturn {\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":1000,"y":320,"wires":[["773cfb97.ec0324"]]},{"id":"cd27f239.89002","type":"inject","z":"8fe6e59f.0879d8","name":"UR2_STOPED","props":[{"p":"payload"},{"p":"topic","v":"","vt":"date"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"true","payloadType":"bool","x":180,"y":380,"wires":[["e016b72d.65d8c8"]]},{"id":"e016b72d.65d8c8","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.HMI.UR2_STOPED","datatype":"Boolean","value":"","name":"","x":420,"y":380,"wires":[["79f2ea65.345f84"]]},{"id":"79f2ea65.345f84","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":380,"wires":[["2312bb76.ca42e4"]]},{"id":"2312bb76.ca42e4","type":"change","z":"8fe6e59f.0879d8","name":"bool to int","rules":[{"t":"change","p":"payload","pt":"msg","from":"true","fromt":"bool","to":"1","tot":"num"},{"t":"change","p":"payload","pt":"msg","from":"false","fromt":"bool","to":"0","tot":"num"}],"action":"","property":"","from":"","to":"","reg":false,"x":840,"y":380,"wires":[["19e1f286.e93bdd"]]},{"id":"19e1f286.e93bdd","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var json={\"UR2_STOPED\":msg.payload};\nreturn {\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":1000,"y":380,"wires":[["773cfb97.ec0324"]]},{"id":"cce72dec.5c0b9","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_EJE1","props":[{"p":"payload"},{"p":"topic","v":"","vt":"num"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"num","x":170,"y":460,"wires":[["682e517a.e99af"]]},{"id":"682e517a.e99af","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.CALC_POS_AXIS_UR1.UR1_EJE1","datatype":"Int32","value":"","name":"","x":420,"y":460,"wires":[["3ba3433f.99385c"]]},{"id":"3ba3433f.99385c","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":460,"wires":[["eedf2f67.23eae"]]},{"id":"830cf9a1.807758","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_EJE2","props":[{"p":"payload"},{"p":"topic","v":"","vt":"num"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"num","x":170,"y":520,"wires":[["b7cfac1f.08c63"]]},{"id":"b7cfac1f.08c63","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.CALC_POS_AXIS_UR1.UR1_EJE2","datatype":"Int32","value":"","name":"","x":420,"y":520,"wires":[["13ee8e4f.4794e2"]]},{"id":"13ee8e4f.4794e2","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":520,"wires":[["7f398610.89c8e8"]]},{"id":"5c4ffbf1.80c764","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_EJE3","props":[{"p":"payload"},{"p":"topic","v":"","vt":"num"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"num","x":170,"y":580,"wires":[["ef5cf942.0bf978"]]},{"id":"ef5cf942.0bf978","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.CALC_POS_AXIS_UR1.UR1_EJE3","datatype":"Int32","value":"","name":"","x":420,"y":580,"wires":[["8b5c2471.13a2b8"]]},{"id":"8b5c2471.13a2b8","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":580,"wires":[["50a01383.3d441c"]]},{"id":"f7cc3b27.a1dc28","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_EJE4","props":[{"p":"payload"},{"p":"topic","v":"","vt":"num"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"num","x":170,"y":640,"wires":[["a0cd85b9.1a7ba8"]]},{"id":"a0cd85b9.1a7ba8","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.CALC_POS_AXIS_UR1.UR1_EJE4","datatype":"Int32","value":"","name":"","x":420,"y":640,"wires":[["3b0558d8.ce9678"]]},{"id":"3b0558d8.ce9678","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":640,"wires":[["eec8a182.b37ef"]]},{"id":"aeeb706a.98d65","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_EJE5","props":[{"p":"payload"},{"p":"topic","v":"","vt":"num"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"num","x":170,"y":700,"wires":[["fb49badd.5467a8"]]},{"id":"fb49badd.5467a8","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.CALC_POS_AXIS_UR1.UR1_EJE5","datatype":"Int32","value":"","name":"","x":420,"y":700,"wires":[["e89e797a.9093c8"]]},

{"id":"e89e797a.9093c8","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":700,"wires":[["34256b15.66e6c4"]]},{"id":"a9221976.4283e8","type":"inject","z":"8fe6e59f.0879d8","name":"UR1_EJE6","props":[{"p":"payload"},{"p":"topic","v":"","vt":"num"}],"repeat":"1","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"num","x":170,"y":760,"wires":[["c4b4770a.4bbb68"]]},{"id":"c4b4770a.4bbb68","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.CALC_POS_AXIS_UR1.UR1_EJE6","datatype":"Int32","value":"","name":"","x":420,"y":760,"wires":[["99e213e9.4eb4c"]]},{"id":"99e213e9.4eb4c","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"read","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":660,"y":760,"wires":[["c8f6d424.6df0b8"]]},{"id":"eedf2f67.23eae","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var json={\"UR1_EJE1\":msg.payload};\nreturn {\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":840,"y":460,"wires":[["8c6734e1.dd83d8"]]},{"id":"7f398610.89c8e8","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var json={\"UR1_EJE2\":msg.payload};\nreturn {\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":840,"y":520,"wires":[["8c6734e1.dd83d8"]]},{"id":"50a01383.3d441c","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var json={\"UR1_EJE3\":msg.payload};\nreturn {\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":840,"y":580,"wires":[["8c6734e1.dd83d8"]]},{"id":"eec8a182.b37ef","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var json={\"UR1_EJE4\":msg.payload};\nreturn {\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":840,"y":640,"wires":[["8c6734e1.dd83d8"]]},{"id":"34256b15.66e6c4","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var json={\"UR1_EJE5\":msg.payload};\nreturn {\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":840,"y":700,"wires":[["8c6734e1.dd83d8"]]},{"id":"c8f6d424.6df0b8","type":"function","z":"8fe6e59f.0879d8","name":"","func":"var json={\"UR1_EJE6\":msg.payload};\nreturn {\"payload\":json};","outputs":1,"noerr":0,"initialize":"","finalize":"","x":840,"y":760,"wires":[["8c6734e1.dd83d8"]]},{"id":"8c6734e1.dd83d8","type":"ubidots_out","z":"8fe6e59f.0879d8","name":"UR1_EJES","token":"BBFF-3T9dR3rjU6cIQQug0RjfILqZeTnmut","label_device":"","device_label":"TFG","tier":"educational","tls_checkbox":true,"x":1230,"y":580,"wires":[]},{"id":"a8c049d8.fa5698","type":"ubidots_in","z":"8fe6e59f.0879d8","tier":"business","name":"POS_DELIVERY_1","token":"BBFF-3T9dR3rjU6cIQQug0RjfILqZeTnmut","device_label":"TFG","tls_checkbox_in":true,"custom_topic_checkbox":false,"label_variable_1":"POS_1","label_variable_2":"","label_variable_3":"","label_variable_4":"","label_variable_5":"","label_variable_6":"","label_variable_7":"","label_variable_8":"","label_variable_9":"","label_variable_10":"","checkbox_variable_1_last_value":true,"checkbox_variable_2_last_value":true,"checkbox_variable_3_last_value":true,"checkbox_variable_4_last_value":true,"checkbox_variable_5_last_value":true,"checkbox_variable_6_last_value":true,"checkbox_variable_7_last_value":true,"checkbox_variable_8_last_value":true,"checkbox_variable_9_last_value":true,"checkbox_variable_10_last_value":true,"x":190,"y":840,"wires":[["6eacc894.a8a2d8"]]},{"id":"6eacc894.a8a2d8","type":"change","z":"8fe6e59f.0879d8","name":"","rules":[{"t":"move","p":"payload.pos_1.value","pt":"msg","to":"payload","tot":"msg"}],"action":"","property":"","from":"","to":"","reg":false,"x":470,"y":840,"wires":[["8b8b97cd.2840d8"]]},{"id":"37e8f8b.215d408","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"write","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":940,"y":840,"wires":[[]]},{"id":"8b8b97cd.2840d8","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.TFG.UBIDOTS_POS_1","datatype":"Boolean","value":"","name":"","x":740,"y":840,"wires":[["37e8f8b.215d408"]]},{"id":"c83ae281.d7425","type":"ubidots_in","z":"8fe6e59f.0879d8","tier":"business","name":"POS_DELIVERY_2","token":"BBFF-3T9dR3rjU6cIQQug0RjfILqZeTnmut","device_label":"TFG","tls_checkbox_in":true,"custom_topic_checkbox":false,"label_variable_1":"POS_2","label_variable_2":"","label_variable_3":"","label_variable_4":"","label_variable_5":"","label_variable_6":"","label_variable_7":"","label_variable_8":"","label_variable_9":"","label_variable_10":"","checkbox_variable_1_last_value":true,"checkbox_variable_2_last_value":true,"checkbox_variable_3_last_value":true,"checkbox_variable_4_last_value":true,"checkbox_variable_5_last_value":true,"checkbox_variable_6_last_value":true,"checkbox_variable_7_last_value":true,"checkbox_variable_8_last_value":true,"checkbox_variable_9_last_value":true,"checkbox_variable_10_last_value":true,"x":190,"y":900,"wires":[["9b7639b1.f363e8"]]},{"id":"9b7639b1.f363e8","type":"change","z":"8fe6e59f.0879d8","name":"","rules":[{"t":"move","p":"payload.pos_2.value","pt":"msg","to":"payload","tot":"msg"}],"action":"","property":"","from":"","to":"","reg":false,"x":470,"y":900,"wires":[["2a0db5e7.cac7fa"]]},{"id":"42018a23.58ef94","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"write","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":940,"y":900,"wires":[[]]},{"id":"2a0db5e7.cac7fa","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.TFG.UBIDOTS_POS_2","datatype":"Boolean","value":"","name":"","x":740,"y":900,"wires":[["42018a23.58ef94"]]},{"id":"e952dd02.d45ab","type":"ubidots_in","z":"8fe6e59f.0879d8","tier":"business","name":"POS_DELIVERY_3","token":"BBFF-3T9dR3rjU6cIQQug0RjfILqZeTnmut","device_label":"TFG","tls_checkbox_in":true,"custom_topic_checkbox":false,"label_variable_1":"POS_3","label_variable_2":"","label_variable_3":"","label_variable_4":"","label_variable_5":"","label_variable_6":"","label_variable_7":"","label_variable_8":"","label_variable_9":"","label_variable_10":"","checkbox_variable_1_last_value":true,"checkbox_variable_2_last_value":true,"checkbox_variable_3_last_value":true,"checkbox_variable_4_last_value":true,"checkbox_variable_5_last_value":true,"checkbox_variable_6_last_value":true,"checkbox_variable_7_last_value":true,"checkbox_variable_8_last_value":true,"checkbox_variable_9_last_value":true,"checkbox_variable_10_last_value":true,"x":190,"y":960,"wires":[["92fbb2a3.b67bf"]]},{"id":"92fbb2a3.b67bf","type":"change","z":"8fe

6e59f.0879d8","name":"","rules":[{"t":"move","p":"payload.pos_3.value","pt":"msg","to":"payload","tot":"msg"}],"action":"","property":"","from":"","to":"","reg":false,"x":470,"y":960,"wires":[["1b291e82.241031"]]},{"id":"c94d1d65.ff514","type":"OpcUa-Client","z":"8fe6e59f.0879d8","endpoint":"818a4c46.d8255","action":"write","deadbandtype":"a","deadbandvalue":1,"time":10,"timeUnit":"s","certificate":"n","localfile":"","localkeyfile":"","securitymode":"None","securitypolicy":"None","name":"","x":940,"y":960,"wires":[[]]},{"id":"1b291e82.241031","type":"OpcUa-Item","z":"8fe6e59f.0879d8","item":"ns=4;s=|var|CODESYS Control Win V3 x64.Application.TFG.UBIDOTS_POS_3","datatype":"Boolean","value":"","name":"","x":740,"y":960,"wires":[["c94d1d65.ff514"]]},{"id":"b9a4f503.806778","type":"debug","z":"8fe6e59f.0879d8","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statusType":"auto","x":1240,"y":80,"wires":[]},{"id":"818a4c46.d8255","type":"OpcUa-Endpoint","endpoint":"opc.tcp://CPOHDELL5820:4840","secpol":"None","secmode":"None","login":false}]

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

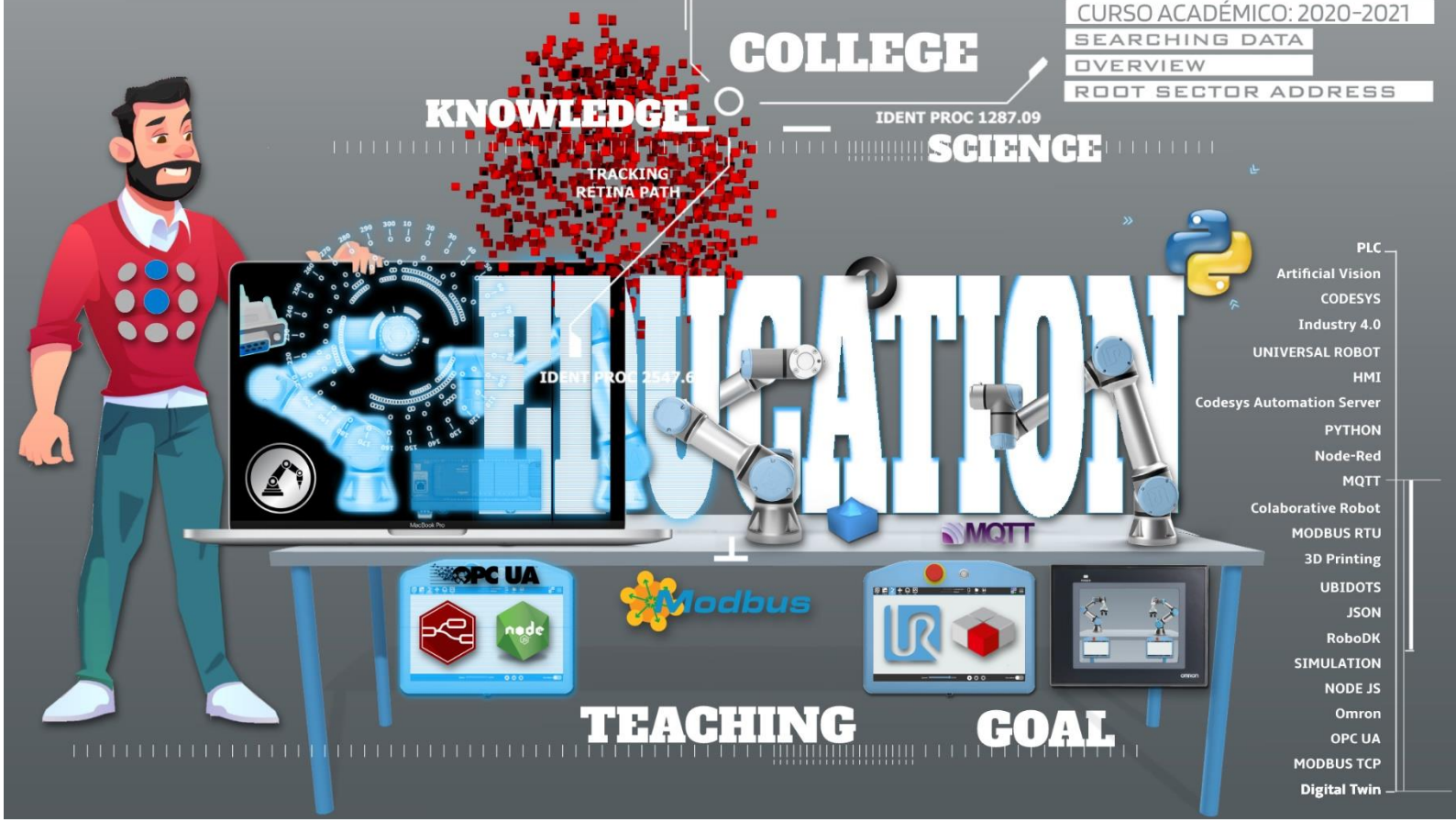
Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA

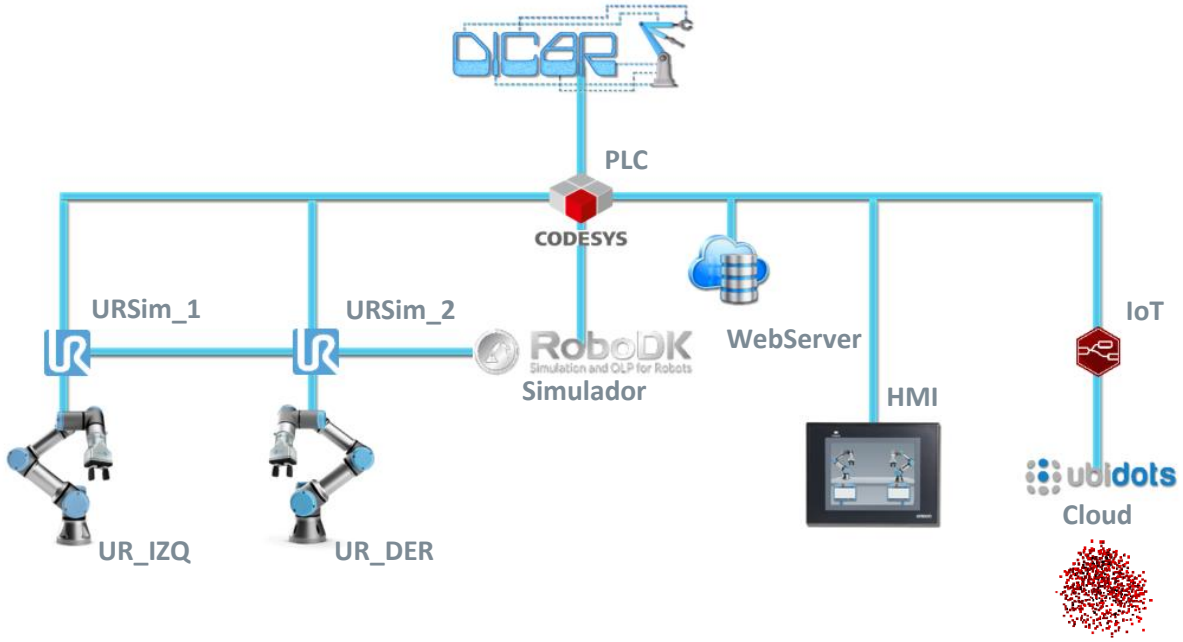
MANUAL DE USUARIO

Dayron Rodríguez Díaz



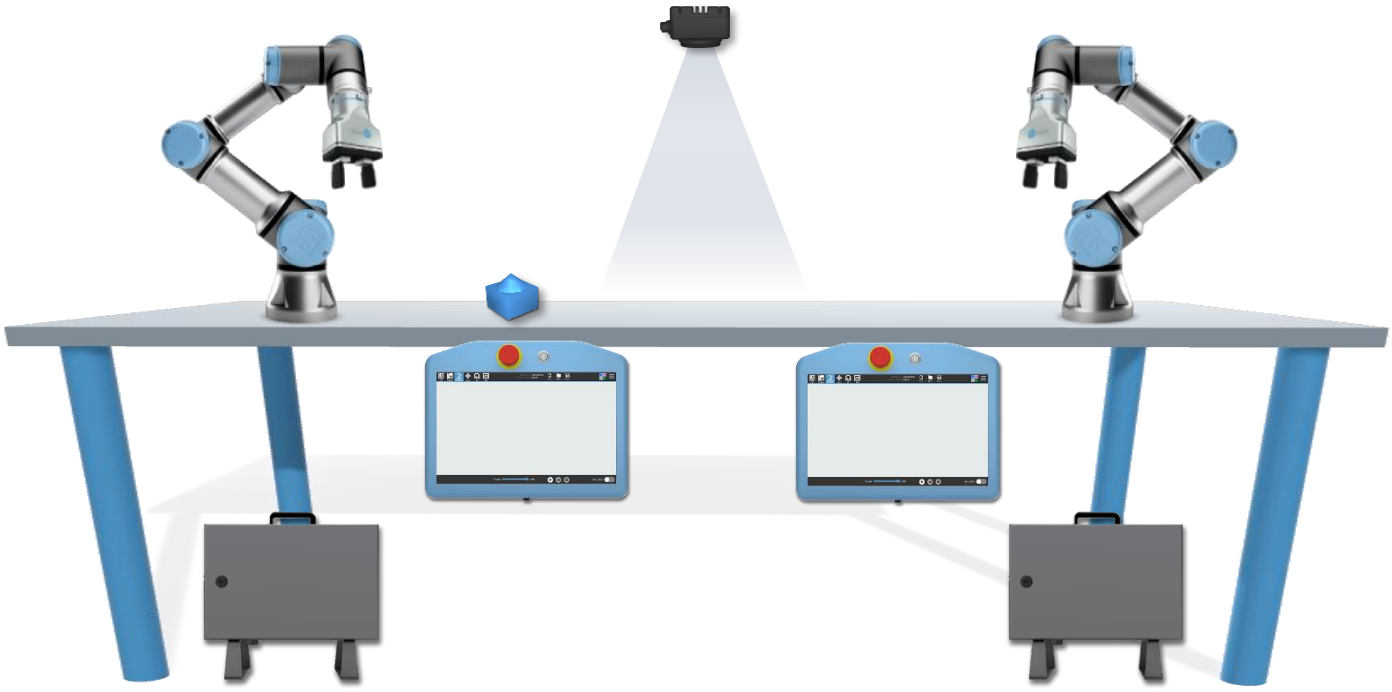
Con el objetivo de suministrar al usuario una guía detallada del uso de la plataforma diseñada en este proyecto, a continuación, se describe la secuencia necesaria para poner en funcionamiento todas las herramientas descritas en apartados anteriores. El orden en el que encontrará los pasos numerados corresponde con la estructura jerárquica de la plataforma ***DICAR***. (Figura 1).

Figura 1. Organigrama que da solución al Diseño e Implementación de una Celda Automatizada con Robótica Colaborativa.



El ciclo de trabajo que podrá visualizar, una vez terminada la configuración aquí descrita, corresponde a un *Digital Twin* de una celda con dos robots colaborativos *UR3e* que intercambian una pieza entre ellos empleando visión artificial para calcular las coordenadas de la posición de entrega. (Figura 2).

Figura 2. Digital Twin de la celda física.



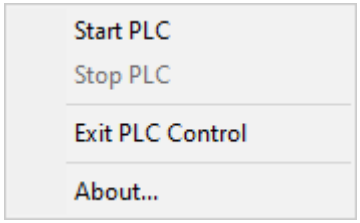
El presente documento le ayudará a utilizar la plataforma que se ha creado en la investigación de este proyecto, todos los programas están disponibles junto al informe del TFG excepto los discos duros de las máquinas virtuales de *URSim*; para ello existe una secuencia de pasos que se enumeran a continuación:

1.1. PUESTA EN MARCHA PLC

Al instalar *CODESYS* se activan en la barra de tareas de *Windows* dos servicios, uno para comunicar el *PLC* con la *Nube* y otro para emular un *softPLC*, este último es el encargado de gobernar toda la celda.

1- Pulse en la barra de tareas de *Windows* sobre "***Mostrar iconos ocultos***" (Figura 3), el simulador de *CODESYS* se encuentra apagado, debe presionar *clic derecho* y seleccionar *Start PLC:*

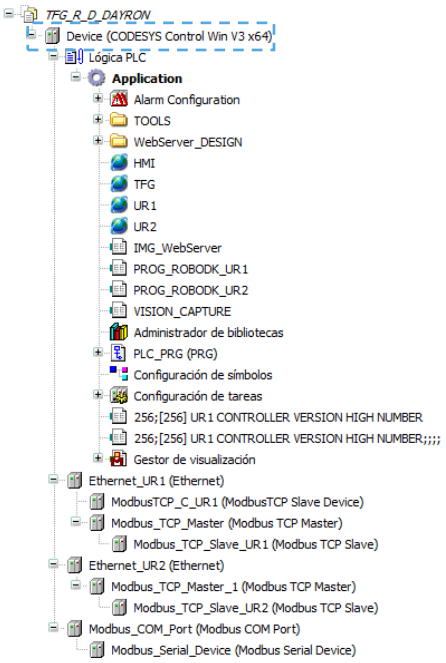
Figura 3. Puesta en marcha del PLC virtual de CODESYS.

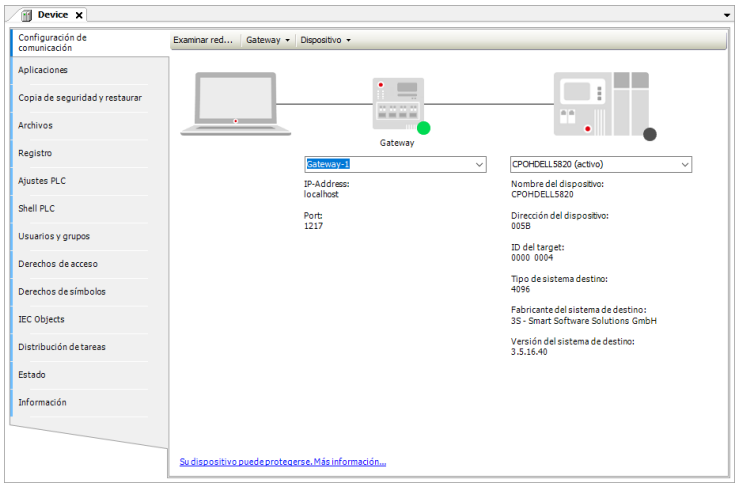
CODESYS Control Win (**OFF**)  

 *CODESYS Control Win* (**ON**)

2- Abra el programa del *PLC* en *CODESYS* y presione *doble clic* sobre "***Device***" en el árbol del proyecto para mostrar la pestaña donde conectará la programación con el *softPLC*. Posteriormente pulse "***Examinar red***" en la pestaña *Configuración de comunicación* (Figura 4)*:*

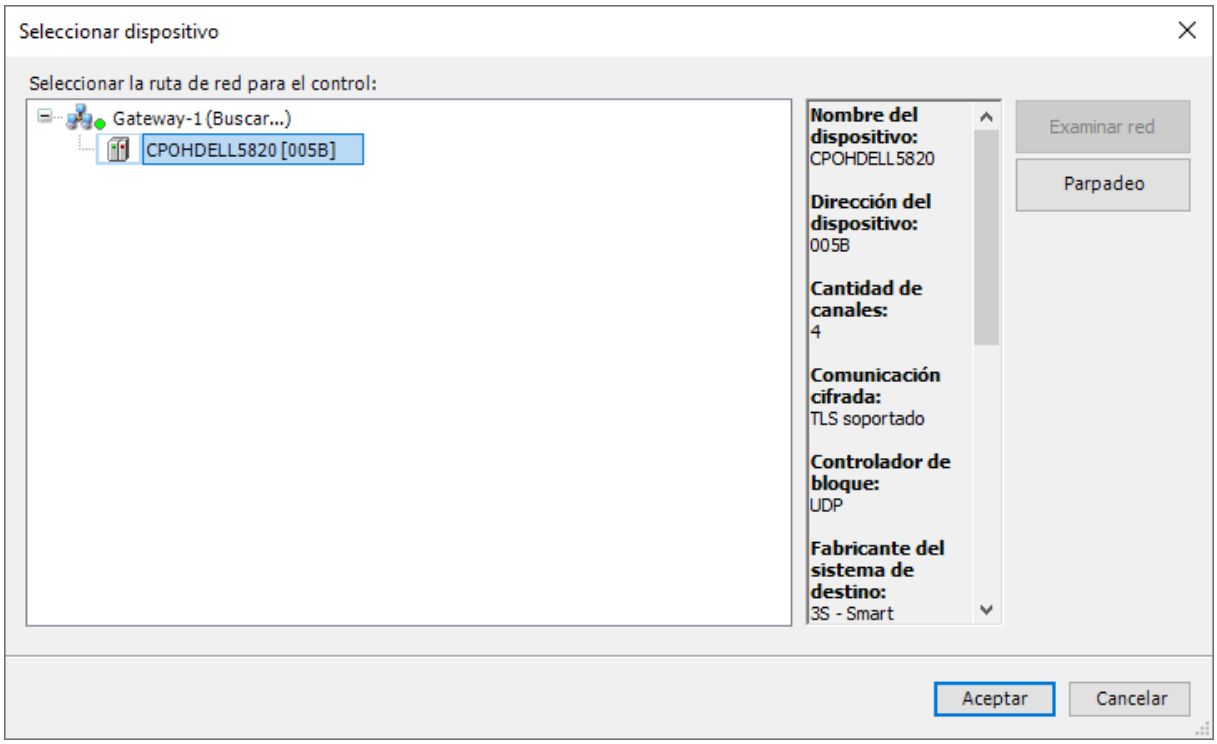
Figura 4. Ventana para examinar los dispositivos detectados por CODESYS.





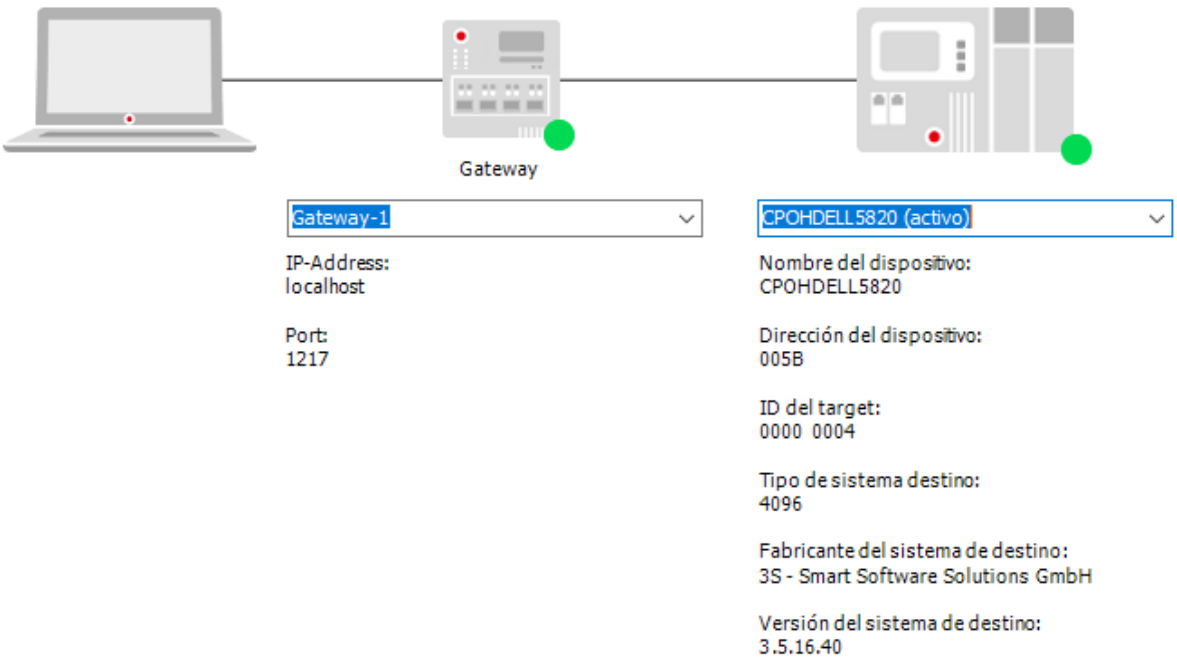
3- La ventana de selección de los dispositivos detectados muestra el *softPLC* que está activo en el *PC*, **presione sobre el dispositivo** y finalmente sobre el botón "***Aceptar"*** (Figura 5)*:*

Figura 5. Enlace del programa con el softPLC de CODESYS.



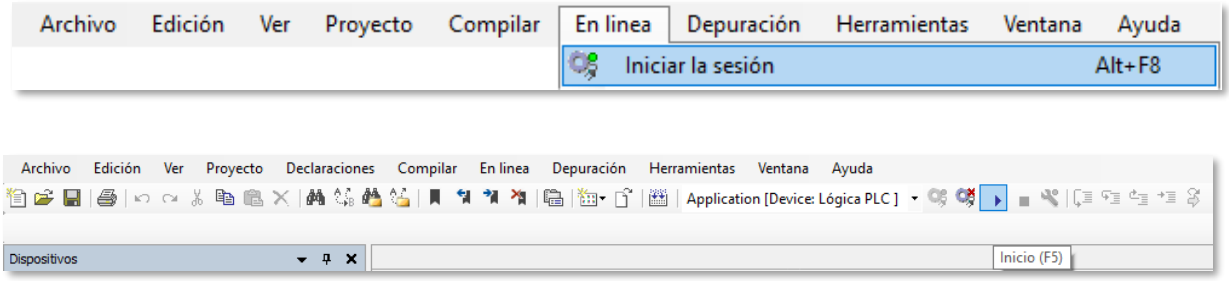
Cuando la conexión se realiza de forma correcta, CODESYS muestra el enlace con un led verde asociado al *softPLC* (Figura 6)*:*

Figura 6. Comprobación de una conexión establecida con éxito.



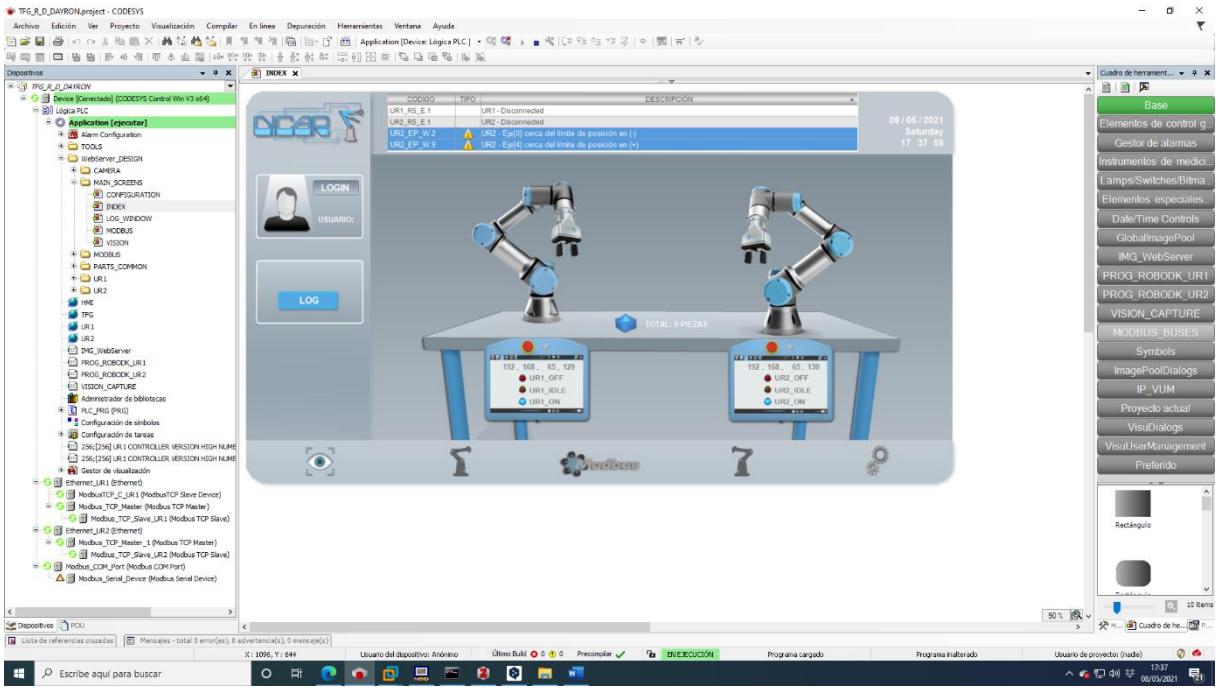
4- Con el dispositivo enlazado al software debe iniciar sesión para visualizar el *WebServer* dentro de la misma aplicación, para ello, diríjase al menú principal opción "***En línea"*** y seleccione "***Iniciar la sesión"*** (Figura 7.a), posteriormente presione "***Ejecutar Programa"*** (F5) (Figura 7.b)*:*

Figura 7. Inicio de sesión desde CODESYS para visualizar el WebServer desde el mismo programa.



5- En el árbol del proyecto navegue hasta la carpeta "***Main Screens"*** accesible a través de la siguiente dirección "***Device / Lógica PLC / Application / WebServer_DESIGN / Main_SCREENS"*** y seleccione "***INDEX"*** (Figura 8), de esta forma tiene la opción de visualizar igualmente al *WebServer* embebido en el *PLC.*

Figura 8. Acceso al WebServer del proyecto desde el IDE de CODESYS.

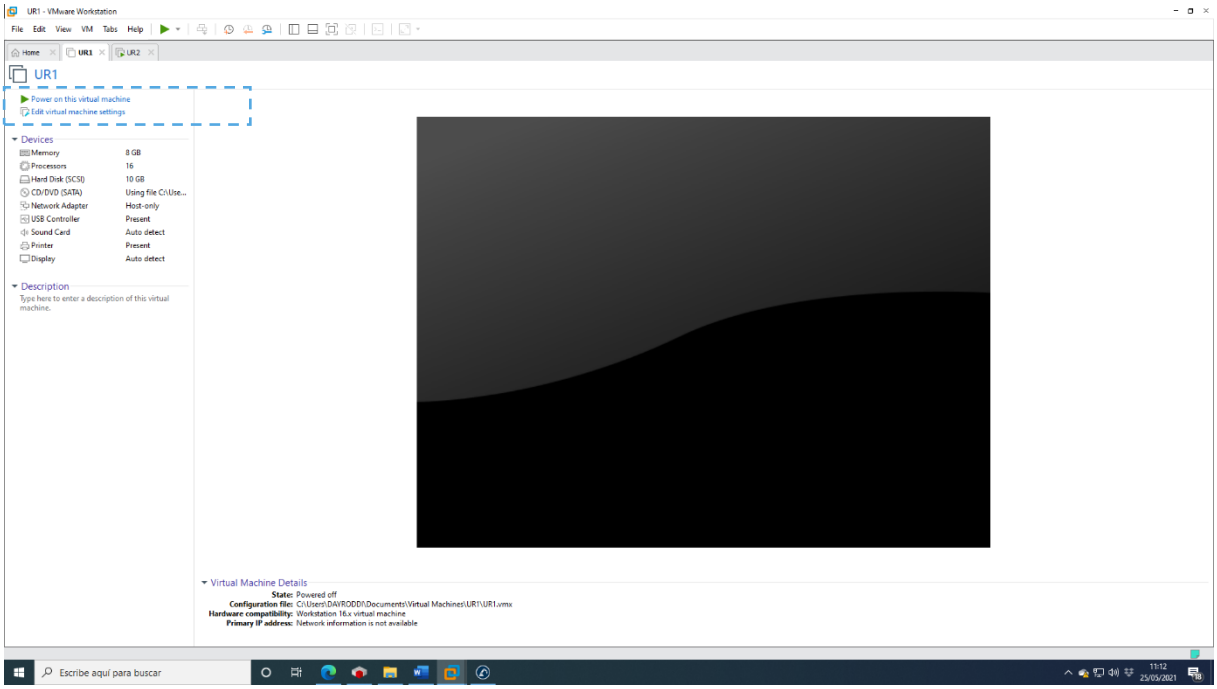


Nota: Accediendo a la carpeta *TOOLS*, tiene a su disposición todos los programas que controlan la secuencia de trabajo de la celda creada, de esta forma puede consultar el valor de cualquiera de las variables contenidas dentro del *POU* que se esté visualizando.

1.2. PUESTA EN MARCHA URSim

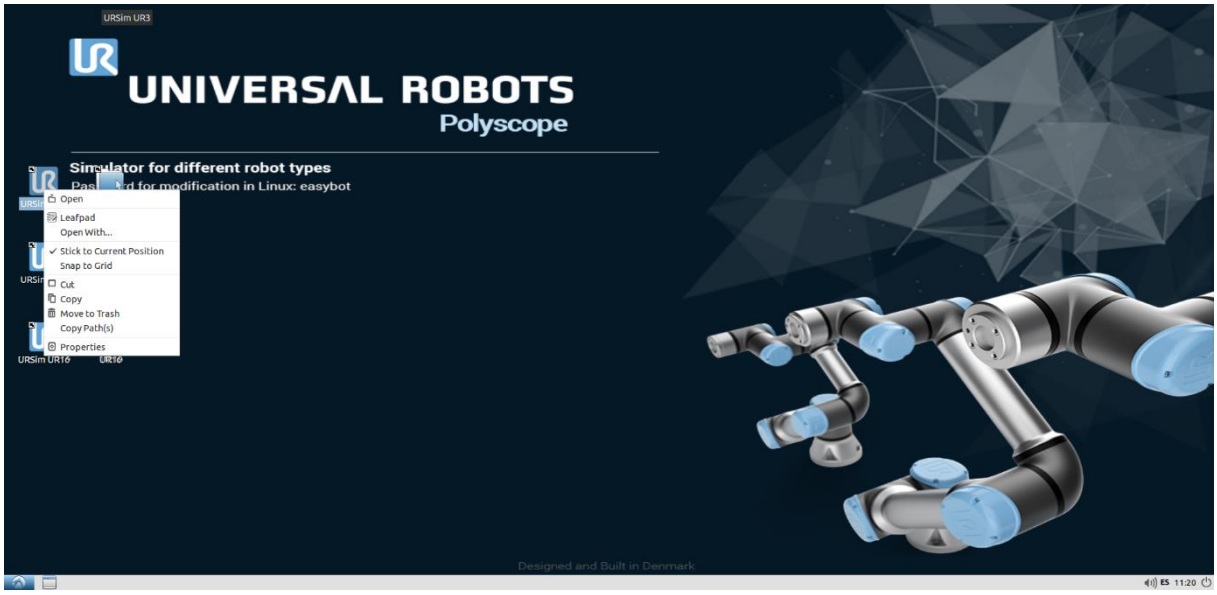
1- Ejecute *VMware Workstation* y cargue las dos máquinas virtuales correspondientes a los modelos *UR1* y *UR2*, posteriormente presione "***Power on this virtual machine***" en cada máquina (Figura 9):

Figura 9. Puesta en funcionamiento de las máquina virtuales de URSim.



2- En cada máquina debe abrir el *IDE* correspondiente al modelo del robot "***URSim UR3***", bien con un doble clic o clic derecho "***Open"*** (Figura 10):

Figura 10. Inicio del simulador para el modelo de robot UR3 en URSim.

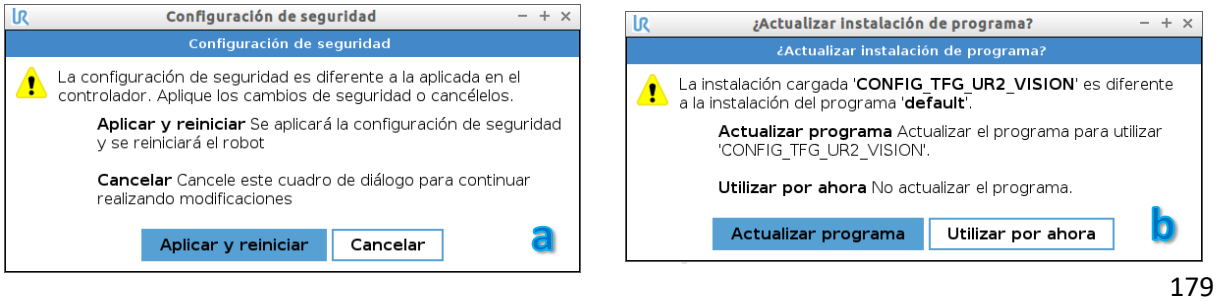


3- Por defecto, *URSim* se activa en el modo de funcionamiento "***Automático"*** (Figura 11.a), lo cual impide cargar la estación configurada por el autor (se entrega junto al presente informe, una copia del archivo de la estación y el programa para cada robot), para llevar a cabo la carga del fichero de configuración de la estación debe presionar sobre el icono con apariencia de una *Espiral* (🌀) y seleccionar en la ventana emergente la opción "***Manual"*** (Figura 11.b), debe ingresar la contraseña para activar dicho modo (*Pass:* ***Dayron***) (Figura 11.c), posteriormente el icono con la carpeta se activa (Figura 11.d) y es posible cargar la estación y el programa, utilice la ventana de navegación para acceder a los ficheros ***CONFIG_TFG_UR(1/2)_VISION.installation*** y ***UR(1/2)_POST_RoboDK.*urp** que corresponden a la estación y el programa de cada robot respectivamente.

Figura 11. Secuencia para cargar los ficheros de configuración de la estación y el programa para cada unidad.

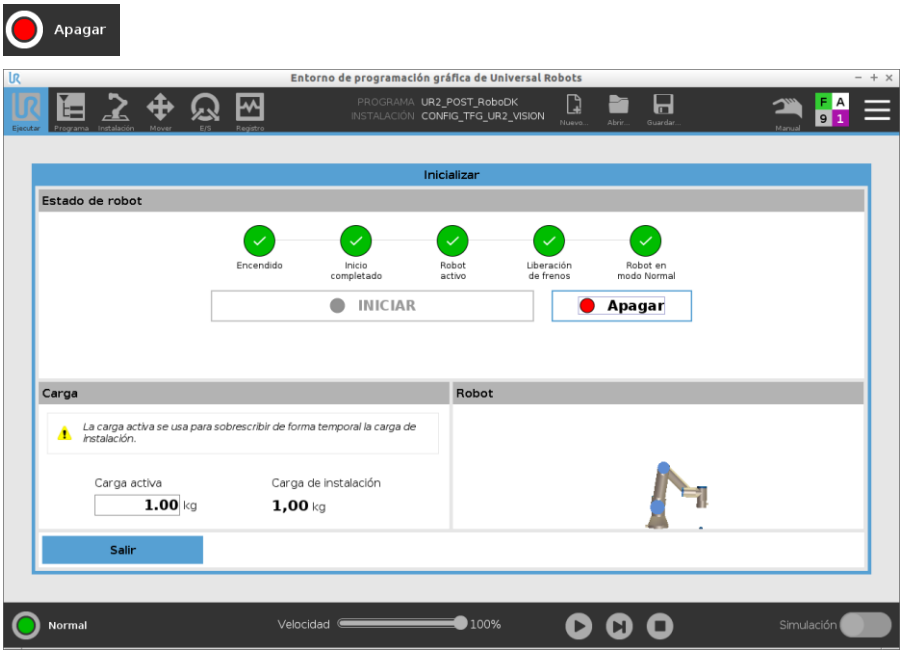
4- Al seleccionar el fichero de la estación aparece una ventana emergente indicando que la configuración de seguridad es diferente a la aplicada en el controlador, presione "***Aplicar y reiniciar"*** (Figura 12.a), posteriormente una nueva ventana emergente notifica al usuario sobre la diferencia ente la instalación cargada y la que se encuentra instalada en el controlador, deber ser actualizada, para ello presione "***Actualizar programa"*** (Figura 12.b).

Figura 12. Ventanas emergentes de configuración de seguridad y actualización de la instalación.



5- El robot se encuentra apagado, para encenderlo, presione sobre el botón (● *Apagar*) que se encuentra en la esquina inferior izquierda de la ventana de *URSim* (Figura 13.a), posteriormente presione en el apartado *Estado de robot* sobre el botón "***ENCENDER"*** y luego "***INICIAR"*** (Figura 13.b), de esta forma cada robot se encuentra preparado para comunicarse con el *softPLC* de *CODESYS* y los robots programados en *RoboDK.*

Figura 13. Secuencia de encendido del brazo robótico UR.



6- En la pestaña "***Mover"*** del menú principal de la aplicación, se representa la posición actual del robot, en ella es posible modificar cada eje por separado, sin embargo, se ha preprogramado la posición *HOME* para cada robot de la estación, presione sobre el botón "***Inicio"*** (Figura 14) y en la nueva ventana mantener presionado el botón "***Mover el robot a: Nueva posición"*** hasta que la posición articular del robot coincida con la mostrada sobreimpresa (Figura 15).

*Figura 14. Menú **Mover** donde se representa la posición articular del robot y se establece el movimiento a **HOME**.*

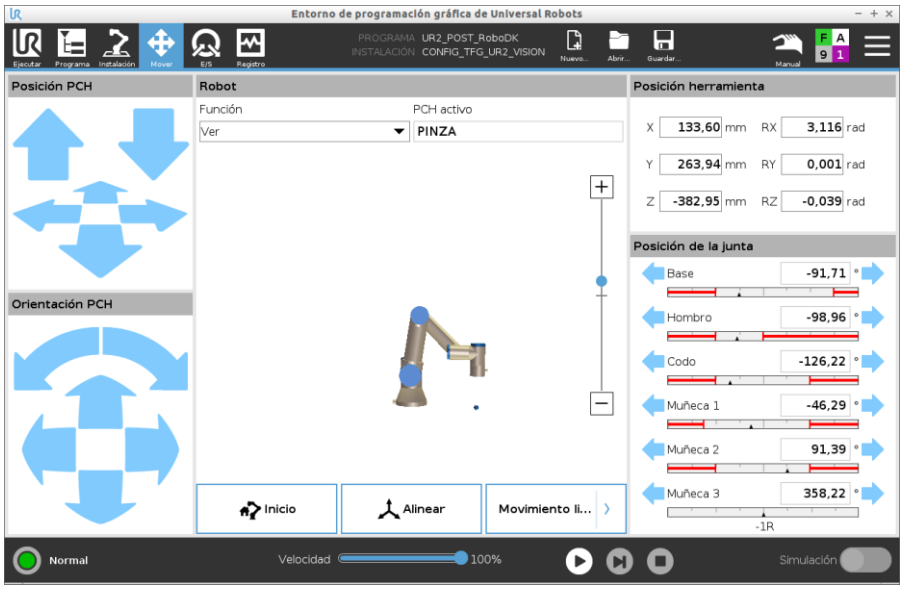
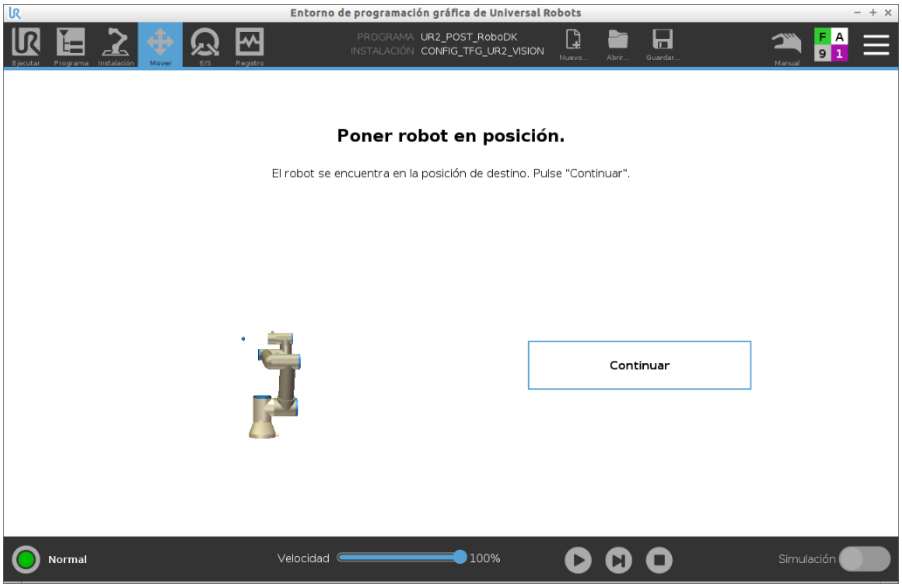


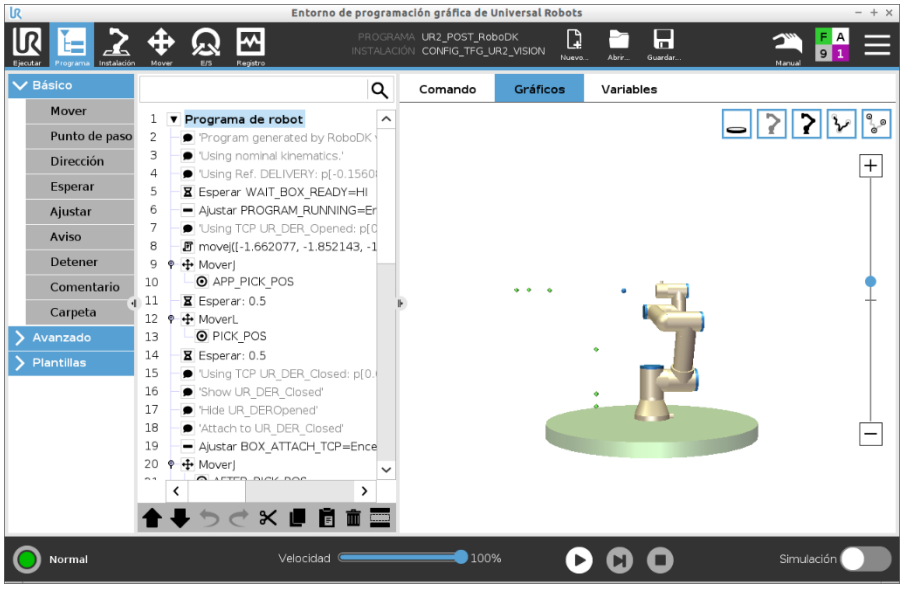
*Figura 15. Robot UR2 en **HOME** y rotado para que coincida con la situación de su homólogo en RoboDK.*



Una vez *UR1* o *UR2* se encuentren en *HOME POS*, el usuario puede rotar la vista para que coincida la representación gráfica del modelo con la situación de los robots en *RoboDK* quedando enfrentados físicamente, solo es necesario llevar a cabo este proceso para *UR2*.

7- Por último, en el menú "***Programa***" de la aplicación, se pueden observar las líneas de código cargadas en la estación y que fueron generadas en *RoboDK* empleando el post-procesador (Figura 16), en la misma ventana se tiene acceso a un apartado gráfico para visualizar en tiempo real la posición del robot en toda la secuencia de trabajo.

*Figura 16. Ventana **Programa** donde se muestra el código y la posición gráfica del robot.*



Nota: Todos los pasos descritos se deben realizar por duplicado, tanto para *UR1* como para *UR2*.

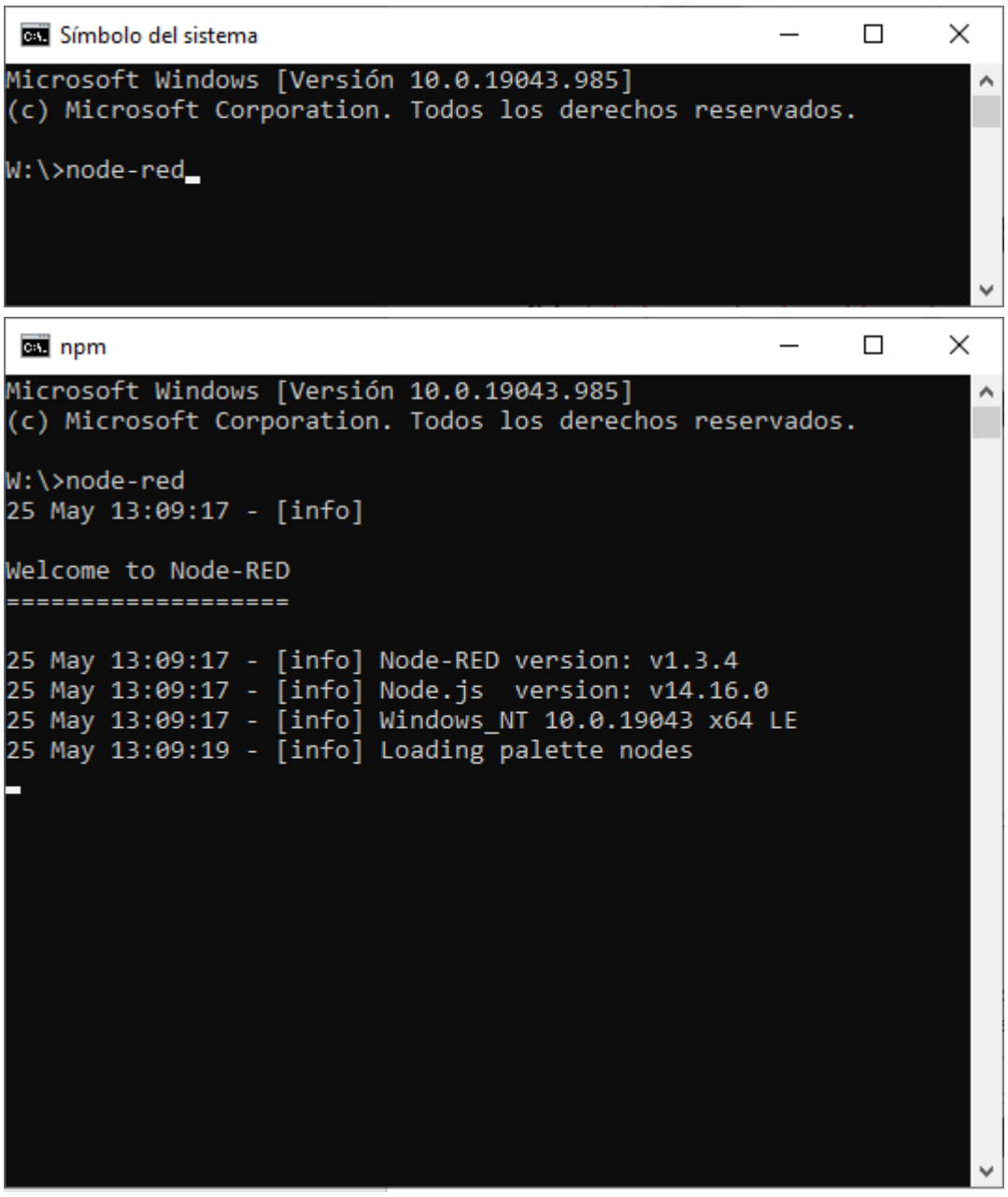
1.3. PUESTA EN MARCHA NODE-RED

NODE-RED es la herramienta que actúa como *driver* para la comunicación entre el *PLC* programado y el servicio de *cloud computing* alojado en *UBIDOTS*, es por ello por lo que previamente el *softPLC* de *CODESYS* debe estar en ejecución (Figura 17), para activar *NODE-RED,* siga la siguiente secuencia:

Figura 17. PLC virtual de CODESYS activo y ejecutándose en el PC.

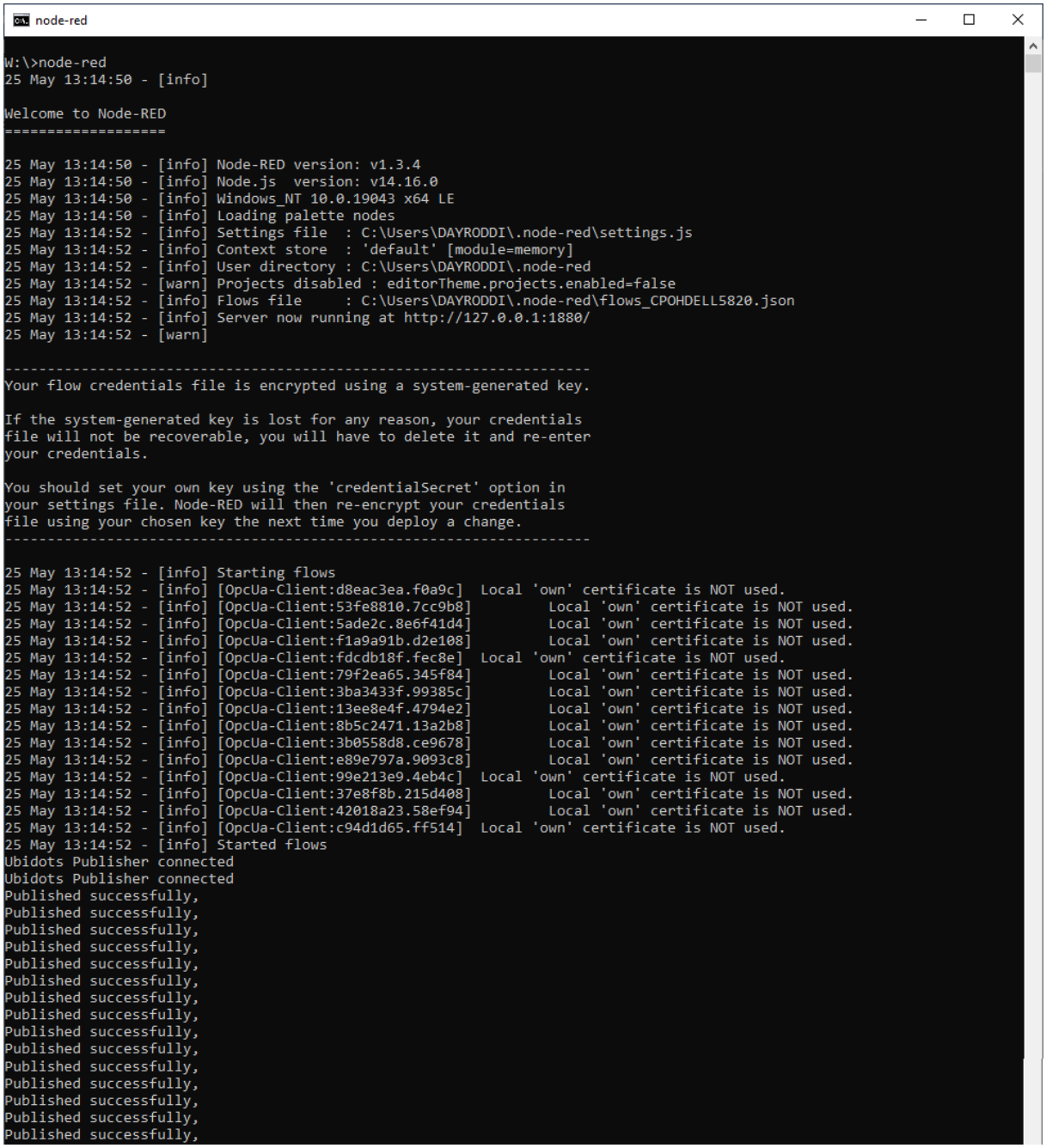
1- Abra un *Símbolo de sistema* o un *PowerShell* en *Windows*¸ en la ventana emergente teclee "***node-red"*** (Figura 18.a), presione "***ENTER"*** en el teclado para iniciar la ejecución del programa (Figura 18.b).

Figura 18. Ejecución del programa NODE-RED.



2- El programa comenzará una conexión cifrada con *UBIDOTS* y una vez establecida (Figura 19), la publicación de las variables se realiza según el período de *inyección* determinado previamente en el proceso de configuración y programación de *NODE-RED* previamente descrito en apartados anteriores. En el primer ciclo se publican las 15 variables que se intercambian entre *CODESYS* y *UBIDOTS* y cada un segundo se repite dicho proceso actualizándose la ventana.

Figura 19. Ejecución del programa NODE-RED donde se observa el primer ciclo de intercambio de las 15 variables.

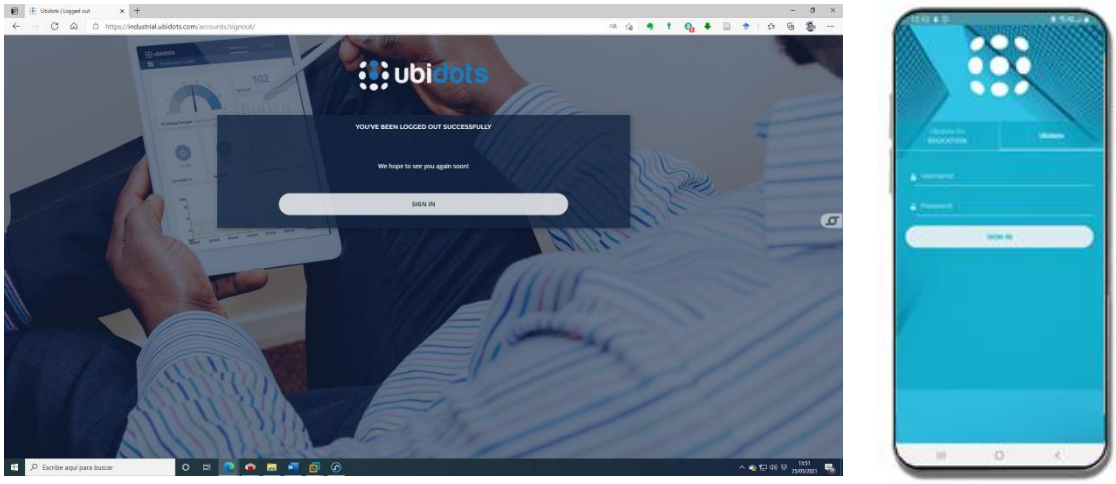


1.4. PUESTA EN MARCHA UBIDOTS

La plataforma tiene acceso desde un navegador de internet o bien desde su aplicación *UBIDOTS* disponible para *Android* y *Windows Phone* (no se encuentra disponible para ***IOS***).

1- Acceda a "Ubidots | Logged out" con el usuario y contraseña (Figura 20) relativos al ***TOKEN*** configurado en los *nodos* de *NODE-RED.*

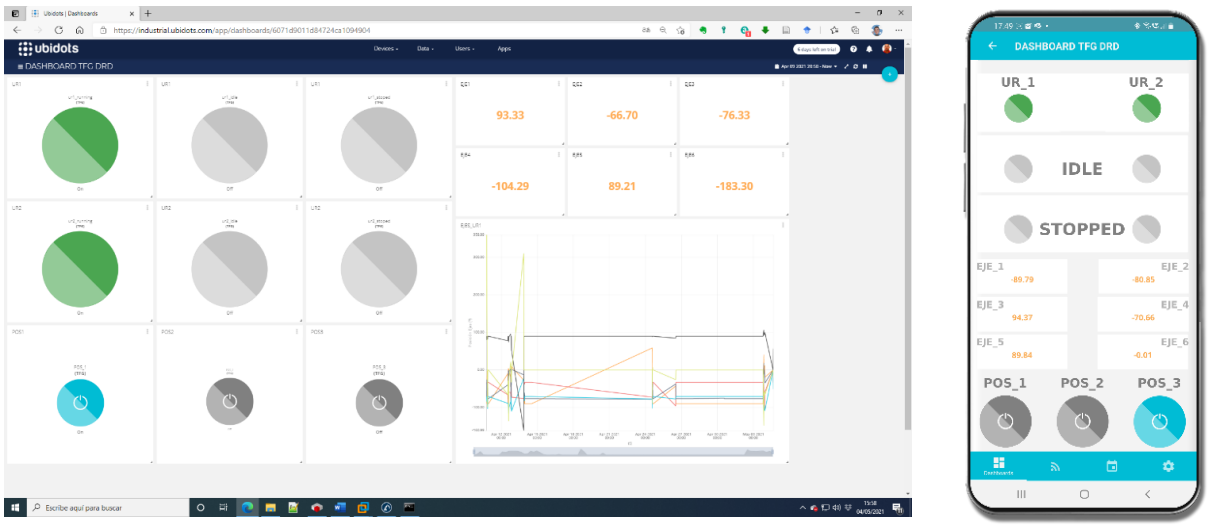
Figura 20. Proceso de autenticación en UBIDOTS desde una navegador Web o la aplicación para Android.



2- En la barra de menú superior seleccione "***Data"*** / "***Dashboard"*** para acceder al panel de control y visualización de las variables compartidas con el *PLC* de *CODESYS* (Figura 21), existen tres botones en la parte inferior izquierda para activar la posición en la cual *UR1* debe entregar la pieza denominados "***POS_1"***, "***POS_2"*** y "***POS_3"***.

IMPORTANTE: Solo debe estar seleccionado uno en cada instante, asegúrese de desactivar el botón que se encuentre en *ON* antes de encender otro, la programación en *NODE-RED* no realiza una depuración de esta tarea. ***UBIDOTS*** **ofrece únicamente un mes de período de prueba gratuito, usted debe elaborar su propio *Dashboard*.**

Figura 21. Visualización del Dashboard de UBIDOTS desde una navegador Web y desde la aplicación para Android.



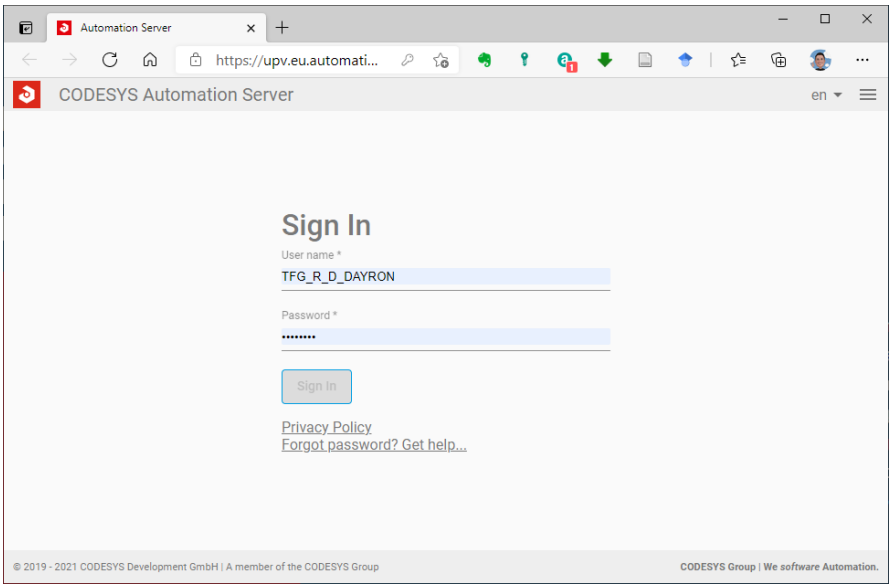
1.5. PUESTA EN MARCHA CODESYS AUTOMATION SERVER WEBSERVER

Acceder a los servicios que ofrece la plataforma de cloud computing de *CODESYS* es un proceso sencillo que se realiza a través de los siguientes pasos:

1- Abra un navegador y copie la dirección suministrada por *CODESYS Automation Server* al crear la cuenta tal y como se mostró en el apartado 5.9.1 en la página 137, cuando se visualice la ventana de autenticación ingrese el nombre de usuario y la contraseña (Figura 22):

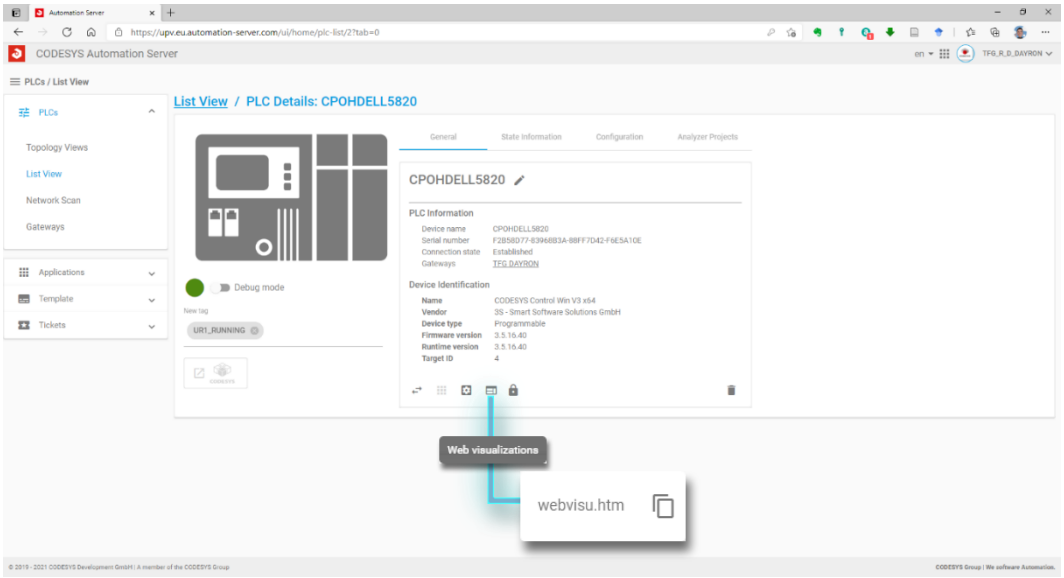
https://upv.eu.automation-server.com/

Figura 22. Autenticación en CODESYS Automation Server con las credenciales configuradas en el proceso de alta del servicio.



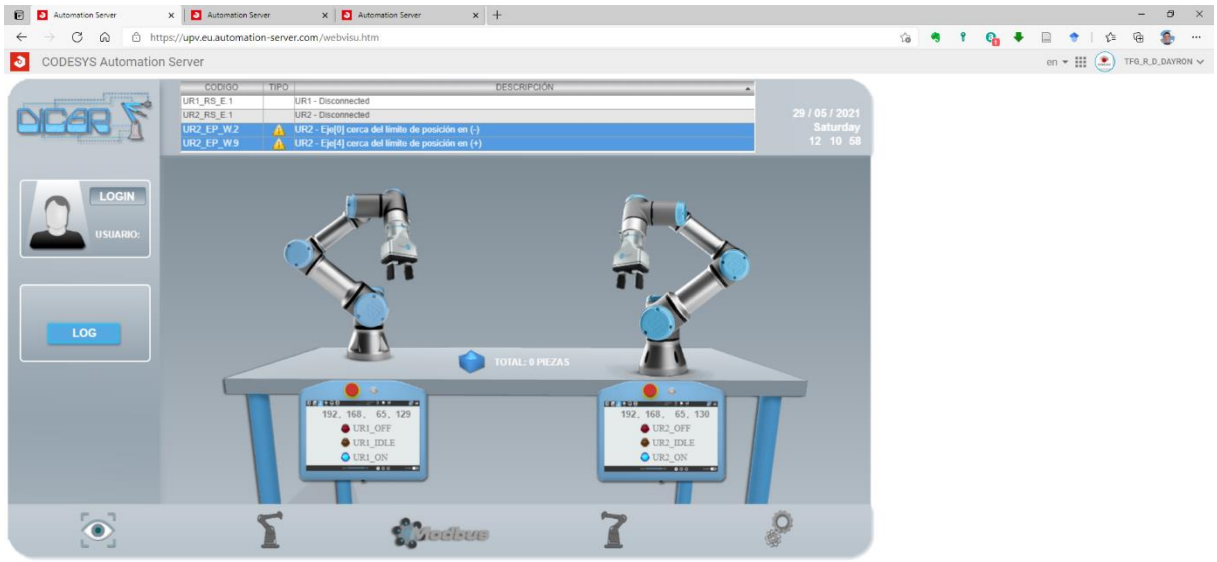
2- Acceda a "***List View***" / "***Web visualization***" / "***webvisu.htm***" (Figura 23) para mostrar el *WebServer* diseñado en *CODESYS* desde el servicio de cómputo en la *Nube:*

Figura 23. Acceso al WebServer desde CODESYS Automation Server.



3- El navegador suele tardar en responder, sobre todo en los primeros instantes al cargar el *WebServer*, espere unos 10 segundos aproximadamente y comience a navegar por las distintas pantallas de diagnóstico y configuración de la celda (Figura 24).

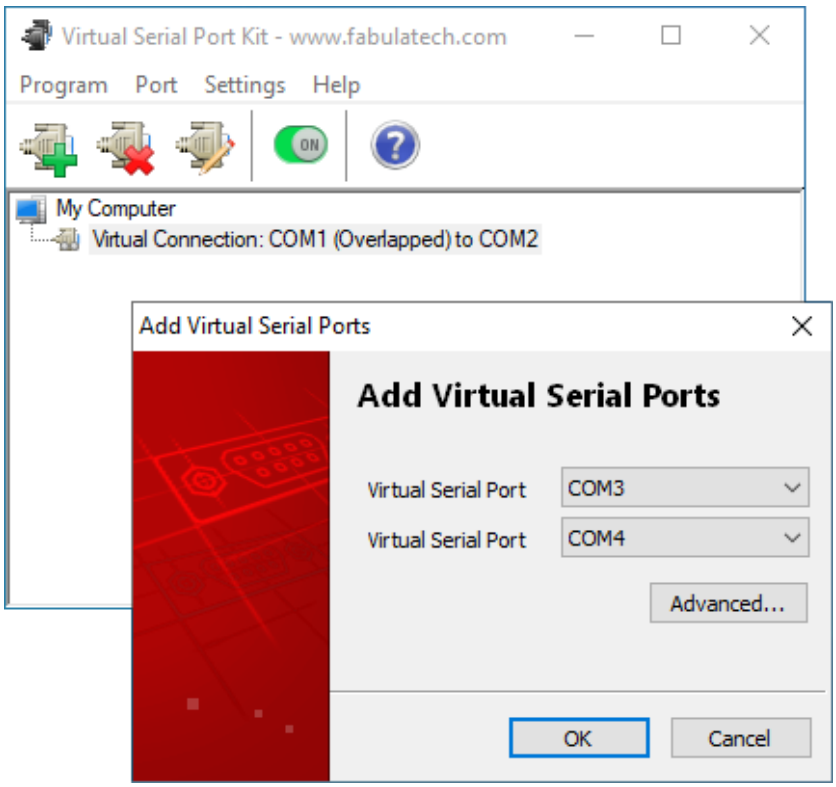
*Figura 24. Visualización del **WebServer** desde CODESYS Automation Server.*



1.6. PUESTA EN MARCHA HMI OMRON

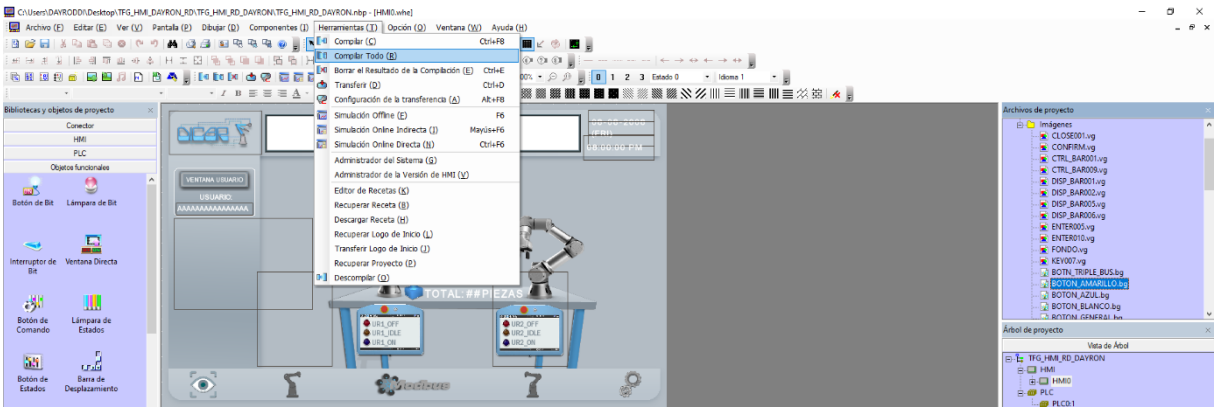
En el apartado 5.6.5 de la página 103 de la memoria del presente proyecto se describe como poner en funcionamiento la herramienta que establece una comunicación virtual entre dos puertos series del PC en aras de simular la conexión física *Modbus_RTU* entre el *PLC* y la pantalla *HMI* (Figura 25), este paso constituye la premisa al proceso de simulación de la interfaz gráfica de *Omron.* En caso de necesidad, diríjase al apartado antes mencionado para configurar *Virtual Serial Port Kit*.

*Figura 25. Creación de una comunicación virtual entre el puerto **COM1** y **COM2** empleando un software de simulación de puerto serie.*



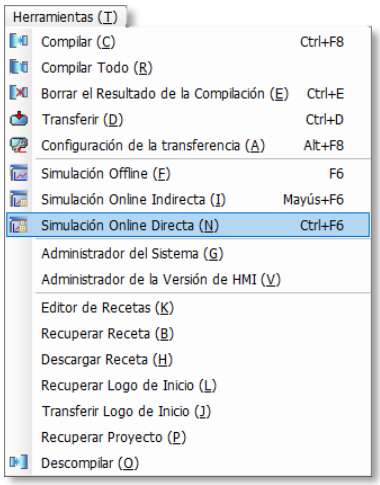
1- Utilizando *NB-Designer*, cargue el programa suministrado por el autor junto con la documentación aportada en la presente investigación. Posteriormente, en el menú principal navegue hasta "***Herramientas"*** y seleccione "***Compilar Todo"*** (Figura 26):

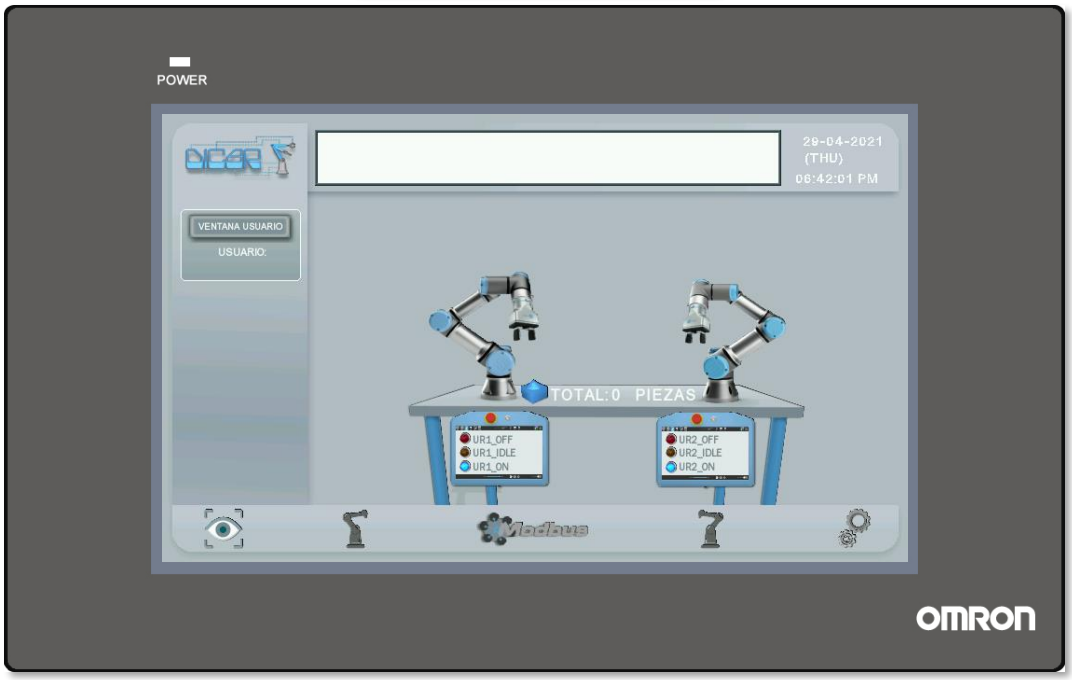
Figura 26. Proceso de compilación del proyecto en NB-Designer.



2- Si el proceso se lleva a cabo de forma satisfactoria, en cuadro de *Mensajes de proyecto*, en la barra inferior del programa, se muestra una leyenda que indica "***Compilación OK. Advertencia 0 Errores 0***" (Figura 27.a), solo resta regresar al menú *Herramientas* y seleccionar "***Simulación Online Directa***" (Figura 27.b), tras lo cual el programa lanza una ventana que simula el dispositivo físico *NB10W-TW01B.*

Figura 27. Secuencia de lanzamiento de la simulación del HMI en NB-Designer.





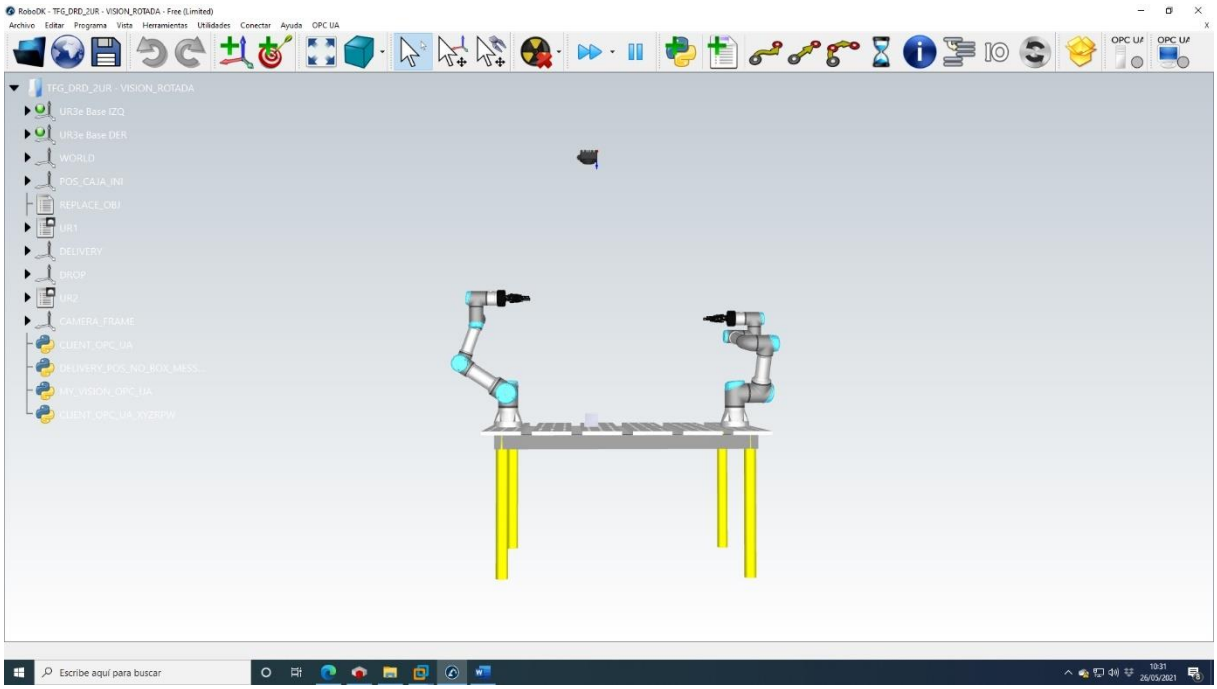
NB-Designer solo permite emplear el simulador por un tiempo máximo de 15 minutos con la versión gratuita del software, tras agotar el período de simulación, el usuario debe repetir los pasos 1 y 2 mencionados.

1.7. PUESTA EN MARCHA RoboDK

Se expone el procedimiento de puesta en marcha de *RoboDK* como último recurso pues es el programa del cual depende la secuencia de trabajo de la celda, tal y como se comentó en el apartado 5.3 del presente proyecto. Para iniciar el funcionamiento de la celda debe:

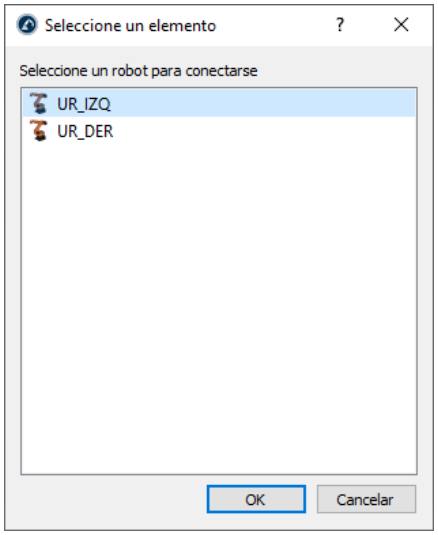
1- Abrir en *RoboDK* el programa suministrado en el repositorio del *TFG* denominado "***TFG_DRD_2UR_VISION_ROTADA.rdk***" (Figura 28):

Figura 28. Celda de trabajo diseñada en RoboDK constituyente del Digital Twin de la estación física real.



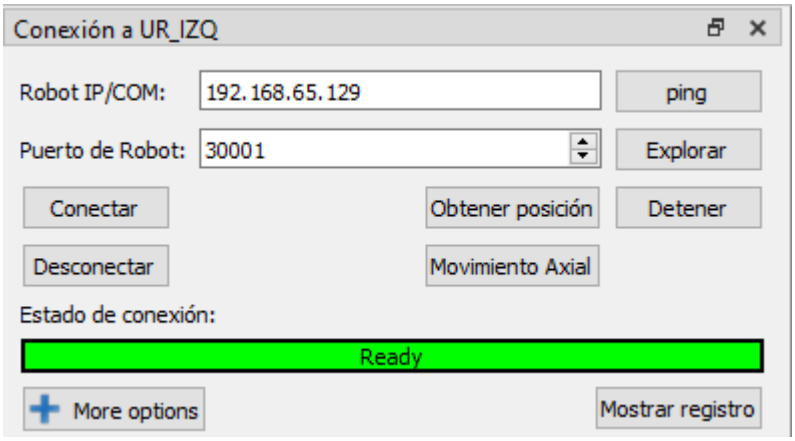
2- En el menú principal diríjase a la opción "***Conectar***" y seleccione "***Conectar Robot***", una ventana emergente muestra los dos robots disponibles (Figura 29), elija "***UR_IZQ***":

Figura 29. Ventana de selección del robot a conectar empleando señales RTDE.



3- A la izquierda del área de trabajo de *RoboDK* una ventana muestra las opciones de configuración para conectar el robot (Figura 30):

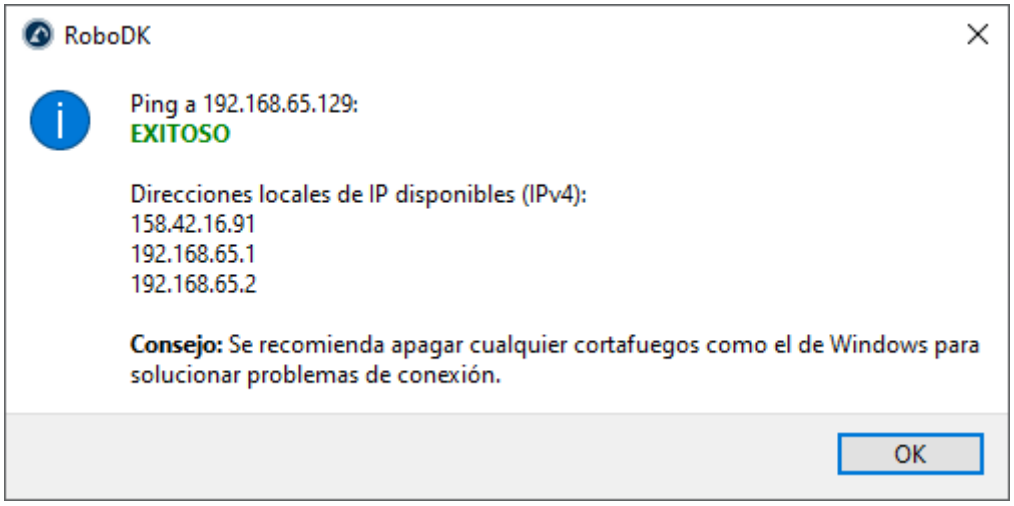
Figura 30. Ventana de conexión para UR_IZQ o UR1.



Compruebe la dirección *IP* de los robots en la máquina virtual de URSim, para ello diríjase al apartado 5.2 en la página 51, cada robots posee una dirección unívoca asignada por *DHCP* que no es posible modificar.

4- Presione el botón "***ping***" para realizar un diagnóstico que compruebe el estado de la comunicación entre *RoboDK* y la máquina *URSim*, si el procedimiento es correcto una ventana emergente muestra el éxito en la comunicación (Figura 31):

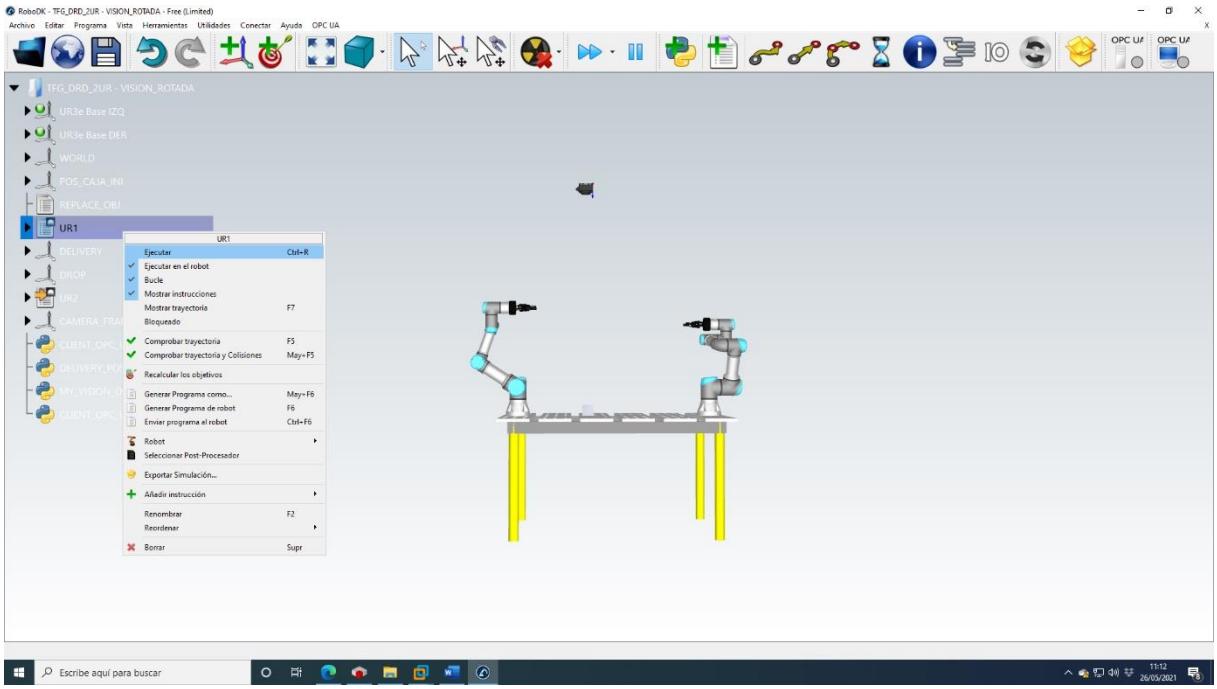
*Figura 31. Diagnóstico de la comunicación entre RoboDK y URSim a través de un procedimiento **PING**.*



5- Presione el botón "***Conectar***" (Figura 30) y observe como la barra de estado pasa de ***"Disconnected"*** (barra de estado gris), a ***"Waiting…"*** (barra de estado color azul claro) y finalmente a ***"Ready"*** (barra de estado color verde claro) (Figura 30). **Repita los pasos 2, 3 y 4 para *UR_DER* teniendo en cuenta la dirección IP de UR2 y el puerto RTDE asignado a cada robot**.

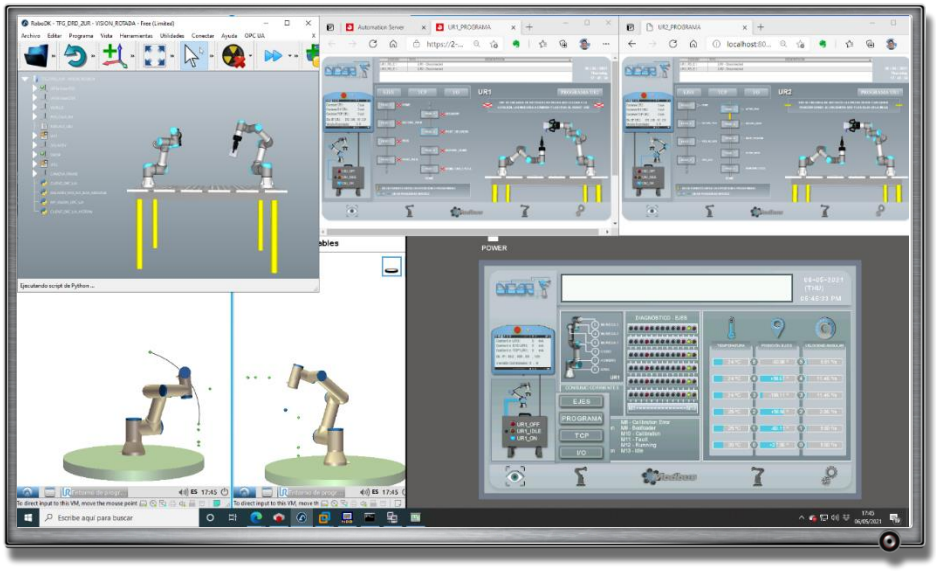
6- En el árbol del proyecto de *RoboDK* presione clic derecho sobre el programa *UR2* y seleccione "***Bucle***", repita el procedimiento anterior pero seleccionado la opción "***Ejecutar***", esta acción deja a *UR2* a esperas de inicio del ciclo de trabajo gobernado por la señal de *UR1* en posición de entrega de pieza y las coordenadas de esta determinadas por la visión artificial. El mismo proceder debe realizarse para *UR1* (Figura 32):

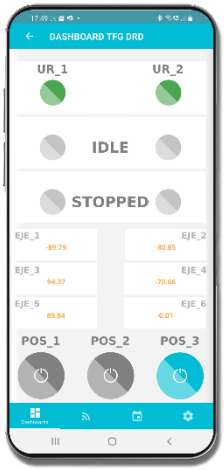
Figura 32. Inicio de la secuencia de trabajo a través de los programas asignados a UR_IZQ y UR_DER en RoboDK.



7- En su pantalla, acomode los distintos elementos que conforman la plataforma de forma que pueda visualizarlos e interactuar con ellos (Figura 33), de esta forma puede comprobar el funcionamiento de la celda y observar la relación entre todos los programas, animaciones, procesos de visión, comunicación de datos a la nube, etc.

Figura 33. Herramientas que conforman la plataforma funcionando en conjunto.





UNIVERSITAT POLITÈCNICA DE VALÈNCIA

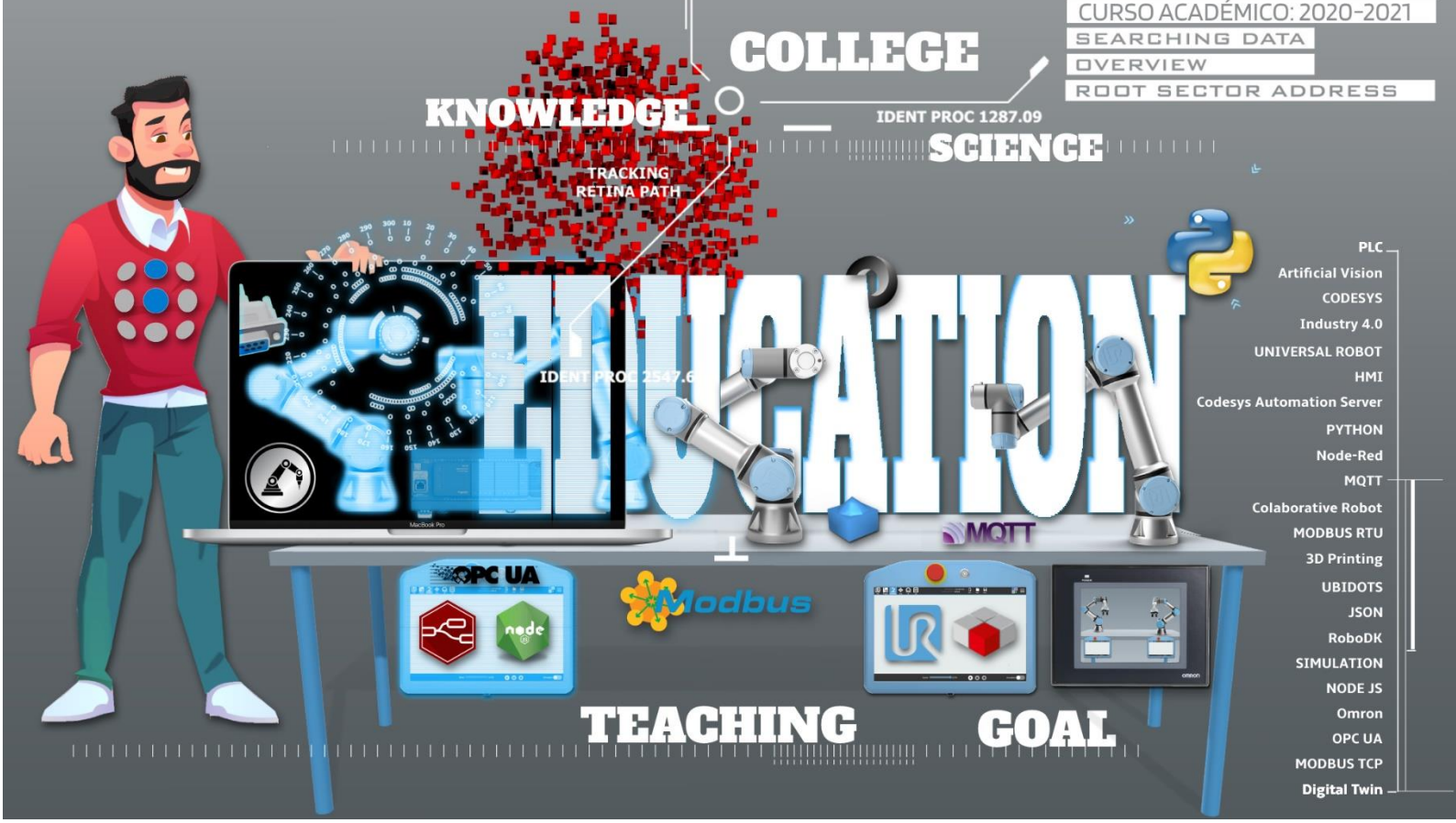
Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA

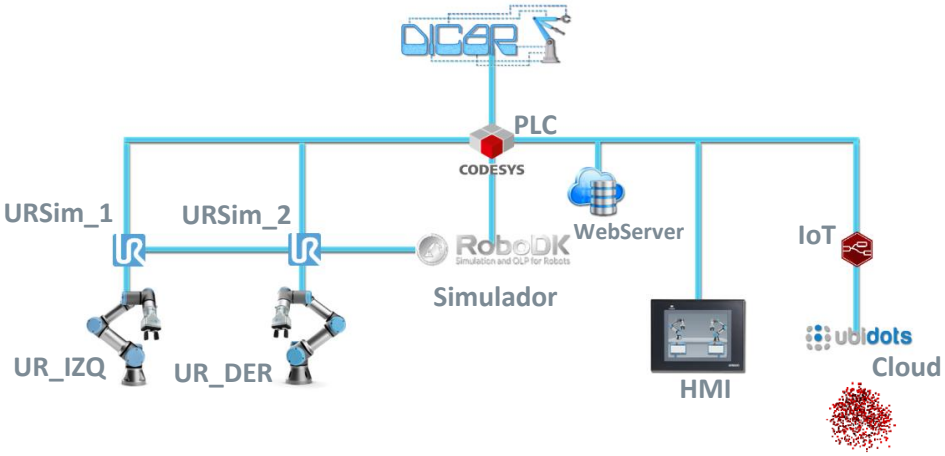
MANUAL DE PROGRAMACIÓN

Dayron Rodríguez Díaz



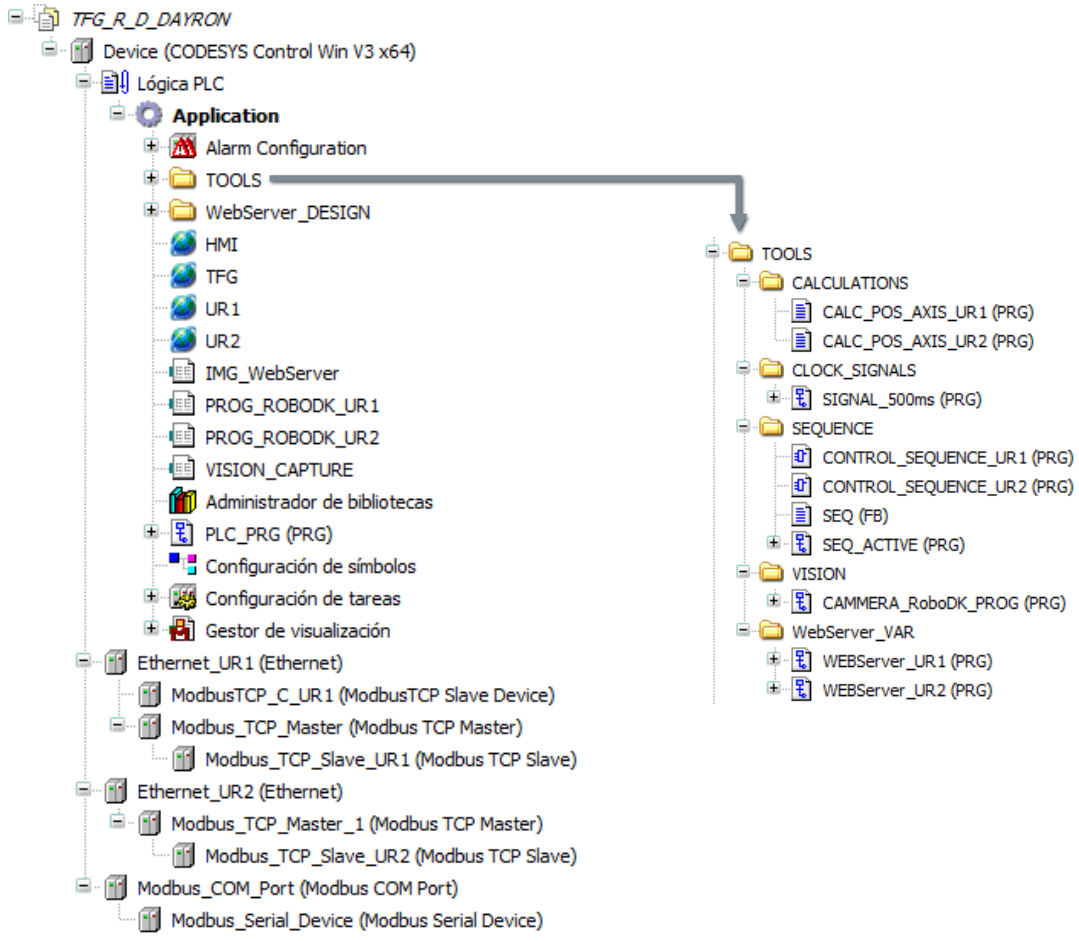
El *manual* describe los programas necesarios para llevar a cabo la implementación de una celda automatizada con robótica colaborativa descrita en la presente memoria y jerarquizada como se muestra en la (Figura 1). Se comenta en cada apartado como hacer uso del código proporcionado:

Figura 1. Organigrama que da solución al Diseño e Implementación de una Celda Automatizada con Robótica Colaborativa.



El presente apartado expone al lector la implementación del código de automatización desarrollado en *CODESYS*. Los lenguajes de programación utilizados cumplen la normativa ***IEC-61131-3*** publicada en febrero del 2013 relativa a Autómatas programables. Parte 3: Lenguajes de programación. La estructura de definición del código y las tablas de variables guardan relación con los programas y las *TVG* representadas en la (Figura 2).

Figura 2. Organización de las carpetas que contienen la lógica de control del proyecto.



1.1. TABLAS DE VARIABLES

El proyecto cuenta con cuatro tablas de variables denominadas ***UR1***, ***UR2***, ***HMI*** y ***TFG***, cada una de ellas relacionadas con los componentes físicos contra los cuales el *PLC* implementa intercambio de datos, con excepción de ***TFG*** que incluye variables de propósito general.

Para evitar la extensión excesiva del documento, el autor omite la publicación de la tabla de variables ***UR2***, pues contiene el mismo número de elementos que su homóloga ***UR1*** con el correspondiente renombramiento de las variables, eliminando el número ***1*** en ***UR*** y sustituyéndolo por el ***2***.

1.1.1. TABLA DE VARIABLES UR1

VAR_GLOBAL

```
                    ////// SEÑALES UR_1 //////

UR1_ROBOT_MODE                  :WORD;      //ESTADO ACTUAL DEL ROBOT [VER ANEXOS: REGISTROS MODBUS SLAVE]
UR1_isSecurityStopped           :WORD;      //PARADA DE SEGURIDAD
UR1_isEmergencyStopped          :WORD;      //ROBOT UR1 PARADO POR EMERGENCIA?
UR1_isTeachButtonPressed        :WORD;      //BOTON HOMBRE MUERTO PRESIONADO
UR1_isPowerButtonPressed        :WORD;      //BOTON DE ENCENDIDO DEL ROBOT PRESIONADO
UR1_isSafetySignalSuchThatWeShouldStop :WORD;   //INDICADOR ROBOT PARADA DE EMERGENCIA

                ////// REGISTRO DE DATOS UR1 PROVENIENTES DE MODBUS SLAVE //////

UR1_POS_EJE1                    :WORD;      //POSICION DE LA BASE DEL ROBOT EN ANGULO
UR1_POS_EJE2                    :WORD;      //POSICION DEL HOMBRO DEL ROBOT EN ANGULO
UR1_POS_EJE3                    :WORD;      //POSICION DEL CODO DEL ROBOT EN ANGULO
UR1_POS_EJE4                    :WORD;      //POSICION DE LA MUÑECA 1 DEL ROBOT EN ANGULO
UR1_POS_EJE5                    :WORD;      //POSICION DE LA MUÑECA 2 DEL ROBOT EN ANGULO
UR1_POS_EJE6                    :WORD;      //POSICION DE LA MUÑECA 3 DEL ROBOT EN ANGULO
UR1_VEL_EJE1                    :WORD;      //VELOCIDAD DE LA BASE DEL ROBOT EN ANGULO
UR1_VEL_EJE2                    :WORD;      //VELOCIDAD DEL HOMBRO DEL ROBOT EN ANGULO
UR1_VEL_EJE3                    :WORD;      //VELOCIDAD DEL CODO DEL ROBOT EN ANGULO
UR1_VEL_EJE4                    :WORD;      //VELOCIDAD DE LA MUÑECA 1 DEL ROBOT EN ANGULO
UR1_VEL_EJE5                    :WORD;      //VELOCIDAD DE LA MUÑECA 2 DEL ROBOT EN ANGULO
UR1_VEL_EJE6                    :WORD;      //VELOCIDAD DE LA MUÑECA 3 DEL ROBOT EN ANGULO
UR1_CURRENT_EJE1                :WORD;      //CORRIENTE CONSUMIDA POR EL EJE 1
UR1_CURRENT_EJE2                :WORD;      //CORRIENTE CONSUMIDA POR EL EJE 2
UR1_CURRENT_EJE3                :WORD;      //CORRIENTE CONSUMIDA POR EL EJE 3
UR1_CURRENT_EJE4                :WORD;      //CORRIENTE CONSUMIDA POR EL EJE 4
```

```
UR1_CURRENT_EJE5        :WORD;    //CORRIENTE CONSUMIDA POR EL EJE 5
UR1_CURRENT_EJE6        :WORD;    //CORRIENTE CONSUMIDA POR EL EJE 6
UR1_TEMP_EJE1           :WORD;    //TEMPERATURA DE LA BASE DEL ROBOT EN ANGULO
UR1_TEMP_EJE2           :WORD;    //TEMPERATURA DEL HOMBRO DEL ROBOT EN ANGULO
UR1_TEMP_EJE3           :WORD;    //TEMPERATURA DEL CODO DEL ROBOT EN ANGULO
UR1_TEMP_EJE4           :WORD;    //TEMPERATURA DE LA MUÑECA 1 DEL ROBOT EN ANGULO
UR1_TEMP_EJE5           :WORD;    //TEMPERATURA DE LA MUÑECA 2 DEL ROBOT EN ANGULO
UR1_TEMP_EJE6           :WORD;    //TEMPERATURA DE LA MUÑECA 3 DEL ROBOT EN ANGULO
UR1_REV_EJE1            :WORD;    //REVOLUCIONES DE LA BASE DEL ROBOT EN ANGULO
UR1_REV_EJE2            :WORD;    //REVOLUCIONES DEL HOMBRO DEL ROBOT EN ANGULO
UR1_REV_EJE3            :WORD;    //REVOLUCIONES DEL CODO DEL ROBOT EN ANGULO
UR1_REV_EJE4            :WORD;    //REVOLUCIONES DE LA MUÑECA 1 DEL ROBOT EN ANGULO
UR1_REV_EJE5            :WORD;    //REVOLUCIONES DE LA MUÑECA 2 DEL ROBOT EN ANGULO
UR1_REV_EJE6            :WORD;    //REVOLUCIONES DE LA MUÑECA 3 DEL ROBOT EN ANGULO
UR1_DIAG_EJE1           :WORD;    //DIAGNÓSTICO DE LA BASE DEL ROBOT
UR1_DIAG_EJE2           :WORD;    //DIAGNÓSTICO DEL HOMBRO DEL ROBOT
UR1_DIAG_EJE3           :WORD;    //DIAGNÓSTICO DEL CODO DEL ROBOT
UR1_DIAG_EJE4           :WORD;    //DIAGNÓSTICO DE LA MUÑECA 1 DEL ROBOT
UR1_DIAG_EJE5           :WORD;    //DIAGNÓSTICO DE LA MUÑECA 2 DEL ROBOT
UR1_DIAG_EJE6           :WORD;    //DIAGNÓSTICO DE LA MUÑECA 3 DEL ROBOT
UR1_TCP_x               :WORD;    //POSICIÓN x DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_y               :WORD;    //POSICIÓN y DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_z               :WORD;    //POSICIÓN z DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_Rx              :WORD;    //ROTACION Rx DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_Ry              :WORD;    //ROTACION Ry DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_Rz              :WORD;    //ROTACION Rz DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_VEL_x           :WORD;    //VELOCIDAD x DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_VEL_y           :WORD;    //VELOCIDAD y DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_VEL_z           :WORD;    //VELOCIDAD z DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_VEL_Rx          :WORD;    //VELOCIDAD ROTACION Rx DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_VEL_Ry          :WORD;    //VELOCIDAD ROTACION Ry DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_VEL_Rz          :WORD;    //VELOCIDAD ROTACION Rz DEL TCP RESPECTO A LA BASE DEL ROBOT
UR1_TCP_OFFSET_x        :WORD;    //POSICIÓN x DEL TCP RESPECTO A LA BRIDA DEL ROBOT
UR1_TCP_OFFSET_y        :WORD;    //POSICIÓN y DEL TCP RESPECTO A LA BRIDA DEL ROBOT
UR1_TCP_OFFSET_z        :WORD;    //POSICIÓN z DEL TCP RESPECTO A LA BRIDA DEL ROBOT
UR1_TCP_OFFSET_Rx       :WORD;    //ROTACION Rx DEL TCP RESPECTO A LA BRIDA DEL ROBOT
UR1_TCP_OFFSET_Ry       :WORD;    //ROTACION Ry DEL TCP RESPECTO A LA BRIDA DEL ROBOT
UR1_TCP_OFFSET_Rz       :WORD;    //ROTACION Rz DEL TCP RESPECTO A LA BRIDA DEL ROBOT
UR1_CURRENT             :WORD;    //CORRIENTE CONSUMIDA POR EL ROBOT
UR1_IO_CURRENT          :WORD;    //CORRIENTE CONSUMIDA POR LAS SEÑALES E/S
UR1_TCP_STATE           :WORD;    //ESTADO DEL TCP DEL ROBOT
UR1_TCP_TEMP            :WORD;    //TEMPERATURA DE LA HERRAMIENTA
UR1_TCP_CURRENT         :WORD;    //CORRIENTE CONSUMIDA POR EL TCP
UR1_CONT_VER_HN         :WORD;    //CONTROLLER VERSION HIGH NUMBER
```

```
UR1_CONT_VER_LN         :WORD;          //CONTROLLER VERSION HIGH NUMBER
            ////// IO DE UR1 POLISCOPE //////

UR1_DI_0                :BOOL:=FALSE;   //ENTRADA DIGITAL INDICA QUE UR2 CONCLUYÓ CICLO ANTERIOR
UR1_DI_1                :BOOL:=FALSE;   //ENTRADA DIGITAL INDICA QUE UR2 TIENE LA CAJA EN EL GRIPPER
UR1_DO_0                :BOOL:=FALSE;   //SALIDA DIGITAL INDICA QUE UR1 ESTÁ EN POSICIÓN DE ENTREGA
UR1_DO_1                :BOOL:=FALSE;   //SALIDA DIGITAL INDICA QUE UR1 TIENE LA CAJA SUJETA EN EL GRIPPER
UR1_DO_2                :BOOL:=FALSE;   //SALIDA DIGITAL INDICA QUE UR1 ABANDONÓ POSICIÓN DE ENTREGA
UR1_RE_1                :WORD:=0;       //REGISTRO D/I QUE INDICA DONDE DEBE ENTREGAR LA CAJA UR1

        ////// ESTADOS UR1 //////

UR1_RS_DISCONNECTED     :BOOL;          //INIDICADOR ROBOT DESCONECTADO
UR1_RS_CONFIRM_SAFETY   :BOOL;          //INDICADOR ROBOT A ESPERA DE CONFIRM. DE PARADA DE SEGURIDAD
UR1_RS_BOOTING          :BOOL;          //INDICADOR ROBOT BOOTING
UR1_RS_POWER_OFF        :BOOL;          //INDICADOR DE ESTADO ROBOT APAGADO
UR1_RS_POWER_ON         :BOOL;          //INDICADOR DE ESTADO ROBOT ENCENDIDO
UR1_RS_IDLE             :BOOL;          //INDICADOR ROBOT ENCENDIDO PERO SIN ALIMENTAR MOTORES
UR1_RS_BACKDRIVE        :BOOL;          //INDICADOR ROBOT EN ESTADO BACKDRIVE
UR1_RS_RUNNING          :BOOL;          //INDICADOR ROBOT ALIMENTADO Y ENCENDIDO
UR1_RS_EJES_OFF         :BOOL;          //INDICADOR ESTADO OFF TODOS LOS EJES UR1
UR1_RS_EJES_ON          :BOOL;          //INDICADOR ESTADO ON TODOS LOS EJES UR1
UR1_RS_EJES_IDLE        :BOOL;          //INDICADOR ESTADO IDLE TODOS LOS EJES UR1
UR1_TCP                 :BOOL;          //INDICADOR ESTADO TCP PARA WEBSERVER Y HMI
UR1_CLOSE_CLAMP         :BOOL;          //VARIABLE PARA EL CONTROL DE APERTURA Y CIERRE DE PINZA
EJES_BUTTON             :BOOL;          //VARIABLE PARA MOSTRAR EL BOTON DE EJES
TCP_BUTTON              :BOOL;          //VARIABLE PARA MOSTRAR EL BOTON DE TCP
IO_BUTTON               :BOOL;          //VARIABLE PARA MOSTRAR EL BOTON DE I/O
PROG_BUTTON             :BOOL;          //VARIABLE PARA MOSTRAR EL BOTON DE PROGRAMA

        ////// COMUNICACION CON RoboDK //////

UR1_DELIVERY_POS        :INT:=1;        //VARIABLE PARA INDICAR EN QUE POSICIÓN DEJARÁ UR1 LA PIEZA

        ////// DIAGNÓSTICO MODBUS SLAVE //////

UR1_COM_STATE           :STRING;        //ALMACENA EL ESTADO DEL BUS MODBUS_TCP SLAVE DE UR1
UR1_MODBUS_ERROR_CODE   :STRING;        //ALMACENA EL SIGNIFICADO DEL ERROR DETECTADO EN EL BUS

        ////// AUXILIARES //////

UR1_IO_DESCRIPTION      :STRING;        //MUESTRA LA LEYENGA DE LA IO SELECCIONADA EN LA VENTANA UR1_IO
```

```
END_VAR
```

1.1.2. TABLA DE VARIABLES HMI

```
VAR_GLOBAL

UR1_RUNNING                          :BOOL;    //UR1 ESTADO RUNNING
UR1_IDLE                             :BOOL;    //UR1 ESTADO STAND-BY
UR1_STOPPED                          :BOOL;    //UR1 ESTADO STOPPED
UR2_RUNNING                          :BOOL;    //UR2 ESTADO RUNNING
UR2_IDLE                             :BOOL;    //UR2 ESTADO STAND-BY
UR2_STOPPED                          :BOOL;    //UR2 ESTADO STOPPED
UR1_RS                               :WORD;    //ALMACENA EL ESTADO DE FUNCTO. DE UR1
UR2_RS                               :WORD;    //ALMACENA EL ESTADO DE FUNCTO. DE UR2
CAMERA_DF                            :WORD;    //ALMACENA EL VALOR DISTANCIA FOCAL
CAMERA_FOV                           :WORD;    //ALMACENA EL VALOR PROFUNDIDAD DE CAMPO
CAMERA_FL                            :WORD;    //ALMACENA EL VALOR LONGITUD LENTE
UR1_POS_E1_0        AT    %MW0       :WORD;    //ÁREA MEMORIA POS_1 (0)
UR1_POS_E1_1        AT    %MW1       :WORD;    //ÁREA MEMORIA POS_1 (1)
UR1_POS_E2_0        AT    %MW2       :WORD;    //ÁREA MEMORIA POS_2 (0)
UR1_POS_E2_1        AT    %MW3       :WORD;    //ÁREA MEMORIA POS_2 (1)
UR1_POS_E3_0        AT    %MW4       :WORD;    //ÁREA MEMORIA POS_3 (0)
UR1_POS_E3_1        AT    %MW5       :WORD;    //ÁREA MEMORIA POS_3 (1)
UR1_POS_E4_0        AT    %MW6       :WORD;    //ÁREA MEMORIA POS_4 (0)
UR1_POS_E4_1        AT    %MW7       :WORD;    //ÁREA MEMORIA POS_4 (1)
UR1_POS_E5_0        AT    %MW8       :WORD;    //ÁREA MEMORIA POS_5 (0)
UR1_POS_E5_1        AT    %MW9       :WORD;    //ÁREA MEMORIA POS_5 (1)
UR1_POS_E6_0        AT    %MW10      :WORD;    //ÁREA MEMORIA POS_6 (0)
UR1_POS_E6_1        AT    %MW11      :WORD;    //ÁREA MEMORIA POS_6 (1)
UR1_POS_AXIS_1                       :WORD;    //POSICION DEL EJE 1
UR1_POS_AXIS_2                       :WORD;    //POSICION DEL EJE 2
UR1_POS_AXIS_3                       :WORD;    //POSICION DEL EJE 3
UR1_POS_AXIS_4                       :WORD;    //POSICION DEL EJE 4
UR1_POS_AXIS_5                       :WORD;    //POSICION DEL EJE 5
UR1_POS_AXIS_6                       :WORD;    //POSICION DEL EJE 6
UR1_VEL_E1_0        AT    %MW12      :WORD;    //ÁREA MEMORIA VEL_EJE_1 (0)
UR1_VEL_E1_1        AT    %MW13      :WORD;    //ÁREA MEMORIA VEL_EJE_1 (1)
UR1_VEL_E2_0        AT    %MW14      :WORD;    //ÁREA MEMORIA VEL_EJE_2 (0)
UR1_VEL_E2_1        AT    %MW15      :WORD;    //ÁREA MEMORIA VEL_EJE_2 (1)
UR1_VEL_E3_0        AT    %MW16      :WORD;    //ÁREA MEMORIA VEL_EJE_3 (0)
UR1_VEL_E3_1        AT    %MW17      :WORD;    //ÁREA MEMORIA VEL_EJE_3 (1)
UR1_VEL_E4_0        AT    %MW18      :WORD;    //ÁREA MEMORIA VEL_EJE_4 (0)
UR1_VEL_E4_1        AT    %MW19      :WORD;    //ÁREA MEMORIA VEL_EJE_4 (1)
```

```
UR1_VEL_E5_0        AT    %MW20     :WORD;    //ÁREA MEMORIA VEL_EJE_5 (0)
UR1_VEL_E5_1        AT    %MW21     :WORD;    //ÁREA MEMORIA VEL_EJE_5 (1)
UR1_VEL_E6_0        AT    %MW22     :WORD;    //ÁREA MEMORIA VEL_EJE_6 (0)
UR1_VEL_E6_1        AT    %MW23     :WORD;    //ÁREA MEMORIA VEL_EJE_6 (1)
UR1_VEL_AXIS_1                      :WORD;    //VELOCIDAD DEL EJE 1
UR1_VEL_AXIS_2                      :WORD;    //VELOCIDAD DEL EJE 2
UR1_VEL_AXIS_3                      :WORD;    //VELOCIDAD DEL EJE 3
UR1_VEL_AXIS_4                      :WORD;    //VELOCIDAD DEL EJE 4
UR1_VEL_AXIS_5                      :WORD;    //VELOCIDAD DEL EJE 5
UR1_VEL_AXIS_6                      :WORD;    //VELOCIDAD DEL EJE 6
UR1_mA_E1_0         AT    %MW24     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_1 (0)
UR1_mA_E1_1         AT    %MW25     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_1 (1)
UR1_mA_E2_0         AT    %MW26     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_2 (0)
UR1_mA_E2_1         AT    %MW27     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_2 (1)
UR1_mA_E3_0         AT    %MW28     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_3 (0)
UR1_mA_E3_1         AT    %MW29     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_3 (1)
UR1_mA_E4_0         AT    %MW30     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_4 (0)
UR1_mA_E4_1         AT    %MW31     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_4 (1)
UR1_mA_E5_0         AT    %MW32     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_5 (0)
UR1_mA_E5_1         AT    %MW33     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_5 (1)
UR1_mA_E6_0         AT    %MW34     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_6 (0)
UR1_mA_E6_1         AT    %MW35     :WORD;    //ÁREA MEMORIA CORRIENTE_EJE_6 (1)
UR1_IP_0                            :WORD;    //BYTE 0 RELATIVO A LA DIRECCION IP UR1
UR1_IP_1                            :WORD;    //BYTE 1 RELATIVO A LA DIRECCION IP UR1
UR1_IP_2                            :WORD;    //BYTE 2 RELATIVO A LA DIRECCION IP UR1
UR1_IP_3                            :WORD;    //BYTE 3 RELATIVO A LA DIRECCION IP UR1

END_VAR
```

1.1.3. TABLA DE VARIABLES TFG

VAR_GLOBAL

```
MODBUS_STATE_BUTTON     :BOOL:=FALSE; //CONTROLA APARIENCIA DEL BOTON MODBUS EN WEBSERVER Y HMI
UR1_BOCADILLO           :BOOL:=FALSE; //CONTROLA APARIENCIA UR_IZQ MODBUS EN WEBSERVER Y HMI
UR2_BOCADILLO           :BOOL:=FALSE; //CONTROLA APARIENCIA UR_DER MODBUS EN WEBSERVER Y HMI
VISION_BOCADILLO        :BOOL:=FALSE; //CTRL APARIENCIA DE LOS BOTONES DE VISION EN EL MENÚ PRPAL. INF.
CONFIG_BOCADILLO        :BOOL:=FALSE; //CTRL APARIENCIA DE LOS BOTONES DE CONFG. EN EL MENÚ PRPAL INF.
SECUENCIA_UR1           :STRING;      //CONTROLA LA SECUENCIA Y LOS PASOS DEL TRABAJO DE UR1
SECUENCIA_UR2           :STRING;      //CONTROLA LA SECUENCIA Y LOS PASOS DEL TRABAJO DE UR2
CAMERA_DF               :WORD:=6;     //VALOR DE DISTANCIA FOCAL DE LA CÁMARA EN RoboDK
CAMERA_FOV              :WORD:=30;    //VALOR DEL CAMPO DE VISION DE LA CÁMARA EN RoboDK
CAMERA_FL               :WORD:=1000;  //VALOR LGTG. DE LA LENTE DE LA CÁMARA EN RoboDK
MODBUS_DETAIL           :BOOL:=FALSE; //CTRL LA APARICIÓN DEL BOCADILLO DE DETALLE DE ERRORES EN EL BUS
UR1_IO_DETAIL           :BOOL:=FALSE; //MUESTRA VENTANA DE DETALLE DE LAS IO SERVER UR1
UR2_IO_DETAIL           :BOOL:=FALSE; //MUESTRA VENTANA DE DETALLE DE LAS IO SERVER UR2
UR1_mA_WINDOW           :BOOL:=FALSE; //MUESTRA VENTANA DE CORRTES. CONSUMIDAS EN CADA EJE DE UR1
UR2_mA_WINDOW           :BOOL:=FALSE; //MUESTRA VENTANA DE CORRTES. CONSUMIDAS EN CADA EJE DE UR1
UBIDOTS_POS_1           :BOOL;        //ALMACENA LA POSICIÓN DE ENTREGA 1 DE LA PIEZA EN UR1
UBIDOTS_POS_2           :BOOL;        //ALMACENA LA POSICIÓN DE ENTREGA 2 DE LA PIEZA EN UR1
UBIDOTS_POS_3           :BOOL;        //ALMACENA LA POSICIÓN DE ENTREGA 3 DE LA PIEZA EN UR1
POS_1                   :BOOL;        //VARIABLE PARA ACTIVAR LA POSICIÓN DELIVERY_1 EN EL ROBOT UR1
POS_2                   :BOOL;        //VARIABLE PARA ACTIVAR LA POSICIÓN DELIVERY_2 EN EL ROBOT UR1
POS_3                   :BOOL;        //VARIABLE PARA ACTIVAR LA POSICIÓN DELIVERY_3 EN EL ROBOT UR1
```

END_VAR

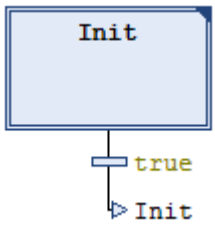
2.2. LÓGICA DE CONTROL CODESYS

2.2.1. PLC_PRG()

2.2.1.1 VARIABLES

```
VAR

AUX_WORD        :WORD;          //VARIABLE PARA CARGAR LOS DATOS REALES QUE SE MANDAN AL HMI
POS1            :BOOL;          //VARIABLE PARA ACTIVAR LA POSICION DE ENTREGA 1
POS2            :BOOL;          //VARIABLE PARA ACTIVAR LA POSICION DE ENTREGA 2
POS3            :BOOL;          //VARIABLE PARA ACTIVAR LA POSICION DE ENTREGA 3

END_VAR
```

2.2.1.2 PROGRAMA



2.2.1.3 PLC_PRG.Init.active

```
CASE Modbus_TCP_Slave_UR1.ComState OF
        0:      UR1.UR1_COM_STATE:='OFF';
        1:      UR1.UR1_COM_STATE:='CONNECTING';
        2:      UR1.UR1_COM_STATE:='CONNECTED';
        3:      UR1.UR1_COM_STATE:='DISCONNECTING';
        4:      UR1.UR1_COM_STATE:='SOCKET_ERROR';
END_CASE

CASE Modbus_TCP_Slave_UR2.ComState OF
        0:      UR2.UR2_COM_STATE:='OFF';
        1:      UR2.UR2_COM_STATE:='CONNECTING';
        2:      UR2.UR2_COM_STATE:='CONNECTED';
        3:      UR2.UR2_COM_STATE:='DISCONNECTING';
        4:      UR2.UR2_COM_STATE:='SOCKET_ERROR';
END_CASE

AUX_WORD:=TFG.CAMERA_DF;
AUX_WORD:=TFG.CAMERA_FOV;
AUX_WORD:=TFG.CAMERA_FL;

        // DETERMINAR LA POSICIÓN DE ENTREGA DE LA PIEZA POR UR1 SÓLO ES POSIBLE CAMBIARLA A TRAVÉS DE UBIDOTS//

IF TFG.UBIDOTS_POS_1 AND NOT TFG.UBIDOTS_POS_2 AND NOT TFG.UBIDOTS_POS_3 THEN
        POS1:=TRUE;
        POS2:=FALSE;
```

```
            POS3:=FALSE;
            END_IF
IF TFG.UBIDOTS_POS_2 AND NOT TFG.UBIDOTS_POS_1 AND NOT TFG.UBIDOTS_POS_3 THEN
            POS1:=FALSE;
            POS2:=TRUE;
            POS3:=FALSE;
END_IF
IF TFG.UBIDOTS_POS_3 AND NOT TFG.UBIDOTS_POS_1 AND NOT TFG.UBIDOTS_POS_2 THEN
            POS1:=FALSE;
            POS2:=FALSE;
            POS3:=TRUE;
END_IF
IF POS1 THEN
            UR1.UR1_DELIVERY_POS:=1;
END_IF
IF POS2 THEN
            UR1.UR1_DELIVERY_POS:=2;
END_IF
IF POS3 THEN
            UR1.UR1_DELIVERY_POS:=3;
END_IF

                                                    // ESTADOS DE UR1 Y UR2 //

HMI.UR1_RUNNING:=UR1.UR1_RS_RUNNING;
HMI.UR1_IDLE:=UR1.UR1_RS_IDLE;
HMI.UR1_STOPED:=UR1.UR1_RS_POWER_OFF;
HMI.UR2_RUNNING:=UR2.UR2_RS_RUNNING;
HMI.UR2_IDLE:=UR2.UR2_RS_IDLE;
HMI.UR2_STOPED:=UR2.UR2_RS_POWER_OFF;

                                                    // ENVÍO DE LA POSICIÓN AL HMI //

AUX_WORD:=HMI.UR1_POS_E1_0;
AUX_WORD:=HMI.UR1_POS_E1_1;
AUX_WORD:=HMI.UR1_POS_E2_0;
AUX_WORD:=HMI.UR1_POS_E2_1;
AUX_WORD:=HMI.UR1_POS_E3_0;
AUX_WORD:=HMI.UR1_POS_E3_1;
AUX_WORD:=HMI.UR1_POS_E4_0;
AUX_WORD:=HMI.UR1_POS_E4_1;
AUX_WORD:=HMI.UR1_POS_E5_0;
AUX_WORD:=HMI.UR1_POS_E5_1;
AUX_WORD:=HMI.UR1_POS_E6_0;
AUX_WORD:=HMI.UR1_POS_E6_1;

HMI.UR1_POS_AXIS_1:=REAL_TO_WORD(CALC_POS_AXIS_UR1.UR1_EJE1);
HMI.UR1_POS_AXIS_2:=REAL_TO_WORD(CALC_POS_AXIS_UR1.UR1_EJE2);
HMI.UR1_POS_AXIS_3:=REAL_TO_WORD(CALC_POS_AXIS_UR1.UR1_EJE3);
HMI.UR1_POS_AXIS_4:=REAL_TO_WORD(CALC_POS_AXIS_UR1.UR1_EJE4);
HMI.UR1_POS_AXIS_5:=REAL_TO_WORD(CALC_POS_AXIS_UR1.UR1_EJE5);
HMI.UR1_POS_AXIS_6:=REAL_TO_WORD(CALC_POS_AXIS_UR1.UR1_EJE6);

                                                    // ENVÍO DE LA VELOCIDAD ANGULAR AL HMI //

AUX_WORD:=HMI.UR1_VEL_E1_0;
AUX_WORD:=HMI.UR1_VEL_E1_1;
AUX_WORD:=HMI.UR1_VEL_E2_0;
AUX_WORD:=HMI.UR1_VEL_E2_1;
```

```
AUX_WORD:=HMI.UR1_VEL_E3_0;
AUX_WORD:=HMI.UR1_VEL_E3_1;
AUX_WORD:=HMI.UR1_VEL_E4_0;
AUX_WORD:=HMI.UR1_VEL_E4_1;
AUX_WORD:=HMI.UR1_VEL_E5_0;
AUX_WORD:=HMI.UR1_VEL_E5_1;
AUX_WORD:=HMI.UR1_VEL_E6_0;
AUX_WORD:=HMI.UR1_VEL_E6_1;

HMI.UR1_VEL_AXIS_1:=REAL_TO_WORD(WEBServer_UR1.UR1_VEL1);
HMI.UR1_VEL_AXIS_2:=REAL_TO_WORD(WEBServer_UR1.UR1_VEL2);
HMI.UR1_VEL_AXIS_3:=REAL_TO_WORD(WEBServer_UR1.UR1_VEL3);
HMI.UR1_VEL_AXIS_4:=REAL_TO_WORD(WEBServer_UR1.UR1_VEL4);
HMI.UR1_VEL_AXIS_5:=REAL_TO_WORD(WEBServer_UR1.UR1_VEL5);
HMI.UR1_VEL_AXIS_6:=REAL_TO_WORD(WEBServer_UR1.UR1_VEL6);

                    // ENVÍO DE LA CORRIENTE CONSUMIDA AL HMI //

AUX_WORD:=HMI.UR1_mA_E1_0;
AUX_WORD:=HMI.UR1_mA_E1_1;
AUX_WORD:=HMI.UR1_mA_E2_0;
AUX_WORD:=HMI.UR1_mA_E2_1;
AUX_WORD:=HMI.UR1_mA_E3_0;
AUX_WORD:=HMI.UR1_mA_E3_1;
AUX_WORD:=HMI.UR1_mA_E4_0;
AUX_WORD:=HMI.UR1_mA_E4_1;
AUX_WORD:=HMI.UR1_mA_E5_0;
AUX_WORD:=HMI.UR1_mA_E5_1;
AUX_WORD:=HMI.UR1_mA_E6_0;
AUX_WORD:=HMI.UR1_mA_E6_1;

                    // ENVÍO DE LA DIRECCIÓN IP DE UR1 AL HMI //

HMI.UR1_IP_0:=Modbus_TCP_Slave_UR1.ComSettings.ipAddress[0];
HMI.UR1_IP_1:=Modbus_TCP_Slave_UR1.ComSettings.ipAddress[1];
HMI.UR1_IP_2:=Modbus_TCP_Slave_UR1.ComSettings.ipAddress[2];
HMI.UR1_IP_3:=Modbus_TCP_Slave_UR1.ComSettings.ipAddress[3];
```

2.2.2. CALC_POS_AXIS_UR1()

Existe el correspondiente programa homólogo para el robot *UR2* con las mismas instrucciones, solamente cambia la denominación de las variables tal y como se comentó en el apartado A_2.1. TABLAS DE VARIABLES.

2.2.2.1 VARIABLES

```
VAR

        UR1_EJE1 AT     %MW0    :REAL:=0;                                   //POSICION ANGULAR DEL EJE1
        UR1_EJE2 AT     %MW2    :REAL:=0;                                   //POSICION ANGULAR DEL EJE2
        UR1_EJE3 AT     %MW4    :REAL:=0;                                   //POSICION ANGULAR DEL EJE3
        UR1_EJE4 AT     %MW6    :REAL:=0;                                   //POSICION ANGULAR DEL EJE4
        UR1_EJE5 AT     %MW8    :REAL:=0;                                   //POSICION ANGULAR DEL EJE5
        UR1_EJE6 AT     %MW10   :REAL:=0;                                   //POSICION ANGULAR DEL EJE6
        UR1_REV1                :WORD:=UR1.UR1_REV_EJE1;                    //REVOLUCIONES DEL EJE1
        UR1_REV2                :WORD:=UR1.UR1_REV_EJE2;                    //REVOLUCIONES DEL EJE2
        UR1_REV3                :WORD:=UR1.UR1_REV_EJE3;                    //REVOLUCIONES DEL EJE3
        UR1_REV4                :WORD:=UR1.UR1_REV_EJE4;                    //REVOLUCIONES DEL EJE3
        UR1_REV5                :WORD:=UR1.UR1_REV_EJE5;                    //REVOLUCIONES DEL EJE5
        UR1_REV6                :WORD:=UR1.UR1_REV_EJE6;                    //REVOLUCIONES DEL EJE6
        PI                      :REAL:=3.14159265358979323846264338327795;  //CONSTANTE MATEMÁTICA

END_VAR
```

2.2.2.2 PROGRAMA

```
                                                /// UR1_EJE[1] BASE///
IF UR1.UR1_REV_EJE1=0 THEN
        UR1_EJE1:=(WORD_TO_REAL(UR1.UR1_POS_EJE1))*((180)/(1000*PI));
                ELSIF
                        UR1.UR1_POS_EJE1=0 THEN
                        UR1_EJE1:=0;
                ELSE
                        UR1_EJE1:=(WORD_TO_REAL(UR1.UR1_POS_EJE1))*((180)/(1000*PI))-360;
        END_IF

                                                /// UR1_EJE[2] HOMBRO///
IF UR1.UR1_REV_EJE2>0 THEN
        UR1_EJE2:=(WORD_TO_REAL(UR1.UR1_POS_EJE2))*((180)/(1000*PI))-360;
                ELSIF
                        UR1.UR1_POS_EJE2=0 THEN
                        UR1_EJE2:=0;
                ELSIF   UR1.UR1_REV_EJE2=0 THEN
                        UR1_EJE2:=(WORD_TO_REAL(UR1.UR1_POS_EJE2))*((180)/(1000*PI));
        END_IF

                                                /// UR1_EJE[3] CODO///
IF UR1.UR1_REV_EJE3=0 THEN
        UR1_EJE3:=(WORD_TO_REAL(UR1.UR1_POS_EJE3))*((180)/(1000*PI));
                ELSIF
                        UR1.UR1_POS_EJE3=0 THEN
                        UR1_EJE3:=0;
                ELSE
```

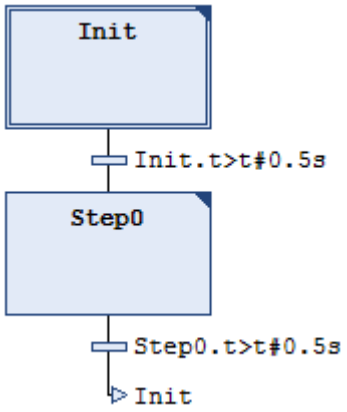
```
                    UR1_EJE3:=(WORD_TO_REAL(UR1.UR1_POS_EJE3))*((180)/(1000*PI))-360;
        END_IF

                                        /// UR1_EJE[4] MUÑECA 1 ///
IF UR1.UR1_REV_EJE4>1 THEN
                    UR1_EJE4:=(WORD_TO_REAL(UR1.UR1_POS_EJE4))*((180)/(1000*PI))-360;
                ELSIF
                    UR1.UR1_POS_EJE4=0 THEN
                    UR1_EJE4:=0;
                ELSIF   UR1.UR1_REV_EJE4=0 THEN
                    UR1_EJE4:=(WORD_TO_REAL(UR1.UR1_POS_EJE4))*((180)/(1000*PI));
        END_IF

                                        /// UR1_EJE[5] MUÑECA 2///
IF UR1.UR1_REV_EJE5=0 THEN
                    UR1_EJE5:=(WORD_TO_REAL(UR1.UR1_POS_EJE5))*((180)/(1000*PI));
                ELSIF
                    UR1.UR1_POS_EJE5=0 THEN
                    UR1_EJE5:=0;
                ELSE
                    UR1_EJE5:=(WORD_TO_REAL(UR1.UR1_POS_EJE5))*((180)/(1000*PI))-360;
        END_IF

                                        /// UR1_EJE[6] MUÑECA 3///
IF UR1.UR1_REV_EJE6=0 THEN
                    UR1_EJE6:=(WORD_TO_REAL(UR1.UR1_POS_EJE6))*((180)/(1000*PI));
                ELSIF
                    UR1.UR1_POS_EJE6=0 THEN
                    UR1_EJE6:=0;
                ELSE
                    UR1_EJE6:=(WORD_TO_REAL(UR1.UR1_POS_EJE6))*((180)/(1000*PI))-360;
        END_IF
```

2.2.3. SIGNAL_500ms()

2.2.3.1 VARIABLES

```
VAR

        T_ON    :BOOL;    //SEÑAL QUE ACTIVA VARIABLES CON UNA FRECUENCIA DE 2Hz (0.5 SEGUNDOS)

END_VAR
```

2.2.3.2 PROGRAMA



2.2.3.3 SIGNAL_500ms.Init.active

```
T_ON:=TRUE;
```

2.2.3.4 SIGNAL_500ms.Step0.active

```
T_ON:=FALSE;
```

2.2.4. SEQ ()

2.2.4.1 VARIABLES

```
FUNCTION_BLOCK SEQ
VAR_INPUT

        UR_E1           :REAL;          //POSICIÓN DEL EJE 1 DE UR EN RoboDK
        UR_E2           :REAL;          //POSICIÓN DEL EJE 2 DE UR EN RoboDK
        UR_E3           :REAL;          //POSICIÓN DEL EJE 3 DE UR EN RoboDK
        UR_E4           :REAL;          //POSICIÓN DEL EJE 4 DE UR EN RoboDK
        UR_E5           :REAL;          //POSICIÓN DEL EJE 5 DE UR EN RoboDK
        UR_E1_SETPOINT  :REAL;          //POSICIÓN DEL EJE 1 DE UR1 SETPOINT
        UR_E2_SETPOINT  :REAL;          //POSICIÓN DEL EJE 2 DE UR1 SETPOINT
        UR_E3_SETPOINT  :REAL;          //POSICIÓN DEL EJE 3 DE UR1 SETPOINT
        UR_E4_SETPOINT  :REAL;          //POSICIÓN DEL EJE 4 DE UR1 SETPOINT
        UR_E5_SETPOINT  :REAL;          //POSICIÓN DEL EJE 5 DE UR1 SETPOINT

END_VAR

VAR_OUTPUT

        PASO            :BOOL:=FALSE;   //TRUE SI SE CUMPLE LA CONDICIÓN DE POSICIÓN

END_VAR

VAR
        Aux_UR_E1       :BOOL:=FALSE;   //VAR. AUX. PARA COMPROBAR SI EL EJE 1 DE UR1 ESTÁ EN LA ZONA DE INTERÉS
        Aux_UR_E2       :BOOL:=FALSE;   //VAR. AUX. PARA COMPROBAR SI EL EJE 2 DE UR1 ESTÁ EN LA ZONA DE INTERÉS
        Aux_UR_E3       :BOOL:=FALSE;   //VAR. AUX. PARA COMPROBAR SI EL EJE 3 DE UR1 ESTÁ EN LA ZONA DE INTERÉS
        Aux_UR_E4       :BOOL:=FALSE;   //VAR. AUX. PARA COMPROBAR SI EL EJE 4 DE UR1 ESTÁ EN LA ZONA DE INTERÉS
        Aux_UR_E5       :BOOL:=FALSE;   //VAR. AUX. PARA COMPROBAR SI EL EJE 5 DE UR1 ESTÁ EN LA ZONA DE INTERÉS

END_VAR
```

2.2.4.2 PROGRAMA

```
IF UR_E1>=UR_E1_SETPOINT-1 AND UR_E1<=UR_E1_SETPOINT+1 THEN
                Aux_UR_E1:=TRUE;
        ELSE
                Aux_UR_E1:=FALSE;
END_IF

IF UR_E2>=UR_E2_SETPOINT-1 AND UR_E2<=UR_E2_SETPOINT+1 THEN
                Aux_UR_E2:=TRUE;
        ELSE
                Aux_UR_E2:=FALSE;
END_IF

IF UR_E3>=UR_E3_SETPOINT-1 AND UR_E3<=UR_E3_SETPOINT+1 THEN
                Aux_UR_E3:=TRUE;
        ELSE
                Aux_UR_E3:=FALSE;
END_IF

IF UR_E4>=UR_E4_SETPOINT-1 AND UR_E4<=UR_E4_SETPOINT+1 THEN
                Aux_UR_E4:=TRUE;
```

```
		ELSE
			Aux_UR_E4:=FALSE;
END_IF

IF UR_E5>=UR_E5_SETPOINT-1 AND UR_E5<=UR_E5_SETPOINT+1 THEN
			Aux_UR_E5:=TRUE;
		ELSE
			Aux_UR_E5:=FALSE;
END_IF

IF (Aux_UR_E1) AND (Aux_UR_E2) AND (Aux_UR_E3) AND (Aux_UR_E4) AND (Aux_UR_E5) THEN
			PASO:=TRUE;
		ELSE
			PASO:=FALSE;
END_IF
```

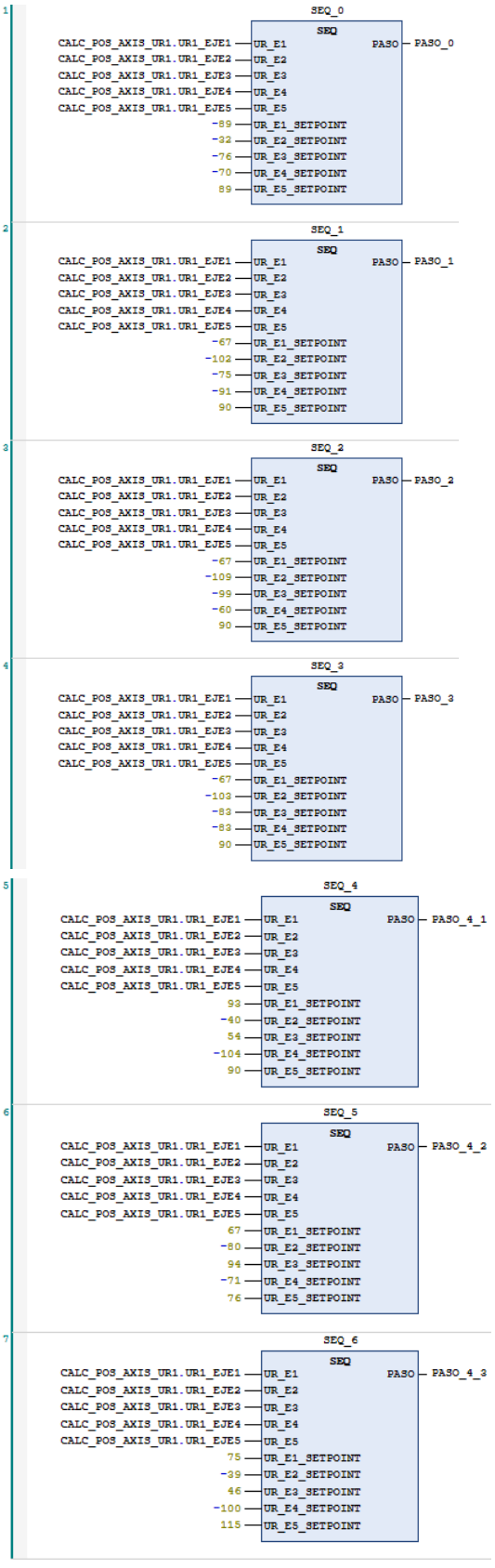
2.2.5. CONTROL_SEQUENCE_UR1 ()

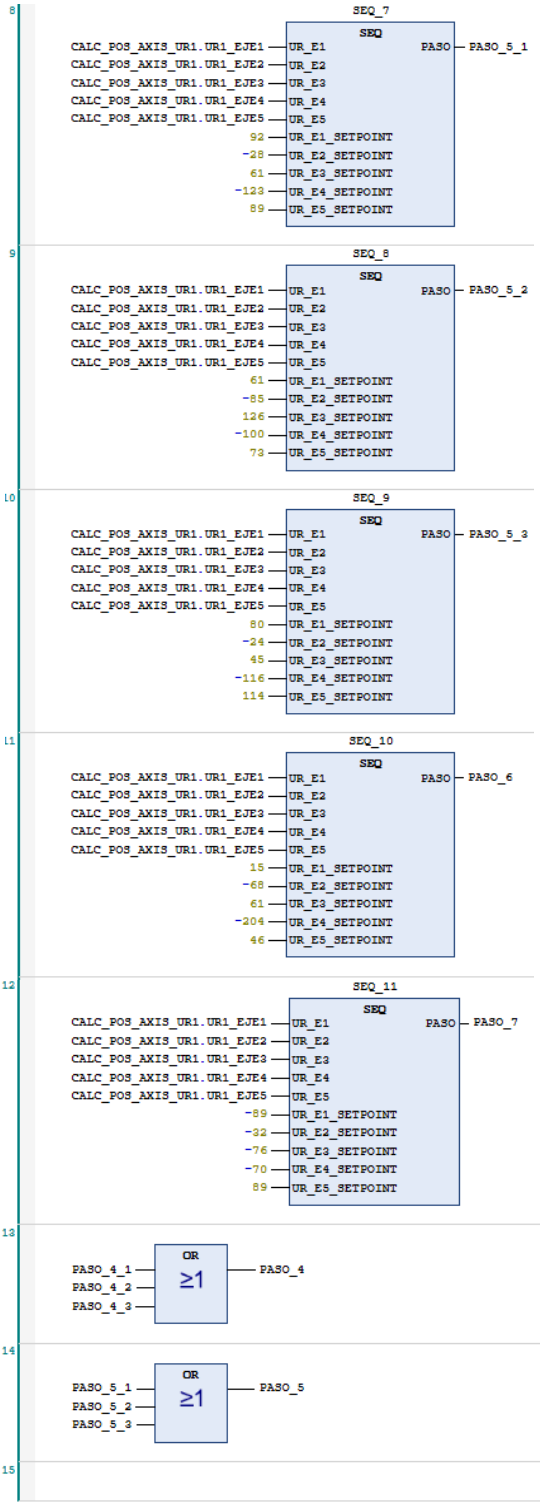
Existe el correspondiente programa homólogo para el robot *UR2* con las mismas instrucciones, solamente cambia la denominación de las variables tal y como se comentó en el apartado A_2.1. TABLAS DE VARIABLES.

2.2.5.1 VARIABLES

```
PROGRAM CONTROL_SEQUENCE_UR1
VAR

		PASO_0		:BOOL:=FALSE;		//UR1 EN HOME
		PASO_1		:BOOL:=FALSE;		//UR1 APPROX_PICK
		PASO_2		:BOOL:=FALSE;		//UR1 EN POS_PICK
		PASO_3		:BOOL:=FALSE;		//UR1 EN POS AFTER_PICK
		PASO_4		:BOOL:=FALSE;		//UR1 EN POS_DEL
		PASO_4_1	:BOOL:=FALSE;		//UR1 EN POS_DEL_1
		PASO_4_2	:BOOL:=FALSE;		//UR1 EN POS_DEL_2
		PASO_4_3	:BOOL:=FALSE;		//UR1 EN POS_DEL_3
		PASO_5		:BOOL:=FALSE;		//UR1 EN POS AFTER_DEL
		PASO_5_1	:BOOL:=FALSE;		//UR1 EN POS AFTER_DEL_1
		PASO_5_2	:BOOL:=FALSE;		//UR1 EN POS AFTER_DEL_2
		PASO_5_3	:BOOL:=FALSE;		//UR1 EN POS AFTER_DEL_3
		PASO_6		:BOOL:=FALSE;		//UR1 EN POS BEFORE HOME
		PASO_7		:BOOL:=FALSE;		//UR1 EN HOME SIN BOX

		SEQ_0		:SEQ;			//UR1 EN HOME
		SEQ_1		:SEQ;			//UR1 APPROX_PICK
		SEQ_2		:SEQ;			//UR1 EN POS_PICK
		SEQ_3		:SEQ;			//UR1 EN POS AFTER_PICK
		SEQ_4		:SEQ;			//UR1 EN POS_DEL_1
		SEQ_5		:SEQ;			//UR1 EN POS_DEL_2
		SEQ_6		:SEQ;			//UR1 EN POS_DEL_3
		SEQ_7		:SEQ;			//UR1 EN POS AFTER_DEL_1
		SEQ_8		:SEQ;			//UR1 EN POS AFTER_DEL_2
		SEQ_9		:SEQ;			//UR1 EN POS AFTER_DEL_3
		SEQ_10		:SEQ;			//UR1 EN POS BEFORE HOME
		SEQ_11		:SEQ;			//UR1 EN HOME SIN BOX

END_VAR
```

2.2.5.2 PROGRAMA

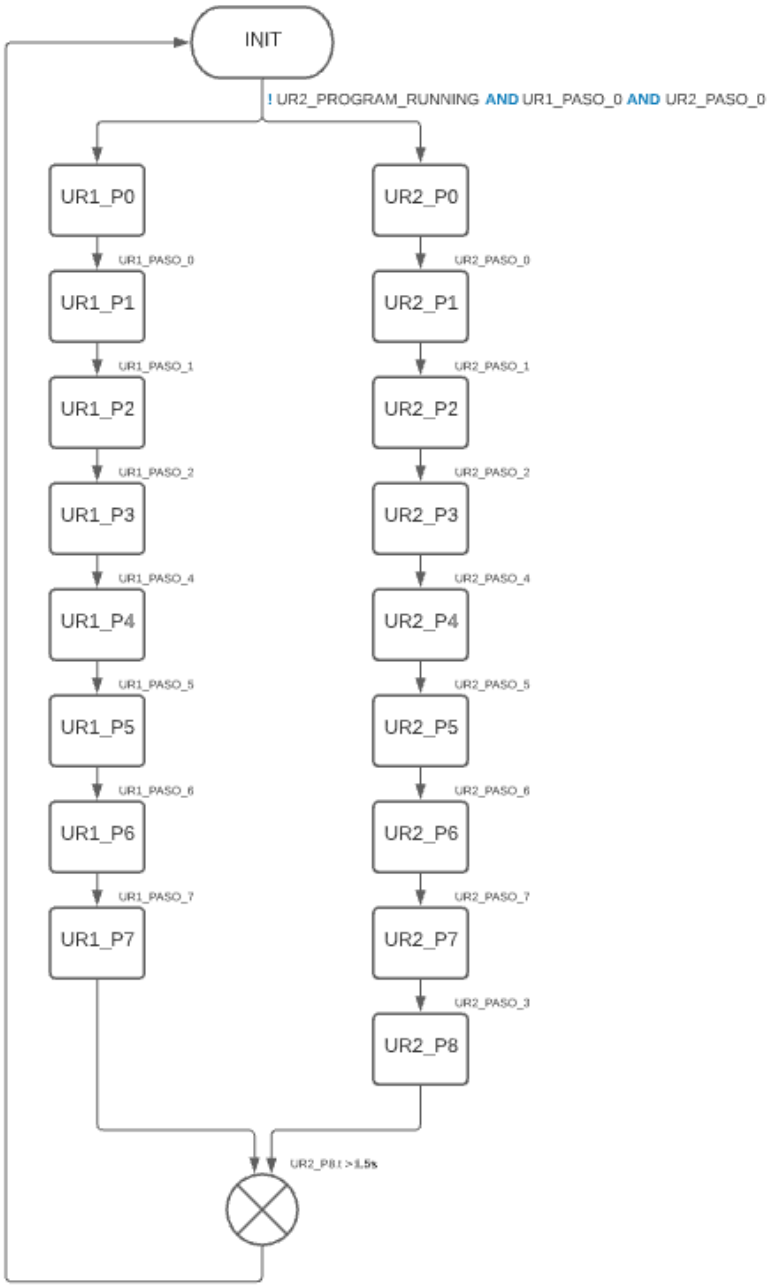



8
SEQ_7
SEQ
CALC_POS_AXIS_UR1.UR1_EJE1
CALC_POS_AXIS_UR1.UR1_EJE2
CALC_POS_AXIS_UR1.UR1_EJE3
CALC_POS_AXIS_UR1.UR1_EJE4
CALC_POS_AXIS_UR1.UR1_EJE5
92
-28
61
-123
89
UR_E1
UR_E2
UR_E3
UR_E4
UR_E5
UR_E1_SETPOINT
UR_E2_SETPOINT
UR_E3_SETPOINT
UR_E4_SETPOINT
UR_E5_SETPOINT
PASO
PASO_5_1
9
SEQ_8
SEQ
CALC_POS_AXIS_UR1.UR1_EJE1
CALC_POS_AXIS_UR1.UR1_EJE2
CALC_POS_AXIS_UR1.UR1_EJE3
CALC_POS_AXIS_UR1.UR1_EJE4
CALC_POS_AXIS_UR1.UR1_EJE5
61
-85
126
-100
73
UR_E1
UR_E2
UR_E3
UR_E4
UR_E5
UR_E1_SETPOINT
UR_E2_SETPOINT
UR_E3_SETPOINT
UR_E4_SETPOINT
UR_E5_SETPOINT
PASO
PASO_5_2
10
SEQ_9
SEQ
CALC_POS_AXIS_UR1.UR1_EJE1
CALC_POS_AXIS_UR1.UR1_EJE2
CALC_POS_AXIS_UR1.UR1_EJE3
CALC_POS_AXIS_UR1.UR1_EJE4
CALC_POS_AXIS_UR1.UR1_EJE5
80
-24
45
-116
114
UR_E1
UR_E2
UR_E3
UR_E4
UR_E5
UR_E1_SETPOINT
UR_E2_SETPOINT
UR_E3_SETPOINT
UR_E4_SETPOINT
UR_E5_SETPOINT
PASO
PASO_5_3
11
SEQ_10
SEQ
CALC_POS_AXIS_UR1.UR1_EJE1
CALC_POS_AXIS_UR1.UR1_EJE2
CALC_POS_AXIS_UR1.UR1_EJE3
CALC_POS_AXIS_UR1.UR1_EJE4
CALC_POS_AXIS_UR1.UR1_EJE5
15
-68
61
-204
46
UR_E1
UR_E2
UR_E3
UR_E4
UR_E5
UR_E1_SETPOINT
UR_E2_SETPOINT
UR_E3_SETPOINT
UR_E4_SETPOINT
UR_E5_SETPOINT
PASO
PASO_6
12
SEQ_11
SEQ
CALC_POS_AXIS_UR1.UR1_EJE1
CALC_POS_AXIS_UR1.UR1_EJE2
CALC_POS_AXIS_UR1.UR1_EJE3
CALC_POS_AXIS_UR1.UR1_EJE4
CALC_POS_AXIS_UR1.UR1_EJE5
-89
-32
-76
-70
89
UR_E1
UR_E2
UR_E3
UR_E4
UR_E5
UR_E1_SETPOINT
UR_E2_SETPOINT
UR_E3_SETPOINT
UR_E4_SETPOINT
UR_E5_SETPOINT
PASO
PASO_7
13
OR
≥1
PASO_4_1
PASO_4_2
PASO_4_3
PASO_4
14
OR
≥1
PASO_5_1
PASO_5_2
PASO_5_3
PASO_5
15

2.2.6. SEQ_ACTIVE ()

Corresponde al último programa de control de la secuencia y en el cual se implementan una serie de tareas relativas a la representación gráfica del estado del proceso, animaciones de las transiciones y etapas, asignación de valores a determinadas variables para la carga de datos relativos a las posiciones en las que se encuentran en todo momento ambos *UR,* así como la comprobación de la recepción de las señales enlazadas al bus de comunicaciones *MODBUS versión Client,* sobre la cual las máquinas virtuales *URSim* envían *I/*O para indicar el punto de trabajo de cada robot por separado. Se presenta un diagrama de flujo del programa para su mejor comprensión. (Figura 3).

Figura 3. Diagrama de flujo que representa el ciclo de trabajo de la estación, gobernado por los cambios de posición de los ejes de los UR.

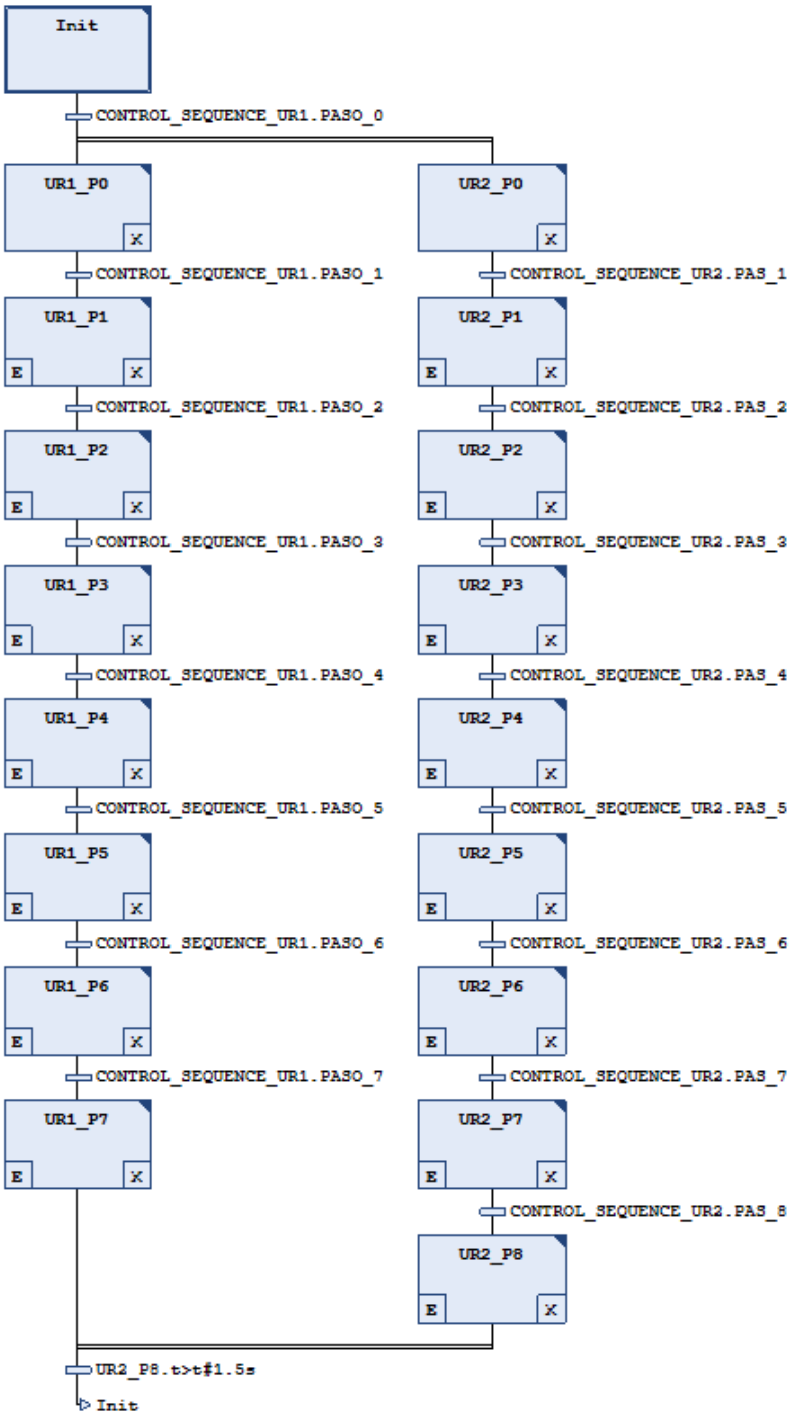


2.2.6.1 VARIABLES

```
PROGRAM SEQ_ACTIVE
VAR

    UR1_P0              :IecSfc.SFCStepType;    //ETAPA DE INICIO DE RUTINA DE UR1
    UR2_P0              :IecSfc.SFCStepType;    //ETAPA DE INICIO DE RUTINA DE UR2
    UR2_P8              :IecSfc.SFCStepType;    //ETAPA ÚLTIMA DE LA RUTINA DE UR2
    UR1_HOME            :STRING;                //ACTIVA VISUALMENTE ETAPA 0 DEL PROGRAMA UR1
    UR1_BEFORE_PICK     :STRING;                //ACTIVA VISUALMENTE ETAPA 1 DEL PROGRAMA UR1
    UR1_PICK            :STRING;                //ACTIVA VISUALMENTE ETAPA 2 DEL PROGRAMA UR1
    UR1_AFTER_PICK      :STRING;                //ACTIVA VISUALMENTE ETAPA 3 DEL PROGRAMA UR1
    UR1_DELIVERY        :STRING;                //ACTIVA VISUALMENTE ETAPA 4 DEL PROGRAMA UR1
    UR1_AFTER_DEL       :STRING;                //ACTIVA VISUALMENTE ETAPA 5 DEL PROGRAMA UR1
    UR1_BEFORE_HOME     :STRING;                //ACTIVA VISUALMENTE ETAPA 7 DEL PROGRAMA UR1
    UR1_HOME_END        :STRING;                //ACTIVA VISUALMENTE ETAPA 8 DEL PROGRAMA UR1
    UR1_T1              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 0 Y ETAPA 1
    UR1_T2              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 1 Y ETAPA 2
    UR1_T3              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 2 Y ETAPA 3
    UR1_T4              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 3 Y ETAPA 4
    UR1_T5              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 4 Y ETAPA 5
    UR1_T6              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 5 Y ETAPA 6
    UR1_T7              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 6 Y ETAPA 7
    UR1_T8              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 7 Y ETAPA 0
    UR1_P0_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E0 Y E1 EN AMARILLO PARPADEANTE
    UR1_P1_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E1 Y E2 EN AMARILLO PARPADEANTE
    UR1_P2_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E2 Y E3 EN AMARILLO PARPADEANTE
    UR1_P3_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E3 Y E4 EN AMARILLO PARPADEANTE
    UR1_P4_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E4 Y E5 EN AMARILLO PARPADEANTE
    UR1_P5_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E5 Y E6 EN AMARILLO PARPADEANTE
    UR1_P6_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E6 Y E7 EN AMARILLO PARPADEANTE
    UR1_P7_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E7 Y E0 EN AMARILLO PARPADEANTE
    UR1_P_ULTIMO        :BOOL;                  //ROBOT HOME TRAS RUTINA PERO NO ESTÁ LISTO PARA COMENZAR

    UR2_HOME            :STRING;                //ACTIVA VISUALMENTE ETAPA 0 DEL PROGRAMA UR2
    UR2_BEFORE_PICK     :STRING;                //ACTIVA VISUALMENTE ETAPA 1 DEL PROGRAMA UR2
    UR2_PICK_NO_BOX     :STRING;                //ACTIVA VISUALMENTE ETAPA 2 DEL PROGRAMA UR2
    UR2_PICK_BOX        :STRING;                //ACTIVA VISUALMENTE ETAPA 3 DEL PROGRAMA UR2
    UR2_AFTER_PICK      :STRING;                //ACTIVA VISUALMENTE ETAPA 4 DEL PROGRAMA UR2
    UR2_BEFORE_DROP     :STRING;                //ACTIVA VISUALMENTE ETAPA 5 DEL PROGRAMA UR2
    UR2_DROP            :STRING;                //ACTIVA VISUALMENTE ETAPA 6 DEL PROGRAMA UR2
    UR2_AFTER_DROP      :STRING;                //ACTIVA VISUALMENTE ETAPA 7 DEL PROGRAMA UR2
    UR2_HOME_END        :STRING;                //ACTIVA VISUALMENTE ETAPA 8 DEL PROGRAMA UR2
    UR2_T1              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 0 Y ETAPA 1
    UR2_T2              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 1 Y ETAPA 2
    UR2_T3              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 2 Y ETAPA 3
    UR2_T4              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 3 Y ETAPA 4
    UR2_T5              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 4 Y ETAPA 5
    UR2_T6              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 5 Y ETAPA 6
    UR2_T7              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 6 Y ETAPA 7
    UR2_T8              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 7 Y ETAPA 8
    UR2_T9              :STRING;                //ACTIVA VISUALMENTE LA TRANSICIÓN ENTRE ETAPA 8 Y ETAPA 0
    UR2_P0_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E0 Y E1 EN AMARILLO PARPADEANTE
    UR2_P1_T            :BOOL;                  //ACTIVA LA TRANSICIÓN ENTRE E1 Y E2 EN AMARILLO PARPADEANTE
    UR2_P2_T            :BOOL;                  //ACTIVA TRANSICIÓN ENTRE E2 Y E3 EN AMARILLO PARPADEANTE
    UR2_P3_T            :BOOL;                  //ACTIVA TRANSICIÓN ENTRE E3 Y E4 EN AMARILLO PARPADEANTE
    UR2_P4_T            :BOOL;                  //ACTIVA TRANSICIÓN ENTRE E4 Y E5 EN AMARILLO PARPADEANTE
    UR2_P5_T            :BOOL;                  //ACTIVA TRANSICIÓN ENTRE E5 Y E6 EN AMARILLO PARPADEANTE
```

```
    UR2_P6_T        :BOOL;          //ACTIVA TRANSICIÓN ENTRE E6 Y E7 EN AMARILLO PARPADEANTE
    UR2_P7_T        :BOOL;          //ACTIVA TRANSICIÓN ENTRE E7 Y E8 EN AMARILLO PARPADEANTE
    UR2_P8_T        :BOOL;          //ACTIVA TRANSICIÓN ENTRE E8 Y E0 EN AMARILLO PARPADEANTE

END_VAR
```

2.2.6.2 PROGRAMA



2.2.6.3 SEQ_AQTIVE.Init.active

```
UR1_HOME_END:='17';
UR2_HOME_END:='17';
UR1_P7_T:=FALSE;
TFG.SECUENCIA_UR1:='0';
UR1_HOME:='18';
UR1_BEFORE_PICK:='17';
UR1_PICK:='17';
UR1_AFTER_PICK:='17';
UR1_DELIVERY:='17';
UR1_AFTER_DEL:='17';
UR1_BEFORE_HOME:='17';
UR1_T2:='19';
UR1_T3:='19';
UR1_T4:='19';
UR1_T5:='19';
UR1_T6:='19';
UR1_T7:='19';
UR1_T8:='19';
TFG.SECUENCIA_UR2:='8';
UR2_HOME:='18';
UR2_BEFORE_PICK:='17';
UR2_PICK_NO_BOX:='17';
UR2_PICK_BOX:='17';
UR2_AFTER_PICK:='17';
UR2_BEFORE_DROP:='17';
UR2_DROP:='17';
UR2_AFTER_DROP:='17';
UR2_T2:='19';
UR2_T3:='19';
UR2_T4:= :='19';
UR2_T5:='19';
UR2_T6:='19';
UR2_T7:='19';
UR2_T8:='19';
UR2_T9:='19';
```

2.2.6.4 SEQ_AQTIVE.UR1_P0.active

```
IF CONTROL_SEQUENCE_UR1.PASO_0 THEN
                        UR1_P0_T:= FALSE;
                        UR1_T1:='19';
            ELSE
                        UR1_P0_T:= TRUE;
                        UR1_T1:='21';
END_IF
```

2.2.6.5 SEQ_AQTIVE.UR1_P0.exit

```
UR1_P0_T:= FALSE;
```

2.2.6.6 SEQ_AQTIVE.UR1_P1.entry

```
UR1_T1:='19';
```

2.2.6.7 SEQ_AQTIVE.UR1_P1.active

```
TFG.SECUENCIA_UR1:='1';
UR1_HOME:='17';
UR1_BEFORE_PICK:='18';
UR1_T2:='21';
UR1_P1_T:=TRUE;
```

2.2.6.8 SEQ_AQTIVE.UR1_P1.exit

```
UR1_T2:='21';
UR1_P1_T:=FALSE;
```

2.2.6.9 SEQ_AQTIVE.UR1_P2.entry

```
UR1_T2:='19';
UR1.UR1_CLOSE_CLAMP:=TRUE;
```

2.2.6.10 SEQ_AQTIVE.UR1_P2.active

```
TFG.SECUENCIA_UR1:='2';
UR1_BEFORE_PICK:='17';
UR1_PICK:='18';
UR1_T3:='21';
UR1_P2_T:=TRUE;
```

2.2.6.11 SEQ_AQTIVE.UR1_P2.exit

```
UR1_P2_T:=FALSE;
```

2.2.6.12 SEQ_AQTIVE.UR1_P3.entry

```
UR1_T3:='19';
```

2.2.6.13 SEQ_AQTIVE.UR1_P3.active

```
TFG.SECUENCIA_UR1:='3';
UR1_PICK:='17';
UR1_AFTER_PICK:='18';

IF CONTROL_SEQUENCE_UR1.PASO_3 THEN
                        UR1_P3_T:=FALSE;
                        UR1_T4:='19';
            ELSE
                        UR1_P3_T:=TRUE;
                        UR1_T4:='21';
END_IF
```

2.2.6.14 SEQ_AQTIVE.UR1_P3.exit

```
UR1_P3_T:=FALSE;
```

2.2.6.15 SEQ_AQTIVE.UR1_P4.entry

```
UR1_T4:='19';
```

2.2.6.16 SEQ_AQTIVE.UR1_P4.active

```
TFG.SECUENCIA_UR1:='4';
UR1_AFTER_PICK:='17';
UR1_DELIVERY:='18';
IF CONTROL_SEQUENCE_UR1.PASO_4 THEN
                        UR1_P4_T:=FALSE;
                        UR1_T5:='19';
            ELSE
                        UR1_P4_T:=TRUE;
                        UR1_T5:='21';
END_IF
IF CONTROL_SEQUENCE_UR1.PASO_4_1 THEN
                        UR1.UR1_RE_1:=1;
END_IF
IF CONTROL_SEQUENCE_UR1.PASO_4_2 THEN
                        UR1.UR1_RE_1:=2;
END_IF
IF CONTROL_SEQUENCE_UR1.PASO_4_3 THEN
```

```
                    UR1.UR1_RE_1:=3;
END_IF
```

2.2.6.17 SEQ_AQTIVE.UR1_P4.exit

```
UR1_P4_T:=FALSE;
```

2.2.6.18 SEQ_AQTIVE.UR1_P5.entry

```
UR1_T5:='19';
```

2.2.6.19 SEQ_AQTIVE.UR1_P5.active

```
TFG.SECUENCIA_UR1:='5';
UR1_DELIVERY:='17';
UR1_AFTER_DEL:='18';
UR1_T6:='21';
UR1_P5_T:=TRUE;
```

2.2.6.20 SEQ_AQTIVE.UR1_P5.exit

```
UR1_P5_T:=FALSE;
UR1_CLOSE_CLAMP:=FALSE;
```

2.2.6.21 SEQ_AQTIVE.UR1_P6.entry

```
UR1_T6:='19';
```

2.2.6.22 SEQ_AQTIVE.UR1_P6.active

```
TFG.SECUENCIA_UR1:='6';
UR1_AFTER_DEL:='17';
UR1_BEFORE_HOME:='18';
UR1_T7:='21';
UR1_P6_T:=TRUE;
```

2.2.6.23 SEQ_AQTIVE.UR1_P6.exit

```
UR1_P6_T:=FALSE;
```

2.2.6.24 SEQ_AQTIVE.UR1_P7.entry

```
UR1_T7:='19';
UR1_P_ULTIMO:=TRUE;
```

2.2.6.25 SEQ_AQTIVE.UR1_P7.active

```
TFG.SECUENCIA_UR1:='7';
UR1_BEFORE_HOME:='17';
UR1_HOME_END:='18';
UR1_T8:='21';
UR1_P7_T:=TRUE;
UR1_P_ULTIMO:=FALSE;
```

2.2.6.26 SEQ_AQTIVE.UR1_P7.exit

```
UR2_T7:='19';
UR2_P8_T:=FALSE;
```

2.2.6.27 SEQ_AQTIVE.UR2_P0.active

```
IF CONTROL_SEQUENCE_UR2.PAS_0 THEN
                    UR2_P0_T:=FALSE;
                    UR2_T1:='19';
          ELSE
                    UR2_P0_T:=TRUE;
                    UR2_T1:='21';
END_IF
```

2.2.6.28 SEQ_AQTIVE.UR2_P0.exit

```
UR2_P0_T:=FALSE;
```

2.2.6.29 SEQ_AQTIVE.UR2_P1.entry

```
UR2_T1:='19';
```

2.2.6.30 SEQ_AQTIVE.UR2_P1.active

```
TFG.SECUENCIA_UR2:='9';
UR2_HOME:='17';
UR2_BEFORE_PICK:='18';
UR2_T2:='21';
UR2_P1_T:=TRUE;
```

2.2.6.31 SEQ_AQTIVE.UR2_P1.exit

```
UR2_T2:='21';
UR2_P1_T:=FALSE;
```

2.2.6.32 SEQ_AQTIVE.UR2_P2.entry

```
UR2_T2:='19';
UR2_CLOSE_CLAMP:=TRUE;
```

2.2.6.33 SEQ_AQTIVE.UR2_P2.active

```
TFG.SECUENCIA_UR2:='10';
UR2_BEFORE_PICK:='17';
UR2_PICK_NO_BOX:='18';
UR2_T3:='21';
UR2_P2_T:=TRUE;
```

2.2.6.34 SEQ_AQTIVE.UR2_P2.exit

```
UR2_P2_T:=FALSE;
```

2.2.6.35 SEQ_AQTIVE.UR2_P3.entry

```
UR2_T3:='19';
```

2.2.6.36 SEQ_AQTIVE.UR2_P3.active

```
TFG.SECUENCIA_UR2:='11';
UR2_PICK_NO_BOX:='17';
UR2_PICK_BOX:='18';
UR2_T4:='21';
UR2_P3_T:=TRUE;
```

2.2.6.37 SEQ_AQTIVE.UR2_P3.exit

```
UR2_P3_T:=FALSE;
```

2.2.6.38 SEQ_AQTIVE.UR2_P4.entry

```
UR2_T4:='19';
```

2.2.6.39 SEQ_AQTIVE.UR2_P4.active

```
TFG.SECUENCIA_UR2:='12';
UR2_PICK_BOX:='17';
UR2_AFTER_PICK:='18';
UR2_T5:='21';
UR2_P4_T:=TRUE;
```

2.2.6.40 SEQ_AQTIVE.UR2_P4.exit

```
UR2_P4_T:=FALSE;
```

2.2.6.41 SEQ_AQTIVE.UR2_P5.entry

```
UR2_T5:='19';
```

2.2.6.42 SEQ_AQTIVE.UR2_P5.active

```
TFG.SECUENCIA_UR2:='13';
UR2_AFTER_PICK:='17';
UR2_BEFORE_DROP:='18';
UR2_T6:='21';
UR2_P5_T:=TRUE;
```

2.2.6.43 SEQ_AQTIVE.UR2_P5.exit

```
UR2_P5_T:=FALSE;
```

2.2.6.44 SEQ_AQTIVE.UR2_P6.entry

```
UR2_T6:='19';
UR1_CLOSE_CLAMP:=FALSE;
```

2.2.6.45 SEQ_AQTIVE.UR2_P6.active

```
TFG.SECUENCIA_UR2:='14';
UR2_BEFORE_DROP:='17';
UR2_DROP:='18';
UR2_T7:='21';
UR2_P6_T:=TRUE;
```

2.2.6.46 SEQ_AQTIVE.UR2_P6.exit

```
UR2_P6_T:=FALSE;
```

2.2.6.47 SEQ_AQTIVE.UR2_P7.entry

```
UR2_T7:='19';
```

2.2.6.48 SEQ_AQTIVE.UR2_P7.active

```
TFG.SECUENCIA_UR2:='15';
UR2_DROP:='17';
UR2_AFTER_DROP:='18';
UR2_T8:='21';
UR2_P7_T:=TRUE;
```

2.2.6.49 SEQ_AQTIVE.UR2_P7.exit

```
UR2_T7:='19';
UR2_P7_T:=FALSE;
```

2.2.6.50 SEQ_AQTIVE.UR2_P8.entry

```
UR2_T8:='19';
```

2.2.6.51 SEQ_AQTIVE.UR2_P8.active

```
TFG.SECUENCIA_UR2:='16';
UR2_AFTER_DROP:='17';
UR2_HOME_END:='18';
UR2_T9:='21';
UR2_P8_T:=TRUE;
```

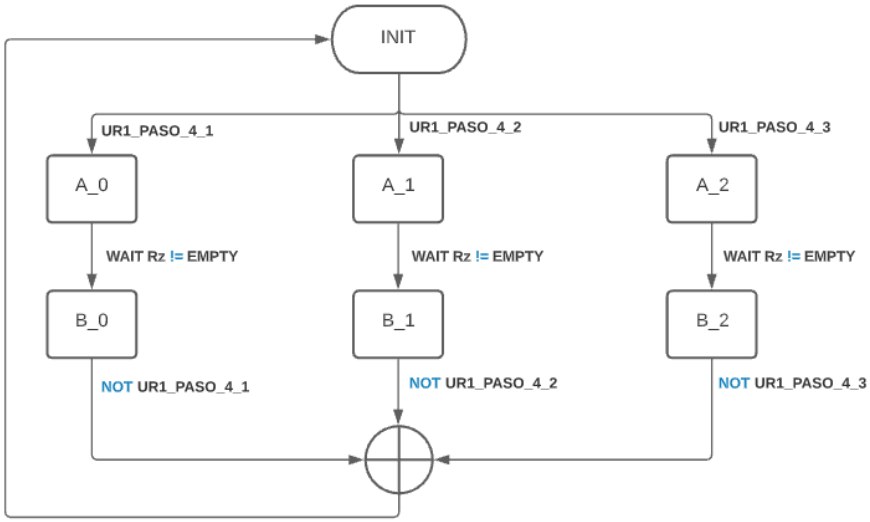
2.2.6.52 SEQ_AQTIVE.UR2_P8.exit

```
UR2_T8:='19';
UR2_P8_T:=FALSE;
```

2.2.7. CAMMERA_RoboDK_PROG ()

Constituye el repositorio del código asociado a todo el procesamiento de los elementos que intervienen en la visión artificial, las animaciones presentadas en la ventana *VISION* en el *WebServer* y el *HMI*, así como el manejo de los datos que se reciben desde *RoboDK* relativos a la posición de la pieza en el espacio de trabajo, en este programa además se lleva un control del número de piezas procesadas, en que posición se entrega, así como la representación del ciclo de medición que se encuentre activo y el anterior. (Figura 4).

Figura 4. Diagrama de flujo de la lógica para tratar la visión artificial desarrollada en el proyecto.



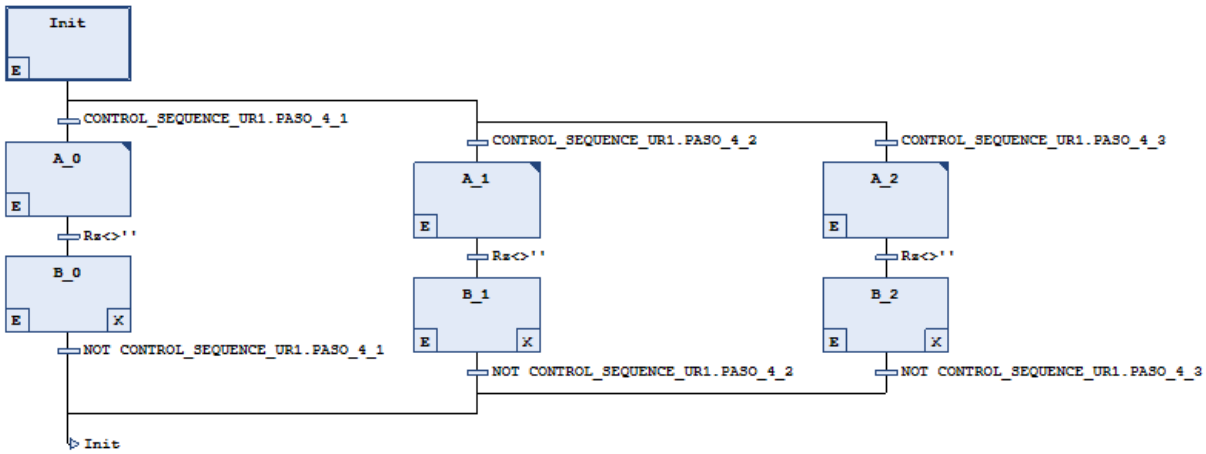
2.2.7.1 VARIABLES

```
PROGRAM CAMMERA_RoboDK_PROG
VAR

    INICIO              :IecSfc.SFCStepType;        //ETAPA DE INICIO DE LA SECUENCIA
    A_0                 :IecSfc.SFCStepType;        //ETAPA QUE ACTIVA LA MEDICIÓN EN POSICIÓN 1
    A_1                 :IecSfc.SFCStepType;        //ETAPA QUE ACTIVA LA MEDICIÓN EN POSICIÓN 2
    A_2                 :IecSfc.SFCStepType;        //ETAPA QUE ACTIVA LA MEDICIÓN EN POSICIÓN 3
    B_0                 :IecSfc.SFCStepType;        //ETAPA DE ESPERA A QUE UR1 ABANDONE LA POSICIÓN 1 DE ENTREGA
    B_1                 :IecSfc.SFCStepType;        //ETAPA DE ESPERA A QUE UR1 ABANDONE LA POSICIÓN 2 DE ENTREGA
    B_2                 :IecSfc.SFCStepType;        //ETAPA DE ESPERA A QUE UR1 ABANDONE LA POSICIÓN 3 DE ENTREGA
    ANI_CIRCULO         :INT:=0;                    //VARIABLE PARA ROTAR LA MIRA
    CAMERA_IMAGE        :STRING;                    //ALMACENA EL # DE IMAGEN A CARGAR EN LA VENTANA VISIÓN
    SQUARE_AND_LINE     :STRING;                    //IMAGEN MUESTRA EL CUADRADO CON LINEA EN LA VENTANA VISIÓN
    SHOW_IMAGE          :BOOL;                      //MUESTRA LA IMAGEN DE LA CÁMARA EN EL VISOR POR UN TIEMPO
    CAMERA_IMAGE_LAST   :STRING;                    //ALMACENA EL # DE IMAGEN A CARGAR EN LA IMAGEN ANTERIOR
    X                   :STRING;                    //VALOR DE X EN LA MEDICIÓN DE LA POSICIÓN
    Y                   :STRING;                    //VALOR DE Y EN LA MEDICIÓN DE LA POSICIÓN
    Z                   :STRING;                    //VALOR DE Z EN LA MEDICION DE LA POSICIÓN
    RX                  :STRING;                    //VALOR DE Rx EN LA MEDICIÓN DE LA POSICIÓN
    Ry                  :STRING;                    //VALOR DE Ry EN LA MEDICIÓN DE LA POSICIÓN
    Rz                  :STRING;                    //VALOR DE Rz EN LA MEDICIÓN DE LA POSICIÓN
```

```
X_LAST          :STRING;        //VALOR DE X EN LA MEDICIÓN DE LA POSICIÓN ANTERIOR
RX_LAST         :STRING;        //VALOR DE Rx EN LA MEDICIÓN DE LA POSICIÓN ANTERIOR
Y_LAST          :STRING;        //VALOR DE Y EN LA MEDICIÓN DE LA POSICIÓN ANTERIOR
Ry_LAST         :STRING;        //VALOR DE Ry EN LA MEDICIÓN DE LA POSICIÓN ANTERIOR
Z_LAST          :STRING;        //VALOR DE Z EN LA MEDICION DE LA POSICIÓN ANTERIOR
Rz_LAST         :STRING;        //VALOR DE Rz EN LA MEDICIÓN DE LA POSICIÓN ANTERIOR
CAM_ON          :BOOL;          //INDICADOR ESTADO DE LA CAMARA (ON_ENCENDIDA, OFF_APAGADA)
CAM_STANDBY     :BOOL;          //INDICADOR DE CÁMARA NO ACTIVADA
TOTAL_P         :WORD;          //TOTAL DE PIEZAS PROCESADAS
T_POS_1         :INT:=0;        //TOTAL DE PIEZAS DEJADAS EN POS_1
T_POS_2         :INT:=0;        //TOTAL DE PIEZAS DEJADAS EN POS_2
T_POS_3         :INT:=0;        //TOTAL DE PIEZAS DEJADAS EN POS_3
```

END_VAR

2.2.7.2 PROGRAMA



2.2.7.3 CAMMERA_RoboDK_PROG.Init.entry

```
ANI_CIRCULO:=0;
SHOW_IMAGE:=FALSE;
X:='';
Rx:='';
Y:='';
Ry:='';
Z:='';
Rz:='';
```

2.2.7.4 CAMMERA_RoboDK_PROG.A0.entry

```
SHOW_IMAGE:=TRUE;
CAMERA_IMAGE:='22';
SQUARE_AND_LINE:='31';
CAM_ON:=TRUE;
CAM_STANDBY:=FALSE;
T_POS_1:=T_POS_1+1;
TOTAL_P:=TOTAL_P+1;
```

2.2.7.5 CAMMERA_RoboDK_PROG.A0.active

```
ANI_CIRCULO:=ANI_CIRCULO+1;
```

2.2.7.6 CAMMERA_RoboDK_PROG.B0.entry

```
CAM_ON:=FALSE;
CAM_STANDBY:=TRUE;
```

2.2.7.7 CAMMERA_RoboDK_PROG.B0.exit

```
X_LAST:=X;
Rx_LAST:=Rx;
Y_LAST:=Y;
Ry_LAST:=Ry;
Z_LAST:=Z;
Rz_LAST:=Rz;
CAMERA_IMAGE_LAST:='35';
```

2.2.7.8 CAMMERA_RoboDK_PROG.A1.entry

```
SHOW_IMAGE:=TRUE;
CAMERA_IMAGE:='23';
SQUARE_AND_LINE:='32';
CAM_ON:=TRUE;
CAM_STANDBY:=FALSE;
T_POS_2:=T_POS_2+1;
TOTAL_P:=TOTAL_P+1;
```

2.2.7.9 CAMMERA_RoboDK_PROG.A1.active

```
ANI_CIRCULO:=ANI_CIRCULO+1;
```

2.2.7.10 CAMMERA_RoboDK_PROG.B1.entry

```
CAM_ON:=FALSE;
CAM_STANDBY:=TRUE;
```

2.2.7.11 CAMMERA_RoboDK_PROG.B1.exit

```
X_LAST:=X;
Rx_LAST:=Rx;
Y_LAST:=Y;
Ry_LAST:=Ry;
Z_LAST:=Z;
Rz_LAST:=Rz;
CAMERA_IMAGE_LAST:='36';
```

2.2.7.12 CAMMERA_RoboDK_PROG.A2.entry

```
SHOW_IMAGE:=TRUE;
CAMERA_IMAGE:='24';
SQUARE_AND_LINE:='33';
CAM_ON:=TRUE;
CAM_STANDBY:=FALSE;
T_POS_3:=T_POS_3+1;
TOTAL_P:=TOTAL_P+1;
```

2.2.7.13 CAMMERA_RoboDK_PROG.A2.active

```
ANI_CIRCULO:=ANI_CIRCULO+1;
```

2.2.7.14 CAMMERA_RoboDK_PROG.B2.entry

```
CAM_ON:=FALSE;
CAM_STANDBY:=TRUE;
```

2.2.7.15 CAMMERA_RoboDK_PROG.B2.exit

```
X_LAST:=X;
Rx_LAST:=Rx;
```

```
Y_LAST:=Y;
Ry_LAST:=Ry;
Z_LAST:=Z;
Rz_LAST:=Rz;
CAMERA_IMAGE_LAST:='37';
```

2.2.8. WEBServer_UR1 ()

Existe el correspondiente programa homólogo para el robot *UR2* con las mismas instrucciones, solamente cambia la denominación de las variables tal y como se comentó en el apartado A_2.1. TABLAS DE VARIABLES.

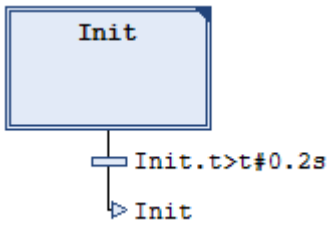
2.2.8.1 VARIABLES

```
PROGRAM WEBServer_UR1
VAR
        UR1_EMERGENCIA                  :BOOL:=FALSE;       //GUARDA LA PARADA DE EMERGENCIA EN EL ROBOT REAL
        UR1_SecurityStopped             :BOOL:=FALSE;       //PARADA DE SEGURIDAD DE UR1
        UR1_EmergencyStopped            :BOOL:=FALSE;       //PARADA DE EMERGENCIA
        UR1_isTeachButtonPressed        :BOOL:=FALSE;       //BOTON HOMBRE MUERTO DEL ROBOT
        UR1_PowerButtonPressed          :BOOL:=FALSE;       //BOTON DE ENCENDIDO PRESIONADO
        UR1_VEL1 AT             %MW12   :REAL:=0;           //VARIABLE PARA VELOCIDAD EJE 1
        UR1_VEL2 AT             %MW14   :REAL:=0;           //VARIABLE PARA VELOCIDAD EJE 2
        UR1_VEL3 AT             %MW16   :REAL:=0;           //VARIABLE PARA VELOCIDAD EJE 3
        UR1_VEL4 AT             %MW18   :REAL:=0;           //VARIABLE PARA VELOCIDAD EJE 4
        UR1_VEL5 AT             %MW20   :REAL:=0;           //VARIABLE PARA VELOCIDAD EJE 5
        UR1_VEL6 AT             %MW22   :REAL:=0;           //VARIABLE PARA VELOCIDAD EJE 1
        UR1_mA_1        AT      %MW24   :REAL:=0;           //VARIABLE PARA LA CORRIENTE EJE 1
        UR1_mA_2        AT      %MW26   :REAL:=0;           //VARIABLE PARA LA CORRIENTE EJE 2
        UR1_mA_3        AT      %MW28   :REAL:=0;           //VARIABLE PARA LA CORRIENTE EJE 3
        UR1_mA_4        AT      %MW30   :REAL:=0;           //VARIABLE PARA LA CORRIENTE EJE 4
        UR1_mA_5        AT      %MW32   :REAL:=0;           //VARIABLE PARA LA CORRIENTE EJE 5
        UR1_mA_6        AT      %MW34   :REAL:=0;           //VARIABLE PARA LA CORRIENTE EJE 6
        UR1_TEMP1                       :REAL:=0;           //VARIABLE PARA TEMPERATURA EJE 1
        UR1_TEMP2                       :REAL:=0;           //VARIABLE PARA TEMPERATURA EJE 2
        UR1_TEMP3                       :REAL:=0;           //VARIABLE PARA TEMPERATURA EJE 3
        UR1_TEMP4                       :REAL:=0;           //VARIABLE PARA TEMPERATURA EJE 4
        UR1_TEMP5                       :REAL:=0;           //VARIABLE PARA TEMPERATURA EJE 5
        UR1_TEMP6                       :REAL:=0;           //VARIABLE PARA TEMPERATURA EJE 6
        UR1_DIAG1                       :WORD:=0;           //DIAGNÓSTICO DE LA BASE DEL ROBOT
        UR1_DIAG2                       :WORD:=0;           //DIAGNÓSTICO DEL HOMBRO DEL ROBOT
        UR1_DIAG3                       :WORD:=0;           //DIAGNÓSTICO DEL CODO DEL ROBOT
        UR1_DIAG4                       :WORD:=0;           //DIAGNÓSTICO DE LA MUÑECA 1
        UR1_DIAG5                       :WORD:=0;           //DIAGNÓSTICO DE LA MUÑECA 2
        UR1_DIAG6                       :WORD:=0;           //DIAGNÓSTICO DE LA MUÑECA 3
        UR1_mA                          :WORD:=0;           //CORRIENTE CONSUMIDA POR EL ROBOT
        UR1_IO_CURRENT                  :WORD:=0;           //CORRIENTE CONSUMIDA POR EL MÓDULO DE IO
        UR1_VER_HN                      :WORD:=0;           //VERSION CONTROLADOR NUMERO PRINCIPAL
        UR1_VER_LN                      :WORD:=0;           //VERSION CONTROLADOR NUMERO SECUNDARIO
        TCP_x                           :REAL:=0;           //POSICIÓN x DEL TCP RESPECTO A LA BASE DEL ROBOT
        TCP_y                           :REAL:=0;           //POSICIÓN y DEL TCP RESPECTO A LA BASE DEL ROBOT
        TCP_z                           :REAL:=0;           //POSICIÓN z DEL TCP RESPECTO A LA BASE DEL ROBOT
        TCP_Rx                          :REAL:=0;           //ROTACION Rx DEL TCP RESPECTO A LA BASE DEL ROBOT
        TCP_Ry                          :REAL:=0;           //ROTACION Ry DEL TCP RESPECTO A LA BASE DEL ROBOT
        TCP_Rz                          :REAL:=0;           //ROTACION Rz DEL TCP RESPECTO A LA BASE DEL ROBOT
        TCP_VEL_x                       :REAL:=0;           //VELOCIDAD x DEL TCP RESPECTO A LA BASE DEL ROBOT
        TCP_VEL_y                       :REAL:=0;           //VELOCIDAD y DEL TCP RESPECTO A LA BASE DEL ROBOT
```

```
TCP_VEL_z                   :REAL:=0;           //VELOCIDAD z DEL TCP RESPECTO A LA BASE DEL ROBOT
TCP_VEL_Rx                  :REAL:=0;           //VELOCIDAD ROTACION Rx DEL TCP A LA BASE DEL ROBOT
TCP_VEL_Ry                  :REAL:=0;           //VELOCIDAD ROTACION Ry DEL TCP A LA BASE DEL ROBOT
TCP_VEL_Rz                  :REAL:=0;           //VELOCIDAD ROTACION Rz DEL TCP A LA BASE DEL ROBOT
TCP_OFFSET_x                :REAL:=0;           //POSICIÓN x DEL TCP A LA BRIDA DEL ROBOT
TCP_OFFSET_y                :REAL:=0;           //POSICIÓN y DEL TCP A LA BRIDA DEL ROBOT
TCP_OFFSET_z                :REAL:=0;           //POSICIÓN z DEL TCP RESPECTO A LA BRIDA DEL ROBOT
TCP_OFFSET_Rx               :REAL:=0;           //ROTACION Rx DEL TCP RESPECTO A LA BRIDA DEL ROBOT
TCP_OFFSET_Ry               :REAL:=0;           //ROTACION Ry DEL TCP RESPECTO A LA BRIDA DEL ROBOT
TCP_OFFSET_Rz               :REAL:=0;           //ROTACION Rz DEL TCP RESPECTO A LA BRIDA DEL ROBOT
TCP_STATE                   :WORD;              //ESTADO DEL TCP DEL ROBOT
TCP_TEMP                    :WORD;              //TEMPERATURA DE LA HERRAMIENTA
TCP_CURRENT                 :WORD;              //CORRIENTE CONSUMIDA POR EL TCP
PI                          :REAL:=3.14159265;  //CONSTANTE MATEMÁTICA
```

```
END_VAR
```

2.2.8.2 PROGRAMA



2.2.8.3 WEBServer_UR1.Init.active

```
/// ASIGNACIÓN DE TEMPERATURA ENTRE OTROS DATOS DESDE LOS REGISTROS DEL SLAVE AL HMI ///

UR1_TEMP1:=WORD_TO_REAL(UR1.UR1_TEMP_EJE1);
UR1_TEMP2:=WORD_TO_REAL(UR1.UR1_TEMP_EJE2);
UR1_TEMP3:=WORD_TO_REAL(UR1.UR1_TEMP_EJE3);
UR1_TEMP4:=WORD_TO_REAL(UR1.UR1_TEMP_EJE4);
UR1_TEMP5:=WORD_TO_REAL(UR1.UR1_TEMP_EJE5);
UR1_TEMP6:=WORD_TO_REAL(UR1.UR1_TEMP_EJE6);
UR1_mA:=UR1.UR1_CURRENT;
UR1_IO_CURRENT:=UR1.UR1_IO_CURRENT;
UR1_VER_HN:=UR1.UR1_CONT_VER_HN;
UR1_VER_LN:=UR1.UR1_CONT_VER_LN;

TCP_STATE:=UR1.UR1_TCP_STATE;
TCP_TEMP:=UR1.UR1_TCP_TEMP;
TCP_CURRENT:=UR1.UR1_TCP_CURRENT;

UR1.UR1_RS_EJES_OFF:=UR1_DIAG1.3 OR UR1_DIAG2.3 OR UR1_DIAG3.3 OR UR1_DIAG4.3 OR UR1_DIAG5.3 OR UR1_DIAG6.3;
UR1.UR1_RS_EJES_ON:=UR1_DIAG1.11 OR UR1_DIAG2.11 OR UR1_DIAG3.11 OR UR1_DIAG4.11 OR UR1_DIAG5.11 OR UR1_DIAG6.11;
UR1.UR1_RS_EJES_IDLE:=UR1_DIAG1.12 OR UR1_DIAG2.12 OR UR1_DIAG3.12 OR UR1_DIAG4v12 OR UR1_DIAG5.12 OR UR1_DIAG6.12;

/// DETERMINAR CORRIENTE CONSUMIDA POR CADA EJE EN UR1 ///

UR1_mA_1:=WORD_TO_REAL(UR1.UR1_CURRENT_EJE1)/1000;
UR1_mA_2:=WORD_TO_REAL(UR1.UR1_CURRENT_EJE2)/1000;
UR1_mA_3:=WORD_TO_REAL(UR1.UR1_CURRENT_EJE3)/1000;
UR1_mA_4:=WORD_TO_REAL(UR1.UR1_CURRENT_EJE4)/1000;
UR1_mA_5:=WORD_TO_REAL(UR1.UR1_CURRENT_EJE5)/1000;
```

```
UR1_mA_6:=WORD_TO_REAL(UR1.UR1_CURRENT_EJE6)/1000;

                                        /// DETERMINAR SI EXISTE PARADA DE EMERGENCIA EN UR1 ///

IF UR1.UR1_isSafetySignalSuchThatWeShouldStop = 1 THEN
        UR1_EMERGENCIA:=TRUE;
ELSE
        UR1_EMERGENCIA:=FALSE;
END_IF

                                        /// DETERMINAR SI EXISTE PARADA DE SEGURIDAD EN UR1 ///

IF UR1.UR1_isSecurityStopped = 1 THEN
        UR1_SecurityStopped:=TRUE;
ELSE
        UR1_SecurityStopped:=FALSE;
END_IF

                                        /// DETERMINAR SI EXISTE PARADA DE EMERGENCIA EN UR1 ///

IF UR1.UR1_isEmergencyStopped = 1 THEN
        UR1_EmergencyStopped:=TRUE;
ELSE
        UR1_EmergencyStopped:=FALSE;
END_IF

                                /// DETERMINAR SI SE PRESIONA EL BOTON DE HOMBRE MUERTO EN UR1 ///

IF UR1.UR1_isTeachButtonPressed = 1 THEN
        UR1_isTeachButtonPressed:=TRUE;
ELSE
        UR1_isTeachButtonPressed:=FALSE;
END_IF

                                        /// DETERMINAR SI SE PRESIONA EL BOTON DE ENCENDIDO EN UR1 ///

IF UR1.UR1_isPowerButtonPressed = 1 THEN
        UR1_PowerButtonPressed:=TRUE;
ELSE
        UR1_PowerButtonPressed:=FALSE;
END_IF

                                        /// DETERMINAR EL ESTADO DE FUNCIONAMIENTO DE UR1 ///

CASE UR1.UR1_ROBOT_MODE OF
0:      HMI.UR1_RS:=0;
        UR1.UR1_RS_DISCONNECTED:=TRUE;
        UR1.UR1_RS_CONFIRM_SAFETY:=FALSE;
        UR1.UR1_RS_BOOTING:=FALSE;
        UR1.UR1_RS_POWER_OFF:=FALSE;
        UR1.UR1_RS_POWER_ON:=FALSE;
        UR1.UR1_RS_IDLE:=FALSE;
        UR1.UR1_RS_BACKDRIVE:=FALSE;
        UR1.UR1_RS_RUNNING:=FALSE;
1:      HMI.UR1_RS:=0;
        UR1.UR1_RS_DISCONNECTED:=FALSE;
        UR1.UR1_RS_CONFIRM_SAFETY:=TRUE;
        UR1.UR1_RS_BOOTING:=FALSE;
        UR1.UR1_RS_POWER_OFF:=FALSE;
```

```
            UR1.UR1_RS_POWER_ON:=FALSE;
            UR1.UR1_RS_IDLE:=FALSE;
            UR1.UR1_RS_BACKDRIVE:=FALSE;
            UR1.UR1_RS_RUNNING:=FALSE;
2:          HMI.UR1_RS:=0;
            UR1.UR1_RS_DISCONNECTED:=FALSE;
            UR1.UR1_RS_CONFIRM_SAFETY:=FALSE;
            UR1.UR1_RS_BOOTING:=TRUE;
            UR1.UR1_RS_POWER_OFF:=FALSE;
            UR1.UR1_RS_POWER_ON:=FALSE;
            UR1.UR1_RS_IDLE:=FALSE;
            UR1.UR1_RS_BACKDRIVE:=FALSE;
            UR1.UR1_RS_RUNNING:=FALSE;
3:          HMI.UR1_RS:=0;
            UR1.UR1_RS_DISCONNECTED:=FALSE;
            UR1.UR1_RS_CONFIRM_SAFETY:=FALSE;
            UR1.UR1_RS_BOOTING:=FALSE;
            UR1.UR1_RS_POWER_OFF:=TRUE;
            UR1.UR1_RS_POWER_ON:=FALSE;
            UR1.UR1_RS_IDLE:=FALSE;
            UR1.UR1_RS_BACKDRIVE:=FALSE;
            UR1.UR1_RS_RUNNING:=FALSE;
4:          HMI.UR1_RS:=0;
            UR1.UR1_RS_DISCONNECTED:=FALSE;
            UR1.UR1_RS_CONFIRM_SAFETY:=FALSE;
            UR1.UR1_RS_BOOTING:=FALSE;
            UR1.UR1_RS_POWER_OFF:=FALSE;
            UR1.UR1_RS_POWER_ON:=TRUE;
            UR1.UR1_RS_IDLE:=FALSE;
            UR1.UR1_RS_BACKDRIVE:=FALSE;
            UR1.UR1_RS_RUNNING:=FALSE;
5:          HMI.UR1_RS:=1;
            UR1.UR1_RS_DISCONNECTED:=FALSE;
            UR1.UR1_RS_CONFIRM_SAFETY:=FALSE;
            UR1.UR1_RS_BOOTING:=FALSE;
            UR1.UR1_RS_POWER_OFF:=FALSE;
            UR1.UR1_RS_POWER_ON:=FALSE;
            UR1.UR1_RS_IDLE:=TRUE;
            UR1.UR1_RS_BACKDRIVE:=FALSE;
            UR1.UR1_RS_RUNNING:=FALSE;
6:          HMI.UR1_RS:=0;
            UR1.UR1_RS_DISCONNECTED:=FALSE;
            UR1.UR1_RS_CONFIRM_SAFETY:=FALSE;
            UR1.UR1_RS_BOOTING:=FALSE;
            UR1.UR1_RS_POWER_OFF:=FALSE;
            UR1.UR1_RS_POWER_ON:=FALSE;
            UR1.UR1_RS_IDLE:=FALSE;
            UR1.UR1_RS_BACKDRIVE:=TRUE;
            UR1.UR1_RS_RUNNING:=FALSE;
7:          HMI.UR1_RS:=2;
            UR1.UR1_RS_DISCONNECTED:=FALSE;
            UR1.UR1_RS_CONFIRM_SAFETY:=FALSE;
            UR1.UR1_RS_BOOTING:=FALSE;
            UR1.UR1_RS_POWER_OFF:=FALSE;
            UR1.UR1_RS_POWER_ON:=FALSE;
            UR1.UR1_RS_IDLE:=FALSE;
            UR1.UR1_RS_BACKDRIVE:=FALSE;
            UR1.UR1_RS_RUNNING:=TRUE;
END_CASE
```

```
IF UR1.UR1_ROBOT_MODE=3 THEN
        UR1.UR1_RS_POWER_OFF:=TRUE;
        UR1.UR1_RS_RUNNING:=FALSE;
        UR1.UR1_RS_IDLE:=FALSE;
ELSIF
        UR1.UR1_ROBOT_MODE=5 THEN
        UR1.UR1_RS_POWER_OFF:=FALSE;
        UR1.UR1_RS_RUNNING:=FALSE;
        UR1.UR1_RS_IDLE:=TRUE;
ELSIF
        UR1.UR1_ROBOT_MODE=7 THEN
        UR1.UR1_RS_POWER_OFF:=FALSE;
        UR1.UR1_RS_RUNNING:=TRUE;
        UR1.UR1_RS_IDLE:=FALSE;
END_IF

                        /// DETERMINAR EL ESTADO DE LA HERRAMIENTA PARA VISUALIZACION ///

IF UR1.UR1_TCP_STATE=253 THEN
        UR1.UR1_TCP:=TRUE;
ELSE
        UR1.UR1_TCP:=FALSE;
END_IF

                                /// CALCULO DE VELOCIDAD DE LAS ARTICULACIONES UR1 ///

                /// UR1 EJE1 BASE ///
IF UR1.UR1_VEL_EJE1<6000 THEN
        UR1_VEL1:=UR1.UR1_VEL_EJE1*((180)/(1000*PI));
ELSE
        UR1_VEL1:=5.9;
END_IF
                /// UR1 EJE2 HOMBRO ///
IF UR1.UR1_VEL_EJE2<6000 THEN
        UR1_VEL2:=UR1.UR1_VEL_EJE2*((180)/(1000*PI));
ELSE
        UR1_VEL2:=5.9;
END_IF
                /// UR1 EJE3 CODO ///
IF UR1.UR1_VEL_EJE3<6000 THEN
        UR1_VEL3:=UR1.UR1_VEL_EJE3*((180)/(1000*PI));
ELSE
        UR1_VEL3:=5.9;
END_IF
                /// UR1 EJE4 MUÑECA 1 ///
IF UR1.UR1_VEL_EJE4<6000 THEN
        UR1_VEL4:=UR1.UR1_VEL_EJE4*((180)/(1000*PI));
ELSE
        UR1_VEL4:=11.46;
END_IF
                /// UR1 EJE5 MUÑECA 2 ///
IF UR1.UR1_VEL_EJE5<6000 THEN
        UR1_VEL5:=UR1.UR1_VEL_EJE5*((180)/(1000*PI));
ELSE
        UR1_VEL5:=11.46;
END_IF
                /// UR1 EJE1 MUÑECA 3 ///
IF UR1.UR1_VEL_EJE6<6000 THEN
```

```
		UR1_VEL6:=UR1.UR1_VEL_EJE6*((180)/(1000*PI));
ELSE
		UR1_VEL6:=11.46;
END_IF

                                                /// DIAGNÓSTICO DE CADA EJE UR1 ///

                        /// UR1 DIAGNÓSTICO EJE 1 ///
IF UR1.UR1_ROBOT_MODE<>5 THEN
CASE UR1.UR1_DIAG_EJE1 OF
		236: UR1_DIAG1:=1;
		237: UR1_DIAG1:=2;
		238: UR1_DIAG1:=4;
		239: UR1_DIAG1:=8;
		245: UR1_DIAG1:=16;
		246: UR1_DIAG1:=32;
		247: UR1_DIAG1:=64;
		248: UR1_DIAG1:=128;
		249: UR1_DIAG1:=256;
		250: UR1_DIAG1:=512;
		252: UR1_DIAG1:=1024;
		253: UR1_DIAG1:=2048;
                        //255: UR1_DIAG1:=4096; EXISTE UN BUG EN URSim, NO ENVÍA ESTE CÓDIGO
END_CASE
ELSE
		UR1_DIAG1:=4096;
END_IF

                        /// UR1 DIAGNÓSTICO EJE 2 ///
IF UR1.UR1_ROBOT_MODE<>5 THEN
CASE UR1.UR1_DIAG_EJE2 OF
		236: UR1_DIAG2:=1;
		237: UR1_DIAG2:=2;
		238: UR1_DIAG2:=4;
		239: UR1_DIAG2:=8;
		245: UR1_DIAG2:=16;
		246: UR1_DIAG2:=32;
		247: UR1_DIAG2:=64;
		248: UR1_DIAG2:=128;
		249: UR1_DIAG2:=256;
		250: UR1_DIAG2:=512;
		252: UR1_DIAG2:=1024;
		253: UR1_DIAG2:=2048;
                        //255: UR1_DIAG1:=4096; EXISTE UN BUG EN URSim, NO ENVÍA ESTE CÓDIGO
END_CASE
ELSE
		UR1_DIAG2:=4096;
END_IF

                        /// UR1 DIAGNÓSTICO EJE 3 ///
IF UR1.UR1_ROBOT_MODE<>5 THEN
CASE UR1.UR1_DIAG_EJE3 OF
		236: UR1_DIAG3:=1;
		237: UR1_DIAG3:=2;
		238: UR1_DIAG3:=4;
		239: UR1_DIAG3:=8;
		245: UR1_DIAG3:=16;
		246: UR1_DIAG3:=32;
		247: UR1_DIAG3:=64;
```

```
            248: UR1_DIAG3:=128;
            249: UR1_DIAG3:=256;
            250: UR1_DIAG3:=512;
            252: UR1_DIAG3:=1024;
            253: UR1_DIAG3:=2048;
                        //255: UR1_DIAG1:=4096; EXISTE UN BUG EN URSim, NO ENVÍA ESTE CÓDIGO
END_CASE
ELSE
            UR1_DIAG3:=4096;
END_IF

                        /// UR1 DIAGNÓSTICO EJE 4 ///
IF UR1.UR1_ROBOT_MODE<>5 THEN
CASE UR1.UR1_DIAG_EJE4 OF
            236: UR1_DIAG4:=1;
            237: UR1_DIAG4:=2;
            238: UR1_DIAG4:=4;
            239: UR1_DIAG4:=8;
            245: UR1_DIAG4:=16;
            246: UR1_DIAG4:=32;
            247: UR1_DIAG4:=64;
            248: UR1_DIAG4:=128;
            249: UR1_DIAG4:=256;
            250: UR1_DIAG4:=512;
            252: UR1_DIAG4:=1024;
            253: UR1_DIAG4:=2048;
                        //255: UR1_DIAG1:=4096; EXISTE UN BUG EN URSim, NO ENVÍA ESTE CÓDIGO
END_CASE
ELSE
            UR1_DIAG4:=4096;
END_IF

                        /// UR1 DIAGNÓSTICO EJE 5 ///
IF UR1.UR1_ROBOT_MODE<>5 THEN
CASE UR1.UR1_DIAG_EJE5 OF
            236: UR1_DIAG5:=1;
            237: UR1_DIAG5:=2;
            238: UR1_DIAG5:=4;
            239: UR1_DIAG5:=8;
            245: UR1_DIAG5:=16;
            246: UR1_DIAG5:=32;
            247: UR1_DIAG5:=64;
            248: UR1_DIAG5:=128;
            249: UR1_DIAG5:=256;
            250: UR1_DIAG5:=512;
            252: UR1_DIAG5:=1024;
            253: UR1_DIAG5:=2048;
                        //255: UR1_DIAG1:=4096; EXISTE UN BUG EN URSim, NO ENVÍA ESTE CÓDIGO
END_CASE
ELSE
            UR1_DIAG5:=4096;
END_IF

                        /// UR1 DIAGNÓSTICO EJE 6 ///
IF UR1.UR1_ROBOT_MODE<>5 THEN
CASE UR1.UR1_DIAG_EJE6 OF
            236: UR1_DIAG6:=1;
            237: UR1_DIAG6:=2;
            238: UR1_DIAG6:=4;
```

```
            239: UR1_DIAG6:=8;
            245: UR1_DIAG6:=16;
            246: UR1_DIAG6:=32;
            247: UR1_DIAG6:=64;
            248: UR1_DIAG6:=128;
            249: UR1_DIAG6:=256;
            250: UR1_DIAG6:=512;
            252: UR1_DIAG6:=1024;
            253: UR1_DIAG6:=2048;
                  //255: UR1_DIAG1:=4096; EXISTE UN BUG EN URSim, NO ENVÍA ESTE CÓDIGO
END_CASE
ELSE
            UR1_DIAG6:=4096;
END_IF

            /// CALCULO DE LA POSICION DEL TCP ///
            /// POSICION TCP_x ///
IF UR1.UR1_TCP_x<6000 THEN
            TCP_x:=(WORD_TO_REAL(UR1.UR1_TCP_x)/10);
ELSE
            TCP_x:=(65535-WORD_TO_REAL(UR1.UR1_TCP_x))/10;
END_IF

            /// POSICION TCP_y ///
TCP_y:=((65535-WORD_TO_REAL(UR1.UR1_TCP_y))/(-10));
            /// POSICION TCP_z ///
TCP_z:=(WORD_TO_REAL(UR1.UR1_TCP_z)/10);

            /// POSICION TCP_Rx ///
IF UR1.UR1_TCP_Rx>6000 THEN
            TCP_Rx:=(65535-WORD_TO_REAL(UR1.UR1_TCP_Rx))/(-1000);
ELSE
            TCP_Rx:=WORD_TO_REAL(UR1.UR1_TCP_Rx)/1000;
END_IF

            /// POSICION TCP_Ry ///
IF UR1.UR1_TCP_Ry>6000 THEN
            TCP_Ry:=(65535-WORD_TO_REAL(UR1.UR1_TCP_Ry))/(-1000);
ELSE
            TCP_Ry:=WORD_TO_REAL(UR1.UR1_TCP_Ry)/1000;
END_IF

            /// POSICION TCP_Rz ///
IF UR1.UR1_TCP_Rz>6000 THEN
            TCP_Rz:=(65535-WORD_TO_REAL(UR1.UR1_TCP_Rz))/(-1000);
ELSE
            TCP_Rz:=WORD_TO_REAL(UR1.UR1_TCP_Rz)/1000;
END_IF

            /// CALCULO DE VELOCIDAD DEL TCP DEL ROBOT ///
            /// VELOCIDAD TCP_x ///
IF UR1.UR1_TCP_VEL_x<6000 THEN
            TCP_VEL_x:=UR1.UR1_TCP_VEL_x;
ELSE
            TCP_VEL_x:=65535-(UR1.UR1_TCP_VEL_x);
END_IF

            /// VELOCIDAD TCP_y ///
IF UR1.UR1_TCP_VEL_y<6000 THEN
```

```
            TCP_VEL_y:=UR1.UR1_TCP_VEL_y;
ELSE
            TCP_VEL_y:=65535-(UR1.UR1_TCP_VEL_y);
END_IF

            /// VELOCIDAD TCP_z ///
IF UR1.UR1_TCP_VEL_z<6000 THEN
            TCP_VEL_z:=UR1.UR1_TCP_VEL_z;
ELSE
            TCP_VEL_z:=65535-(UR1.UR1_TCP_VEL_z);
END_IF

            /// VELOCIDAD TCP_Rx ///
IF UR1.UR1_TCP_VEL_Rx<6000 THEN
            TCP_VEL_Rx:=UR1.UR1_TCP_VEL_Rx*((180)/(1000*PI));
ELSE
            TCP_VEL_Rx:=(65535-(UR1.UR1_TCP_VEL_Rx))*((180)/(1000*PI));
END_IF

            /// VELOCIDAD TCP_Ry ///
IF UR1.UR1_TCP_VEL_Ry<6000 THEN
            TCP_VEL_Ry:=UR1.UR1_TCP_VEL_Ry*((180)/(1000*PI));
ELSE
            TCP_VEL_Ry:=(65535-(UR1.UR1_TCP_VEL_Ry))*((180)/(1000*PI));
END_IF
            /// VELOCIDAD TCP_Rz ///
IF UR1.UR1_TCP_VEL_Rz<6000 THEN
            TCP_VEL_Rz:=UR1.UR1_TCP_VEL_Rz*((180)/(1000*PI));
ELSE
            TCP_VEL_Rz:=(65535-(UR1.UR1_TCP_VEL_Rz))*((180)/(1000*PI));
END_IF

            /// CALCULO DE LA POSICION DEL TCP OFFSET///
            /// POSICION TCP__OFFSET_x ///
IF UR1.UR1_TCP_OFFSET_x<6000 THEN
            TCP_OFFSET_x:=(WORD_TO_REAL(UR1.UR1_TCP_OFFSET_x));
ELSE
            TCP_OFFSET_x:=(65535-WORD_TO_REAL(UR1.UR1_TCP_OFFSET_x));
END_IF

            /// POSICION TCP__OFFSET_y ///
IF UR1.UR1_TCP_OFFSET_y<6000 THEN
            TCP_OFFSET_y:=(WORD_TO_REAL(UR1.UR1_TCP_OFFSET_y));
ELSE
            TCP_OFFSET_y:=(65535-WORD_TO_REAL(UR1.UR1_TCP_OFFSET_y));
END_IF

            /// POSICION TCP__OFFSET_z ///
IF UR1.UR1_TCP_OFFSET_z<6000 THEN
            TCP_OFFSET_z:=(WORD_TO_REAL(UR1.UR1_TCP_OFFSET_z));
ELSE
            TCP_OFFSET_z:=(65535-WORD_TO_REAL(UR1.UR1_TCP_OFFSET_z));
END_IF

            /// POSICION TCP__OFFSET_Rx ///
IF UR1.UR1_TCP_OFFSET_Rx<6000 THEN
            TCP_OFFSET_Rx:=(WORD_TO_REAL(UR1.UR1_TCP_OFFSET_Rx));
ELSE
            TCP_OFFSET_Rx:=(65535-WORD_TO_REAL(UR1.UR1_TCP_OFFSET_Rx));
```

```
END_IF

        /// POSICION TCP__OFFSET_Ry ///
IF UR1.UR1_TCP_OFFSET_Ry<6000 THEN
        TCP_OFFSET_Ry:=(WORD_TO_REAL(UR1.UR1_TCP_OFFSET_Ry));
ELSE
        TCP_OFFSET_Ry:=(65535-WORD_TO_REAL(UR1.UR1_TCP_OFFSET_Ry));
END_IF

        /// POSICION TCP__OFFSET_Rz ///
IF UR1.UR1_TCP_OFFSET_Rz<6000 THEN
        TCP_OFFSET_Rz:=(WORD_TO_REAL(UR1.UR1_TCP_OFFSET_Rz));
ELSE
        TCP_OFFSET_Rz:=(65535-WORD_TO_REAL(UR1.UR1_TCP_OFFSET_Rz));
END_IF
```

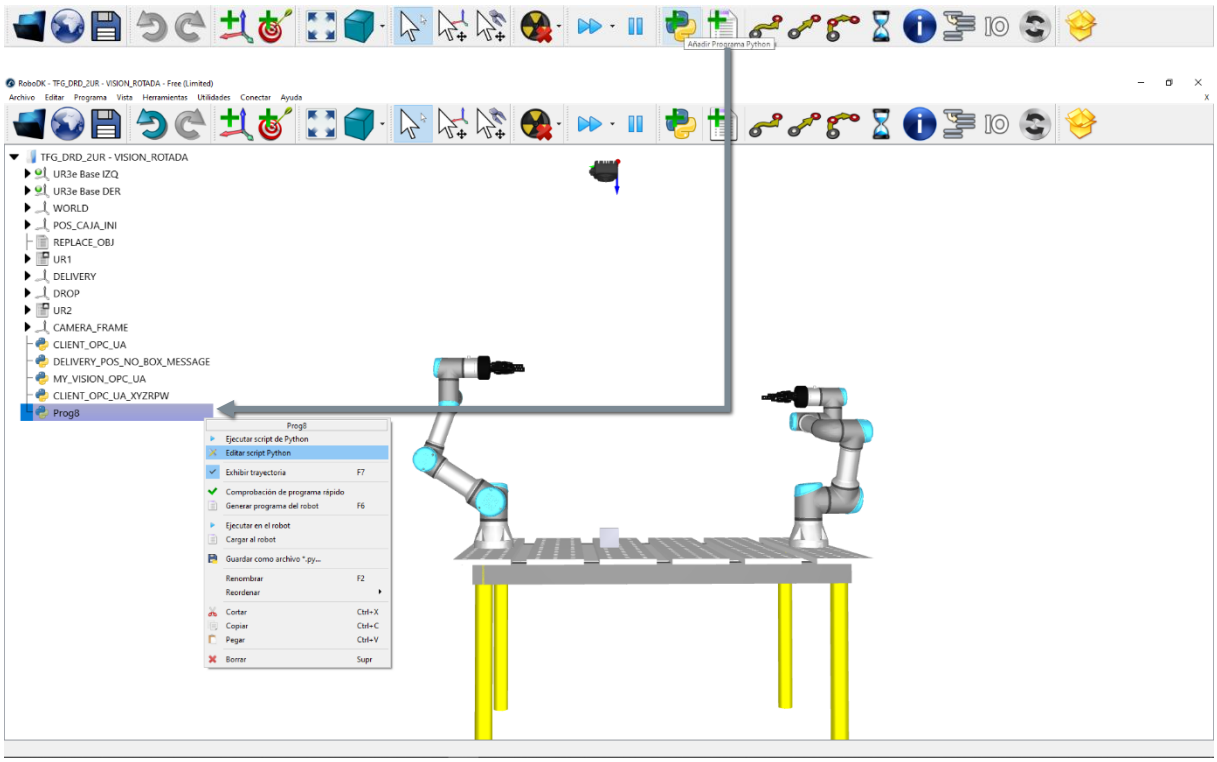
2.2.9. WEBServer PROGRAMACION GRAFICA

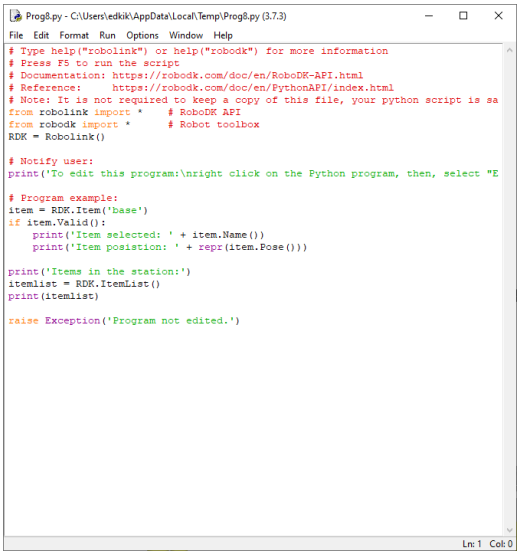
Se omite del manual de programación todo el código en el diseño del WEBServer debido a su extensión, existen en el proyecto 64 pantallas creadas, cada una con múltiples elementos que incorporan lógica de control y es imposible incluir en el presente documento, puede consultarla directamente en el fichero fuente entregado.

2.3. PROGRAMAS PYTHON – RoboDK

El presente apartado expone al lector la implementación del de los programas empleados para el envío y recepción de datos empleando *OPC UA* entre el *softPLC* de *CODESYS* y *RoboDK*. Para crear un programa *Python* debe presionar sobre el icono correspondiente en la barra de acceso a las herramientas de la aplicación (Figura 5) y posteriormente dirigirse al árbol del proyecto a la izquierda de la aplicación y sobre el programa creado *clic derecho* y seleccionar *Editar script Python*, en la ventana emergente que se muestra puede pegar los códigos suministrados a continuación:

Figura 5. Secuencia para definir un nuevo programa Python en RoboDK .





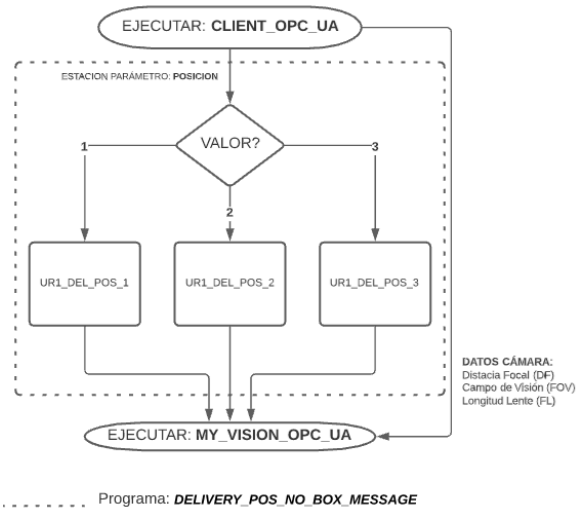
2.3.1. CLIENT_OPC_UA.py

```
# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Documentation: https://robodk.com/doc/en/RoboDK-API.html
# Reference:     https://robodk.com/doc/en/PythonAPI/index.html
# Note: It is not required to keep a copy of this file, your python script is saved with the station
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox
PARAM_POS = 'POSICION'
PARAM_DF = 'DF'
PARAM_FOV = 'FOV'
PARAM_FL = 'FL'

RDK = Robolink()

from opcua import ua, Client
import time
import logging
import sys

logging.basicConfig(level=logging.INFO)
_logger = logging.getLogger('opcua')

if __name__ == "__main__":
    client = Client("opc.tcp://CPOHDELL5820:4840")
    #client.set_user('admin')
    #client.set_password('3127fb3a')
    #client.set_security_string("Basic256Sha256,SignAndEncrypt,certificate-example.der,private-key-example.pem")

    try:
        client.connect()
        root = client.get_root_node()
        _logger.info('Objects node is: %r', root)

        objects = client.get_objects_node()
        node_DEL_POS = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.UR1.UR1_DELIVERY_POS")
        node_DF = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.TFG.CAMERA_DF")
        node_FOV = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.TFG.CAMERA_FOV")
        node_FL = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.TFG.CAMERA_FL")

        RDK.setParam(PARAM_POS, node_DEL_POS.get_value())
        RDK.setParam(PARAM_DF, node_DF.get_value())
        RDK.setParam(PARAM_FOV, node_FOV.get_value())
        RDK.setParam(PARAM_FL, node_FL.get_value())

    finally:
        client.disconnect()
```

2.3.2. DELIVERY_POS_NO_BOX_MESSAGE.py

Este programa solo analiza el valor almacenado en el parámetro *POSICION* de la estación y de acuerdo con el número contenido (1, 2 ó 3) ejecuta una sencilla lógica para decidir si *UR1* entrega la pieza en la posición asociada a cada número. La (Figura 6) muestra un diagrama de flujo con el principio de funcionamiento de este programa.

Figura 6. Diagrama de flujo de la lógica implementada para la discriminación de la posición de entrega de la pieza.



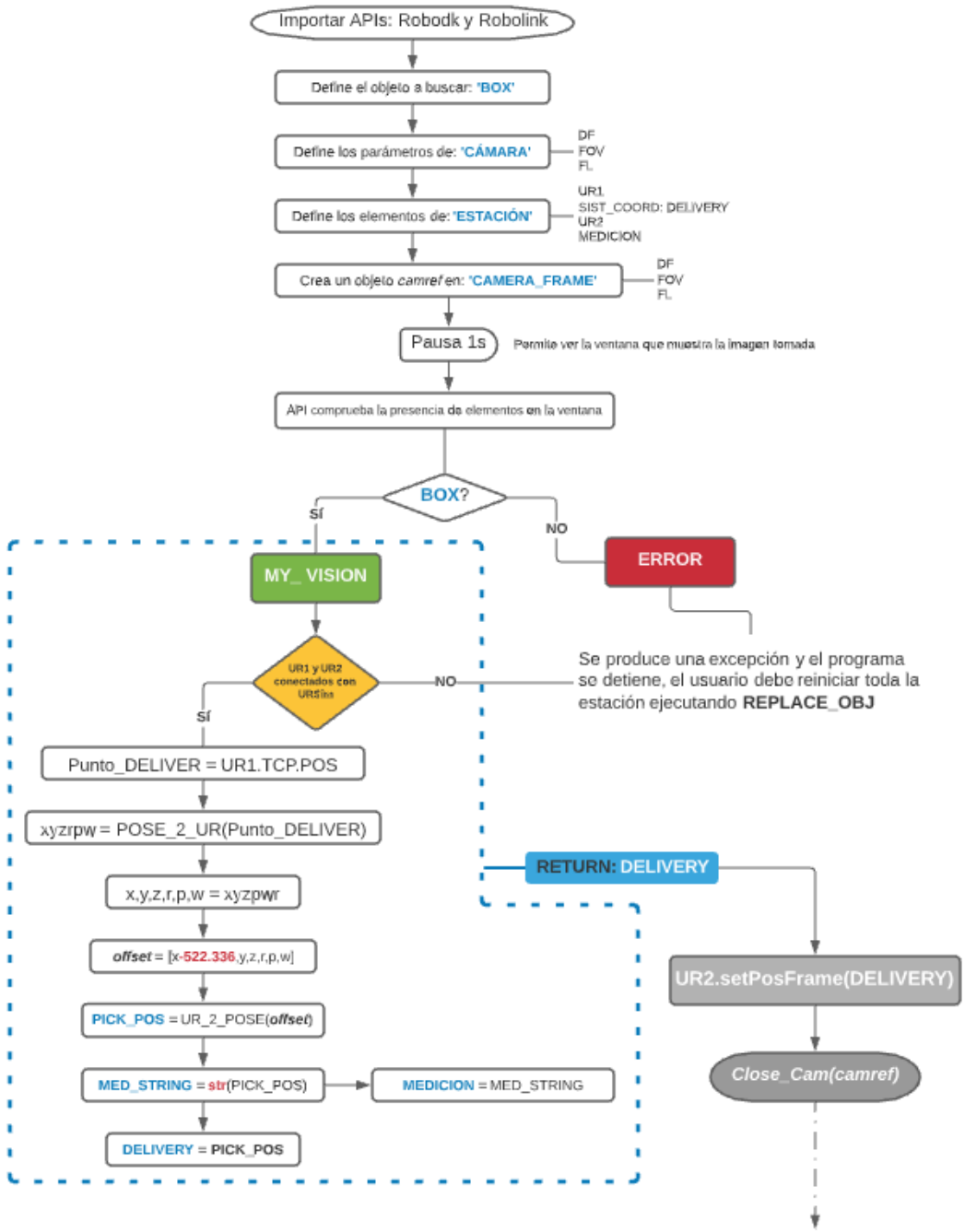
PROGRAMA:

```
# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Documentation: https://robodk.com/doc/en/RoboDK-API.html
# Reference:    https://robodk.com/doc/en/PythonAPI/index.html
# Note: It is not required to keep a copy of this file, your python script is saved with the station
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox

PARAM_POS = 'POSICION'

RDK = Robolink()

POS_DEL = RDK.getParam(PARAM_POS)

robot       = RDK.Item('UR_IZQ', ITEM_TYPE_ROBOT)
frame_robot  = RDK.Item('UR3e Base IZQ', ITEM_TYPE_FRAME)
target_DEL_1 = RDK.Item('DELIVERY_POINT_1', ITEM_TYPE_TARGET)
target_DEL_2 = RDK.Item('DELIVERY_POINT_2', ITEM_TYPE_TARGET)
target_DEL_3 = RDK.Item('DELIVERY_POINT_3', ITEM_TYPE_TARGET)


robot.setPoseFrame(frame_robot)
POS_DEL = RDK.getParam(PARAM_POS)

if POS_DEL == 1:
   robot.MoveJ(target_DEL_1)
elif POS_DEL == 2:
   robot.MoveJ(target_DEL_2)
else:
   robot.MoveJ(target_DEL_3)
```

2.3.3. MY_VISION_OPC_UA.py

Programa encargado de desarrollar toda la lógica de visión artificial. Con ayuda de las *API* de *RoboDK* se activa una cámara interna en la estación enlazada a un sistema de coordenadas para ajustar la posición de origen de la visión, en el caso de este proyecto, el sistema de coordenadas es *CAMERA_FRAME* en el cual se fijó la cámara *MATROX Iris GTR*, realmente solo representa un modelo gráfico del dispositivo real del cual toma su geometría y denominación. Con los parámetros recibidos por *OPC UA* para la configuración de la visión y los pasos que se presentan en el siguiente diagrama de flujo que describe el funcionamiento del presente programa. (Figura 7).

Figura 7. Diagrama de flujo que describe la lógica del programa de visión artificial.



PROGRAMA:

```
# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Documentation: https://robodk.com/doc/en/RoboDK-API.html
# Reference:    https://robodk.com/doc/en/PythonAPI/index.html
# Note: It is not required to keep a copy of this file, your python script is saved with the station
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox

#USE ROBODK API AS RL
RDK = Robolink()

PART_KEYWORD = 'box'

#DEFINE CAMERA PARAMS
DISTANCIA_FOCAL = 'DF'
CAMPO_VISION = 'FOV'
LONGITUD_LENTE = 'FL'
PARAMETRO_POSE = 'POSE'


#DEFINE ROBOT, TOOLs, FRAMEs, TARGETs
robot_IZQ         = RDK.Item('UR_IZQ', ITEM_TYPE_ROBOT)
robot_DER         = RDK.Item('UR_DER', ITEM_TYPE_ROBOT)
frame_delivery    = RDK.Item('DELIVERY', ITEM_TYPE_FRAME)

#close any Camera on Robodk
RDK.Cam2D_Close()

#GET CAMERA PARAMS
FOCAL_LEN = str(RDK.getParam(DISTANCIA_FOCAL))
FOV_1 = str(RDK.getParam(CAMPO_VISION))
FAR_LEN = str(RDK.getParam(LONGITUD_LENTE))

#SETTING CAMERA FRAME
camref           = RDK.Item('CAMERA_FRAME',ITEM_TYPE_FRAME)
#cam_id = RDK.Cam2D_Add(camref, 'FOCAL_LENGHT=6 FOV=32 FAR_LENGHT=500 SIZE=640x480 BG_COLOR=black')
cam_id = RDK.Cam2D_Add(camref, 'FOCAL_LENGHT='+FOCAL_LEN+ ' FOV='+FOV_1+ ' FAR_LENGHT='+FAR_LEN)

#GIVING REAL MEASUREMENT FEELING
pause(1)

#SEARCH BOX ON WINDOW
all_objects       = RDK.ItemList(ITEM_TYPE_OBJECT, True)

# Get object items in a list (faster) and filter by keyword
check_objects = []
for i in range(len(all_objects)):
   if all_objects[i].count(PART_KEYWORD) > 0:
      check_objects.append(RDK.Item(all_objects[i]))

# Make sure that there is at least one part that we are expecting
if len(check_objects) == 0:
   raise Exception('No parts to check for. Name at least one part with the name: %s.' % PICKABLE_OBJECTS_KEYWORD)

#SIMULATES THE BEHAVIOR OF THE CAMERA
def MY_VISION():
```

```
  """Simulate camera detection"""
  if RDK.RunMode() == RUNMODE_RUN_ROBOT:
    # Simulate the camera by waiting for an object to be detected
    for box in check_objects:
      punto_delivery = robot_IZQ.Pose()
      xyzrpw = Pose_2_UR(punto_delivery)
      x,y,z,r,p,w, = xyzrpw
      cogida = [x-522.336,y,z,r,p,w]
      PICK = UR_2_Pose(cogida)
      T=str(PICK)
      RDK.setParam(PARAMETRO_POSE,T)
      frame_delivery.setPose(PICK)
    return frame_delivery
  else:
    RDK.RunProgram('MY_VISION')
  return 0,0,0

#-----------------------------------------------------------------------------
frame_delivery = MY_VISION()

robot_DER.setPoseFrame(frame_delivery)
                                        #   CODE TO GET SNAPSHOT
                                        #import datetime
                                        #date_str = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
                                        #file_name = RDK.getParam('PATH_OPENSTATION') + "/Image_" + date_str + ".png"
                                        #print("Saving camera snapshot to the file:" + file_name)
                                        #RDK.Cam2D_Snapshot(file_name)
#CLOSING CAMERA WINDOW
RDK.Cam2D_Close()
```

2.3.4. CLIENT_OPC_UA_XYZRPW.py

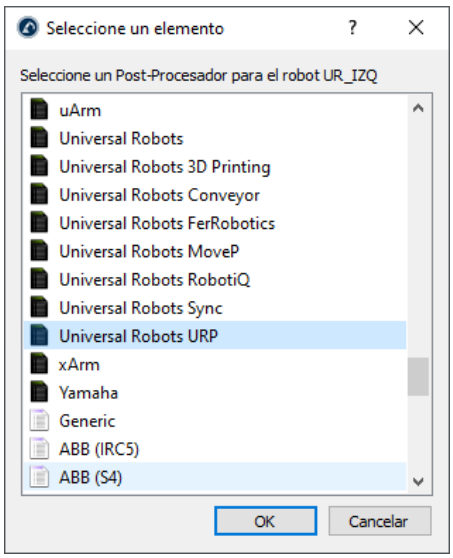
```
# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Documentation: https://robodk.com/doc/en/RoboDK-API.html
# Reference:     https://robodk.com/doc/en/PythonAPI/index.html
# Note: It is not required to keep a copy of this file, your python script is saved with the station
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox

#PARAMS CELL DECLARATION
PARAM_P = 'POSE'
PARAM_STRING='STRING'
PARAM_X='X'
PARAM_Y='Y'
PARAM_Z='Z'
PARAM_Rx='R'
PARAM_Ry='P'
PARAM_Rz='W'

RDK = Robolink()

from opcua import ua, Client
import time
import logging
import sys

logging.basicConfig(level=logging.INFO)
_logger = logging.getLogger('opcua')

if __name__ == "__main__":
    client = Client("opc.tcp://CPOHDELL5820:4840")

    try:
        client.connect()
        root = client.get_root_node()
        _logger.info('Objects node is: %r', root)

        objects = client.get_objects_node()
        node_x = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.CAMMERA_RoboDK_PROG.X")
        node_y = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.CAMMERA_RoboDK_PROG.Y")
        node_z = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.CAMMERA_RoboDK_PROG.Z")
        node_Rx = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.CAMMERA_RoboDK_PROG.RX")
        node_Ry = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.CAMMERA_RoboDK_PROG.Ry")
        node_Rz = client.get_node("ns=4;s=|var|CODESYS Control Win V3 x64.Application.CAMMERA_RoboDK_PROG.Rz")

        A=str(RDK.getParam(PARAM_P))
        buscar=A.find(")")
        B=A[5:buscar]
        RDK.setParam(PARAM_STRING,B)
        A=str(RDK.getParam(PARAM_STRING))
        buscar=A.find(",")
        B=A[:buscar]
        A=A[buscar+1:]
        RDK.setParam(PARAM_X,B)
        X=str(RDK.getParam(PARAM_X))
        node_x.set_value(X)
        RDK.setParam(PARAM_STRING,A)
        A=str(RDK.getParam(PARAM_STRING))
```

```
    buscar=A.find(",")
    B=A[1:buscar]
    A=A[buscar+1:]
    RDK.setParam(PARAM_Y,B)
    RDK.setParam(PARAM_STRING,A)
    Y=str(RDK.getParam(PARAM_Y))
    node_y.set_value(Y)
    A=str(RDK.getParam(PARAM_STRING))
    buscar=A.find(",")
    B=A[1:buscar]
    A=A[buscar+1:]
    RDK.setParam(PARAM_Z,B)
    RDK.setParam(PARAM_STRING,A)
    Q=str(RDK.getParam(PARAM_Z))
    node_z.set_value(Q)
    A=str(RDK.getParam(PARAM_STRING))
    buscar=A.find(",")
    B=A[1:buscar]
    A=A[buscar+1:]
    RDK.setParam(PARAM_Rx,B)
    RDK.setParam(PARAM_STRING,A)
    R=str(RDK.getParam(PARAM_Rx))
    node_Rx.set_value(R)
    A=str(RDK.getParam(PARAM_STRING))
    buscar=A.find(",")
    B=A[1:buscar]
    A=A[buscar+1:]
    RDK.setParam(PARAM_Ry,B)
    RDK.setParam(PARAM_STRING,A)
    P=str(RDK.getParam(PARAM_Ry))
    node_Ry.set_value(P)
    A=str(RDK.getParam(PARAM_STRING))
    buscar=A.find(",")
    B=A[1:buscar]
    A=A[buscar+1:]
    RDK.setParam(PARAM_Rz,B)
    RDK.setParam(PARAM_STRING,A)
    W=str(RDK.getParam(PARAM_Rz))
    node_Rz.set_value(W)

finally:
    client.disconnect()
```

2.4. PROGRAMAS URP – UNIVERSAL ROBOT

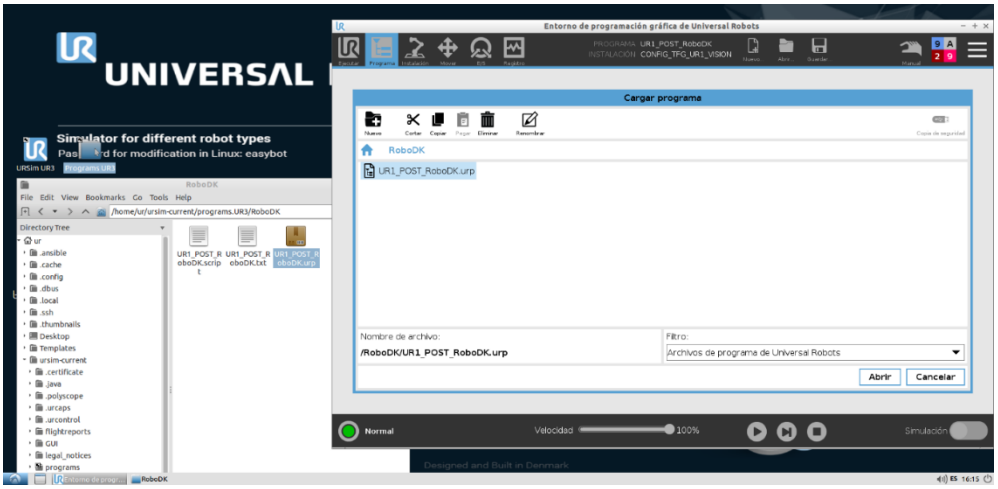
Para insertar los programas *UR1_POST_RoboDK.urp* y *UR2_POST_RoboDK.urp* dentro de las máquinas virtuales de *UR1* y *UR2* respectivamente, es necesario generar dichos programas empleando un post-procesador de *Universal Robot* en *RoboDK*. Para lograr este paso, se debe presionar *clic derecho* sobre el programa *UR1*, posteriormente *clic izquierdo* en la opción *Seleccionar Post Procesador* y active *Universal Robot URP* (Figura 8).

*Figura 8. Selección del post-procesador por defecto para los robots de **Universal Robot.***



Presione nuevamente *clic derecho* sobre *UR1* y seleccione *Generar Programa de robot*, por defecto *RoboDK* emplea *Visual Studio Code* para editar el código generado, guarde el programa en la carpeta que tiene configurada como compartida para el intercambio de ficheros entre la máquina virtual de *Universal Robot* y el *Host*, posteriormente, desde el navegador de archivos de *Linux* acceda a la ubicación anterior y copie el archivo generado en *RoboDK* y péguelo en la carpeta que contiene los programas de *UR3e*. Solo debe acceder al *IDE URSim UR3e* y en la consola cargar el programa. (Figura 5).

Figura 5. Carga del programa UR1.urp en Polyscope.



2.4.1. UR1_POST_RoboDK.urp

```
def UR1():
 # Global parameters:
 global speed_ms    = 0.250
 global speed_rads  = 0.750
 global accel_mss   = 1.200
 global accel_radss = 1.200
 global blend_radius_m = 0.000

 #---------------------------
 # Add any default subprograms here
 # For example, to drive a gripper as a program call:
 # def Gripper_Open():
 #   ...
 # end
 #
 # Example to drive a spray gun:
 def SprayOn(value):
  # use the value as an output:
  DO_SPRAY = 5
  if value == 0:
   set_standard_digital_out(DO_SPRAY, False)
  else:
   set_standard_digital_out(DO_SPRAY, True)
  end
 end

 # Example to synchronize 2
 def Synchronize():
  # Use the following digital output to signal the state of the robot:
  DO_SYNC = 1

  # Use the following digital input to get the state of another robot:
  DI_SYNC = 1

  if (get_standard_digital_out(DO_SYNC) == get_standard_digital_in(DI_SYNC)):
   set_standard_digital_out(DO_SYNC, not (get_standard_digital_out(DI_SYNC)))
   sleep(0.1)
   thread Thread_wait_1():
    while (True):
     sleep(0.01)
    end
   end
   if (get_standard_digital_out(DO_SYNC) != get_standard_digital_in(DI_SYNC)):
    global thread_handler_1=run Thread_wait_1()
    while (get_standard_digital_out(DO_SYNC) != get_standard_digital_in(DI_SYNC)):
     sync()
    end
    kill thread_handler_1
   end
  else:
   if (get_standard_digital_out(DO_SYNC) != get_standard_digital_in(DI_SYNC)):
    set_standard_digital_out(DO_SYNC, not (get_standard_digital_out(DO_SYNC)))
   end
  end
 end

 #
```

```
# Example to move an external axis
def MoveAxis(value):
 # use the value as an output:
 DO_AXIS_1 = 1
 DI_AXIS_1 = 1
 if value <= 0:
  set_standard_digital_out(DO_AXIS_1, False)

  # Wait for digital input to change state
  #while (get_standard_digital_in(DI_AXIS_1) != False):
  #  sync()
  #end
 else:
  set_standard_digital_out(DO_AXIS_1, True)

  # Wait for digital input to change state
  #while (get_standard_digital_in(DI_AXIS_1) != True):
  #  sync()
  #end
 end
end
#-------------------------


# Main program:
# Program generated by RoboDK v5.2.1 for UR_IZQ on 27/05/2021 12:08:50
# Using nominal kinematics.
# Using Ref. POS_CAJA_INI: p[0.023491, -0.375935, 0.000000, 0.000000, 0.000000, 0.000000]
# set_reference(p[0.023491, -0.375935, 0.000000, 0.000000, 0.000000, 0.000000])
while (get_standard_digital_in(10) != False):
 sync()
end
# Using TCP UR_IZQ_Opened: p[0.000000, 0.000000, 0.130000, 0.000000, 0.000000, 0.000000]
set_tcp(p[0.000000, 0.000000, 0.130000, 0.000000, 0.000000, 0.000000])
movej([-1.179038, -1.795071, -1.315474, -1.601877, 1.570716, 0.391758],1.20000,0.25000,0,0.0000)
sleep(0.500)
movel(p[0.013491, -0.375935, 0.073866, 0.000000, -3.141505, -0.000000],accel_mss,speed_ms,0,0.000)
sleep(0.500)
movel(p[0.013491, -0.375935, 0.053866, 0.000000, -3.141505, 0.000000],accel_mss,speed_ms,0,0.000)
sleep(0.500)
# Adjuntar a UR_IZQ_Closed
# Ocultar UR_IZQ_Opened
# Mostrar UR_IZQ_Closed
sleep(0.500)
movel(p[0.013497, -0.375935, 0.142957, 0.000000, -3.141505, 0.000000],accel_mss,speed_ms,0,0.000)
# Using Ref. UR3e Base IZQ: p[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000]
# set_reference(p[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000])
CLIENT_OPC_UA()
DELIVERY_POS_NO_BOX_MESSAGE()
MY_VISION_OPC_UA()
CLIENT_OPC_UA_XYZRPW()
set_standard_digital_out(1, True)
# Detach from UR_IZQ_Closed
while (get_standard_digital_in(2) != True):
 sync()
end
sleep(0.500)
# Ocultar UR_IZQ_Closed
# Mostrar UR_IZQ_Opened
```

```
 sleep(0.500)
 # Using Ref. DELIVERY: p[0.158491, -0.470081, 0.478098, -0.000000, 0.000000, 3.141593]
 # set_reference(p[0.158491, -0.470081, 0.478098, -0.000000, 0.000000, 3.141593])
 movel(p[0.156114, -0.470081, 0.374190, -0.018466, -0.000000, -3.141536],accel_mss,speed_ms,0,0.000)
 set_standard_digital_out(1, False)
 sleep(0.500)
 # Using Ref. UR3e Base IZQ: p[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000]
 # set_reference(p[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000])
 movej([0.271442, -1.187834, 1.072329, -3.560751, 0.809409, -1.412629],1.20000,0.25000,0,0.0000)
 sleep(0.500)
 movej([-1.567655, -0.575435, -1.332559, -1.233599, 1.567655, 0.000000],1.20000,0.25000,0,0.0000)
 # End of main program
end

UR1()
```

2.4.2. UR2_POST_RoboDK.urp

```
def UR2():
 # Global parameters:
 global speed_ms    = 0.250
 global speed_rads  = 0.750
 global accel_mss   = 1.200
 global accel_radss = 1.200
 global blend_radius_m = 0.001

 #---------------------------
 # Add any default subprograms here
 # For example, to drive a gripper as a program call:
 # def Gripper_Open():
 #   ...
 # end
 #
 # Example to drive a spray gun:
 def SprayOn(value):
  # use the value as an output:
  DO_SPRAY = 5
  if value == 0:
   set_standard_digital_out(DO_SPRAY, False)
  else:
   set_standard_digital_out(DO_SPRAY, True)
  end
 end

 # Example to drive an extruder:
 def Extruder(value):
  # use the value as an output:
  if value < 0:
   # stop extruder
  else:
   # start extruder
  end
 end

 # Example to move an external axis
 def MoveAxis(value):
  # use the value as an output:
  DO_AXIS_1 = 1
```

```
  DI_AXIS_1 = 1
  if value <= 0:
    set_standard_digital_out(DO_AXIS_1, False)

    # Wait for digital input to change state
    #while (get_standard_digital_in(DI_AXIS_1) != False):
    #  sync()
    #end
  else:
    set_standard_digital_out(DO_AXIS_1, True)

    # Wait for digital input to change state
    #while (get_standard_digital_in(DI_AXIS_1) != True):
    #  sync()
    #end
  end
end
#------------------------
 # Main program:
# Program generated by RoboDK v5.2.1 for UR_DER on 27/05/2021 12:11:14
# Using nominal kinematics.
while (get_standard_digital_in(1) != True):
  sync()
end
set_tool_digital_out(0, True)
set_tcp(p[0.000000, 0.000000, 0.130000, 0.000000, 0.000000, 0.000000])
movej([-1.662077, -1.852143, -1.139002, -0.144164, 1.662252, 1.582315],accel_radss,speed_rads,0,0)
sleep(0.500)
movel(p[0.001348, 0.022364, 0.025100, 1.190627, 1.221004, -1.190627],accel_mss,speed_ms,0,0.000)
# Show UR_DER_Closed
# Hide UR_DER_Opened
# Attach to UR_DER_Closed
set_standard_digital_out(2, True)
while (get_standard_digital_in(1) != False):
  sync()
end
movel(p[-0.001510, 0.155756, 0.025100, 1.190622, 1.221007, -1.190622],accel_mss,speed_ms,0,0.000)
sleep(0.500)
# set_reference(p[-0.014219, -0.212430, 0.000000, 0.000000, 0.000000, 3.116390])
movej([-0.974573, -1.166784, -1.393539, -2.152061, 1.570796, 0.621421],accel_radss,speed_rads,0,blend_radius_m)
sleep(0.500)
movel([-0.974592, -1.019272, -2.294061, -1.399056, 1.570796, 0.621337],accel_mss,speed_ms,0,0)
movel([-0.974592, -1.106190, -2.489538, -1.116662, 1.570796, 0.621337],accel_mss,speed_ms,0,0)
sleep(0.500)
# Hide UR_DER_Closed
# Show UR_DER_Opened
# Detach from UR_DER_Closed
sleep(0.500)
movel([-0.974592, -1.019272, -2.294061, -1.399056, 1.570796, 0.621337],accel_mss,speed_ms,0,0)
sleep(0.500)
# set_reference(p[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000])
movej([-3.032736, -1.182324, -1.779381, -0.179895, 3.032736, 0.000000],accel_radss,speed_rads,0,0)
sleep(1.500)
REPLACE_OBJ()
# End of main program
end
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

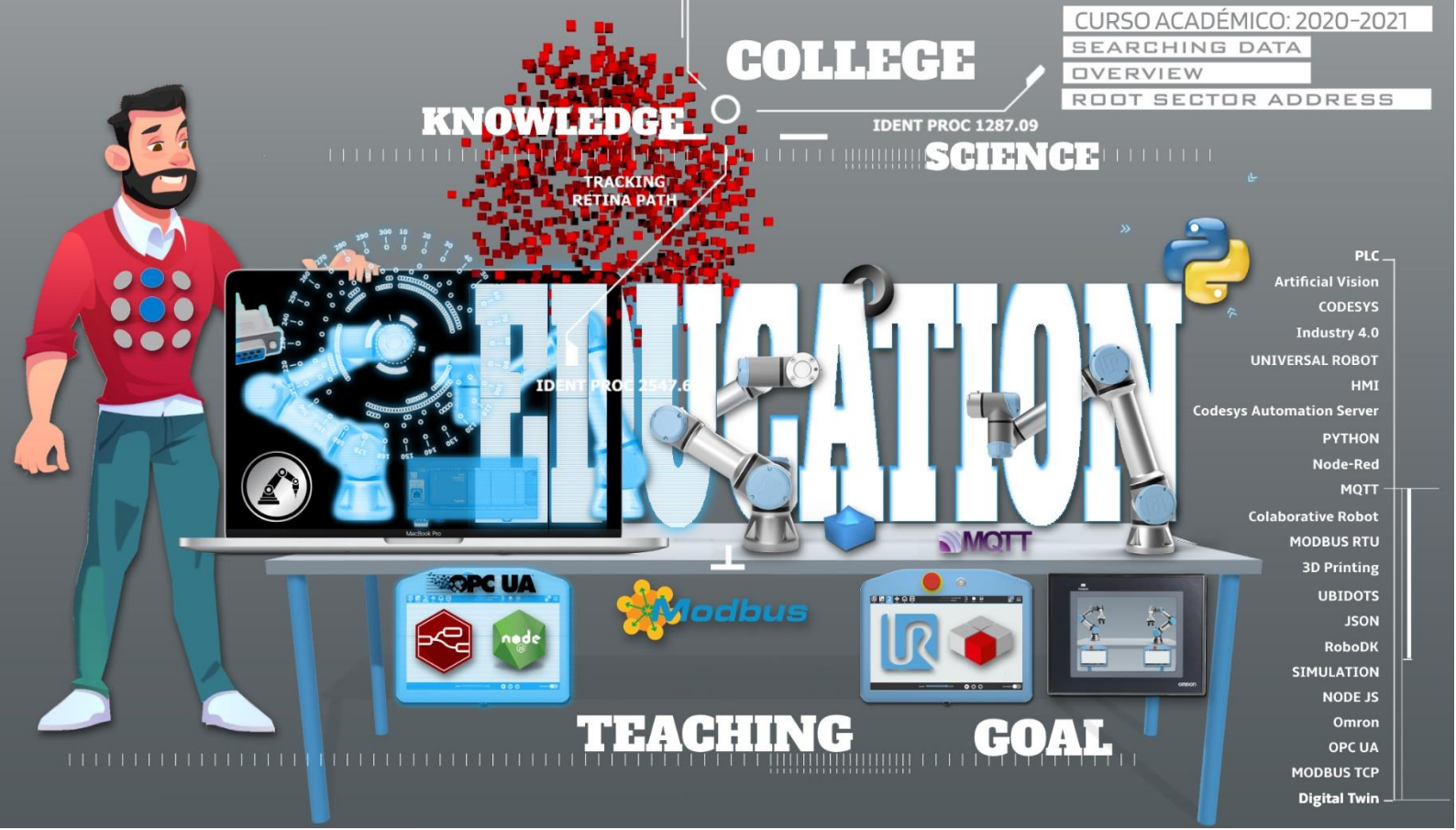
Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA

PLIEGO DE CONDICIONES

Dayron Rodríguez Díaz



Contenido

1. OBJETO

El presente documento constituye las especificaciones técnicas que se deben cumplir para asegurar el nivel de calidad, fiabilidad y seguridad en el uso de la plataforma que garantiza el correcto funcionamiento de los distintos elementos involucrados en la resolución de la investigación.

La plataforma denominada ***DICAR*** (Diseño e Implementación de una Celda Automatizada con Robótica Colaborativa) constituye una herramienta para la futura creación de proyectos de investigación relacionados con la robótica, el control lógico implementado por autómatas programables, el intercambio de datos entre dispositivos y la *Nube*, la representación de la información en distintos soportes gráficos, así como la interacción hombre-máquina, además, esta herramienta integra un conjunto de elementos que le permiten a estudiantes e interesados en el campo de la *Industria 4.0.* poder entrenar programas propios en vistas a su futuro desarrollo profesional.

Queda excluido del proyecto el desarrollo de la lógica de control superior de la celda que se puede implementar siguiendo la guía GEMMA o cualquier método similar. El inicio del ciclo de trabajo se realiza desde *RoboDK*, pudiéndose llevar a cabo desde cualquiera de los elementos de interacción desarrollados en la plataforma (*WebServer* o *HMI*), este aspecto no se amplía en la investigación.

Además, no es objetivo del *TFG* duplicar todas las pantallas diseñadas en el *WebServer* nuevamente en el *HMI*, es por ello por lo que solo se representa en este último la pantalla principal de la celda así como la relativa a los datos de los ejes del robot *UR1* o *UR_IZQ*. De igual forma, en el *Dashboard* de *UBIDOTS* se muestra al interesado como incorporar un conjunto de señales tipo ***bool*** y ***real***, del mismo modo, se hizo uso de la comunicación bidireccional entre el *PLC* y la *Nube*, permitiendo que el usuario contase con todas las posibles combinaciones para futuras investigaciones o entrenamientos personales.

2. CONDICIONES Y NORMAS DE CARÁCTER GENERAL

Este pliego de condiciones, con todos sus articulados, durante el montaje de la plataforma y hasta el uso de esta, manteniéndose a las partes que hace referencia, deberán cumplirse y se aceptarán. En caso de discrepancia hay un orden de prioridad de los documentos básicos del proyecto a revisar:

- Pliego de Condiciones.
- Presupuesto.
- Memoria.

- UNE-EN 61131-3:2013 (Ratificada): Autómatas programables. Parte 3: Lenguajes de programación.
- UNE-EN 60848:2013: Lenguaje de especificación *GRAFCET* para diagramas funcionales secuenciales.
- UNE-EN 60870-5-101:2003: Norma para la monitorización de sistemas de energía, control y las comunicaciones asociadas a los mismos.
- UNE-EN 61508:2011: Seguridad funcional de los sistemas eléctricos/electrónicos/electrónicos programables relacionados con la seguridad.
- UNE-EN ISO 13850:2016: Redes de comunicación industrial. Redes de automatización de alta disponibilidad.
- UNE-EN IEC 61158-4-24:2019 (Ratificada): Redes de comunicaciones industriales. Especificación de *Fieldbus*. Parte 4-24: Especificación del protocolo de la capa de enlace de datos.
- UNE-EN IEC 62769-115-2:2020: Integración de dispositivos de campo (*FDI*). Parte 115-2: Perfiles. *Modbus-RTU*.
- ISO 15745-4:2003/Amd 2:2007: Profiles for *Modbus TCP*.
- BS EN IEC 62453-71. Field device tool (*FDT*) interface specification. Part 71. *OPC UA* Information Model for *FDT*.
- UNE-EN ISO 10218-1: Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 1: Robots.
- UNE-EN ISO 10218-2: Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 2: Sistemas robots e integración.
- ISO/TS 15066:2016: Robots y dispositivos robóticos. Robots colaborativos.
- UNE 0060: *Industria 4.0*. Sistema de gestión para la digitalización. Requisitos.
- UNE 0061: *Industria 4.0*. Sistemas de gestión para la digitalización. Criterios para la evaluación de requisitos.
- ISO/IEC 30161:2020: Internet of Things (*IoT*) -- Requirements of *IoT* data exchange platform for various *IoT* services
- ISO/IEC TR 30166:2020: Internet of things (*IoT*) -- Industrial *IoT*.
- ISO/IEC 20933:2019: Information technology—Distributed application platforms and services (DAPS) -- Framework for distributed real-time access systems.

- ISO/IEC 19944-1:2020: Cloud computing and distributed platforms - Data flow, data categories and data use—Part 1: Fundamentals.
- ANSI/ISA-101.01-2015: Interfaces Humano-Máquina para Sistemas de Automatización de Procesos.
- ISO/TR 24464:2020: Automation systems and integration—Industrial data—Visualization elements of digital twins.
- UNE-EN ISO/ASTM 52900:2017: Fabricación aditiva. Principios generales. Terminología.
- UNE-EN ISO/ASTM 52921:2017: Terminología normalizada para la fabricación aditiva. Sistemas de coordenadas y métodos de ensayo.

3. CONDICIONES PARTICULARES

3.1. TECNICAS

3.1.1. OBJETO

En este apartado se describirán todas las características del hardware empleado en el desarrollo de la plataforma, así como las versiones de todos los programas incluidos, además, se especifican cada uno de los parámetros que constituyen la configuración máquinas virtuales, licencias, establecimiento de comunicaciones y se establecen los controles de calidad necesarios para desarrollar e interactuar con una plataforma *Digital Twin* de una celda automatizada con robótica colaborativa al nivel actual que marcan los estándares relativos al entorno *Industria 4.0.*

3.1.2. MATERIALES

En el presente capítulo se describe el hardware físico empleado en el diseño de la plataforma, para ello se presenta el modelo de los dispositivos así como sus características técnicas.

3.1.2.1. ORDENADOR

Debido a la alta demanda de recursos de representación gráfica del proyecto, es necesario contar con un ordenador que pueda emplear los recursos mínimos necesarios para asegurar un correcto funcionamiento, para ello, el autor se sirve de una *Conexión a Escritorio Remoto* contra un *PC* ***DELL Precision Tower 5820*** ubicado en el *Departamento de Ingeniería y Automática de la UPV* con las siguientes características (Tabla 1):

Tabla 1. Características del ordenador DELL Precision Tower 5820 del Departamento de Ingeniería Automática de la UPV.

Procesador	Intel(R) Core(TM) i9-10900X
Velocidad	@ 3.70GHz 3.70 GHz
Memoria Caché	19.25 Mb
Sistema Operativo	Windows 10 Pro VERSION 21H1
Tipo Pantalla	TV Samsung UHD Led
Tamaño Pantalla	48”
Resolución	3840 x 2160
Memoria RAM	32,0 GB
Disco Duro	500 Gb
Tipo Tarjeta	NVIDIA
Procesador Gráfico	Quadro RTX 4000
Tarjeta Red	Intel(R) Ethernet Connection (5) I219-LM

3.1.2.2. PLC VIRTUAL

El *PLC* empleado para el control y centralización de las comunicaciones esta basado en un *softPLC* implementado sobre el *PC* descrito, convirtiendo dicho ordenador en un controlador industrial compatible con el estándar IEC 61131-3. La emulación del *PLC* en el ordenador se realiza a través del servicio ***CODESYS Control Win SysTray -x64***.

3.1.3. SOFTWARE

3.1.3.1. CODESYS

Se emplea el software ***CODESYS*** en su versión ***V 3.5 SP16 Patch 4*** para llevar a cabo el control, análisis, monitorización y representación de los datos procesados en la plataforma.

3.1.3.1.1. REQUISITOS

Los requisitos de sistema para su instalación y funcionamiento son:

- Windows 8 / 10 (32/64 bits).
- Procesador 2.5 Gb.
- RAM 8 Gb.
- Disco Duro 12 Gb.

3.1.3.2. URSim

URSim es un software de simulación que se utiliza para la programación y simulación sin conexión de programas de robots de la marca ***Universal Robots***. *URsim* está hecho para el sistema operativo *Linux*. El proyecto ejecuta la versión ***URSim 5.10.0*** descargable desde:

UR Download | Support Site | Universal Robots (universal-robots.com)

3.1.3.2.1. REQUISITOS

Los requisitos de sistema para su instalación y funcionamiento son:

- VMWare Player. En su defecto:
- VirtualBox.

Para ejecutar el simulador en otro sistema operativo distinto de *Linux*, se necesita una máquina virtual:

3.1.3.2.2. MAQUINA VIRTUAL

Se emplea ***VMware Workstation 16 Pro*** para montar los discos duros que contienen el *IDE* para la programación de los robots UR, con una licencia propiedad de la *Universidad Politécnica de Valencia* con una fecha de vigencia del 30/04/2022. Se crean dos máquinas virtuales idénticas con el disco duro descargado para modelar al robot de la izquierda que inicia el ciclo y entrega la pieza, o ***UR1*** y ***UR2*** que representa el robot de la derecha, el cual, tras el proceso de medición retira la pieza y la coloca en la posición final de la secuencia. La configuración de la máquina virtual se presenta en la (Tabla 2):

Tabla 2. Parámetros de configuración de las máquinas virtuales de URSim.

Memory	8 GB
Processors	16 (4 x 4)
Hard Disk	10 GB
CD/DVD (SATA)	*** Cargar el disco virtual de URSim ***
Network Adapter	Host – only
Virtual Machine Name	UR1 o UR2
Guest Operating System	Linux
Versión	Ubuntu 64-bits
Share Folders	Enable (Always Enable) *Se comparte el escritorio *PC HOST* *

3.1.3.2.3. REQUISITOS

Los requisitos de sistema para su instalación y funcionamiento son:

- Windows 10/ 8.
- Procesador compatible 64-bit x86/AMD64 posterior al año 2011.
- Procesador 1.3 Gb.
- RAM 2 Gb mínimo, 4 Gb recomendado.

3.1.3.2.4. UR3e

Dentro del entorno *Linux Ubuntu,* existe ***PolyScope*** o la interfaz de usuario de robot (*GUI*), constituye la simulación de la pantalla táctil en su panel consola portátil. Existen en el escritorio las 4 versiones correspondiente a cada modelo de robot de la marca. La plataforma ***DICAR*** está diseñada para el robot ***UR3e*** por lo que el *IDE* empleado es ***URSim UR3e***.

3.1.3.3. RoboDK

RoboDK es un software para la simulación y programación fuera de línea. Se emplea para visualizar la secuencia de trabajo, permite programar todos los modelos de robot de *Universal Robot*, establecer conexión física a través de protocolos de intercambio en tiempo real de datos con la máquina de *URSim* y además, permite obtener los programas en el formato del entorno *UR.* La versión del empleada es ***RoboDK v5.2.1*** y el post-procesador ***Universal Robot URP***.

3.1.3.3.1. REQUISITOS

Los requisitos de sistema para su instalación y funcionamiento son:

- Windows Vista / 7 / 8 / 10 (32/64 bits).
- Procesador i3-2600 CPU @ 2.20 Gb.
- Memoria mínima 2 Gb, recomendada 4 Gb o más.

- Configuración gráfica mínima 1024x768 pixel de resolución, el controlador de gráficos debe ser compatible con OpenGL 3.0 o posterior
- Espacio en el disco duro: 40 GB, 1 GB libre.
- Ratón: de 2 botones (como mínimo), se recomiendan 3 botones o 2 botones con la rueda central del ratón.
- Internet: Conexión a Internet para activar las licencias de Red (licencias por defecto). Los cortafuegos no deben bloquear los puertos 80 y 443 (sólo licencias de red).

3.1.3.4. NB-DESIGNER

El software *NB-Designer* ofrece todas las funcionalidades y funciones para crear pantallas intuitivas para el operador, pertenece a la marca *Omron* y es gratuito. El modo *Simulación Online Directa* permite operar con un panel virtual por un período de 15 minutos, tras los cuales se debe lanzar nuevamente el simulador. Para el diseño del *HMI* se escoge una pantalla ***NB10W-TW01B*** de 10¨. La versión de ***NB-Designer*** utilizada es ***v1.48***.

3.1.3.4.1. REQUISITOS

Los requisitos de sistema para su instalación y funcionamiento son:

- Windows XP Service Pack 1 o superior, VISTA / 7 / 8 /10

3.1.3.5. VIRTUAL SERIAL PORT KIT

Es una herramienta de software que permite emular puertos serie. Estos puertos **virtuales *RS232*** creados se pueden conectar con un cable de módem nulo virtual. Constituye el driver para la implementación de la comunicación serie entre el *PLC* virtual y la pantalla simulada de *Omron*. La versión empleada es ***Virtual Serial Port Kit v5.8***.

3.1.3.5.1. REQUISITOS

Los requisitos de sistema para su instalación y funcionamiento son:

- Windows VISTA / 7 / 8 / 10.
- CPU compatible con doble núcleo x86 o x64.
- RAM 2 Gb.
- 16 Mb de espacio disponible en HDD / SSD.

3.1.3.5. NODE-RED

Node-Red es una plataforma de desarrollo que corre en *Node.JS*, que es a su vez un motor de ejecución asíncrono de *Javascript* para servidores. Permite implementar aplicaciones de *IoT* si hacer uso de código de programación, sustituyéndolo por un lenguaje gráfico basado en nodos. Se utiliza la versión ***v1.3.4***. La versión ***npm*** empleada para instalar *Node-Red* es ***v6.14.11***.

3.1.3.6. UBIDOTS

Es una plataforma de desarrollo de aplicaciones *IoT* que automatiza el proceso de creación de aplicaciones basadas en internet para que las empresas y los particulares puedan desplegar cualquier solución *IoT* a escala, y hacerlo rápidamente. Para elaborar el *Dashboard* de la plataforma se crea una cuenta ***FOR BUSINESS*** con un período de prueba de un mes. Existe además, la posibilidad de descargar una *App* para el uso de este recurso en el teléfono.

3.1.3.6.1. REQUISITOS

Los requisitos para instalar *UBIDOTS APP*:

- Teléfono OS Android.
- Teléfono OS Windows Mobile.

3.1.3.7. WebServer

La variante de visualización basada en la web de *CODESYS Visualization* permite el acceso remoto, la supervisión, así como el servicio y el diagnóstico de una planta a través de Internet. Un navegador web se comunica a través de *JavaScript* (o encriptación *SSL*) con el servidor web del controlador y muestra la visualización mediante *HTML5*. Esta tecnología es compatible con casi todos los navegadores y, por tanto, está disponible en los dispositivos móviles *iOS* y *Android*.

3.1.3.8. CODESYS AUTOMATION SERVER

La interface web permite la gestión y monitorización de todas las estaciones, ya sea en modo local o remoto, así como ejecutar las actualizaciones de los programas. La conexión puede ser directa desde el controlador al servidor y, en los casos en los que la seguridad requerida sea mayor, se puede utilizar un *Gateway* de *CODESYS* con seguridad *TLS* integrada.

3.1.3.8.1. REQUISITOS

Los requisitos de sistema para su instalación y funcionamiento son:

- Descargar desde *CODESYS Store* el paquete *CODESYS Automation SERVER.*
- *CODESYS Automation Server Conector.*
- *CODESYS Edge Gateway for WINDOWS.*

4. PRUEBAS

Una vez puestas en funcionamiento todas las herramientas descritas en el presente documento, con las indicaciones incluidas en el *Anexo I: Manual de Usuario* relativo a la *Memoria Técnica Descriptiva*, se debe llevar a cabo un ciclo completo de trabajo para comprobar que la comunicación entre los elementos fundamentales del proyecto se desarrolla de forma correcta.

Se debe prestar especial atención al movimiento de los dos robot desde la ventana gráfica de los *IDE* de *URSim UR3e*, así como la representación de todos los datos asociados a ellos desde el *WebServer,* el *HMI* o el servicio de *CODESYS AUTOMATION SERVER* (web visualization).

5. ENTREGA

Todos los programas descritos en la memoria, relativos a cada elemento de la plataforma, serán entregados en el *RiuNet* que es el *Repositorio Institucional de la Universitat Politècnica de València*, cuyo objetivo es ofrecer acceso en Internet a la producción científica, académica y corporativa de la comunidad universitaria con la finalidad de aumentar su visibilidad y hacerla accesible y preservable.

De esta forma, el usuario tiene a su disposición todo el código accesible listo para ser utilizado. Se quiere recalcar que en los *Anexos* de la memoria se incluye una copia seleccionable de todos los programas, preparados para que se puedan modificar a las necesidades del interesado.

No es posible dejar entre los archivos del proyecto, una copia de las máquinas virtuales de URSim debido al peso de estas, es por ello por lo que el autor facilita el enlace para la descarga de los discos duros desde la página de *UR* así como la configuración en VMware Workstation PRO, logrando reproducir el usuario las condiciones de trabajo descritas por el autor.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

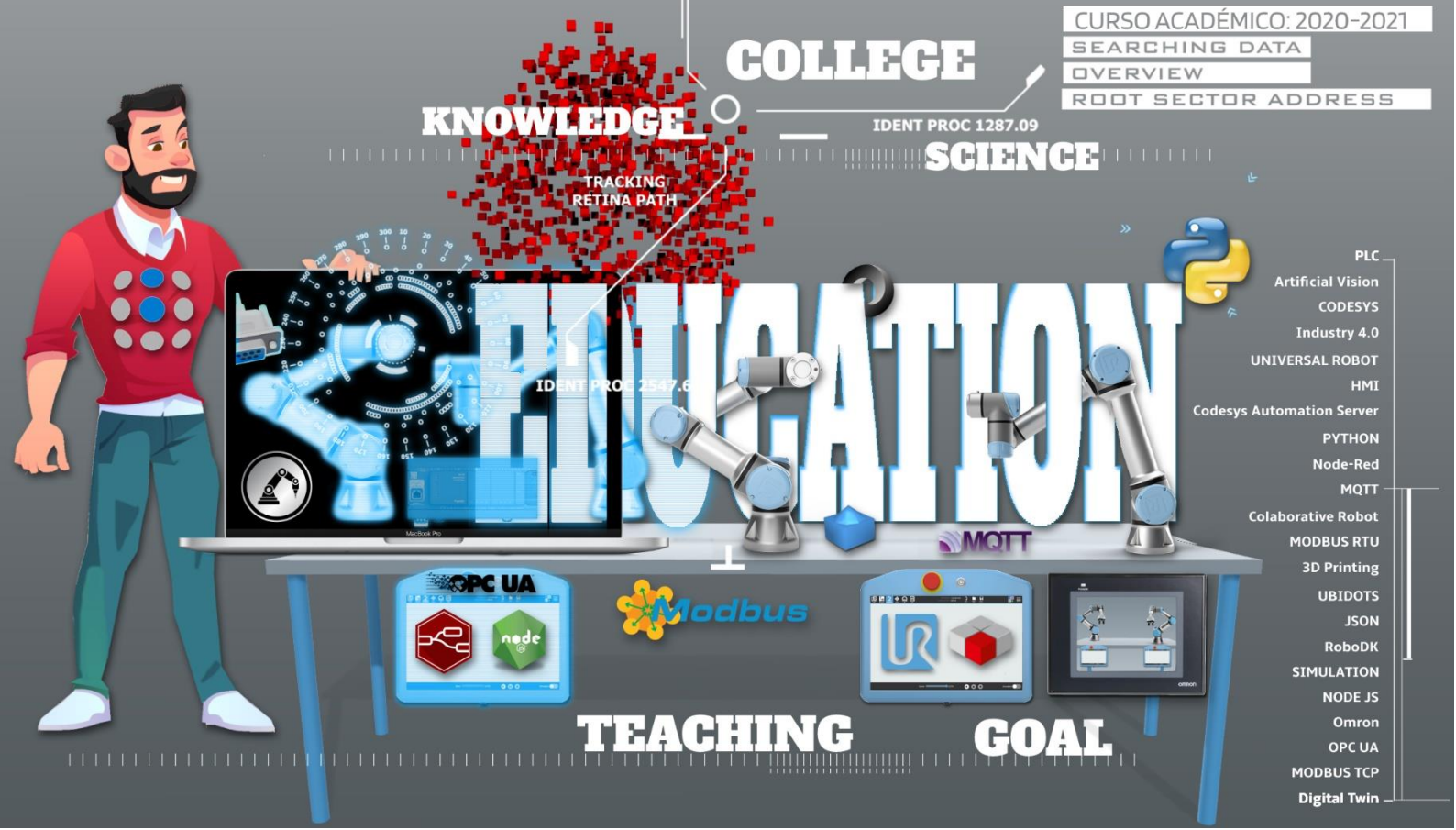
Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA CELDA AUTOMATIZADA CON ROBÓTICA COLABORATIVA

PRESUPUESTO

Dayron Rodríguez Díaz



Contenido

1. OBJETO

El presente documento expone el presupuesto realizado para la obtención del *Digital Twin* una Celda Automatizada con Robótica Colaborativa (***DICAR***), el proyecto constituye una herramienta para la futura creación de planes de investigación relacionados con la robótica, el control lógico implementado por autómatas programables, el intercambio de datos entre dispositivos y la *Nube*, la representación de la información en distintos soportes gráficos, así como la interacción hombre-máquina, además, esta herramienta integra un conjunto de elementos que le permiten a estudiantes e interesados en el campo de la *Industria 4.0.* poder entrenar programas propios en vistas a su futuro desarrollo profesional.

El presente informe se estructura en ocho grandes listados que conforman el presupuesto del proyecto concluyendo con un resumen general de las partidas que incluye los honorarios de redacción del documento, un porcentaje de gastos generales y otro de beneficio industrial.

2. LISTADOS QUE CONFORMAN EL PRESUPUESTO

CUADRO DE MANO DE OBRA

El proyecto se basa en crear una plataforma virtual que simule el comportamiento de una celda automatizada real, lo que hace inmaterial los objetivos de este, es decir, los resultados de la investigación no dan lugar a la creación de un producto tangible. Es por ello por lo que el autor cuantifica el monto de las horas invertidas en cada tarea del *TFG* como si fuesen partidas relativas a la mano de obra llevada a cabo por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática).

Cuadro de mano de obra

Num.	Código	Denominación de la mano de obra	Precio	Horas	Total
1	MOOA_BD	Proceso de búsqueda de distintos trabajos anteriores que permitiesen apoyar los objetivos iniciales planteados en el TFG. Documentación de la Normativa relativa al ámbito de la investigación. Visualización de conferenciantes representativos del sector de la Industria 4.0. Lectura de folletos del Ministerio de Industria Nacional, entre otra documentación, todas plasmadas en el capítulo Bibliografía de la Memoria Técnica Descriptiva de este proyecto. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	93,000 h	2.242,23
2	MOOA_BM	Esta partida justifica como darle forma al concepto de diseño, para ello se definen funcionalidades, estructura y objetivos para elaborar un diseño que cumpla con las expectativas esperadas y que se encuentre al nivel de la plataforma desarrollada. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	30,000 h	723,30
3	MOOA_CA	Proceso de configuración de las alarmas, eventos e infos implementados en el PLC para el correcto diagnóstico de los procesos controlados en la celda, en ellas se incluyen la vigilacia de la posición de todos los ejes de UR1 y UR2, la temperatura de cada uno de ellos, paradas de emergencias externas, estado de funcionamiento de los robot, comunicaciones con los periféricos, etc. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	21,000 h	506,31
4	MOOA_CAS_PLC	Proceso de vinculación de la lógica de control desarrollada en CODESYS a través de un servicio instalado en el PC denominado Gateway Control Win SysTray (64x). Comunicación entre el CODESYS server y el IDE. Envío de distintas versiones del software al repositorio de la nube de CODESYS así como el acceso al WebServer desde la plataforma. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	18,000 h	433,98

Cuadro de mano de obra

Num.	Código	Denominación de la mano de obra	Precio	Horas	Total
5	MOOA_CB	Para crear el esquema general de diseño de la interfaz se utilizó el software Concepts, Concepts es una aplicación construida para el diseño, es una versión avanzada de bocetos de papel, donde las herramientas naturales cumplen con la manipulación de vectores para que las ideas puedan cambiar y crecer a medida que lo hace el croquis. El primer esquema relativo a la pantalla principal de la interfaz o landing page, donde se muestran los robots en la mesa para indicar que se corresponde con dicha pantalla, además, se disponen algunos elementos para navegar, como son el menú inferior, el logo del proyecto en la esquina superior izquierda (constituye el botón HOME), una ventana para la gestión de permisos de usuarios y logging, un cuadro de mensajes de errores de la aplicación así como la disposición general de la ventana que constituirá la unidad básica del proyecto. La segunda pantalla representa el mayor conjunto de datos de la aplicación, en ella existen tres paneles en el centro de la vista, correspondientes a una barra de navegación que permite el salto entre ventanas relativas a un mismo robot, mencionando por ejemplo las pantallas de Programa, I/O, Ejes y TCP, un cuadro con elementos descriptivos de los parámetros mostrados (leyenda) y un panel de representación de datos o Dashboard. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	30,000 h	723,30
6	MOOA_CC_HMI	Desarrollo de la configuración de la pantalla NB10W-TW01B de 10 pulgadas de Omron, configuración de cada elemento de representación y envío de información. Protocolo de envío/recepción segmentado de los datos a través del canal serie Modbus_RTU entre el PLC y el HMI. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	15,000 h	361,65
7	MOOA_CMC	Estudio de los principios de funcionamiento e implementación de los protocolos empleados para interconectar los distintos elementos físicos entre sí. A través de una interfaz Ethernet se despliega la comunicación Modbus_TCP con los servidores y los clientes alojados en cada máquina virtual de UR1 y UR2 respectivamente y un módulo Modbus_RTU sujeto un terminal de comunicación serie para el enlace entre el PLC y el HMI de Omron. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	27,000 h	650,97
8	MOOA_CMV	Búsqueda y descarga de la máquina virtual de URSIM versión v5.10.0 desde la página web oficial [https://www.universal-robots.com/download/?query=&type[]=98769]. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	39,000 h	940,29

Cuadro de mano de obra

Num.	Código	Denominación de la mano de obra	Precio	Horas	Total
9	MOOA_CPG	Concluida y depurada la secuencia de trabajo en RoboDK se exportan los programas de cada robot UR en formato .urp para ser cargados en el IDE URSim UR3e de cada máquina virtual. Estos programas deben ser revisados para eliminar llamadas a rutinas de animación incluidas en el código y que no aportan nada en Polyscope, además, se asigna la activación/desactivación de las correspondientes I/O configuradas en la tarjeta Modbus_TCP Client de cada UR. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	30,000 h	723,30
10	MOOA_C_CAS	Acceso a la plataforma y configuración de las credenciales para la conexión con el IDE de CODESYS. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	15,000 h	361,65
11	MOOA_C_Dev	Creación de un dispositivo libre de marca, en el cual se definen las variables que se intercambiaran con el PLC a través de Node-Red. El autor define 15 señales para mostrar el proceso de creación de un Dashboard, entre ellas 6 señales digitales de entrada para mostrar el estado de los 2 robots UR, 6 señales tipo real para representar la posición de los ejes de UR1 y tres señales digitales de salida para configurar la posición de entrega de la pieza por parte de UR1. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	15,000 h	361,65
12	MOOA_C_N-HMI	Desarrollo de la lógica para el intercambio de datos entre el PLC y la Nube UBIDOTS con la cual se configura la posición de entrega de la pieza por parte de UR1. Implementación de la comunicación Modbus_RTU para el intercambio de señales y datos con el HMI de Omron que simula una pantalla NB10W-TW01B de 10 pulgadas. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	177,000 h	4.267,47
13	MOOA_C_PLC_U	Implamentación de las comunicaciones entre los robots modelados en URSim y los mecanismos que simulan su geometría incluidos en RoboDK, se emplean protocolos RTDE para intercomunicar RoboDK-URSim y Modbus-TCP para la interconexión entre CODESYS-URSim. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	234,000 h	5.641,74
14	MOOA_C_R_C_U	Proceso de desarrollo e implemtnación de las comunicaciones entre loe tres softwares para llevar a cabo la interconexión de todas las seañes necesarias para desarrollar la plataforma. Se emplean protocolos RTDE y Modbus_TCP para desarrollar esta partida. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	177,000 h	4.267,47

Cuadro de mano de obra

Num.	Código	Denominación de la mano de obra	Precio	Horas	Total
15	MOOA_DSH	Proceso de diseño y configuración de una interfaz de usuario que represente el estado de las variables creadas en el device a través de distintos Widgetc o componentes como por ejemplo, Indicators, Switch y Bar Graph. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	48,000 h	1.157,28
16	MOOA_EV	Programación y generación de eventos relacionados por las variables medidas en el device. Se implementa una lógica de envío de emails a una dirección de correo electrónico para indicar cuando alguno de los UR no se encuentra conectado o algún eje de UR1 está posicionado fuera del rango establecido. Es posible realizar envíos de SMS a número de teleféfonos etc. pero es necesario tener una cuenta premiun para acceder a estas características. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	9,000 h	216,99
17	MOOA_GP	Proceso de obtención de los programas para ser cargados en URSim, utilizando los post-procesadores Universal Robot URP. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	15,000 h	361,65
18	MOOA_I	Proceso de selección de la paleta de colores empleada ene l diseño de la plataforma, constituyendo el elemento central en el proceso de creación de las pantallas implementadas en el WebServer y el HMI. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	15,000 h	361,65
19	MOOA_IO_TCP	Proceso de búsqueda de los elementos virtuales importados en RoboDK para modelas los sistemas físicos planteados en los objetivos del proyecto. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	21,000 h	506,31
20	MOOA_IT	Proceso de instalación de los distintos programas seleccionados para el desarrollo de la plataforma. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	30,000 h	723,30
21	MOOA_JQ	Envío de datos con los formatos actuales de encriptación, empleo de lenguajes comunes de progrmación para el intercambio de datos y señales en proyectos de IoT industriales. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	45,000 h	1.084,95
22	MOOA_NR	Instalación de Node.js, instalación de Node_Red, y comprensión del lenguaje de programación en nodos. Búsqueda de códigos ejemplo en Github para soportar el objetivo del TFG de comunicar la plataforma creada con la Nube y establecer un intercambio de información.	24,110	27,000 h	650,97

Cuadro de mano de obra

Num.	Código	Denominación de la mano de obra	Precio	Horas	Total
23	MOOA_P	Desarrollo y planificación de todo el código de control del PLC. Se incluyen traeas de depuración de errores, cálculo de la posición de los ejes de cada robot, diagnóstico de los buses de comunicación en tiempo real, diseño de los algoritmos de control de los elementos visuales del WebServer, etc.Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	240,000 h	5.786,40
24	MOOA_PP	Proceso de inserción de todas las pantallas secundarias creadas previamente, se definen el diagrama de flujo de navegación del WebServer quedando constituido por 5 ventanas de carácter general y 8 relativas a los datos de cada UR: - Pantallas Generales (Index, LOG, Vision, Diag Comunicaciones y Configuración) - Pantallas de Datos UR (Ejes, TCP, Programa e I/O) para UR1 y UR2 (Total: 8) Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	150,000 h	3.616,50
25	MOOA_PRB	Pruebas de comprobación desarrolladas para concluir apartados generales del TFG. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	162,000 h	3.905,82
26	MOOA_PS	Conceptaulización de las pantallas secundarias que permiten la fragmentación del gtrabajo de diseño y programación de los elementos que se representan en las pantallas principales. El principio de funcionamiento se basa en crear módulos de datos pequeños que sean instanciados posteriormente, pudiendo se modificados de forma más rápida y sencilla en caso de necesidad. En total se crean 64 módulos secundarios en el proyecto. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	177,000 h	4.267,47
27	MOOA_PY	Creación de los programas para solventar la visión artificial así como la comunicación empleando protocolo OPC UA para el envío de los datos de las mediciones desarrolladas empleando las APIs de RoboDK, simulando el comportamiento de una medición llevada a cabo con una cámara Matrox Iris GTR. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	87,000 h	2.097,57
28	MOOA_P_HMI1	No es objetivo del proyecto replicar todas y cada una de las pantallas del WebServer, a modo de ejemplo solo se diseñan aquellas homólogas de INDEX y UR1_EJES. En ellas se utilizan todos los componentes esenciales para desarrollar una futura interfaz empleando una pantalla HMI plenamente funcional y con los estándares marcados en la actual Industria 4.0. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	54,000 h	1.301,94

Cuadro de mano de obra

Num.	Código	Denominación de la mano de obra	Precio	Horas	Total
29	MOOA_P_NODO	Programa para el intercambio de datos entre el PLC de CODESYS y la Nube implementada en UBIDOTS, para llevar a cabo este apartado, las comunicaciones se apoyan en el protocolo de comunicación industrial OPC UA que soporta el envío de datos sobre TCP/IP, la encriptación y protección de la información entre otras tareas, se desarrollaron en esta partida. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	171,000 h	4.122,81
30	MOOA_SH	Desarrollo de los criterios de selección de las distintas herramientas que dan solución a la creación de un Digital Twin para una celda automatizada con robótica colaborativa. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	69,000 h	1.663,59
31	MOOA_ST	Desarrollo de toda la lógica necesaria para elaborar un ciclo donde UR1 inicie una rutina de pick sobre una pieza que se encuentra en la base de este, posteriormente recibe desde el PLC los datos de configuración de la cámara, así como la posición de entre de la pieza, una vez en posición de dejada se inicia el ciclo de medición de la posición para actualizar el punto de recogida por parte de UR2, el cual una vez tiene en el gripper la pieza, realiza su entrega en la mesa, concluyendo el ciclo e indicando a UR1 que puede comenzar nuevamente la secuencia. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	45,000 h	1.084,95
32	MOOA_T	Proceso de diseño del tipo de letra seleccionado para usar en el proyecto, creación de todo el cunjunto de etiquetas mostradas tanto en el WebServer como en el HMI. Tarea desarrollada por un Ingeniero Técnico (Grado en Ingeniería Industrial Automática)	24,110	9,000 h	216,99
				Total mano de obra:	55.332,45

CUADRO DE MAQUINARIA

Cuadro de maquinaria

Num.	Código	Denominación de la maquinaria	Precio	Cantidad	Total
1	SF_VM	Es un hipervisor alojado que se ejecuta en versiones x64 de los sistemas operativos Windows y Linux. Permite a los usuarios configurar máquinas virtuales (VM) en una sola máquina física y utilizarlas simultáneamente junto con la máquina host. Cada máquina virtual puede ejecutar su propio sistema operativo, incluidas las versiones de Microsoft Windows, Linux, BSD y MS-DOS. VMware Workstation admite la conexión de adaptadores de red de host existentes y el uso compartido de unidades de disco físico y dispositivos USB con una máquina virtual. Puede simular unidades de disco; un archivo de imagen ISO se puede montar como una unidad de disco óptico virtual y las unidades de disco duro virtual se implementan como archivos .vmdk.	163,990	1,000 u	163,99
2	SF_R_DK	RoboDK es un software desarrollado por una empresa de Canadá y su principal objetivo es aportar potentes capacidades de simulación y programación de robótica a todas las empresas. En la actualidad, es compatible con más de 200 robots de 50 fabricantes, como ABB, Fanuc, KUKA, Yaskawa, Stäubli y Universal Robots. El simulador es una herramienta de programación universal fuera de línea, que facilita la programación de todo tipo de robots, así como la generación de programas de robot específicos de la marca. Se utiliza habitualmente en aplicaciones industriales como el fresado con robot, la soldadura con robot, impresión 3D de pintura y calibración de robots.	145,000	1,000 u	145,00
3	SF_UBI	Es una plataforma de desarrollo de aplicaciones IoT que automatiza el proceso de creación de aplicaciones basadas en internet para que las empresas y los particulares puedan desplegar cualquier solución IoT a escala, y hacerlo rápidamente. La plataforma es un constructor de aplicaciones IoT de enlace de nodos centrado en el usuario, con herramientas de análisis de datos y funciones en la Nube, visualizaciones de cuadros de mando, herramientas de gestión de dispositivos, eventos de BI y motor de alarmas, y autenticación/acceso del usuario final para dar a los usuarios y operadores los datos que necesitan y nada más. Con la plataforma UBIDOTS, los usuarios recopilan, mejoran y entregan datos de sensores, actuadores y balizas que son importantes para las empresas y los usuarios para tomar decisiones basadas en datos que mejoran la eficiencia y la eficacia. Para elaborar el Dashboard de la plataforma se crea una cuenta FOR BUSINESS.	40,160	1,000 u	40,16

Cuadro de maquinaria

Num.	Código	Denominación de la maquinaria	Precio	Cantidad	Total
4	SF_NB	El software NB-Designer ofrece todas las funcionalidades y funciones para crear pantallas intuitivas para el operador, pertenece a la marca Omron y es gratuito. El modo Simulación Online Directa permite operar con un panel virtual por un período de 15 minutos, tras los cuales se debe lanzar nuevamente el simulador. Para el diseño del HMI se escoge una pantalla NB10W-TW01B de 10¨. La versión de NB-Designer utilizada es v1.48.	1,000	1,000 u	1,00
5	SF_NR	Node-Red es una plataforma de desarrollo que corre en Node.JS, que es a su vez un motor de ejecución asíncrono de Javascript para servidores. Permite implementar aplicaciones de IoT si hacer uso de código de programación, sustituyéndolo por un lenguaje gráfico basado en nodos. Se utiliza la versión v1.3.4.	1,000	1,000 u	1,00
6	SF_UR	Software de simulación que se utiliza para la programación offline y la simulación de programas de robots de la marca Universal Robot, y está hecho para el sistema operativo Linux.	1,000	1,000 u	1,00
7	SF_COD	El software CODESYS de la empresa alemana 3S-Smart Software Solutions GMBH es un entorno de desarrollo para la programación de controladores conforme con el estándar industrial internacional IEC 61131-3, abarca 6 lenguajes de programación diferentes. Codesys tiene dos servicios que se ejecutan en Windows con las siguientes denominaciones, Codesys Gateway SysTray y Codesys Control Win SysTray estos programas son necesarios para simular un controlador y de esta forma llevar a cabo al virtualización del componente. Software para la programación y la simulación de un PLC virtual de la empresa alemana 3S-Smart Software Solution GMBH.	1,000	1,000 u	1,00
				Total maquinaria:	353,15

CUADRO DE MATERIALES

Cuadro de materiales

Num.	Código	Denominación del material	Precio	Cantidad	Total
1	MAT_DELL	Procesador Intel Xeon W-2275 (14 núcleos, 3,3 GHz, 4,8 GHz con Turbo, HT, 19,25 MB, (165 W), DDR4-2933). Windows 10 Pro para estaciones de trabajo (hasta 4 núcleos) .Radeon Pro WX 2100, 2 GB, DP, 2 Mini DP (5820 T). 8GB 1x8GB DDR4 2933MHz RDIMM ECC Memory. Disco duro SATA de 500GB a 7200 rpm de 2,5". Precision 5820 Tower, 425 W, chasis, CL	2.522,290	0,199 u	501,94
2	MAT_PLC	El PLC empleado para el control y centralización de las comunicaciones está basado en un softPLC implementado sobre el PC descrito anteriomente, convirtiendo dicho ordenador en un controlador industrial compatible con el estándar IEC 61131-3. La emulación del PLC en el ordenador se realiza a través del servicio CODESYS Control Win SysTray -x64.	1,000	1,000 u	1,00
				Total materiales:	502,94

CUADRO ANEJO JUSTIFICACIÓN DE PRECIOS

Anejo de justificación de precios

Nº	Código	Ud	Descripción			Total
			1 DESARROLLO DE LA INVESTIGACION			
1.1	CAP_01_INV.1	**h**	**Proceso inicial de investigación para recavar ideas que encaminen los objetivos del TFG. Brainstorming sobre las distintas herramientas a incluir en el proyecto, selección del conjunto de softwares finales para dar solución a los objetivos, búsqueda de documentación sobre la cual apoyar el proceso de investigación así como la delimitación de los puntos centrales del TFG. Este proceso se inicia con fecha 01-02-2021 y concluye el 26-02-2021.**			
	MOOA_SH		69,000 h	Selección Herramientas	24,110	1.663,59
	MOOA_BD		93,000 h	Búsqueda Documentación	24,110	2.242,23
	MOOA_IT		30,000 h	Instalación Software	24,110	723,30
	%CDC		2,000 %	Costes Directos Complementarios	4.629,120	92,58
				Precio total por h		**4.721,70**
1.2	CAP_01_INV.2	**h**	**Proceso de investigación, programación, configuración y depuración de la lógica de control implementada en el PLC. Se realizan los siguientes trabajos es esta partida:** **- Configuración Módulos Comunicación** **- Programación Lógica** **- Configuración Alarmas** **- Comunicación Nube-HMI** **Este proceso se inicia el 05-02-2021 y concluye el día 09-04-2021**			
	MOOA_CMC		27,000 h	Configuración Módulos Comunicaciones	24,110	650,97
	MOOA_P		240,000 h	Programación Lógica	24,110	5.786,40
	MOOA_CA		21,000 h	Configuración Alarmas	24,110	506,31
	MOOA_C_N-HMI		177,000 h	Comunicación Nube_HMI	24,110	4.267,47
	MOOA_PRB		27,000 h	Pruebas	24,110	650,97
	%CDC		2,000 %	Costes Directos Complementarios	11.862,120	237,24
				Precio total por h		**12.099,36**
1.3	CAP_01_INV.3	**h**	**Proceso de investigación, aprendizaje y entrenamiento para poder manejar los robots UR3e empleados para intercambiar una pieza en la celda. Se desarrollan las siguientes tareas en este apartado:**			
	MOOA_CMV		39,000 h	Configuración Máquinas Virtuales	24,110	940,29
	MOOA_C_R_C_U		177,000 h	Conexión RoboDK-CODESYS-URSim	24,110	4.267,47
	MOOA_CPG		30,000 h	Configuración Programas Generados	24,110	723,30
	MOOA_PRB		27,000 h	Pruebas	24,110	650,97
	%CDC		2,000 %	Costes Directos Complementarios	6.582,030	131,64
				Precio total por h		**6.713,67**
1.4	CAP_01_INV.4	**h**	**Proceso de investigación, aprendizaje y manejo de las herramientas presentes en el programa. Se desarrollas las siguientes tareas en la presente partida:** **- Primeros pasos, importar objetos (TCP)** **- Comunicación PLC-URSim** **- Python (Visión y comunicación OPC UA)** **- Secuencia de trabajo** **- Generar programas para URSim** **Este proceso se inicia el 08-02-2021 y concluye el día 09-04-2021**			
	MOOA_IO_TCP		21,000 h	Primeros Pasos, Importación Objetos (TCP)	24,110	506,31
	MOOA_C_PLC_U		234,000 h	Comunicación PLC_URSim	24,110	5.641,74
	MOOA_PY		87,000 h	Programación Python	24,110	2.097,57
	MOOA_ST		45,000 h	Secuencia Trabajo	24,110	1.084,95
	MOOA_GP		15,000 h	Generación Programas URP	24,110	361,65
	MOOA_PRB		27,000 h	Pruebas	24,110	650,97
	%CDC		2,000 %	Costes Directos Complementarios	10.343,190	206,86
				Precio total por h		**10.550,05**

Anejo de justificación de precios

Nº	Código	Ud	Descripción			Total
1.5	CAP_01.INV.5	h	**Uno de los objetivos marcados en la confección del presente proyecto, es la presentación de un plan de diseño tanto de la experiencia de usuario como la interfaz de usuario que asegure una representación, interacción y relación de todos los datos y procesos incluidos en la plataforma con una presentación depurada, funcional y equiparable a las actuales tendencias de diseño de aplicaciones digitales empleando una pantalla HMI o el servicio alojado en el PLC WebServer. Las tareas desarrolladas en esta partida son:** **- Brainstorming Modboard** **- Conceptos y bocetos** **- Tipografía** **- Infografía**			
	MOOA_BM		30,000 h	Brainstorming Modboard	24,110	723,30
	MOOA_CB		30,000 h	Conceptos y Bocetos	24,110	723,30
	MOOA_T		9,000 h	Tipografía	24,110	216,99
	MOOA_I		15,000 h	Infografía	24,110	361,65
	%CDC		2,000 %	Costes Directos Complementarios	2.025,240	40,50
				Precio total por h		**2.065,74**
1.6	CAP_01.INV.6	h	**Proceso de desarrollo y programación de la unidad de interfaz gráfica fundamental del proyecto. Servidos web alojado dentro del softPLC que emula el control de la celda a través de un PLC virtual. Las tareas desarrolladas en esta partida son:** **- Pantallas secundarias** **- Pantallas principales**			
	MOOA_PS		177,000 h	Diseño Pantallas Secundarias	24,110	4.267,47
	MOOA_PP		150,000 h	Diseño Pantallas Principales	24,110	3.616,50
	%CDC		2,000 %	Costes Directos Complementarios	7.883,970	157,68
				Precio total por h		**8.041,65**
1.7	CAP_01_INV.7	h	**Proceso de desarrollo e implementación de la lógica de control y comunicación del HMI. Las tareas desarrolladas en esta partida incluyen:** **- Configuración comunicaciones** **- Pantallas**			
	MOOA_CC_HMI		15,000 h	Configuración Comunicaciones	24,110	361,65
	MOOA_P_HMI1		54,000 h	Diseño Pantallas	24,110	1.301,94
	MOOA_PRB		27,000 h	Pruebas	24,110	650,97
	%CDC		2,000 %	Costes Directos Complementarios	2.314,560	46,29
				Precio total por h		**2.360,85**
1.8	CAP_01.INV.8	h	**Desarrollo de aplicaciones IoT. Proceso de configuración de Node. js y Node-Red, entendimiento de protocolos MQQT y JSON, enlace del PLC con la Nube. Las tareas desarrolladas en esta partida son:** **- Instalación Node-Red** **- Programación Nodos** **- JSON y MQTT**			
	MOOA_NR		27,000 h	Node-Red	24,110	650,97
	MOOA_P_NODO		171,000 h	Programación	24,110	4.122,81
	MOOA_JQ		45,000 h	JSON - MQQT	24,110	1.084,95
	%CDC		2,000 %	Costes Directos Complementarios	5.858,730	117,17
				Precio total por h		**5.975,90**

Anejo de justificación de precios

Nº	Código	Ud	Descripción			Total
1.9	CAP_01.INV.9	h	**Proceso de creación de toda la infraestructura que da soporte al diseño necesario para el envío y la respresentación de datos de la celda en un servidor en la Nube. Se crea una cuenta FOR BUSINESS en la página web del proveedor de servicios de cloud computing [https://ubidots.com/] que tiene un período de prueba de 1 mes con la mayoría de las carcaterísticas disponibles. Las tareas de desarrolladas en esta partida son:** **- Configuración device** **- Dahsboard** **- Programación eventos**			
	MOOA_C_Dev		15,000 h	Configuración Device	24,110	361,65
	MOOA_DSH		48,000 h	Dashboard	24,110	1.157,28
	MOOA_EV		9,000 h	Eventos	24,110	216,99
	MOOA_PRB		27,000 h	Pruebas	24,110	650,97
	%CDC		2,000 %	Costes Directos Complementarios	2.386,890	47,74
				Precio total por h		**2.434,63**
1.10	CAP_01.INV.10	h	**CODESYS Automation Server es la plataforma Industry 4.0 para más de 100,000 usuarios de CODESYS en todo el mundo. El servidor les ayuda a administrar sus controladores y aplicaciones relacionadas y a implementar actualizaciones. Al mismo tiempo, la plataforma ofrece una infraestructura segura para acceso remoto, depuración o visualización web. Las tareas desarrolladas en esta partida son:** **- Configuración** **- Enlace PLC-Plataforma**			
	MOOA_C_CAS		15,000 h	Configuración	24,110	361,65
	MOOA_CAS_PLC		18,000 h	Enlace PLC_CODESYS AUTOMATION SERVER	24,110	433,98
	MOOA_PRB		27,000 h	Pruebas	24,110	650,97
	%CDC		2,000 %	Costes Directos Complementarios	1.446,600	28,93
				Precio total por h		**1.475,53**

Anejo de justificación de precios

Nº	Código	Ud	Descripción			Total
			2 HARDWARE			
2.1	CAP_02_Ordenador	**u**	**Ordenador ubicado en el departamento de Ingeniería de Sistemas y Automática de la UPV sobre el cual se diseño la plataforma DICAR presentada en la Memoria Técnica Descriptiva del TFG.**			
	MAT_DELL	0,199 u	DELL Precision Tower 5820	2.522,290		501,94
	MAT_PLC	1,000 u	PLC Virtual	1,000		1,00
	%CDC	2,000 %	Costes Directos Complementarios	502,940		10,06
			Precio total por u			**513,00**

Anejo de justificación de precios

Nº	Código	Ud	Descripción	Total
			3 SOFTWARE	
3.1	CAP_03_CODESYS	u	**La plataforma virtual permite crear un Digital Twin de una celda automatizada con dos robots colaborativos que intercambian una pieza empleando visión artificial. La plataforma es completamente gratuita y permite a futuros alumnos e interesados entrenarse o desarrollar proyectos de investigación rompiendo las barreras de acceso a estas tecnologías involucradas en la Industria 4.0. El Digital Twin habilita la migración de todos los programas implementados a los dispositivos físicos en cualquier instante tras la depuración de posibles colisiones y errores en el código. Para materializar los módulos, se utilizan simuladores de programación fuera de línea como RoboDK, para la representación gráfica del proceso y la secuencia de trabajo, URSim como modelo fisicomatemático de los robot UR3e, CODESYS permite emular un softPLC con la lógica de control, este componente además tiene embebido un WebServer para representar todos los datos y el control de la celda, el cuál presenta un concepto de diseño gráfico desarrollado desde los cimientos del proyecto por el autor, al mismo tiempo y con el mismo propósito, una pantalla HMI de Omron actúa como interfaz hombre-máquina. La visión artificial se programa empleando lenguaje Python, el cual aprovecha las APIs de RoboDK para simular una cámara 2D. Todos los elementos se interrelacionan entre sí en tiempo real y se comunican con dos servicios de cloud computing, comenzando por UBIDOTS para monitorizar parámetros de la celda y establecer la posición en la cual el primer robot entrega la pieza en el proceso de intercambio a través de variables que se manipulan por medio de protocolos como OPC UA, MQTT, JSON y nodos implementados en Node-Red, la segunda plataforma está desarrollada por CODESYS Automation Server, un servicio web donde el usuario puede consultar el estado del controlador, distintas versiones del código entre las cuales se puede elegir cual ejecutar, o programar directamente el autómata desde un navegador web rompiendo la barrera de la programación anclada a un PC convencional, pudiendo realizar dicha tarea en una Tablet o un Smartphone en cualquier punto geográfico.**	
	SF_COD	1,000 u	CODESYS IDE	1,000	1,00
	SF_UR	1,000 u	URSim	1,000	1,00
	SF_R_DK	1,000 u	RoboDK	145,000	145,00
	SF_VM	1,000 u	VMware Workstation PRO 16	163,990	163,99
	SF_NB	1,000 u	NB-Designer	1,000	1,00
	SF_NR	1,000 u	Node_Red	1,000	1,00
	SF_UBI	1,000 u	UBIDOTS	40,160	40,16
	%CDC	2,000 %	Costes Directos Complementarios	353,150	7,06
			Precio total por u		**360,21**

CUADRO DE PRECIOS UNITARIOS

Cuadro de precios nº 1

Nº	Designación	Importe	
		En cifra (Euros)	En letra (Euros)
1	h CODESYS Automation Server es la plataforma Industry 4.0 para más de 100,000 usuarios de CODESYS en todo el mundo. El servidor les ayuda a administrar sus controladores y aplicaciones relacionadas y a implementar actualizaciones. Al mismo tiempo, la plataforma ofrece una infraestructura segura para acceso remoto, depuración o visualización web. Las tareas desarrolladas en esta partida son: - Configuración - Enlace PLC-Plataforma	1.475,53	MIL CUATROCIENTOS SETENTA Y CINCO EUROS CON CINCUENTA Y TRES CÉNTIMOS
2	h Uno de los objetivos marcados en la confección del presente proyecto, es la presentación de un plan de diseño tanto de la experiencia de usuario como la interfaz de usuario que asegure una representación, interacción y relación de todos los datos y procesos incluidos en la plataforma con una presentación depurada, funcional y equiparable a las actuales tendencias de diseño de aplicaciones digitales empleando una pantalla HMI o el servicio alojado en el PLC WebServer. Las tareas desarrolladas en esta partida son: - Brainstorming Modboard - Conceptos y bocetos - Tipografía - Infografía	2.065,74	DOS MIL SESENTA Y CINCO EUROS CON SETENTA Y CUATRO CÉNTIMOS
3	h Proceso de desarrollo y programación de la unidad de interfaz gráfica fundamental del proyecto. Servidos web alojado dentro del softPLC que emula el control de la celda a través de un PLC virtual. Las tareas desarrolladas en esta partida son: - Pantallas secundarias - Pantallas principales	8.041,65	OCHO MIL CUARENTA Y UN EUROS CON SESENTA Y CINCO CÉNTIMOS
4	h Desarrollo de aplicaciones IoT. Proceso de configuración de Node. js y Node-Red, entendimiento de protocolos MQQT y JSON, enlace del PLC con la Nube. Las tareas desarrolladas en esta partida son: - Instalación Node-Red - Programación Nodos - JSON y MQTT	5.975,90	CINCO MIL NOVECIENTOS SETENTA Y CINCO EUROS CON NOVENTA CÉNTIMOS
5	h Proceso de creación de toda la infraestructura que da soporte al diseño necesario para el envío y la respresentación de datos de la celda en un servidor en la Nube. Se crea una cuenta FOR BUSINESS en la página web del proveedor de servicios de cloud computing [https://ubidots.com/] que tiene un período de prueba de 1 mes con la mayoría de las carcaterísticas disponibles. Las tareas de desarrolladas en esta partida son: - Configuración device - Dahsboard - Programación eventos	2.434,63	DOS MIL CUATROCIENTOS TREINTA Y CUATRO EUROS CON SESENTA Y TRES CÉNTIMOS

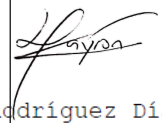
Cuadro de precios nº 1

Nº	Designación	Importe	
		En cifra (Euros)	En letra (Euros)
6	h Proceso inicial de investigación para recavar ideas que encaminen los objetivos del TFG. Brainstorming sobre las distintas herramientas a incluir en el proyecto, selección del conjunto de softwares finales para dar solución a los objetivos, búsqueda de documentación sobre la cual apoyar el proceso de investigación así como la delimitación de los puntos centrales del TFG. Este proceso se inicia con fecha 01-02-2021 y concluye el 26-02-2021.	4.721,70	CUATRO MIL SETECIENTOS VEINTIUN EUROS CON SETENTA CÉNTIMOS
7	h Proceso de investigación, programación, configuración y depuración de la lógica de control implementada en el PLC. Se realizan los siguientes trabajos es esta partida: - Configuración Módulos Comunicación - Programación Lógica - Configuración Alarmas - Comunicación Nube-HMI Este proceso se inicia el 05-02-2021 y concluye el día 09-04-2021	12.099,36	DOCE MIL NOVENTA Y NUEVE EUROS CON TREINTA Y SEIS CÉNTIMOS
8	h Proceso de investigación, aprendizaje y entrenamiento para poder manejar los robots UR3e empleados para intercambiar una pieza en la celda. Se desarrollan las siguientes tareas en este apartado: - Configuración máquinas virtuales - Conexión RoboDK-CODESYS-URSim - Configuración programas generados Este proceso se inicia el 04-02-2021 y concluye el día 09-04-2021	6.713,67	SEIS MIL SETECIENTOS TRECE EUROS CON SESENTA Y SIETE CÉNTIMOS
9	h Proceso de investigación, aprendizaje y manejo de las herramientas presentes en el programa. Se desarrollas las siguientes tareas en la presente partida: - Primeros pasos, importar objetos (TCP) - Comunicación PLC-URSim - Python (Visión y comunicación OPC UA) - Secuencia de trabajo - Generar programas para URSim Este proceso se inicia el 08-02-2021 y concluye el día 09-04-2021	10.550,05	DIEZ MIL QUINIENTOS CINCUENTA EUROS CON CINCO CÉNTIMOS
10	h Proceso de desarrollo e implementación de la lógica de control y comunicación del HMI. Las tareas desarrolladas en esta partida incluyen: - Configuración comunicaciones - Pantallas	2.360,85	DOS MIL TRESCIENTOS SESENTA EUROS CON OCHENTA Y CINCO CÉNTIMOS
11	u Ordenador ubicado en el departamento de Ingeniería de Sistemas y Automática de la UPV sobre el cual se diseño la plataforma DICAR presentada en la Memoria Técnica Descriptiva del TFG.	513,00	QUINIENTOS TRECE EUROS

Cuadro de precios nº 1

Nº	Designación	Importe	
		En cifra (Euros)	En letra (Euros)
12	u La plataforma virtual permite crear un Digital Twin de una celda automatizada con dos robots colaborativos que intercambian una pieza empleando visión artificial. La plataforma es completamente gratuita y permite a futuros alumnos e interesados entrenarse o desarrollar proyectos de investigación rompiendo las barreras de acceso a estas tecnologías involucradas en la Industria 4.0. El Digital Twin habilita la migración de todos los programas implementados a los dispositivos físicos en cualquier instante tras la depuración de posibles colisiones y errores en el código. Para materializar los módulos, se utilizan simuladores de programación fuera de línea como RoboDK, para la representación gráfica del proceso y la secuencia de trabajo, URSim como modelo fisicomatemático de los robot UR3e, CODESYS permite emular un softPLC con la lógica de control, este componente además tiene embebido un WebServer para representar todos los datos y el control de la celda, el cuál presenta un concepto de diseño gráfico desarrollado desde los cimientos del proyecto por el autor, al mismo tiempo y con el mismo propósito, una pantalla HMI de Omron actúa como interfaz hombre-máquina. La visión artificial se programa empleando lenguaje Python, el cual aprovecha las APIs de RoboDK para simular una cámara 2D. Todos los elementos se interrelacionan entre sí en tiempo real y se comunican con dos servicios de cloud computing, comenzando por UBIDOTS para monitorizar parámetros de la celda y establecer la posición en la cual el primer robot entrega la pieza en el proceso de intercambio a través de variables que se manipulan por medio de protocolos como OPC UA, MQTT, JSON y nodos implementados en Node-Red, la segunda plataforma está desarrollada por CODESYS Automation Server, un servicio web donde el usuario puede consultar el estado del controlador, distintas versiones del código entre las cuales se puede elegir cual ejecutar, o programar directamente el autómata desde un navegador web rompiendo la barrera de la programación anclada a un PC convencional, pudiendo realizar dicha tarea en una Tablet o un Smartphone en cualquier punto geográfico.	360,21	TRESCIENTOS SESENTA EUROS CON VEINTIUN CÉNTIMOS

VALENCIA, Junio de 2021
Ingeniero Industrial Automático



Dayron Rodríguez Díaz

CUADRO DE MEDICIÓN

Presupuesto parcial nº 1 DESARROLLO DE LA INVESTIGACION

Nº	Ud	Descripción	Medición
1.1	H	Proceso inicial de investigación para recavar ideas que encaminen los objetivos del TFG. Brainstorming sobre las distintas herramientas a incluir en el proyecto, selección del conjunto de softwares finales para dar solución a los objetivos, búsqueda de documentación sobre la cual apoyar el proceso de investigación así como la delimitación de los puntos centrales del TFG. Este proceso se inicia con fecha 01-02-2021 y concluye el 26-02-2021.	
		Total h .....:	1,000
1.2	H	Proceso de investigación, programación, configuración y depuración de la lógica de control implementada en el PLC. Se realizan los siguientes trabajos es esta partida: - Configuración Módulos Comunicación - Programación Lógica - Configuración Alarmas - Comunicación Nube-HMI Este proceso se inicia el 05-02-2021 y concluye el día 09-04-2021	
		Total h .....:	1,000
1.3	H	Proceso de investigación, aprendizaje y entrenamiento para poder manejar los robots UR3e empleados para intercambiar una pieza en la celda. Se desarrollan las siguientes tareas en este apartado: - Configuración máquinas virtuales - Conexión RoboDK-CODESYS-URSim - Configuración programas generados Este proceso se inicia el 04-02-2021 y concluye el día 09-04-2021	
		Total h .....:	1,000
1.4	H	Proceso de investigación, aprendizaje y manejo de las herramientas presentes en el programa. Se desarrollas las siguientes tareas en la presente partida: - Primeros pasos, importar objetos (TCP) - Comunicación PLC-URSim - Python (Visión y comunicación OPC UA) - Secuencia de trabajo - Generar programas para URSim Este proceso se inicia el 08-02-2021 y concluye el día 09-04-2021	
		Total h .....:	1,000
1.5	H	Uno de los objetivos marcados en la confección del presente proyecto, es la presentación de un plan de diseño tanto de la experiencia de usuario como la interfaz de usuario que asegure una representación, interacción y relación de todos los datos y procesos incluidos en la plataforma con una presentación depurada, funcional y equiparable a las actuales tendencias de diseño de aplicaciones digitales empleando una pantalla HMI o el servicio alojado en el PLC WebServer. Las tareas desarrolladas en esta partida son: - Brainstorming Modboard	
1.6	H	Proceso de desarrollo y programación de la unidad de interfaz gráfica fundamental del proyecto. Servidos web alojado dentro del softPLC que emula el control de la celda a través de un PLC virtual. Las tareas desarrolladas en esta partida son: - Pantallas secundarias - Pantallas principales	
		Total h .....:	1,000
1.7	H	Proceso de desarrollo e implementación de la lógica de control y comunicación del HMI. Las tareas desarrolladas en esta partida incluyen: - Configuración comunicaciones - Pantallas	
		Total h .....:	1,000
1.8	H	Desarrollo de aplicaciones IoT. Proceso de configuración de Node. js y Node-Red, entendimiento de protocolos MQQT y JSON, enlace del PLC con la Nube. Las tareas desarrolladas en esta partida son: - Instalación Node-Red - Programación Nodos - JSON y MQTT	
		Total h .....:	1,000

Presupuesto parcial nº 1 DESARROLLO DE LA INVESTIGACION

Nº	Ud	Descripción	Medición
1.9	H	Proceso de creación de toda la infraestructura que da soporte al diseño necesario para el envío y la respresentación de datos de la celda en un servidor en la Nube. Se crea una cuenta FOR BUSINESS en la página web del proveedor de servicios de cloud computing [https://ubidots.com/] que tiene un periodo de prueba de 1 mes con la mayoría de las carcaterísticas disponibles. Las tareas de desarrolladas en esta partida son: - Configuración device - Dahsboard - Programación eventos	
		Total h .....:	1,000
1.10	H	CODESYS Automation Server es la plataforma Industry 4.0 para más de 100,000 usuarios de CODESYS en todo el mundo. El servidor les ayuda a administrar sus controladores y aplicaciones relacionadas y a implementar actualizaciones. Al mismo tiempo, la plataforma ofrece una infraestructura segura para acceso remoto, depuración o visualización web. Las tareas desarrolladas en esta partida son: - Configuración - Enlace PLC-Plataforma	
		Total h .....:	1,000

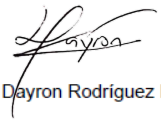
Presupuesto parcial nº 2 HARDWARE

Nº	Ud	Descripción	Medición
2.1	U	Ordenador ubicado en el departamento de Ingeniería de Sistemas y Automática de la UPV sobre el cual se diseño la plataforma DICAR presentada en la Memoria Técnica Descriptiva del TFG.	
		Total u .....:	**1,000**

Presupuesto parcial nº 3 SOFTWARE

Nº	Ud	Descripción	Medición
3.1	U	La plataforma virtual permite crear un Digital Twin de una celda automatizada con dos robots colaborativos que intercambian una pieza empleando visión artificial. La plataforma es completamente gratuita y permite a futuros alumnos e interesados entrenarse o desarrollar proyectos de investigación rompiendo las barreras de acceso a estas tecnologías involucradas en la Industria 4.0. El Digital Twin habilita la migración de todos los programas implementados a los dispositivos físicos en cualquier instante tras la depuración de posibles colisiones y errores en el código. Para materializar los módulos, se utilizan simuladores de programación fuera de línea como RoboDK, para la representación gráfica del proceso y la secuencia de trabajo, URSim como modelo fisicomatemático de los robot UR3e, CODESYS permite emular un softPLC con la lógica de control, este componente además tiene embebido un WebServer para representar todos los datos y el control de la celda, el cuál presenta un concepto de diseño gráfico desarrollado desde los cimientos del proyecto por el autor, al mismo tiempo y con el mismo propósito, una pantalla HMI de Omron actúa como interfaz hombre-máquina. La visión artificial se programa empleando lenguaje Python, el cual aprovecha las APIs de RoboDK para simular una cámara 2D. Todos los elementos se interrelacionan entre sí en tiempo real y se comunican con dos servicios de cloud computing, comenzando por UBIDOTS para monitorizar parámetros de la celda y establecer la posición en la cual el primer robot entrega la pieza en el proceso de intercambio a través de variables que se manipulan por medio de protocolos como OPC UA, MQTT, JSON y nodos implementados en Node-Red, la segunda plataforma está desarrollada por CODESYS Automation Server, un servicio web donde el usuario puede consultar el estado del controlador, distintas versiones del código entre las cuales se puede elegir cual ejecutar, o programar directamente el autómata desde un navegador web rompiendo la barrera de la programación anclada a un PC convencional, pudiendo realizar dicha tarea en una Tablet o un Smartphone en cualquier punto geográfico.	
		Total u .....:	**1,000**

VALENCIA, Junio de 2021
Ingeniero Industrial Automático



Dayron Rodríguez Díaz

PRESUPUESTO

Presupuesto parcial nº 1 DESARROLLO DE LA INVESTIGACION

Num. Código	Ud	Denominación	Cantidad	Precio (€)	Total (€)
1.1 CAP_01_INV.1	h	Proceso inicial de investigación para recavar ideas que encaminen los objetivos del TFG. Brainstorming sobre las distintas herramientas a incluir en el proyecto, selección del conjunto de softwares finales para dar solución a los objetivos, búsqueda de documentación sobre la cual apoyar el proceso de investigación así como la delimitación de los puntos centrales del TFG. Este proceso se inicia con fecha 01-02-2021 y concluye el 26-02-2021.	1,000	4.721,70	4.721,70
1.2 CAP_01_INV.2	h	Proceso de investigación, programación, configuración y depuración de la lógica de control implementada en el PLC. Se realizan los siguientes trabajos es esta partida: - Configuración Módulos Comunicación - Programación Lógica - Configuración Alarmas - Comunicación Nube-HMI Este proceso se inicia el 05-02-2021 y concluye el día 09-04-2021	1,000	12.099,36	12.099,36
1.3 CAP_01_INV.3	h	Proceso de investigación, aprendizaje y entrenamiento para poder manejar los robots UR3e empleados para intercambiar una pieza en la celda. Se desarrollan las siguientes tareas en este apartado: - Configuración máquinas virtuales - Conexión RoboDK-CODESYS-URSim - Configuración programas generados Este proceso se inicia el 04-02-2021 y concluye el día 09-04-2021	1,000	6.713,67	6.713,67
1.4 CAP_01_INV.4	h	Proceso de investigación, aprendizaje y manejo de las herramientas presentes en el programa. Se desarrollas las siguientes tareas en la presente partida: - Primeros pasos, importar objetos (TCP) - Comunicación PLC-URSim - Python (Visión y comunicación OPC UA) - Secuencia de trabajo - Generar programas para URSim Este proceso se inicia el 08-02-2021 y concluye el día 09-04-2021	1,000	10.550,05	10.550,05
1.5 CAP_01.INV.5	h	Uno de los objetivos marcados en la confección del presente proyecto, es la presentación de un plan de diseño tanto de la experiencia de usuario como la interfaz de usuario que asegure una representación, interacción y relación de todos los datos y procesos incluidos en la plataforma con una presentación depurada, funcional y equiparable a las actuales tendencias de diseño de aplicaciones digitales empleando una pantalla HMI o el servicio alojado en el PLC WebServer. Las tareas desarrolladas en esta partida son: - Brainstorming Modboard - Conceptos y bocetos - Tipografía - Infografía	1,000	2.065,74	2.065,74

Presupuesto parcial nº 1 DESARROLLO DE LA INVESTIGACION

Num.	Código	Ud	Denominación	Cantidad	Precio (€)	Total (€)
1.6	CAP_01.INV.6	h	Proceso de desarrollo y programación de la unidad de interfaz gráfica fundamental del proyecto. Servidos web alojado dentro del softPLC que emula el control de la celda a través de un PLC virtual. Las tareas desarrolladas en esta partida son: - Pantallas secundarias - Pantallas principales	1,000	8.041,65	8.041,65
1.7	CAP_01_INV.7	h	Proceso de desarrollo e implementación de la lógica de control y comunicación del HMI. Las tareas desarrolladas en esta partida incluyen: - Configuración comunicaciones - Pantallas	1,000	2.360,85	2.360,85
1.8	CAP_01.INV.8	h	Desarrollo de aplicaciones IoT. Proceso de configuración de Node. js y Node-Red, entendimiento de protocolos MQQT y JSON, enlace del PLC con la Nube. Las tareas desarrolladas en esta partida son: - Instalación Node-Red - Programación Nodos - JSON y MQTT	1,000	5.975,90	5.975,90
1.9	CAP_01.INV.9	h	Proceso de creación de toda la infraestructura que da soporte al diseño necesario para el envío y la respresentación de datos de la celda en un servidor en la Nube. Se crea una cuenta FOR BUSINESS en la página web del proveedor de servicios de cloud computing [https://ubidots.com/] que tiene un período de prueba de 1 mes con la mayoría de las carcaterísticas disponibles. Las tareas de desarrolladas en esta partida son: - Configuración device - Dahsboard - Programación eventos	1,000	2.434,63	2.434,63
1.10	CAP_01.INV.10	h	CODESYS Automation Server es la plataforma Industry 4.0 para más de 100,000 usuarios de CODESYS en todo el mundo. El servidor les ayuda a administrar sus controladores y aplicaciones relacionadas y a implementar actualizaciones. Al mismo tiempo, la plataforma ofrece una infraestructura segura para acceso remoto, depuración o visualización web. Las tareas desarrolladas en esta partida son: - Configuración - Enlace PLC-Plataforma	1,000	1.475,53	1.475,53
			Total presupuesto parcial nº 1 DESARROLLO DE LA INVESTIGACION :			**56.439,08**

Presupuesto parcial nº 2 HARDWARE

Num.	Código	Ud	Denominación	Cantidad	Precio (€)	Total (€)
2.1	CAP_02_Ordena…	u	Ordenador ubicado en el departamento de Ingeniería de Sistemas y Automática de la UPV sobre el cual se diseño la plataforma DICAR presentada en la Memoria Técnica Descriptiva del TFG.	1,000	513,00	513,00
			Total presupuesto parcial nº 2 HARDWARE :			**513,00**

Presupuesto parcial nº 3 SOFTWARE

Num.	Código	Ud	Denominación	Cantidad	Precio (€)	Total (€)
3.1	CAP_03_CODE...	u	La plataforma virtual permite crear un Digital Twin de una celda automatizada con dos robots colaborativos que intercambian una pieza empleando visión artificial. La plataforma es completamente gratuita y permite a futuros alumnos e interesados entrenarse o desarrollar proyectos de investigación rompiendo las barreras de acceso a estas tecnologías involucradas en la Industria 4.0. El Digital Twin habilita la migración de todos los programas implementados a los dispositivos físicos en cualquier instante tras la depuración de posibles colisiones y errores en el código. Para materializar los módulos, se utilizan simuladores de programación fuera de línea como RoboDK, para la representación gráfica del proceso y la secuencia de trabajo, URSim como modelo fisicomatemático de los robot UR3e, CODESYS permite emular un softPLC con la lógica de control, este componente además tiene embebido un WebServer para representar todos los datos y el control de la celda, el cuál presenta un concepto de diseño gráfico desarrollado desde los cimientos del proyecto por el autor, al mismo tiempo y con el mismo propósito, una pantalla HMI de Omron actúa como interfaz hombre-máquina. La visión artificial se programa empleando lenguaje Python, el cual aprovecha las APIs de RoboDK para simular una cámara 2D. Todos los elementos se interrelacionan entre sí en tiempo real y se comunican con dos servicios de cloud computing, comenzando por UBIDOTS para monitorizar parámetros de la celda y establecer la posición en la cual el primer robot entrega la pieza en el proceso de intercambio a través de variables que se manipulan por medio de protocolos como OPC UA, MQTT, JSON y nodos implementados en Node-Red, la segunda plataforma está desarrollada por CODESYS Automation Server, un servicio web donde el usuario puede consultar el estado del controlador, distintas versiones del código entre las cuales se puede elegir cual ejecutar, o programar directamente el autómata desde un navegador web rompiendo la barrera de la programación anclada a un PC convencional, pudiendo realizar dicha tarea en una Tablet o un Smartphone en cualquier punto geográfico.	1,000	360,21	360,21
			Total presupuesto parcial nº 3 SOFTWARE :			**360,21**

Presupuesto de ejecución material

	Importe (€)
1 DESARROLLO DE LA INVESTIGACION	56.439,08
2 HARDWARE	513,00
3 SOFTWARE	360,21
Total	57.312,29

Asciende el presupuesto de ejecución material a la expresada cantidad de CINCUENTA Y SIETE MIL TRESCIENTOS DOCE EUROS CON VEINTINUEVE CÉNTIMOS.

VALENCIA, Junio de 2021
Ingeniero Industrial Automático

Dayron Rodríguez Díaz

RESUMEN DEL PRESUPUESTO

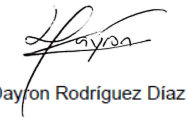
Capítulo	Importe	%
Capítulo 1 DESARROLLO DE LA INVESTIGACION	56.439,08	98,48
Capítulo 2 HARDWARE	513,00	0,90
Capítulo 3 SOFTWARE	360,21	0,63
Presupuesto de ejecución material	**57.312,29**	
13% de gastos generales	7.450,60	
6% de beneficio industrial	3.438,74	
Suma	68.201,63	
21% IVA	14.322,34	
Presupuesto de ejecución por contrata	**82.523,97**	

Honorarios de Redacción del Proyecto

Proyecto	4,13% sobre PEM	2.367,00
IVA	21% sobre honorarios de Proyecto	497,07
	Total honorarios de Proyecto	2.864,07
Dirección de obra	4,13% sobre PEM	2.367,00
IVA	21% sobre honorarios de Dirección de obra	497,07
	Total honorarios de Dirección de obra	2.864,07
	Total honorarios de Redacción del Proyecto	**5.728,14**
	Total honorarios	**5.728,14**
	Total presupuesto general	**88.252,11**

Asciende el presupuesto general a la expresada cantidad de OCHENTA Y OCHO MIL DOSCIENTOS CINCUENTA Y DOS EUROS CON ONCE CÉNTIMOS.

VALENCIA, Junio de 2021
Ingeniero Industrial Automático



Dayron Rodríguez Díaz