INSTITUTO TECNOLOGICO SUPERIOR DE SAN ANDRES TUXTLA

SIMULACIÓN DE SISTEMAS ROBÓTICOS

DR. JOSÉ ÁNGEL NIEVES VÁZQUEZ

INVESTIGACIÓN UNIDAD 3:

Operación y control de robots en un ambiente virtual.

- Espinosa Cruz Shady Guadalupe
- González Martínez Aldo Alfredo
- Hernández González André Jaffeth
- Palagot Vega Azucena
- Tepach Fonseca Cristian Jair
- Zacarías Sinta Ismael

811-A

San Andrés Tuxtla a 16 de mayo de 2023

INTRODUCCIÓN.

La operación y control de robots en un ambiente virtual es un tema fascinante y relevante en el campo de la robótica. Con los avances tecnológicos, los entornos virtuales han demostrado ser herramientas valiosas para simular y desarrollar sistemas robóticos antes de su implementación en el mundo real. La operación de robots en un ambiente virtual implica el diseño, la programación y el control de robots utilizando software especializado que recrea de manera realista el comportamiento físico y el entorno en el que operarían los robots reales. Esto proporciona una plataforma segura y eficiente para experimentar, probar algoritmos de control, realizar simulaciones complejas y optimizar el rendimiento de los robots, todo ello sin los costos y riesgos asociados con los entornos físicos. En este contexto, el control preciso, la planificación de movimientos, la interacción con el entorno y la retroalimentación en tiempo real son aspectos cruciales a considerar para lograr un funcionamiento eficiente y seguro de los robots en un ambiente virtual.

3.1 IMPLEMENTACIÓN DE UN SISTEMA OPERADO POR UN ROBOT MEDIANTE UN SIMULADOR DE LAS PROPUESTOS EN LA INDUSTRIA.

La implementación de un sistema operado por un robot mediante un simulador es una tarea compleja que requiere de conocimientos avanzados en robótica, programación y simulación. A continuación, se presenta un ejemplo general de cómo se podría llevar a cabo la implementación de un sistema operado por un robot mediante un simulador.

Selección del simulador: Existen diversos simuladores en la industria, algunos de los más utilizados son Gazebo, V-REP, ROS, Unity, entre otros. Es importante seleccionar el simulador que mejor se adapte a las necesidades y requerimientos del proyecto.

Selección del robot: Una vez seleccionado el simulador, es necesario elegir el robot que se utilizará para la implementación del sistema. Esto dependerá de los objetivos del proyecto y de las características del simulador seleccionado.

Programación del robot: El siguiente paso es programar el robot para que pueda llevar a cabo las tareas asignadas en el sistema. Esto implica diseñar el modelo de control del robot y definir las interacciones entre el robot y el entorno.

Definición del entorno: Es necesario definir el entorno en el que el robot realizará las tareas asignadas. El entorno puede ser una fábrica, un laboratorio o cualquier otro ambiente que se desee simular.

Simulación y validación: Una vez que se ha programado el robot y se ha definido el entorno, se procede a la simulación y validación del sistema. En esta etapa se prueban las funcionalidades del sistema y se realizan ajustes para mejorar el desempeño.

Implementación en el mundo real: Una vez que se ha validado el sistema mediante el simulador, se procede a implementarlo en el mundo real. Para ello, se debe transferir el software desarrollado en el simulador al robot físico y realizar pruebas para asegurarse de que el sistema funciona correctamente.

En resumen, la implementación de un sistema operado por un robot mediante un simulador requiere de un conjunto de habilidades y conocimientos en robótica, programación y simulación. Es importante seleccionar el simulador y el robot adecuados para el proyecto, programar el robot y definir el entorno en el que se llevarán a cabo las tareas asignadas, validar el sistema mediante la simulación y finalmente, implementarlo en el mundo real.

3.2 PROGRAMACIÓN VIRTUAL DE ROBOTS.

La programación virtual de robots es una técnica de simulación que permite desarrollar, probar y validar programas de control para robots en un entorno virtual antes de implementarlos en el mundo real. Esto permite ahorrar tiempo y dinero al evitar errores costosos y reducir el tiempo de inactividad del robot.

Existen diversos softwares de simulación que permiten la programación virtual de robots, algunos de los más utilizados son: RobotStudio, Gazebo, V-REP, ROS, Unity, KUKA Sim Pro, entre otros. Estos softwares ofrecen una amplia variedad de herramientas de modelado, simulación, visualización y programación de robots.

La programación virtual de robots implica el diseño de programas de control para robots en un entorno virtual, definiendo las trayectorias, movimientos, interacciones y tareas que el robot debe realizar. La programación se realiza utilizando lenguajes de programación específicos para robots, como KRL

(KUKA Robot Language), URScript (Universal Robots Script), ROS (Robot Operating System) y otros.

Una vez que se ha diseñado y programado el robot virtual, se procede a la simulación y verificación del programa. Durante la simulación, se pueden realizar pruebas para evaluar el rendimiento del robot y hacer ajustes en el programa para mejorar su desempeño. La simulación también permite identificar posibles problemas y errores que podrían ocurrir en el mundo real.

La programación virtual de robots tiene múltiples ventajas, como:

- Ahorro de tiempo y costos: Permite reducir el tiempo y los costos asociados a la programación y verificación de robots en el mundo real.
- Mayor precisión: Permite verificar la precisión del robot y hacer ajustes en el programa para mejorar su desempeño.
- Identificación de problemas: Permite identificar problemas y errores antes de que ocurran en el mundo real, lo que puede reducir el tiempo de inactividad del robot.
- Seguridad: Permite probar programas en un entorno seguro y controlado, reduciendo el riesgo de accidentes.

3.3 OPERACIÓN VIRTUAL DE CELDAS ROBOTICAS.

La operación virtual de celdas robotizadas se refiere a la simulación de procesos de producción y el comportamiento de los robots encargados de realizar esas funciones. Se utiliza para optimizar nuevos diseños de celdas productivas, determinar la capacidad de producción y encontrar cuellos de botella, lo que reduce los costos al minimizar fallos, reducir tiempos muertos y mejorar el rendimiento.

Una de las ventajas de la operación virtual es que los diseñadores pueden ver cómo se comportará una celda robótica antes de que se construya, lo

que reduce la cantidad de prototipos físicos necesarios y, por lo tanto, reduce los costos de producción generales. Además, la simulación es segura y la gente puede trabajar sin riesgos experimentando con diferentes diseños, mejoras y modificaciones de los procesos y los robots.

Los sistemas de operación virtual utilizan entornos de software especializado para la simulación. Los programas de simulación de celdas robóticas crean entornos virtuales donde es posible demostrar operaciones robóticas y trabajos en el taller, y pueden simular diferentes escenarios y entornos operativos en tiempo real. Estos sistemas llevan la operación virtual un paso más allá, permitiendo a las empresas realizar un análisis detallado del flujo de trabajo y del rendimiento de la celda robótica.

En conclusión, la operación virtual de celdas robóticas es esencial para las compañías que buscan mejorar la eficiencia de la producción y el tiempo de desarrollo, reducir los costos y mejorar la calidad. A través de la simulación, se puede experimentar con diferentes diseños y procesos de producción y mejorar la productividad general de la celda robótica.

3.4 CONTROL DE ROBOTS VIRTUALES.

El control de robots virtuales se refiere a la capacidad de controlar el comportamiento y las operaciones de un robot en un entorno virtual simulado. En la simulación de sistemas robóticos, el control es esencial para modelar y analizar el comportamiento de los robots y evaluar la eficiencia y el rendimiento. El control de robots virtuales utiliza un software especializado que permite a los usuarios programar y controlar el comportamiento del robot en el entorno virtual.

Una de las principales ventajas del control de robots virtuales es que se puede experimentar con diferentes estrategias de control y algoritmos antes de implementarlos en un robot real. Los diseñadores pueden optimizar la programación y reducir los tiempos de desarrollo y los costos ofreciendo

mejoras para el diseño del robot y los sistemas de control. Esto también reduce los riesgos y los costos de fallas, reduciendo la cantidad de prototipos físicos necesarios para prueba y mejora del robot.

El control de robots virtuales requiere de una simulación precisa del entorno y su interacción con el robot, así como de la capacidad de modelar diferentes sensores y actuadores con precisión. También es importante contar con herramientas de visualización y monitoreo para permitir la visibilidad en cada paso y prevenir problemas de desempeño y errores. Las herramientas de visualización hacen que el seguimiento de las tareas del robot sea fácil y aumentan la eficacia del diseño.

En resumen, el control de robots virtuales es una herramienta esencial en la simulación de sistemas robóticos, que permite a los diseñadores programar y controlar el comportamiento del robot en un entorno virtual y optimizar el rendimiento antes de implementarlo en un robot real. Esto reduce los costos y riesgos y mejora el diseño general del robot y los sistemas de control.

3.5 USO DE SOFTWARE FUERA DE LÍNEA EN UN PROBLEMA REAL.

El uso de software fuera de línea se refiere a la capacidad de trabajar con software de simulación de sistemas robóticos sin estar conectado a un robot físico en tiempo real. La tecnología se basa en la captura y la simulación de los movimientos del robot para su posterior análisis y mejora del patrón de movimiento.

El uso de software fuera de línea es útil para los diseñadores de robots ya que les permite analizar y mejorar el desempeño de los robots sin la necesidad de tener el robot real en el sitio para pruebas. El software permite a los diseñadores simular el comportamiento del robot y analizar los datos de movimiento y detectar oportunidades de mejora.

Uno de los casos de uso del software fuera de línea es el de la programación de robots en el sitio de producción. En lugar de programar la tarea de un robot en tiempo real en el sitio de producción, el programador puede simular la tarea en el software fuera de línea, modificar la programación en el software y luego subir la programación al robot físico para su ejecución.

Otro uso del software fuera de línea es la simulación para entrenamiento y educación. Los aprendices pueden trabajar con simulaciones para aprender cómo operar y programar robots sin el riesgo de dañar un robot físico.

Además, el uso de software fuera de línea también puede mejorar la seguridad, ya que los diseñadores pueden trabajar con robots específicos en simulación y crear un modelo seguro y preciso de las operaciones del robot antes de la producción en masa.

En resumen, el uso de software fuera de línea en la simulación de sistemas robóticos es una herramienta valiosa para diseñadores, programadores y aprendices ya que permite analizar y mejorar el comportamiento del robot sin la necesidad de tener el robot real en el sitio para pruebas. Esto reduce los costos y riesgos de producción y mejora la seguridad y el rendimiento de los robots.

CONCLUSIONES.

En conclusión, la operación y control de robots en un ambiente virtual ofrece numerosas ventajas y oportunidades en diversos campos. La implementación de un sistema operado por un robot mediante un simulador se ha convertido en una práctica común en la industria, permitiendo el desarrollo, la validación y la optimización de sistemas robóticos sin la necesidad de utilizar recursos físicos costosos. La programación virtual de robots ofrece flexibilidad y agilidad al permitir la depuración y optimización de algoritmos de control antes de su implementación en entornos reales. La operación virtual de celdas robóticas brinda una visión integral del funcionamiento y la interacción de múltiples robots, facilitando la planificación de tareas complejas y la optimización de procesos. El control de robots virtuales permite evaluar y mejorar el rendimiento de los robots en un entorno seguro y repetible. Por último, el uso de software fuera de línea en un problema real ofrece la posibilidad de realizar simulaciones y pruebas exhaustivas antes de aplicar soluciones en el mundo físico, reduciendo costos y riesgos asociados. En conjunto, la operación y control de robots en un ambiente virtual ofrece una poderosa herramienta para la innovación y el avance en el campo de la robótica.

LISTA DE COTEJO INVESTIGACION

SIMULACIÓN DE SISTEMAS ROBÓTICOS DMD-2204.

Nombre del estudiante: Espinosa Cruz Shady Guadalupe

Tema: Operación y control de robots en un ambiente virtual.

Portada	2 %	2 %
Introducción	5 %	5 %
Desarrollo	20 %	19 %
Conclusiones	5 %	5%
Referencias	3 %	0 %
Entrega en tiempo y forma	5 %	5%
Total	40 %	36%

LISTA DE COTEJO DE PRÁCTICAS

SIMULACIÓN DE SISTEMAS ROBÓTICOS DMD-2204.

PRÁCTICA NÚMERO 3.

Nombre del estudiante: Espinosa Cruz Shady Guadalupe.

Tema: Desarrolla un programa utilizando software de simulación y resuelva un problema real.

Portada	5 %	5 %
Introducción	5 %	5 %
Desarrollo (explicación)	35 %	35 %
Conclusiones	5 %	5 %
Referencias	5 %	5 %
Entrega en tiempo y forma	5 %	5 %
Total	60 %	60 %

INSTITUTO TECNOLOGICO SUPERIOR DE SAN ANDRES TUXTLA

SIMULACIÓN DE SISTEMAS ROBÓTICOS

DR. JOSÉ ÁNGEL NIEVES VÁZQUEZ

PRÁCTICA UNIDAD III:

OPERACIÓN Y CONTROL DE UN ROBOT EN UN AMBIENTE VIRTUAL

- Espinosa Cruz Shady Guadalupe
- Hernández González André Jaffeth
- Palagot Vega Azucena
- Tepach Fonseca Cristian Jair
- Zacarías Sinta Ismael

811-A

San Andrés Tuxtla a 10 de mayo de 2023

INDICE

AZUCENA PALAGOT VEGA

INTRODUCCIÓN

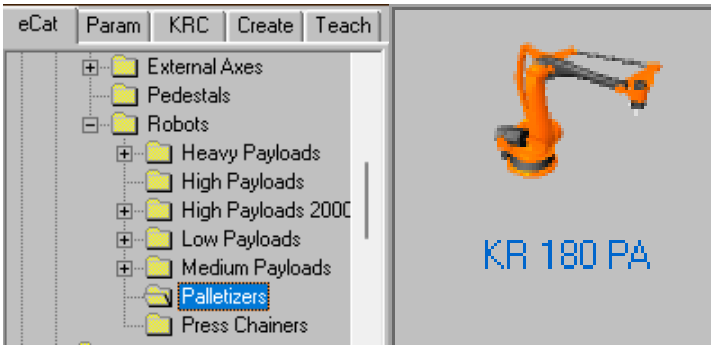
La automatización de procesos industriales es cada vez más común en la industria moderna. El uso de robots y sistemas de control avanzados puede mejorar la eficiencia y la precisión de la producción, lo que a su vez puede aumentar la rentabilidad de las empresas. En esta práctica se utilizó el software Kuka Sim Pro 1.1 para simular el funcionamiento de una cinta transportadora y un robot KR 180 PA, con el objetivo de automatizar el proceso de acomodar cajas sobre una plataforma.

PRACTICA

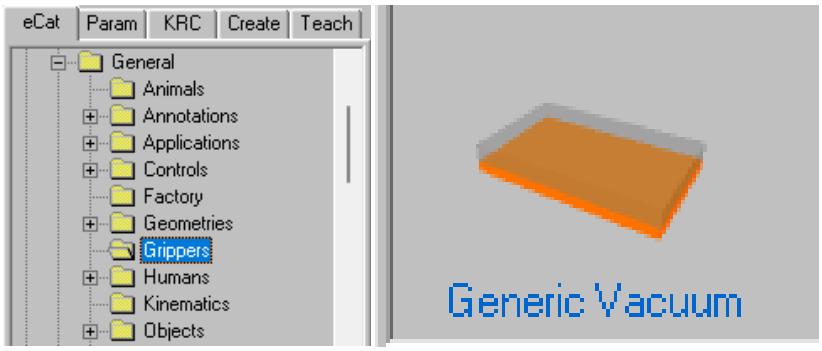
Creación de objetos

Para realizar esta práctica lo primero que debemos hacer es añadir los objetos con los que trabajaremos.

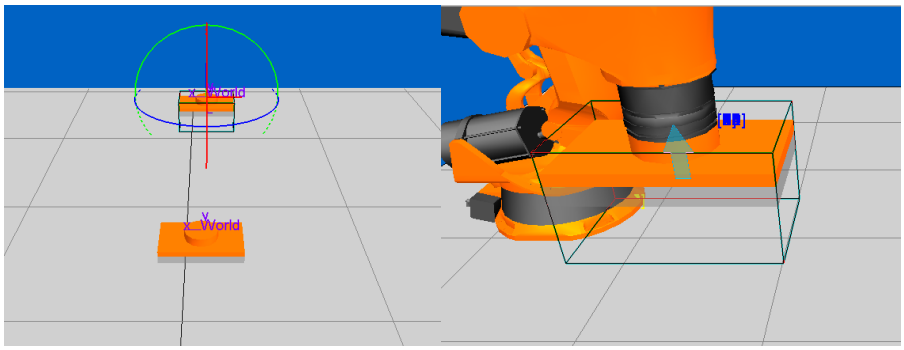
1. Al iniciar seleccionaremos el robot, en este caso utilizaremos el robot "KR 180 PA" que se encuentra en la catergoria de PAlletizers.



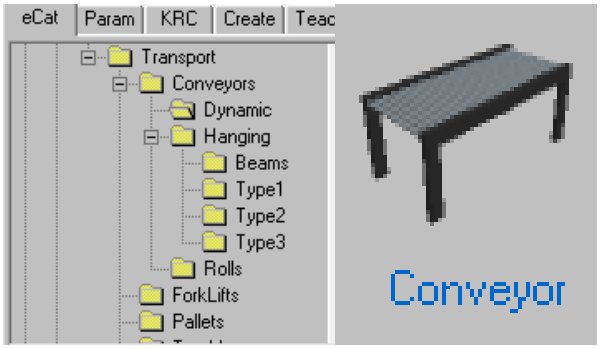
2. Despues seleccionamos el tipo de gripper que se usará, para el cual será "Generic Vaccum".



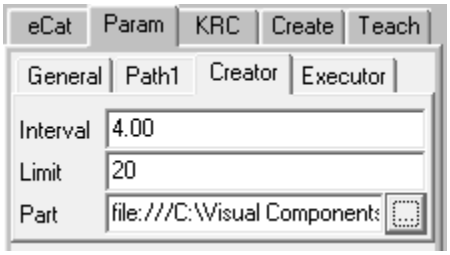
3. Para concatenar el gripper al robot debemos seleccionar el gripper y moverlo al robot, sabremos que entan conectados cuando nos aparezca una flecha azul.



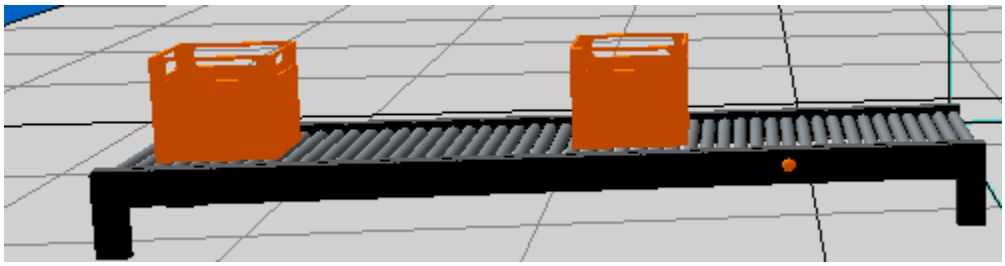
4. Tambien debemos añadir una cinta transportadora para las piezas.



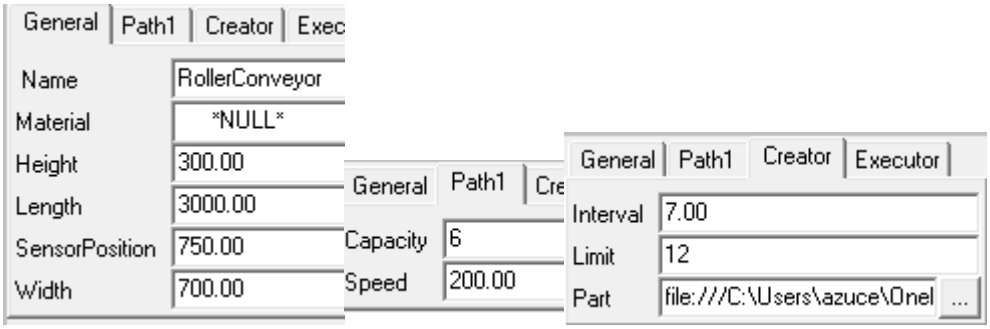
5. A la cinta transportadora debemos cambiarle el componente a crear, una creará las cajas. Eso se realiza en el apartado de "Part"



6. Las cinta transportadora se vera de la siguiente manera:

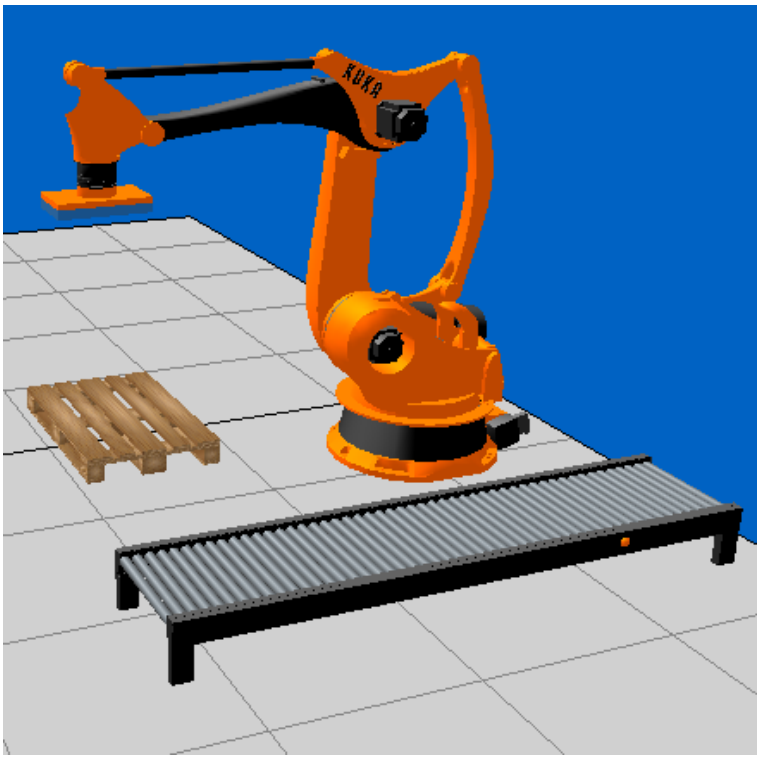


7. Tambien debemos ajustar diferentes parametros para que se ajuste a los requerimientos del proceso:



8. Añadiremos una plataforma para acomodar las cajas:

9. Acomodamos todos los componentes para que el entorno se vea de la siguiente manera:



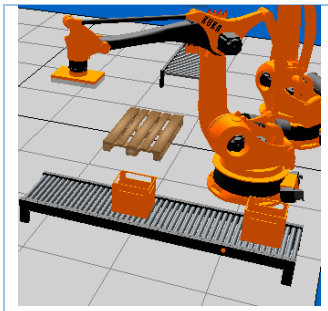
Movimientos de los robots

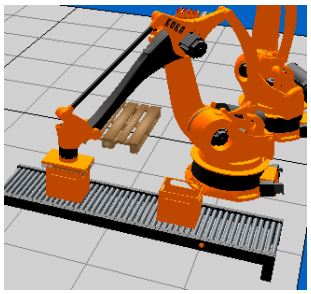
Ya que tenemos todos los objetos ahora lo que debemos hacer es añadir los movimientos de los robots, para lo cual, selecionaremos el robot y en la pestaña de Teach iremos añadiendo cada una de las posiciones que necesitamos del robot.

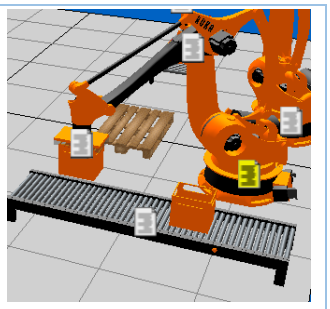
Para lograr esto usaremos las opciones de Jog joints: para mover cada punto de forma libre, Trn tool: para mover en coordenadas cartesianas y Rot tool: para mover en coordenadas esfericas.

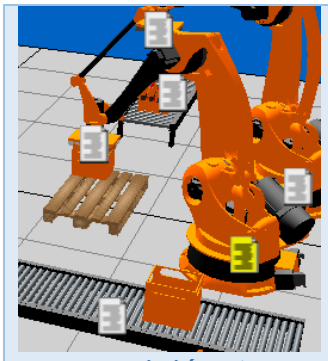
Esta parte es sencilla ya que solo debemos posicionar los robots en donde lo necesitemos y marcar ese punto.

Cinta transportadora

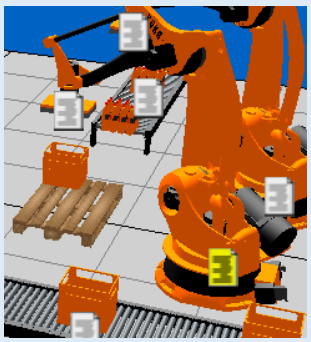

Posición 1


Posición 2

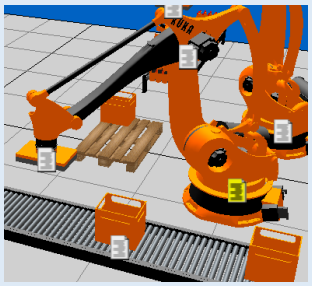

Posición 3



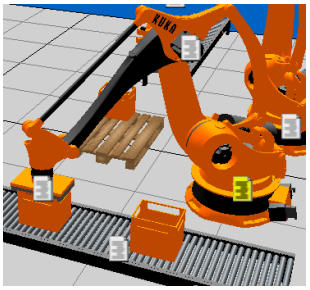
Posición 4



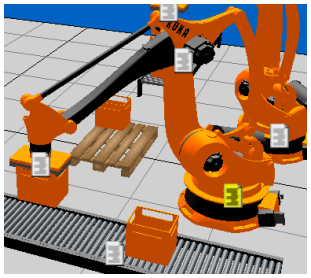
Posición 5



Posición 6

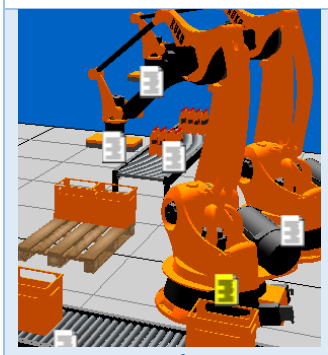


Posición 7

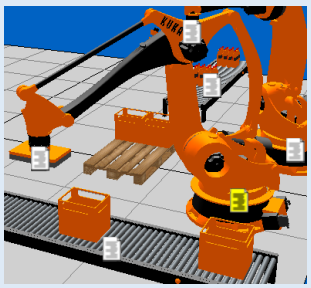


Posición 8

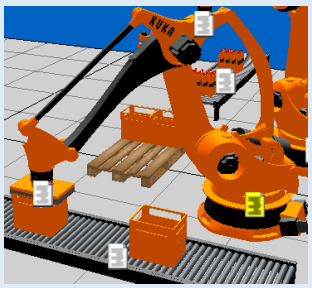
Posición 9



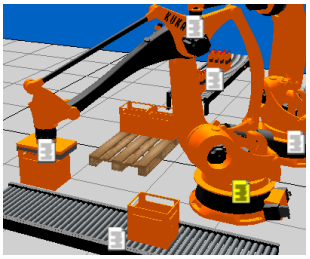
Posición 10



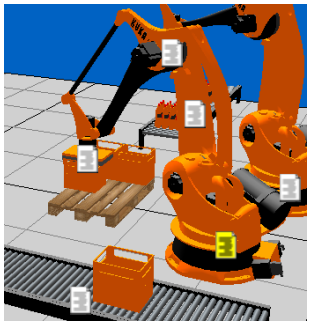
Posición 11



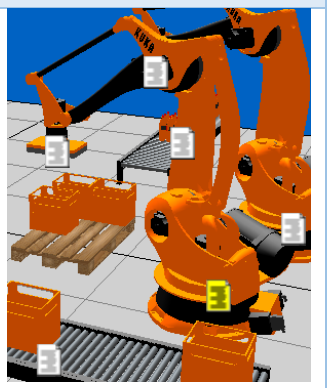
Posición 12



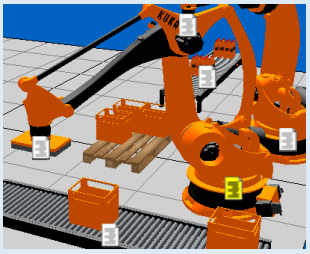
Posición 13



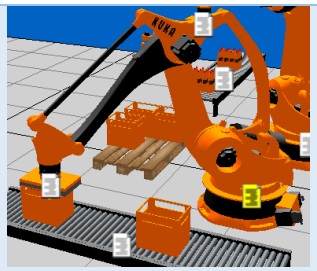
Posición 14



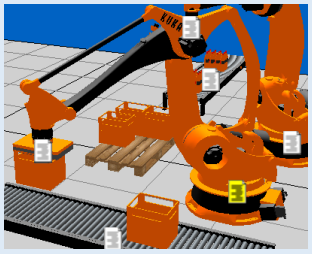
Posición 15



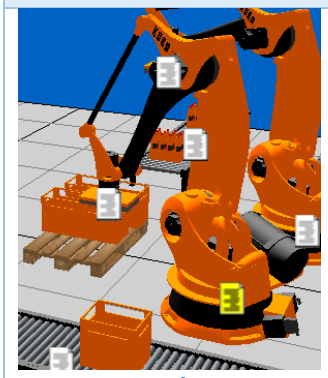
Posición 16



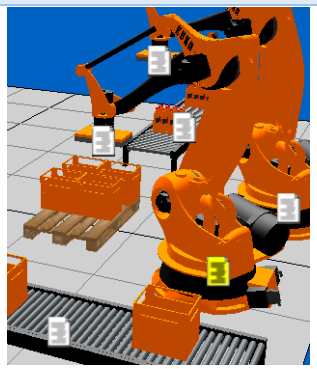
Posición 17



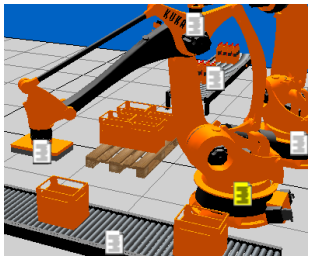
Posición 18



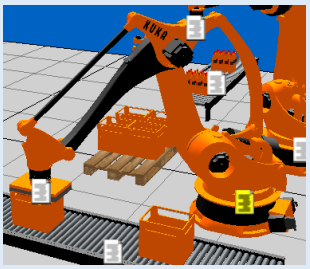
Posición 19



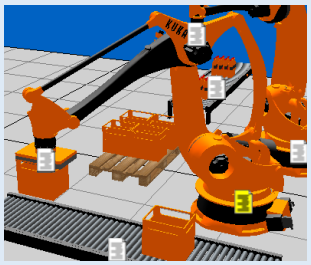
Posición 20



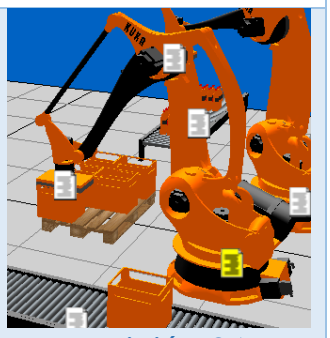
Posición 21



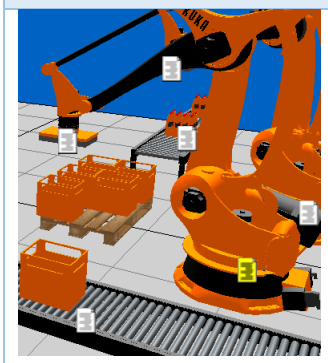
Posición 22



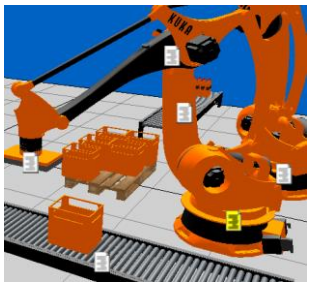
Posición 23



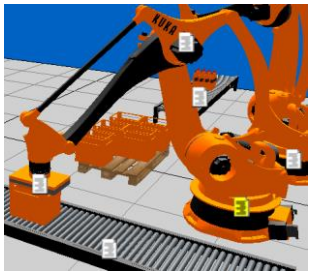
Posición 24



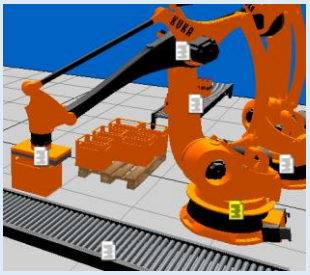
Posición 25



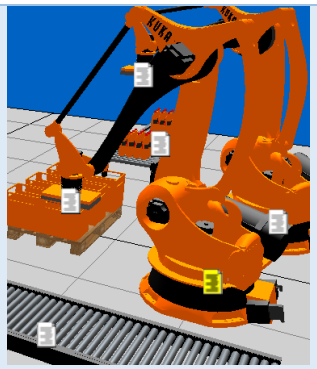
Posición 26



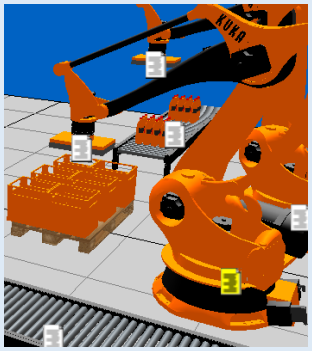
Posición 27



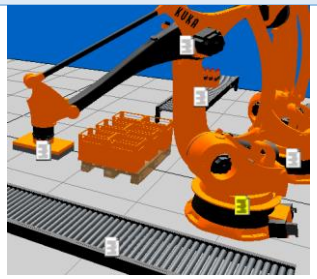
Posición 28



Posición 29



Posición 30



Posición 31

Aquí se muestran todas las posiciones del robot, los tiempos de reposo y las aberturas y cierres de gripper:

CONCLUSIÓN

En conclusión, la práctica realizada utilizando el software Kuka Sim Pro 1.1 fue un éxito en la simulación de un sistema de automatización de producción. La cinta transportadora y el robot KR 180 PA trabajaron juntos para mover y colocar las cajas de manera precisa y eficiente en la plataforma, lo que demuestra el potencial de la automatización en la industria. La simulación proporcionó una representación visual del proceso y permitió la optimización de los parámetros antes de la implementación del sistema en la vida real. En general, el uso de tecnología avanzada como el software Kuka Sim Pro 1.1 puede mejorar significativamente la eficiencia de la producción y reducir los costos operativos en la industria.

ANDRÉ JAFFETH HERNÁNDEZ GONZÁLEZ

INTRODUCCIÓN

La simulación y el control de robots en entornos virtuales juegan un papel fundamental en el desarrollo y la implementación de sistemas robóticos. Uno de los softwares más destacados en este campo es CoppeliaSim. CoppeliaSim proporciona un entorno de simulación versátil y poderoso que permite a los ingenieros y desarrolladores diseñar, operar y controlar robots de manera eficiente y precisa.

La importancia de CoppeliaSim radica en su capacidad para crear un ambiente virtual realista donde se pueden modelar y probar diferentes aspectos del funcionamiento de un robot. Permite simular la física del mundo real, incluyendo la dinámica de los objetos, la interacción con el entorno y la detección de colisiones. Esto resulta invaluable en el diseño y la validación de algoritmos de control, planificación de movimientos, sistemas de visión, manipulación de objetos y muchos otros aspectos relacionados con la operación de robots.

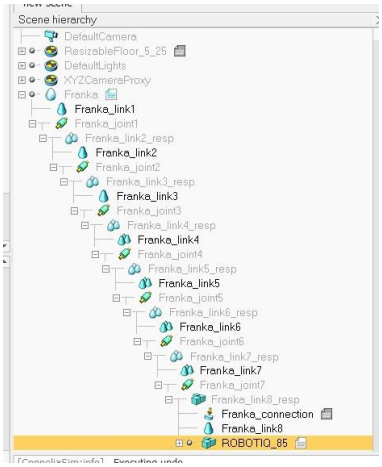
Al utilizar CoppeliaSim, los ingenieros pueden desarrollar y perfeccionar algoritmos de control sin tener que depender de un hardware físico. Esto les brinda la flexibilidad de experimentar con diferentes enfoques y optimizar el rendimiento del robot antes de implementarlo en el mundo real. Además, la capacidad de simular múltiples robots y escenarios complejos en un mismo entorno permite la evaluación y el análisis exhaustivo de sistemas robóticos completos.

En resumen, CoppeliaSim desempeña un papel crítico en la operación y control de robots en ambientes virtuales. Su capacidad para simular de manera realista el comportamiento de los robots y su entorno, junto con su interfaz intuitiva y herramientas de programación avanzadas, lo convierten en una herramienta indispensable para ingenieros, investigadores y

desarrolladores que buscan diseñar, probar y optimizar sistemas robóticos antes de su implementación en el mundo real.

PRACTICA U3: SIMULACIÓN DE UN SISTEMA ROBÓTICO PARTE 2

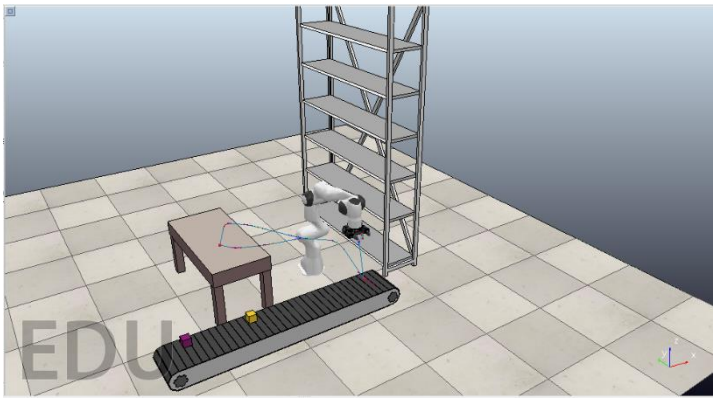
Como primer punto es de suma importancia conocer que Coppelia maneja los elementos de la "escena" de manera jerárquica. Cada componente que agreguemos al espacio de trabajo se ira sumando a este árbol de organización.



Comencé agregando los elementos necesarios para esta simulación los cuales fueron:

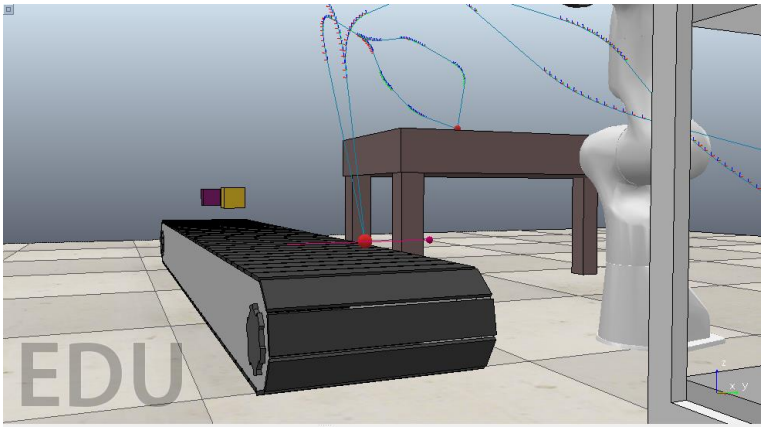
- 1 brazo robótico de Franka
- 1 gripper Robotiq 85
- 1 banda transportadora común
- 2 cubos
- 1 sensor de proximidad tipo rayo
- 1 Mesa modificable
- 1 rack

Los cuales acomodé con la siguiente disposición:



Para comenzar acoplé el gripper al brazo robótico mediante la unión de sus coordenadas cartesianas. Marqué dentro de la jerarquía de objetos al gripper dentro de la ultima junta del brazo robótico y una vez que los centros de ambos objetos quedaron alineados en las coordenadas (0,0,0) tomando como referencia el centro de la última junta, procedí a darle un poco de espacio al gripper para que tuviera un área donde maniobrar. El acomodo del gripper se hizo con la herramienta de traslación de Coppelia.

Una vez que el brazo estaba listo para usarle, procedí a colocar los cubos al principio de la banda transportadora. Estos cubos se tuvieron que configurar de manera que tuvieran unas medidas de 2 cm por lado, por otra parte se les agregó la propiedad de ser detectables para que al llegar al final de la banda, el sensor los pudiera detectar y así detuviera la banda. Estos quedaron colocados como se muestra a continuación.



Una vez que se ubicaron de manera adecuada, procedí con la programación de la banda transportadora, en el código se indica que al detectar las "cajas" el sensor mandara una señal a la banda y esta se detendrá. Esto para que de tiempo al brazo de colocarlas en sus respectivos espacios. El código usado en la banda es el que se muestra a continuación.

```
function sysCall_init()
    pathHandle=sim.getObjectHandle("ConveyorBeltPath")
    sim.setPathTargetNominalVelocity(pathHandle,0) -- for backward compatibility

    sensor = sim.getObjectHandle('Proximity_sensor')

end

function sysCall_actuation()
    local beltVelocity=sim.getScriptSimulationParameter(sim.handle_self,"conveyorBeltVelocity")

       if (sim.readProximitySensor(sensor)>0) then
            beltVelocity = 0
    end

    local dt=sim.getSimulationTimeStep()
    local pos=sim.getPathPosition(pathHandle)
    pos=pos+beltVelocity*dt
    sim.setPathPosition(pathHandle,pos) -- update the path's intrinsic position
end
```

Al mismo tiempo que se mueve la banda transportadora y moviliza los cubos, el brazo robótico realiza una trayectoria para estar a tiempo y recoger los

```
-- Definimos la trayectoria
path = sim.getObjectHandle("Path")
path1 = sim.getObjectHandle("Path0")
path2 = sim.getObjectHandle("Path1")
path3 = sim.getObjectHandle("Path2")
path4 = sim.getObjectHandle("Path3")
--Definimos el objetivo
target = sim.getObjectHandle("Target")

--Definimos la velocidad
Vel = 0.05

--Conexion con el gripper
connection = sim.getObjectHandle('Franka_connection')
gripper = sim.getObjectChild(connection,0)
gripperName = "ROBOTIQ_85"

Cube = sim.getObjectHandle("CuboAmarillo")
Cube1 = sim.getObjectHandle("CuboMorado")

Gripper = sim.getObjectHandle("ROBOTIQ_85")

--Estableciendo el objeto como dinamico
sim.setObjectInt32Parameter(Cube,3003,0)
sim.setObjectInt32Parameter(Cube1,3003,0)
```

cubos sin demorar mucho. Se colocaron varios puntos dentro de la trayectoria para hacerla un poco mas suave y evitar movimientos bruscos los cuales pueden resultar en daño a nuestras "cajas" e incluso en soltarlas. El código que se implementó en el brazo robótico es el siguiente:

```
26
27
28  function sysCall_threadmain()
29  --Hacemos que nuestro robot siga la trayectoria
30  sim.followPath(target,path,3,0,Vel,0.05)
31
32  sim.wait(1)
33  --Cerramos el gripper
34  sim.setIntegerSignal(gripperName..'_close',1)
35  sim.setObjectInt32Parameter(Cube,3003,1)
36  sim.resetDynamicObject(Cube)
37  sim.setObjectParent(Cube,Gripper,true)
38  sim.wait(2)
39  sim.followPath(target,path1,3,0,Vel,0.05)
40  --Toma el cubo amarillo
41  sim.clearIntegerSignal(gripperName..'_close')
42  sim.setObjectInt32Parameter(Cube,3003,0)
43  sim.setObjectParent(Cube,-1,true)
44  sim.wait(2)
45  --Lo lleva a la mesa
46  sim.followPath(target,path2,3,0,Vel,0.05)
47  sim.wait(2)
48  --Lo deja en la mesa
49  sim.setIntegerSignal(gripperName..'_close',1)
50  sim.setObjectInt32Parameter(Cube1,3003,1)
51  sim.resetDynamicObject(Cube1)
52  sim.setObjectParent(Cube1,Gripper,true)
53  sim.wait(2)
54  --Regresa a la banda
55  sim.followPath(target,path3,3,0,Vel,0.05)
56  --Toma el cubo morado
57  sim.clearIntegerSignal(gripperName..'_close')
58  sim.setObjectInt32Parameter(Cube1,3003,0)
59  sim.setObjectParent(Cube,-1,true)
60  sim.wait(2)
61
62  sim.followPath(target,path4,3,0,Vel,0.05)
63  end
```

Como se observa en la programación lo primero es declarar variables y algunas condiciones iniciales y posteriormente se inicializa el programa inicial el cual permite que el robot siga las trayectorias, así como ejecutar

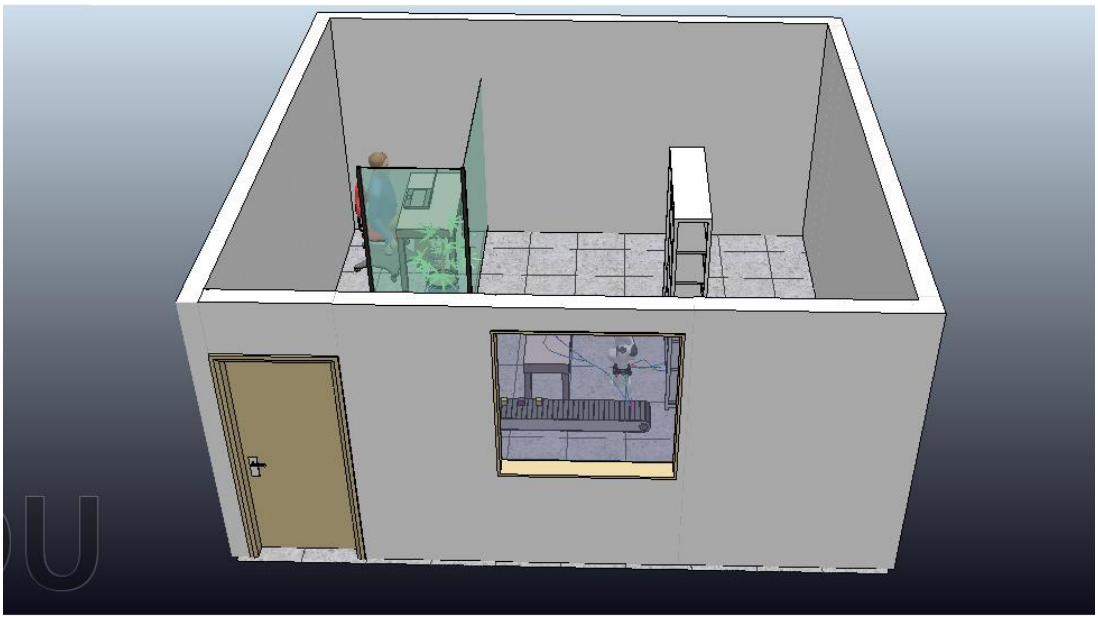
acciones tales como abrir y cerrar la mano robótica, modificar los giros y las perspectivas.

Por otra parte, agregamos partes, mobiliario e infraestructura del que sería un sistema real de producción y almacenaje.

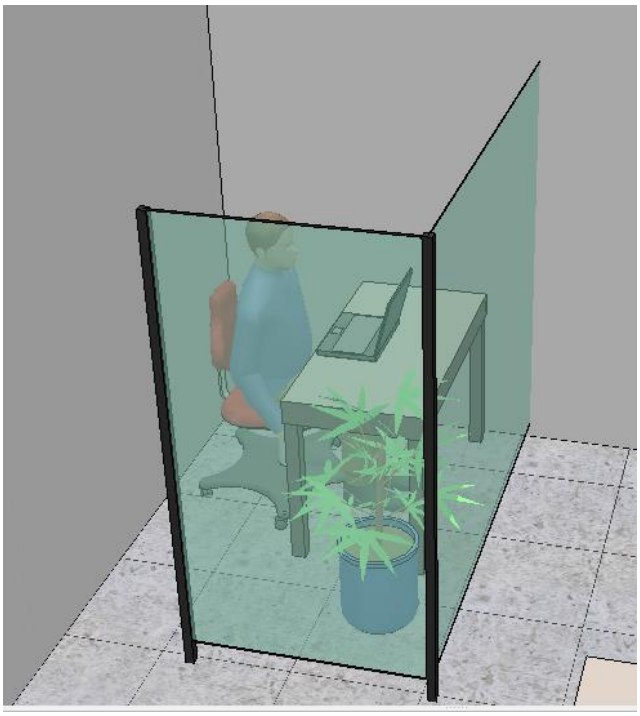
Buscamos desarrollar una simulación que pueda emular lo mas fielmente a un entorno cotidiano para poder contemplar todos los factores como espacios limitados y zonas donde se mueva el personal ya que tenemos que examinar todos estos escenarios para poder crear una programación de los robots de manera adecuada, parte esencial de un entorno eficaz y eficiente poder prever todas las posibilidades y adelantarse a posibles complicaciones para evitarlas.

Los robots usados en esta simulación tienen un área específica donde trabajar y que tiene que ser respetado por los operadores para prevenir accidentes o colisiones que puedan resultar en daños humanos y materiales.

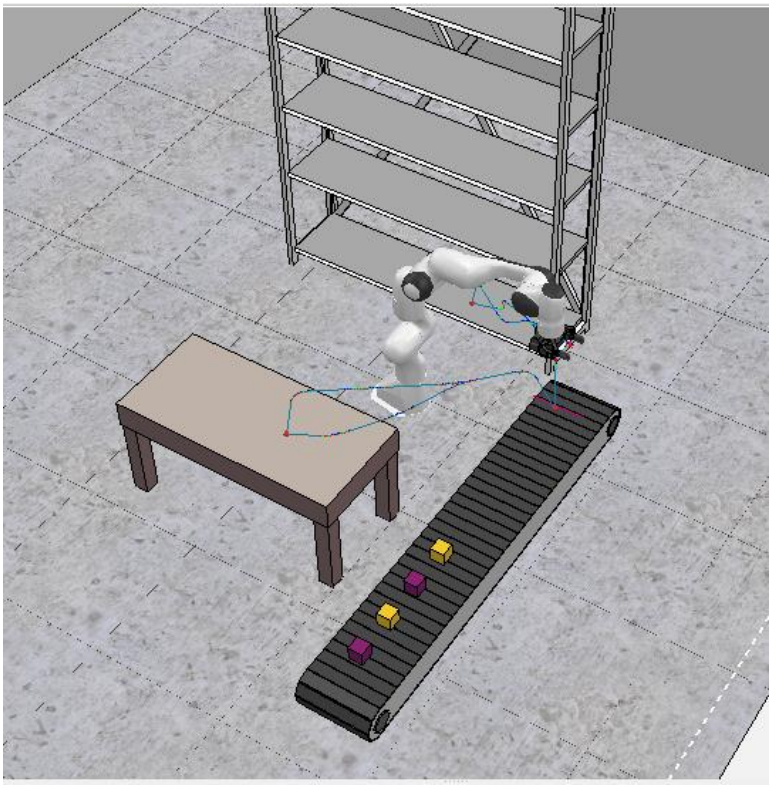
ENTORNO ACTUAL DE TRABAJO

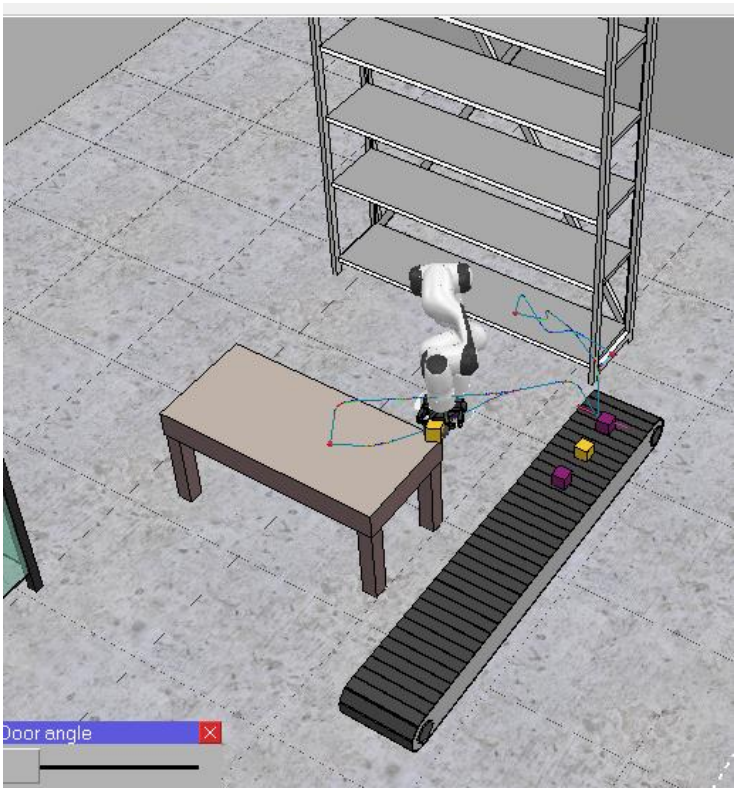


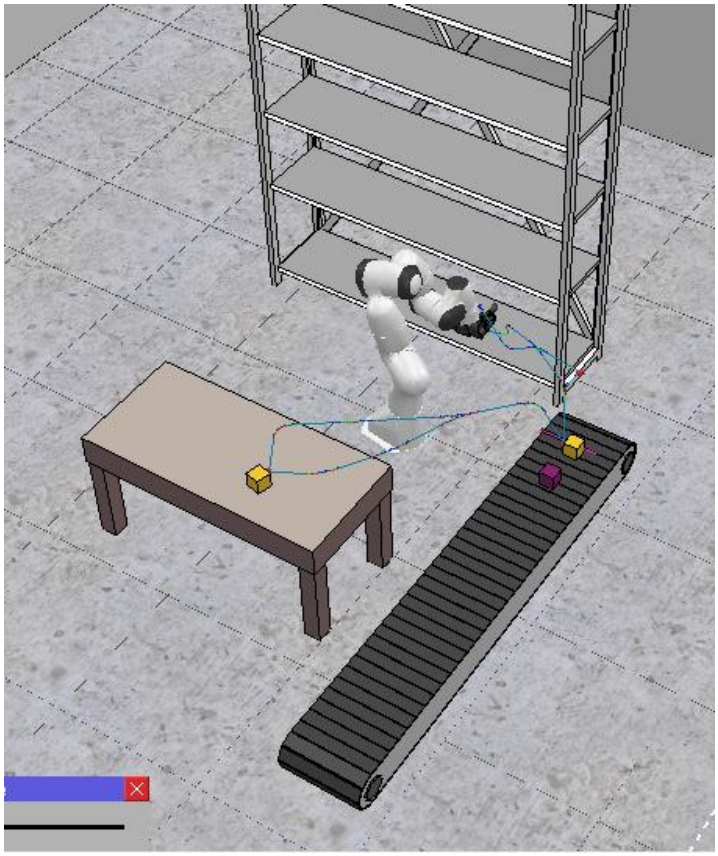
ZONA DEL OPERADOR EN PLANTA



ZONA DE PRODUCCIÓN Y ALMACENAJE




Door angle



CONCLUSIÓN

En conclusión, el software CoppeliaSim desempeña un papel crucial en la operación y control de robots en entornos virtuales. Proporciona a los ingenieros y desarrolladores la capacidad de diseñar, probar y optimizar sistemas robóticos de manera eficiente y precisa. Gracias a su capacidad para simular la física del mundo real, los usuarios pueden modelar y validar algoritmos de control, planificación de movimientos y manipulación de objetos, entre otros aspectos, antes de implementarlos en el mundo real. Esto permite reducir costos, acelerar el proceso de desarrollo y minimizar riesgos asociados con la implementación de robots físicos.

Además, la interfaz intuitiva y las herramientas avanzadas de programación de CoppeliaSim facilitan la tarea de programar y controlar robots virtuales. Los usuarios pueden interactuar con los modelos de robots, crear escenarios personalizados, implementar algoritmos de control y visualizar datos en tiempo real de manera eficiente. Esta versatilidad y flexibilidad hacen que CoppeliaSim sea una herramienta indispensable para ingenieros, investigadores y desarrolladores en el campo de la robótica, brindando un entorno virtual realista y poderoso para el desarrollo y la validación de sistemas robóticos de vanguardia.

SHADY GUADALUPE ESPINOSA CRUZ

INTRODUCCIÓN

La robótica ahora juega un papel importante en nuestras vidas, especialmente el entorno social en diferentes campos de la industria. A medida que aumenta la complejidad de las aplicaciones en robótica, es esencial la importancia de realizar una fase de simulación antes de la construcción implementación de robots físicos en el mundo real. De este modo simule sus pruebas para ahorrar costos y tiempo, nuevo progreso en el campo de la robótica, se pueden diseñar y construir prototipos los robots virtuales se están convirtiendo en un auténtico reto porque son necesarios criterios rigurosos para su simulación de forma que lo abarque todo diferentes aspectos de adquisición de señal y movimiento puede utilizar escenarios para realizar análisis cuantitativos resultados medibles del rendimiento del robot.

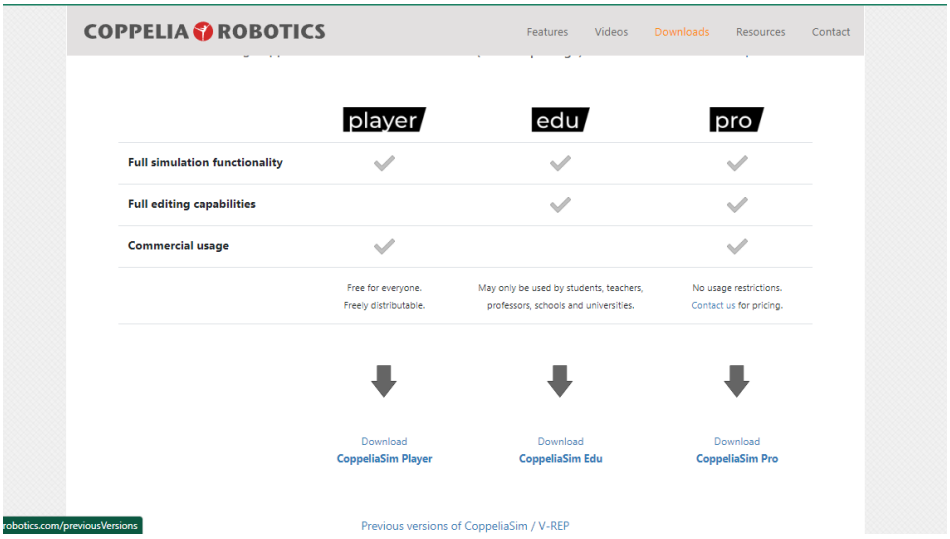
El simulador de robótica CoppeliaSim versión educativa 4.1.0, con entorno de desarrollo integrado, se basa en una arquitectura de control distribuido: cada objeto/modelo se puede controlar individualmente a través de un script integrado, un complemento, un nodo ROS, un cliente API remoto o una solución personalizada. Esto hace que CoppeliaSim sea muy versátil e ideal para aplicaciones multi-robot. Los controladores se pueden escribir en C/C++, Python, Java, Lua, Matlab u Octave.

CoppeliaSim se utiliza para el desarrollo rápido de algoritmos, simulaciones de automatización de fábricas, creación rápida de prototipos y verificación, educación relacionada con la robótica, monitoreo remoto, verificación doble de seguridad, como gemelo digital y mucho más.

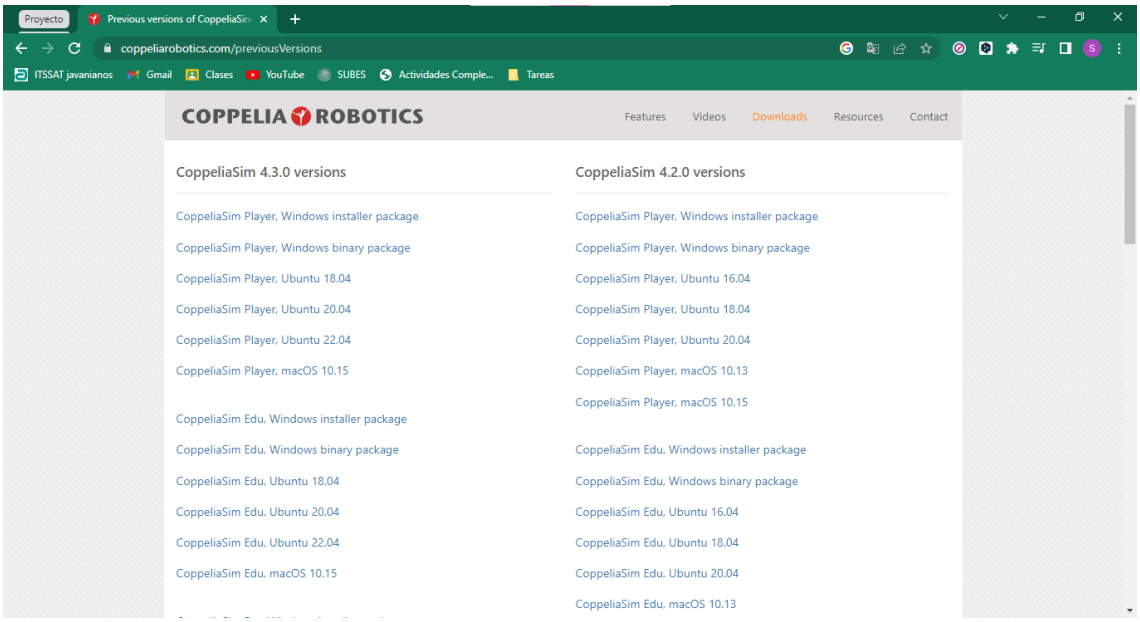
INSTALACIÓN

Para poder descargar el software, se debe visitar la página oficial (www.coppeliarobotics.com) en la cual podremos encontrar información respecto a este programa y material de apoyo para su uso; el sitio se encuentra en inglés, pero cuenta con la opción de traducirlo al español para su mejor compresión.

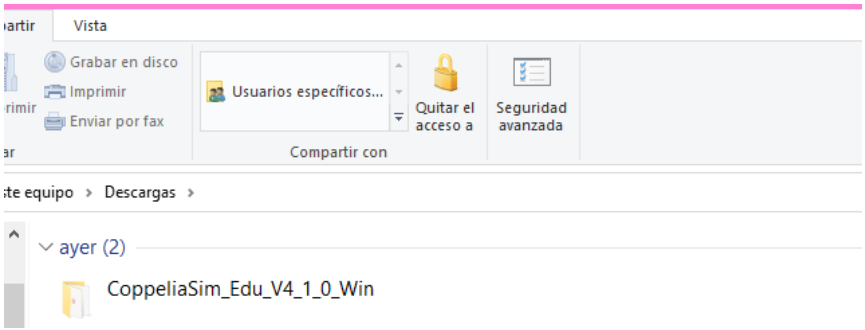
Una vez estando en el inicio, nos dirigiremos a la opción Downloads y daremos click en ella, así seremos dirigidos a las opciones que tenemos disponibles para descargar del software y que versiones ofrece.



Seleccionaremos la opción que se encuentra al final de la pagina "Previous versions of CoppeliaSim / V-REP", ese enlace nos llevará a todas las versiones gratuitas (nuevas o antiguas) que existen del programa.



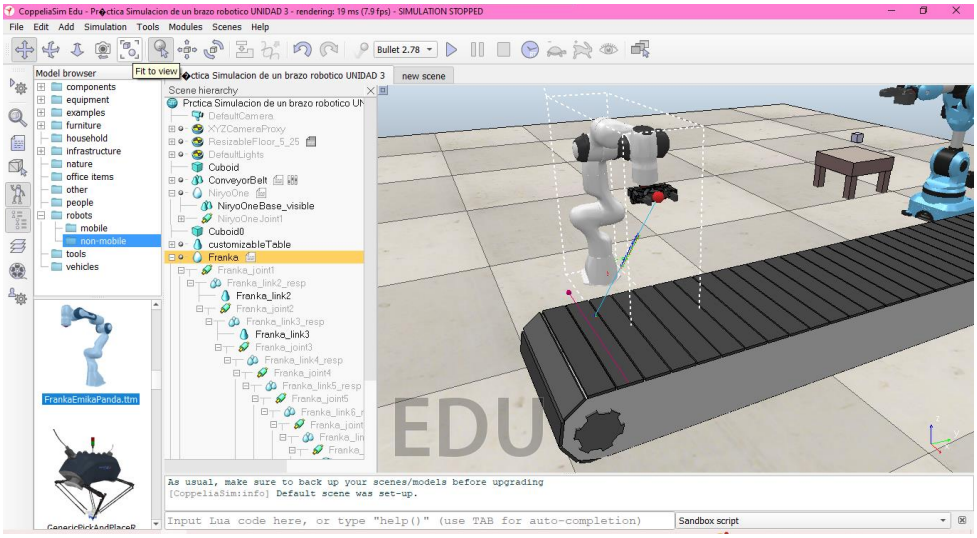
Estando allí, seleccionaremos la opción que sea más cómoda para nuestro uso, ya sea por la versión que necesitaremos o el equipo que poseemos, en este caso usaremos la versión 4.1.0, descargando el archivo tipo "Binary package" ya que este no presentas tantos bugs al se usada, de igual forma, esta versión ofrece una cantidad de opciones que no están disponibles en las versiones actuales, una vez que seleccionemos el link , el archivo se descargara de forma inmediata en nuestro equipo, pero será como un archivo comprimido, así que se necesitará el uso de un programa externo para descomprimir la carpeta (el uso de ese tipo de software es independiente de cada usuario)



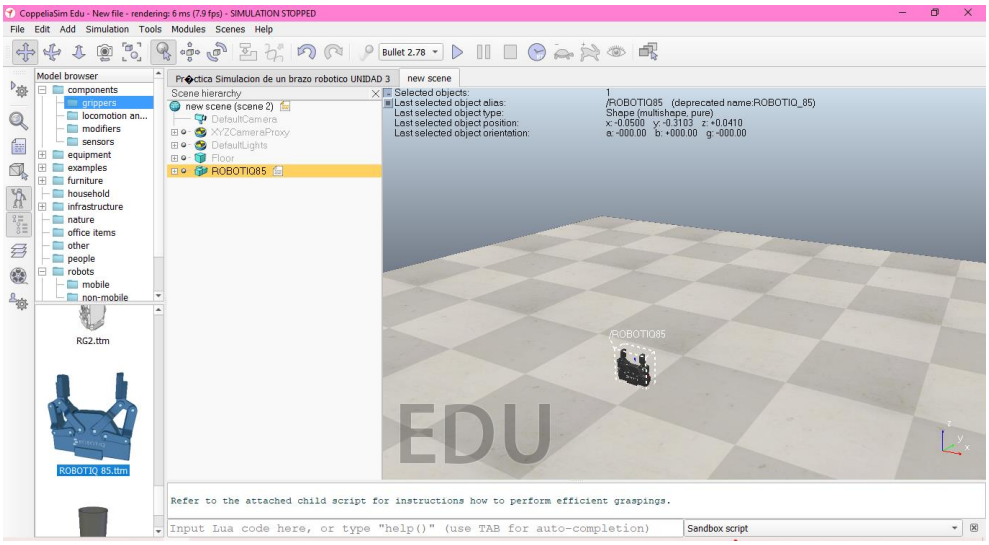
Una vez descargado y descomprimido el archivo, nos aparecerá una carpeta con el nombre del software y la versión que descargamos, entraremos a esta y buscaremos el archivo que lleve por nombre "CoppeliaSim", daremos doble-click, y así se instalará el programa en nuestro equipo y estará listo para usar.

Práctica: "Uso de brazo robótico para el desplazamiento de producto sobre barras transportadoras – Segunda Parte"

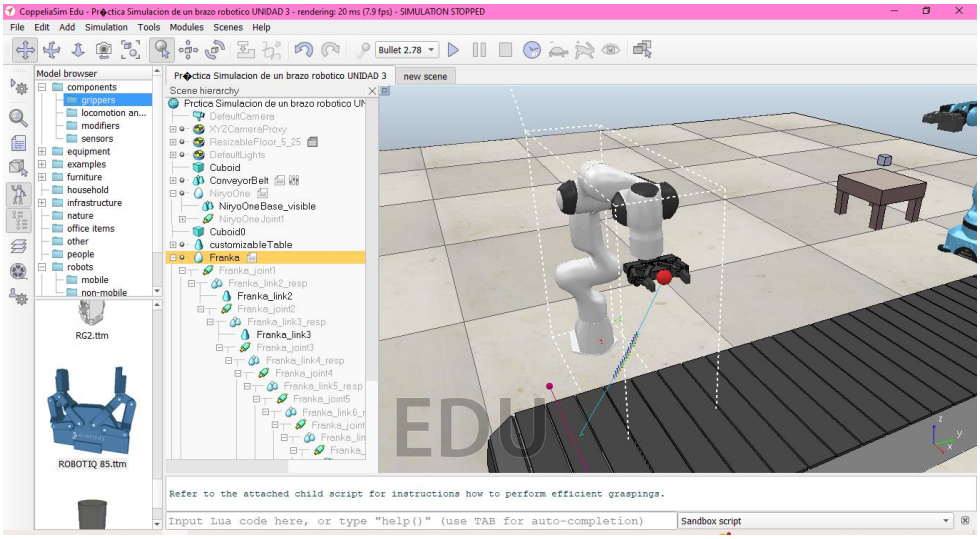
Retomando la práctica anterior, teniendo ya los objetos, la banda y el brazo Niryo One programado y listo para la simulación. Ahora iremos a la barra de componentes y en la carpeta anterior de robot, ahora seleccionamos el brazo industrial "FrankaEmikaPanda" y lo colocamos en nuestra escena.



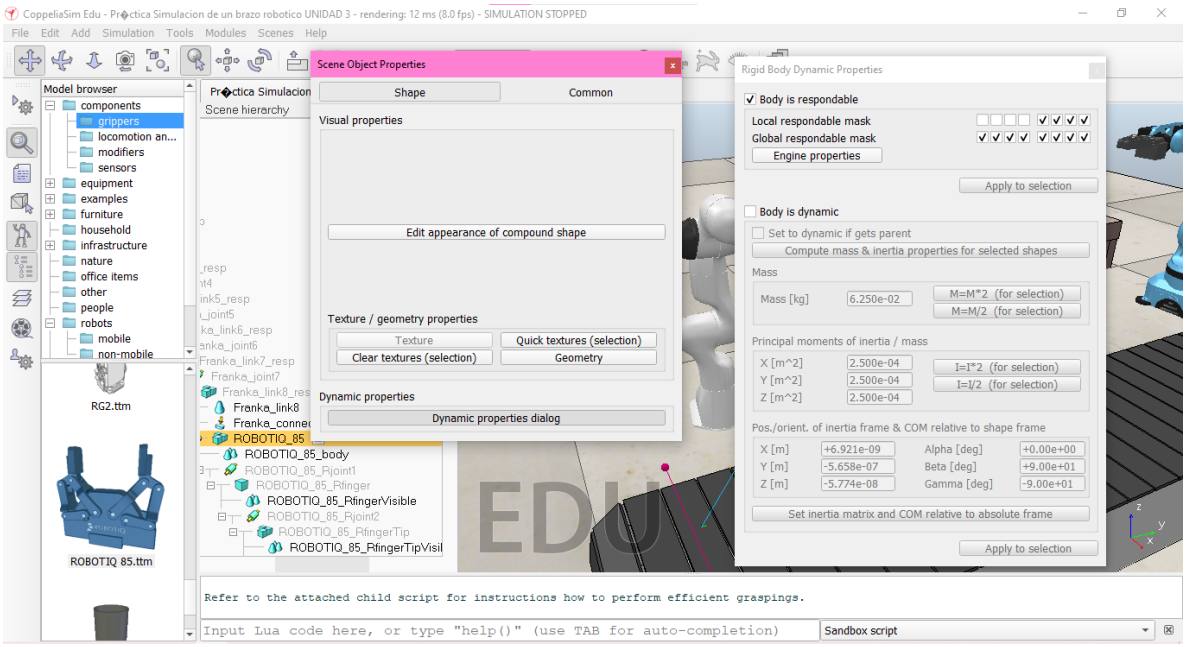
Lo siguiente es trabajar con el brazo, sus conexiones y su programación; una vez que tenemos nuestro robot de nuestra preferencia vamos a proceder a agregar la gripper están aquí en components vamos aquí a elegir yo creo que esté el robótico 85 porque es el más fácil de saber más versátil entonces lo vamos a agregar a escena.



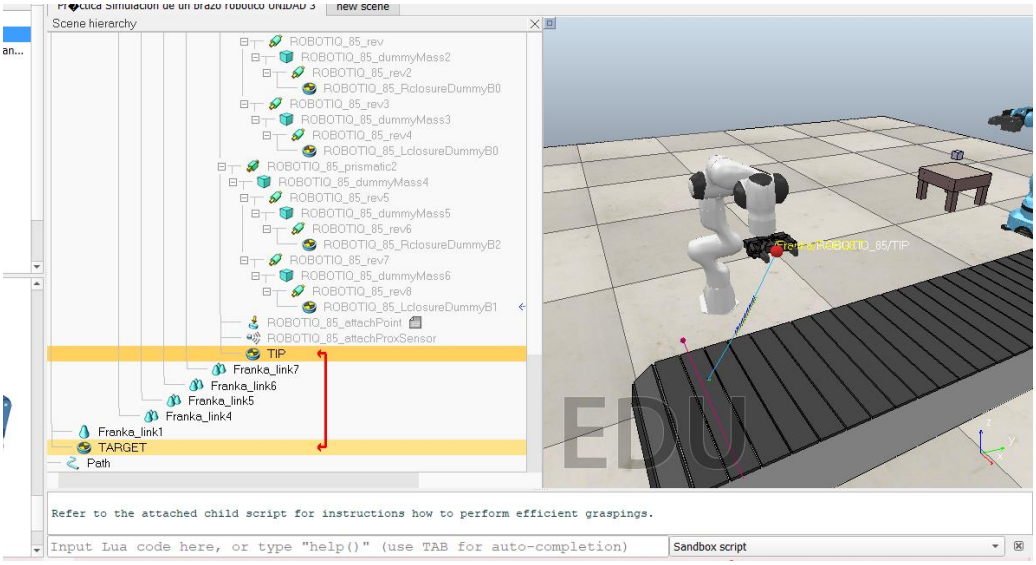
Lo que necesitamos es que el gripper quede en el último eslabón de nuestro brazo robot, entonces nos vamos a ir al último link del robot que hayan elegido al resto, arrastramos el robot y el gripper va. Ahora nos iremos a la posición del gripper y le pongo que su posición sea respecto a su padre si le ponemos ceros el gripper se va a mover a la posición donde está la muñeca, como se muestra en la imagen.



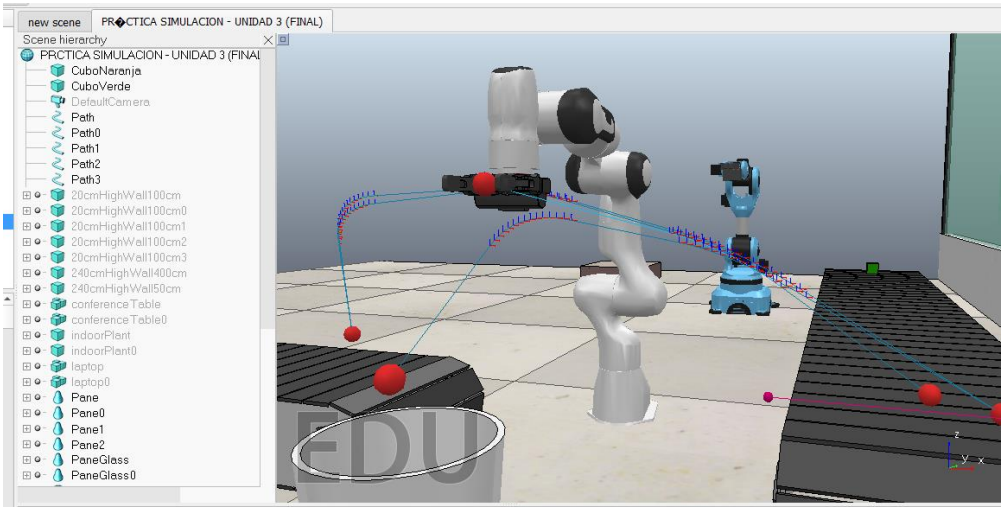
Ahora iremos al icono de nuestro gripper, para que este en sintonía con el movimiento de brazo, es activar us opciones y nos vamos a "Dynamic properties dialog" y desactivamos la opción "Body is Dynamic"



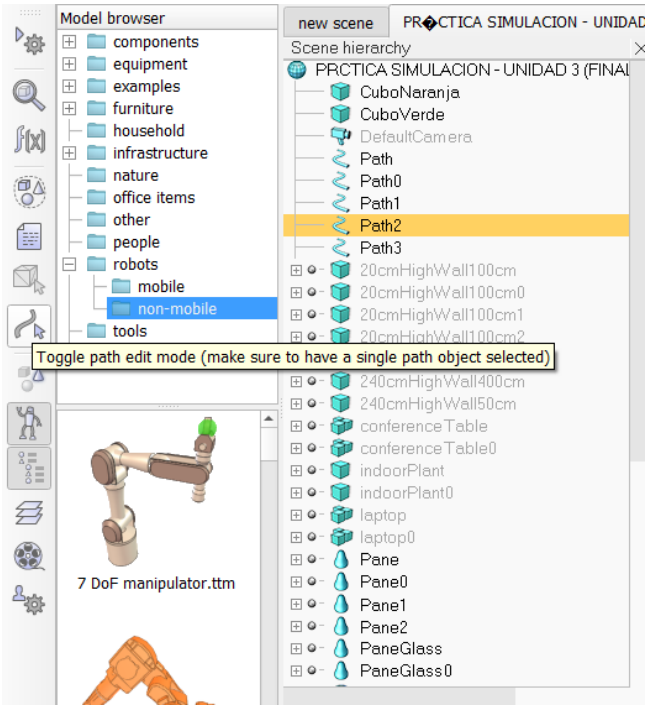
Crearemos un tip, que pondremos dentro de nuestro gripper, para después crear otro tip que será llamado TARGET, y haremos las conexiones como hicimos en la práctica anterior para que nuestro brazo robótico pueda tener trayectorias. De esta manera en nuestra lista de componentes podremos ver cómo están unidos estos dos "tips".



Lo siguiente que hicimos fue crear trayectorias, agregando “Paths” a la escena y colocando nuestra trayectoria como muestra en la imagen, son 2 trayectorias, pero como una está en inversa para no afectar el movimiento del robot no logra apreciarse correctamente. De igual forma agregamos todas las rutas que usará esta simulación.



Al final tendremos 5 paths al final, estas rutas fueron editadas en el apartado para editar los paths, el cual nos permite agregar más puntos de control, nos permite invertir la selección de los puntos ya creados. Para activar esta herramienta solo debemos seleccionar la ruta que queremos editar y solo presionar el icono en la barra de la izquierda.



Al final obtuvimos 5 Paths con 5 movimientos individuales cada uno, cumpliendo con los 20 movimientos que requería la práctica.

Las últimas modificaciones que se agregaron fue en la zona de códigos, los cuales solo tuvieron declaraciones extras respectos a los nuevos movimientos que realizaron los brazos robóticos.

Threaded child script (Franka)

```
--Definimos la trayectoria
path= sim.getObjectHandle("Path")
path1= sim.getObjectHandle("Path0")
path2= sim.getObjectHandle("Path1")
path3=sim.getObjectHandle("Path2")
path4=sim.getObjectHandle("Path3")
--Definimos el objetivo
target = sim.getObjectHandle("TARGET")

--Definimos la velocidad
Vel=0.05

--Conexion con el Gripper
connection = sim.getObjectHandle('Franka_connection')
gripper = sim.getObjectChild(connection,0)
gripperName = "ROBOTIQ_85"

Cube = sim.getObjectHandle("CuboVerde")
Cube2 = sim.getObjectHandle("CuboNaranja")

Gripper = sim.getObjectHandle("ROBOTIQ_85")

--Establecer objeto dinamico
sim.setObjectInt32Parameter(Cube,3003,0)

function sysCall_threadmain()

--Definir rutas
sim.followPath(target,path,3,0,Vel,0.05)
sim.wait(1)
```

Threaded child script (Franka)

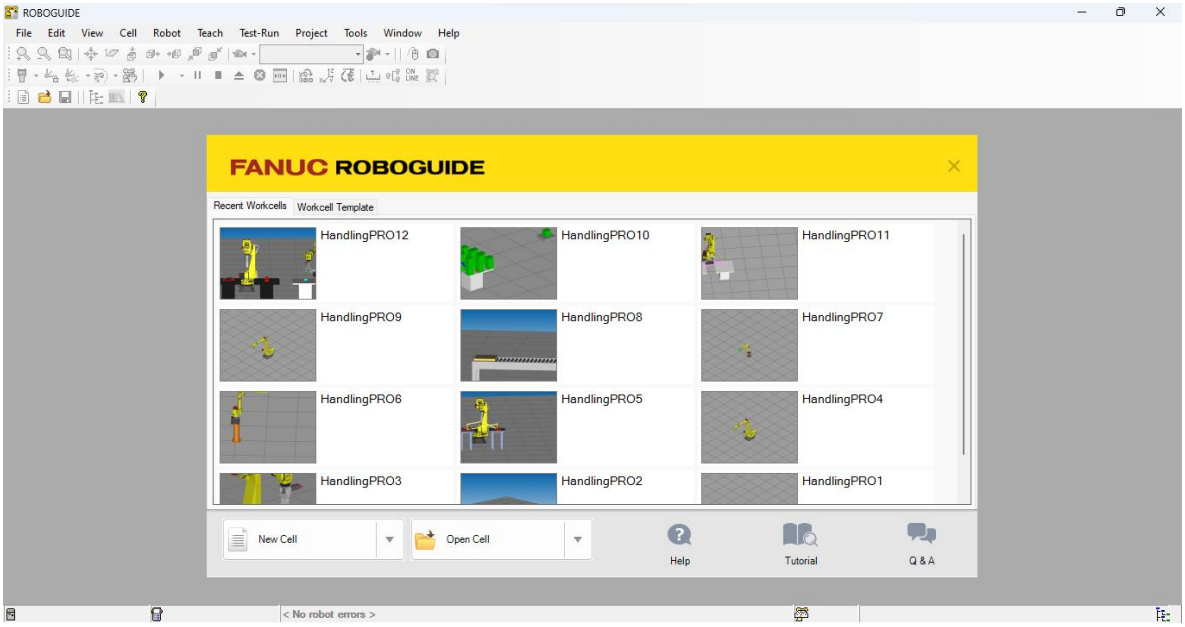
```
31
32  --Cerramos el gripper
33  sim.setIntegerSignal(gripperName..'_close',1)
34  sim.setObjectInt32Parameter(Cube,3003,1)
35  sim.resetDynamicObject(Cube)
36  sim.setObjectParent(Cube,Gripper,true)
37  sim.wait(2)
38  sim.followPath(target,path1,3,0,Vel,0.05)
39  sim.wait(2)
40  --El gripper suelta el objeto
41  sim.clearIntegerSignal(gripperName..'_close')
42  sim.setObjectInt32Parameter(Cube,3003,0)
43  sim.resetDynamicObject(Cube)
44  sim.setObjectParent(Cube,-1,true)
45
46  sim.wait(2)
47  --Sigue la ruta para obtener el nuevo cubo
48  sim.followPath(target,path2,3,0,Vel,0.05)
49  sim.wait(2)
50  --El gripper se cierra nuevamente
51  sim.setIntegerSignal(gripperName..'_close',1)
52  sim.setObjectInt32Parameter(Cube2,3003,1)
53  sim.resetDynamicObject(Cube2)
54  sim.setObjectParent(Cube2,Gripper,true)
55  sim.wait(2)
56  --Sigue la ruta para desechar el cubo recogido
57  sim.followPath(target,path3,3,0,Vel,0.05)
58  sim.wait(2)
59  --Suelta el cubo y regresa para esperar a más objetos
60  sim.clearIntegerSignal(gripperName..'_close')
61  sim.setObjectInt32Parameter(Cube2,3003,0)
62  sim.resetDynamicObject(Cube2)
63  sim.setObjectParent(Cube2,-1,true)
64  sim.wait(2)
65  sim.followPath(target,path4,3,0,Vel,0.05)
```

CONCLUSIONES

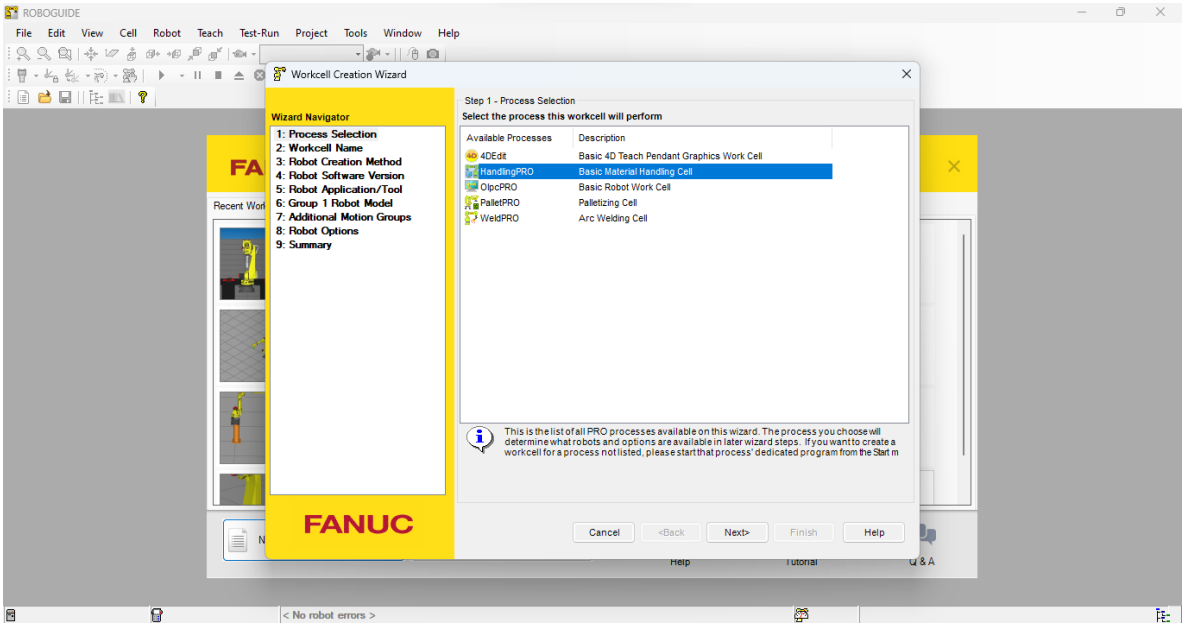
Se ha desarrollado una herramienta de desarrollo que cumple con todos los requisitos y por tanto permite al usuario realizar simulaciones complejas del código de su robot de forma sencilla y transparente. Luego de una etapa inicial de análisis, se eligió el simulador CoppeliaSim para realizar el trabajo, el cual resultó ser una herramienta muy completa y accesible, cuyas características permitieron cumplir con los requerimientos. Aunque se tuvo que recurrir a un cambio de versión por falta de herramientas en versiones actuales, permitió trabajar de forma rápida y sencilla con el brazo robotico Niryo On. Después de completar todo el trabajo, se puede argumentar que la elección fue la correcta. La herramienta de simulación desarrollada resuelve muchos de los problemas que causan los robots reales. Para hacer esto, los usuarios solo necesitan ejecutar el simulador CoppeliaSim y ver si nuestro robot sufre de colisiones o no, si están bien conectados o no, de forma que el usuario pueda inferir en problemas reales y sepa como arreglarlos con su simulación, esto facilitará el proceso de educación del usuario.

CRISTIAN JAIR TEPACH FONSECA

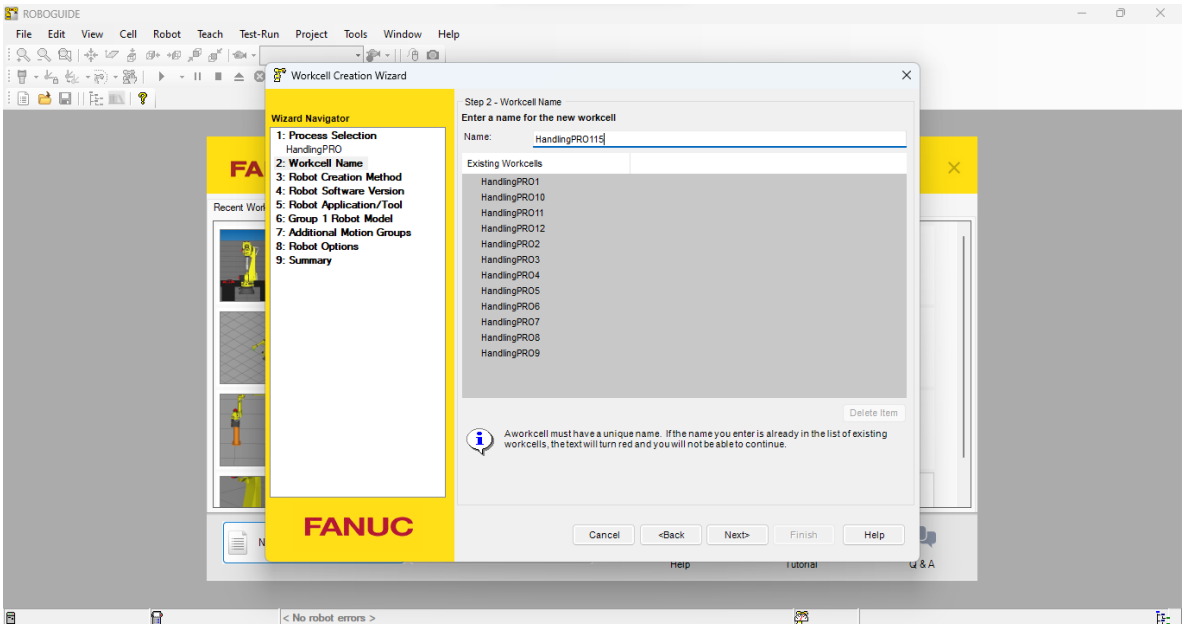
PASO 1: Abriremos el programa a utilizar, en este caso el software ROBOGUIDE, insertamos en NEW CELL para abrir una nueva pestaña.



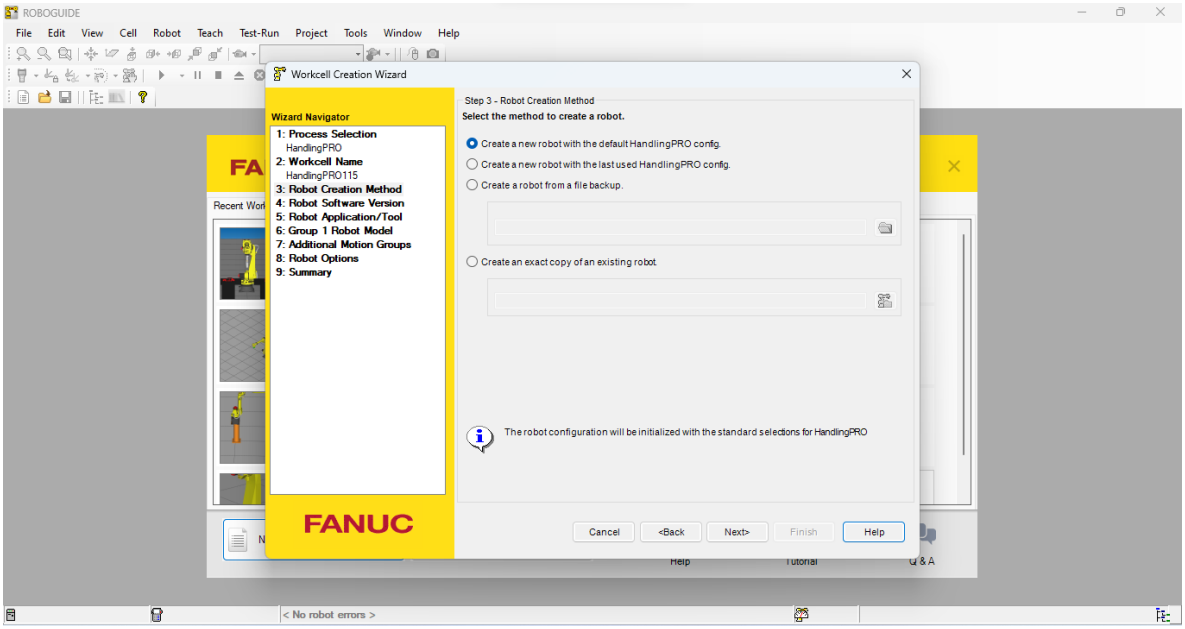
PASO 2: Seleccionamos la segunda opción que es HANDLINGPRO, para abrir una nueva pestaña.



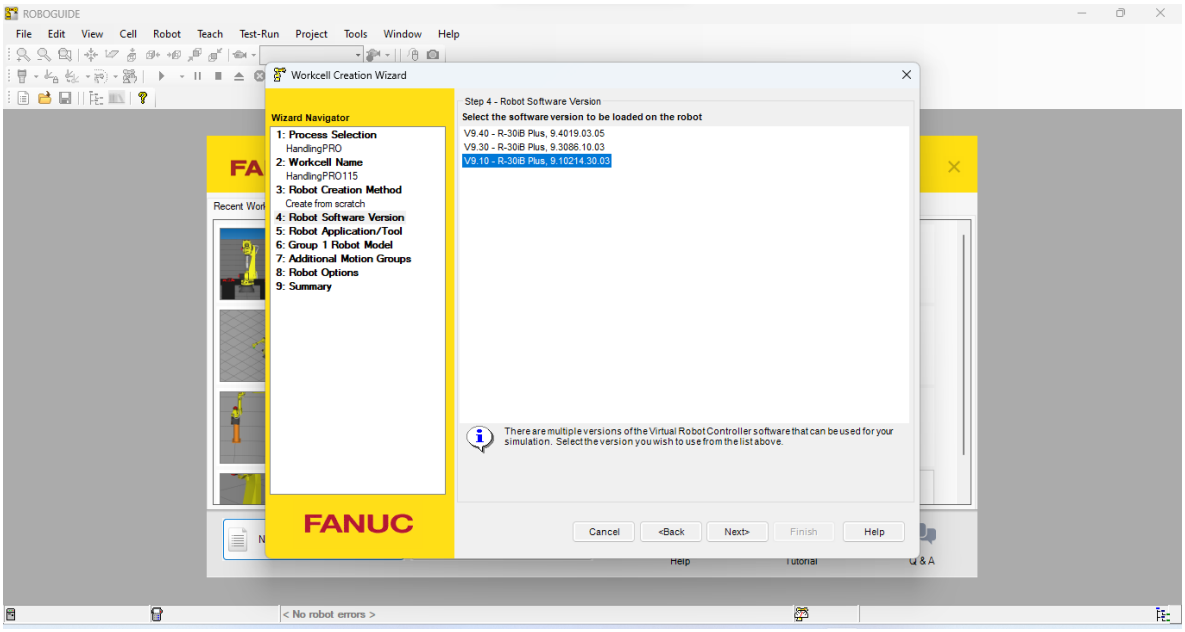
PASO 3: Le proporcionaremos un nombre a nuestra celda de trabajo.



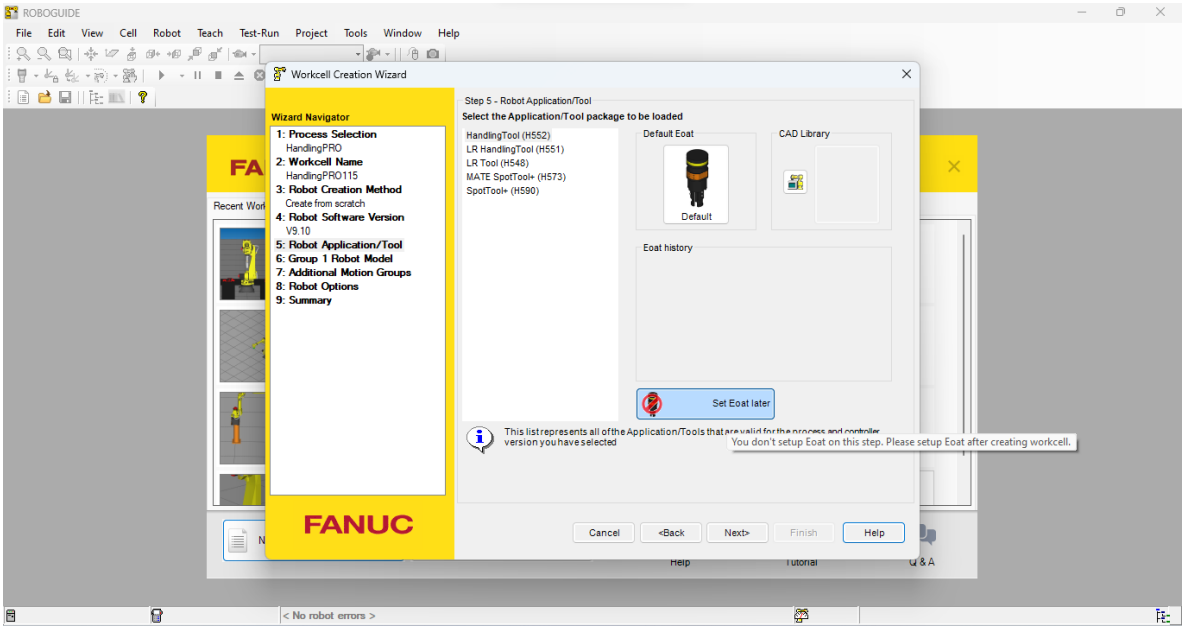
PASO 4: Crearemos un robot desde cero con nuestras configuraciones desde cero.



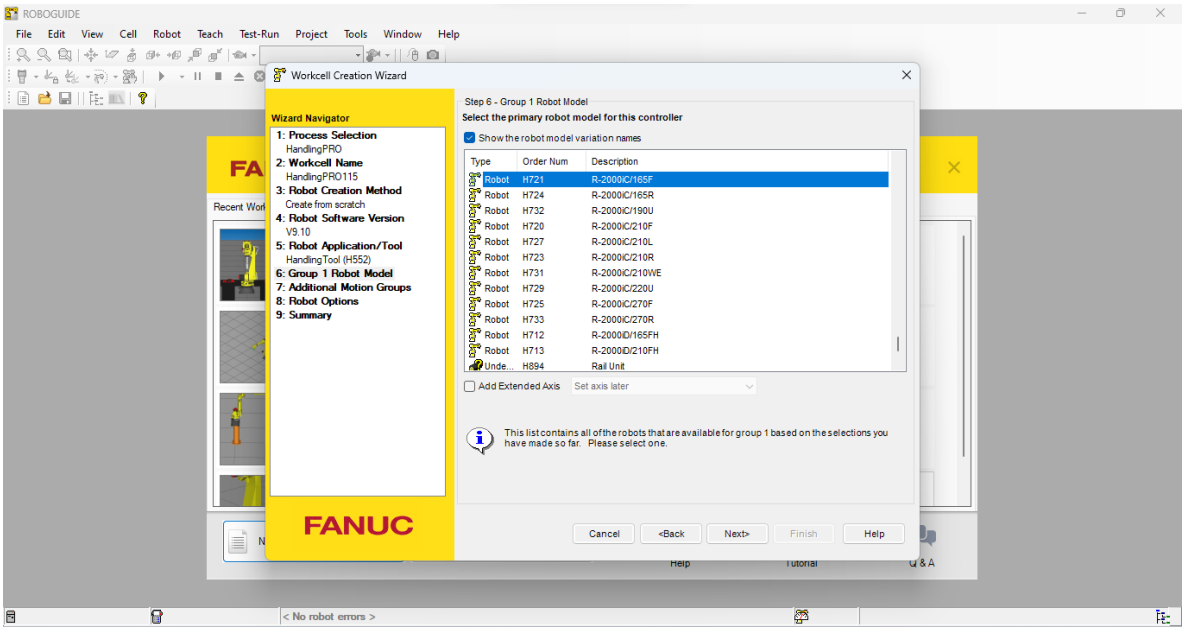
PASO 5: Se utilizará la versión que sea más conveniente para el usuario que esta utilizando el programa.



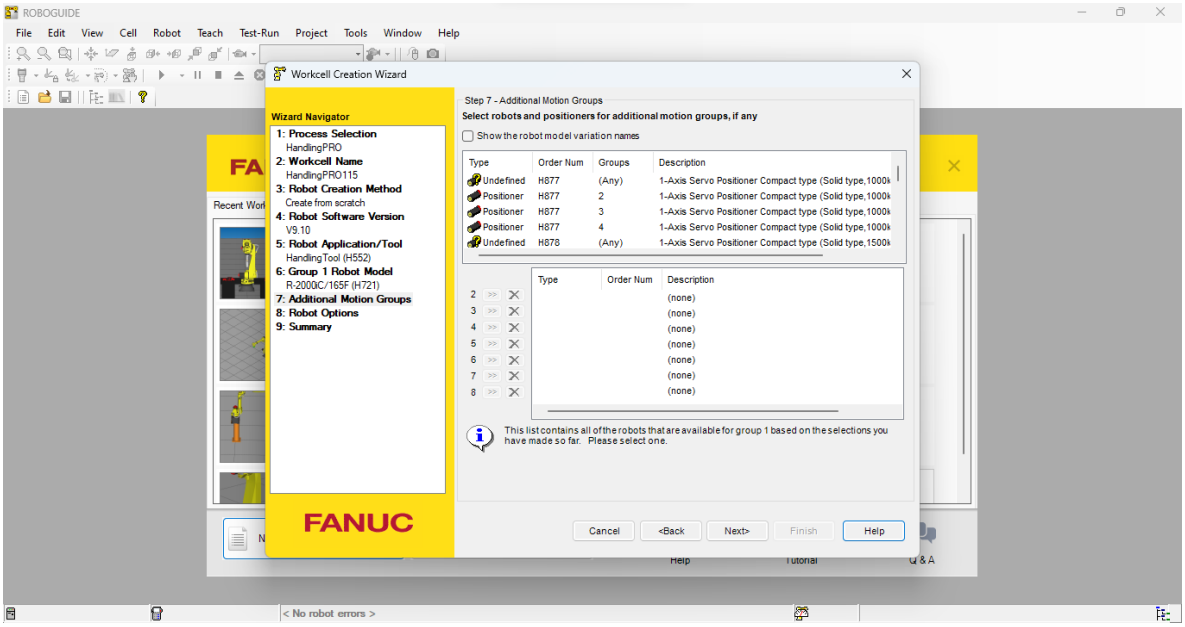
PASO 6: Posteriormente utilizaremos un brazo robótico sin gripper, ya que mas adelante se le adoptara uno de acuerdo a las necesidades y funciones que se requieran.



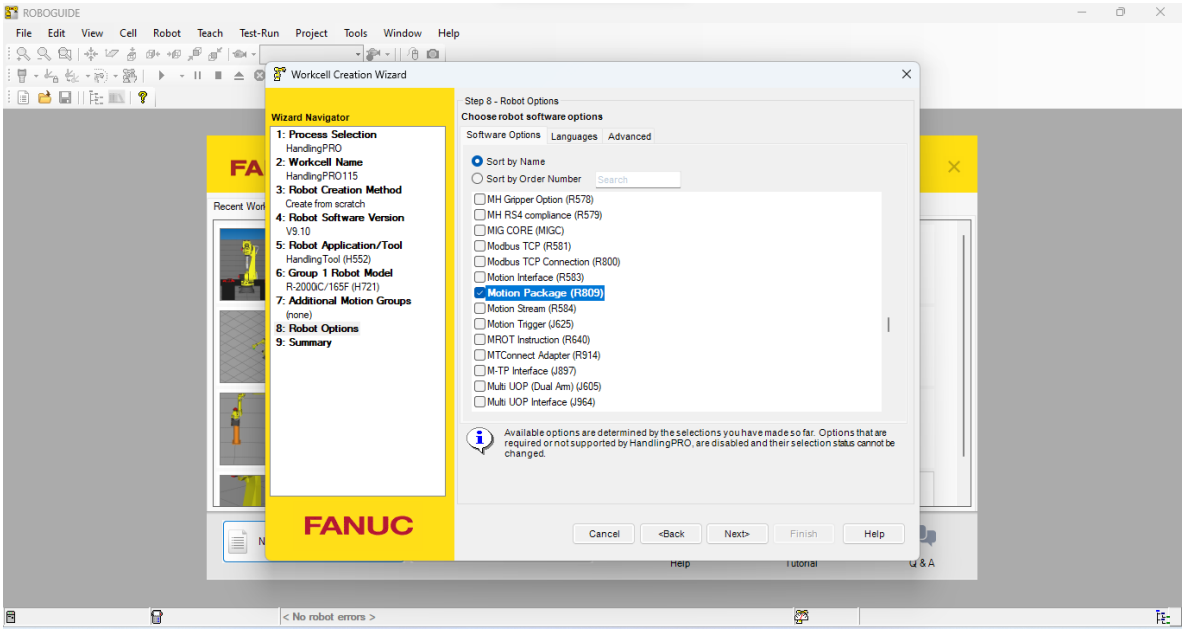
PASO 7: Existen múltiples brazos robóticos es por ello que para la función que necesitamos utilizaremos el modelo que se muestra en la imagen



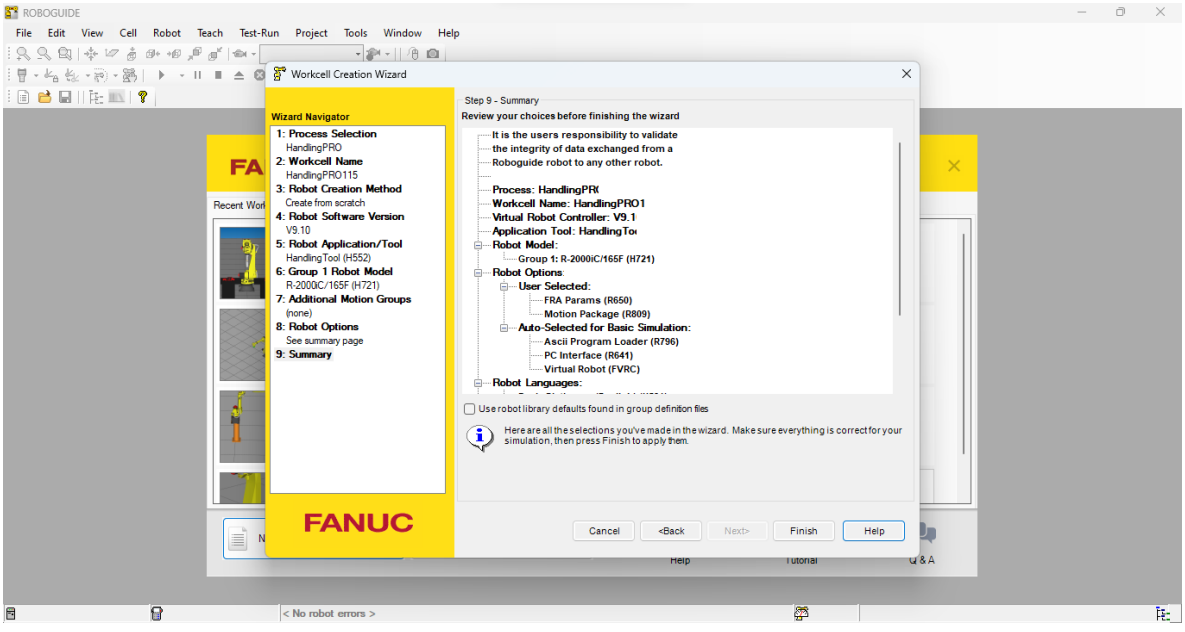
PASO 8: En este paso se puede omitir ya que es solo agregar un movimiento extra al brazo de igual manera se puede colocar más adelante del proceso.



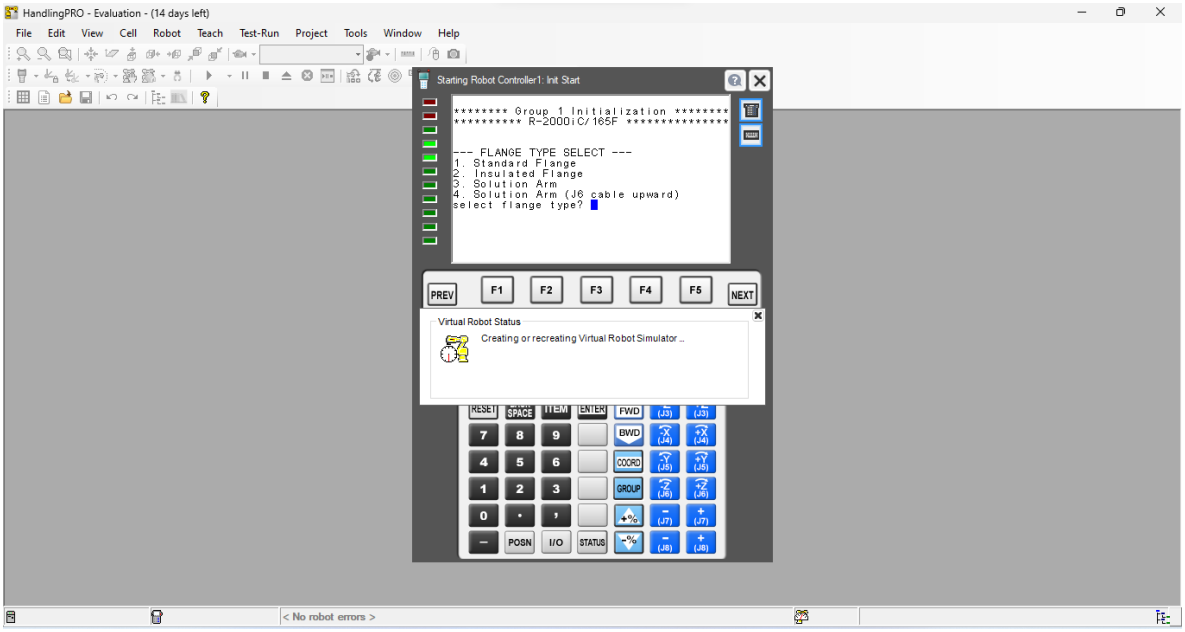
PASO 9: Para este caso es solo cuestión de agregar librerías, si al inicio no se agregan mas adelante se pueden agregar y no afecta el funcionamiento del trabajo



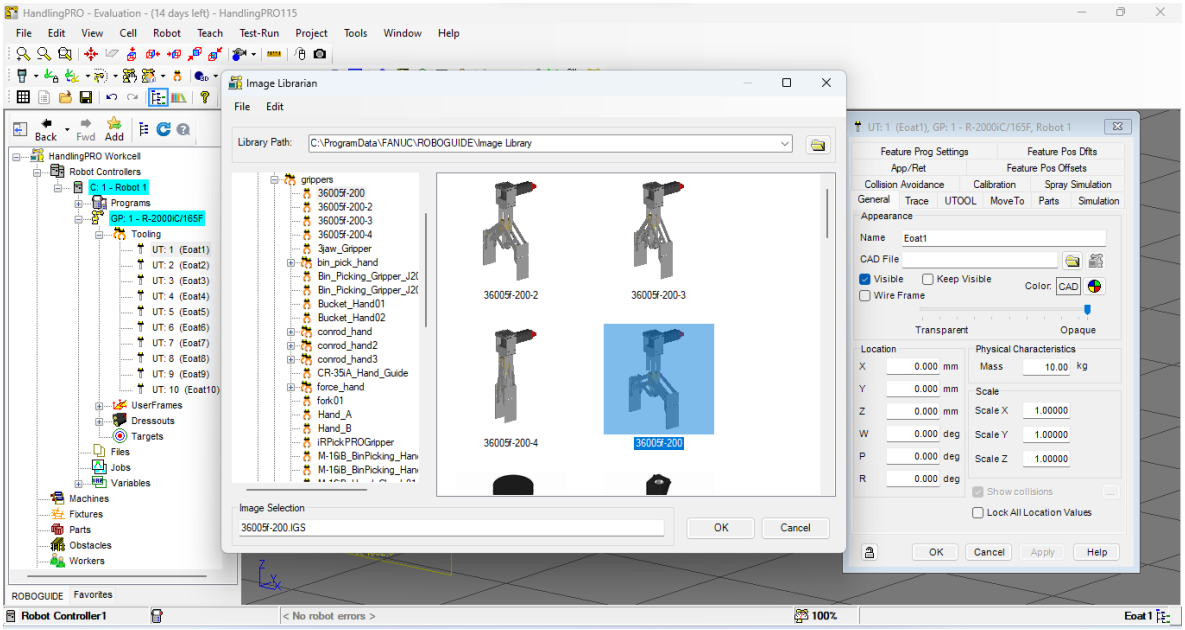
PASO 10: En esta pantalla se cargará un resumen del robot que ocuparemos en este software.



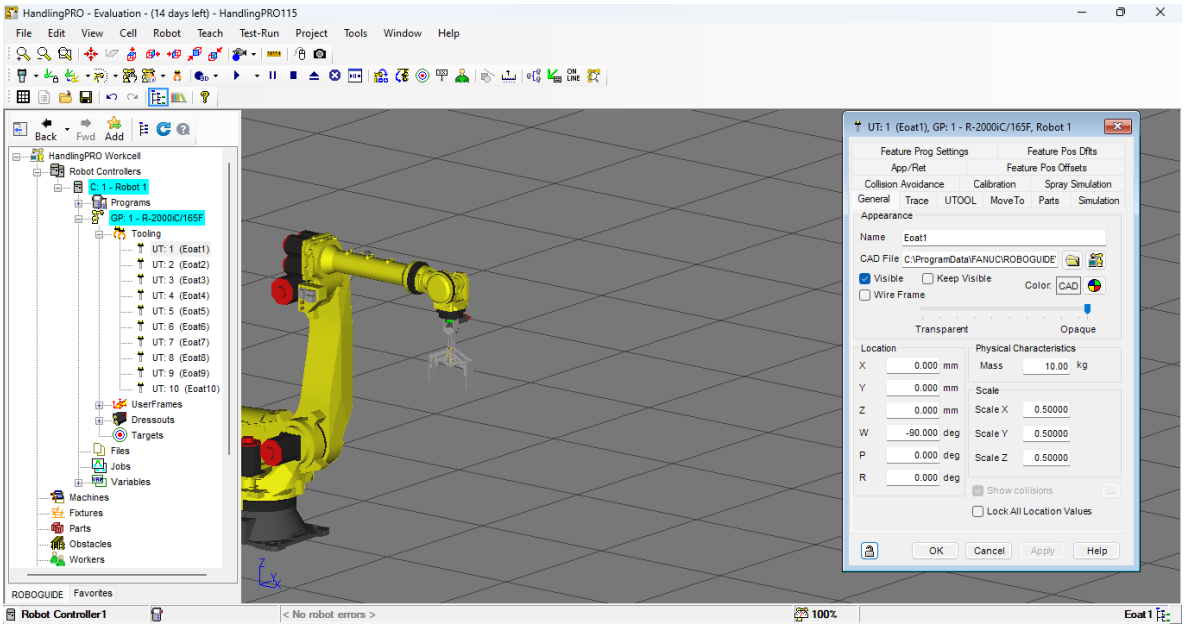
PASO 11: Procedemos a Configurarlo



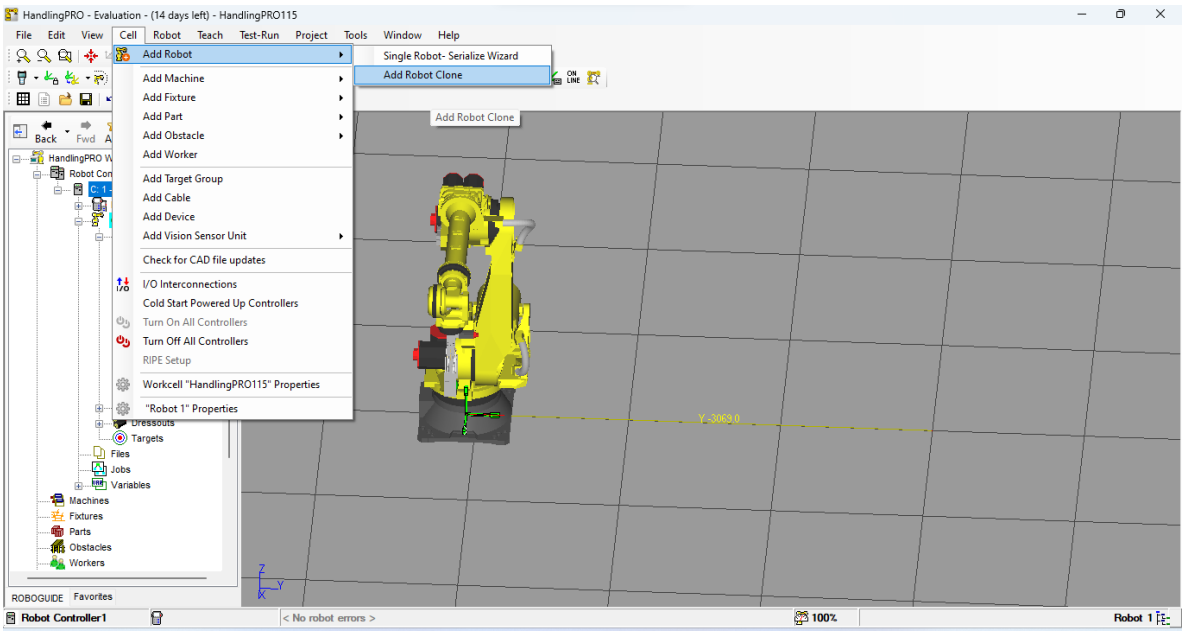
PASO 12: Una ves ingresado a la pantalla de inicio, procederemos a agregar un gripper a nuestro brazo robótico como se muestra en la siguiente imagen.



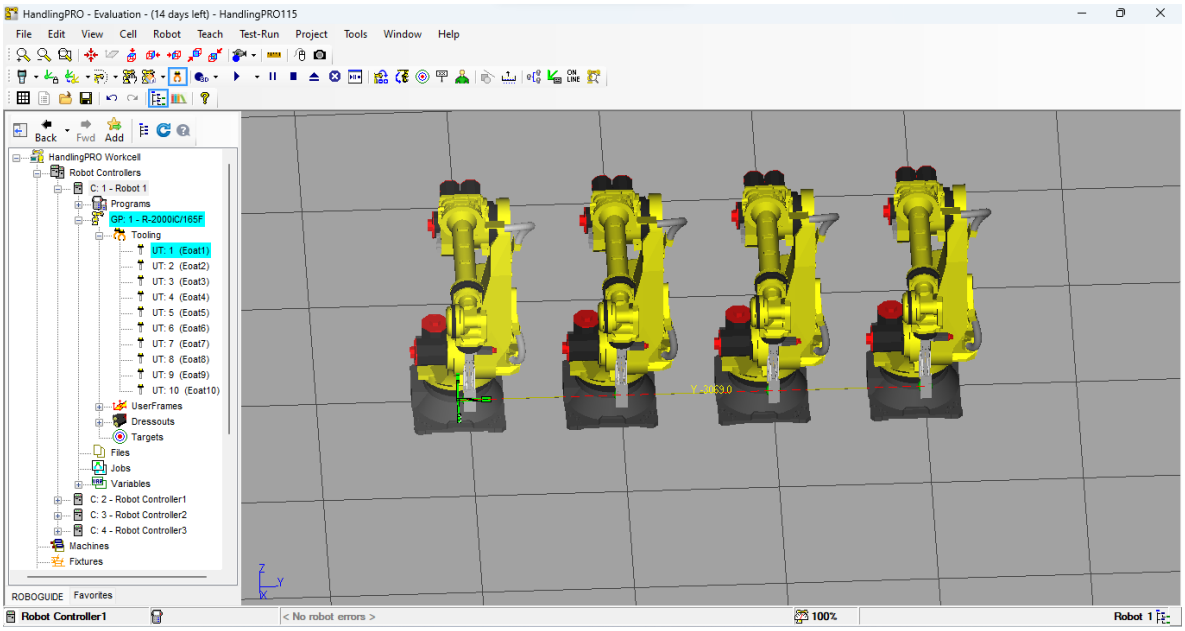
PASO 13: Procedemos a configurara nuestro gripper y hacer todas las configuraciones necesarias.



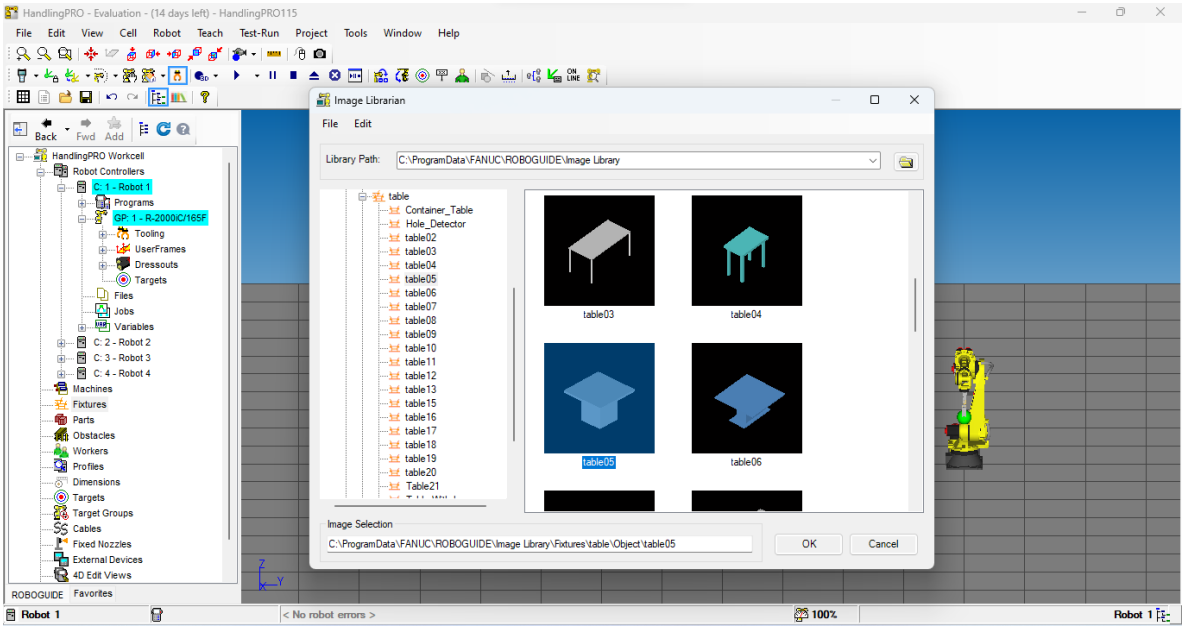
PASO 14: Procederemos a copiar nuestro brazo 3 veces más para nuestro trabajo.



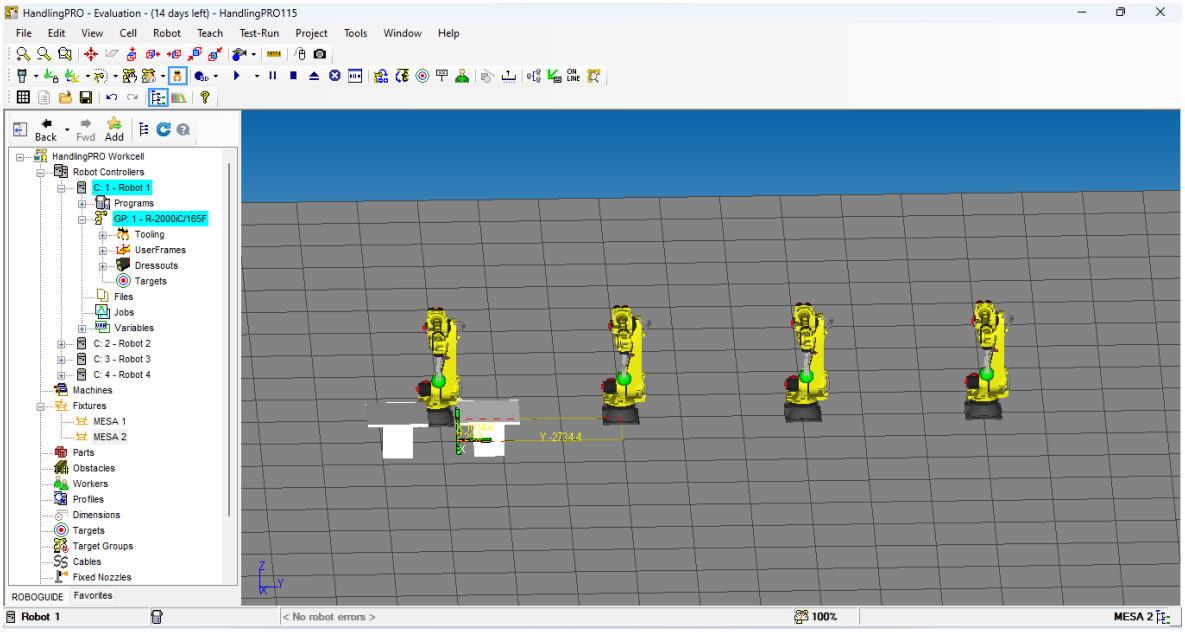
PASO 15: Procedemos a separar cada brazo a una cierta distancia y configurar cada uno pro si solo para que trabajen por separados.



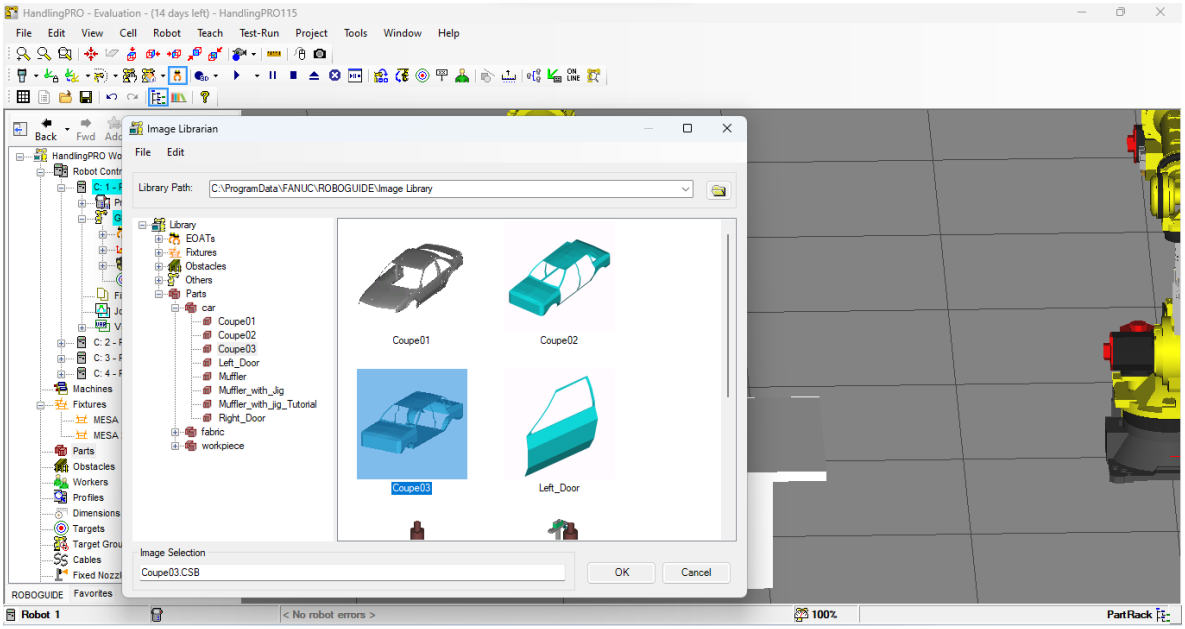
PASO 16: Colocaremos 8 mesas para colocar nuestro objecto encima y lo pueda recoger el brazo robótico



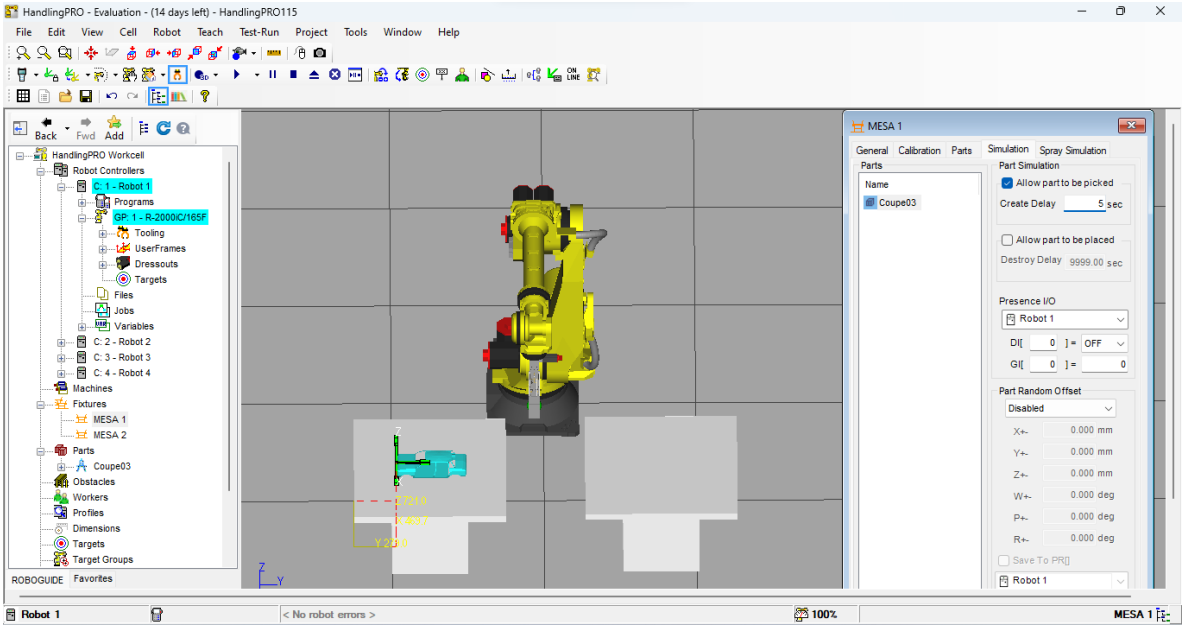
PASO 17: Se colocarán dos mesas por brazo robótico para su ejecución.



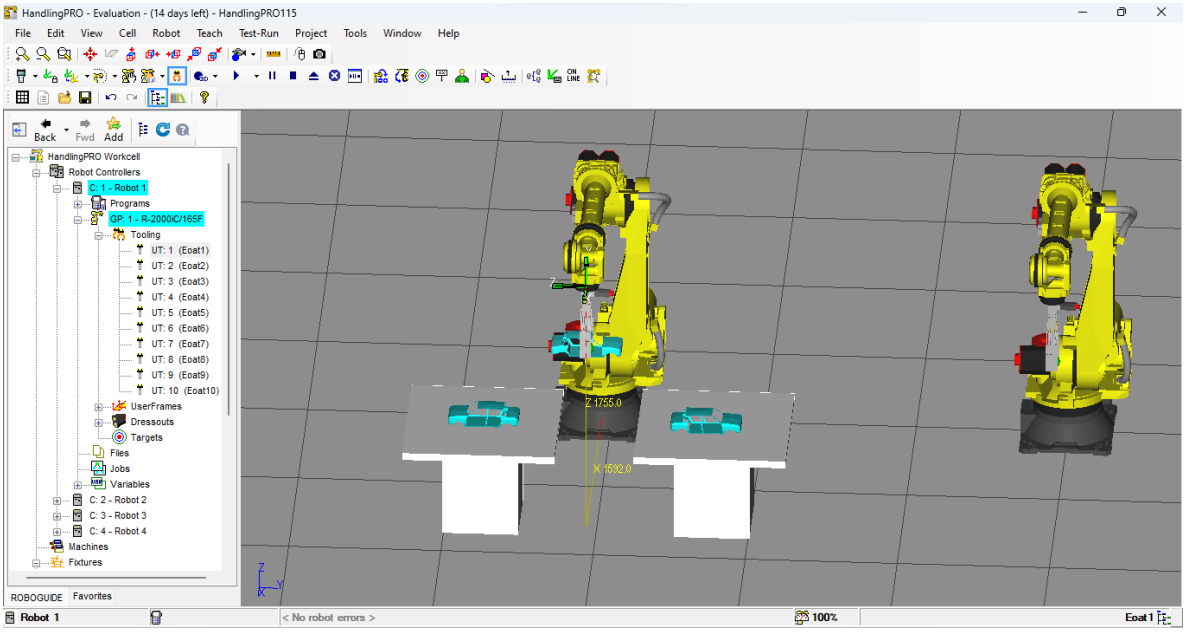
PASO 18: Se hará la simulación de una planta de coches por lo tanto cada brazo tendrá su propia función para su armado de este mismo.



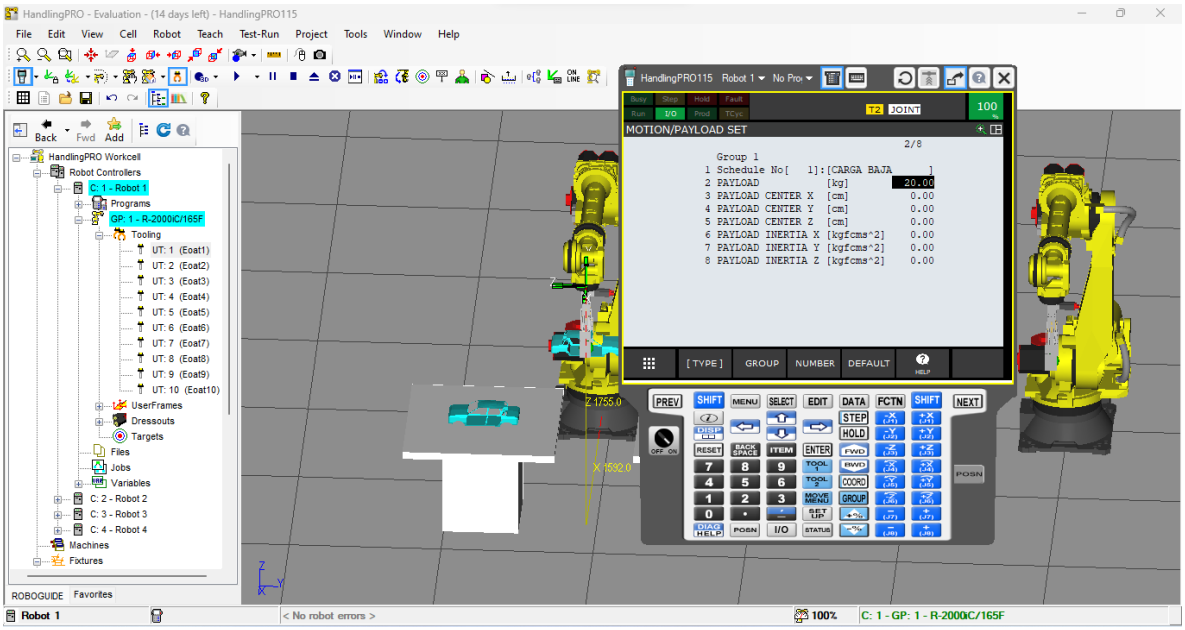
PASO 19: Agregamos la parte del carro a la mesa, se coloca la simulación a 5 segundos y se procede a mover al otro ángulo en la otra mesa.

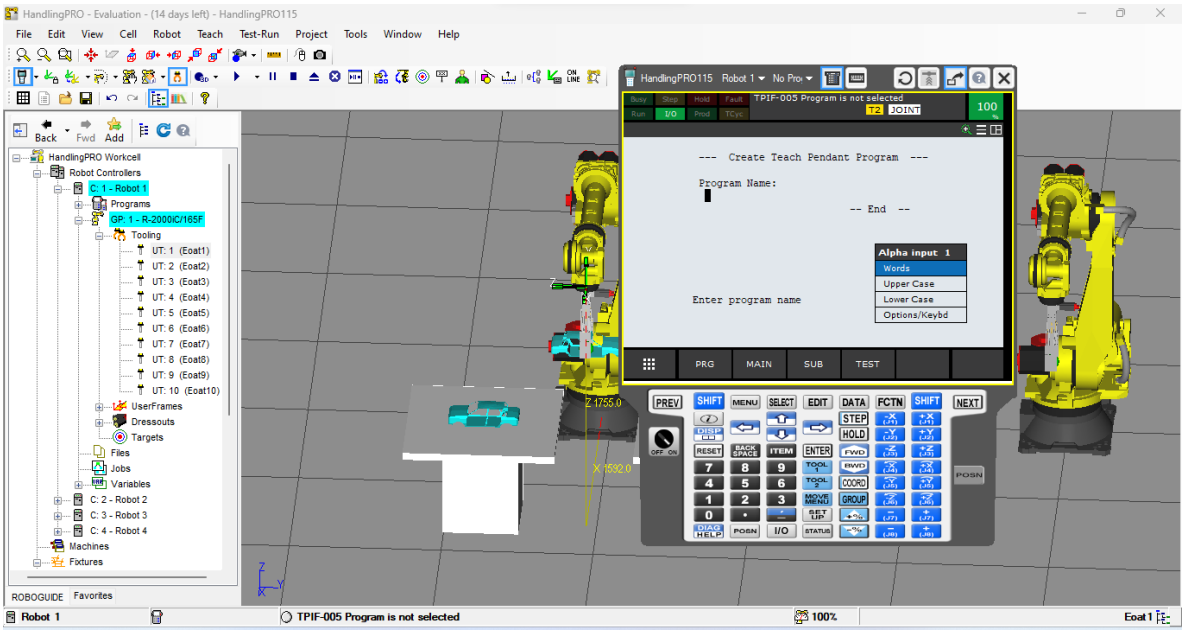


PASO 20: Una ves colocado las posiciones del auto se procede a el registro de posiciones y a la programación de este mismo para su simulación

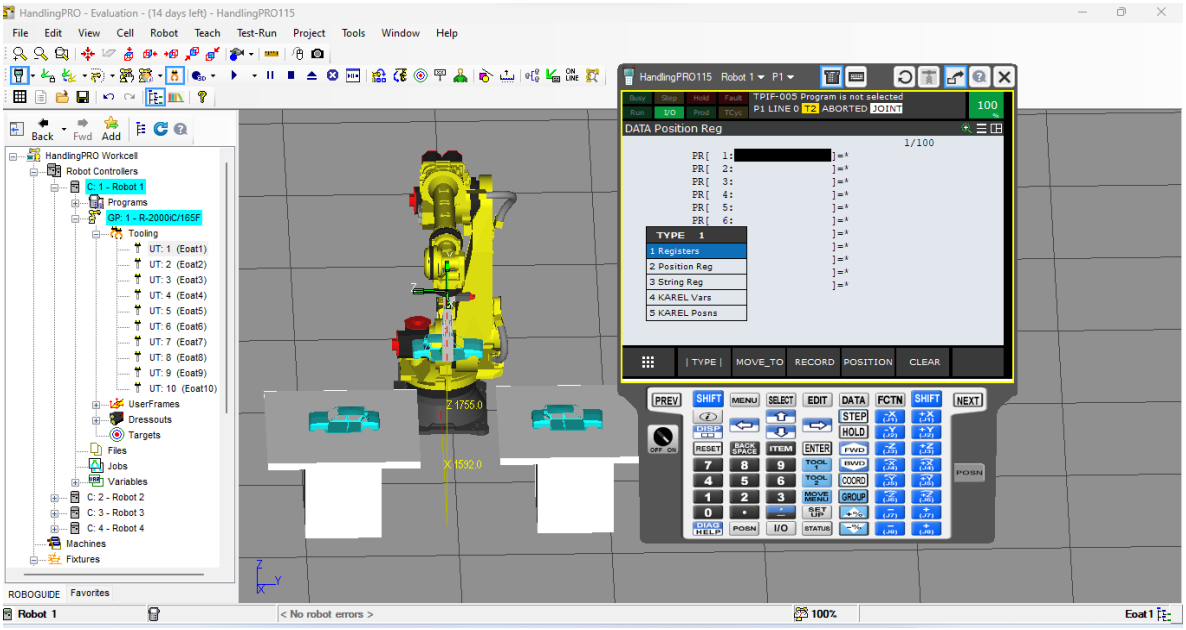


PASO 21: Abriremos el **Teach Pendal**, damos clic en detail, ingresamos detalles y posteriormente la carga limitante que aguantará el robot que es 20kg.

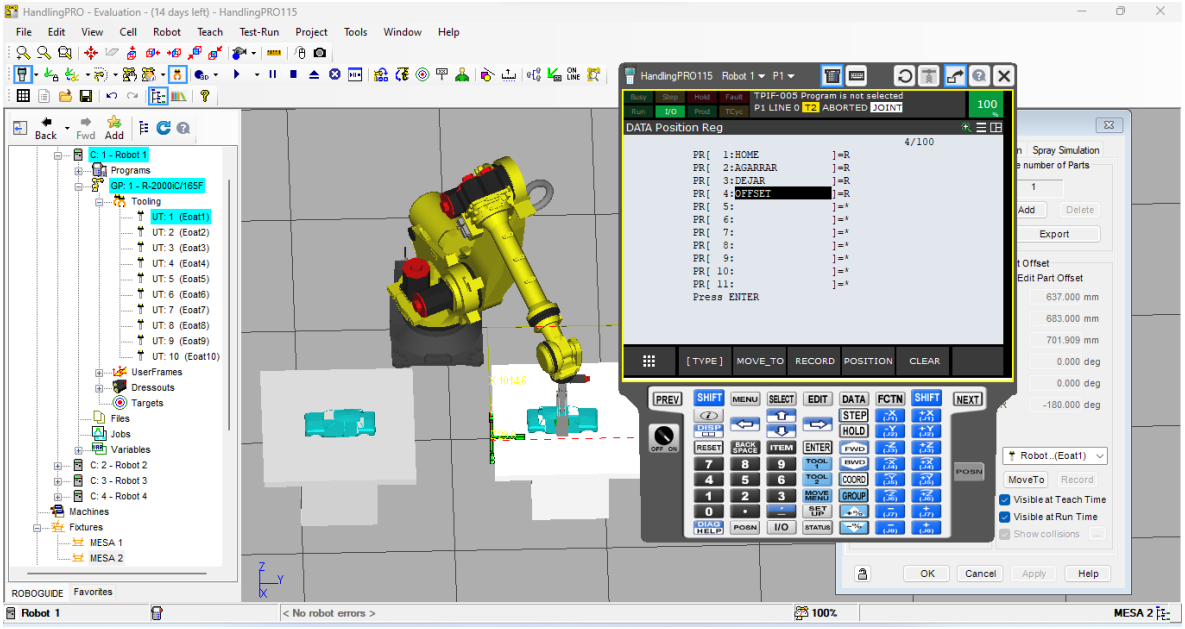




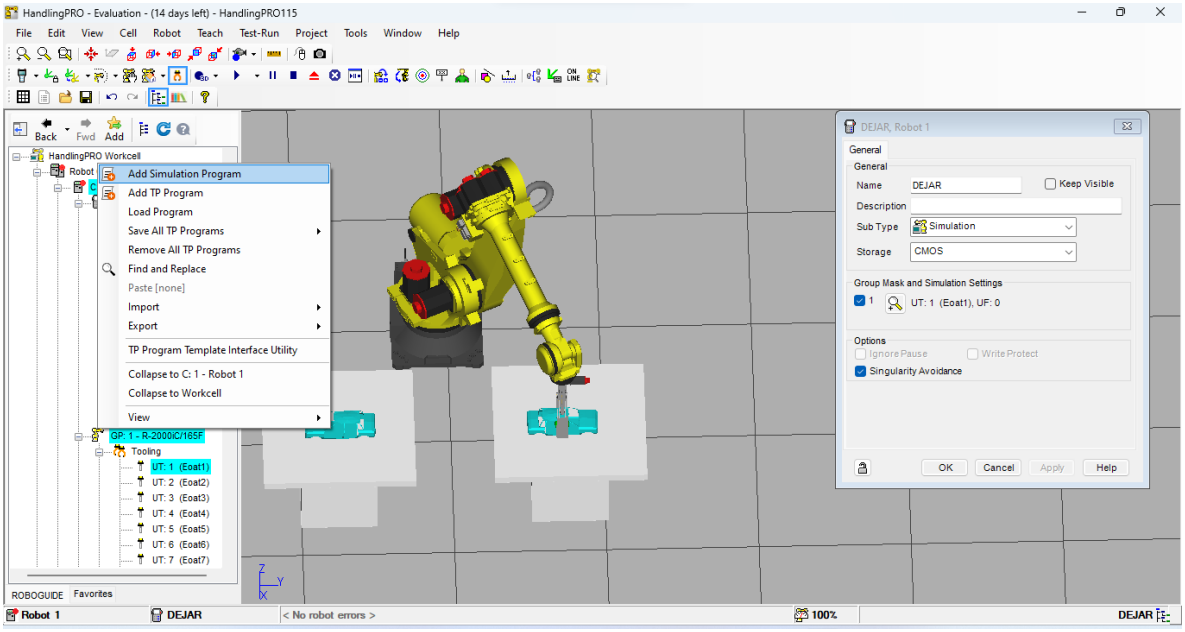
PASO 22: Colocamos posiciones de registro para el movimiento del robot.



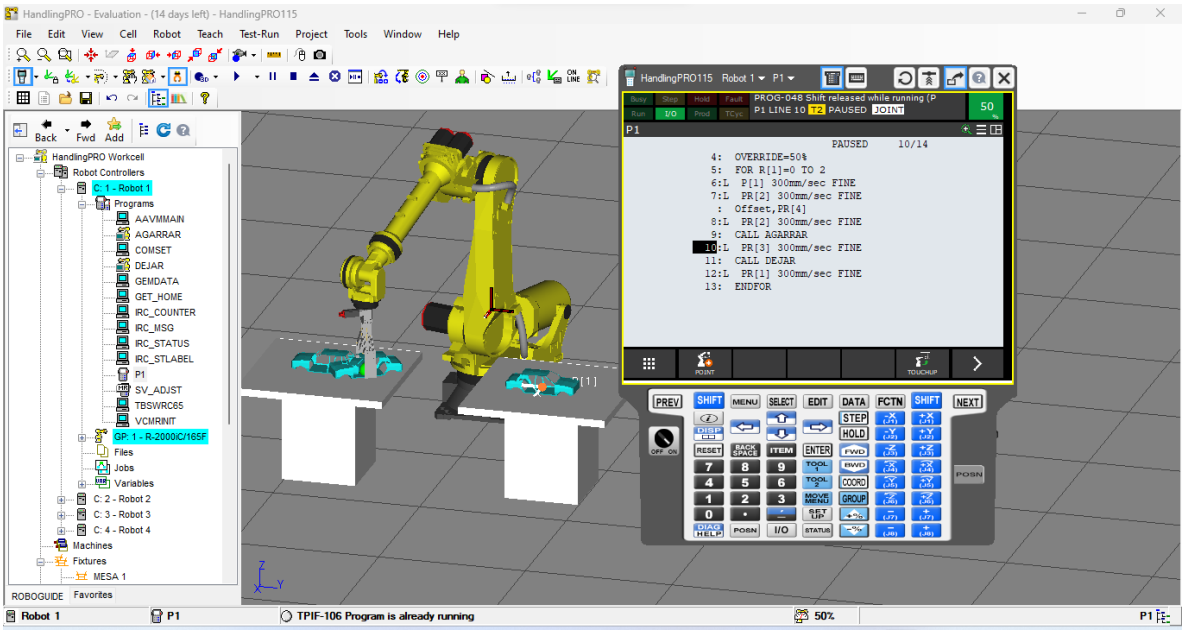
Quedaron registradas las posiciones de nuestro robot, es decir, los movimientos de este mismo.



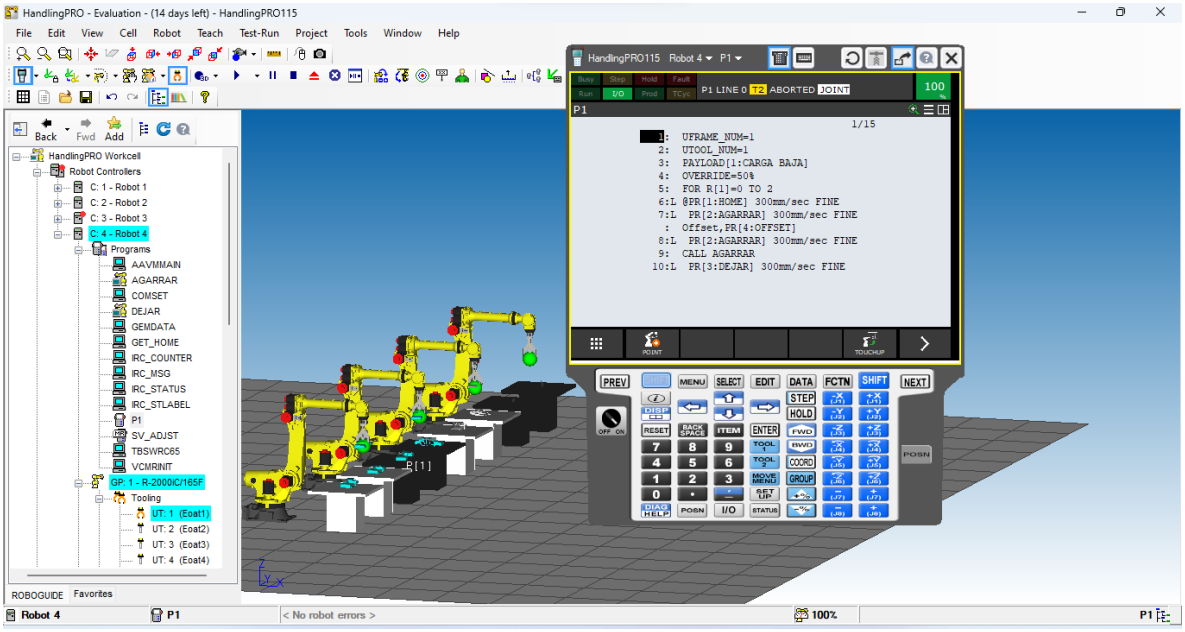
PASO 23. Se configura la simulación de agarrar y dejar.

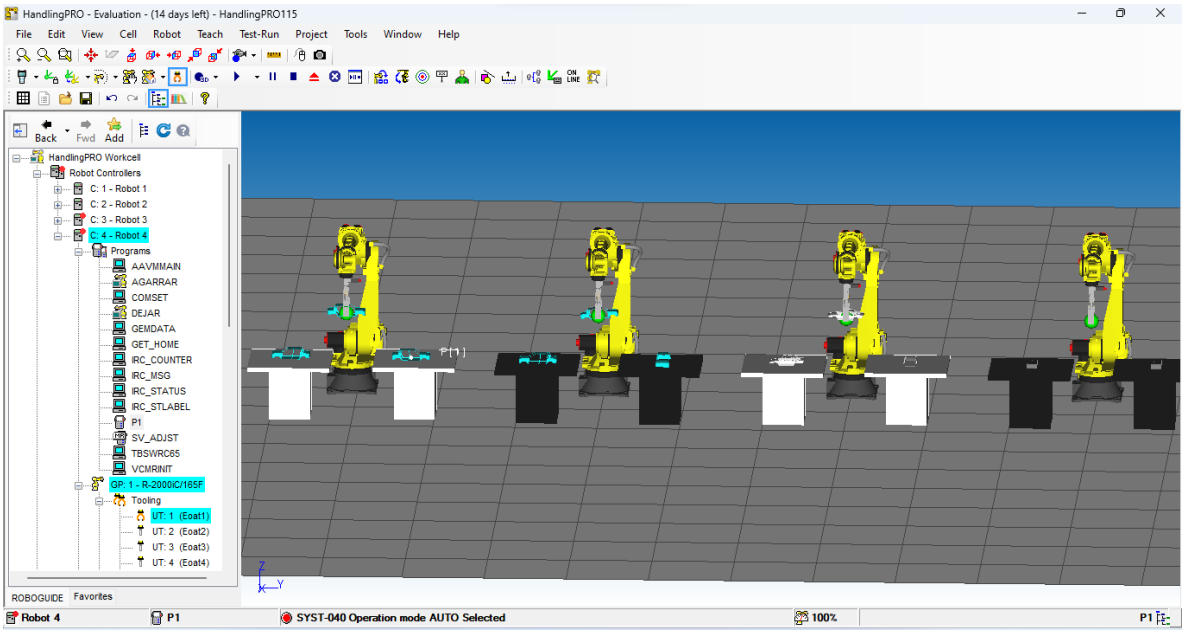


PASO 24: Una vez terminado el primer brazo los demás son con los mismos datos y frecuencias.



PASO 25: Una vez repetido el mismo proceso para los brazos restantes. Se procede a ejecutar todos al mismo tiempo y así es como se puede mostrar la simulación de cómo se ensamblan autos.


HandlingPRO - Evaluation - (14 days left) - HandlingPRO115
File Edit View Cell Robot Teach Test-Run Project Tools Window Help
HandlingPRO115 Robot 4 P1
P1 LINE 0 T2 ABORTED JOINT
100
P1
1/15
1: UFRAME_NUM=1
2: UTOOL_NUM=1
3: PAYLOAD[1:CARGA BAJA]
4: OVERRIDE=50%
5: FOR R[1]=0 TO 2
6:L @PR[1:HOME] 300mm/sec FINE
7:L PR[2:AGARRAR] 300mm/sec FINE
: Offset,PR[4:OFFSET]
8:L PR[2:AGARRAR] 300mm/sec FINE
9: CALL AGARRAR
10:L PR[3:DEJAR] 300mm/sec FINE
Back Fwd Add
HandlingPRO Workcell
Robot Controllers
C: 1 - Robot 1
C: 2 - Robot 2
C: 3 - Robot 3
C: 4 - Robot 4
Programs
AAVMMAIN
AGARRAR
COMSET
DEJAR
GEMDATA
GET_HOME
IRC_COUNTER
IRC_MSG
IRC_STATUS
IRC_STLABEL
P1
SV_ADJST
TBSWRC65
VCMRINIT
GP: 1 - R-2000iC/165F
Tooling
UT: 1 (Eoat1)
UT: 2 (Eoat2)
UT: 3 (Eoat3)
UT: 4 (Eoat4)
ROBOGUIDE Favorites
Robot 4 P1 < No robot errors > 100% P1


HandlingPRO - Evaluation - (14 days left) - HandlingPRO115
File Edit View Cell Robot Teach Test-Run Project Tools Window Help
Back Fwd Add
HandlingPRO Workcell
Robot Controllers
C: 1 - Robot 1
C: 2 - Robot 2
C: 3 - Robot 3
C: 4 - Robot 4
Programs
AAVMMAIN
AGARRAR
COMSET
DEJAR
GEMDATA
GET_HOME
IRC_COUNTER
IRC_MSG
IRC_STATUS
IRC_STLABEL
P1
SV_ADJST
TBSWRC65
VCMRINIT
GP: 1 - R-2000iC/165F
Tooling
UT: 1 (Eoat1)
UT: 2 (Eoat2)
UT: 3 (Eoat3)
UT: 4 (Eoat4)
ROBOGUIDE Favorites
Robot 4 P1 SYST-040 Operation mode AUTO Selected 100% P1

ISMAEL ZACARIAS SINTA

INTRODUCCIÓN.

El desarrollo de la actividad que consistía en simular un brazo robótico realizando 5 movimientos mediante software de simulación, se realizo en con el programa de Free CAD gracias a su interfaz fácil de usar. El brazo robótico que se utilizo como modelo de simulación es KUKA 210 el cual es más robusto en comparación de otros modelos y tiene usos industriales mas pesados. La simulación que se presenta se llevo a cabo a través de marcar puntos en el espacio con los cuales se traza una trayectoria que el robot sigue, por supuesto que, el robot da la opción de cambiar diferentes velocidades entre puntos. La simulación que se realizo consta de 7 diferentes puntos que conforman la trayectoria que el robot debe seguir, sin embargo, el robot no cuenta con alguna herramienta extra, simplemente es el brazo

robótico realizando un movimiento complejo.

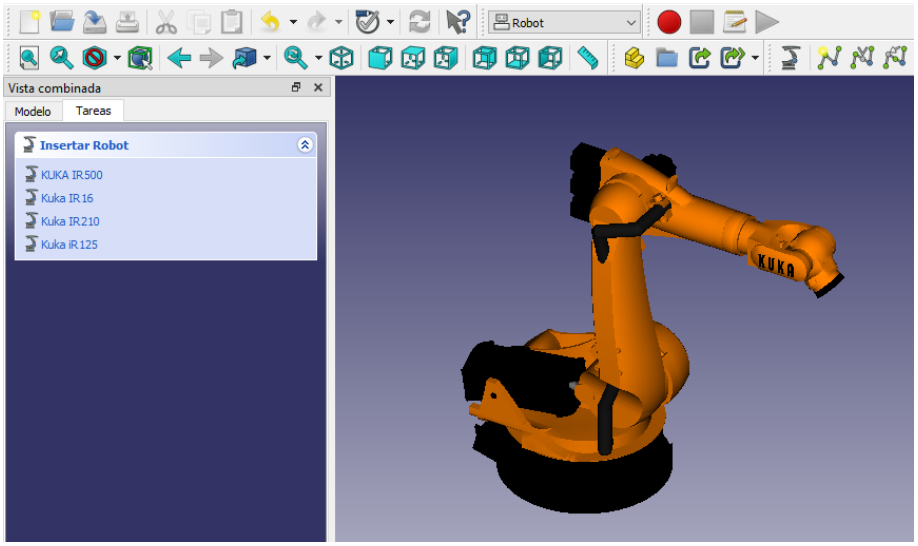
Realización de la simulación.

Paso 1

El primer paso para la realización de la simulación fue la selección de un software para simular un brazo robótico. En el mercado hay muchas opciones, sin embargo, se utilizó un software libre que en este caso es Free CAD.

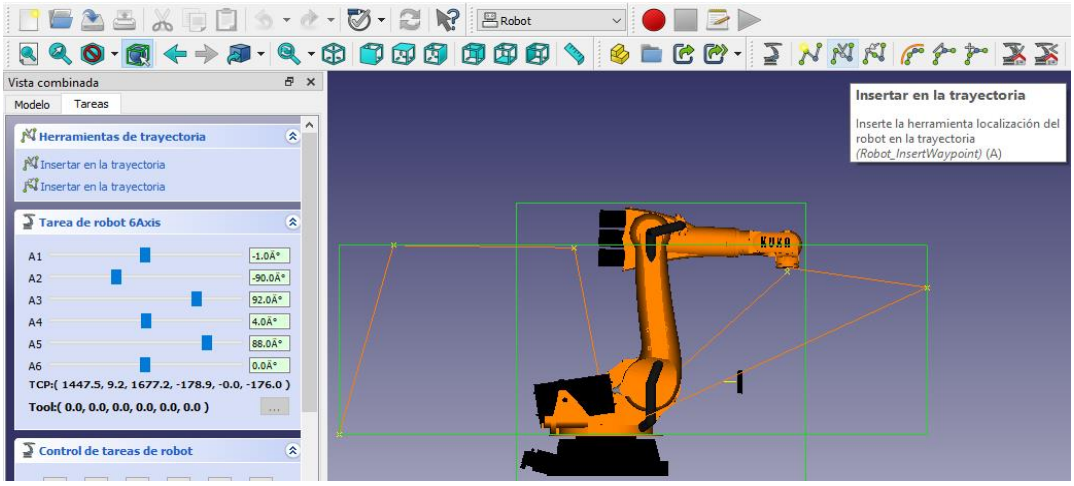
Paso 2

una vez que se analizó el software y se decide implementar, se examina la lista de ejemplares que el programa proporciona para seleccionar el robot que se desea simular. Para este caso se seleccionó el robot kuka 210.



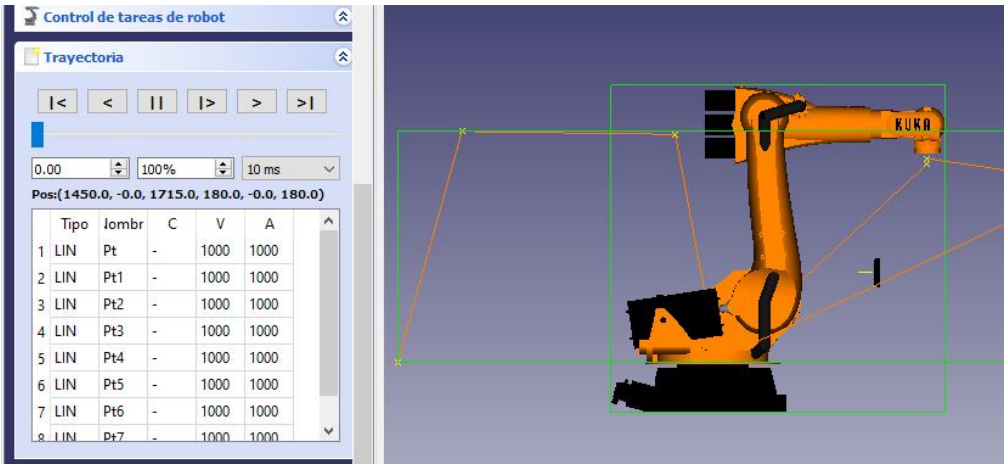
Paso 3

Con la selección del robot podemos empezar a manipularlo, el programa de Free CAD permite manipular la máquina de forma manual. Con el control manual se puede crear un movimiento a través del trazado de trayectoria marcando diferentes posiciones del robot, posiciones las cuales el robot repetirá de manera seguida recreando así la simulación deseada.



Paso 4

el paso final es la simulación. Para ello se selecciona el icono de "*simular una trayectoria*" la cual hará que se muestre un apartado para reproducir y detener la simulación del brazo robótico, en la cual para ver la simulación solo seleccionaremos "*play*".



CONCLUSIÓN.

Free CAD es un software completo que a pesar de ser de uso libre tiene una interfaz sencilla, además de ser didacta e interactiva con el usuario, no solo se limita a los robots que se muestra si no que te da la opción de descargar más robots del agrado del usuario. La programación del movimiento del robot no fue compleja como en otros programas, personas principiantes pueden experimentar y e interactuar con la plataforma. Los robots tienen excelentes gráficas y sus movimientos son fluidos para el poco espacio que consume el programa. Free CAD como simulador es más que excelente con relación a lo que pagas y a lo que ofrece. La simulación es un conjunto de movimientos que fueron fáciles de realizar ya que son movimientos aleatorios, sin embargo, aunque el programa es muy bueno tiene defectos y esto es que las simulaciones son limitadas, pero funciona para actividades pequeñas.

VIDEOS

Las simulaciones mostradas en el reporte se pueden ver en el siguiente enlace:

https://drive.google.com/file/d/1SEEwDaC2Miw2wxxbP6J3I4O9vHnFcZ0e/view?usp=share_link