Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Servicio Industrial de Reconocimiento de Patrones mediante Tecnología Serverless

Autor: Antonio León Martín

Tutor/es: Francisco Javier Rodríguez Sánchez

2021

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado

Servicio Industrial de Reconocimiento de Patrones mediante Tecnología Serverless

Autor: Antonio León Martín

Director/es: Francisco Javier Rodríguez Sánchez

TRIBUNAL:

Presidente: Pedro Martín Sánchez

Suplente: Ignacio Fernández Lorenzo

Vocal 2º: Julio Pastor Mendoza

CALIFICACIÓN: .....

FECHA: .....

Gracias a mis padres, por todas las facilidades que me han dado y me siguen dando para el desarrollo de mi vida académica y profesional.

A mis hermanos, por sus continuos ánimos y ayuda.

A Laura, cuyo ejemplo y constante apoyo han sido fundamentales para llevar a cabo el presente proyecto.

A mis amigos, por animarme a seguir adelante.

La presente versión del documento ha sido realizada de acuerdo con las modificaciones propuestas por el tribunal durante el proceso de evaluación y defensa de el trabajo de fin de grado.

1. Índice

2. Índice de figuras

3. Resumen en castellano

Los cobots se postulan como una alternativa a los trabajadores para la realización de tareas monótonas y de alta repetibilidad. Poseen el beneficio de poder trabajar en un entorno compartido con humanos.

Por otro lado, la tecnología sin servidor ofrece numerosas ventajas como reducción de costos de mantenimiento de servidores y facilidad de escalado.

El presente proyecto pretende realizar la integración de estas 2 tecnologías, a través de un autómata, en un entorno de simulación y mostrar las ventajas y posibilidades que pueden ofrecer a las aplicaciones industriales.

4. Abstract

Cobots are postulated as an alternative to workers for the performance of monotonous and highly repeatable tasks. They have the benefit of being able to work in an environment shared with humans.

On the other hand, serverless technology offers numerous advantages such as reduced server maintenance costs and ease of scaling.

This project aims to integrate these 2 technologies, through a PLC, in a simulation environment and show the advantages and possibilities that they can offer to industrial applications.

5. Palabras clave

- Cobot
- Serverless
- Rekognition
- Robot
- Autómata
- PLC
- AWS

6. Introducción

El objetivo principal del presente proyecto es mostrar tanto los beneficios que puede llegar a proporcionar el uso de tecnologías *serverless* en el ámbito industrial como las virtudes que nos otorgan los robots colaborativos, cada vez más presentes en el día a día de las empresas que buscan una mejora notable de la eficiencia y productividad de sus procesos de producción.

A continuación, se realiza una breve introducción de los sistemas principales que constituyen el grueso del presente proyecto.

6.1. Serverless

Serverless (sin servidor) hace referencia a la arquitectura nativa de la nube. En las arquitecturas *serverless* los servidores "dejan de existir" para los desarrolladores, de manera que les permite centrarse en la implementación, evolución y desarrollo del producto principal delegando las tareas de mantenimiento, gestión, administración y actualización de la infraestructura a proveedores externos como pueden ser Microsoft, Google, Amazon, IBM, etc.

Este tipo de arquitecturas otorgan numerosos beneficios tanto a los desarrolladores particulares como a las empresas, en especial a las pequeñas entidades que no pueden disponer de recursos, equipos o departamentos destinados exclusivamente a las tareas de gestión y mantenimiento de la infraestructura. De esta forma dichas labores, que para pequeñas organizaciones y en entornos con arquitecturas con servidor recaerían sobre el propio desarrollador, pueden ser externalizadas otorgando al equipo de desarrollo la posibilidad de enfocarse única y exclusivamente en la implementación del producto que le proporcionará valor tanto a la propia entidad como a sus clientes.

En la computación sin servidor el proveedor de la nube es el encargado de realizar una asignación dinámica de los recursos para ejecutar uno o varios fragmentos de código. Estos fragmentos de código se ejecutan, por norma general, en contenedores que carecen de estado y que son activados por una serie de eventos como pueden ser servicios de colas, peticiones http, eventos programados y un largo etcétera. El código que se ejecuta en la nube, normalmente, lo hace en forma de función, es por esto por lo que en determinadas ocasiones a la tecnología *serverless* se la conoce también como "Funciones como servicio" ("*FaaS*"). Las principales *FaaS* que encontramos en la actualidad en el mercado son:

- Azure Function (Microsoft Azure)
- AWS Lambda (AWS)
- Cloud Function (Google Cloud)
- IBM Cloud Function (IBM Cloud)

Los contenedores donde se ejecutan las funciones se activan bajo demanda y una vez que ha finalizado la función puede que se mantengan activos durante un determinado periodo de tiempo. Esto hace que, si se desencadena un nuevo evento que actúe como disparador de esa función durante este tiempo, la respuesta sea mucho más rápida que la primera vez que se ejecutó el código. A esto se le conoce como "arranque en caliente", mientras que la primera ejecución de la función se denomina "arranque en frío". El tiempo de duración de un arranque

en frío depende de varios factores cómo pueden ser el proveedor de la nube, el lenguaje de código utilizado para el desarrollo de la función y el tamaño de esta.

En el lado opuesto se encuentran los modelos estándar de computación en la nube de "Infraestructura como servicio" ("*IaaS*") en la que los clientes son los encargados de ampliar o reducir la capacidad en función de la demanda que tengan que abastecer. Es por ello, que esta asignación automática de recursos se está convirtiendo en un atractivo cada vez mayor para todo tipo de corporaciones dado que les permite reducir costes en departamentos y tareas del tipo IT (*Information Technology*) y destinar el correspondiente presupuesto a tareas y proyectos cuyo retorno sea mayor.

A continuación, se listan algunas de las funciones que desempeñan los proveedores en la nube en la computación sin servidor y las cuales hacen de este tipo de informática una de las alternativas más potentes y cada vez más elegida para la ejecución de proyectos de diversos campos:

- Gestión del sistema operativo
- Gestión de archivos
- Ejecución de parches de seguridad
- Balanceo de carga (distribución del trabajo a realizar en varios procesos, máquinas, recursos, etc)
- Administración de la capacidad de memoria
- Adaptación de recursos
- Registro y supervisión del sistema

Serverless proporciona una serie de beneficios y ventajas que hacen de la elección de esta tecnología a la hora de llevar a cabo proyectos una decisión cada vez más común y arraigada entre las entidades. Las principales ventajas que se encuentran a la hora de hacer uso de esta tecnología son las siguientes:

- Reducción de los tiempos de desarrollo. El hecho de que los desarrolladores no tengan que ocuparse de ningún servidor hace que puedan mantener el foco en la implementación del producto principal, por lo que los tiempos de desarrollo y de comercialización se reducen significativamente.
- Disminución de los costes. Al ser una tecnología sin servidor hace que se ahorren los gastos tanto de infraestructura como los derivados en la mano de obra responsable del mantenimiento de esta. Además, el hecho de que el pago sea por tiempo de uso del proceso hace de esta opción la mejor, económicamente hablando, a la hora de valorar el uso de una tecnología con servidor o sin él.
- Escalamiento fácil y eficiente. Este tipo de arquitectura excluye el riesgo de no poder afrontar un crecimiento exponencial dado que las aplicaciones pueden escalarse de forma automática o, en el peor de los casos, mediante unos pocos "clics" que permitan aumentar las capacidades y recursos del sistema.
- Autonomía para las entidades. Al no tener que realizar las tareas de bajo nivel de mantenimiento de servidores, las organizaciones pueden enfocarse en los productos y ofertas comerciales principales destinando a estos el dinero y tiempo derivados

otorgándoles la necesidad de mejorar su competitividad frente a otras entidades del mismo sector.

A continuación, se listan algunos casos de uso de arquitecturas sin servidor interesantes:

- Procesamiento de macrodatos, como el ofrecido por Azure [1] y que determina la velocidad media de los viajes en taxi en la ciudad de Nueva York por día en el año 2017.
- Construcción de un repositorio de almacenamiento (*data lake*) en AWS por Coca-Cola Andina que les permitió aumentar la productividad de análisis en un 80% para una mejor toma de decisiones basadas en datos [2].
- Automatización de procesos por parte de Amazon Robotics a través de modelos de *Machine Learning* que retornó en un aumento del rendimiento de un 40% y una reducción de costos del 20% como se puede ver en [3].

6.1.1. AWS

Para el presente proyecto se ha decidido trabajar con AWS debido a la amplia gama de servicios que presenta esta plataforma en la nube con el fin de que este pueda servir como precedente para la investigación o desarrollo de futuros proyectos de naturaleza similar.

AWS comenzó su andadura en 2006, cuando comenzó a proporcionar algunos servicios de infraestructura IT para algunas compañías en forma de servicio web. Hoy en día, abastece de infraestructura a empresas de 190 países y posee centros de datos en:

- Estados Unidos
- Europa
- Brasil
- Singapur
- Japón
- Australia

AWS ofrece multitud de soluciones, algunas de las más populares son:

- Backup y almacenamiento de datos
- TI empresarial
- Bases de datos
- Sitios Web
- Alojamiento de aplicaciones
- Entrega de contenido

Amazon Web Services ofrece más de 200 servicios que hacen de esta plataforma en la nube ser la más adoptada y completa. A continuación, se presentan algunos de estos interesantes servicios:

- Amazon EC2: Proporciona capacidad informática en la nube segura y de tamaño ajustable que permite casi cualquier carga de trabajo.
- Amazon S3: Proporciona almacenamiento de objetos ofreciendo una gran escalabilidad, seguridad, disponibilidad de datos y rendimiento. Permite recuperar cualquier volumen de datos desde cualquier ubicación.

- Amazon Lambda: Este servicio permite ejecutar código para casi cualquier tipo de aplicación sin necesidad de administrar los servidores.
- Amazon SageMaker: Proporciona a los desarrolladores y analistas de datos facilidades a la hora de trabajar con modelos de aprendizaje automático.
- **Amazon Rekognition**: Este servicio, que es el aplicado en este proyecto, permite el análisis de imágenes y vídeos y la identificación de objetos, personas, textos, escenas y actividades en imágenes y videos y detectar cualquier contenido inapropiado.

6.1.1.1. Amazon Rekognition

Amazon Rekognition es el servicio que ofrece AWS para el procesamiento de imágenes y el cual se utilizará en el presente proyecto para la identificación de piezas.

Uno de los puntos fuertes que ofrece Rekognition es que no requiere de experiencia previa en *Machine Learning* para su uso y además dispone de una API sencilla y fácil de utilizar. Además, se encuentra en continuo proceso de aprendizaje por lo que se agregan, por ejemplo, etiquetas y funciones de comparación facial de manera continua.

Rekognition permite multitud de casos de uso, a continuación, se enumeran los más habituales:

- Verificación facial: Rekognition permite confirmación de usuario en las aplicaciones comparando la imagen en vivo con una imagen de referencia.
- Detección de equipos de protección personal (PPE) como mascarillas, cascos y guantes y que puede ser utilizado en industrias donde la seguridad tenga una prioridad máxima como la construcción o la manufactura.
- Detección de contenido inapropiado: Rekognition puede detectar contenido para adultos y violento en imágenes y vídeos. El uso de etiquetas permite un filtrado granular y la gestión de grandes volúmenes de contenido.
- Detección de texto: Amazon Rekognition admite la mayoría de las fuentes, es capaz de detectar texto y números en diferentes orientaciones.
- Etiquetas personalizadas: Mediante las etiquetas personalizadas se pueden identificar en imágenes objetos y escenas específicas para las necesidades del usuario. De esta manera se puede desarrollar un modelo personalizado cuyos requerimientos de tiempo, experiencia y recursos dependen mucho de las imágenes y objetos o escenas a analizar.
- Análisis por etiquetas: Las etiquetas se pueden referir a objetos, eventos, conceptos o actividades y Amazon Rekognition puede detectar etiquetas tanto en imágenes como en vídeos. Por ejemplo, el presente proyecto se basa en el análisis de imágenes que contengan las etiquetas “Gear” o “Washer” para la clasificación de piñones y arandelas.

6.2. Cobots

El término cobot aparece de la unión de las palabras “robot” y “colaboración”. Por definición un cobot es un robot articulado industrial diseñado y fabricado para trabajar en un entorno común a los operarios y que permita y facilite la interacción entre ambas partes para potenciar la productividad de las organizaciones.

Los cobots, a diferencia de los robots industriales tradicionales, que se encuentran en las fábricas aislados en espacios delimitados por vallas o jaulas de seguridad, comparten el puesto de trabajo con los operarios. Permiten realizar tareas de automatización repetitivas, que pueden resultar tediosas o peligrosas para los humanos, otorgando a estos últimos la posibilidad de enfocarse en labores más específicas. De esta forma, estos robots (en la gran mayoría de los casos brazos articulados) permiten aunar su precisión y potencial a la capacidad para atender problemas de los humanos.

En la actualidad los cobots se instalan especialmente en fábricas del sector industrial; en los sectores en los que se encuentran más presentes son en el del automóvil, en el aeroespacial, en la construcción y en la fabricación de componentes electrónicos. En estos sectores desempeñan una gran diversidad de labores como pueden ser tareas de soldadura, paletizado, *Pick & Place*, atornillado, etc.

Existen diferentes particularidades que diferencian a los cobots de los modelos tradicionales industriales. El más característico, y al cual ya se ha hecho referencia en el presente, es que no disponen de un espacio o entorno de seguridad como los modelos tradicionales, sino que comparten espacio con los trabajadores. Además de esta, se encuentran otras características claramente diferenciales como pueden ser sus reducidas dimensiones y su coste notablemente inferior al de los robots industriales que hace que la amortización de los cobots sea mucho más rápida y bastante más notoria. Se detallan a continuación pues las principales ventajas que ofrece un cobot frente a un robot industrial tradicional:

- Programación mucho más sencilla. Los robots colaborativos pueden ser programados y configurados por personas sin experiencia de manera relativamente simple.
- Rápida instalación y puesta en marcha. El montaje y la instalación de los cobots para su puesta en marcha abarca unas horas frente a las semanas que requieren las mismas tareas en el caso de un robot tradicional.
- Precio económico que se amortiza fácilmente y de manera rápida. El retorno de inversión suele ser inferior a un año.
- Gran flexibilidad. Sus reducidas dimensiones facilitan su reasignación a múltiples tareas y/o aplicaciones, permitiendo una gran adaptabilidad a las necesidades oportunas en cada momento.
- Minimizan el cansancio y el riesgo laboral dado que permiten liberar a las personas de las tareas más monótonas, aburridas y peligrosas.
- Aportan total seguridad al disponer de un sistema programado para detenerse y evitar accidentes.

Como ya se ha mencionado con anterioridad en este mismo apartado existen diferencias de seguridad entre los robots industriales y los cobots. Esto se debe a que los cobots están equipados con sensores que les ofrecen la capacidad de detenerse en caso de obstrucción o necesidad. Esto, añadido a su reducido peso hace que un choque con ellos no sea peligroso y posibilita su instalación sin necesidad de jaulas de seguridad.

Si bien, aunque las diferencias en la seguridad son importantes lo que más distingue a un robot industrial de un cobot es en el propósito de cada uno de ellos. Los robots industriales, por un lado, están diseñados, planificados y construidos con un fin específico y que realizará de manera

óptima. Por el contrario, un brazo robótico colaborativo se puede desarrollar para realizar una única tarea o varias en función de las necesidades de la producción gracias, por ejemplo, a su facilidad de reubicación o de reprogramación.

Además, aunque los cobots normalmente se destinan a realizar los trabajos más repetitivos y manuales tanto en cadenas de producción como en series de trabajos también pueden encargarse de trabajos que supongan un riesgo para la integridad de los trabajadores como puede ser la clasificación o movimiento de piezas cortantes o trabajos en condiciones ambientales hostiles. Según un estudio del MIT (*Massachusetts Institute of Technology*) la colaboración entre humanos y cobots hace que la productividad de ambos aumente en 85% con respecto a cuando realizan los trabajos de manera independiente.

Los robots colaborativos son un elemento clave en la Industria 4.0 permitiendo una mejora de la productividad mediante la automatización industrial, esto ha hecho que muchas entidades hayan optado por la innovación y por fabricar cobots. Algunas de las marcas más destacadas son:

- Universal Robots
- ABB
- Omron
- Kuka
- Yaskawa
- Fanuc

Los cobots, aparte de colaborar con los trabajadores como ya se ha comentado, pueden comunicarse con otros equipos y periféricos de manera que pueden compartir tareas con otros robots y equipos externos y periféricos como pueden ser PLCs, CNCs o cintas transportadoras.

Las principales aplicaciones que llevan a cabo los brazos robóticos colaborativos son las siguientes:

- Montaje: Los cobots permiten realizar procesos de montaje precisos y repetitivos como atornillar, instalar e insertar piezas.
- Dispensación: Los robots colaborativos permiten agregar flexibilidad, eficiencia y libertad por ejemplo a las tareas de pegado, sellado y pintura.
- Acabado: Tareas de pulido, abrillantado o lijado requieren normalmente un contacto delicado fácilmente de conseguir con el control de fuerza de los cobots.
- Alimentación de máquinas: Los cobots permiten disminuir las tareas más físicamente exigentes y repetitivas para los operarios de las máquinas ocupándose de las máquinas de control numérico, de moldeo por inyección y plegadoras, por ejemplo.
- Manipulación de materiales: Los brazos robóticos colaborativos también pueden optimizar las labores de manipulación, empaquetado y paletizado, etiquetado, etc. permitiendo a los trabajadores liberarse de tareas simples pero repetitivas.
- Extracción de material: Los cobots otorgan flexibilidad, libertad y eficiencia a las operaciones de rectificado, desbarbado, fresado, enrutamiento, perforación y otras tareas de extracción de material.

- Control de calidad: Los cobots realizan de manera regular y repetida los trabajos definidos de manera que proporcionan las condiciones óptimas de cara a mediciones o inspecciones de calidad.
- Soldadura: Los cobots permiten otorgar eficiencia, flexibilidad y libertad a las tareas de soldadura. Pueden realizar soldaduras por arco, TIG, rayos láser, MIG, ultrasónicas, por plasma y por puntos.

A continuación, se enumeran una serie de casos de uso de soluciones que emplean robots colaborativos:

- Automatización de la carga y descarga de placas PCB, así como el ensamblaje de componentes por parte de Continental Automotive Spain mediante brazos robóticos de Universal Robots del modelo UR10e. Como retorno tuvo una reducción de tiempos del 50% y una disminución notable de los materiales defectuosos o desperdiciados [4].
- Inserción y extracción de componentes plásticos en torno de control numérico por parte de Linatex mediante el robot colaborativo UR5e [5].

6.3. Objetivos

El objetivo del presente proyecto consiste en la realización de un aplicativo industrial que realice una operación de pick & place a través del robot colaborativo UR5e de Universal Robots.

La operación de pick & place se basa en la clasificación de dos tipos de piezas: piñones y arandelas. Si se detecta que la pieza a clasificar es una arandela, el robot realizará el movimiento "A" colocándola en su destino final, por el contrario, si la pieza que se debe de clasificar se trata de un piñón el robot ejecutará el movimiento "B" y lo dejará en su posición de clasificación.

Para la operación de clasificación del UR5e se realizarán dos programas distintos:

- Un programa de simulación que permita reproducir la totalidad de eventos y movimientos que realiza el robot a lo largo de la operación.
- Un programa real en el cual exista comunicación con una herramienta de tipo pinza y que esté gobernado por un PLC, CPU 1215C, externo.

Para la operación de clasificación es necesario realizar un reconocimiento y procesamiento de imágenes del cual se encargará el servicio de AWS Rekognition, con el cual comunicará un módulo de Java a desarrollar también en el presente.

El PLC mencionado anteriormente en este apartado será el encargado de la comunicación entre el cobot y el módulo de Java, es decir, se encargará de gobernar el proceso indicando al robot eventos como:

- Cuando tiene el cobot que esperar a que se realice el procesamiento de la foto.
- Cuando tiene que realizar el cobot el movimiento "A" para colocar una arandela.

Además, el autómata también será el encargado de comunicar al módulo de Java qué foto ha de procesarse y cuando de manera que se mantenga la sincronía de los diferentes ciclos de la operación de pick & place.

A continuación, en la figura 1, se muestra un diagrama de la arquitectura planteada en el presente proyecto en el que se puede ver de manera gráfica el objetivo y alcance de cada uno de los módulos que lo componen:

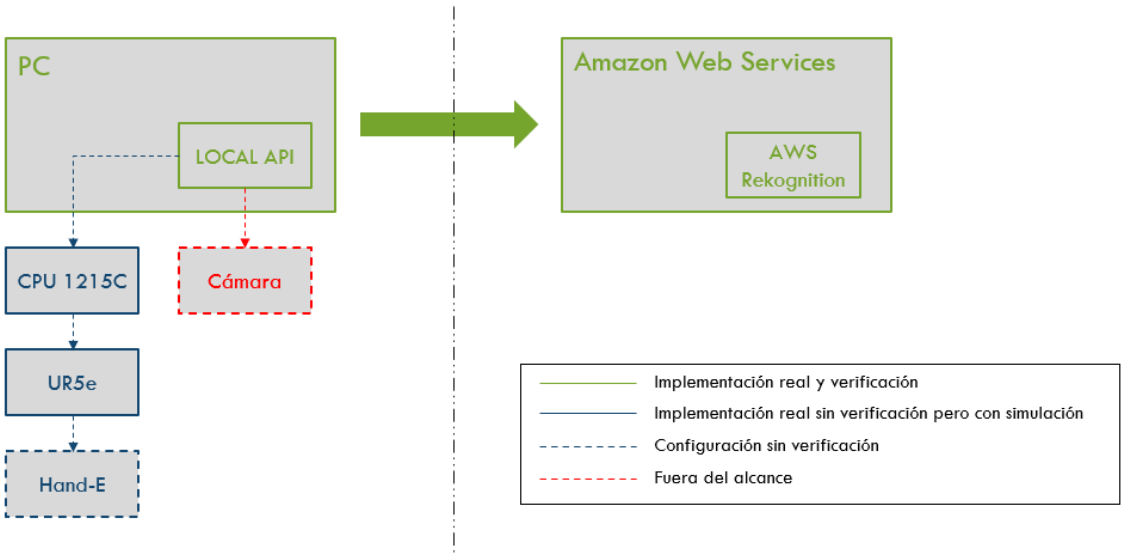

Figura 1. Arquitectura planteada

Por último, en la figura 2, se presenta el diagrama de flujo simplificado que permite mostrar la operativa de la aplicación de forma visual:

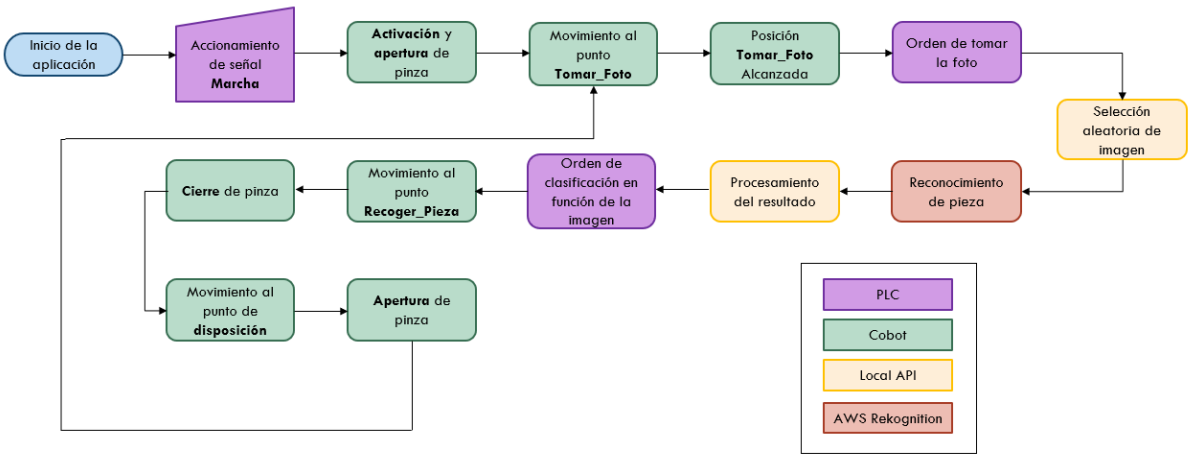

Figura 2. Diagrama de flujo simplificado

6.4. Organización del documento

El presente documento se organiza de forma que se pueden apreciar 4 apartados claramente diferenciados:

1. En el primer apartado estará destinado al robot colaborativo UR5e. En él se hablará de las características básicas del robot, sus especificaciones, así como de la herramienta tipo pinza a utilizar en el caso de la aplicación real. Además, se explicará la configuración y desarrollo del proyecto a través de la herramienta de simulación URSim que facilita Universal Robots.
2. El segundo apartado estará destinado al PLC SIEMENS CPU1215C y en el cual se describirá tanto la configuración del proyecto de manera que permita la comunicación con el cobot UR5e como el desarrollo del programa que permita gobernar la operación de la aplicación final, a través de la herramienta TIA Portal.
3. En el tercer apartado se detallarán los pasos necesarios para configurar el entorno de trabajo para el desarrollo del módulo de Java y permitir la comunicación con AWS Rekognition. Además, se explicará tanto el código desarrollado como la manera de compilarlo y ejecutarlo.
4. Por último, se demostrará el resultado conjunto de la simulación de forma que se permita entender y visualizar el resultado final de la totalidad de la operación.

7. Descripción

En el presente capítulo se explicará de manera detallada las configuraciones, implementaciones y desarrollos llevados a cabo en el proyecto, así como las respectivas pruebas y simulaciones, las cuales se dividen en las tres tecnologías mencionadas en el capítulo anterior.

- Cobot UR5e
- PLC SIEMENS CPU1215C
- Módulo Java para procesamiento de imágenes a través de Amazon Rekognition.

7.1. UR5e

Para la realización del presente proyecto se ha seleccionado el UR5e de Universal Robots como robot colaborativo debido a una serie de características y ventajas que se describen y se detallan en el transcurso del actual capítulo.

El UR5e es un robot industrial colaborativo ligero cuyo nivel de complejidad de configuración y programación es muy bajo, lo cual le otorga una versatilidad y adaptabilidad óptimas para ser integrado en una gran variedad de aplicaciones. Posee una carga útil de 5 kg y tiene un radio de acción de 850 mm, lo que hace de él un modelo idóneo para la realización de tareas de automatización que requieran un procesamiento de medio-bajo peso. A continuación, en la figura 3 se muestra el aspecto del cobot UR5e.

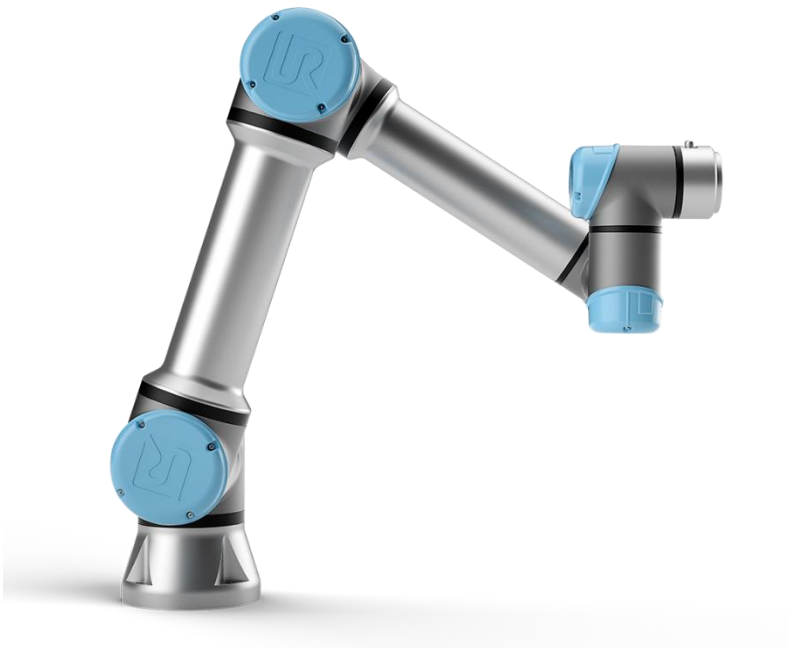

Figura 3. Robot colaborativo UR5e

7.1.1. Especificaciones técnicas

Según la documentación facilitada por Universal Robots [6], se detallan a continuación las especificaciones técnicas del modelo UR5e.

7.1.1.1. Especificaciones técnicas del brazo robot

7.1.1.1.1. Especificaciones

- Carga útil: 5 kg
- Alcance: 850 mm
- Grados de libertad: 6 articulaciones
- Programación: Interfaz gráfica PolyScope

7.1.1.1.2. Rendimiento

- Consumo de energía: 200 W aproximadamente para una aplicación habitual.
- Operación de colaboración: 17 funciones avanzadas de seguridad configurables y control remoto acorde a la ISO 10218.
- Rango: 50 N
- Resolución: 2,5 N
- Precisión: 4N
- Rango de temperatura ambiente: De 0 a 50°C
- Humedad: 90% RH (sin condensación)

7.1.1.1.3. Movimiento

- Repetibilidad de posición: ±0,03 mm con carga, acorde a ISO 9283
- Movimiento de los ejes del UR5e:
  - Base: Radio de acción de ±360° y velocidad máxima de ±180°/s
  - Hombro: Radio de acción de ±360° y velocidad máxima de ±180°/s
  - Codo: Radio de acción de ±360° y velocidad máxima de ±180°/s
  - Muñeca 1: Radio de acción de ±360° y velocidad máxima de ±180°/s
  - Muñeca 2: Radio de acción de ±360° y velocidad máxima de ±180°/s
  - Muñeca 3: Radio de acción de ±360° y velocidad máxima de ±180°/s
- Velocidad típica de TCP: Velocidad máxima de 1m/s

7.1.1.1.4. Funciones

- Clasificación IP54
- Clase ISO Sala limpia 6
- Ruido: Menor a 65 dB
- Posibilidad de montaje del robot en cualquier orientación
- Puertos de entradas y salidas:
  - 2 entradas digitales
  - 2 salidas digitales

- 2 entradas analógicas
- Interfaz UART 9,6k – 5 Mbps

7.1.1.1.5. Características físicas

- Diámetro de huella de 149 mm
- Materiales de fabricación: Aluminio, plásticos de polipropileno y acero
- Conector tipo M8 8-pin para herramienta del cobot
- Longitud del cable: 6 m
- Peso: 20,6 kg (incluyendo el cable)

7.1.1.2. Especificaciones técnicas de la caja de control

7.1.1.2.1. Funciones

- Clasificación IP44
- Clase ISO Sala limpia 6
- Rango de temperatura ambiente: De 0 a 50°C
- Puertos de entradas y salidas:
  - 16 entradas digitales
  - 16 salidas digitales
  - 2 entradas analógicas
  - 2 salidas analógicas
  - Control a 500Hz, dedicadas 4 entradas digitales en cuadratura de alta velocidad
- Entrada/Salida de fuente de alimentación: 24 V/2 A
- Comunicación:
  - Frecuencia de control: 50 Hz
  - Modbus TCP, frecuencia de señal de 500 Hz
  - PROFINET y EtherNet/IP, frecuencia de señal de 500 Hz
  - 1 puerto USB 2.0
  - 1 puerto USB 3.0
- Fuente de alimentación: 100 – 240 VAC, 47 – 400 Hz
- Humedad: 90% RH (sin condensación)

7.1.1.2.2. Características físicas

- Tamaño de la caja de control: 475 mm x 423 mm x 268 mm
- Peso: Máximo de 13,6 kg
- Materiales: Acero

7.1.2. Pinza Hand-E

La pinza Hand-E, confeccionada especialmente para su montaje en robots colaborativos y la cual se muestra en la figura 4 del presente documento, está fabricada por Robotiq. Tiene un recorrido de 50 mm y posee una forma ergonómica que facilita y acomoda el posicionamiento

manual de la misma. Además, tiene una interfaz intuitiva que simplifica en gran medida su programación y su uso.

Figura 4. Pinza Hand-E

7.1.2.1. Especificaciones mecánicas

Según la información técnica extraída de la documentación aportada por Robotiq [7], se define:

- Abertura de la pinza (ajustable): 50 mm
- Fuerza del agarre (ajustable): de 20 a 185 N
- Carga útil máxima recomendada (agarre completo): 5 kg
- Carga útil máxima recomendada (agarre parcial): 4,7 kg
- Peso de la pinza: 1 kg
- Velocidad de cierre: de 20 a 150 mm/s
- Repetibilidad de posición: 0,025 mm
- Resolución de posición: 0,2 mm.
- Detección mínima de pieza: 0,5 mm
- Opciones de protocolo de comunicación: IP67

Según indica el fabricante, todas estas especificaciones han sido calculadas para el uso de cubiertas de silicona en las puntas de los dedos de la pinza y para agarrar un objeto de acero a una aceleración muy baja del robot.

7.1.2.2. Especificaciones eléctricas

- Tensión de alimentación: 24 V DC ± 10%
- Consumo mínimo de energía: 1 W
- Corriente de pico 680 mA

7.1.2.3. Dimensiones técnicas

En la figura 5 se muestran las dimensiones de la Hand-E con los ejes X, Y, Z y el origen referenciado para el movimiento de los dedos de esta.

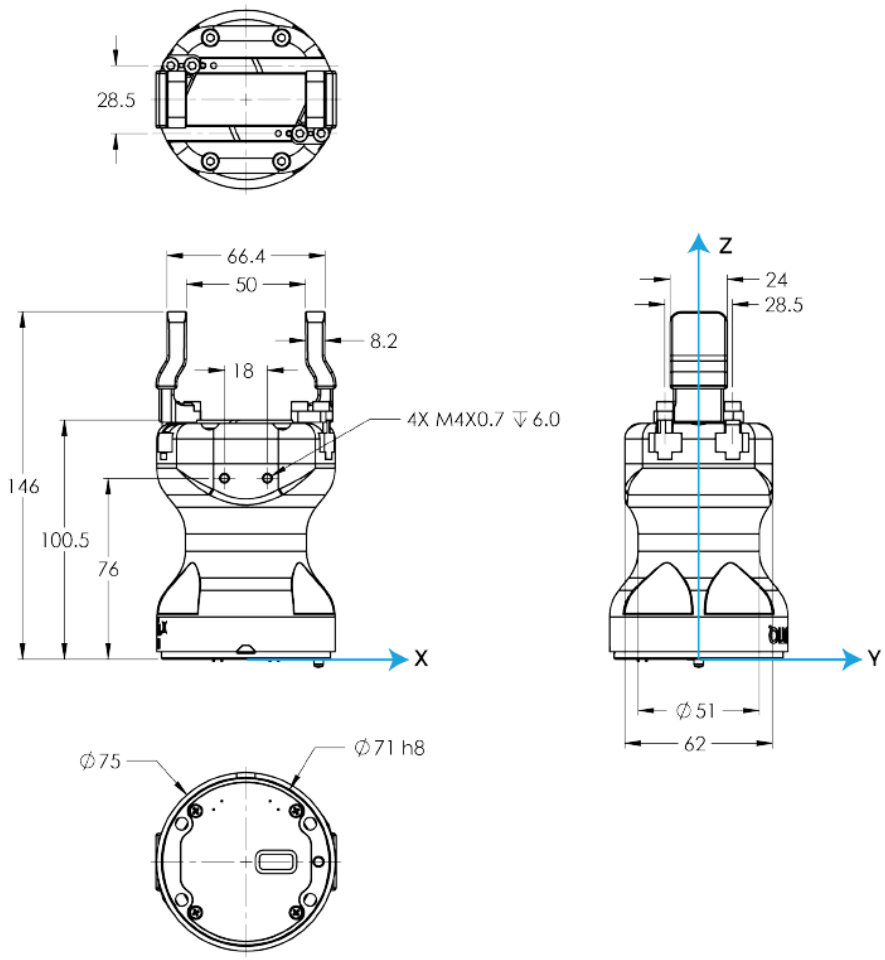

Figura 5. Pinza Hand-E. Dimensiones generales

7.1.3. Máquina virtual de PolyScope

Para la programación y configuración de sus cobots, Universal Robots proporciona una interfaz gráfica de usuario que permite el desarrollo de todo tipo de aplicaciones de manera sencilla e intuitiva y otorgando una gran adaptabilidad y flexibilidad llamada PolyScope.

PolyScope está basado en el sistema operativo Linux y en la página web del fabricante, https://www.universal-robots.com/download/?query=, se pueden encontrar imágenes de discos de máquinas virtuales, que permiten la programación y la simulación offline de sus robots colaborativos industriales mediante el software URSim.

URSim, como cualquier otro software de simulación, presenta algunas limitaciones con respecto a la versión de PolyScope que se encuentra en la consola de programación del robot real.

Estas limitaciones afectan principalmente al control de la fuerza del UR5e y se deben a que no existe una conexión con un brazo robótico real.

Para ejecutar el simulador offline es necesaria la creación de una máquina virtual. En el sitio web del fabricante se pueden encontrar una gran cantidad de máquinas atendiendo tanto a la

versión del software instalado en ellas como al sistema operativo del anfitrión. Para la realización del proyecto se ha seleccionado la siguiente versión:

- OFFLINE SIMULATOR – E-SERIES – UR SIM FOR NON LINUX 5.10.2

Para la instalación y ejecución de la máquina virtual, y atendiendo a la documentación proporcionada por Universal Robots de configuración de un entorno virtual [8], se requiere de un virtualizador. El virtualizador elegido para el presente es Oracle VM VirtualBox, desarrollado por Oracle Corporation, en su versión 6.1.22 r144080.

7.1.3.1. Creación de la máquina virtual en VirtualBox

Para la creación y configuración de la máquina virtual en VirtualBox únicamente hay que seguir una serie de sencillos pasos que se detallan a continuación.

En primer lugar, se crea una nueva máquina virtual en VirtualBox tal y como se muestra en la figura 6.

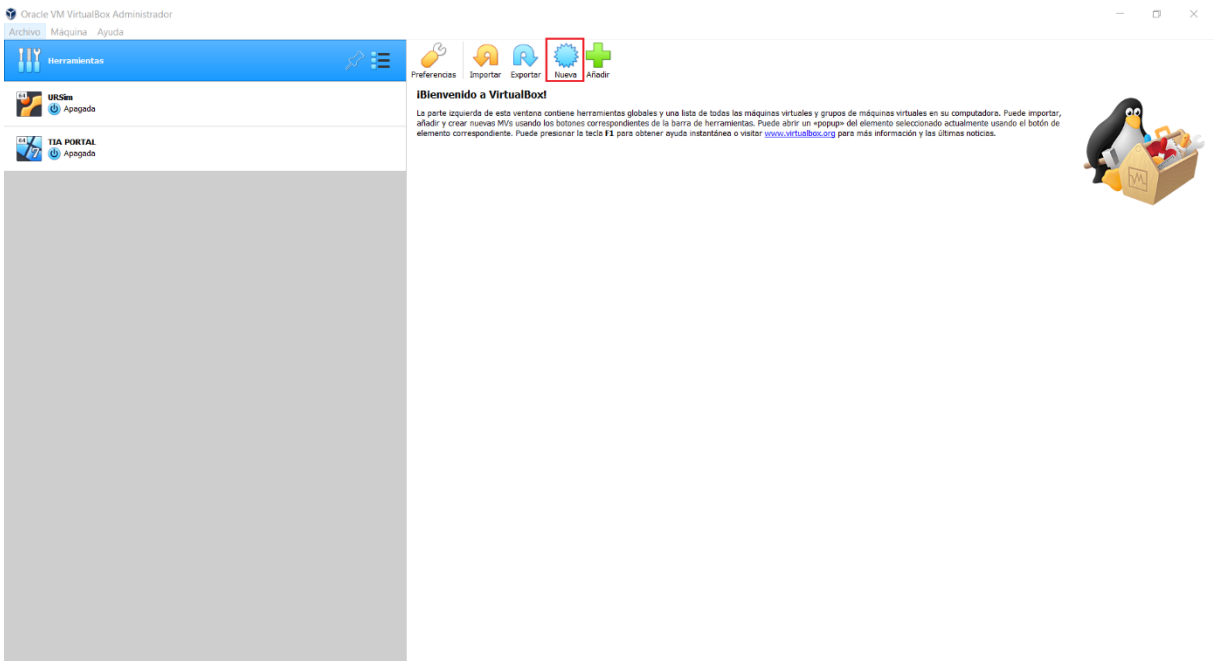

Figura 6. URSim - Máquina virtual. Nueva máquina virtual

A continuación, se ha de configurar la máquina. Para ello se elige y se le otorga un nombre y se selecciona la carpeta donde se ubicará dicha máquina en nuestro ordenador anfitrión y el sistema operativo y versión de la virtualización (Linux/Ubuntu 64-bit), como se muestra en la figura 7.

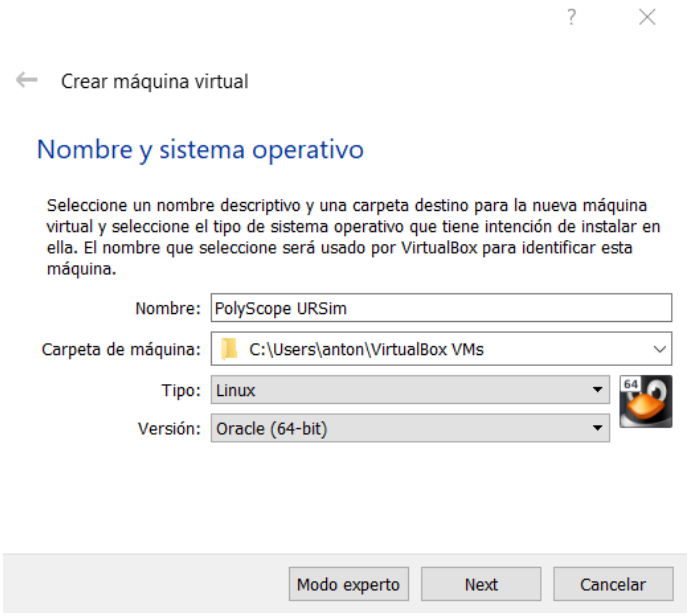

Figura 7. URSim - Máquina virtual. Nombre y sistema operativo

Para continuar con la creación, se ha de seleccionar el tamaño de memoria RAM en megabytes que se quiere reservar para la máquina virtual. Universal Robots recomienda que el tamaño de esta memoria sea de al menos 768 MB, por lo que se decide seleccionar un tamaño de 1024 MB como se puede ver en la figura 8.

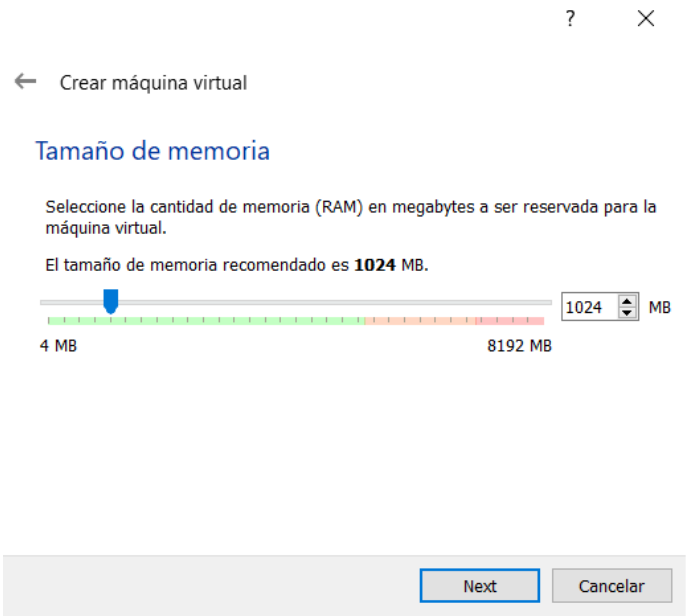

Figura 8. URSim - Máquina virtual. Tamaño de memoria

En el siguiente paso se selecciona la opción “Usar un archivo de disco duro virtual existente” para generar la máquina a partir del archivo descargado desde la página web de Universal Robots, al que se hace mención en el comienzo del presente apartado y seleccionar el siguiente archivo.

- URSim_VIRTUAL-5.10.2.106319.vmdk

A continuación, en la figura 9, se muestra el paso anteriormente descrito.

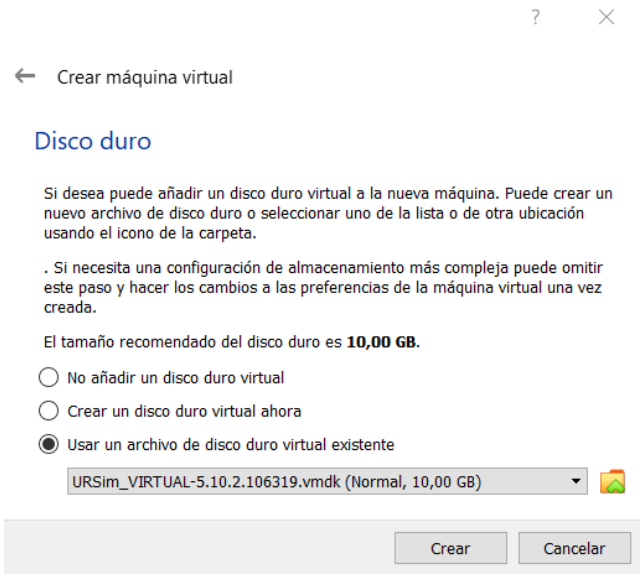

Figura 9. URSim - Máquina virtual. Disco duro

De esta forma se habrá finalizado la creación de la máquina virtual y esta aparecerá en el menú principal como se puede apreciar en la figura 10.

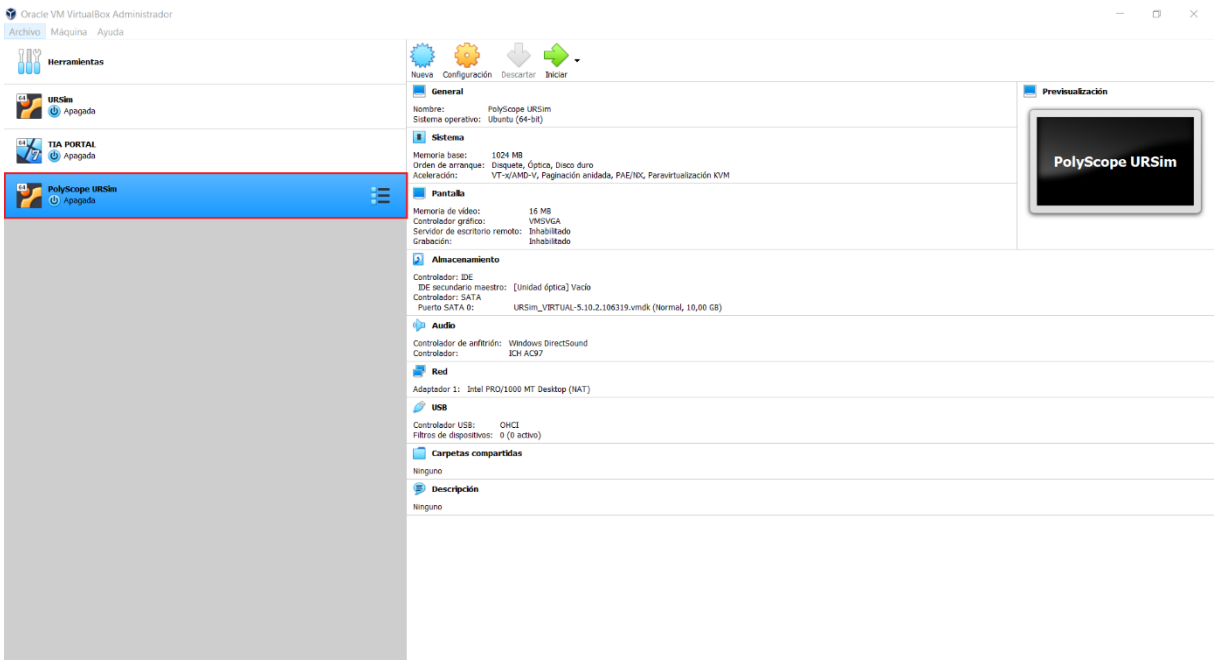

Figura 10. URSim - Máquina virtual. Máquina creada

7.1.3.2. Configuración de la máquina virtual en VirtualBox

Antes de iniciar la máquina virtual es recomendable hacer algunos cambios en la configuración de esta tal y como se muestra en las figuras 11, 12 y 13.

Desde el menú de “Configuración” se han de realizar las siguientes modificaciones con respecto a la pantalla de la máquina virtual:

- Se aumenta la memoria de vídeo a 128MB, lo que permitirá que la máquina vaya mucho más fluida a la hora de poder ver la simulación de los movimientos del cobot.
- Se selecciona el controlador gráfico VBoxSVGA. Hacer este cambio es importante puesto que sino cuando se inicie la máquina virtual puede funcionar de forma incorrecta, por lo que se debe ignorar el mensaje de “Configuración inválida detectada” que aparece en el pie de la ventana de configuración. Esto no afectará al rendimiento ni correcto funcionamiento de la máquina.

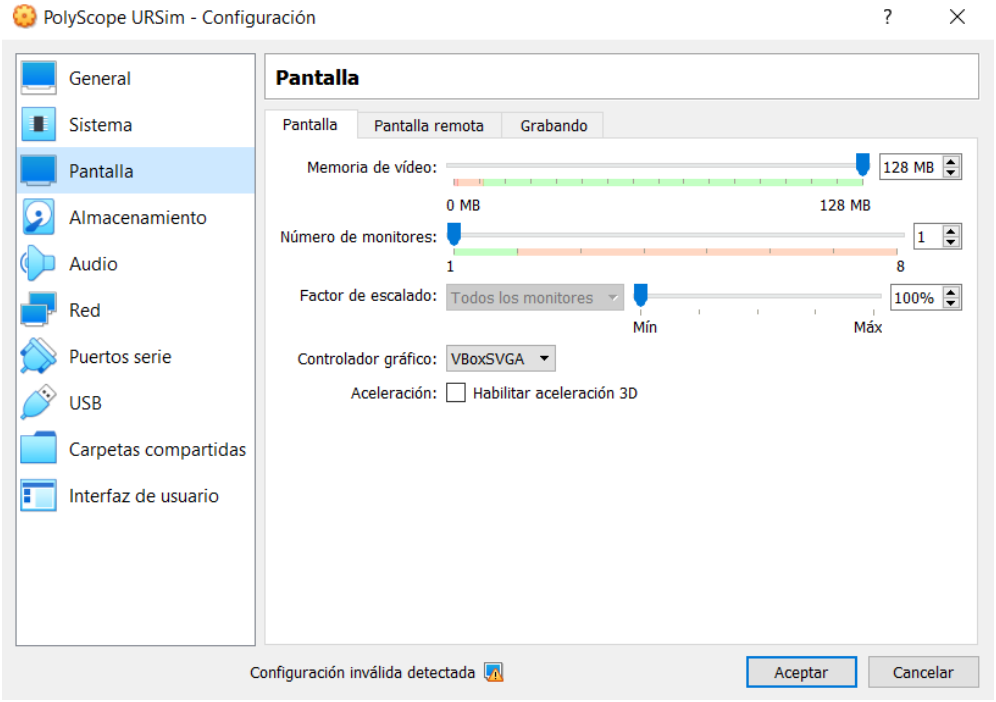

Figura 11. URSim - Máquina virtual. Configuración de pantalla

Ahora que se han realizado los cambios oportunos en el apartado “Pantalla”, desde el apartado “Almacenamiento” se selecciona como IDE secundario maestro el archivo de disco “VBoxGuestAdditions.iso” como se muestra a continuación:

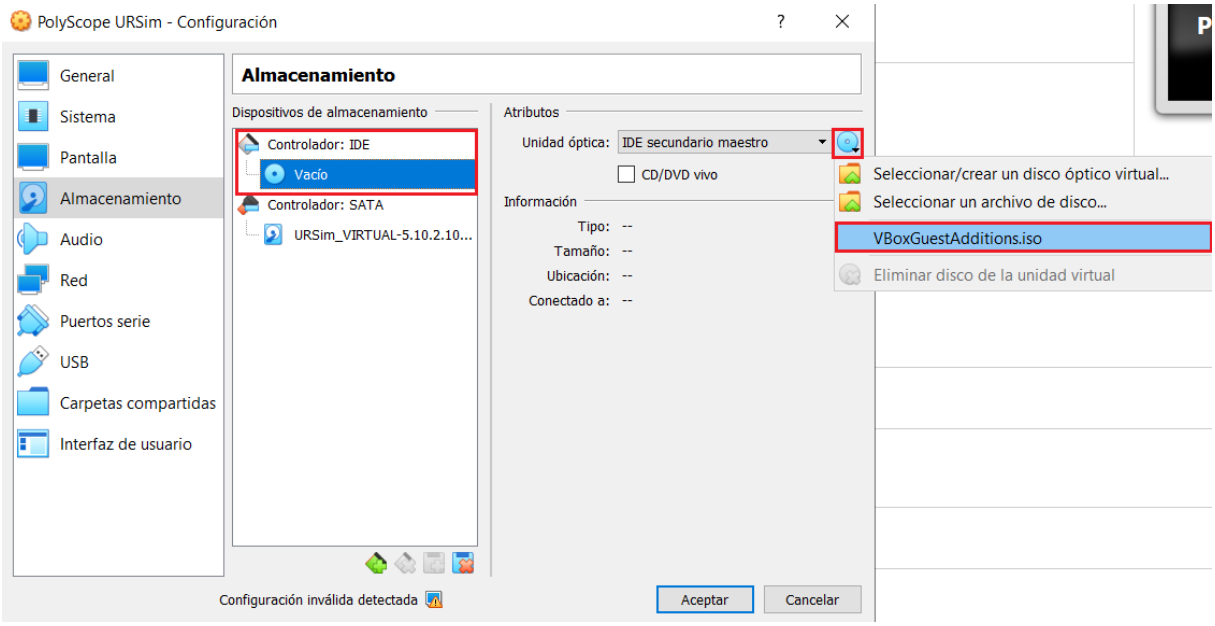

Figura 12. URSim - Máquina virtual. Almacenamiento.

Por último, en el apartado "Sistema", en la pestaña "Procesador" se deselecciona la opción "Habilitar PAE/NX".

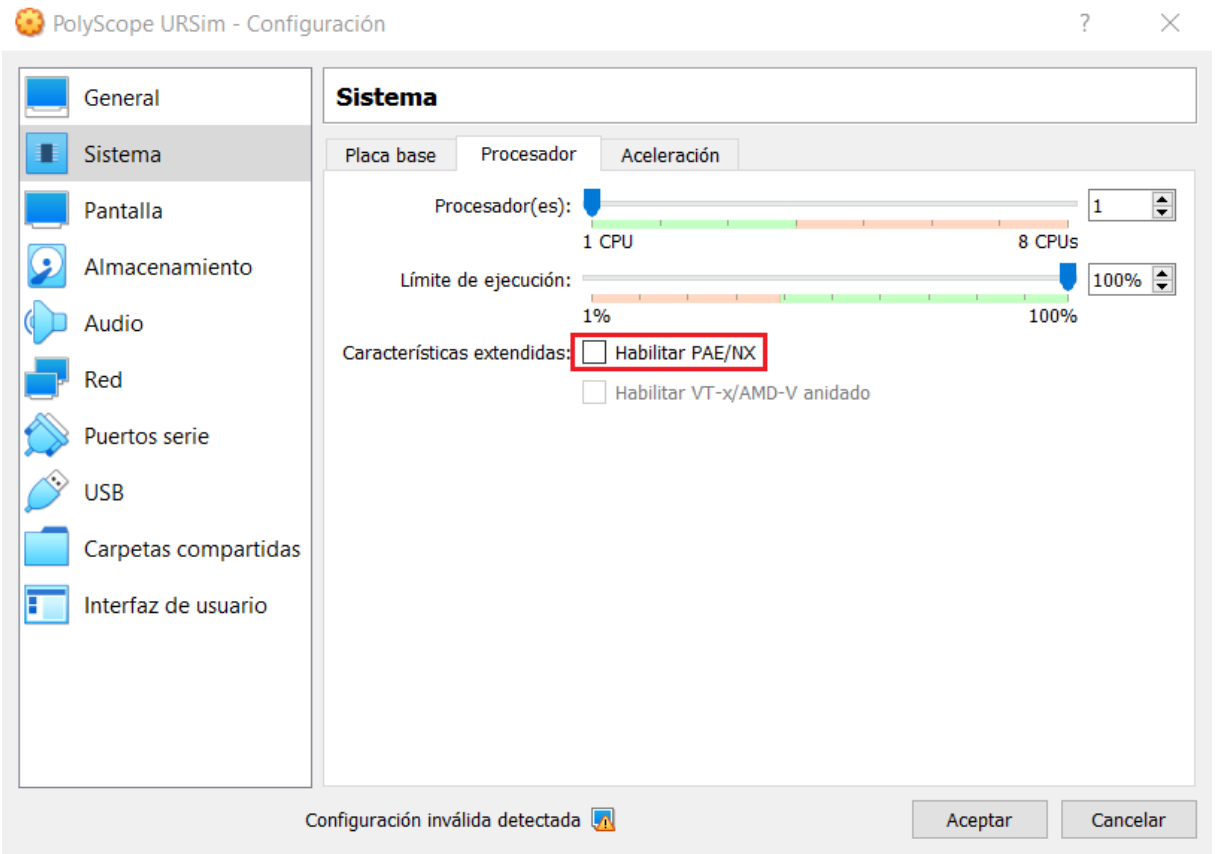

Figura 13. URSim - Máquina virtual. Sistema

Una vez que se ha terminado de configurar la máquina virtual ya se puede iniciar.

Cuando se tiene la máquina corriendo, en el escritorio, se pueden distinguir una serie de archivos y carpetas de documentos correspondientes a los simuladores de los distintos modelos de robots colaborativos que posee Universal Robots, UR3e, UR5e, UR10e y UR16e, tal y como se visualiza en la figura 14.

Para el desarrollo de este proyecto se hará uso del ejecutable "URSim UR5", el cual se corresponde con el cobot UR5e y la carpeta "ProgramsUR5" donde se ubicarán los distintos programas o aplicaciones necesarias para la ejecución del presente.

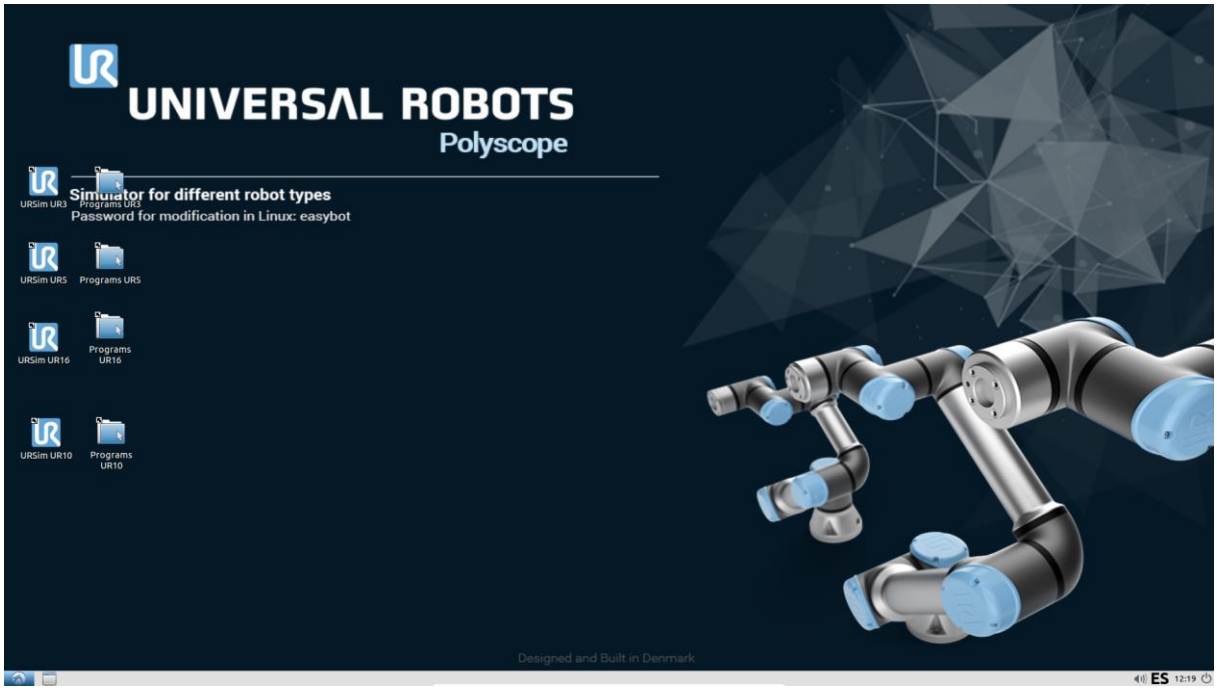

Figura 14. URSim - Máquina virtual. Escritorio

7.1.4. URSim UR5

7.1.4.1. Configuración general

Para realizar la configuración del proyecto de URSim, se atiende a la información descrita y detallada en el manual de usuario del UR5e [6].

Una vez arrancado se confirman los parámetros de configuración de seguridad, que se muestran en la figura 15, para poder empezar con la configuración del proyecto y la programación de las aplicaciones deseadas.

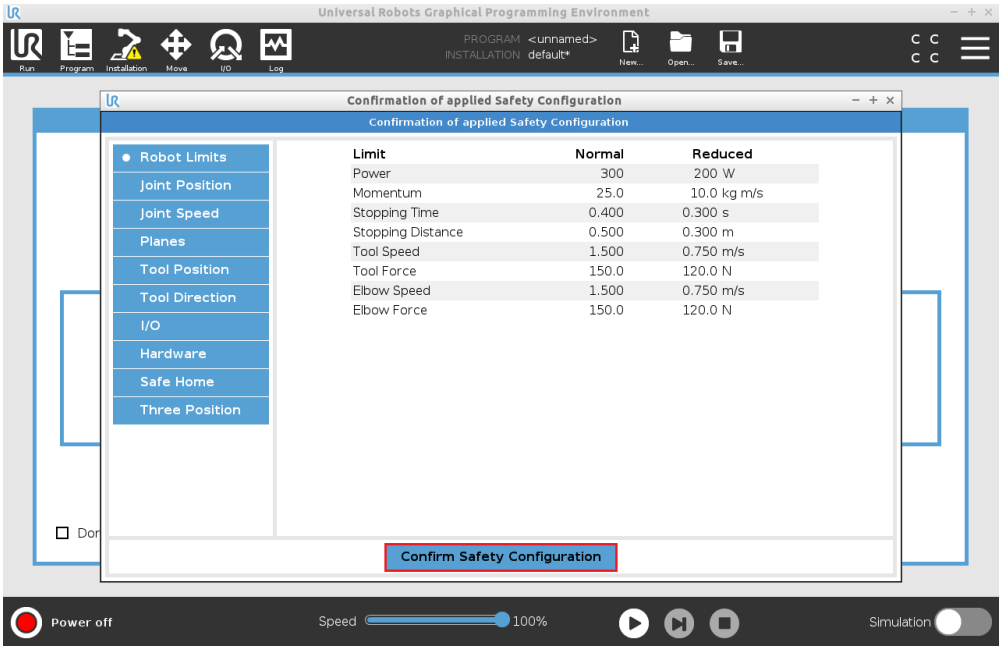

Figura 15. URSim UR5. Parámetros de seguridad

Una vez confirmada la configuración de seguridad, se decide cambiar el idioma de la interfaz de usuario a través del menú de “Ajustes” para hacer más llevadera la experiencia de usuario a través de los pasos que se muestran en las figuras 16 y 17.

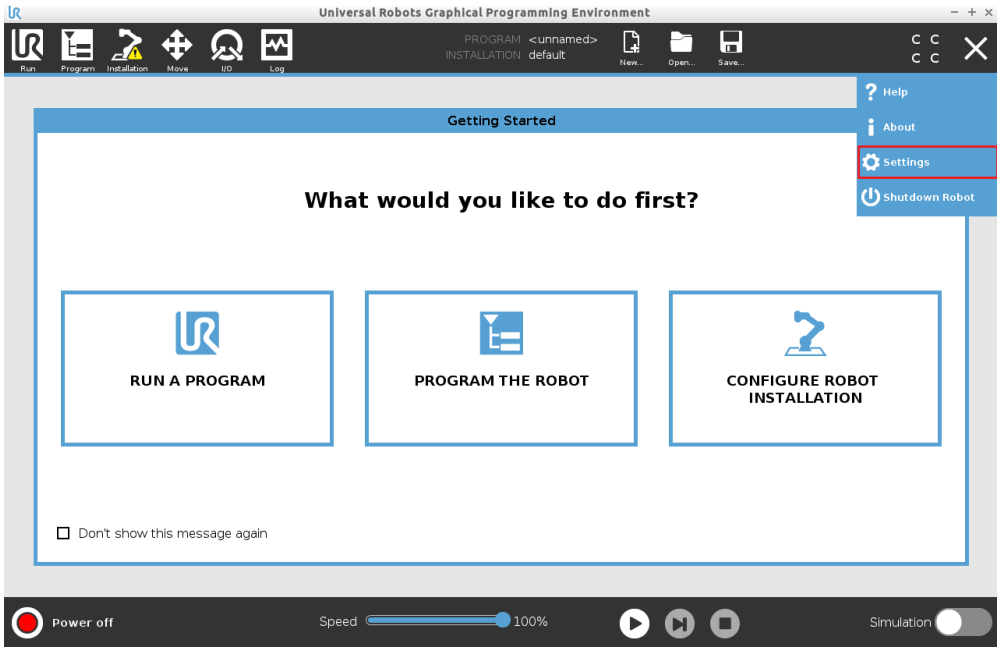

Figura 16. URSim UR5. Ajustes

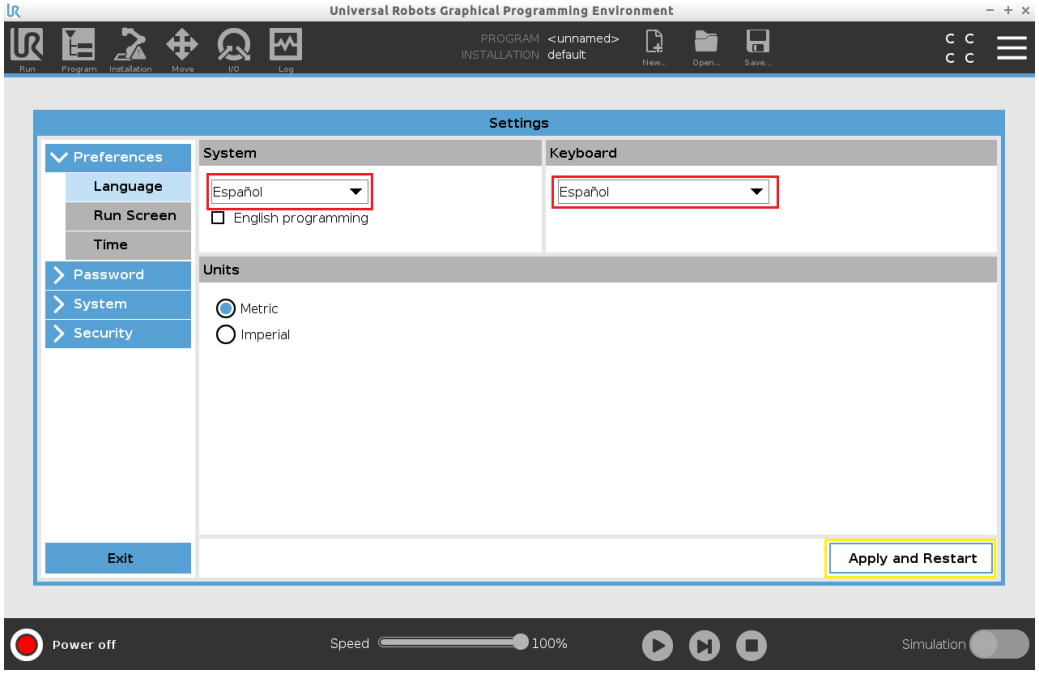

Figura 17. URSim UR5. Selección de idioma

7.1.4.2. Configuración de la instalación del robot

Una vez realizados los cambios de configuración general de la interfaz, se procede a configurar la instalación del robot a utilizar en el desarrollo del proyecto tal y como se ve en la figura 18.

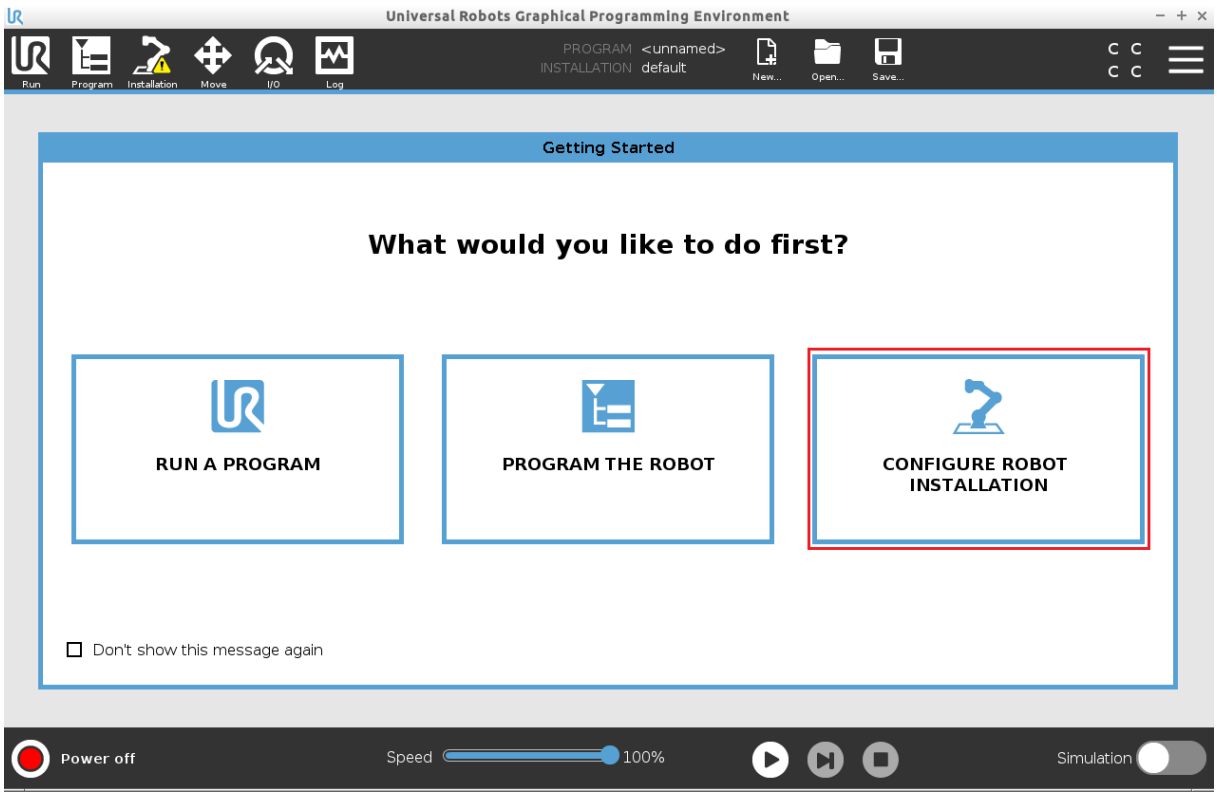

Figura 18. URSim UR5. Configuración de la instalación

En primer lugar, se procederá a la configuración del punto central de la herramienta o PCH. Para la ejecución del presente proyecto se ha seleccionado el gripper adaptable de Robotiq Hand-E y a la cual se hace referencia en el apartado 7.1.2 de este mismo documento.

Basándose en las especificaciones de Robotiq [7] y como se puede observar en la figura 3 de este documento, se llega a la conclusión de que el PCH deberá situarse a 146 mm de Z = 0, estando este en el centro de la brida de la herramienta del robot tal y como se muestra en las figuras 19 y 20.

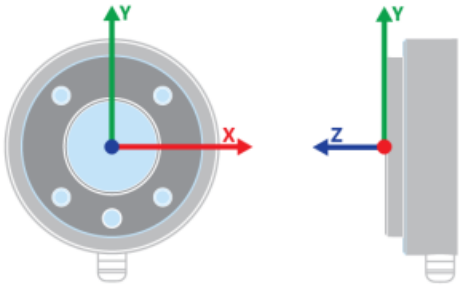

Figura 19. URSim UR5. Brida de la herramienta

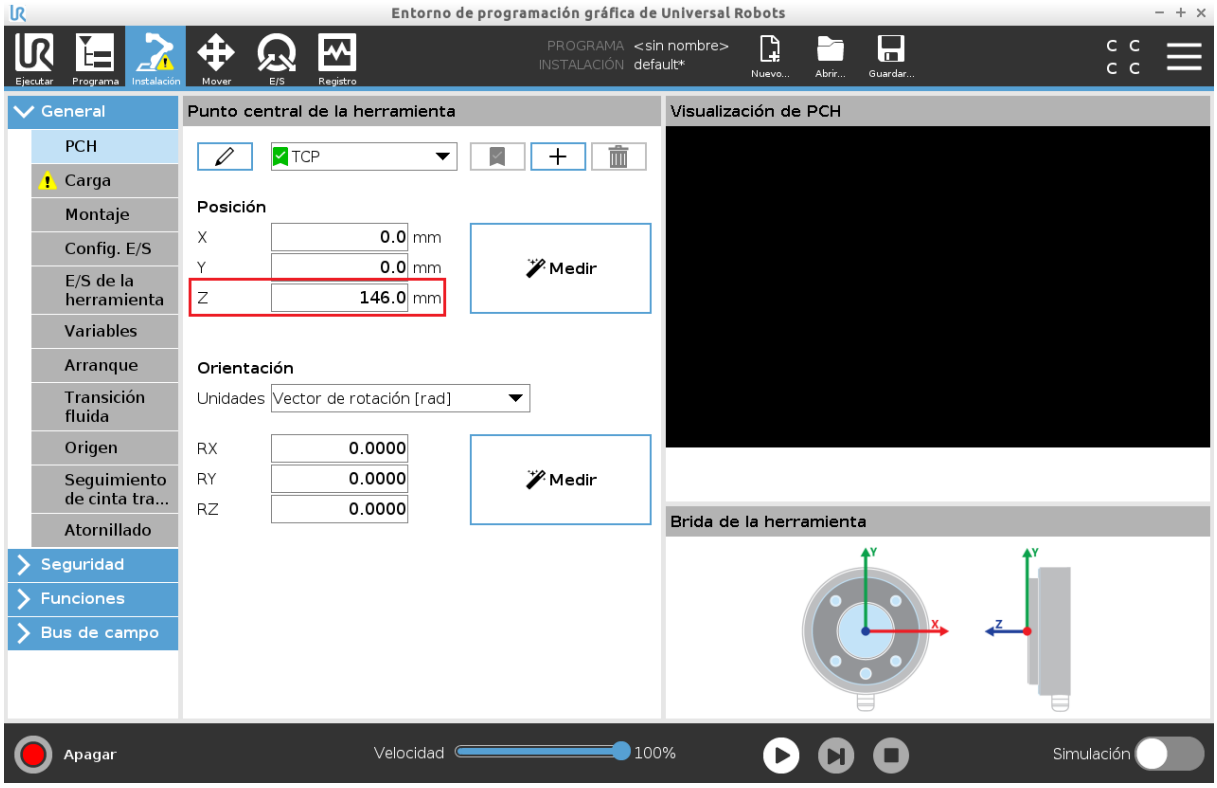

Figura 20. URSim UR5. Punto central de la herramienta

Posteriormente se procede a configurar la carga útil tal y como se muestra en la figura 21. Para ello, atendiendo también a las especificaciones de Robotiq, se considera que la carga útil es de 1,5kg. Esto es:

- Pieza para reubicar: 0,5 kg aproximadamente
- Pinza Hand-E: 1 kg

- Carga útil total: 1,5 kg

Además, al dejar sin configurar el centro de gravedad de la carga, este se considera que está ubicado en el PCH.

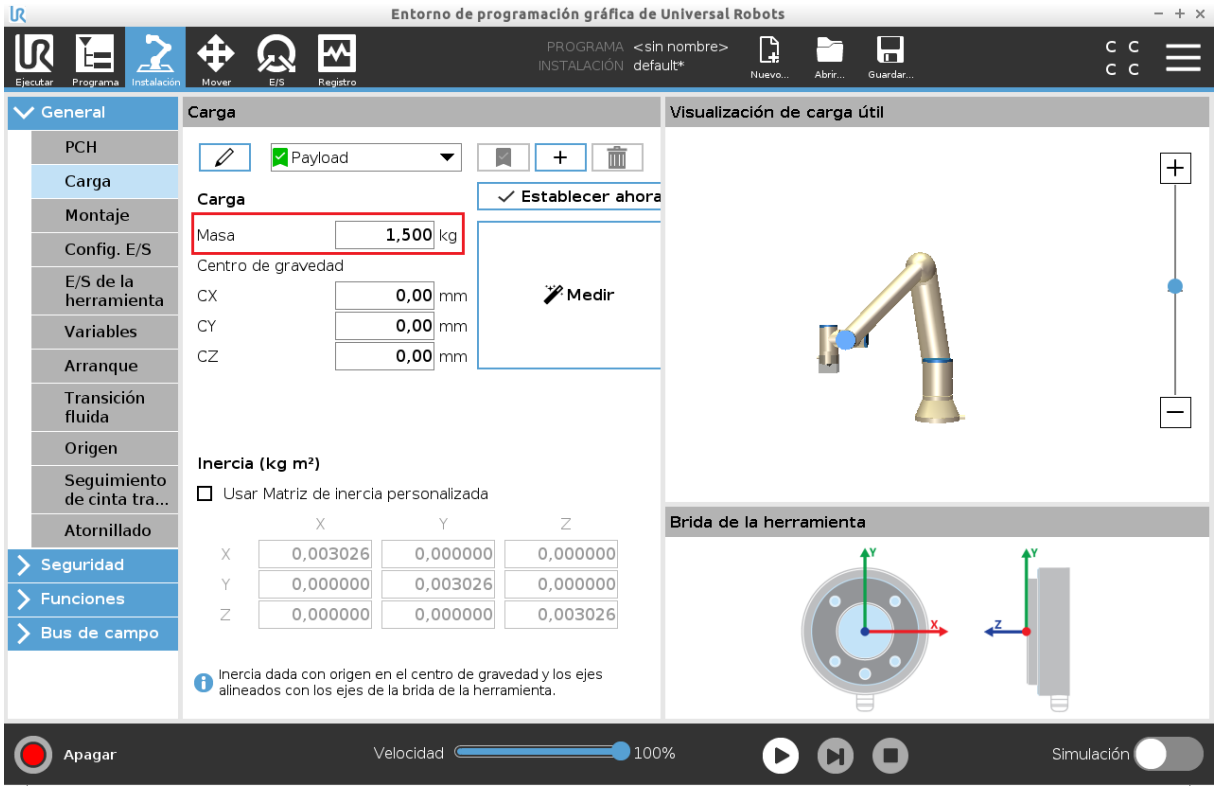

Figura 21. URSim UR5. Carga útil

Como se puede apreciar en la figura 22, se puede configurar el tipo de instalación que se quiere simular:

- Montaje sobre suelo
- Montaje sobre pared
- Montaje sobre techo

Para el presente proyecto, se decide escoger un montaje sobre suelo dado que la aplicación simulada consiste en coger y dejar piezas en cintas transportadoras horizontales y se considera que es la forma más eficiente para la programación y ejecución de esta.

Además, también pueden configurarse tanto la inclinación del brazo robot como la orientación de la fijación de la base de este. Para la realización del presente estos valores tomarán el valor de 0º.

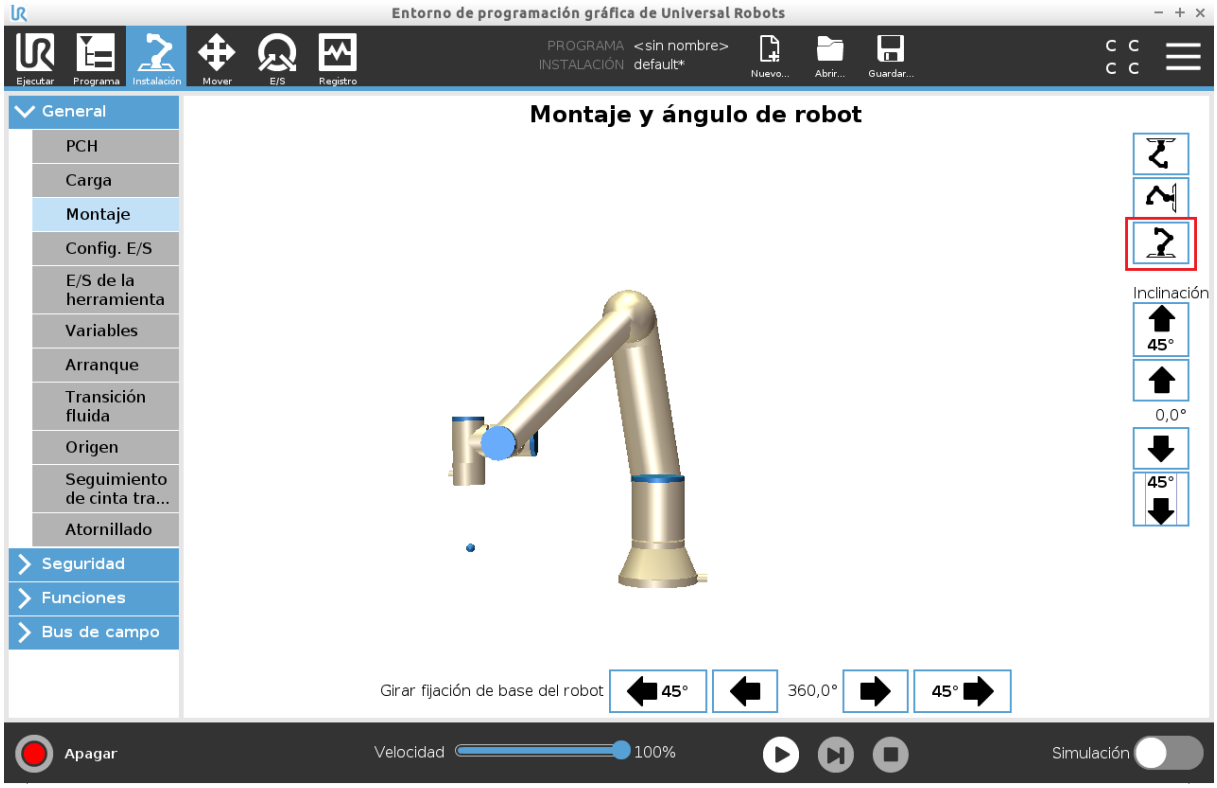

Figura 22. URSim UR5. Montaje y ángulo de robot

Por último, en esta primera parte de configuración de la instalación, se modifica la posición de origen del robot. Para ello, al contrario que para el resto de las modificaciones que se han realizado hasta ahora, es necesario que el robot esté encendido.

El proceso de encendido del robot queda recogido visualmente en las figuras de la 23 a la 26 que se pueden ver en las siguientes páginas del documento.

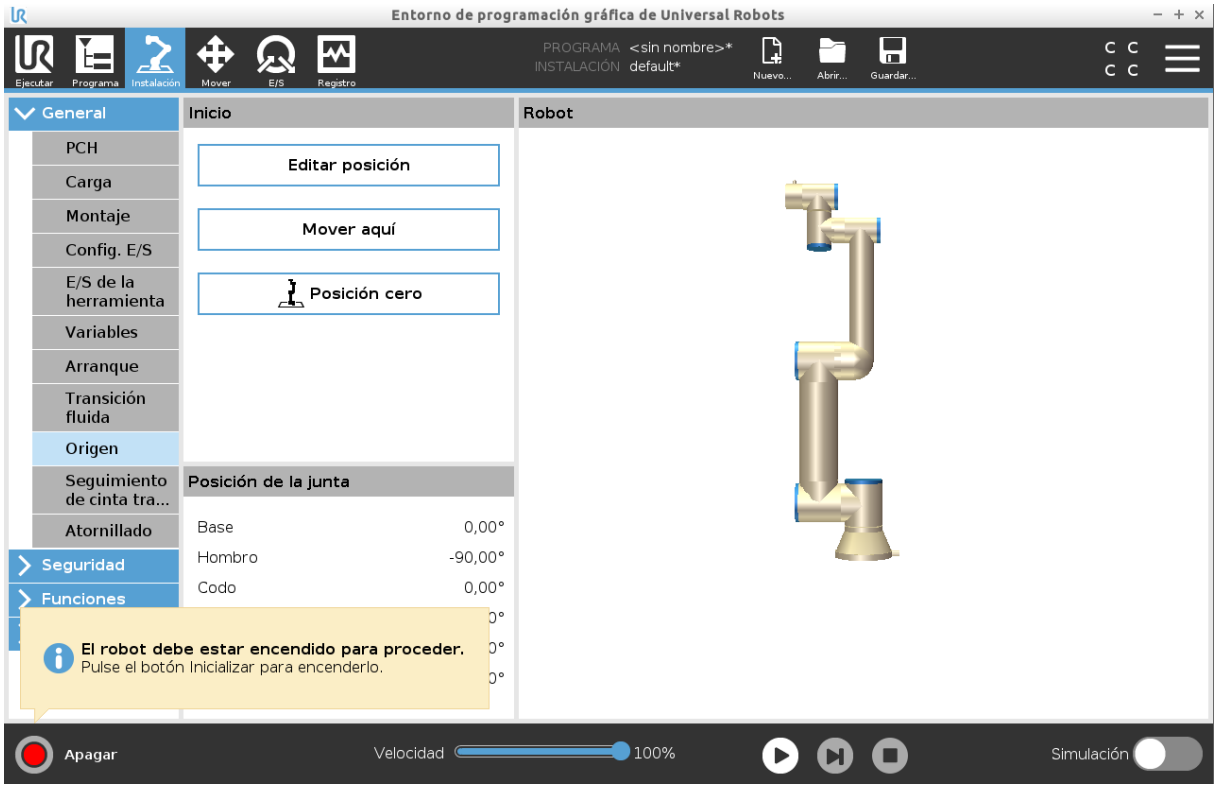

Figura 23. URSim UR5. Editar origen.

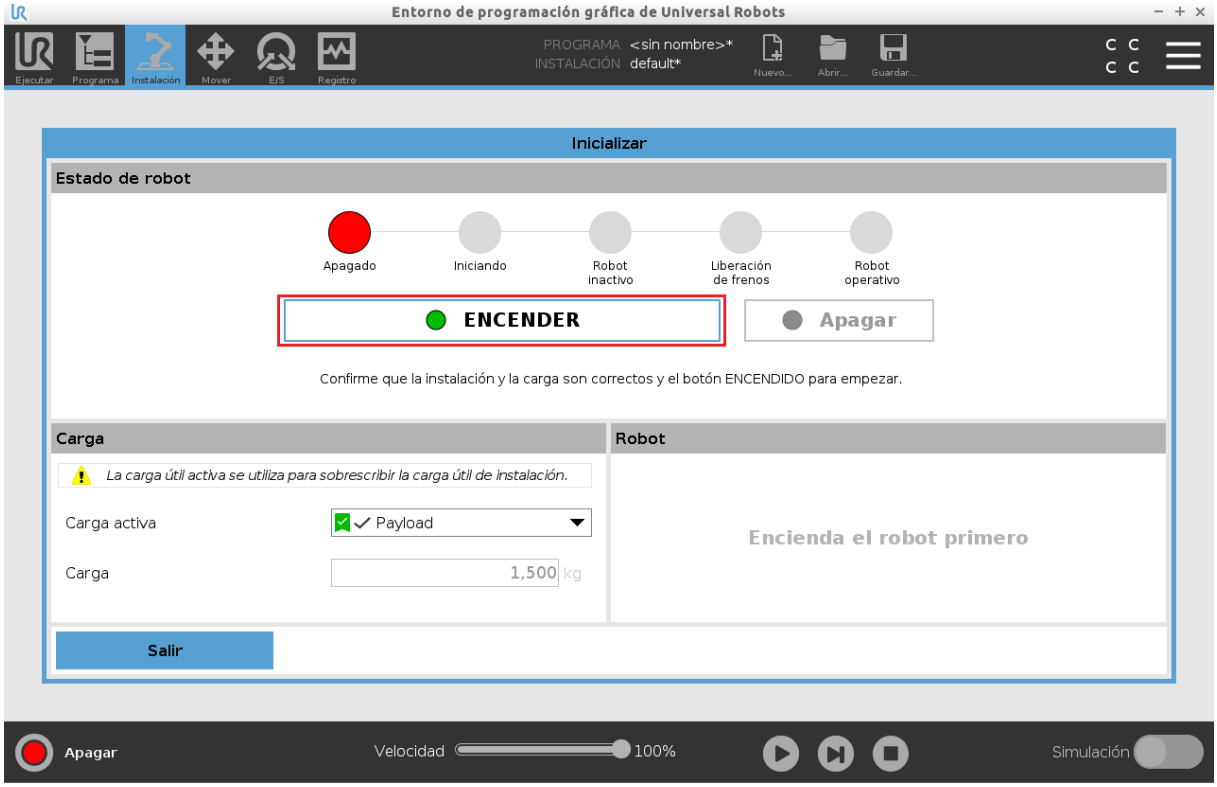

Figura 24. URSim UR5. Inicializar robot

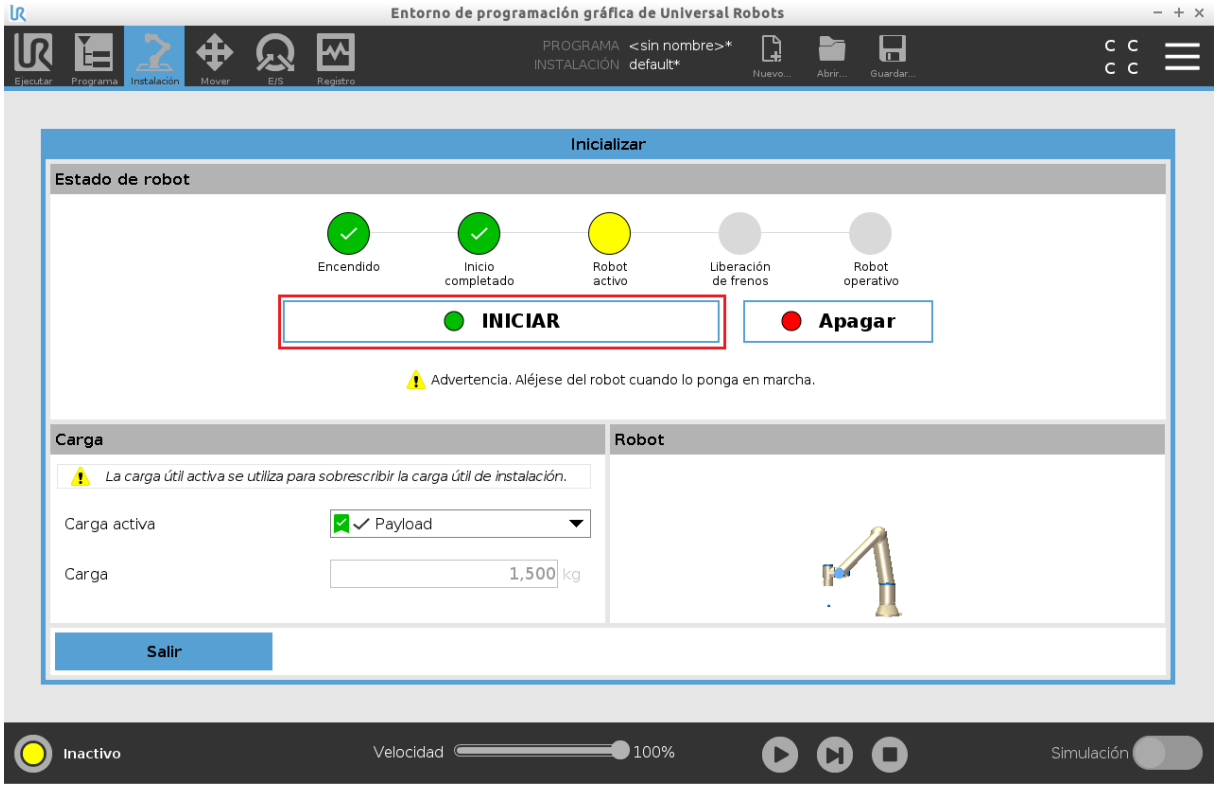

Figura 25. URSim UR5. Robot encendido

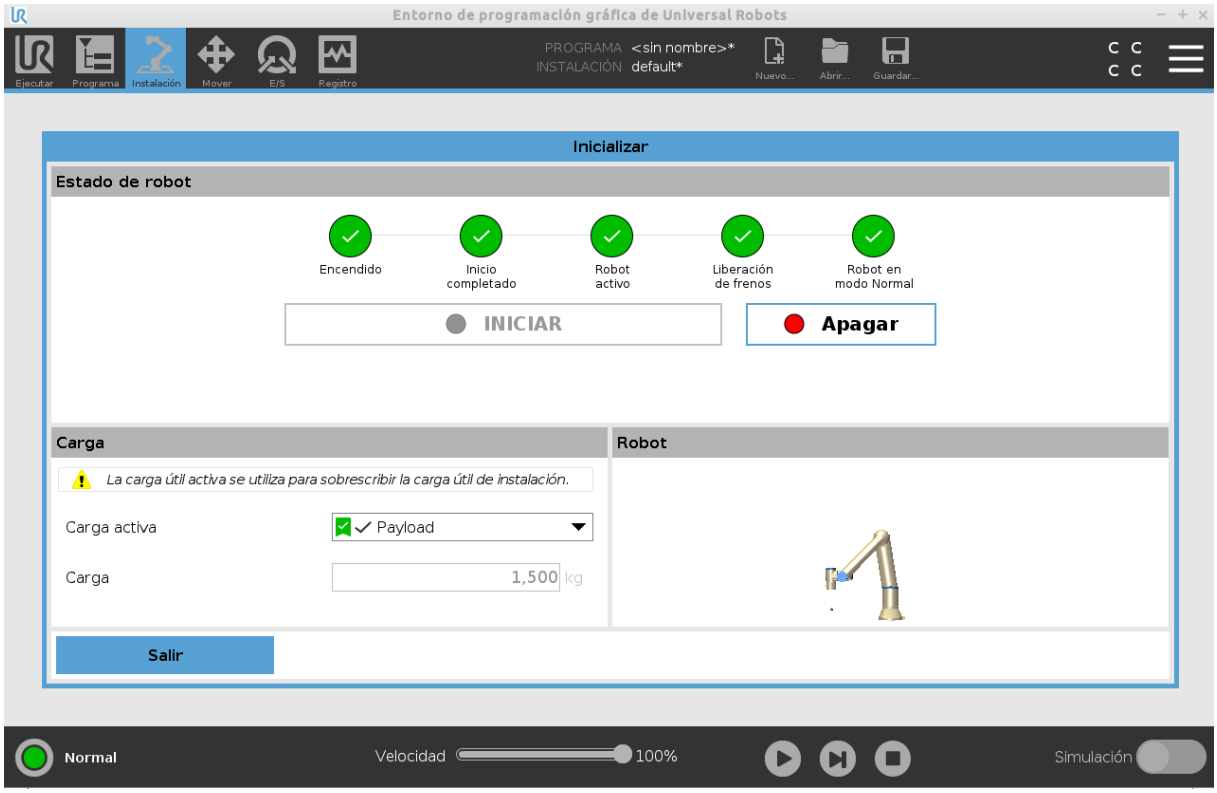

Figura 26. URSim UR5. Robot iniciado

Una vez que se tiene el robot iniciado, se procede a la edición de la posición del punto de origen del UR5e.

Para el posicionamiento en el origen se tendrá en cuenta los siguientes puntos:

- El plano de instalación de la base del robot es el mismo que el plano de las cintas transportadoras, es decir, tanto la base del robot como las cintas transportadoras se encuentran a la misma altura respecto del suelo instalados.
- El punto central de la base del robot se encuentra alineado con el punto de la cinta transportadora en la que se recogerán las piezas para su identificación y posterior clasificación.
- El extremo de la pinza se situará a una distancia de seguridad de 100 mm, justo por encima de la pieza a recoger, de manera que el primer movimiento de la aplicación sea siempre un desplazamiento única y exclusivamente en el eje Z.

Para editar la posición de origen, teniendo en cuenta que el TCP de la herramienta está situado en el punto (0; 0; 0) mm, la base del robot se debe ubicar en el punto (0; -500; 100) mm respecto del TCP de la herramienta. Este proceso se muestra en las figuras 27, 28, 29 y 30.

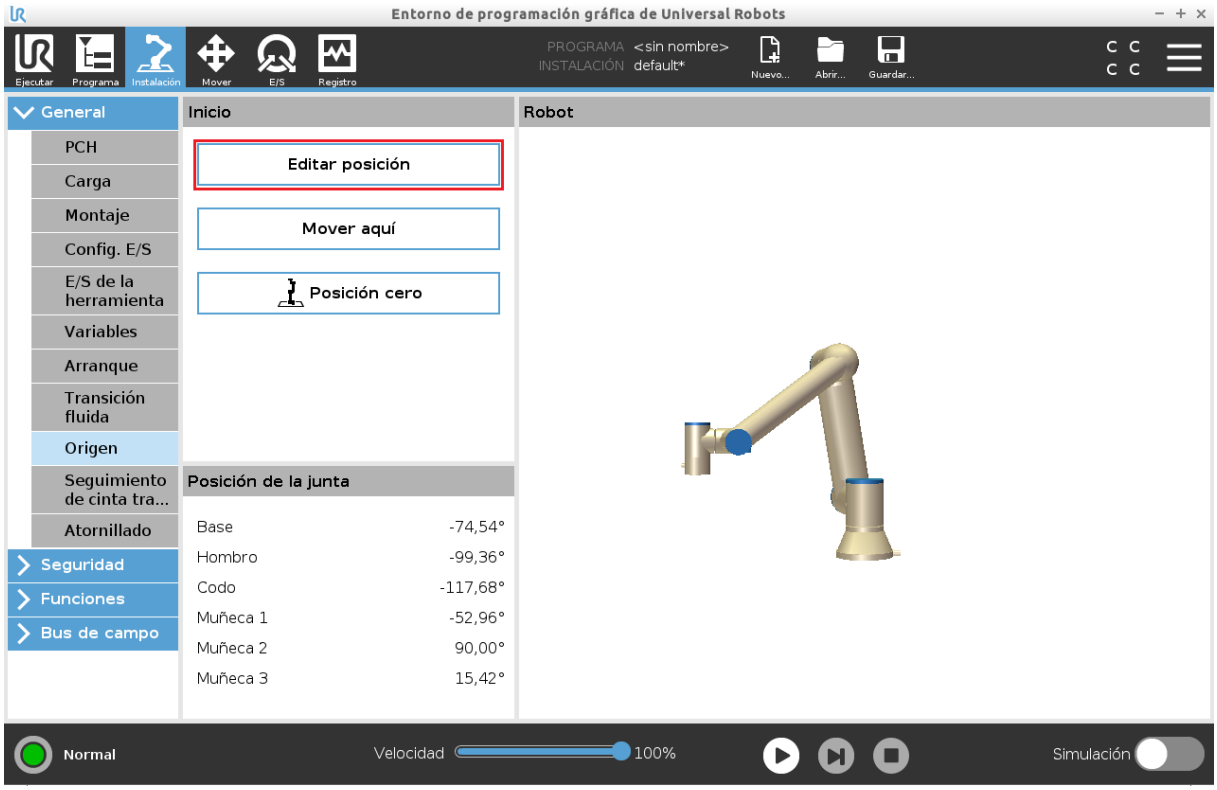

Figura 27. URSim UR5. Editar posición de inicio

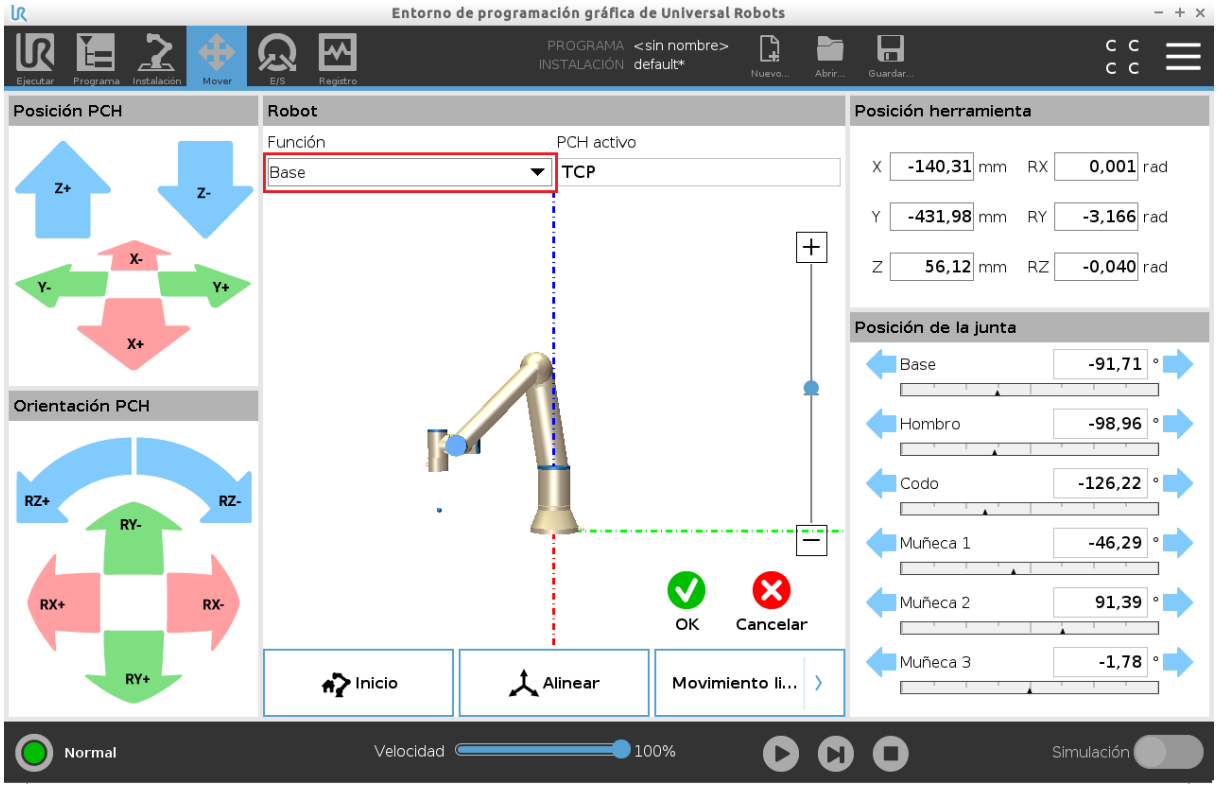

Figura 28. URSim UR5. Posición herramienta

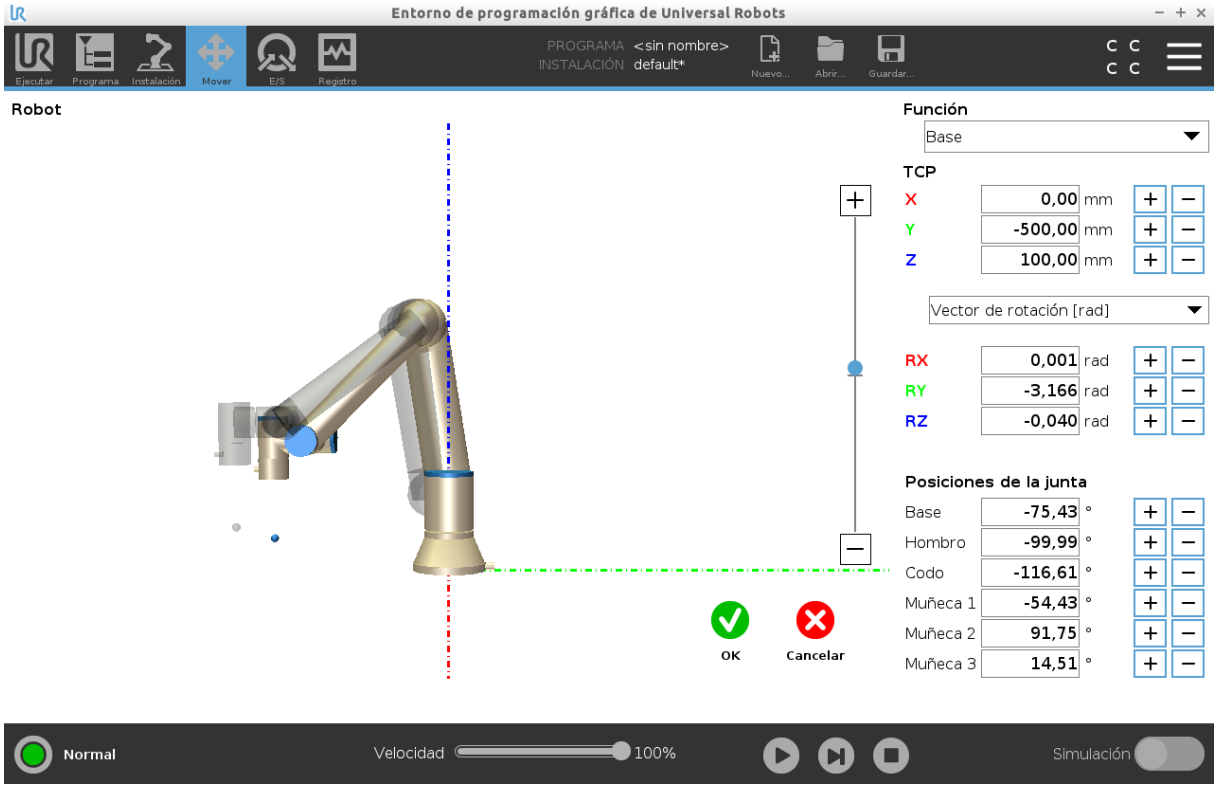

Figura 29. URSim UR5. Nuevo origen

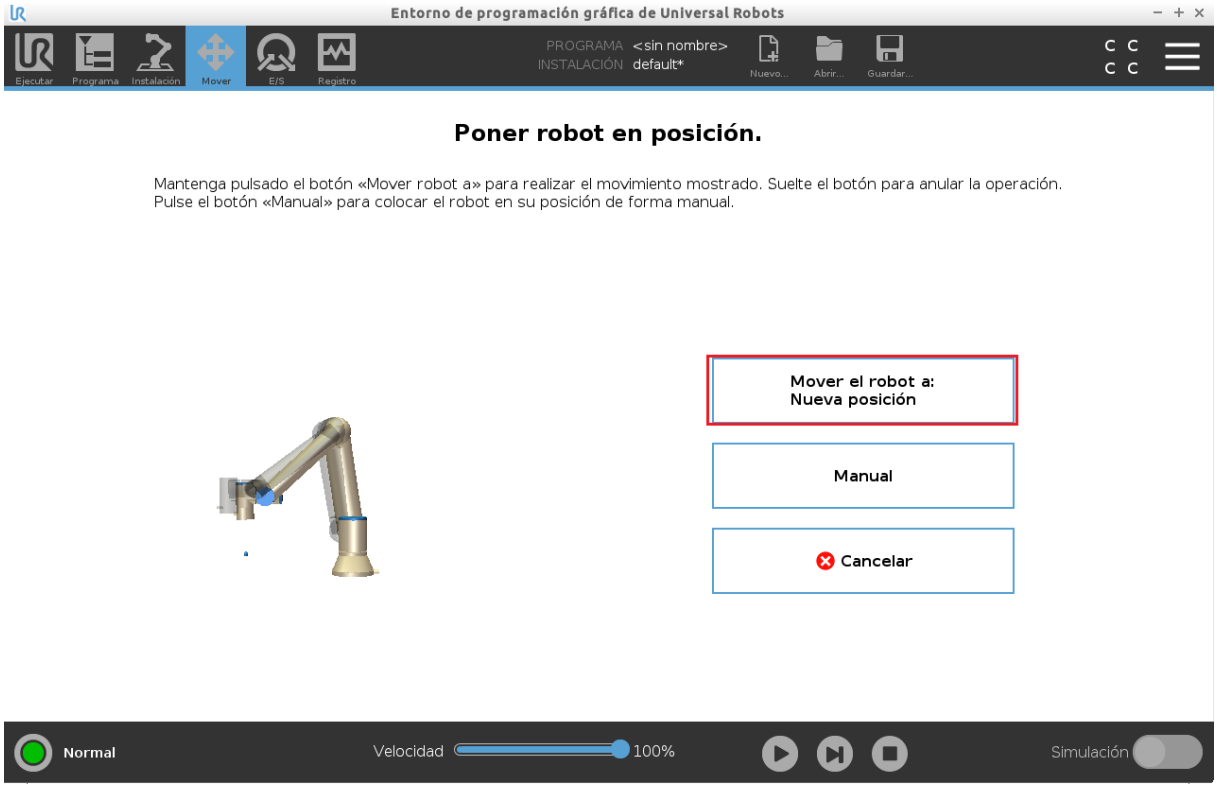

Figura 30. *URSim UR5. Poner robot en posición*

Por último, se alinea la herramienta con el eje Z y que de esta manera se encuentre perpendicular tanto al plano de la cinta transportadora, como al de la base del robot, Este proceso se muestra en las figuras 31 y 32.

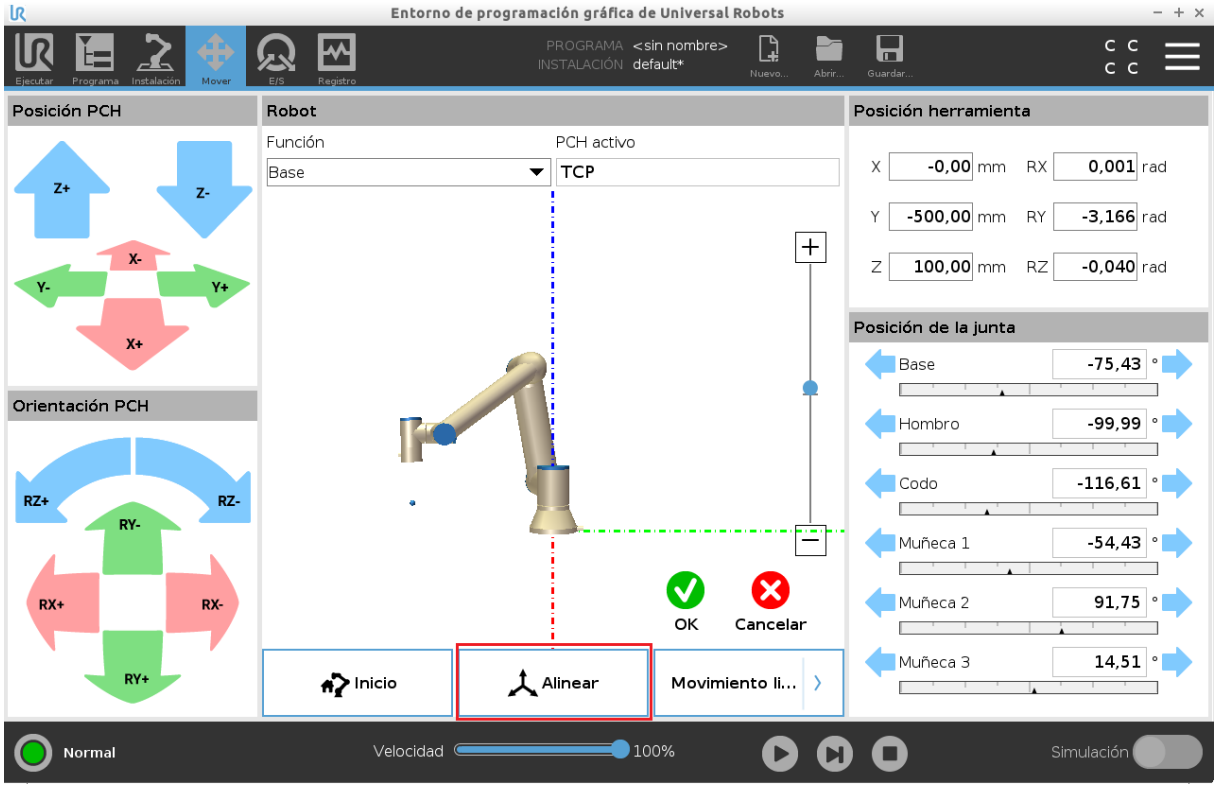

Figura 31. URSim UR5. Alinear herramienta, parte 1

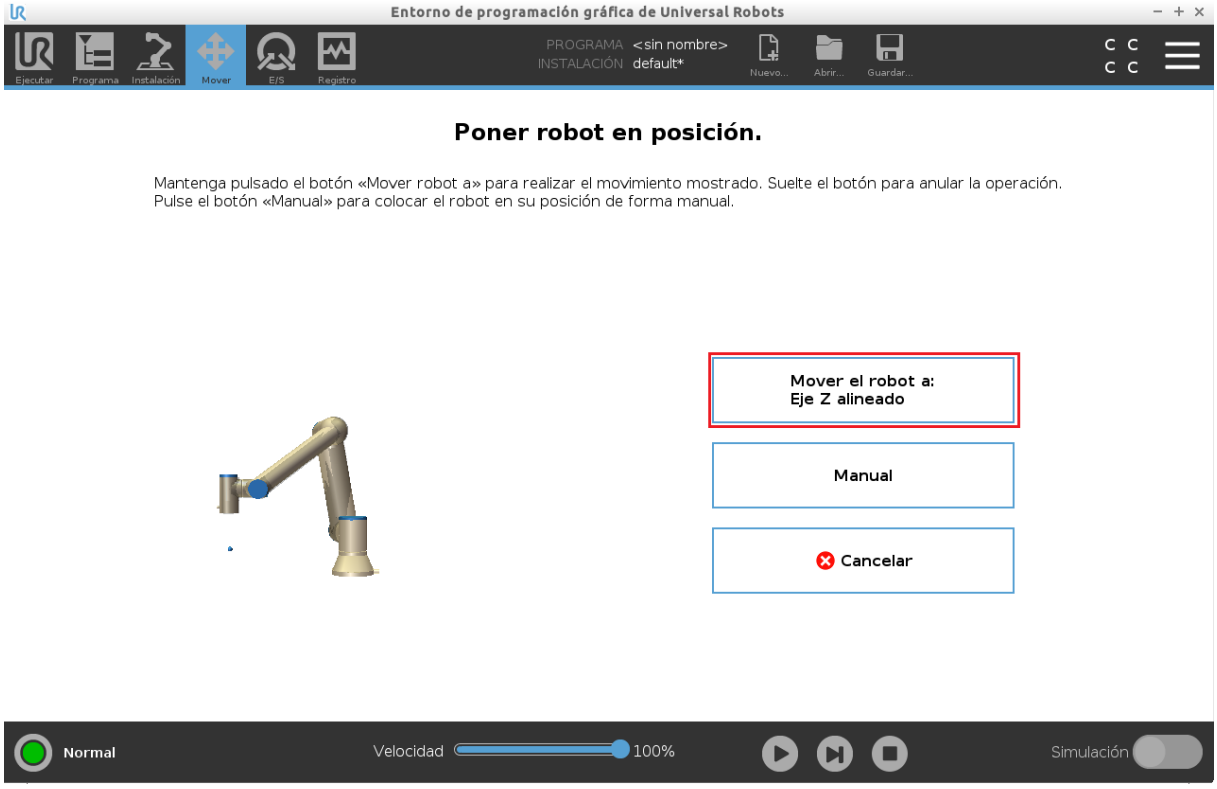

Figura 32. URSim UR5. Alinear herramienta, parte 2

7.1.4.3. Instalación de Hand-E en PolyScope

Antes de comenzar con el desarrollo de la aplicación, se realizará la instalación del software de la pinza Hand-E de cara al uso de los comandos y funciones de comunicación entre la pinza y controlador del robot. Para ello, desde el sitio web de Robotiq, se facilita la descarga del archivo .urcap para la instalación de dicho software, https://robotiq.com/support/hand-e-adaptive-robot-gripper **Universal Robots>Software>Gripper Software>Universal Robots URCap (UCG-1.8.7 for PolyScope 3.10+/5.4+)**.

Cómo el archivo se descarga desde el anfitrión, para poder traspasar el archivo a la máquina virtual se habilita una carpeta compartida "CarpetaCompartida" tal y como se muestra en la figura 33.

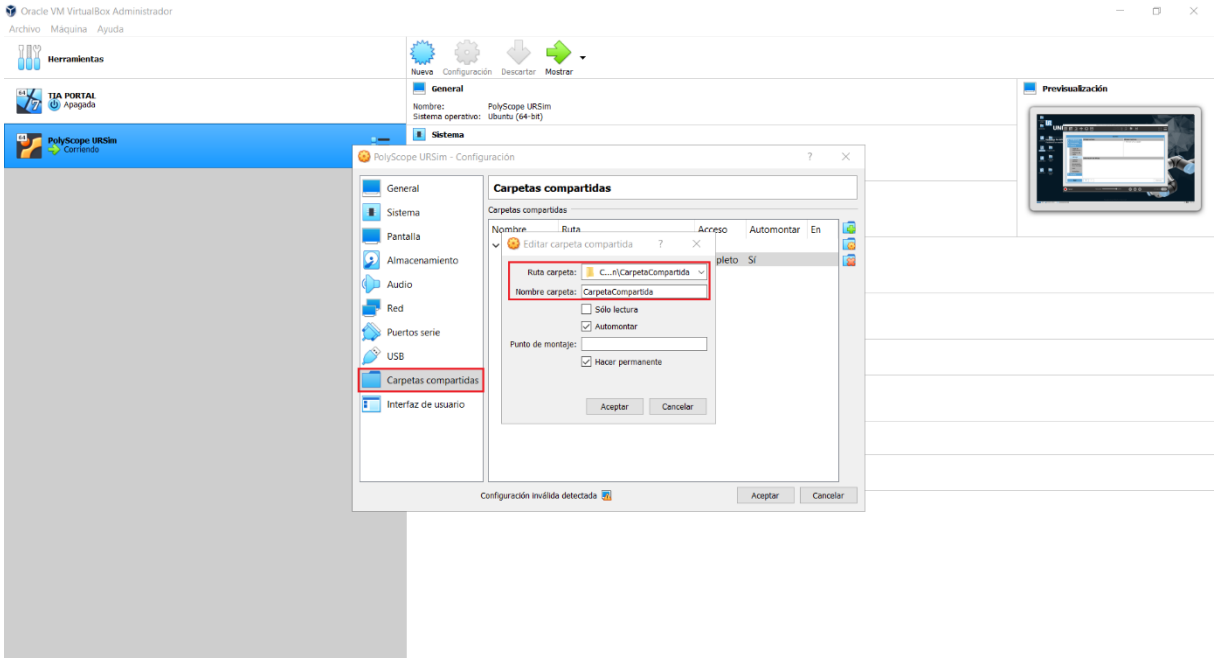

Figura 33. Máquina virtual. Carpeta compartida

En la máquina virtual se crea una carpeta "CarpComp" en el escritorio que será la que se comunique con la carpeta compartida que hemos creado previamente en el anfitrión, en este caso, "CarpetaCompartida". Por último, se abre un terminal en la máquina virtual de PolyScope y se ejecuta el siguiente comando:

- sudo mount -t vboxsf CarpetaCompartida /home/ur/Desktop/CarpComp

De esta forma, ya se tienen comunicadas ambas carpetas y se puede compartir el archivo "Robotiq_Grippers-1.8.7.10599.urcap" que se ha descargado de la web de Robotiq. Este archivo se tiene que mover a la carpeta "Programs UR5" que se encuentra en el escritorio y posteriormente, en la interfaz gráfica de PolyScope, se navega a **Ajustes>Sistema>URCaps** y se añade el archivo .urcap. Para terminar la instalación del software, es necesario reiniciar URSim. Este proceso queda recogido en las figuras 34 y 35 que se pueden ver a continuación.

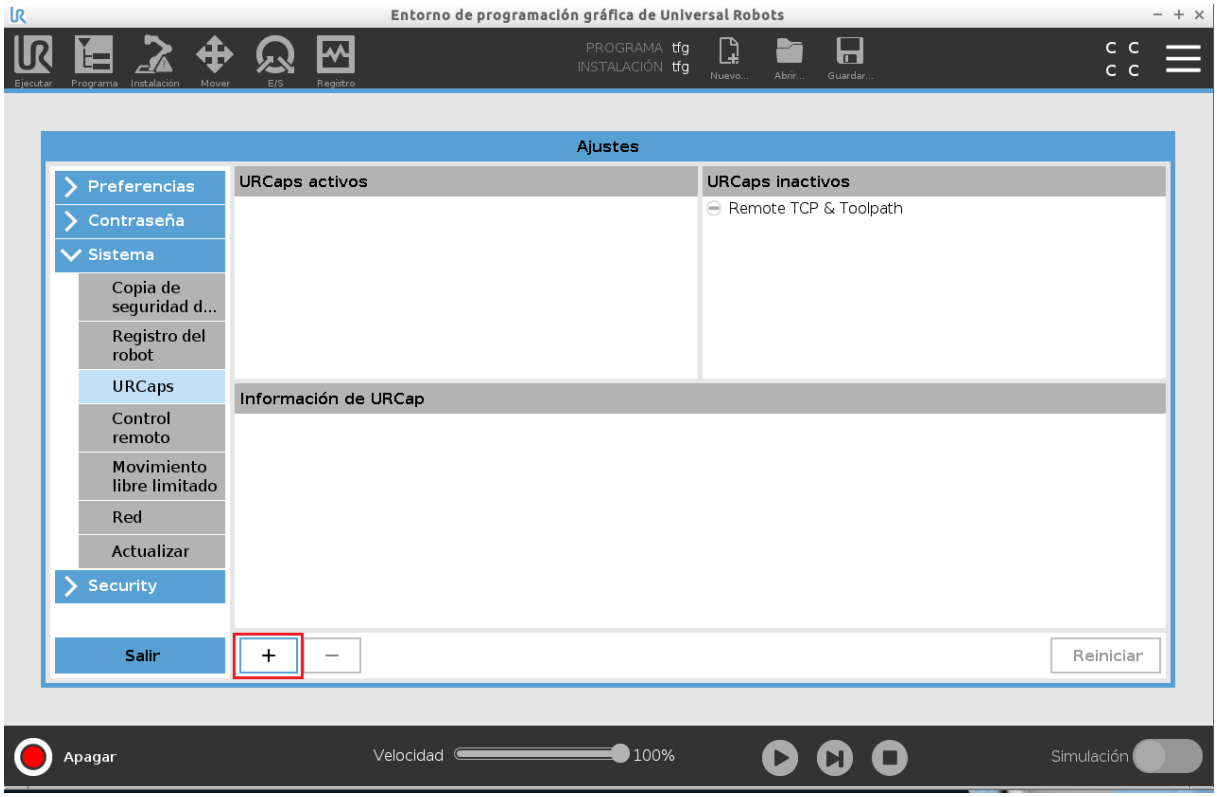

Figura 34. URSim UR5. URCaps

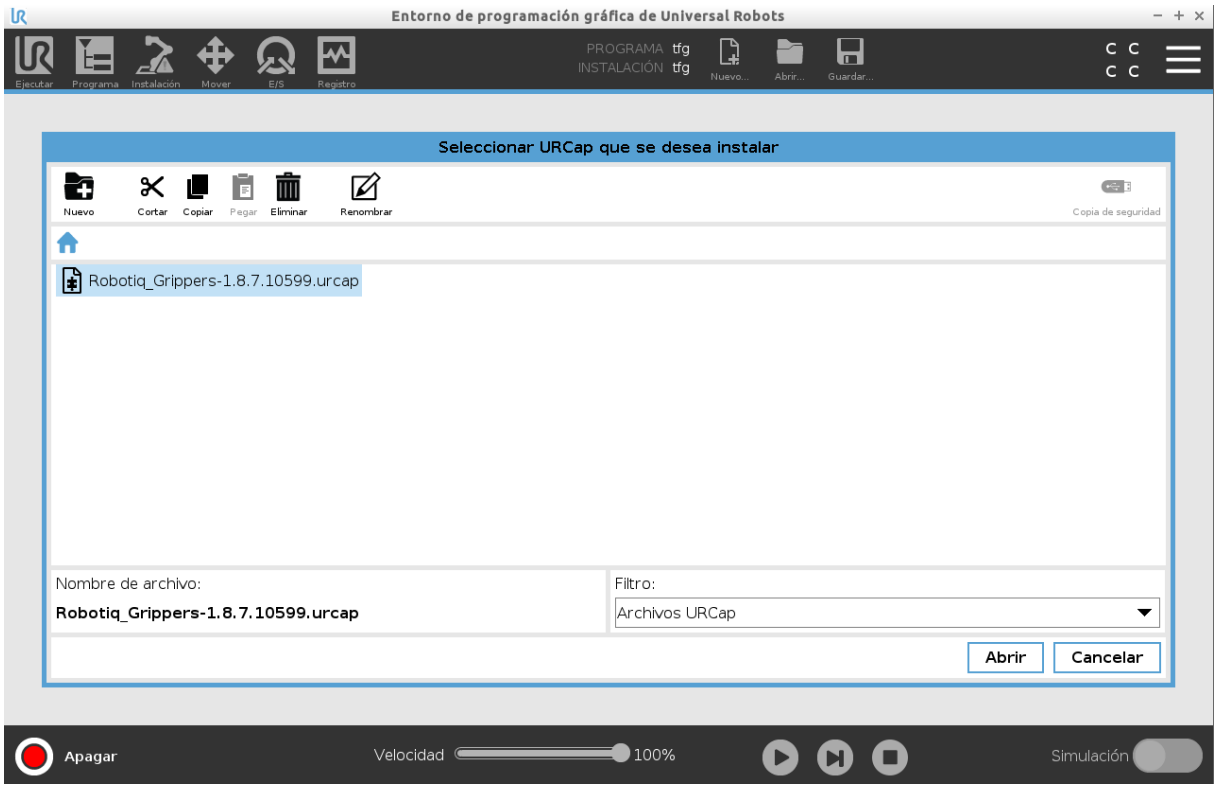

Figura 35. URSim UR5. Instalar URCap

Una vez que el software ha sido instalado, para poder hacer uso de las funciones de la pinza en el programa a desarrollar se selecciona como controlador de la interfaz de entradas y salidas de la herramienta el Robotiq_Grippers como se visualiza en la figura 36.

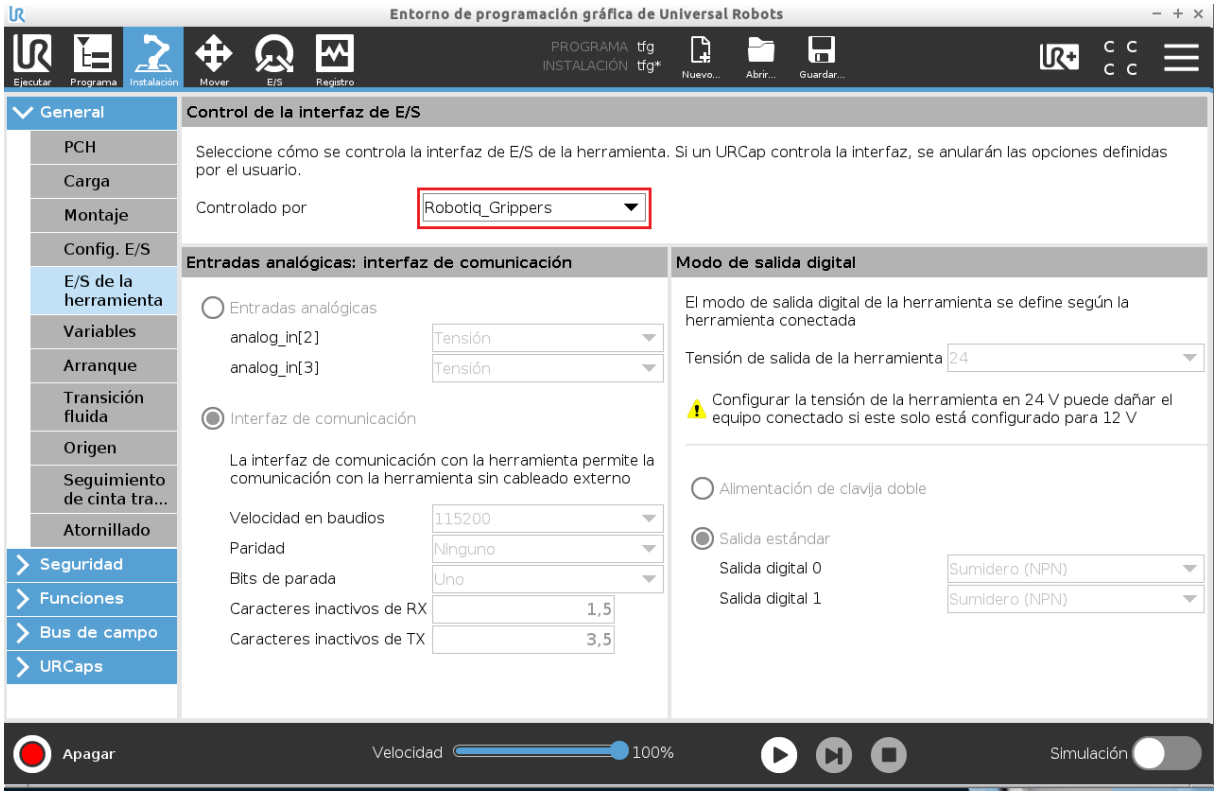

Figura 36. URSim UR5. Control de E/S de la herramienta

Además, en el caso de una instalación real, sería necesaria la activación de la pinza tal y como se muestra en la figura 37.

Dado que para la realización del presente proyecto no se dispone del brazo de Universal Robots, se desarrolla un programa en el que tanto las salidas y entradas de la herramienta, como las correspondientes al autómata que controla el robot, se accionen desde la propia interfaz de PolyScope permitiendo así mostrar el funcionamiento de la aplicación.

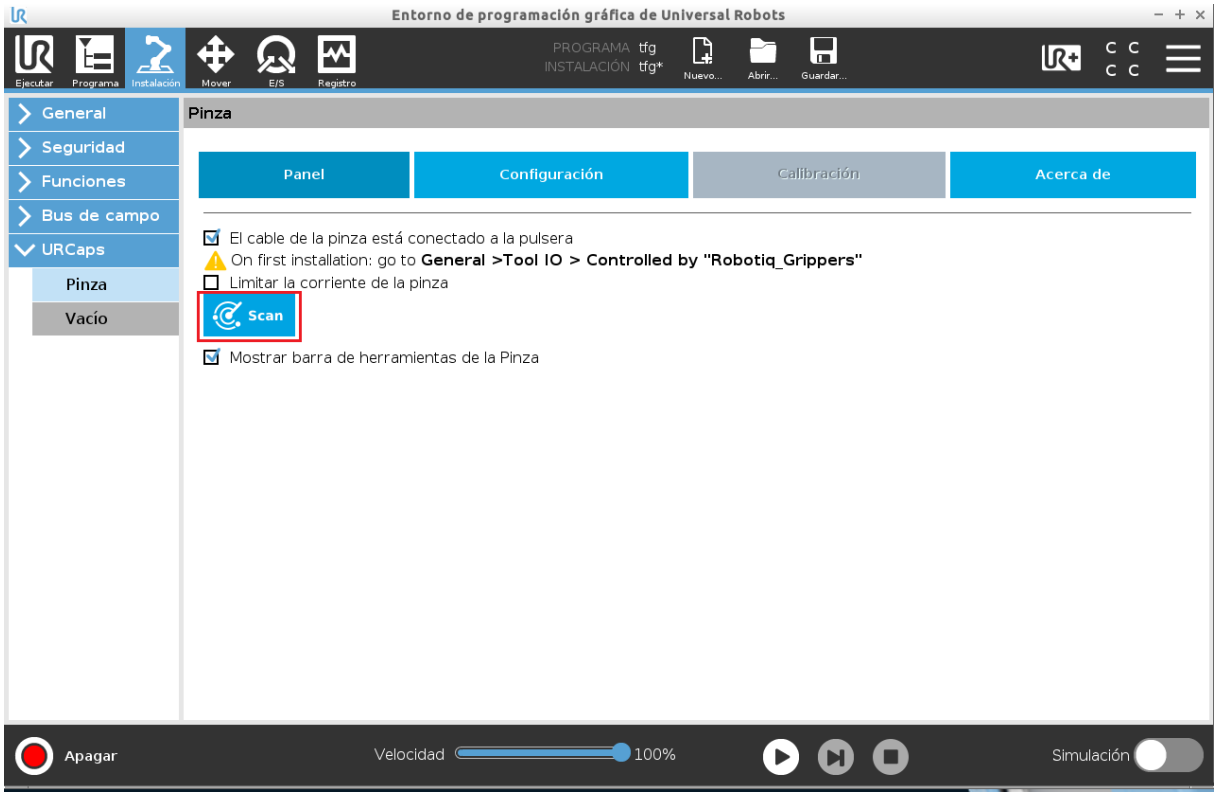

Figura 37. URSim UR5. Escanear herramienta

7.1.4.4. Desarrollo del programa para simulación

En el presente apartado se muestra un programa que permite la demostración de la operación de la aplicación a desarrollar en este proyecto. Dado que no se dispone del brazo robótico para la realización del presente trabajo de fin de grado, se considera este apartado necesario para entender el funcionamiento del aplicativo.

Para comenzar con el desarrollo del programa, se navega a la pantalla de "Programa" y una vez allí se dispone de una serie de funciones básicas y avanzadas con los que se puede proceder de manera clara e intuitiva al desarrollo de la aplicación deseada.

En el caso del presente proyecto la aplicación consistirá en un pick & place que clasificará dos tipos de piezas diferentes: arandelas y piñones.

En primer lugar, se deberá insertar un bucle, función que se encuentra en el apartado "Avanzado" como se puede ver en la figura 38. Después, posicionar el brazo robótico encima de la pieza a clasificar. Para ello, como se explica en el apartado anterior, se parte de la suposición de que la cinta transportadora por las que pasan las piezas a clasificar se sitúa a la misma altura Z en la que se encuentra instalada la base del robot y que el punto donde se sitúa la pieza a recoger se sitúa alineada a una distancia de 500 mm a la izquierda del centro de la base, esto es: X = 0 mm e Y = -500 mm.

Para posicionar el robot se inserta una función "Mover" y se asigna las siguientes coordenadas al punto de paso "Tomar_foto" (tal y como se muestra en las figuras 39, 40 y 41):

- X = 0
- Y = -500
- Z = 100

De esta forma, el extremo de la pinza quedará situada 100 mm por encima de la mesa evitando golpear con ningún objeto y permitiendo realizar la foto mediante una cámara instalada en la última extremidad del robot, pero cuyo funcionamiento y activación no es objeto del presente proyecto.

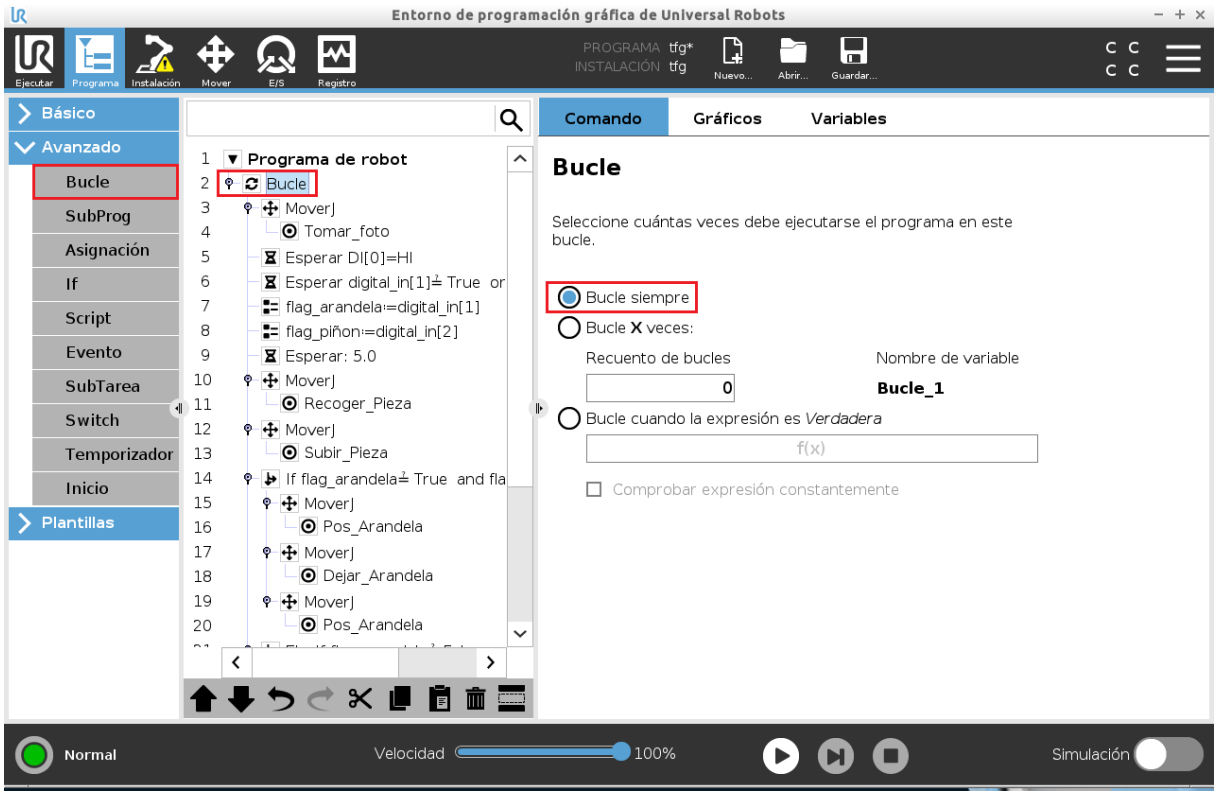

Figura 38. URSim UR5. Programa de robot - Bucle

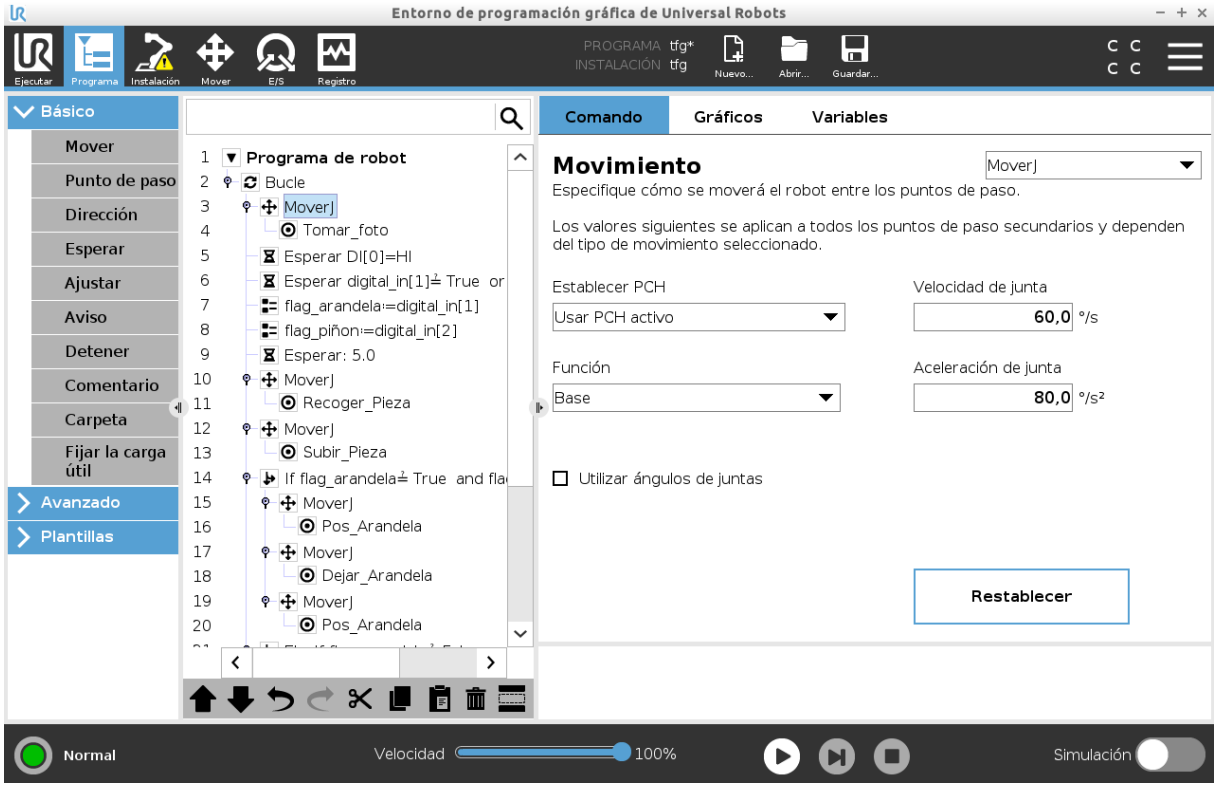

Figura 39. URSim UR5. Programa de robot - Mover

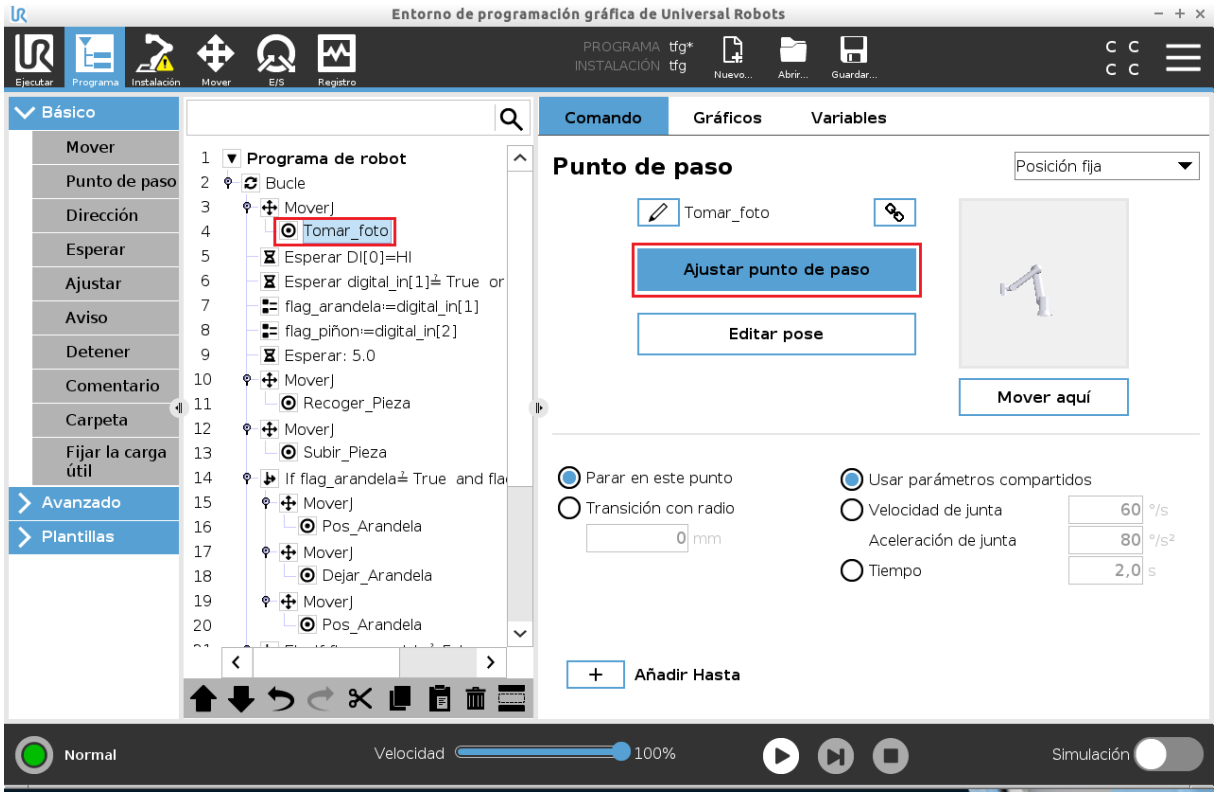

Figura 40. URSim UR5. Programa de robot - Tomar foto

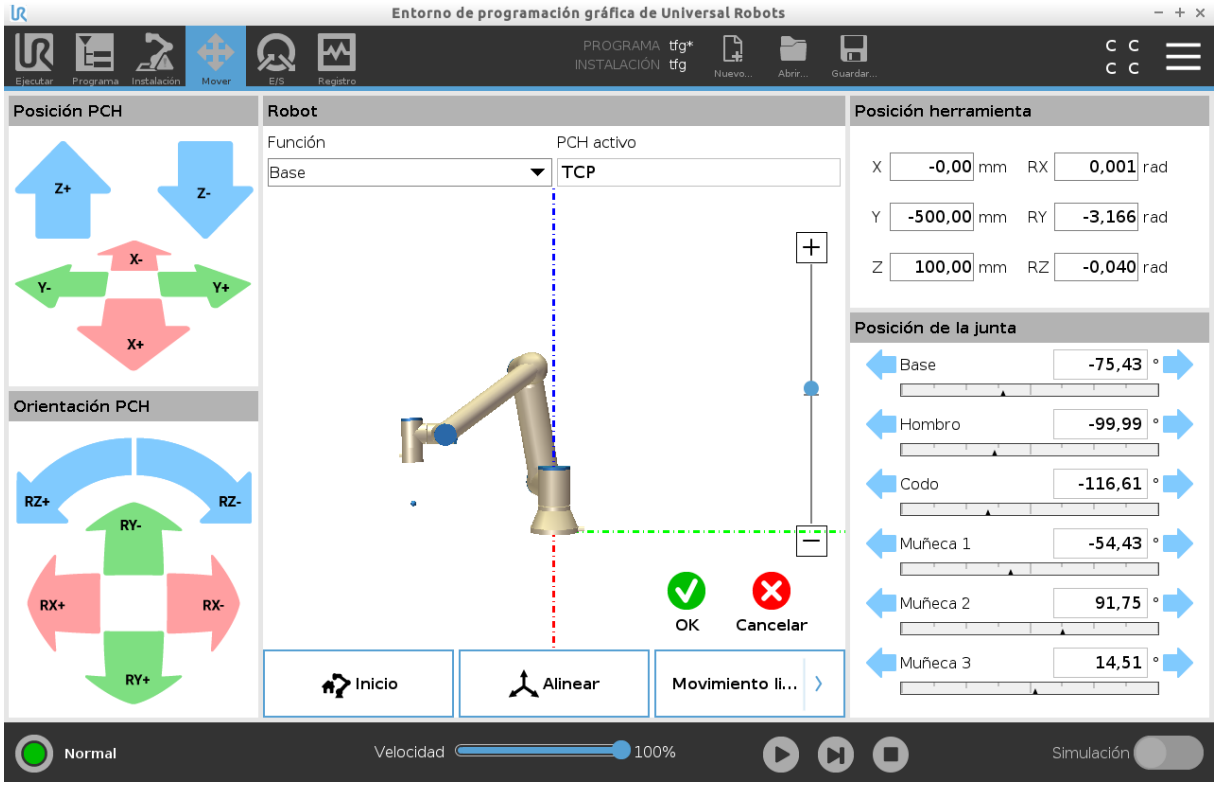

Figura 41. URSim UR5. Programa de robot - Ajustar punto de paso

Una vez realizada la fotografía, esta se envía a la nube de AWS (Amazon Web Services) y es procesada para determinar si se corresponde con una arandela. Cuando la fotografía ha sido procesada se activa la entrada digital del robot “digital_in[0]”, que actuará como flag del procesamiento y bien la entrada “digital_in[1]” (flag de arandela) o la entrada “digital_in[2]” (flag de piñón). Mientras se ha estado realizando este procesamiento el robot se ha encontrado en espera. Esta espera se logra mediante la función “Esperar” como se muestra en la figura 42.

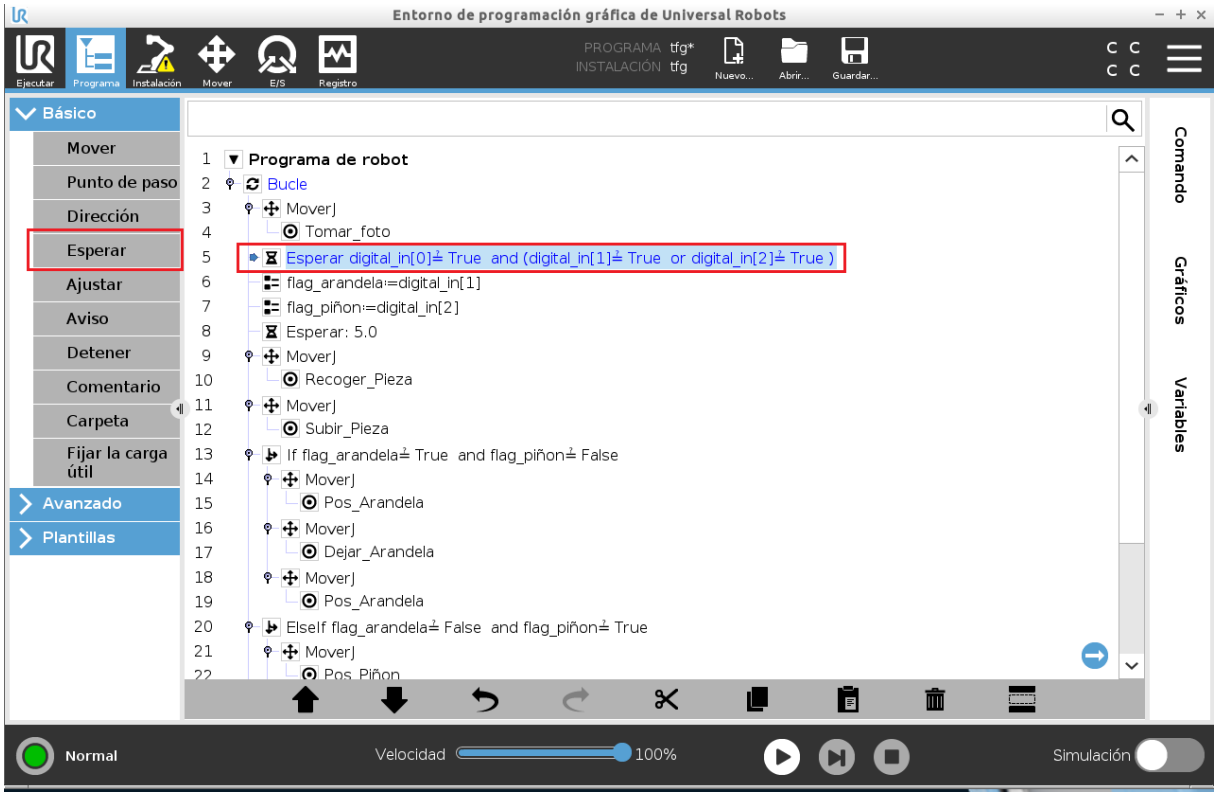

Figura 42. URSim UR5. Programa de robot - Esperar procesamiento de imagen

En el siguiente punto, se crean dentro del programa las variables “flag_arandela” y “flag_piñon” igualándolas a las entradas digitales “digital_in[1]” y “digital_in[2]” respectivamente para hacer más intuitivo y sencillo la continuación del programa.

Posteriormente el robot bajará 100 mm en el eje Z para poder recoger la pieza a clasificar y se cerrará la pinza mediante la activación de la salida de herramienta “tool_out[0]” y esperará 3 segundos para volver a subir la pieza a la altura Z = 100 mm para hacer el movimiento de clasificación sin peligro de golpear en los distintos objetos que puedan estar presentes en las inmediaciones del robot.

- “Recoger_Pieza”: X = 0 mm, Y = -500 mm, Z = 0 mm
- “Subir_Pieza”: X = 0 mm, Y = -500 mm, Z = 100 mm

Una vez que el robot tiene la pieza en la posición (0, -500, 100) mm, evalúa una función if/else que es la encargada de determinar los siguientes movimientos del robot en función de si la pieza se trata de una arandela o de un piñón. Si se trata de una arandela, esto es que la variable “flag_arandela” ha tomado un valor TRUE al activarse la entrada “digital_in[0]”, se realizará un movimiento al punto “Pos_Arandela” y posteriormente al punto “Dejar_Arandela”.

Suponiendo que el punto de disposición de las arandelas se encuentra en otra cinta transportadora situada justo delante del robot, a una distancia de 500 milímetros y a la misma altura de instalación del robot, los puntos de posicionamiento del robot y de disposición de la pieza son los siguientes:

- “Pos_Arandela”: X = 500 mm, Y = 0 mm, Z = 100 mm

- “Dejar Arandela”: X = 500 mm, Y = 0 mm, Z = 0 mm

De manera análoga se realizará el posicionamiento del robot y disposición cuando la pieza detectada sea un piñón. Con la única diferencia de que esta vez la disposición se realizará en una tercera cinta transportadora ubicada a la derecha del robot, a 500 mm. De esta forma los puntos “Pos_Piñon” y “Dejar_Piñon” son los siguientes:

- “Pos_Piñon”: X = 0 mm, Y = 500 mm, Z = 100 mm
- “Dejar_Piñon”: X = 0 mm, Y = 500 mm, Z = 0 mm

Una vez alcanzada la posición de disposición de la pieza, independientemente de si se trata de una arandela o un piñón, se realizará una espera a que se desactive la salida digital de la herramienta “tool_out[0]”, lo que simulará la apertura de la pinza y, posteriormente, 3 segundos para volver a la posición inicial de la aplicación para realizar la fotografía de la siguiente pieza que haya llegado por la primera cinta transportadora, realizar su procesamiento y continuar con el ciclo de la aplicación.

En las figuras 43 y 44 queda recogido el programa del robot desarrollado para su simulación.

En el siguiente apartado, se muestra una serie de capturas que permiten visualizar un ciclo de recogida y clasificación de una arandela.

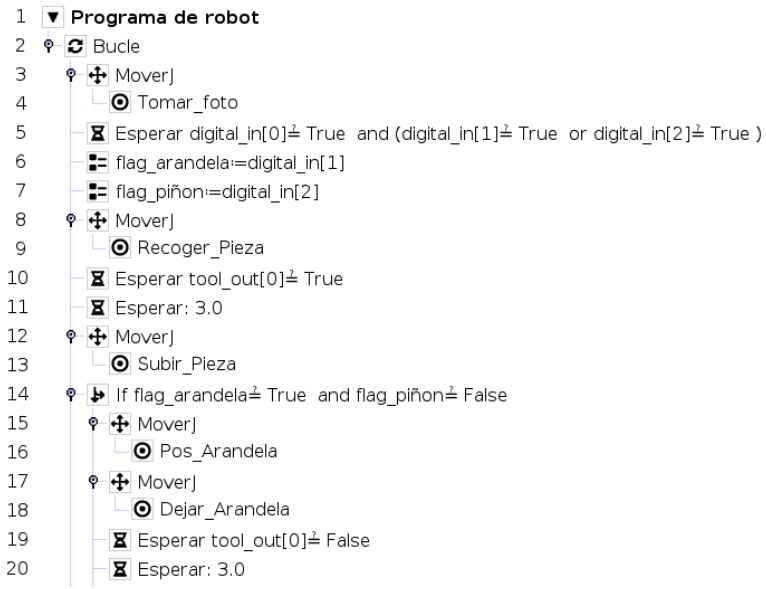

Figura 43. URSim UR5. Programa, primera parte

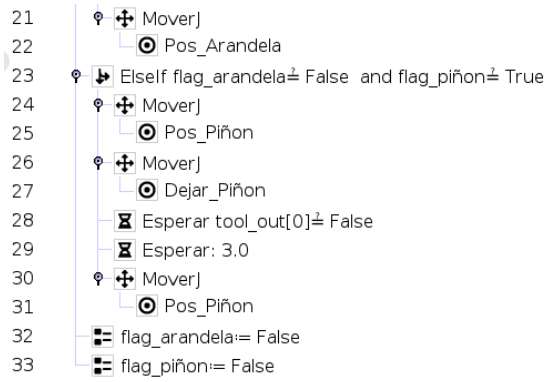

Figura 44. URSim UR5. Programa, segunda parte

7.1.4.5. Simulación de la aplicación

En primer lugar, para poder ver una simulación de la aplicación, es decir, ver los movimientos en 3D que realiza el robot, se necesita encender el UR5e. Una vez encendido hay que activar la opción de "Simulación" y hacer clic en el botón de "Play" tal y como se muestra en la figura 45.

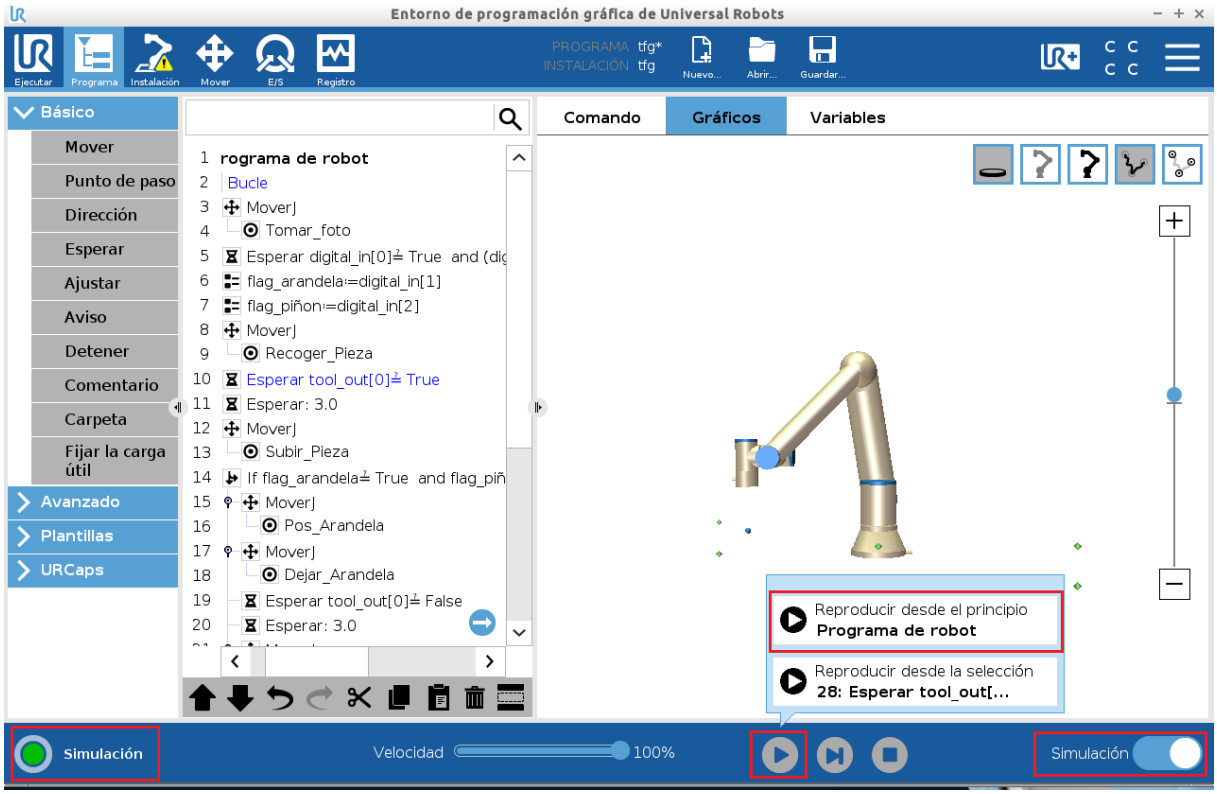

Figura 45. URSim UR5. Simular programa de robot

La simulación de la aplicación queda recogida de la figura 46 a la 52.

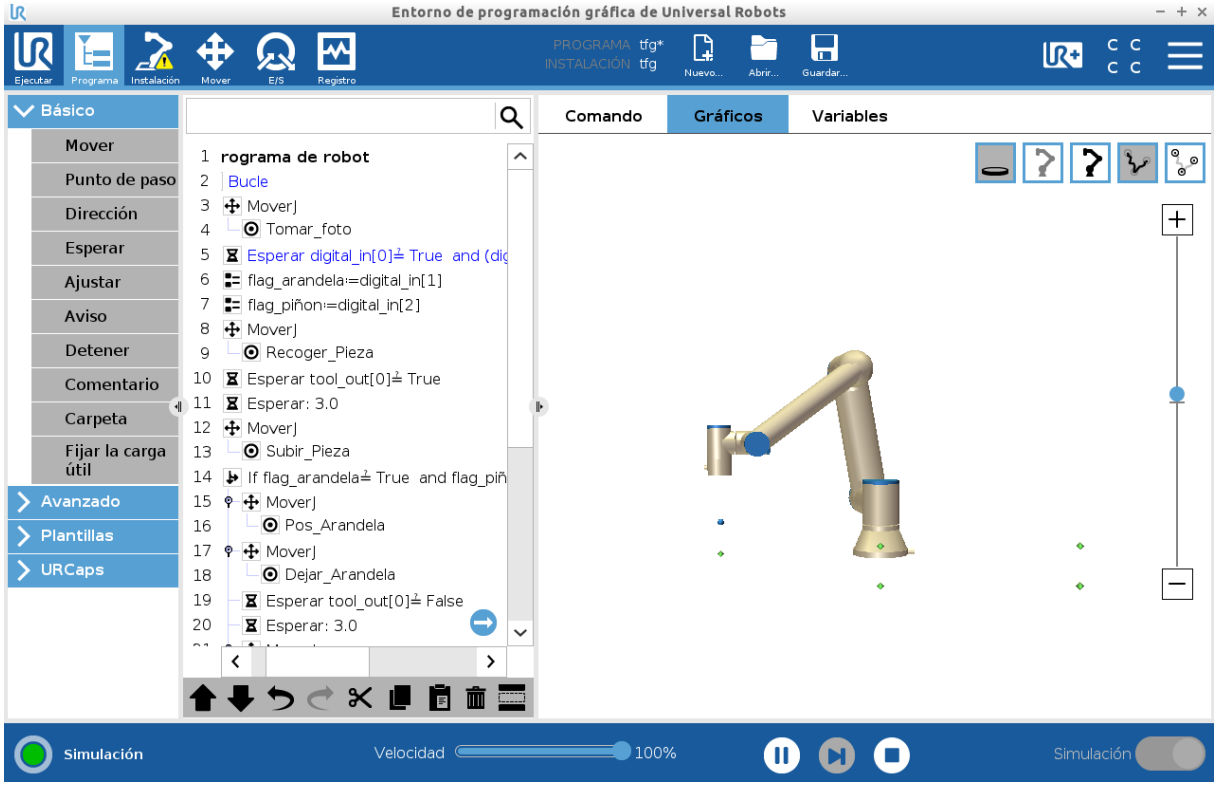

Figura 46. URSim UR5. Simulación - Tomar foto

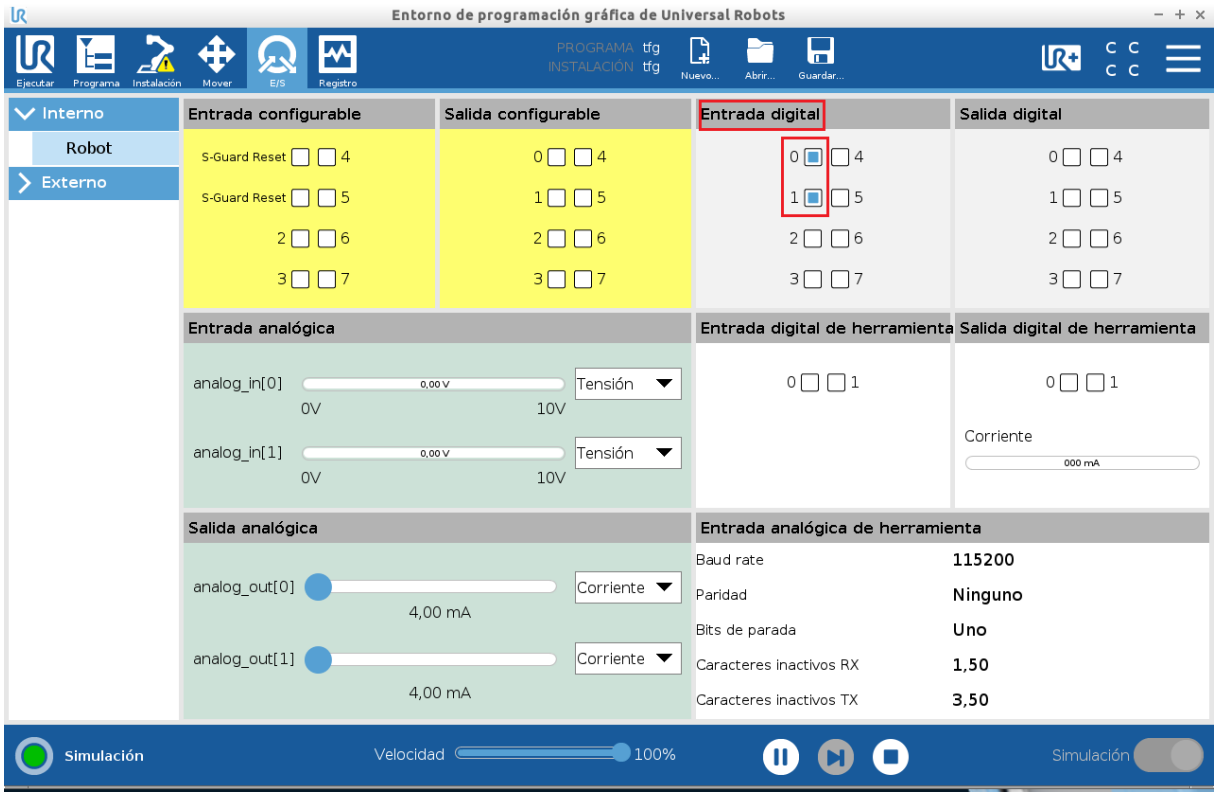

Figura 47. URSim UR5. Simulación - Foto de una arandela procesada

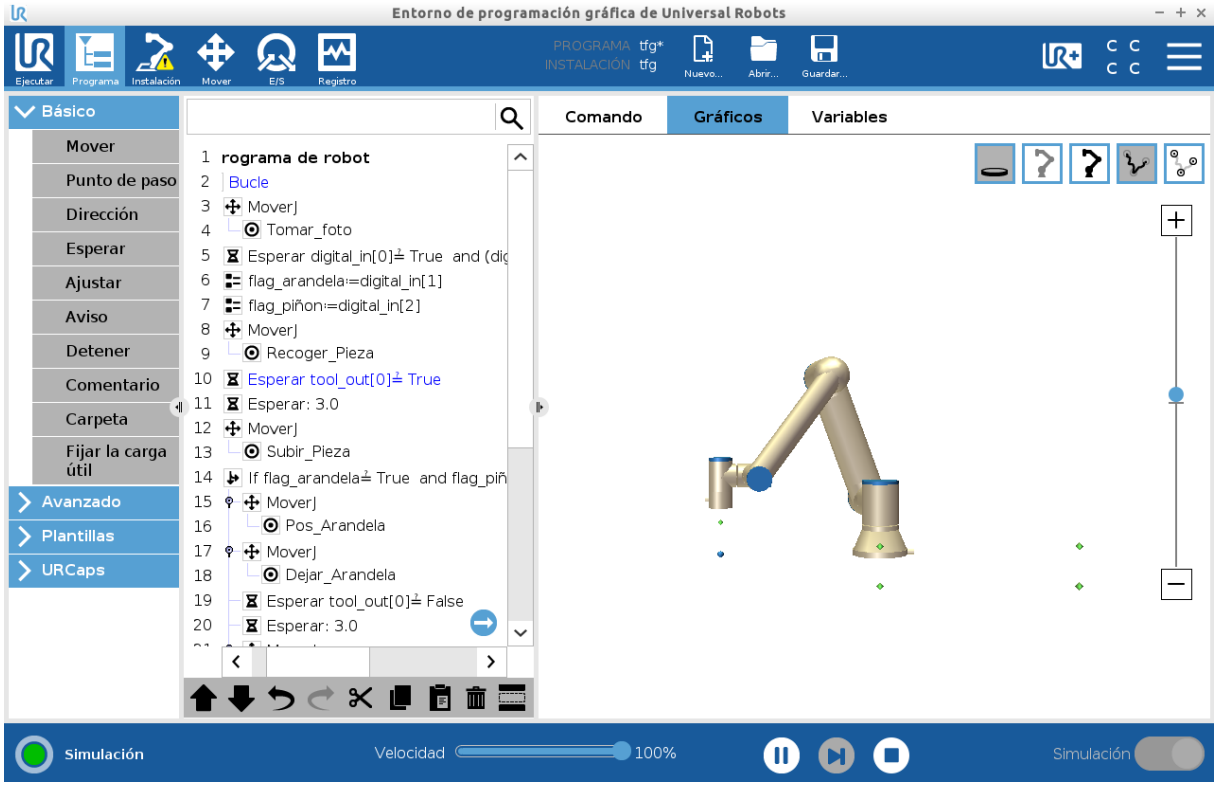

Figura 48. URSim UR5. Simulación - Esperar a cerrar pinza para coger la arandela

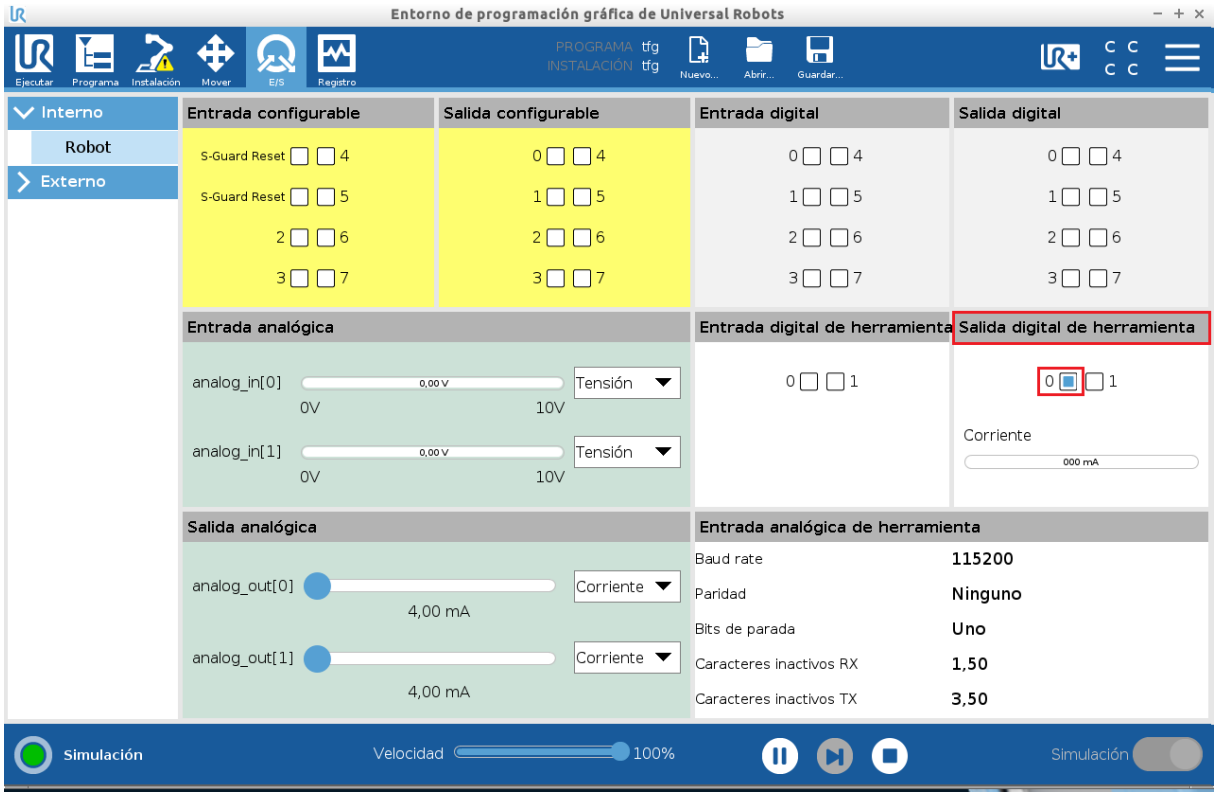

Figura 49. URSim UR5. Simulación - Pinza cerrada

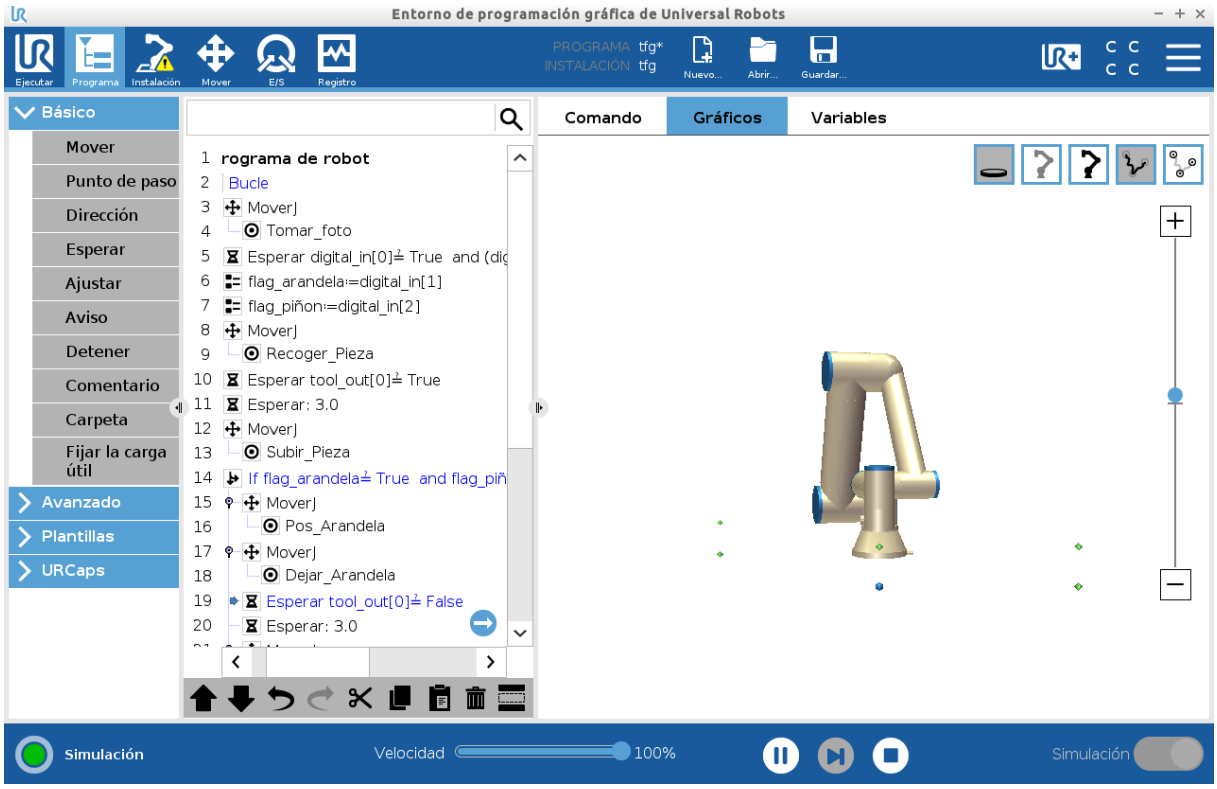

Figura 50. URSim UR5. Simulación - Posición de dejar arandela, esperando a que se abra de nuevo la pinza

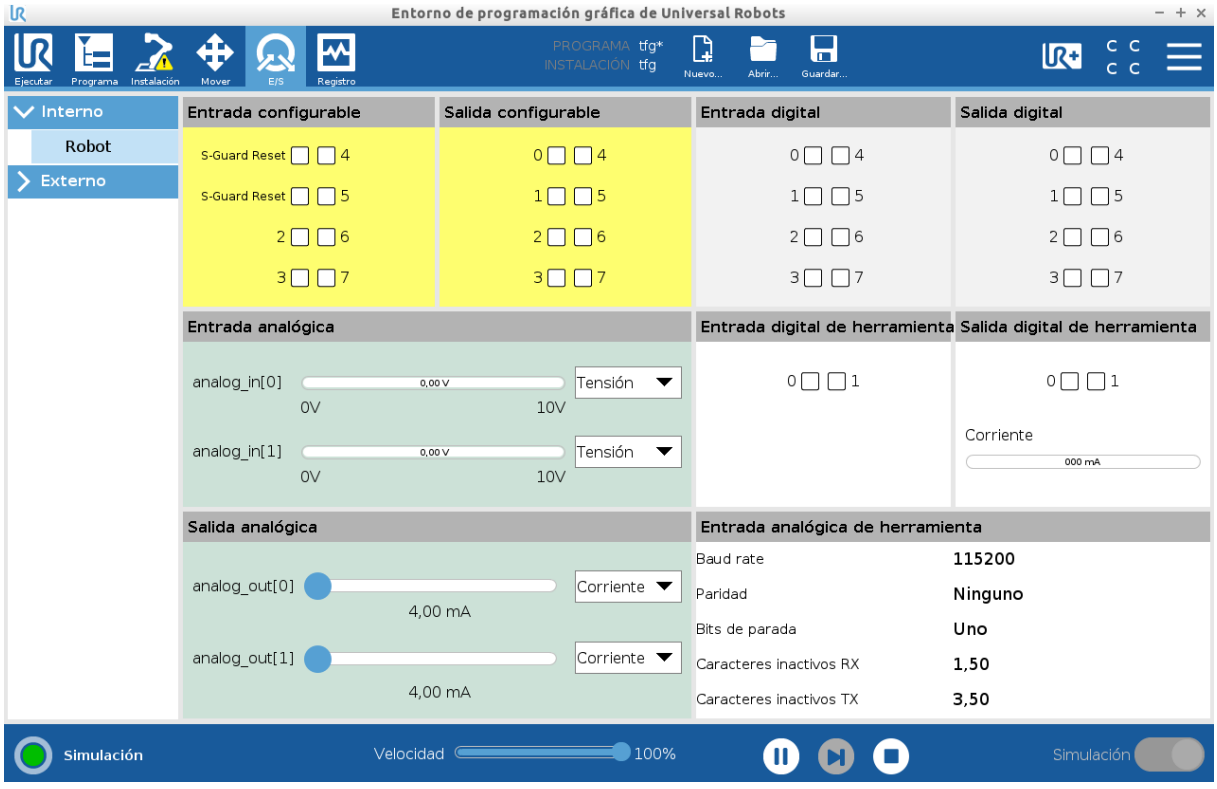

Figura 51. URSim UR5. Simulación - Pinza abierta

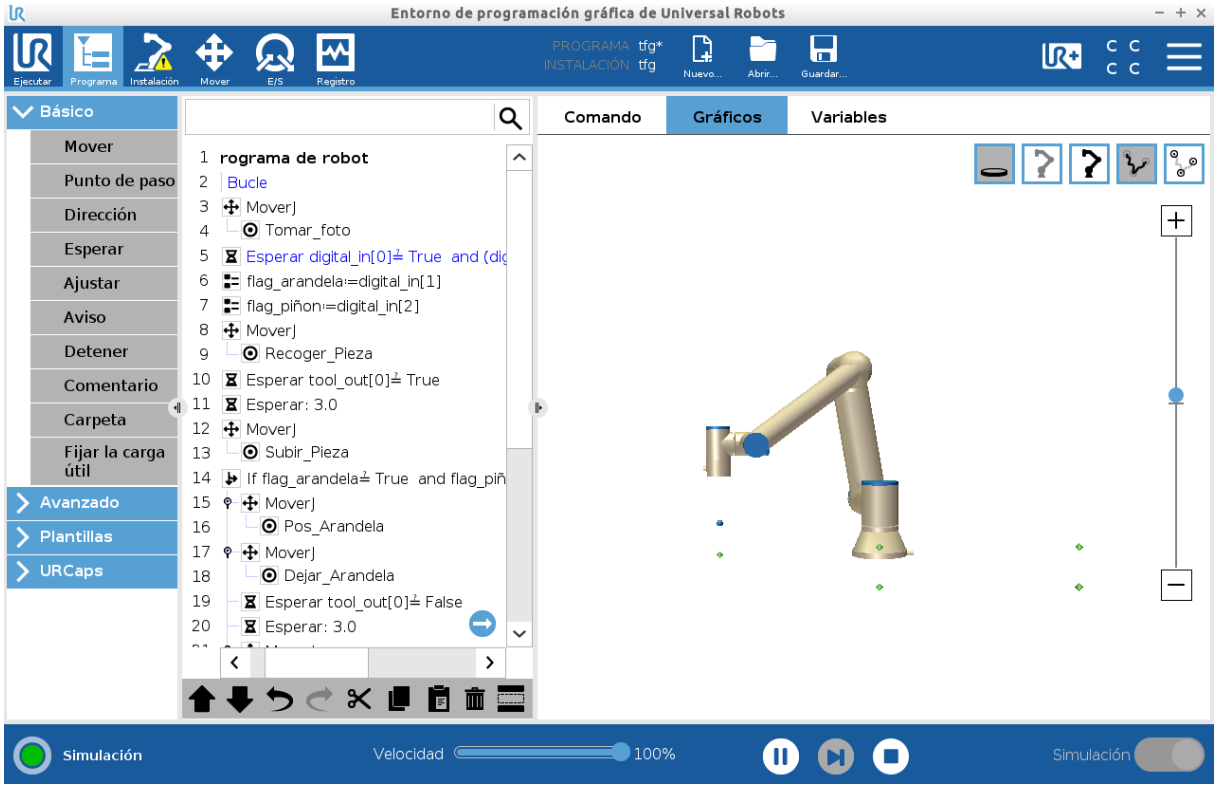

Figura 52. URSim UR5. Simulación - Vuelta a origen para repetir ciclo

7.1.4.6. Desarrollo del programa real

Como se indica en el apartado 7.1.3.2.4. de este mismo documento, el programa desarrollado en el mismo se corresponde con un programa simulador. Este programa se usa para poder demostrar la operación ante la imposibilidad de poder mostrarla con el programa real dado que no se dispone de los dispositivos involucrados en este proyecto sino de sus respectivos simuladores.

En este apartado se pretende mostrar y explicar el conjunto de secuencias que conforman el programa real y el que habría que cargar en el UR5e real en caso de disponer de este junto a la pinza de Robotiq, Hand-E y el PLC de SIEMENS CPU 1214C.

El desarrollo de este programa está basado en los comandos citados y explicados tanto en manual de Robotiq para la Hand-E [7] como en la documentación de URScript [9] facilitada por Universal Robots.

En primer lugar, se introduce la función “Esperar read_input_boolean_register(0) = True”. Esta función permitirá leer el bit 0 del registro de entradas booleanas del robot, que se activará al pulsar el botón que activa a su vez la “Marcha” desde el PLC. Una vez que se ha puesto en marcha la operación, el robot se mueve hasta la posición de “Tomar_foto” y después se activa y abre la pinza. Esto lo conseguimos mediante la siguiente secuencia de comandos:

- **MoverJ, Tomar_foto**
- **rq_reset()**: Este comando reinicia el estado de activación de la pinza y es necesario ejecutarlo antes de un “rq_activate()” o un “rq_activate_and_wait” desde la versión 1.0.2

del URCap. En el presente proyecto se trabaja con la versión 1.8.7, por lo que es necesario este reinicio.
- **rq_activate_and_wait()**: Envía a la pinza el comando de activación. En el caso de que la pinza ya esté activada en ese momento no pasa nada. Además, la ejecución del programa espera a que la pinza esté activada.
- **rq_open_and_wait()**: Mueve la pinza hasta su posición totalmente abierta y espera a que la apertura haya terminado para proseguir con la ejecución del programa.

Una vez activa y abierta la pinza, entramos en el bucle que repetirá el ciclo de programa. Antes de iniciarse los movimientos necesarios para la clasificación de las piezas tenemos la siguiente secuencia:

- **write_output_boolean_register(0, True)**: Con este comando se activa un nivel alto en el bit 0 del registro de salidas booleanas del robot. De esta forma se avisa al PLC de que se ha alcanzado la posición adecuada para tomar la fotografía y que puede ejecutar el disparo de la cámara.
- **Esperar read_input_boolean_register(1) = True**: Obliga al robot a esperar a que el PLC le comunique mediante el bit 1 del registro de entradas booleanas que ha terminado el procesamiento de la imagen tomada.
- **write_output_boolean_register(0, False)**: Se vuelve a poner a nivel bajo el bit 0 del registro de salidas booleanas de cara a la ejecución del siguiente ciclo de trabajo.

A partir de este punto, el programa es bastante parecido al descrito en la simulación, con la salvedad de que en este caso se usan las funciones oportunas para apertura y cierre de la pinza y las señales de salida del PLC (entrada del UR5e) para identificar si la pieza de ese ciclo se trata de una arandela o de un piñón. Estas funciones son los que se muestran a continuación:

- **rq_move_and_wait_norm(90)**: Una vez que el robot ha alcanzado la posición "Recoger_Pieza" se ejecuta esta función que lo que hace es cerrar la pinza hasta su 90% de apertura. Si se tiene en cuenta que la apertura máxima de la pinza con 50 mm y que las arandelas y piñones a clasificar tienen un diámetro de 45 mm, la pinza deberá cerrarse hasta el 90%. Además, se espera a que haya concluido el movimiento para continuar con la ejecución de la aplicación. De manera equivalente se podría haber utilizado la función **rq_move_and_wait_mm(45)**.
- **If read_input_boolean_register(2) = True**. En el caso de que la pieza se trate de una arandela, el PLC activará el bit 2 del registro de entradas booleanas del robot. De esta forma si se cumple esta condición se realizarán los movimientos correspondientes para dejar la arandela en su posición final.
- **Elseif read_input_boolean_register(3) = True**. De manera análoga a la arandela, el bit 3 del registro de entradas booleanas indica al robot que debe realizar los movimientos oportunos para colocar un piñón en su posición final.
- **rq_open_and_wait()**: Cuando se alcanza los puntos "Dejar_Arandela" o "Dejar_Piñon" se procede a la apertura de la pinza, a través de esta función, para soltar la pieza y que el robot vuelva a su posición inicial ("Tomar_Foto").

En las figuras 53 y 54 se puede visualizar el total del contenido del programa real del robot.

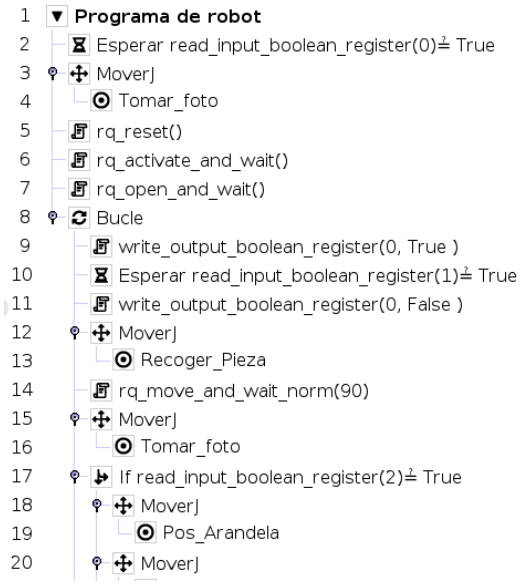

Figura 53. URSim UR5. Programa real, parte 1

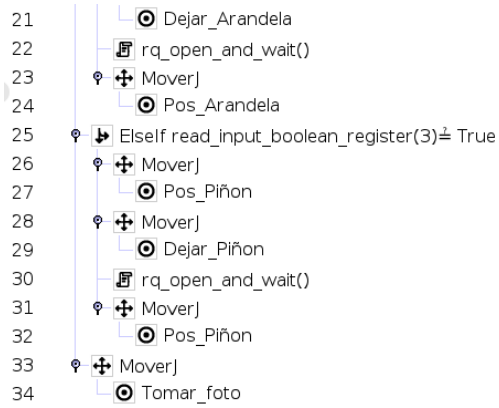

Figura 54. URSim UR5. Programa real, parte 2

Para facilitar la compresión de la aplicación, en la figura 55 se muestra un diagrama de flujo de esta:

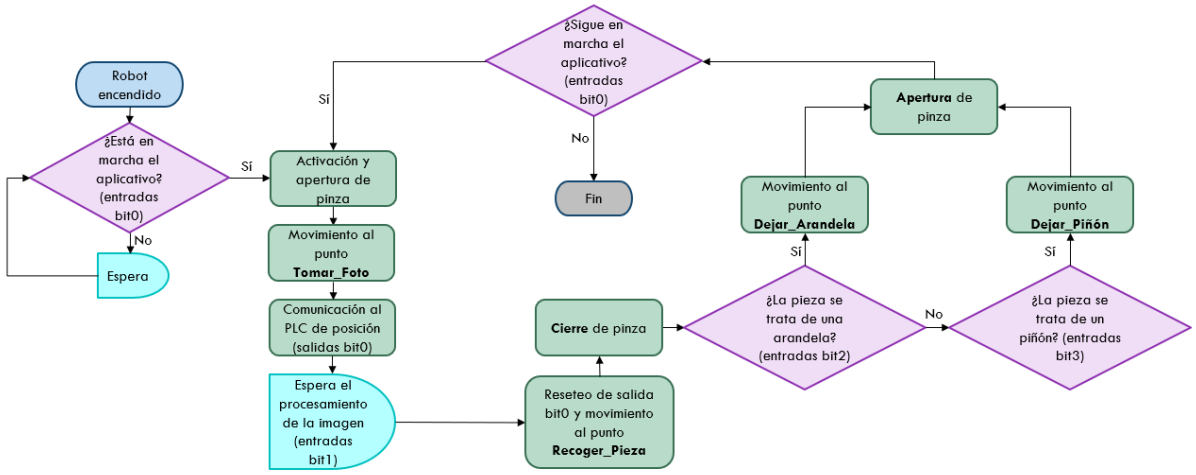

Figura 55. URSim UR5. Programa real - Diagrama de flujo

7.1.4.7. Comunicación UR5e - PLC

Para la aplicación real es necesario que tenga éxito la comunicación entre el PLC y el robot, para se debe habilitar el servicio de PROFINET desde **Instalación>Bus de campo>PROFINET** como se muestra en la figura 56.

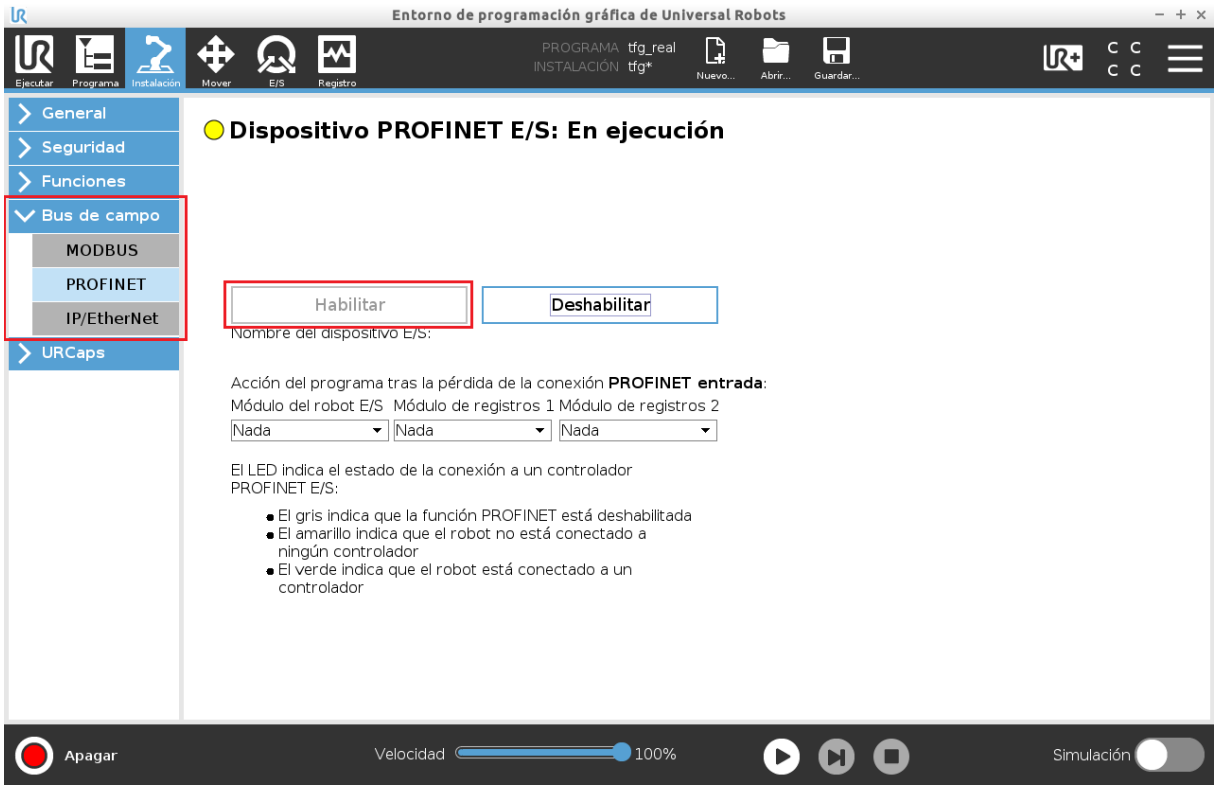

Figura 56. URSim UR5. Comunicación - Habilitar PROFINET

Es importante tener en cuenta que las imágenes del presente proyecto han sido tomadas desde el simulador del robot, es por ello por lo que el indicador de “Dispositivo PROFINET E/S” está

en amarillo. En una instalación real, con un PLC real y el UR5e, este piloto se vería en verde si existiese un cable que conectara ambos dispositivos.

Además de habilitar el protocolo PROFINET en el robot, también es necesario verificar su dirección IP para que esta coincida con la configurada en el PLC, mediante TIA Portal, tal y como se explica más adelante en este mismo documento. En PolyScope, esta modificación se realiza desde **Ajustes>Sistema>Red**, como se puede ver en la figura 57.

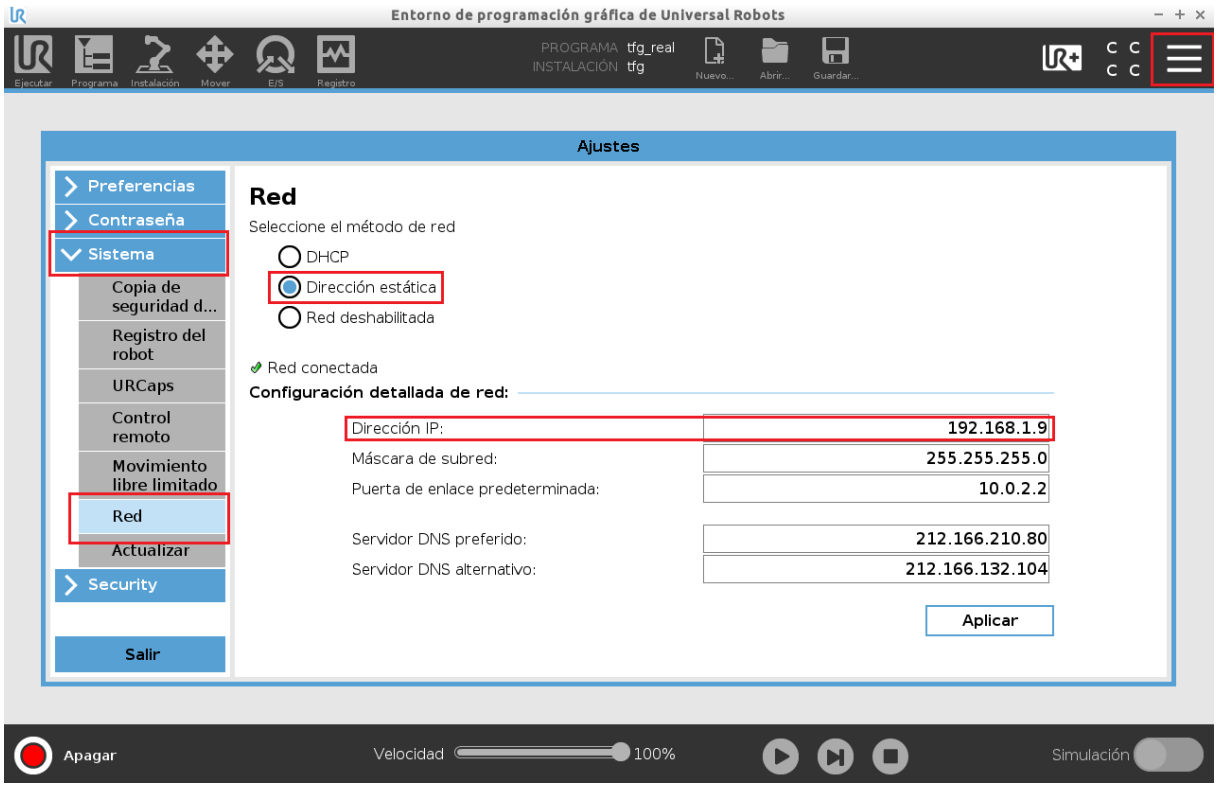

Figura 57. URSim UR5. Comunicación - Modificar IP

7.2. PLC S7-1200

7.2.1. TIA Portal

TIA Portal es un software de la compañía SIEMENS, que permite generar soluciones de automatización para optimizar procesos de ingeniería de forma intuitiva y eficiente. Las siglas TIA son un acrónimo de “Totally Integrated Automation”.

Para el uso de TIA Portal, se crea una máquina virtual de manera análoga a como se ha hecho con la máquina de PolyScope de Universal Robots, con la salvedad de que esta vez la computadora huésped tiene como sistema operativo Windows 2016 (64-bit).

El presente proyecto está realizado con la versión de TIA Portal V16 y toda la información necesaria para su ejecución se ha extraído del manual de sistema [10] facilitado por SIEMENS.

7.2.1.1. Parámetros de configuración de la máquina virtual

En la figura 58 se puede ver una imagen de la configuración general de la máquina de TIA Portal empleada para la realización del proyecto.

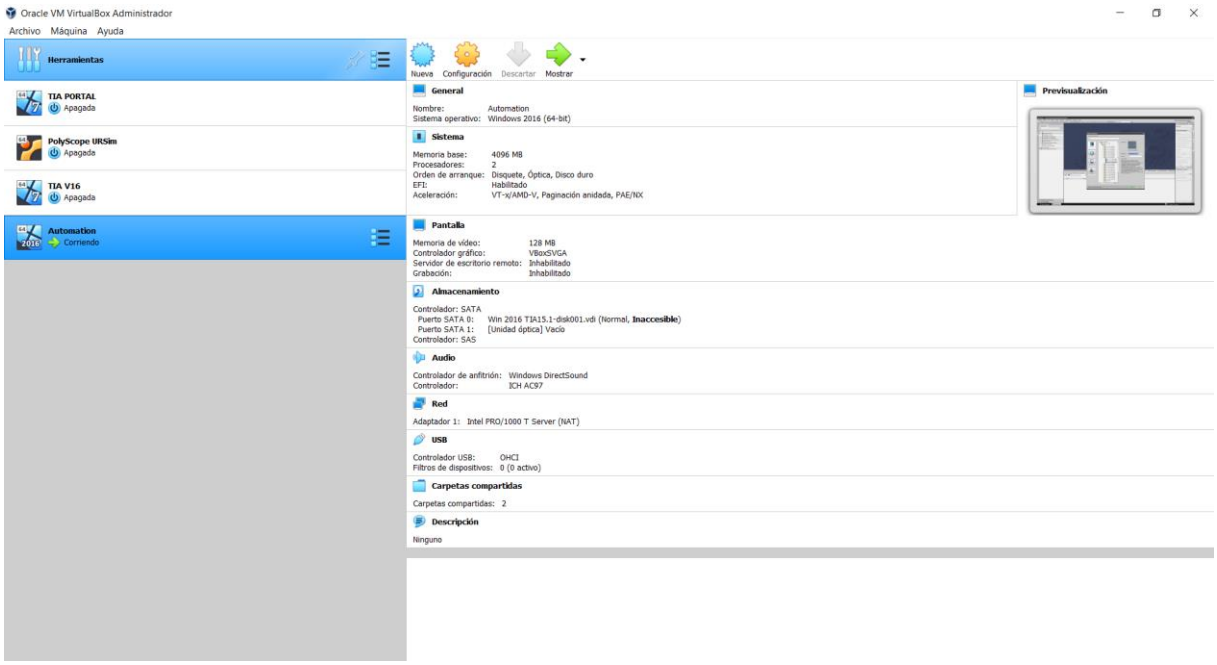

Figura 58. TIA Portal. Máquina virtual

7.2.1.1.1. General

- Nombre: Automation
- Sistema operativo: Windows 2016 (64-bit)

7.2.1.1.2. Sistema

- Memoria base: 4096 MB
- Orden de arranque: Disquete, Óptica, Disco duro
- EFI: Habilitado
- Aceleración: VT-x/AMD-V, Paginación anidada, PAE/NX

7.2.1.1.3. Pantalla

- Memoria de vídeo: 128 MB
- Controlador gráfico: VBoxSVGA

7.2.1.1.4. Almacenamiento

- Controlador: SATA
  - Puerto SATA 0: Win 2016 TIA15.1-disk001.vdi

7.2.1.2. Creación del proyecto

En primer lugar, se crea un proyecto de TIA Portal desde el apartado de "Crear proyecto" otorgando un nombre al proyecto y eligiendo la ruta de la máquina virtual donde este se guardará como se aprecia en la figura 59.

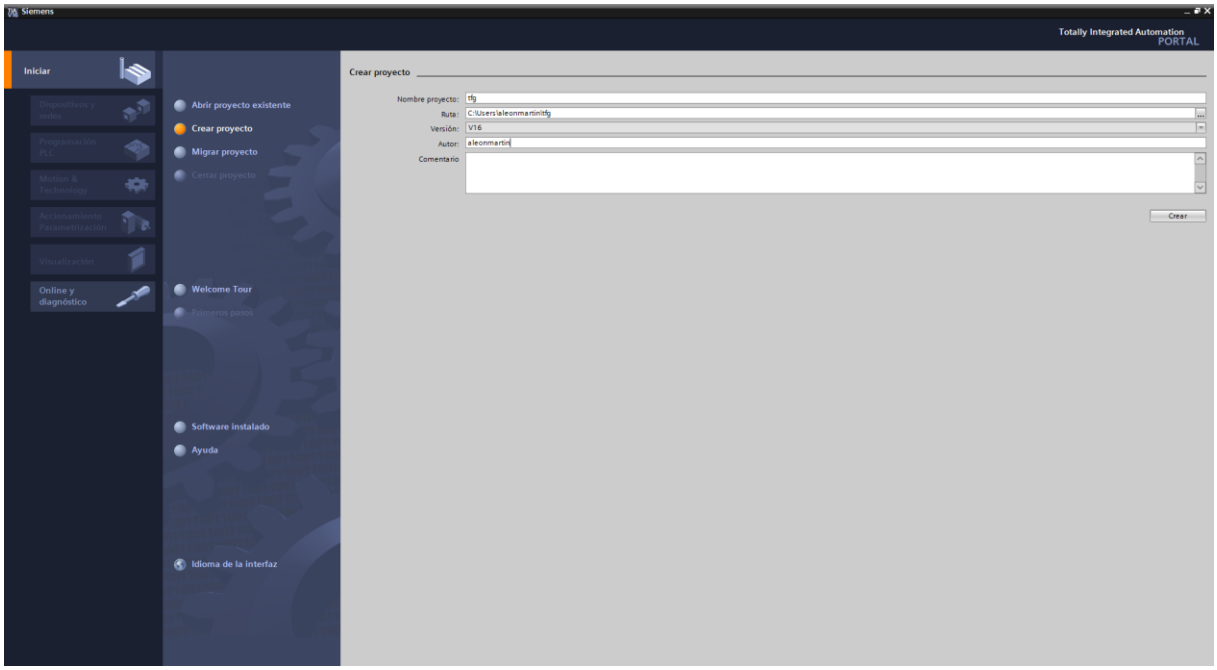

Figura 59. TIA Portal. Crear proyecto

7.2.1.3. Configuración del proyecto

7.2.1.3.1. Agregar autómata

Una vez se tiene el proyecto creado, lo primero que se hace es agregar el autómata desde el apartado de "Agregar dispositivo" como se aprecia en la figura 60. En este caso se escoge la CPU 1215C con número de referencia 6ES7 215-1AG40-0XB0 y versión V4.1, cuyas especificaciones son las que se detallan a continuación:

- Memoria de trabajo: 125 KB
- Fuente de alimentación: 24 V DC
- Señales:
    - 14 entradas digitales 24 V DC
    - 10 salidas digitales 24 V DC
    - 2 entradas analógicas
    - 2 salidas analógicas
    - 6 contadores rápidos
    - 4 salidas de impulso integradas
- Hasta 3 módulos de comunicaciones para comunicación serie
- Hasta 8 módulos de señales para ampliación de E/S
- 2 interfaces de PROFINET para programación, HMI y comunicación PLC-PLC

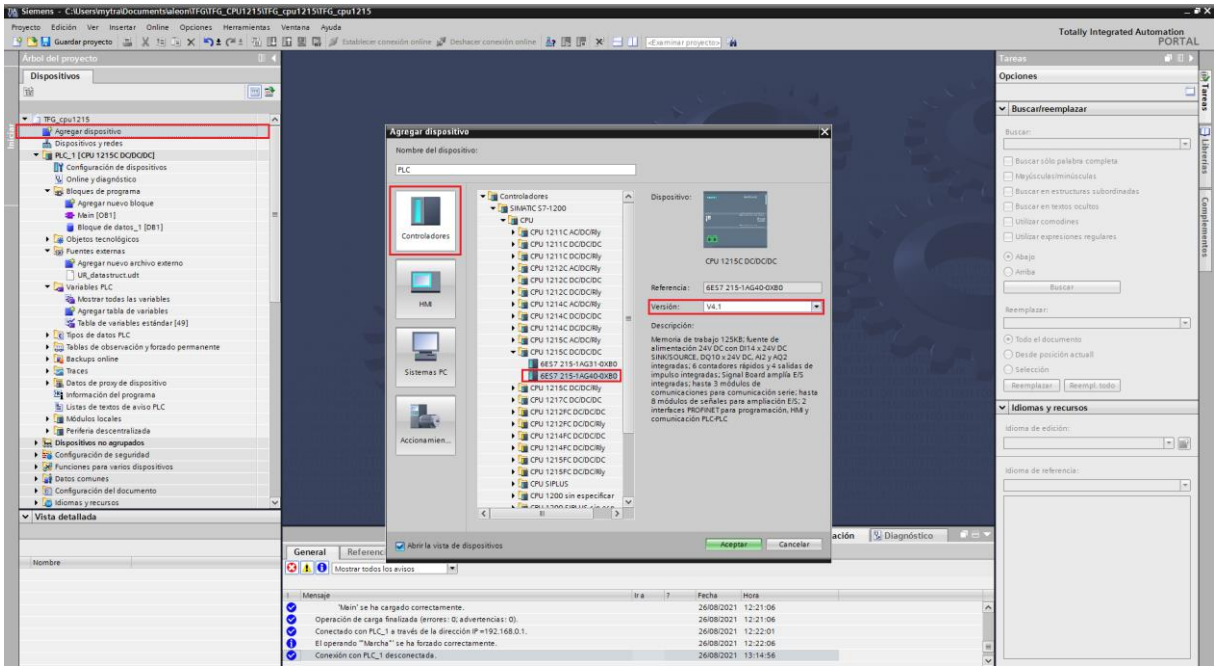

Figura 60. TIA Portal. Configuración de proyecto - Agregar autómata

7.2.1.3.2. Instalación de GSDML del UR5e

En este paso se debe instalar el GSDML del robot, el cual se puede encontrar en la página web de Universal Robots, la cual servirá también como guía para la parte de configuración del proyecto relativa al UR5e:

https://www.universal-robots.com/articles/ur/interface-communication/profinet-guide/

Un GSD es un archivo que contiene las características y capacidades básicas de un equipo. Los GSDMLs son archivos GSD con la particularidad de que están escritos en formato XML y describen las propiedades de los equipos PROFINET. Este tipo de archivos permiten posibilitan la integración de los dispositivos con otras tecnologías y herramientas.

Para la instalación el GSDML se navega hasta **Opciones>Administrar archivos de descripción de dispositivos** y se selecciona el archivo "GSDML-V2.31-ur-UR-20160505", como se visualiza en la figura 61. De esta forma el catálogo de hardware de TIA Portal se actualiza de manera que ya aparece el equipo de Universal Robots.

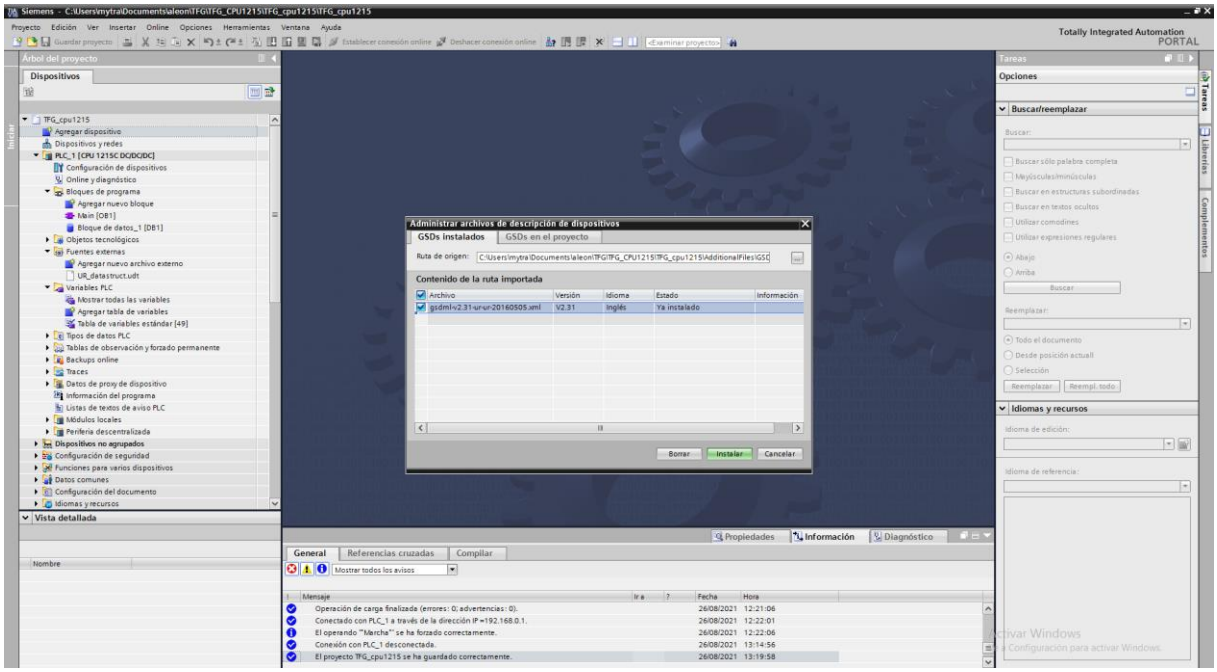

Figura 61. TIA Portal. Configuración de proyecto - Instalar GSDML

7.2.1.3.3. Agregar UR5e

Cuando ya se tiene instalado el GSDML, se agrega el UR5e desde la dirección **Otros dispositivos de campo>PROFINET IO>I/O>Universal Robots A/S Collaborative Robot** del catálogo.

En la figura 62 se puede ver el dispositivo de Universal Robots ya agregado al proyecto.

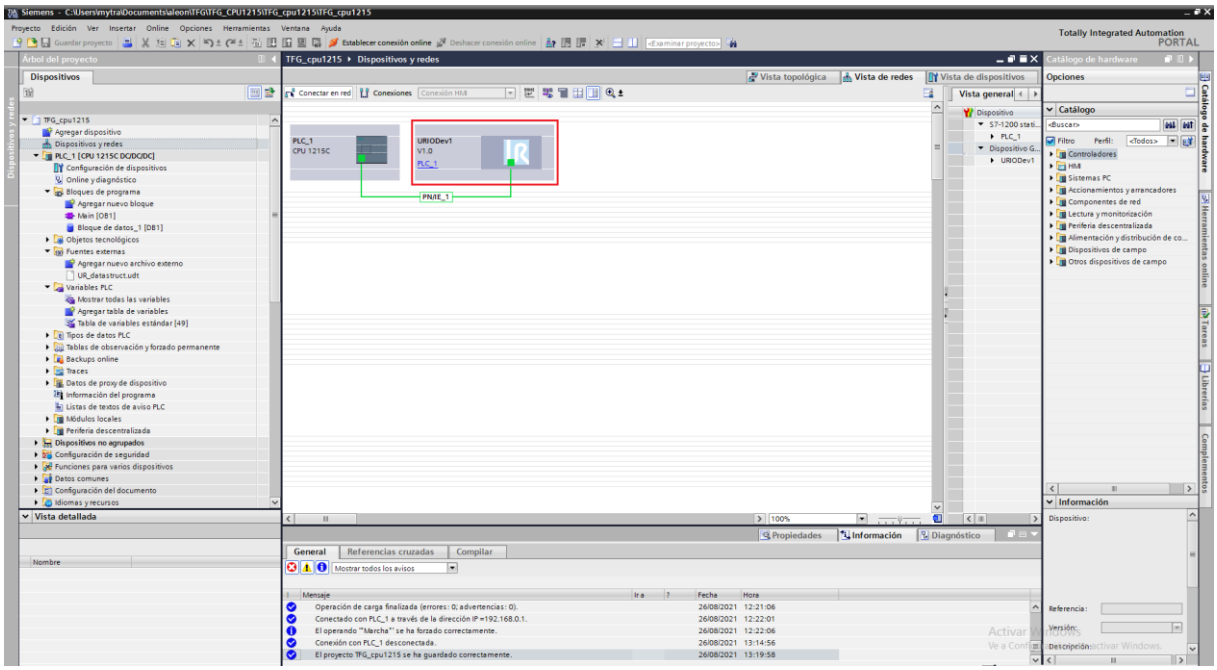

Figura 62. TIA Portal. Configuración de proyecto - Agregar UR5e

7.2.1.3.4. Añadir módulos de comunicación del UR5e

Además, es necesario añadir los módulos de comunicación del robot tal y como se muestra en la figura 63.

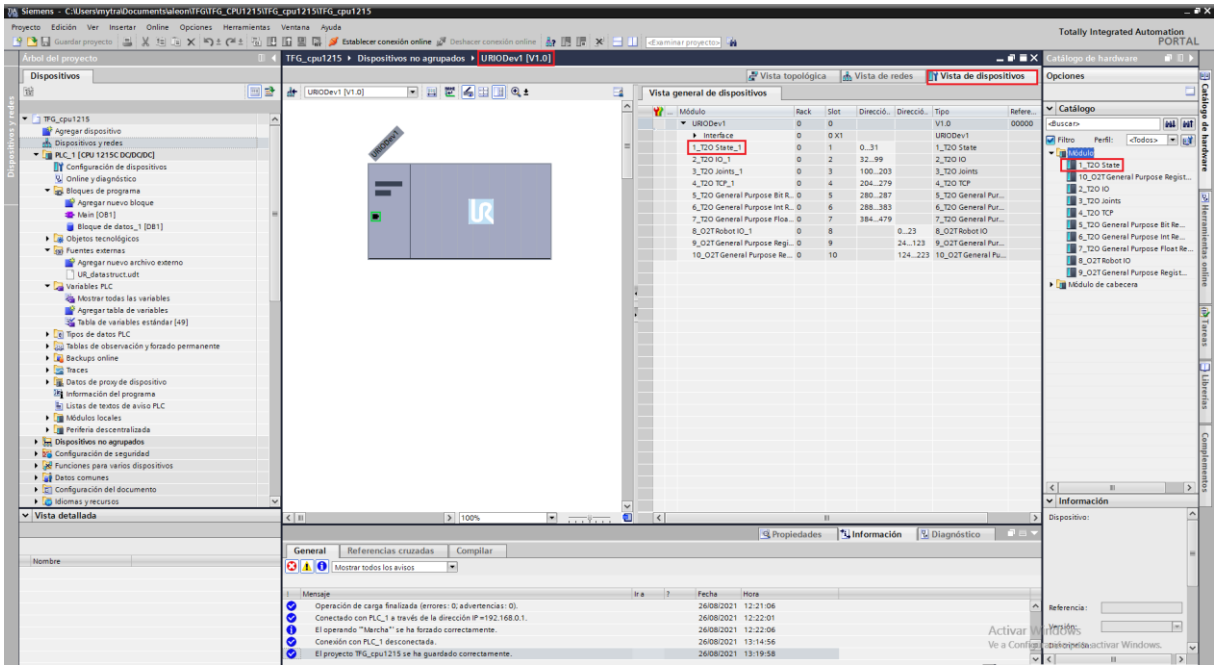

Figura 63. TIA Portal. Configuración de proyecto - Añadir módulos de comunicación del UR5e

7.2.1.3.5. Desplazamiento de señales del autómata

Dado que las entradas y salidas con las que se puede trabajar son muchas, debido al elevado número de variables que tiene el robot internamente, es conveniente realizar un desplazamiento de las señales de entrada y salida, tanto digitales como analógicas, del autómata.

Esto se hace también desde la pantalla de “Vista general de dispositivos” pero esta vez seleccionando el equipo PLC_1 como se visualiza en la figura 64.

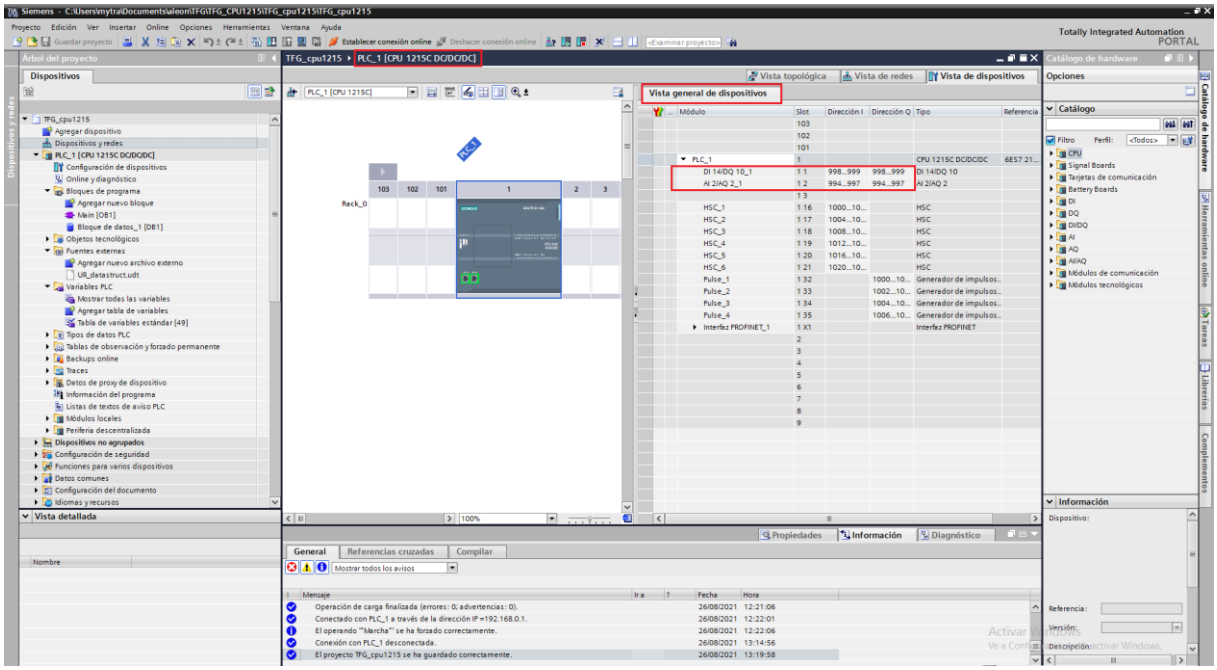

Figura 64. TIA Portal. Configuración de proyecto - Desplazamiento de E/S del autómata

Una vez desplazadas las señales de entrada y salida se selecciona el PLC_1 como controlador del robot de Universal Robots como se aprecia en la figura 65.

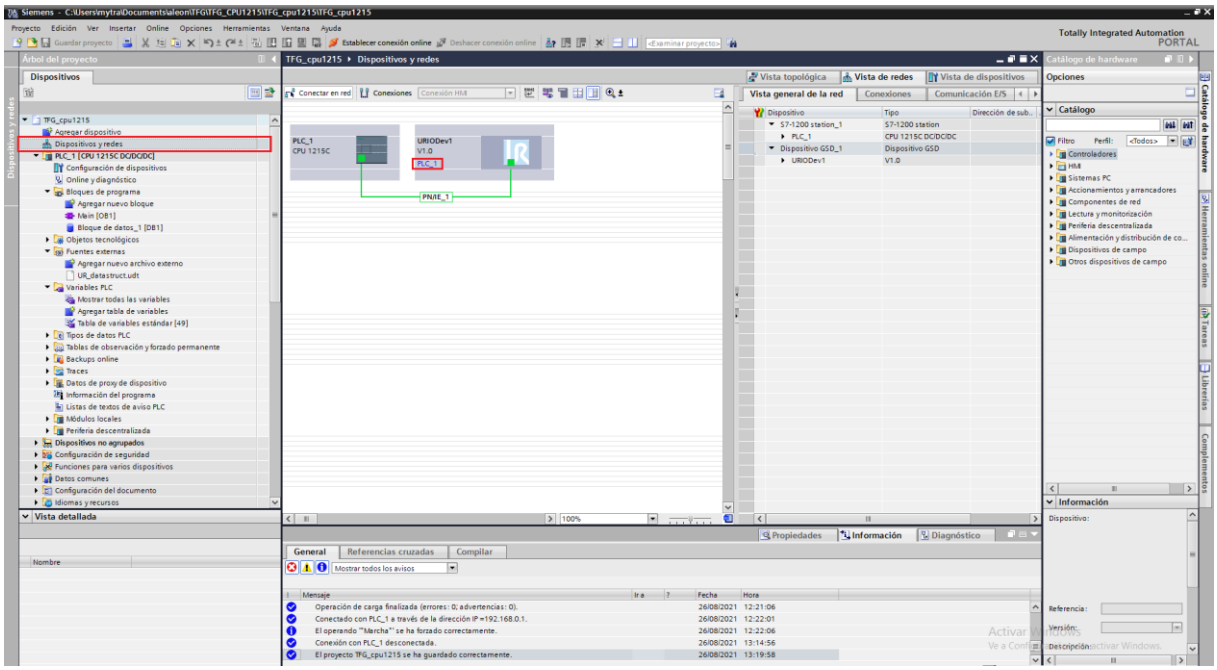

Figura 65. TIA Portal. Configuración de proyecto - Seleccionar controlador

7.2.1.3.6. Importación de datos UDT del UR5e

Desde la página web de Universal Robots se puede descargar un archivo que contiene los datos UDT del robot que facilitan de manera significativa el entendimiento de la información que nos proporciona el UR5e.

Para importar el archivo “UR_datastruct.udt” en TIA Portal, hay que agregar un nuevo archivo externo desde **PLC_1 [CPU 1214C DC/DC/DC]>Fuentes externas** y, una vez agregado el fichero .udt, se hace clic derecho en él y se generan los bloques a partir del mismo tal y como se visualiza en la figura 66.

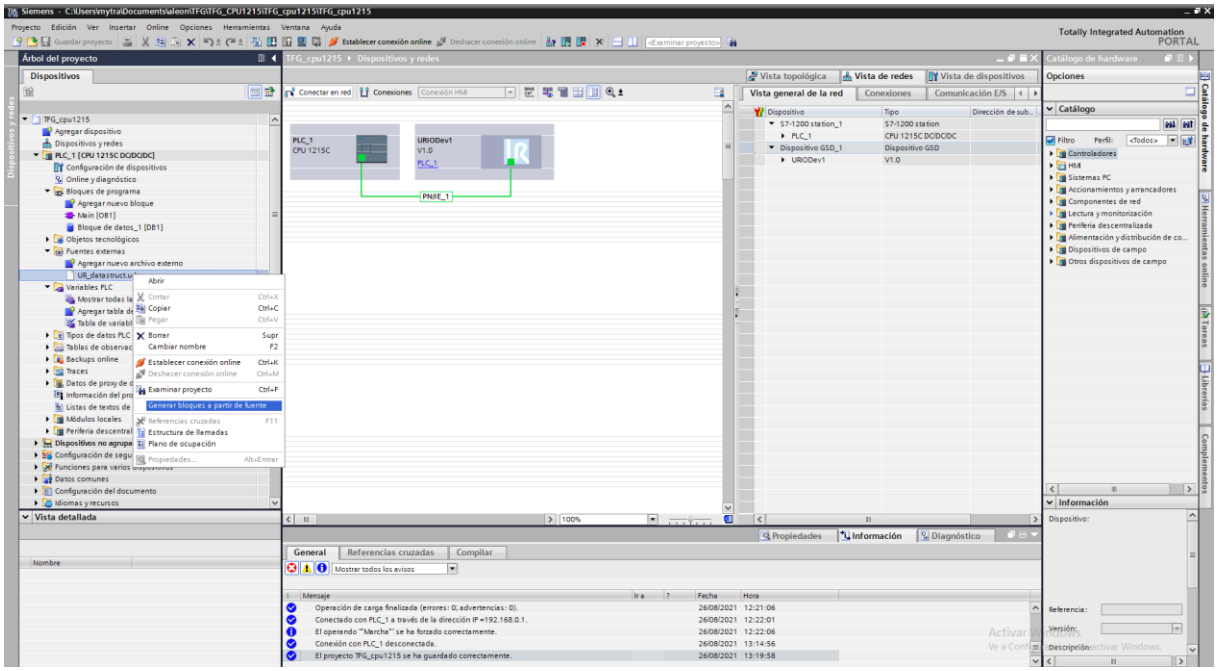

Figura 66. TIA Portal. Configuración de proyecto - Generar bloques de los tipos de dato de UR

Una vez realizado este paso, ya aparecen los bloques en los tipos de dato del PLC y se puede ir a las variables del PLC y asignar las variables que se requieran. Las variables de entrada y salida del robot para el presente proyecto son las siguientes:

- URI: Para los datos de entrada del PLC, que se corresponde con el tipo de datos “UR_T2O” y se le asigna la dirección %I0.0.
- URO: Para los datos de salida del PLC, que se corresponde con el tipo de datos “UR_O2T” y se le asigna la dirección %Q0.0.

En la figura 67 se muestra una lista de las señales de entrada y salida del robot disponibles.

	Nombre	Tipo de datos	Dirección	Rema...	Acces...	Escrib...	Visibl...	Comentario
1	URI	"UR_T2O"	%I0.0		☑	☑	☑	
2	State	UR_1_T2O_State	%I0.0		☐	☐	☐	
3	IO	UR_2_T2O_IO	%I32.0		☑	☑	☑	
4	Joints	UR_3_T2O_Joints	%I100.0		☑	☑	☑	
5	TCP	UR_4_T2O_TCP	%I204.0		☑	☑	☑	
6	Bits	UR_5_T2O_BitR...	%I280.0		☑	☑	☑	
7	Ints	UR_6_T2O_IntR...	%I288.0		☑	☑	☑	
8	Floats	UR_7_T2O_Floa...	%I384.0		☑	☑	☑	
9	URO	"UR_O2T"	%Q0.0		☑	☑	☑	
10	Robot IO	UR_8_O2T_Rob...	%Q0.0		☑	☑	☑	
11	Reg 1	UR_9_O2T_Regi...	%Q24.0		☑	☑	☑	
12	Reg 2	UR_9_O2T_Regi...	%Q124.0		☑	☑	☑	

Figura 67. TIA Portal. Configuración de proyecto - Tabla de variables del UR5e

7.2.1.4. Diseño del programa del autómata

Lo primero que se tiene en cuenta a la hora de diseñar el programa es que es necesario crear 2 variables:

- Una señal "Marcha" que sea la encargada de comunicar al robot el inicio de la aplicación de pick & place.
- Una señal de salida "Tomar_Foto", que se la encargada de comunicar a la cámara que tiene que realizar una fotografía.

En la figura 68 se muestran las variables anteriormente mencionadas:

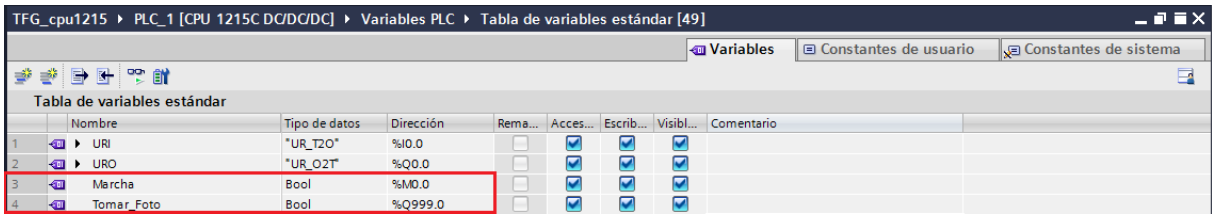

Figura 68. TIA Portal. Diseño de programa - Variables

Además de estas 2 variables, es necesario crear un bloque de datos con los siguientes 4 booleanos:

- "Tomar_Foto": Este dato se lee desde el módulo de Java y cuando tiene un valor TRUE se selecciona la imagen a procesar con AWS Rekognition.
- "Foto_Procesada": Este dato se actualiza desde el módulo de Java, poniéndose a TRUE cuando ha terminado el procesamiento de la imagen enviada a Rekognition.
- "Flag_Arandela": Este dato se actualiza desde el módulo de Java poniéndose a TRUE cuando Rekognition ha identificado una arandela en la fotografía de manera que permite al robot realizar los movimientos oportunos para la clasificación de la arandela.
- "Flag_Pinon": Este dato se actualiza desde el módulo de Java poniéndose a TRUE cuando Rekognition ha identificado un piñón en la fotografía de manera que permite al robot realizar los movimientos oportunos para la clasificación del piñón.

En la figura 69 se muestra la tabla de los 4 booleanos anteriormente descritos:

Figura 69. TIA Portal. Diseño de programa - Bloque de datos 1

Dado que el acceso a los bloques de datos desde Java se hace a nivel de byte, con la intención de simplificar lo máximo posible el módulo Java encargado de conectar el PLC con Rekognition, se crea un segundo DB donde se agrega el dato de "Marcha" que indique cuando se tiene que ejecutar el código correspondiente. En la figura 70 se puede ver la tabla correspondiente al DB2.

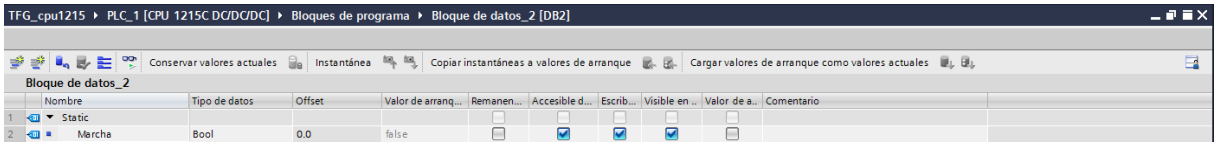

Figura 70. TIA Portal. Diseño de programa - Bloque de datos 2

Una vez creadas las variables y datos necesarios para el programa, se diseña el bloque de operación principal.

Lo primero que se tendrá en cuenta es que la señal de "Marcha" de tipo booleano con dirección %M0.0 debe comunicar al robot el inicio de la aplicación de pick & place que se desea realizar. A esta señal de marcha se le asigna una bobina que se corresponde con el bit 0 del registro de salidas del PLC hacia el robot UR5e, con dirección %Q24.0 y otra bobina que se corresponde con el dato de "Marcha", DB2.DBX0.0. De esta forma, en el programa del robot real, no comenzará el proceso hasta que en el bit 0 del registro de entradas booleanas tenga el valor TRUE, es decir, hasta que el PLC active la salida correspondiente. Este proceso queda reflejado a nivel visual en la figura 71.

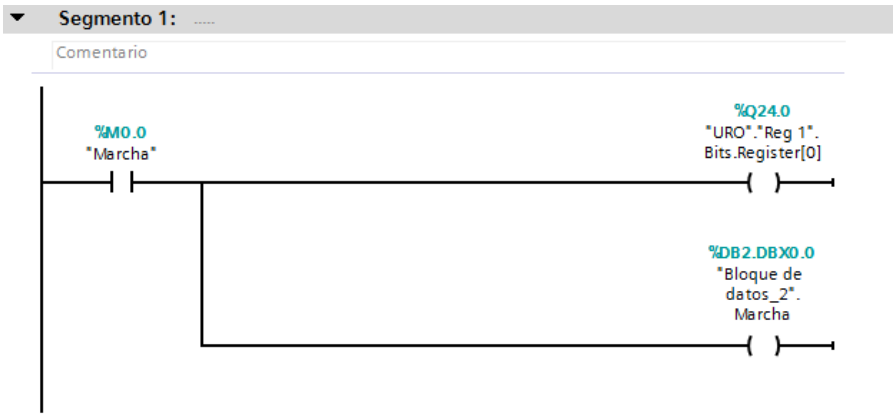

Figura 71. TIA Portal. Diseño de programa - Marcha

En segundo lugar, el autómata es el encargado de lanzar el disparo para tomar la foto. Esto lo hace activando la salida %Q999.0, que se corresponde con la variable "Tomar_Foto" en TIA Portal. Este disparo se realiza cuando hay un nivel alto en la entrada %I280.0, que se corresponde con el bit 0 del registro de salidas booleanas del robot, el cual se activará cuando el UR5e llega al punto de "Tomar_foto" como se muestra en el apartado del presente proyecto "*Desarrollo del programa real*". Además de activar la salida "Tomar_Foto" se actualiza a TRUE el valor del dato "Tomar_Foto" del bloque de datos creado.

En la figura 72 se muestra este proceso de manera gráfica.

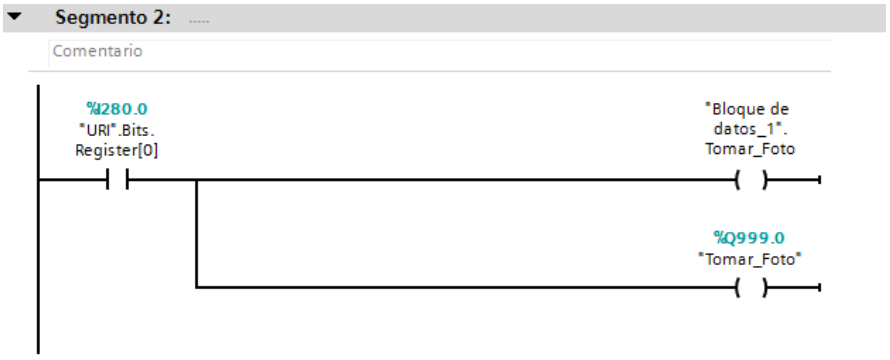

Figura 72. TIA Portal. Diseño de programa - Tomar foto

Por último, el PLC es el que se ocupa de comunicar al robot de cuando se ha realizado el procesamiento de la foto y de qué pieza se trata para que el brazo robótico realice los movimientos y operaciones oportunas en cada caso. Esta operación se realiza a través de los datos actualizados por el módulo de Java, “Foto_Procesada”, “Flag_Arandela” y “Flag_Pinon”

Teniendo en cuenta que la cinta transportadora por la que pasan las piezas a clasificar está programada para avanzar cuando el robot recoge la pieza o bien cuando pasan 30 segundos, para que el robot comience el movimiento de pick & place es necesario que esté a TRUE el dato “Foto_Procesada” y uno de los dos flags, “Flag_Arandela” o “Flag_Pinon”. De esta forma si el objeto que se ha fotografiado no se corresponde ni con una arandela ni con un piñón, Rekognition no ha podido identificar de qué pieza se trata o simplemente no ha llegado ninguna pieza al punto, sino que el final de carrera oportuno u otro tipo de sensor se ha activado por algún error externo el robot ignorará ese ciclo y esperará a que llegue la siguiente pieza.

La finalización de este procesamiento descrito se comunica al robot desde el bit 1 de su registro de entradas booleanas, que se corresponde con la salida del autómata con dirección %Q24.1 como se puede ver en la figura 73.

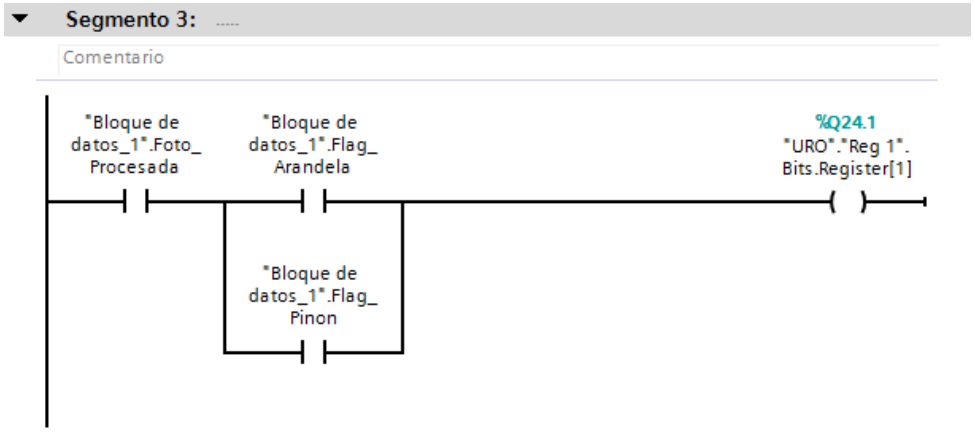

Figura 73. TIA Portal. Diseño de programa - Foto procesada

En el caso de que alguno de los dos flags se activen, simplemente se realizará el movimiento de clasificación correspondiente y se dará paso al siguiente ciclo de trabajo. Estos flags se comunican al robot desde los bits 2 y 3 de su registro de entradas booleanas para las arandelas y para los piñones respectivamente como se puede ver en la figura 74.

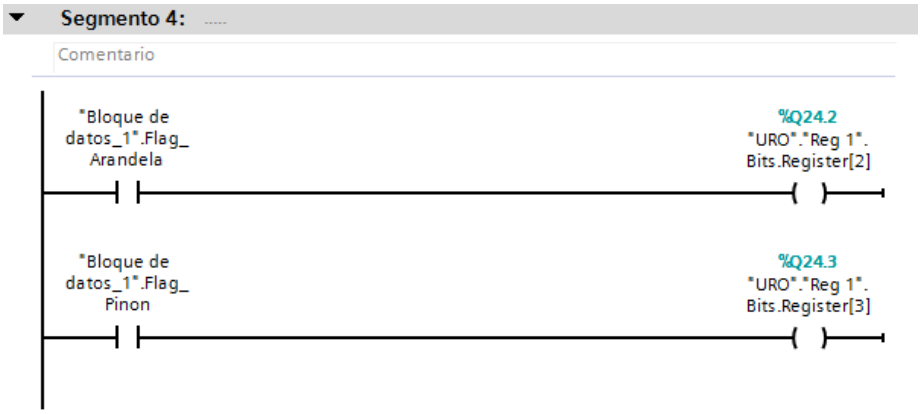

Figura 74. TIA Portal. Diseño de programa - Definición del movimiento de clasificación

En la figura 75 se puede visualizar un diagrama de flujo del programa de PLC con la finalidad de esclarecer el mismo.

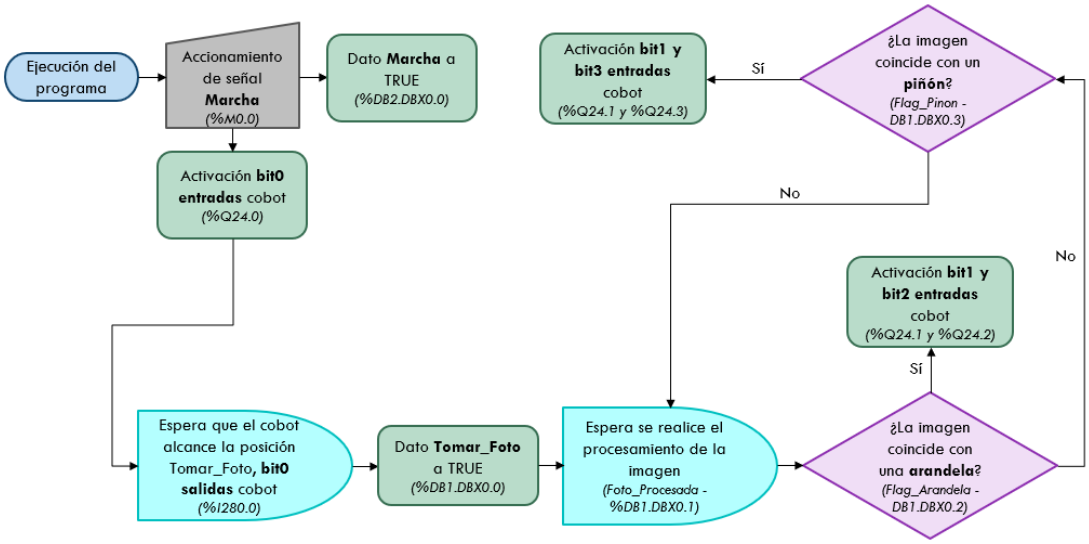

Figura 75. TIA Portal. Diseño del programa - Diagrama de flujo

7.2.1.5. Comunicación PLC - UR5e

Para establecer la comunicación entre el autómata y el robot hay que modificar una serie de configuraciones tanto en TIA Portal como en la interfaz gráfica de PolyScope.

En primer lugar, hay que poner ambos equipos en red. Para ello se asigna la dirección IP 192.168.1.8 al CPU 1214C y la dirección 192.168.1.9 al UR5e.

Para modificar la dirección IP del PLC se navega a la pantalla de “Dispositivos y redes” y se selecciona el puerto de PROFINET del autómata CPU 1214C. Con el puerto seleccionado y estando en la dirección **Propiedades>General>Direcciones Ethernet** se procede a la edición de la dirección IP como se observa en la figura 76.

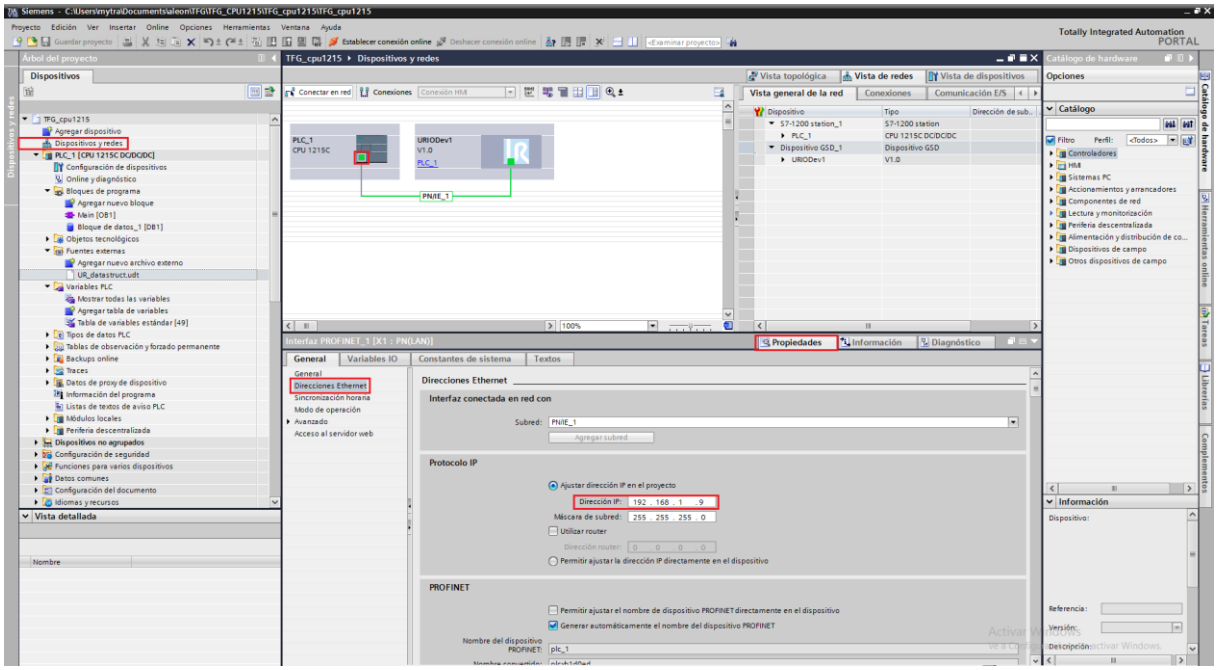

Figura 76. TIA Portal. Comunicación - Modificar IP

Para el caso del robot, de forma análoga, también se le asigna una IP dentro de la misma red en la que se encuentra el autómata, en este caso la 192.168.1.9.

Además, cuando el programa se carga en el PLC real, al robot se le debe asignar el nombre de dispositivo PROFINET que se tiene en TIA Portal.

Además de estos cambios que tienen lugar en TIA Portal, para que la comunicación tenga efecto, también se deben realizar las modificaciones y configuraciones a nivel de la interfaz gráfica del robot que se explican en el apartado 7.1.3.2.7. de este mismo documento.

7.3. Amazon Web Services

Amazon Web Services o AWS es un conjunto de servicios de computación en la nube pública y que todos juntos forman la plataforma de computación en la nube más amplia que existe. Presenta numerosos e interesantes servicios como pueden ser:

- EC2: Servidores virtuales en la nube
- DynamoDB: Base de datos NoSQL administrada
- AWS Lambda: Funciones de ejecución de código en la nube
- Rekognition: Analizador de imágenes y videos
- S3: Almacenamiento escalado

Este proyecto se centra en el servicio de Rekognition para analizar las fotografías tomadas por el aplicativo industrial y realizar las funciones de clasificación de acuerdo con el procesamiento llevado a cabo en la nube. Para ello, AWS CLI, la interfaz de línea de comandos de Amazon Web Services posee también un peso importante en el desarrollo del presente.

7.3.1. AWS CLI

La interfaz de línea de comandos de Amazon Web Services, AWS CLI, es una herramienta unificada que permite controlar todos los servicios de AWS, por lo que su instalación y configuración es esencial para el manejo y control del servicio de Rekognition, el cual es indispensable para la realización del proyecto.

La instalación y configuración de la AWS CLI se realiza de acuerdo con la guía de usuario [11] proporcionada por Amazon Web Services.

7.3.1.1. Instalación

Para el presente proyecto se instala la versión 2 de AWS CLI en un ordenador con sistema operativo Windows 10 de 64 bits.

Se instala la última versión de AWS CLI mediante el siguiente comando:

```
msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```

Se comprueba la correcta instalación de AWS CLI a través el comando:

```
aws --version
```

El cual devuelve la versión instalada en el ordenador como se muestra en la figura 77.

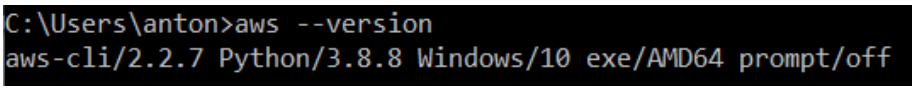

Figura 77. AWS CLI. Versión instalada

De esta forma se llega a la conclusión de que la instalación se ha ejecutado correctamente y se puede proceder con la configuración del AWS CLI.

7.3.1.2. Configuración

Para empezar con la configuración de AWS CLI se crean las claves de acceso desde la consola de administración de AWS, en la ruta **IAM>Usuarios** como se observa en la figura 78. Una vez en este punto se agrega un usuario administrador con acceso a AWS mediante programación y mediante la consola de administración de AWS. Además, es importante agregar este usuario a un grupo de usuarios que posea el permiso del tipo "AdministratorAccess". La creación de este grupo de usuarios se realiza desde **IAM>Grupos de usuarios**, tal y como se aprecia en la figura 79.

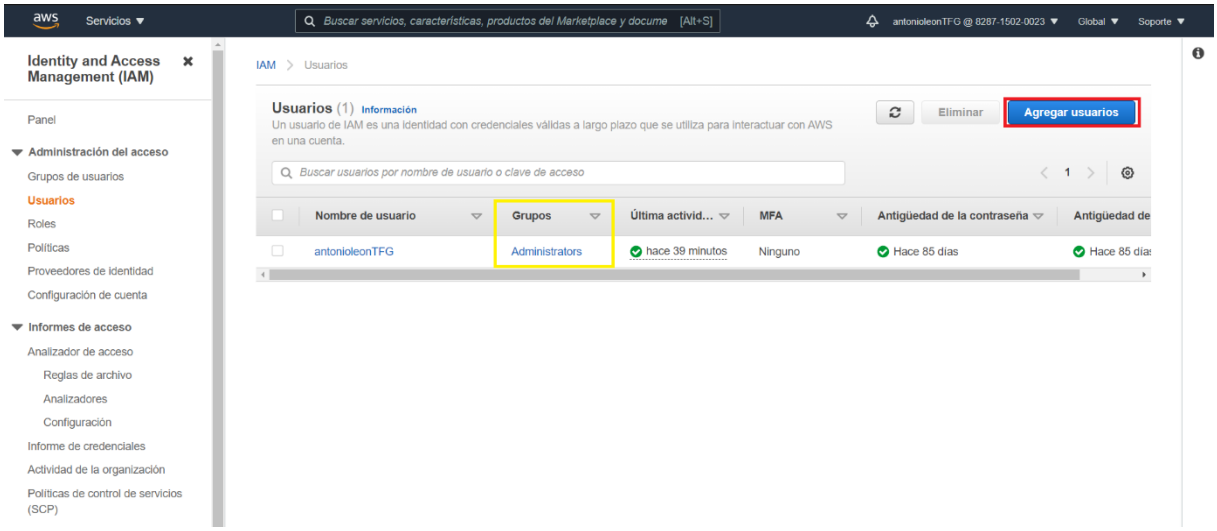

Figura 78. AWS IAM. Agregar usuario

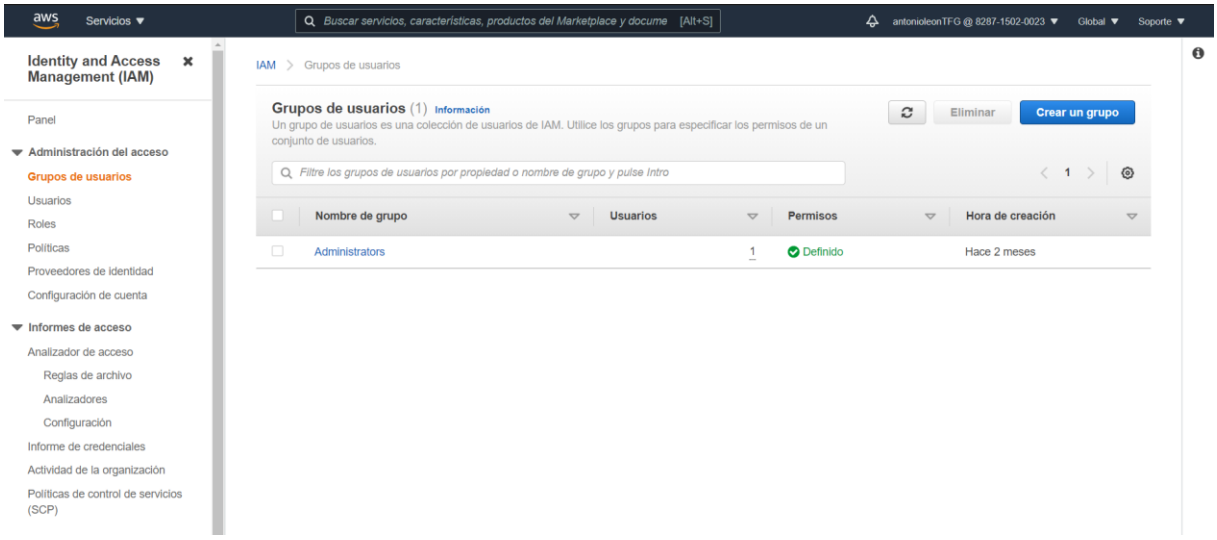

Figura 79. AWS IAM. Crear grupo de usuarios

Una vez creado el usuario y generadas sus claves de acceso, se crea un directorio .aws en la ruta **C:\Users\USERNAME** donde crear y guardar los archivos sin extensión "credentials" y "config" que contienen el "aws_access_key_id" y la "aws_secret_access_key" y la "region" y "output" respectivamente. A continuación, se muestran dos ejemplos de ambos archivos y el resultado de la ejecución del comando "aws configure" en la figura 80.

…/.aws/credentials:

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

…/.aws/config:

```
[default]
region=eu-central-1
output=json
```

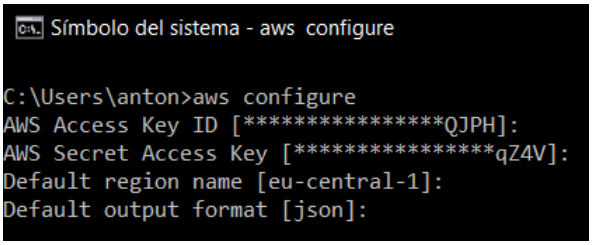

Figura 80. AWS CLI. aws configure

7.3.2. Amazon Rekognition

Amazon Rekognition es un servicio de AWS que facilita la integración de análisis de imágenes y videos a cualquier aplicación. En la aplicación desarrollada en el presente proyecto se utiliza para detectar la presencia en la imagen de dos tipos de piezas: arandelas o piñones. Además, Rekognition dispone de una API que facilita la integración con otras tecnologías.

Aunque Rekognition permite numerosos tipos de análisis como reconocimiento facial o descubrimiento de contenido ofensivo o inapropiado, para el presente proyecto se hace uso del tipo de análisis por etiquetas. Estas etiquetas se pueden referir a: objetos, eventos, conceptos o actividades. En este caso las etiquetas se van a referir siempre a objetos (arandelas o piñones).

La aplicación desarrollada para el envío de imágenes a la nube, su procesamiento y la posterior respuesta es desarrollada en el lenguaje Java, por lo que es necesario realizar la instalación y configuración del AWS SDK Java.

7.3.2.1. AWS SDK Java

Según la guía de desarrollador [12] facilitada por Amazon web Services, para instalar y configurar el AWS SDK para Java son necesarios los siguientes requisitos:

1. Tener creada una cuenta de AWS
2. Tener creado un usuario IAM
3. Instalar Java y Apache Maven
4. Configurar las credenciales

Los puntos 1, 2 y 4 ya se han explicado en el apartado anterior, por lo que ahora se explica cómo instalar Java y Apache Maven para proseguir después con la creación del proyecto.

7.3.2.1.2. Instalación de Java

Para instalar Java se navega a la página web de Oracle, https://www.oracle.com/java/technologies/javase-downloads.html y se descarga y ejecuta el instalador de la versión que se requiera. Para la realización del presente proyecto se opta por la instalación de la versión 11.0.11 (OpenJDK11U-jdk_x64_windows_hotspot_11.0.11_9) , como se muestra en la figura 81.

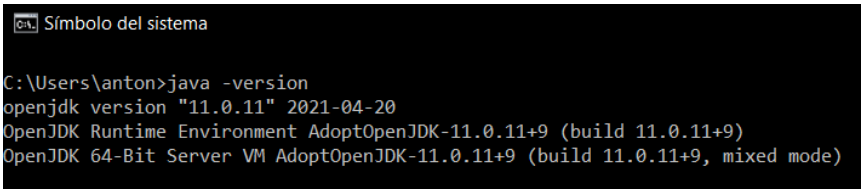

Figura 81. Versión de Java

7.3.2.1.3. Instalación de Apache Maven

De manera análoga a lo descrito en el apartado de Java, para la instalación de Apache Maven también se encuentran los instaladores de las distintas versiones disponibles en la página web del desarrollador, https://maven.apache.org/. Para este la realización de este proyecto se utiliza la versión 3.8.1 como se muestra en la figura 82.

Figura 82. Versión de Apache Maven

7.3.2.1.4. Creación del proyecto

Para crear el proyecto de Maven se sigue la guía de desarrollador para AWS SDK Java [12]. Se abre un terminal, se navega hasta la ruta en la que se quiere guardar el proyecto y se escribe el siguiente comando en el cmd:

```
mvn -B archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -
DgroupId=com.tfg.myapp -DartifactId=myapp
```

En la figura 83 se muestra el proceso de creación del proyecto Maven correspondiente al presente.

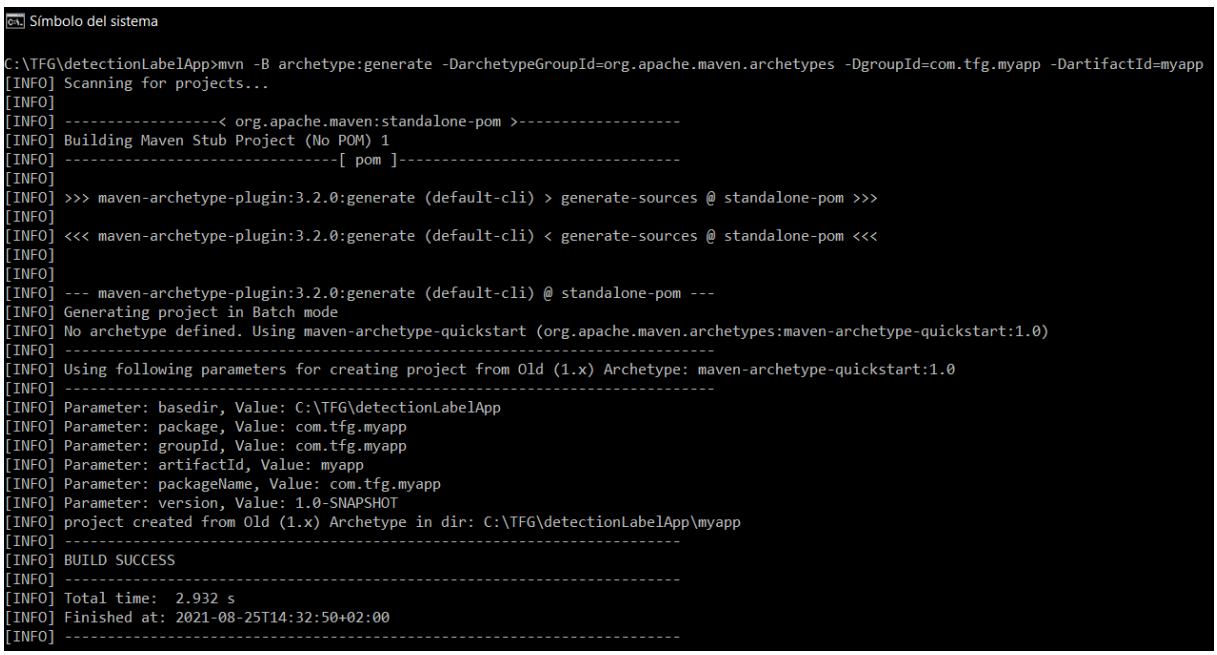

Figura 83. Apache Maven. Creación de proyecto

De esta manera se tiene, en la ruta especificada, creado el proyecto base de Java para su edición y desarrollo del aplicativo del presente proyecto.

7.3.2.2. Desarrollo de la aplicación

Para el desarrollo de la aplicación Java, se pone el foco en dos archivos principales: pom.xml y App.java.

El archivo pom.xml es el modelo de objetos del proyecto, del inglés Project Object Model. Es la unidad fundamental de trabajo en Maven. Es un archivo XML que contiene información del proyecto y todos los detalles de configuración para la construcción de este. Contiene valores predeterminados para la mayoría de los proyectos, como puede ser el directorio de compilación (que es target) o el directorio de origen (que es src/main/java). Pero además de estos valores predeterminados, en el POM se especifican otras configuraciones del proyecto como pueden ser sus dependencias, los perfiles de compilación, etc. También se puede especificar otra información como la descripción, la versión del proyecto, etc.

El archivo App.java es el que contiene el programa principal, en él se escribe la lógica, procesos y eventos que tienen lugar en la aplicación.

7.3.2.2.1. Archivo POM

En el archivo POM se encuentran las siguientes dependencias:

- BOM, necesario para realizar las administraciones de dependencias para clientes java individuales, https://mvnrepository.com/artifact/software.amazon.awssdk/bom/2.15.15.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>bom</artifactId>
  <version>2.15.15</version>
  <type>pom</type>
  <scope>import</scope>
</dependency>
```

- JUnit, que es marco de pruebas unitarias para proyectos y aplicaciones Java, https://mvnrepository.com/artifact/junit/junit/3.8.1.

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
</dependency>
```

- AWS Java SDK para Rekognition, que contiene las clases de cliente que se utilizan para la comunicación con Amazon Rekognition, https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-rekognition/1.12.8

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-java-sdk-rekognition</artifactId>
```

```
    <version>1.12.8</version>
  </dependency>
```

- Jackson Databind, que es una funcionalidad general de enlace de datos, https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.12.3

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.12.3</version>
</dependency>
```

- JSON, que es un paquete que permite la codificación y decodificación de JSONs en Java, https://mvnrepository.com/artifact/org.json/json/20210307. Además, también permite, por ejemplo, a conversión entre JSON y XML.
  JSON es un formato de intercambio de datos ligero y que no tiene dependencia del idioma y en este proyecto se utiliza para trabajar con la respuesta que envía Rekognition tras el procesamiento de una imagen.

```
<dependency>
  <groupId>org.json</groupId>
  <artifactId>json</artifactId>
  <version>20210307</version>
</dependency>
```

- S7 Connector, que es un conector S7 que permite la comunicación con el PLC de SIEMENS, https://mvnrepository.com/artifact/com.github.s7connector/s7connector/2.1.

```
<dependency>
  <groupId>com.github.s7connector</groupId>
  <artifactId>s7connector</artifactId>
  <version>2.1</version>
</dependency>
```

Además de todas estas dependencias también se requiere del plugin del compilador de Maven, de manera que permita la compilación del programa para su posterior prueba, https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-compiler-plugin/3.8.1.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>8</source>
        <target>8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

7.3.2.2.2. App.java

En la aplicación principal en primera instancia se encuentra el "package" mediante el cual se indica que el módulo de código que se encuentra en el archivo "App.java" pertenece al paquete "com.tfg.myapp".

```
package com.tfg.myapp;
```

Después se encuentra la importación de dependencias necesarias para el correcto funcionamiento de la clase principal "App" y que son las siguientes:

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.util.IOUtils;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.List;
import org.json.JSONArray;
import com.github.s7connector.api.DaveArea;
import com.github.s7connector.api.S7Connector;
import com.github.s7connector.api.factory.S7ConnectorFactory;
```

Después de toda la importación de dependencias se encuentra la clase principal "App" en la que lo primero que se halla es la declaración de un conjunto de palabras clave que se corresponden con colores, los cuales se usan para hacer más visual y eficiente la lectura de los logs que se sacan por consola durante la ejecución de la aplicación.

```
public static final String ANSI_RESET = "\u001B[0m";
public static final String ANSI_BLACK = "\u001B[30m";
public static final String ANSI_RED = "\u001B[31m";
public static final String ANSI_GREEN = "\u001B[32m";
public static final String ANSI_YELLOW = "\u001B[33m";
public static final String ANSI_BLUE = "\u001B[34m";
public static final String ANSI_PURPLE = "\u001B[35m";
public static final String ANSI_CYAN = "\u001B[36m";
public static final String ANSI_WHITE = "\u001B[37m";
```

Una vez han sido declarados los colores ya se halla el programa principal "main" en el que lo primero que se encuentra es una declaración de variables que se usarán a lo largo del aplicativo:

```
boolean flagGear = false; //Para la detección de un piñón
boolean flagWasher = false; //Para la detección de una arandela
float confidence_id = 0; //Variable que indicará el nivel de confianza de los tags
detectados en la imagen procesada
byte DB1 = 0x0; //Byte que va a permitir evaluar el DB1DBX0
```

```
byte marcha = 0x0; //Byte que va a permitir evaluar el DB2DBX0 para la señal de
marcha
```

Cuando ya han sido declaradas las variables, comienza el bucle do-while que va a permitir ejecutar la aplicación hasta que la señal de marcha del PLC pase a tener un valor de FALSE.

Lo primero que se hace en el bucle do-while es crear el conector de S7 para la comunicación con el PLC. Toda la información relativa a la librería S7Connector se ha extraído de la documentación, disponible en https://s7connector.github.io/s7connector/.

```
S7Connector connector = S7ConnectorFactory
.buildTCPConnector()
.withHost("192.168.1.9")
.withRack(0)
.withSlot(1)
.build();
```

El siguiente fragmento del programa permite esperar a que el PLC indique que el robot ha alcanzado la posición de "Tomar_Foto" para, en este caso, seleccionar la fotografía a procesar.

```
while(((DB1) & 0x1) != 0x1){ //Esperamos hasta que el DB1DBX0.0 esté a TRUE
  DB1 = connector.read(DaveArea.DB, 1, 1, 0); //Se lee el primer byte del DB1
(DB1.DBX0)
}
```

Para entender la función anterior, hay que tener en cuenta que el acceso al bloque de datos DB1 del PLC tiene que ser a nivel de byte y que el dato "Tomar_Foto" se encuentra en el bit 0 del byte 0 de este DB.

Posteriormente, dado que el presente proyecto no se centra en la cámara que toma las fotografías, se encuentra un fragmento de código que permite proporcionar una imagen aleatoria de un directorio definido en el equipo. En este caso, se escoge una imagen aleatoria de la ruta C:\TFG\Images.

```
List<String> images = new ArrayList<String>();
File directory = new File("C:\\TFG\\Images");
File[] files = directory.listFiles();
for (File f : files) {
  images.add(f.getName());
}
int countImages = images.size();
int imageNumber = (int) (Math.random() * countImages);
String image = images.get(imageNumber);
String photo = "C:\\TFG\\Images\\" + image;
```

A continuación, se crea el buffer de bytes y el correspondiente canal de entrada a la imagen a procesar.

```
ByteBuffer imageBytes;
try (InputStream inputStream = new FileInputStream(new File(photo))) {
  imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
}
```

En el siguiente paso se crea el constructor del cliente de Rekognition y se configuran los parámetros para la detección de etiquetas. En la configuración elegida se define:

- La imagen por procesar, que será la seleccionada de manera aleatoria en el fragmento de programa ya analizado.
- Un máximo de 5 etiquetas para el procesamiento de manera que se otorgue velocidad a la función.
- Un nivel mínimo de confianza del 90%, esto es que únicamente aparezcan aquellas etiquetas de las que Rekognition tenga la certeza de acertar en un 90%.

```
AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();
DetectLabelsRequest request = new DetectLabelsRequest()
  .withImage(new Image().withBytes(imageBytes))
  .withMaxLabels(5)
  .withMinConfidence(90F);
```

A continuación, se halla el "try" en el que se encuentra la evaluación de la respuesta de Rekognition. Lo primero que se ve es un fragmento de código en el que primero se guardan las etiquetas de la respuesta en una lista para posteriormente transformarla en un array de JSON, que permita la correcta evaluación de la respuesta.

```
DetectLabelsResult result = rekognitionClient.detectLabels(request);
List<Label> labels = result.getLabels();
ObjectMapper objectMapper = new ObjectMapper();
String jsonString = objectMapper
  .writerWithDefaultPrettyPrinter()
  .writeValueAsString(labels);
JSONArray arr = new JSONArray(jsonString);
```

Después, se halla el bucle "for" en el que se procesa el JSON de la respuesta y se pasan a TRUE "flagWasher" o "flagGear" en función de si en la respuesta se ha encontrado o no la etiqueta correspondiente.

```
for (int i = 0; i < arr.length(); i++) {
  String name_id = arr.getJSONObject(i).getString("name");
  confidence_id = (float) arr.getJSONObject(i).getDouble("confidence");
  if (name_id.equals("Gear")) {
     flagGear = true;
     flagWasher = false;
     break;
  } else if (name_id.equals("Washer")) {
    flagWasher = true;
    flagGear = false;
    break;
  } else {
    flagGear = false;
    flagWasher = false;
  }
}
```

En el fragmento de código anterior, se declara la variable “confidence_id“, esta únicamente se utiliza para mostrar por la pantalla de la consola el nivel de confianza de la etiqueta de arandela o de piñón al finalizar la aplicación.

A continuación, aparece la función if/else en la que se indica qué hacer en cada uno de los casos posibles.

En el caso de que el flag para el piñón tome el valor TRUE y el de arandela el valor FALSE, se escribe un TRUE en los bits DB1.DBX0.1 y DB1.DBX0.3 (o lo que es lo mismo 0xA). De esta forma se indica al PLC que la foto ha sido procesada y que se la pieza se trata de un piñón para que este comunique al UR5e de que proceda con la clasificación.

```
if (flagGear == true && flagWasher == false) {
  connector.write(DaveArea.DB, 1, 0, 0xA);
}
```

Para el caso de que sea el flag de arandela el que tome el valor de TRUE y el de piñón un valor FALSE, el proceso es exactamente igual, solo que esta vez se pone a TRUE el DB1.DB0.2 en lugar del DB1.DB0.3.

```
else if (flagWasher == true && flagGear == false) {
  connector.write(DaveArea.DB, 1, 0, 0x6);
}
```

En cualquier, no se hace nada, únicamente se sacará un log por consola que indique que la imagen procesada no se corresponde con ninguna de las piezas esperadas.

```
else if System.out.println(
  "\nSe ha analizado la imagen " +
  ANSI_PURPLE +
  image +
  ANSI_RESET +
  " y se ha determinado que no corresponde ni a un engranaje ni a una arandela.\n"
);
```

La última sentencia del “try” del procesamiento es el cierre de la comunicación con Rekognition.

```
((AmazonRekognition) rekognitionClient).shutdown();
```

Después, se puede encontrar el “catch” correspondiente al “try” del procesamiento, la lectura de la señal de marcha para ver si el bucle se tiene que seguir ejecutando y el cierre de la conexión del conector S7.

```
marcha = connector.read(DaveArea.DB, 2, 1, 0);
catch (AmazonRekognitionException e) {
  e.printStackTrace();
}
connector.close();
```

Por último, se encuentra el “while” correspondiente al do-while que evaluará si se continua con la ejecución del bloque o no.

```
while(marcha == 0x1);
```

En la figura 84 se puede apreciar el diagrama de flujo de la aplicación que selecciona una imagen de manera aleatoria y realiza la comunicación con AWS Rekognition para su procesamiento y posterior comunicación con el PLC.

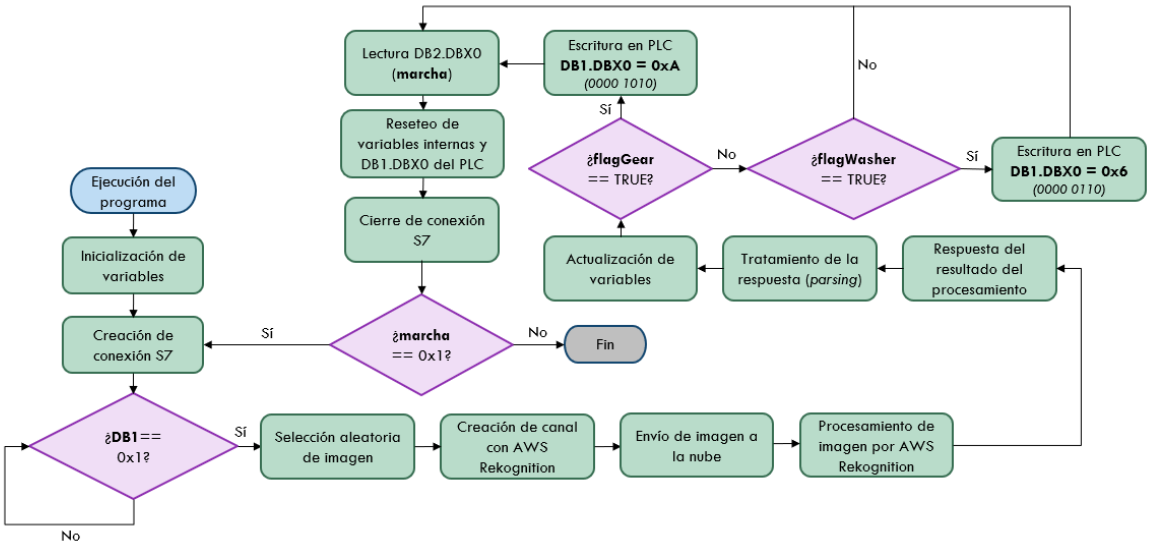

Figura 84. Aplicación Java - Diagrama de flujo

7.3.2.3. Compilación y ejecución de la aplicación

Para compilar la aplicación de Java, basta con abrir un terminal en el directorio donde se encuentre el archivo pom.xml y ejecutar el siguiente comando:

```
mvn package
```

En la figura 85 se muestra el resultado de la compilación del proyecto Java relativo al presente trabajo.

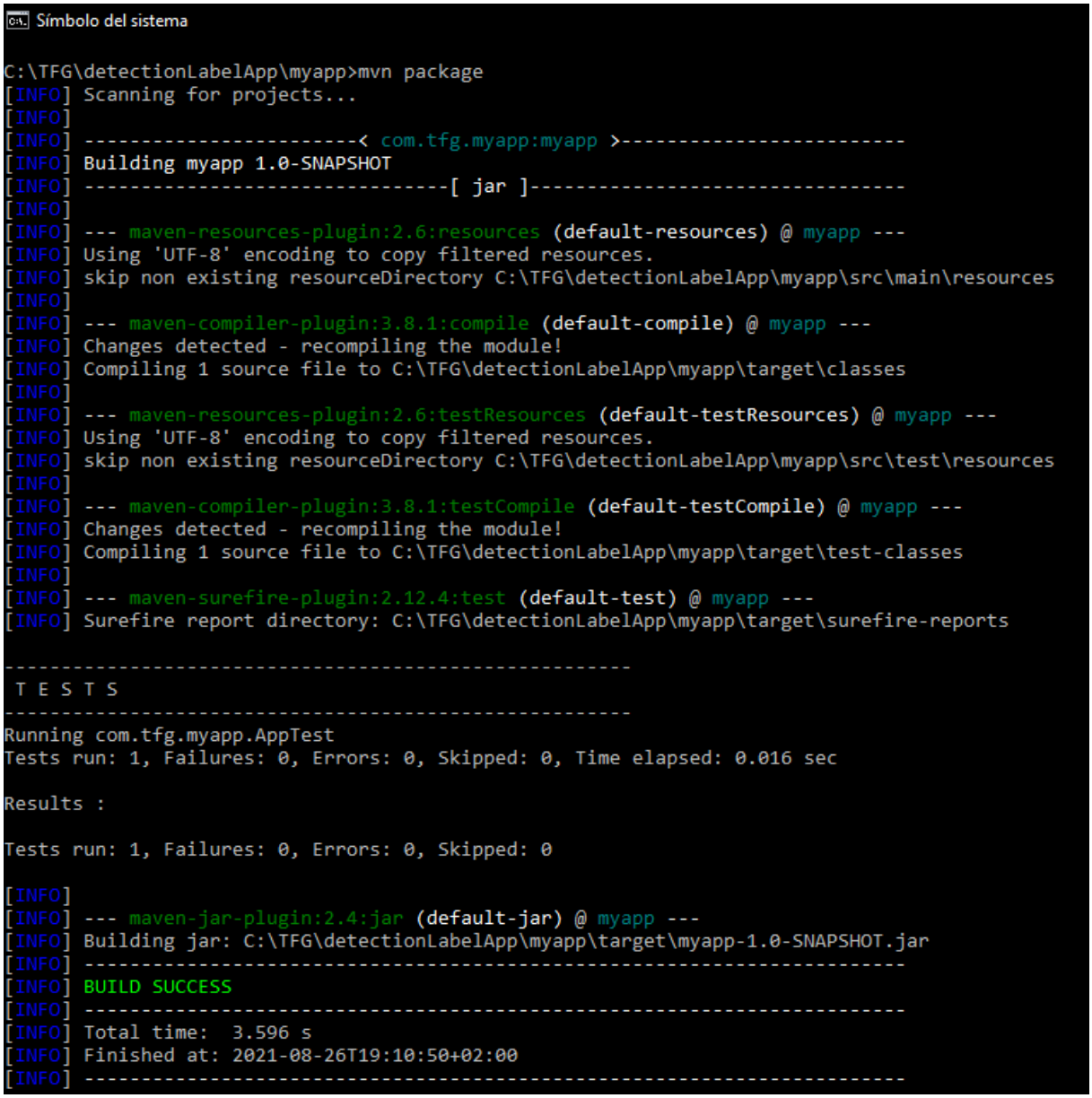

Figura 85. Apache Maven. Compilación del programa

Una vez compilado se puede ejecutar el programa a través del comando:

```
mvn exec:java -Dexec.mainClass="com.tfg.myapp.App"
```

En la figura 86 se puede visualizar el resultado de la ejecución del programa pertinente.

```
C:\TFG\detectionLabelApp\myapp>mvn exec:java -Dexec.mainClass="com.tfg.myapp.App"
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.tfg.myapp:myapp >-------------------------
[INFO] Building myapp 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ myapp ---

Se ha analizado la imagen pinon1.jpg y se ha determinado que corresponde a un ENGRANAJE con una confianza del 95.71753%

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.637 s
[INFO] Finished at: 2021-08-26T19:17:08+02:00
[INFO] ------------------------------------------------------------------------
```

Figura 86. Apache Maven. Ejecución de un ciclo de trabajo

7.4. Resultado final

En el presente apartado se describirá y visualizará el resultado final de un ciclo de trabajo de la operación de pick & place llevada a cabo en el proyecto.

Al comienzo de la aplicación el robot se encuentra en reposo esperando a que el PLC de la orden de inicio como se aprecia en la figura 87.

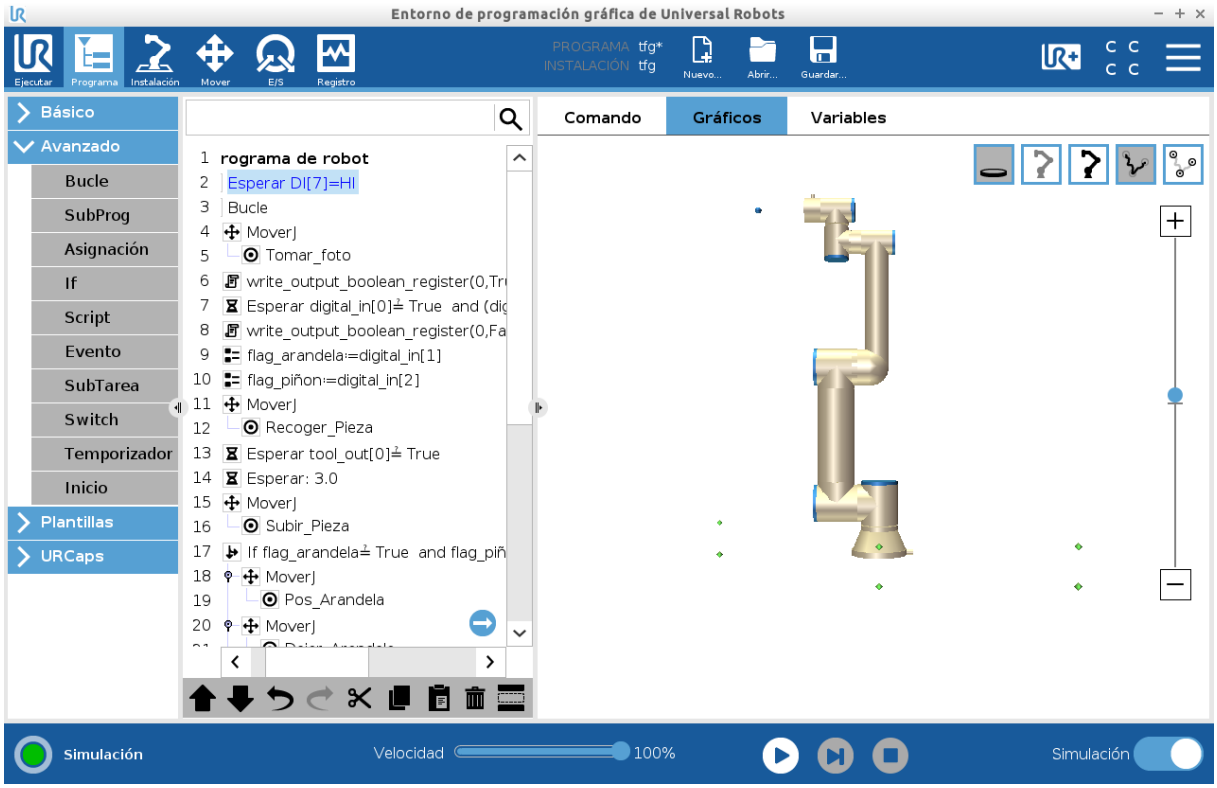

Figura 87. Resultado Final. UR5e en reposo

Una vez que el autómata ha ordenado el inicio de la aplicación, el UR5e se mueve a la posición ”Tomar_Foto” tal y como se puede ver en las figuras 88 y 89 respectivamente.

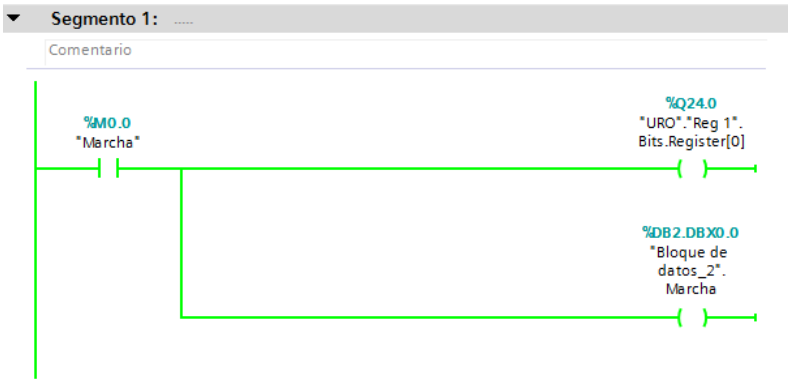

Figura 88. Resultado final. Aplicación en marcha

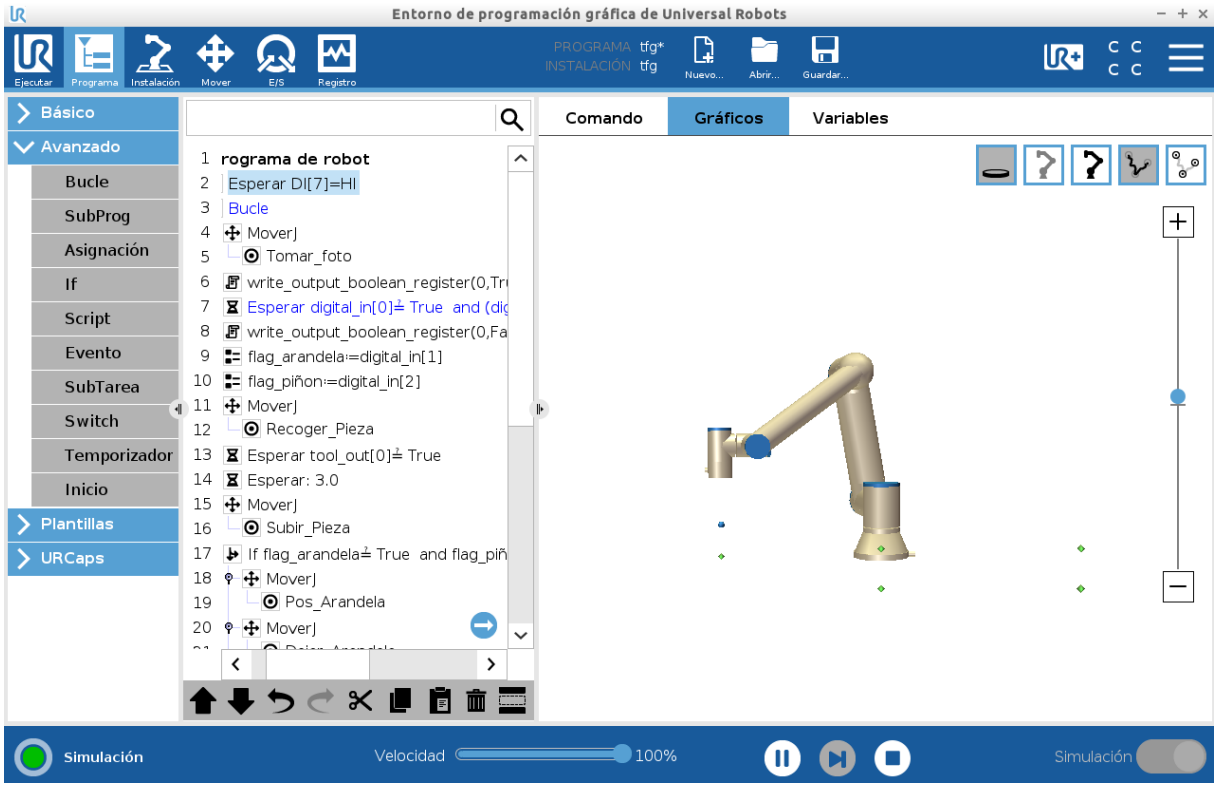

Figura 89. Resultado final. UR5e en posición "Tomar_Foto"

Al alcanzar la posición “Tomar_Foto” el bit 0 del registro de salidas toma el valor TRUE, lo cual hace que el PLC ordene la realización de la foto oportuna tal y como se puede ver en la figura 90.

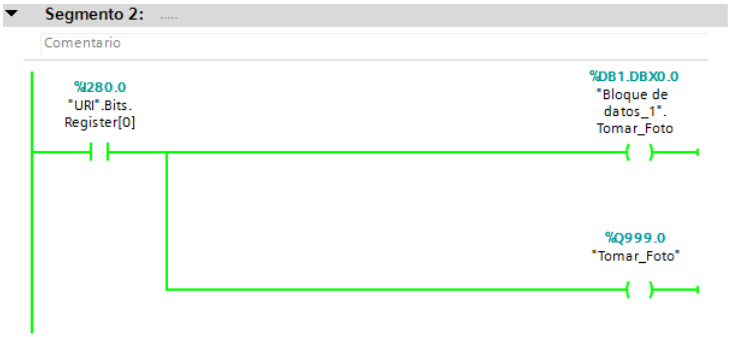

Figura 90. Resultado final. Toma de foto

Esto hace que se realice la fotografía oportuna, figura 91, y que se ejecute el módulo de Java para el análisis y procesamiento de esta.

Figura 91. Resultado final. Foto que procesar

Como se puede apreciar en el log de la figura 92, el procesamiento ha determinado que la pieza se trata de una arandela.

```
Se ha analizado la imagen arandela1.jpg y se ha determinado que corresponde a una ARANDELA con una confianza del 99.999954%
```

Figura 92. Resultado final. Log del procesamiento

Esto implica que el módulo de Java haya escrito en las direcciones correspondientes del PLC un valor TRUE tanto para indicar que se ha realizado el procesamiento de la imagen como para indicar que la pieza detectada se trata de una arandela. Esto, a su vez, desencadena en una escritura por parte del autómata de un valor TRUE en los bits 1 y 2 del registro de entradas booleanas del robot tal y como se muestra en las figuras 93 y 94.

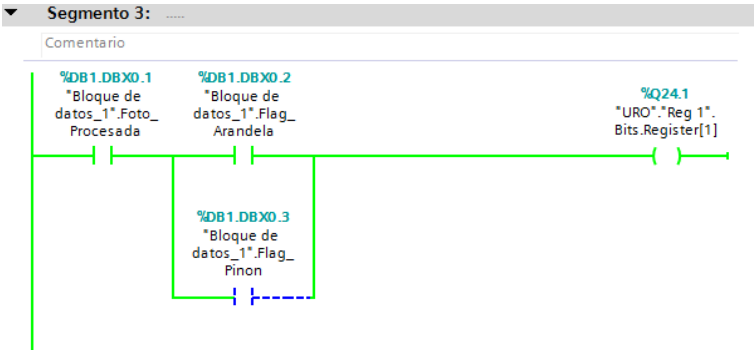

Figura 93. Resultado final. Comunicación de foto procesada a UR5e

Figura 94. Resultado final. Orden de clasificación de arandela al UR5e

El UR5e al ser conocedor de que la foto ha sido procesada y que se trata de una arandela en primer lugar recoge la pieza posicionándose en el punto "Recoger_Pieza" como se muestra en la figura 95 y, en el caso real, cerrando la pinza mediante el comando "rq_move_and_wait_norm(90)".

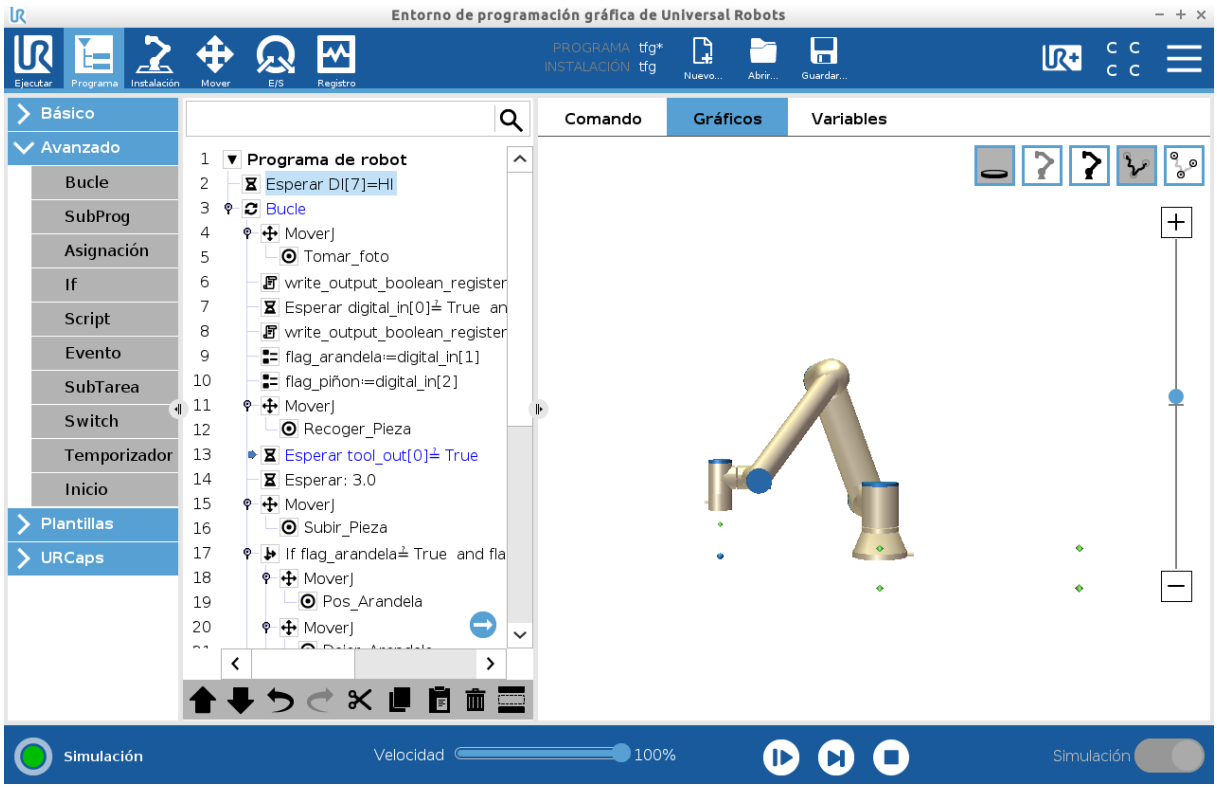

Figura 95. Resultado final. UR5e recogiendo arandela

Una vez que se ha completado la recogida de la arandela el robot realiza el movimiento de disposición de esta en su punto de clasificación "Dejar_Arandela" como se observa en la figura 96 y, en el caso real, abrir la herramienta de pinza mediante el comando "rq_open_and_wait()".

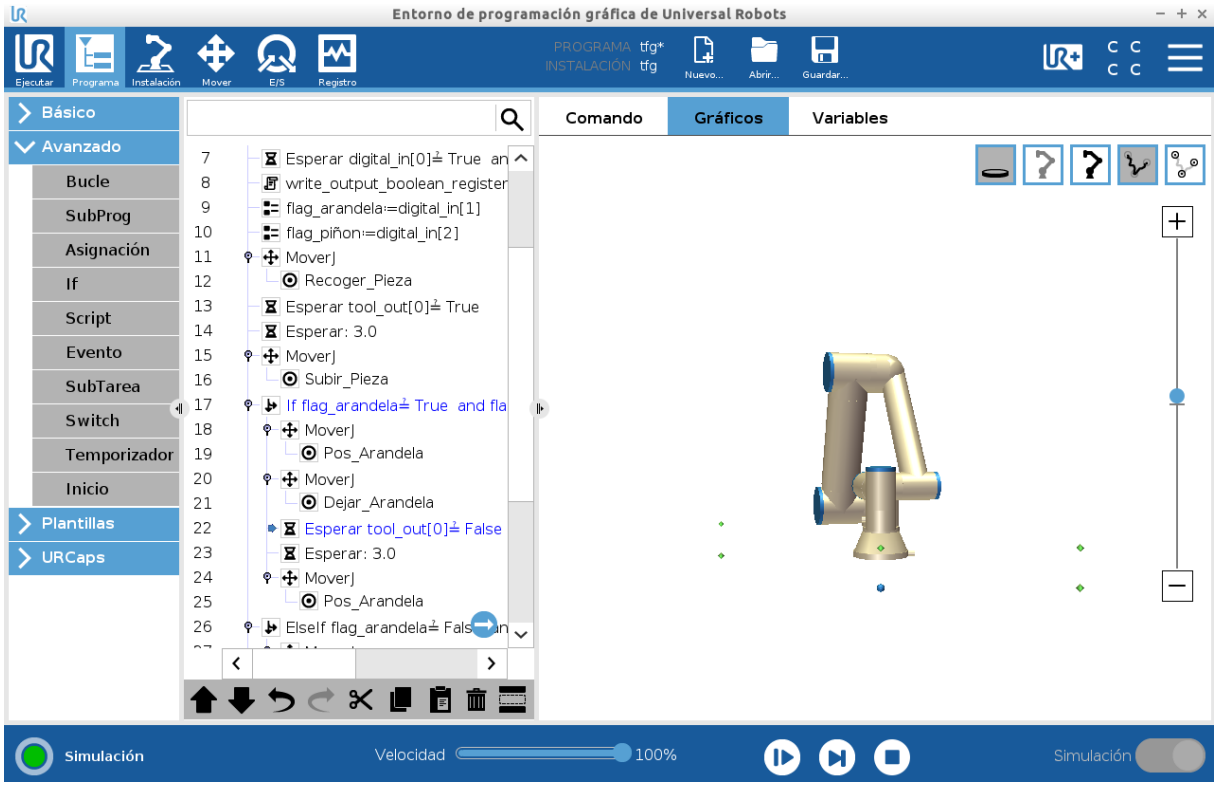

Figura 96. Resultado final. Disposición de arandela en punto final

Una vez que el robot ha soltado la arandela, este vuelve a la posición “Tomar_Foto” para repetir el ciclo de trabajo con la siguiente pieza tal y como se ve en la figura 97.

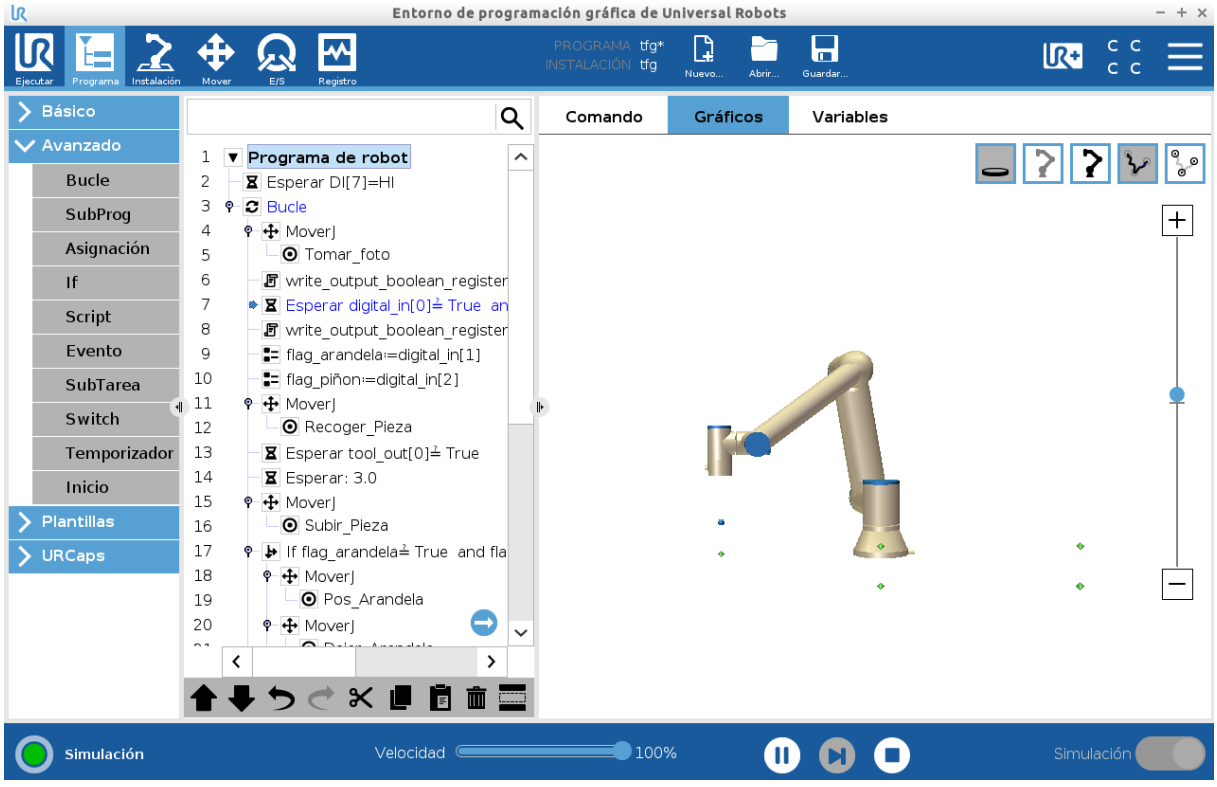

Figura 97. Resultado final. Comienzo de nuevo ciclo

8. Conclusión y trabajo futuro

Tanto las tecnologías industriales como las de telecomunicación y las de la información disponen de numerosas y amplias posibilidades de integración.

El presente proyecto ha permitido demostrar la sencilla e intuitiva implementación de aplicaciones en robots colaborativos de manera que ofrecen un amplio abanico de posibilidades a soluciones industriales tanto en grandes como en pequeñas y medianas empresas.

También se ha puesto en manifiesto la facilidad para integrar servicios en la nube con aplicaciones industriales gracias a las SDK de AWS. En este proyecto únicamente se ha interactuado con un servicio de Amazon Web Services, Rekognition, pero existen multitud de servicios que podrían ser de gran utilidad para otros proyectos industriales como por ejemplos aquellos que abarquen Machine Learning o aquellos que requieran de un gran almacenamiento de datos.

Tras la realización de este trabajo de fin de grado, se pueden definir diferentes líneas de trabajo:

- Realización de caso práctico que permita comprobar la correcta configuración de red y que posibilite la verificación de las comunicaciones.
- Estudio de posibilidad de integración con hardware de otros fabricantes.
- Implementación de otras aplicaciones industriales que permitan o requieran la incorporación a las mismas de otros servicios de AWS.
- Comparativa de una misma aplicación industrial a distintas plataformas en la nube como podrían ser AWS, Google Cloud y Microsoft Azure.

9. Presupuesto

9.1. Hardware

EQUIPO	PRECIO	DURACIÓN	Coef. de amortización	TOTAL
Asus ZenBook UX430UA	864€	4 años	0,083	72€
Total gastos por hardware				**72€**

9.2. Licencias

LICENCIA	PRECIO	DURACIÓN	Coef. de amortización	TOTAL
- OFFLINE SIMULATOR – E-SERIES – UR SIM FOR NON LINUX 5.10.2	0€	Indefinida	---	0€
- 3 x SIMATIC STEP 7 incl. Safety and WinCC V16 TRIAL Download	0€	21 días	---	0€
- Word Office	69€	1 año	0,333	23€
Total gastos por licencias				**23€**

9.3. Costos de AWS

Para los costes de Amazon Rekognition se debe tener en cuenta que el límite de uso de capa gratuita es de 5000 imágenes al mes y que los precios estipulados a partir de dicha capa son los siguientes:

TIPO DE COSTE	PRECIO UNITARIO
Primer millón de imágenes procesadas al mes	0,0012USD
Siguientes 9 millones de imágenes procesadas al mes	0,00096USD
Siguientes 90 millones de imágenes procesadas al mes	0,00072USD
Más de 100 millones de imágenes procesadas al mes	0,00048USD

SERVICIO	PRECIO UNITARIO	MES	USABILIDAD	TOTAL
Amazon Rekognition	0,0012USD	Junio	25	0€
Amazon Rekognition	0,0012USD	Julio	38	0€
Amazon Rekognition	0,0012USD	Agosto	17	0€
Total gastos por servicios de AWS				**0€**

9.4. Costes de mano de obra

TRABAJADOR	TAREA	COSTE de JORNADA	Nº de JORNADAS	TOTAL

Ingeniero	Planificación e investigación	240€	5	1200€
Ingeniero	Configuración de entornos	240€	10	2400€
Ingeniero	Implementación	240€	15	3600€
Ingeniero	Pruebas	240€	10	2400€
Ingeniero	Documentación y redacción	120€	10	1200€
Total gastos por personal				**10800€**

9.5. Coste de ejecución material

COSTE	**TOTAL**
Coste hardware	72€
Coste de licencias	23€
Costes de AWS	0€
Costes de mano de obra	10800€
Total coste de ejecución material	**10895€**

9.6. Gastos generales y beneficio industrial

COSTE	**% PEM**	**COSTE de JORNADA**
Gastos generales	15%	1634,25€
Beneficio industrial	6%	653,70€
Total costes por contrata		**2287,95€**

9.7. Presupuesto total

COSTE	**TOTAL**
Coste de ejecución material	10895€
Coste de ejecución por contrata	2287,95€
Total coste de ejecución material	**13182,95€**

10. Referencias

[1] Jeremy Likness, olprod (17 de abril de 2020). *Escenarios y casos de uso empresariales sin servidor*. https://docs.microsoft.com/es-es/dotnet/architecture/serverless/serverless-business-scenarios

[2] Amazon Web Services (2021). *Coca-Cola Andina Builds Data Lake on AWS, Increases Analytics Productivity by 80% for More Data-Driven Decision-Making*. AWS. https://aws.amazon.com/es/solutions/case-studies/coca-cola-andina-case-study/?did=cr_card&trk=cr_card

[3] Amazon Web Services (2021). *Amazon Robotics Uses Amazon SageMaker to Enable ML Inferencing at Scale*. AWS. https://aws.amazon.com/es/solutions/case-studies/amazon-robotics-case-study/?did=cr_card&trk=cr_card

[4] Jordi Pellegrí (11 de junio de 2020) *HABLAMOS SOBRE EL FABRICANTE ESPAÑOL MÁS "COBOTIZADO" EN EL NEGOCIO DE LA AUTOMOCIÓN*. The Universal Robots Blog. https://blog.universal-robots.com/es/hablamos-sobre-el-fabricante-espanol-mas-cobotizado-en-el-negocio-de-la-automocion

[5] Universal Robots (17 de enero de 2019). *DIEZ AÑOS TRANSFORMANDO LA AUTOMATIZACIÓN INDUSTRIAL*. The Universal Robots Blog. https://blog.universal-robots.com/es/diez-anos-universal-robots

[6] Universal Robots (2021). *UR5e User Manual*. Universal Robots. https://s3-eu-west-1.amazonaws.com/ur-support-site/115737/99404_UR5e_User_Manual_en_Global.pdf

[7] Robotiq, Inc. (2021). *Robotiq Hand-E for Universal Robots. Instruction Manual*. Robotiq. https://assets.robotiq.com/website-assets/support_documents/document/Hand-E_Manual_UniversalRobots_PDF_20210708.pdf

[8] Universal Robots (2020). *Guide: Setting up the virtual environment to simulate UR robots*. Universal Robots. https://academy.universal-robots.com/media/jiehhszc/ursim_vmoracle_installation_guidev03_en.pdf

[9] Universal Robots (2021). *The URScript Programming Language for e-Series*. Universal Robots. https://s3-eu-west-1.amazonaws.com/ur-support-site/115824/scriptManual_SW5.11.pdf

[10] SIEMENS, SIMATIC (11/2019). *STEP 7 and WinCC Engineering V16. System Manual.*

[11] Amazon Web Services (2021). *AWS Command Line Interface User Guide*. AWS. https://docs.aws.amazon.com/cli/latest/userguide/aws-cli.pdf#cli-chap-welcome

[12] Amazon Web Services (2021). *AWS SDK for Java Developer Guide*. AWS. https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/aws-sdk-java-dg-v2.pdf#get-started

Además de estas referencias, las cuales tienen una explícita versión a lo largo del presente documento, existen otras que han sido de gran ayuda para la investigación y para el desarrollo

del proyecto, aunque no se citen a lo largo de la memoria de manera concreta y que se enumeran a continuación:

[13] Amazon Web Services (2021). *AWS Lambda – Guía del desarrollador*. AWS. https://docs.aws.amazon.com/es_es/lambda/latest/dg/lambda-dg.pdf#welcome

[14] Amazon Web Services (2021). *Creación de aplicaciones con arquitecturas sin servidor*. AWS. https://aws.amazon.com/es/lambda/serverless-architectures-learn-more/

[15] Serverless Stack (2021). *¿Qué es Serverless?* Serverless Stack. https://serverless-stack.com/chapters/es/what-is-serverless.html

[16] César Gallego Rodríguez, Fernando Alvarez, Martin Evgeniev (2019). *Serverless*. https://www.bbva.com/es/serverless/

[17] Codster (2021). *5 ventajas de las aplicaciones serverless en cloud computing*. Codster. https://codster.io/blog/5-ventajas-de-las-aplicaciones-serverless-en-cloud-computing/

[18] Jordi Pellegrí (2020). *¿CUÁLES SON LAS DIFERENCIAS ENTRE UN COBOT Y UN ROBOT INDUSTRIAL?* The Universal Robots Blog. https://www.universal-robots.com/es/blog/diferencias-cobot-y-robot-industrial/

[19] Tecnología para los negocios (2021). *Cobots: ejemplos de tecnología que busca parecerse más a los humanos.* Tecnología para los negocios. https://ticnegocios.camaravalencia.com/servicios/tendencias/cobots-ejemplos-de-tecnologia-que-busca-parecerse-mas-a-los-humanos/