UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Desarrollo de un gemelo digital correspondiente a una célula robotizada para el etiquetado, empaquetado y paletizado de botellas mediante RoboDK

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

AUTOR/A: Olleta Beaus, Ignacio

Tutor/a: Vallés Miquel, Marina

CURSO ACADÉMICO: 2023/2024

AGRADECIMIENTOS

Me gustaría comenzar la redacción de este proyecto dándole las gracias a todas aquellas personas que, de forma directa o indirecta, han contribuido a la realización de este trabajo. Agradecer especialmente a mi tutora Marina Vallés Miquel por su paciencia y dedicación a la hora de guiarme y alentarme durante todo este tiempo.

También me gustaría que se reconociera en este documento la inestimable ayuda del profesor Stefano Seriani que, durante mi estancia en Italia, siempre se ha mostrado dispuesto a ayudar y apoyar este proyecto.

RESUMEN

En este proyecto se realiza la automatización de una planta de etiquetado, empaquetado y paletizado de un conjunto de botellas. Las botellas están colocadas en cintas transportadoras y mediante el uso de robots son etiquetadas, puestas en cajas y colocadas sobre palés de la forma más optimizada posible. La planta se ha simulado, programado y automatizado a través del programa RoboDK.

RESUM

En este projecte es realitza l'automatització d'una planta d'etiquetatge, empaquetat i paletitzat d'un conjunt de botelles. Les botelles estan col·locades en cintes transportadores i mitjançant l'ús de robots són etiquetades, posades en caixes i col·locades sobre palets de la forma més optimitzada possible. La planta s'ha simulat, programat i automatitzat a través del programa RoboDK.

ABSTRACT

This project involves the automation of a labelling, packaging, and palletizing plant for a set of bottles. The bottles are placed on conveyor belts and using robots they are labelled, placed in boxes and placed on pallets in the most optimised way possible. The plant has been simulated, programmed and automated through the RoboDK program.

ÍNDICE

DOCUMENTOS CONTENIDOS EN EL TFG

- Memoria
- Presupuesto
- Pliego de condiciones

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

ÍNDICE DEL PLIEGO DE CONDICIONES

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN GEMELO DIGITAL CORRESPONDIENTE A UNA CÉLULA ROBOTIZADA PARA EL ETIQUETADO, EMPAQUETADO Y PALETIZADO DE BOTELLAS MEDIANTE ROBODK.

MEMORIA

ÍNDICE DE LA MEMORIA

1 RESUMEN EJECUTIVO

En la industria moderna el mundo de la automatización ofrece una amplia gama de posibilidades y ventajas productivas. La implementación de la robótica industrial en el mundo empresarial supone un avance en materia de seguridad y eficiencia para la mayoría de compañías. Dentro de la automatización de procesos, la generación de gemelos digitales permite solucionar, de forma virtual, problemas relacionados con la gestión del tiempo y recursos, garantizando, además, la seguridad de los trabajadores y el ahorro energético.

En este proyecto se ha intentado generar un gemelo digital del proceso de etiquetado, empaquetado y paletizado de un conjunto de botellas empleado en la empresa "Baxter Laboratories" compañía dedicada a la producción y distribución de servicios para el cuidado de la piel, protección solar y productos farmacéuticos de uso cutáneo con sede en Melbourne, Australia

Los objetivos principales del proyecto han sido: la mejora en la automatización de la planta robótica en la que se ha inspirado este proyecto a partir de las posibilidades que ofrece RoboDk y, además, el estudio y optimización del tiempo de ciclo del proceso desarrollado.

Cuando se estuvo analizando las posibles soluciones al problema planteado se tuvo claro que la opción más viable era la de desarrollar un gemelo digital puesto que, la situación "a distancia" que conllevaba el programa Erasmus, limitaba mucho las posibilidades de trabajar con maquetas o células reales.

Una vez tomada esta decisión, se optó por el software RoboDk para la realización del gemelo digital, puesto que cuenta con un grandísimo potencial en el mundo de la ingería actual permitiendo el diseño, programación y simulación de cualquier tipo de planta robótica industrial de manera sencilla e intuitiva. Además, dicho software cuenta la posibilidad de usarse con una versión gratuita, muy útil a la hora de aprender su funcionamiento, lo cual aligera mucho la gestión de licencias que suele causar bastantes problemas.

En este proyecto se han cumplido la gran mayoría de objetivos planteados inicialmente. Se ha conseguido generar una versión virtual de la planta robótica que permite llevar a cabo el proceso deseado de manera rápida y eficiente y, asimismo, el proyecto permite el estudio del tiempo de ciclo de todos los procesos que intervienen en el procesamiento de los paquetes.

La idea desarrollada en este proyecto puede contribuir de manera notable a la mejora en la eficiencia productiva de una empresa, pues puede ser usado como modelo para poder prevenir errores, ahorrar tiempo y recursos e implementar mejoras en cualquier proceso automático de carácter industrial.

2 OBJETO DEL PROYECTO

El objeto principal de este proyecto consiste en la generación de un gemelo digital de un proceso real de etiquetado, empaquetado y paletizado de un conjunto de botellas mediante el uso del software RoboDK. Posteriormente, se hará un análisis del tiempo de ciclo del proceso desarrollado para conseguir una mejora en el tiempo de producción. Durante el estudio del tiempo de ciclo se desarrollaron tres versiones distintas del gemelo digital, en esta memoria, se explica la distribución y programación correspondiente a la primera versión. No obstante, en el *Anexo II* y *Anexo III* se explican de forma detallada todos los cambios realizados en las versiones posteriores.

Para poder alcanzar este objetivo se ha hecho uso de las herramientas de modelado y diseño de RoboDk. De igual forma, para realizar la automatización se han aprovechado las herramientas de programación que ofrece el propio RoboDk y se ha hecho uso del lenguaje de programación Python para el funcionamiento de los distintos mecanismos del proyecto (cintas trasportadoras, reinicio de la simulación, generación y borrado de objetos…).

3 ANTECEDENTES, MOTIVACIÓN Y JUSTIFCACIÓN.

La robótica industrial se define como la disciplina que establece los parámetros fundamentales para realizar manipulaciones multifuncionales, permitiendo mover piezas, materiales, herramientas y cualquier tipo de dispositivo a través de trayectorias variables y programables.

El uso de la robótica industrial ha permitido desarrollar protocolos de programas de automatización general fundamentales para el avance y el desarrollo de los procesos productivos en la industria moderna. La automatización industrial ha permitido sustituir la intervención humana en procesos repetitivos o peligrosos por robots que, trabajando de forma coordinada, logran mejores resultados en términos de efectividad y productividad empresarial.

A día de hoy uno de los intereses más habituales de la gran mayoría de industrias consiste en la automatización de sus procesos productivos para hacer un uso más eficiente de sus recursos y mejorar sus posibilidades a la hora de competir en un mercado cada vez más diferenciado.

Además, la robótica industrial puede ocupar un papel fundamental en el cumplimiento de varios de los Objetivos de Desarrollo Sostenible (ODS) (Solène Guenat y otros, 2022) ya que podría reducir enormemente la pobreza, fomentar la salud y ayudar a afrontar la lucha contra el cambio climático. La mínima generación de residuos y, por tanto, mejor aprovechamiento de la materia prima, las condiciones ambientales flexibles (los robots pueden trabajar independientemente de la temperatura y el horario presentes en un proceso productivo lo que supone un ahorro energético) o la mejora continua de le eficiencia energética de los robots son solo algunos de los ejemplos de su notable aportación al cuidado del planeta.

Por tanto, algunos de los ODS con los que se puede relacionar este proyecto son los siguientes:

- ODS nº8: **Trabajo decente y crecimiento económico.** La automatización y digitalización de procesos industriales generan ambientes de trabajo seguros, en los que las tareas más peligrosas pueden ser llevadas a cabo por robots. Además, cabe destacar que el avance de la robótica moderna suele generar una mejora en la productividad y eficiencia de la industria lo que conlleva un crecimiento económico.

- ODS nº9: **Industria, innovación e infraestructura.** La digitalización de plantas robóticas es una forma innovadora de entender los procesos productivos de las empresas. De igual manera, fomenta el uso de herramientas técnicas fundamentales a la hora de mejorar la infraestructura tecnológica moderna.

- ODS nº12: **Producción y consumo responsable.** La optimización de los procesos productivos a través del uso de gemelos digitales y herramientas computacionales ayudan a disminuir el consumo energético y emisión de gases de efecto invernadero. Contribuyendo así, a un consumo más responsable y sostenible.

En el caso particular de este proyecto se ha realizado el gemelo digital de una planta de automatización de la empresa "Baxter Laboratories" con la motivación de mejorarla mediante el estudio de técnicas alternativas durante el desarrollo de las diferentes líneas de producción. Además, se tiene la intención de estudiar el tiempo de ciclo del proceso e intentar reducirlo al máximo mediante una simulación fuera de línea.

4 PLANTEAMIENTO DEL PROBLEMA INICIAL

Una vez seleccionado cual va a ser el proceso sobre el que se va a centrar el proyecto, a continuación, se intentará dar contexto al problema a resolver.

4.1 Descripción del proceso a controlar

Las próximas líneas tratarán de explicar brevemente las diferentes etapas que constituyen el proceso productivo.

- **Etiquetado de las botellas:** al principio del proceso un conjunto de botellas situadas en una cinta transportadora se mueve hasta alcanzar una posición determinada. Una vez alcanzada esta posición, se procede al etiquetamiento de cada una de las botellas y al reinicio del movimiento de la cinta. El proceso de detección de las botellas se podría haber llevado a cabo a partir de un sensor óptico, pero, puesto que RoboDk no permite la implementación de éstos, se ha definido en un programa de Python la distancia necesaria que ha de recorrer la cinta transportadora para que las botellas lleguen al punto deseado.

- **“Pick and Place” de las botellas:** una vez etiquetadas, las botellas llegan al final de la cinta donde son recogidas, en grupos de seis, en dos tandas diferentes, por un robot UR5. Posteriormente, el robot deposita el conjunto de doce botellas en el interior de una caja, previamente preparada por un robot UR16e, situada en otra cinta transportadora de rodillos que se encuentra a su derecha.

- **Doblado y preparado de cajas:** al mismo tiempo que el “pick and place” de las botellas, el robot UR16e trabaja en la preparación de las cajas que posteriormente serán paletizadas. El proceso consta de varias fases: primero el robot coge un cartón plano de una mesa situada a sus espaldas. Segundo, el robot, junto con la ayuda de un “case erector”, da forma al cartón plano hasta conseguir la caja deseada. En último lugar, el robot deja la caja en la cinta transportadora de rodillos y vuelve a su posición inicial.

- **Sellado del paquete:** una vez preparado el paquete, éste se cierra y se sella combinando el movimiento de la cinta con una máquina de sellado de cajas.

- **Paletizado de las cajas:** finalmente, cuando un paquete alcanza el final de la cinta transportadora de rodillos, un robot UR10 coloca los diferentes paquetes sobre un pallet situado a su izquierda. Cuando el pallet está listo, un operario lo retira y coloca uno nuevo para poder repetir el proceso.

El video del proceso en el que se ha inspirado este proyecto andrewdonalddesign. (2017, 7 de mayo). *Case Packing and Palletising at Baxter Laboratories* [video]. Youtube puede ser de mucha ayuda para comprender de manera visual el proceso anteriormente descrito. Se advierte al lector de que el operario ha sido sustituido por el robot UR16e para el doblado y preparado de las cajas y de qué el robot UR10 encargado del paletizado se ha situado en un pedestal en lugar de una grúa de eje vertical.

4.2 Descripción de la planta

Para entender la distribución de los elementos de la célula robótica se ha realizado una ilustración, a modo de esquema, para poder visualizar la disposición espacial de las diferentes estaciones y zonas que conforman el proyecto. La descripción detallada de las diferentes líneas y elementos del proyecto se hará más adelante.

Ilustración 1. Distribución de la planta.

Asimismo, se ha decidido elaborar un diagrama de flujo con la finalidad de esclarecer el flujo de materiales e intervención de la maquinaria comentada.

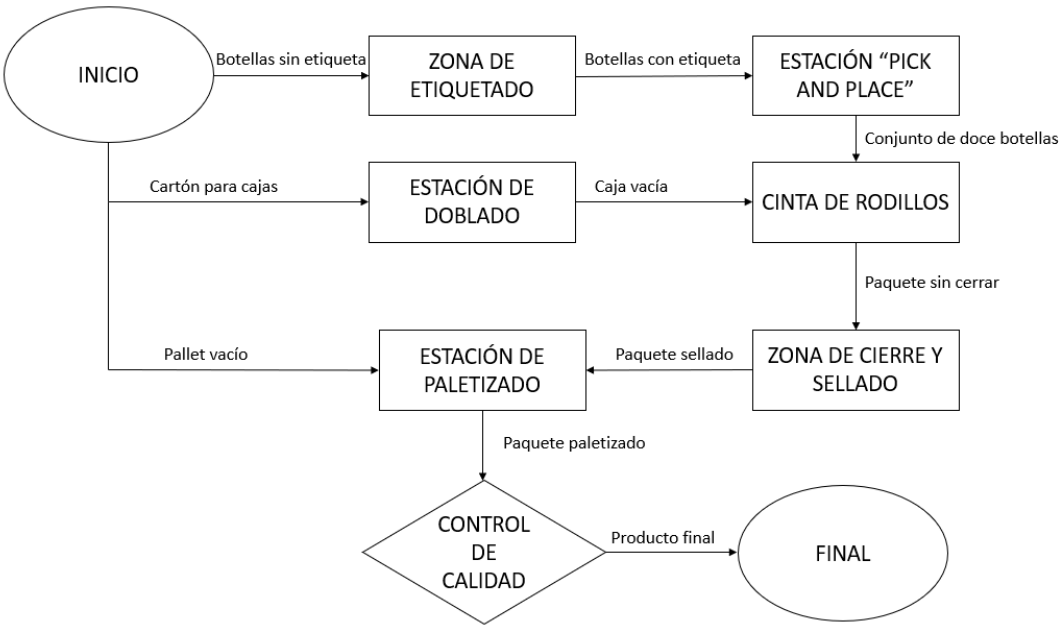


Ilustración 2. Funcionamiento de la planta.

5 SOFTWARE

RoboDk es un software empleado en el mundo de la robótica y automatización que permite la programación fuera de línea de un robot o la generación de gemelos digitales en los que varios robots trabajan simultáneamente (ejecutando múltiples programas al mismo tiempo). Este software permite crear programas libres de errores de forma sencilla, evitando, además, las singularidades de los robots y posibles colisiones no deseadas. Por otra parte, RoboDk permite la conversión de modelos CAD en código para robots y admite el uso de más de trescientos tipos de robots corporativos de una gran cantidad de marcas.

En este apartado se tiene la intención de explicar, de forma sencilla y resumida, algunas de las herramientas de diseño, programación y simulación que ofrece RoboDk y que han sido fundamentales a la hora de realizar el proyecto. La finalidad del mismo reside en orientar al lector sobre algunas de las funcionalidades con las que cuenta RoboDk para hacer más fácil la comprensión del gemelo digital realizado.

Una vez se abre RoboDk se genera una estación por defecto sobre la cual ya se puede trabajar.

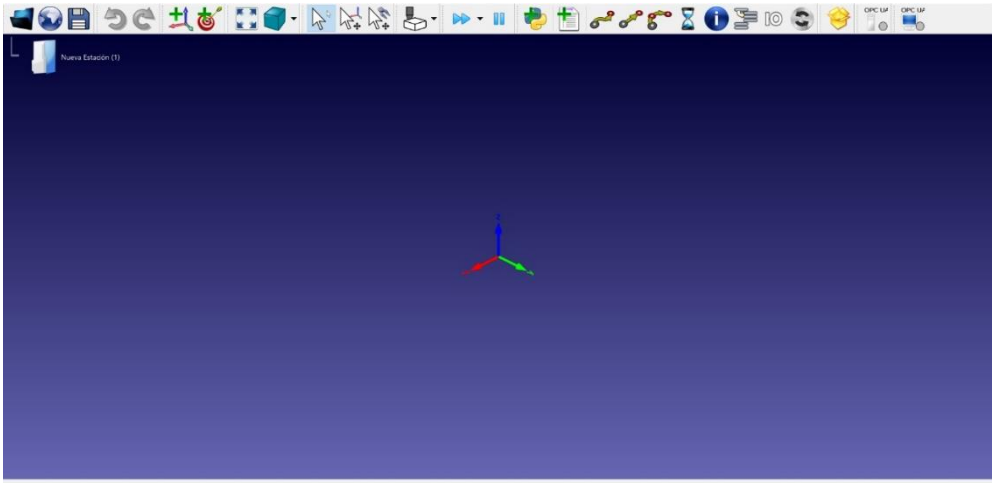


Ilustración 3. Pantalla de inicio RoboDk.

En primer lugar, se va a proceder a explicar algunas de las herramientas de diseño con las que cuenta RoboDk (escalar, cambiar color de un objeto y conocer rango de alcance del robot) que pueden ser de mucha utilidad a la hora de mejorar la distribución de la planta de un proyecto.

Para poder editar un objeto se ha de comenzar pulsando “click” derecho sobre el objeto, en ese momento, aparecerá una ventana similar a la que se puede observar en la *ilustración 4.* A continuación se ha de pulsar Opciones.

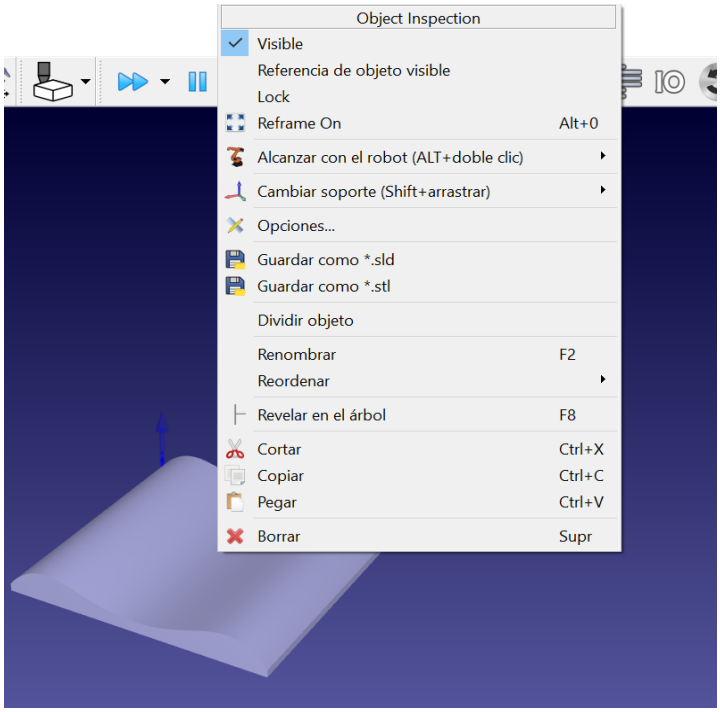


Ilustración 4. Ventana de un objeto en RoboDk.

Para poder editar el tamaño del objeto deseado se ha de pulsar More options>Aplicar Escala. Una vez se define la escala deseada se pulsa OK.

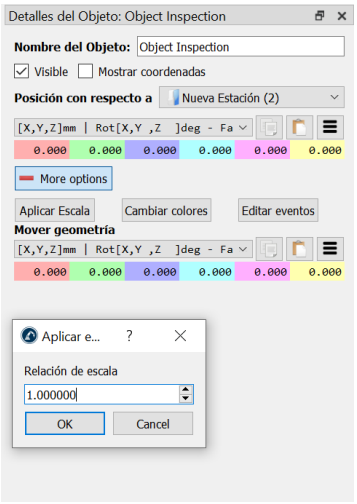


Ilustración 5. Cambiar escala de un objeto en RoboDk.

Para cambiar el color de un objeto se ha seguir la siguiente secuencia: More Options>Cambiar colores>Seleccionar objetos. Posteriormente se pulsa alguna de las capas que se quiere cambiar de color y se selecciona el color apropiado. Finalmente se ha de pulsar OK.

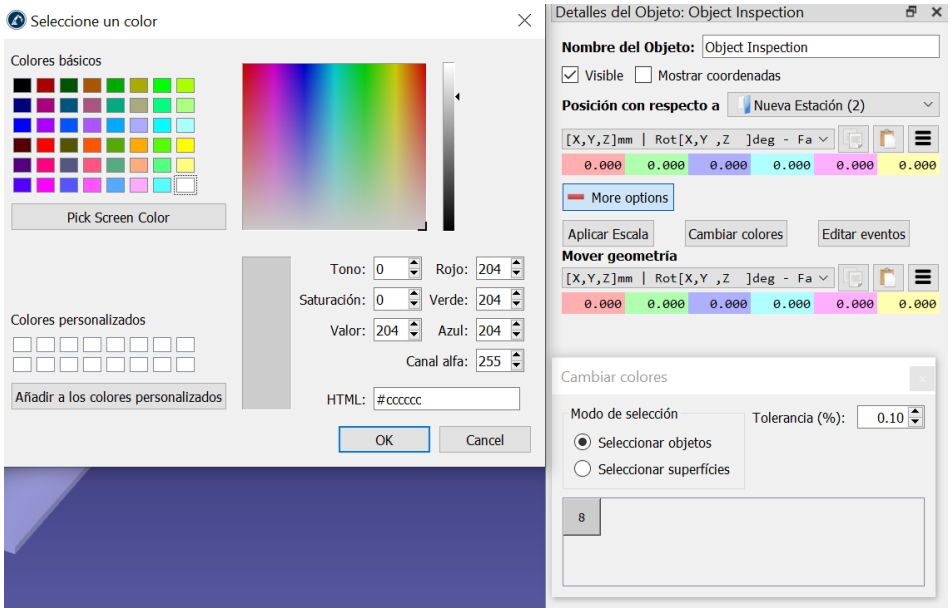


Ilustración 6. Cambiar color de un objeto en RoboDk.

Seleccionando la opción Pick Screen Color se puede cambiar el color a cualquier color presente en la ventana del proyecto lo que resulta muy útil si se quiere copiar el color de algún objeto del proyecto.

Para conocer el rango de alcance de un robot basta con pulsar la tecla "*". Esta herramienta resulta muy útil a la hora de decidir dónde situar las cosas en una célula robótica.

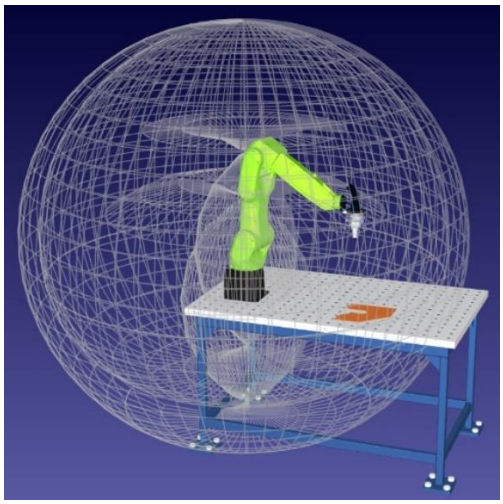


Ilustración 7. Conocer el rango de alcance de un robot en RoboDk.

A continuación, se procederá a explicar alguna de las funciones de la barra de tareas muy importantes a la hora de programar y simular en RoboDk.

Ilustración 8. Algunas de las funciones de RoboDk.

En la *ilustración 8.* se puede observar las siguientes funciones:

1. **Cargar archivo:** permite abrir cualquier archivo descargado en la unidad local compatible con RoboDK.

2. **Librería "online" de RoboDk:** muy importante a la hora de decidir qué elementos formarán parte del proyecto (robots, herramientas, objetos, mecanismos…).

3. **Opción de guardado de la estación:** se debe pulsar siempre que se desee guardar el progreso realizado.

4. **Rehacer/Deshacer:** opción para rehacer o deshacer alguna de las modificaciones realizadas, no sirve a la hora de editar programas.

5. **Añadir un sistema de referencia:** opción que permite añadir un sistema de referencia cartesiano, clave para poder situar, de manera sencilla, los diferentes elementos que conforman un proyecto (robots, objetos, objetivos para los robots…) y definir distancias entre ellos.

6. **Añadir nuevo objetivo para el robot:** esta función permite crear un objetivo a alcanzar por un robot cuando se considere necesario. Es imprescindible que exista previamente un robot para poder ejecutar esta acción.

7. **Opciones de vista:** estas dos opciones permiten ajustar la vista del proyecto desde diferentes perspectivas.

8. **No mover sistema de referencia, objeto o herramientas:** forma en la que el usuario ha de "desplazarse" por defecto en la estación, sin mover ningún elemento.

9. **Mover sistema de referencia:** permite mover de forma rápida y visual los sistemas de referencia creados. También se puede activar esta función manteniendo la tecla ALT.

10. **Mover herramienta:** se usa para mover la herramienta del robot (TCP) respecto a la brida del robot. Otra forma de activar esta opción es manteniendo la tecla ALT+Shift.

11. **Detección de colisión:** Una vez se ha terminado un proyecto es importante comprobar que no existe contacto entre dos elementos que no deben colisionar. Activando esta herramienta RoboDK avisará al usuario cada vez que se dé esta situación.

12. **Regular velocidad de simulación:** permite subir o bajar la velocidad de simulación del proyecto creado. La velocidad por defecto es cinco veces más rápida que en la realidad.

13. **Detener programa:** permite detener los programas que se están ejecutando.

Ilustración 9. Otras funciones de RoboDk.

En la *ilustración 9.* se pueden observar las siguientes opciones:

14. **Añadir programa Python:** permite crear un programa en lenguaje Python. Este lenguaje es clave a la hora de programar con RoboDk, más adelante se explicará la importancia del mismo a lo hora de realizar este proyecto.

15. **Crear programa:** esta función permite crear un programa ligado a un robot en particular. Es el elemento de programación clave en RoboDk ya que sin esta herramienta no se puede ejecutar ninguna acción programada por el usuario.

16. **Añadir movimiento no lineal al programa:** con esta herramienta se consigue que un robot alcance, mediante un movimiento no lineal, un objetivo determinado. Es necesario definir el robot y el punto que se desea alcanzar.

17. **Añadir movimiento lineal al programa:** da la orden a un robot de alcanzar, si es posible, un objetivo mediante un movimiento lineal.

18. **Añadir movimiento circular al programa:** permite que el robot alcance un punto definido en el espacio mediante una trayectoria circular. Es necesario definir el punto medio y final del arco para poder hacer efectiva la orden.

19. **Añadir instrucción de pausa al programa:** introduce una pausa del tiempo que considere el usuario, en un momento determinado, durante la ejecución de un programa.

20. **Mostrar mensaje:** se usa para mostrar el mensaje que considere oportuno el usuario en la “teach pendant” de un robot.

21. **Ejecutar subprograma o insertar código de instrucción:** principalmente se usa para ejecutar un programa dentro de otro programa (más general). También se puede usar para añadir códigos de instrucción como bucles “while” o “for”.

22. **Establecer o esperar una salida digital específica:** Herramienta fundamental para la coordinación y la correcta ejecución de los diferentes programas que conforman un proyecto. Permite al usuario crear y asignar un valor a una variable en particular además de esperar a que una salida digital específica adquiera un valor determinado.

23. **Simulación de evento:** las principales funcionalidades de esta herramienta son: hacer que una herramienta de un robot (p. ej. una pinza) coja o suelte un objeto, hacer aparecer o desparecer un elemento particular y reiniciar la posición de una serie de objetos en particular, muy útil a la hora de ensayar una simulación.

Con lo explicado en este apartado ya se puede entender, de forma superficial, pero suficiente, el funcionamiento de un proyecto en RoboDK.

Por último, se considera interesante comentar que, una vez se ha terminado con la programación de un robot, se puede generar el código del programa del robot haciendo “click” derecho sobre el programa y seleccionando la opción de “Generar programa del robot”. El lenguaje de programación empleado en el código generado dependerá de la corporación a la que pertenezca el robot con el que se ha trabajado.

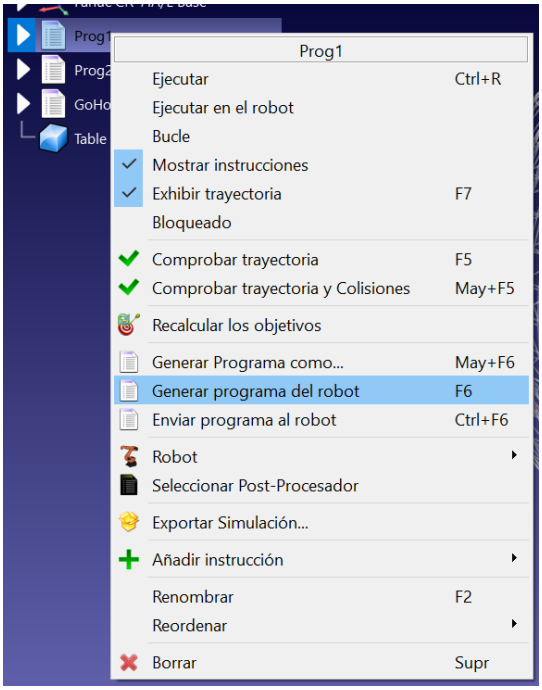
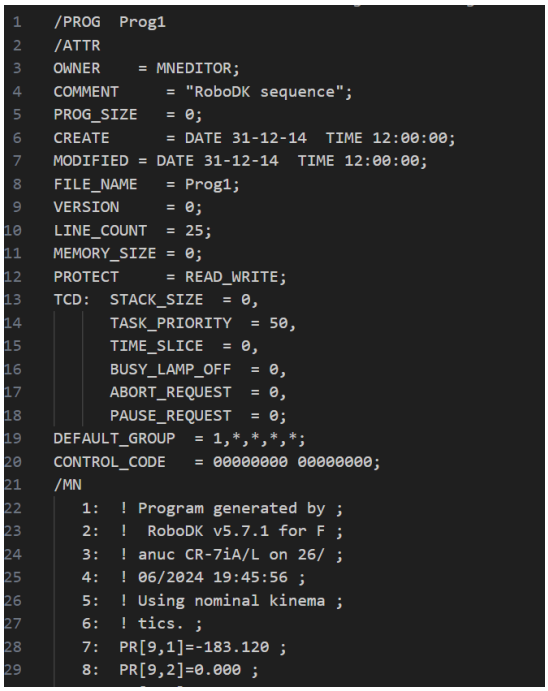


Ilustración 10. Generar programa del robot.



```
/PROG  Prog1
/ATTR
OWNER    = MNEDITOR;
COMMENT     = "RoboDK sequence";
PROG_SIZE   = 0;
CREATE      = DATE 31-12-14  TIME 12:00:00;
MODIFIED = DATE 31-12-14  TIME 12:00:00;
FILE_NAME   = Prog1;
VERSION     = 0;
LINE_COUNT  = 25;
MEMORY_SIZE = 0;
PROTECT     = READ_WRITE;
TCD:  STACK_SIZE  = 0,
      TASK_PRIORITY  = 50,
      TIME_SLICE  = 0,
      BUSY_LAMP_OFF  = 0,
      ABORT_REQUEST  = 0,
      PAUSE_REQUEST  = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE   = 00000000 00000000;
/MN
   1:  ! Program generated by ;
   2:  !  RoboDK v5.7.1 for F ;
   3:  ! anuc CR-7iA/L on 26/ ;
   4:  ! 06/2024 19:45:56 ;
   5:  ! Using nominal kinema ;
   6:  ! tics. ;
   7:  PR[9,1]=-183.120 ;
   8:  PR[9,2]=0.000 ;
```

Ilustración 11. Código generado.

A la hora de adquirir los conocimientos necesarios para la realización de este apartado y de todo el proyecto en general, se usó como fuente de información principal, una serie de noventa y cuatro vídeos organizados en forma de tutorial que se pueden encontrar en el canal oficial de RoboDK de YouTube RoboDk. (2021, 16 de noviembre). *Intro and Installation - Module 1.01 - RoboDK Pro Training* [video]. YouTube.

6 LENGUAJE DE PROGRAMACIÓN

Para la realización del proyecto se ha hecho uso del lenguaje de programación Python. La versión predeterminada de RoboDk ya cuenta con la disponibilidad de una versión de este lenguaje que permite trabajar de forma bastante cómoda y eficiente. Además, requiere del uso de menos líneas de código en comparación con otros lenguajes, haciéndolo relativamente intuitivo y fácil de aprender.

En el caso particular de este proyecto se ha usado para controlar los siguientes procesos:

- Movimiento de las cintas transportadoras.
- Reinicio cintas y pallets.
- Generación y borrado de objetos.

Estos puntos serán debidamente explicados y desarrollados más adelante en el punto 8.2 de la memoria.

7 DESCRIPCIÓN DE LA SOLUCIÓN ADOPTADA

En este apartado se explica detalladamente los diferentes elementos que conforman el proyecto. De ahora en adelante se distinguirán las siguientes partes para hacer más fácil la descripción del diseño de la planta:

- Línea de etiquetado y “Pick and Place” de botellas.
- Línea de doblado y preparado de cajas.
- Línea de sellado y paletizado de cajas.

Antes de entrar a describir cada uno de los puntos anteriormente comentados es necesario observar la *ilustración 12.* para poder entender, desde un punto de vista global, el proyecto desarrollado.

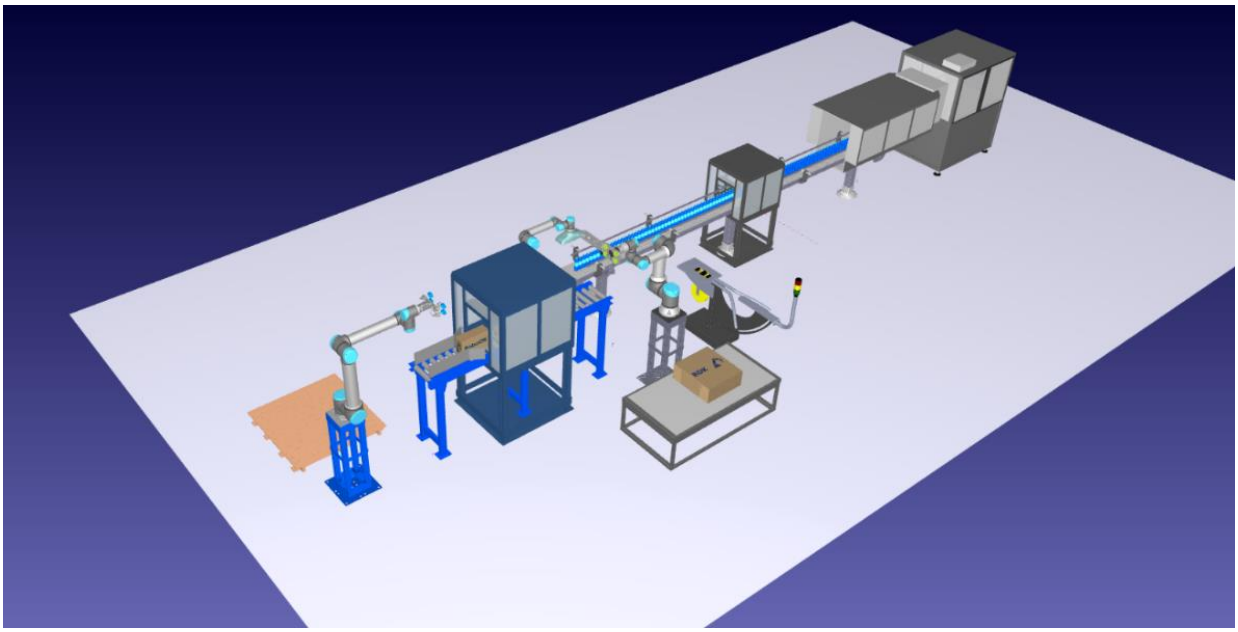


Ilustración 12. Vista general de la estación en RoboDk.

7.1 Línea de etiquetado y “Pick and Place”

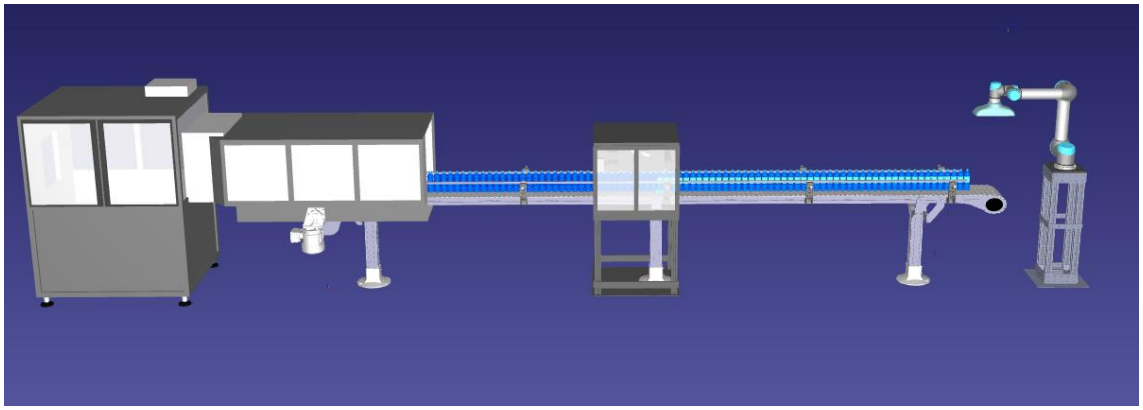


Ilustración 13. Línea de etiquetado y “Pick and Place” en RoboDk.

En este apartado se explica el funcionamiento de la línea dedicada al etiquetado y al “Pick and Place” de las botellas, para ello se detallarán los distintos elementos que la constituyen (objetos, mecanismos y robots) explicando, además, las funciones de cada uno y la secuencia de funcionamiento general de la línea.

Máquina de botellas:

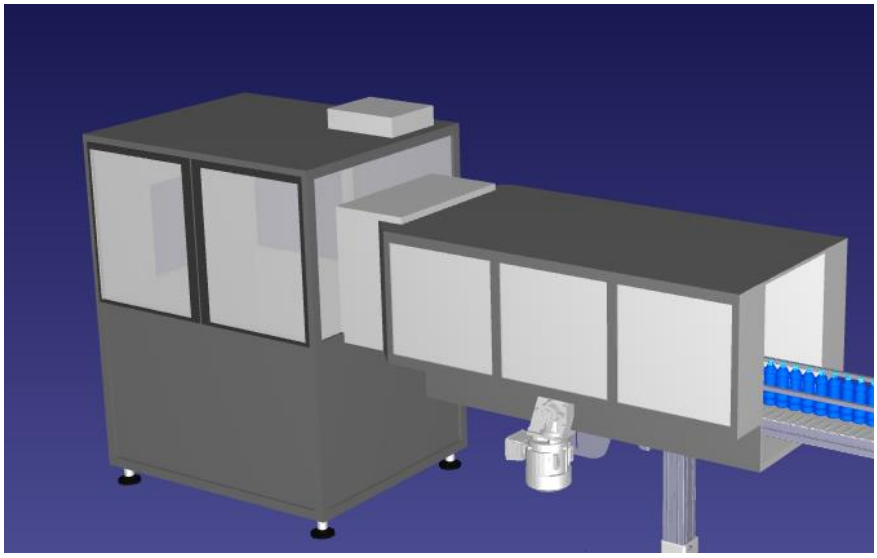


Ilustración 14. Máquina de botellas en RoboDk.

Su función principal es producir botellas para asegurar que se puedan procesar tantos paquetes como sean posibles. Aunque, realmente, su función es meramente decorativa ya que, como se explicará más adelante, la producción de botellas está controlada por un fichero escrito en lenguaje Python.

Cinta transportadora de 4.5 m:

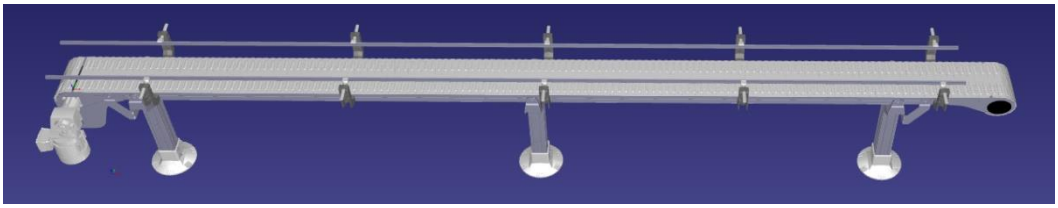


Ilustración 15. Cinta transportadora en RoboDk.

Su función principal es transportar las botellas antes y después de ser etiquetadas. En RoboDk se define como un mecanismo de un eje lineal, este tipo de mecanismo se usan en otras muchas aplicaciones como la generación de puertas, pedestales retráctiles o railes.

Botellas sin etiquetar (lote de seis):

Ilustración 16. Lote de sesis botellas en RoboDk.

Consiste en un grupo de seis botellas diseñadas como un solo objeto con el fin de conseguir un mayor parecido con el proceso llevado a cabo en "Baxter Laboratories". En el proyecto también encontramos este mismo objeto, pero con las botellas etiquetadas. No desempeñan ninguna función en particular.

Máquina etiquetadora de botellas

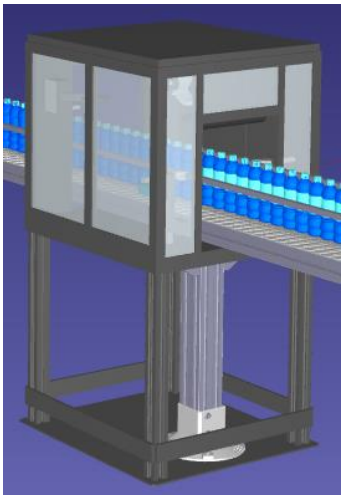


Ilustración 17. Máquina etiquetadora de botellas en RoboDk.

Al igual que la máquina de botellas, su función es decorativa ya que el proceso de etiquetado de las botellas se controla desde un programa escrito en lenguaje Python.

Robot UR5 y herramienta:

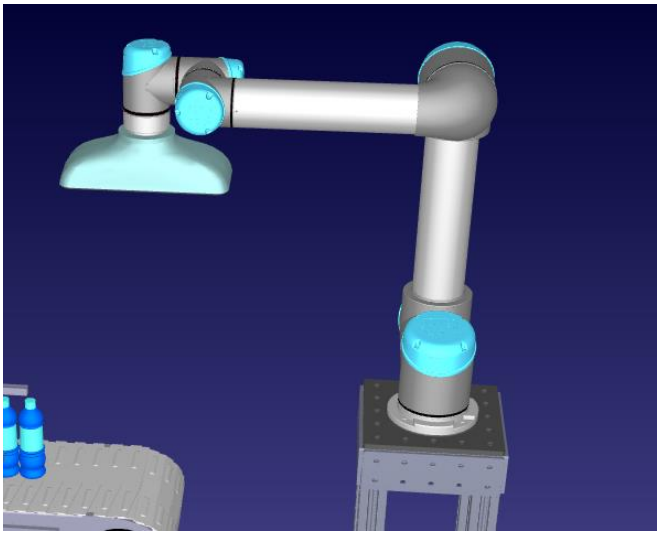


Ilustración 18. Robot UR 5 en RoboDk.

Robot encargado de recoger las botellas de la cinta transportadora y depositarlas en el interior de la caja previamente preparada por la estación de doblado. Cuenta con una herramienta acoplada a su brida de montaje. Sus características son las siguientes:

Brand	Universal Robots
Model	UR5
Axes	6
Reach	850 mm
Payload	5 kg
Weight	18 kg
Repeatability	0.100 mm

Ilustración 19. Características del robot UR5.

Otro objeto que aparece en el proyecto es el pedestal sobre el que está situado el robot UR5. Necesario para dotar de realismo a la simulación.

Variables o salidas digitales que intervienen en la línea:

NOMBRE DE LA SALIDA DIGITAL	FUNCIÓN
PACK_LISTO	Asegura que se han depositado correctamente las botellas en la caja.
DOBLADO	Asegura que la caja dónde se depositarán las botellas está debidamente colocada en la cinta de rodillos.

Tabla1. Salidas digitales línea de etiquetado y "Pick and Place".

Funcionamiento secuencial de la línea:

La línea de etiquetado y "Pick and Place" de las botellas presenta el siguiente funcionamiento secuencial:

En un primer momento, una vez se han situado las botellas en la posición inicial, es decir, todas las botellas (con y sin etiqueta) se han colocado en la cinta de manera que el proceso puede empezar, el robot coge un lote de seis botellas.

Posteriormente, la cinta transportadora avanza hasta que un nuevo lote de botellas alcanza el final de la cita. Momento en el que el robot coge otro lote sumando un total de doce botellas recogidas.

A continuación, el robot se dirige hacia la cinta transportadora de rodillos y espera a que la caja preparada por la estación de doblado esté debidamente colocada ("DOBLADO" vale uno). Mientras tanto, la cinta se ha vuelto a mover para asegurar que cuando el robot regrese a su posición inicial otro lote de botellas esté listo para ser recogido y, paralelamente, el robot deja el conjunto de doce botellas en el interior de la caja.

Acto seguido, una vez ha dejado el conjunto de botellas, el robot regresa a su posición inicial, advirtiendo al programa de que el paquete está listo para ser cerrado ("PACK_LISTO" pasa a valer uno).

7.2 Línea de doblado y preparado de cajas

En este apartado se va a explicar el funcionamiento de la línea dedicada al doblado y preparado de las cajas dónde se depositarán las botellas, con esta finalidad se detallarán los distintos elementos que la constituyen (objetos, mecanismos y robots) explicando, además, las funciones de cada uno y la secuencia de funcionamiento general de la línea.

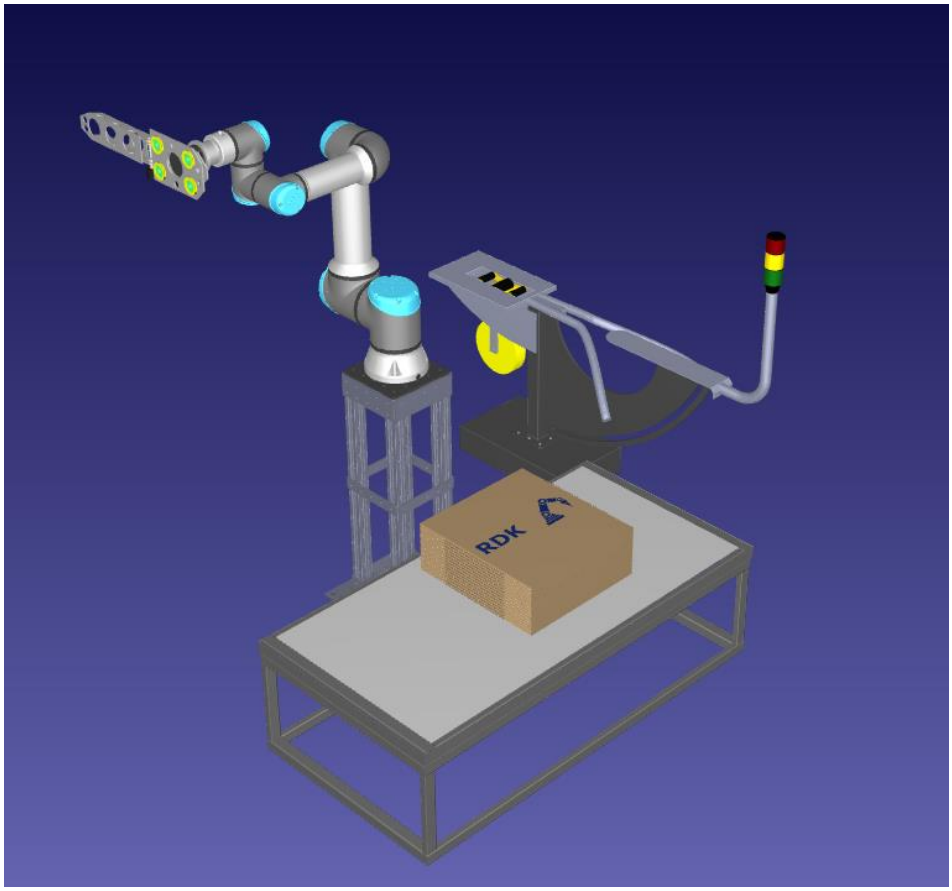


Ilustración 20. Vista general de la estación de doblado en RoboDk.

Conjunto de cajas planas y mesa:

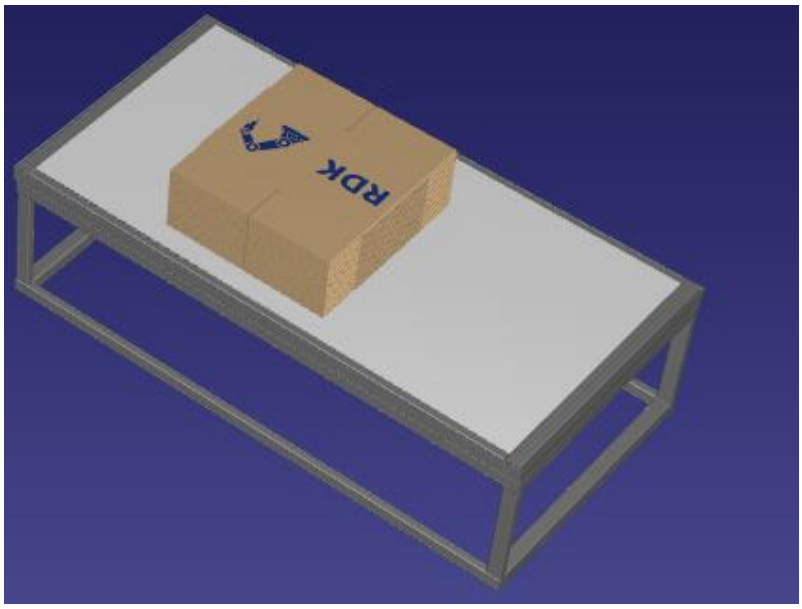


Ilustración 21. Cajas planas y mesa en RoboDk.

Objetos con la única intención de dotar de realismo y credibilidad a la simulación.

Case erector:

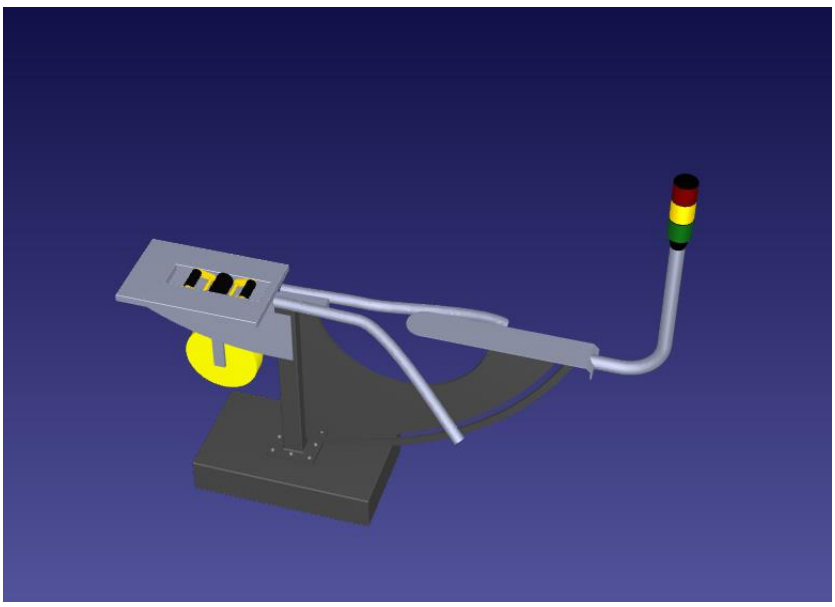


Ilustración 22. Case erector en RoboDk.

Como se comentó en el punto 4.1 de este proyecto, el “case erector” consiste en una herramienta que permite al robot UR16e dar forma a la caja que, posteriormente, se utilizará en el proceso de paletizado. El proceso de dar forma a la caja se explicará más adelante en el punto 8.1 de esta memoria.

Robot UR16e y herramienta:

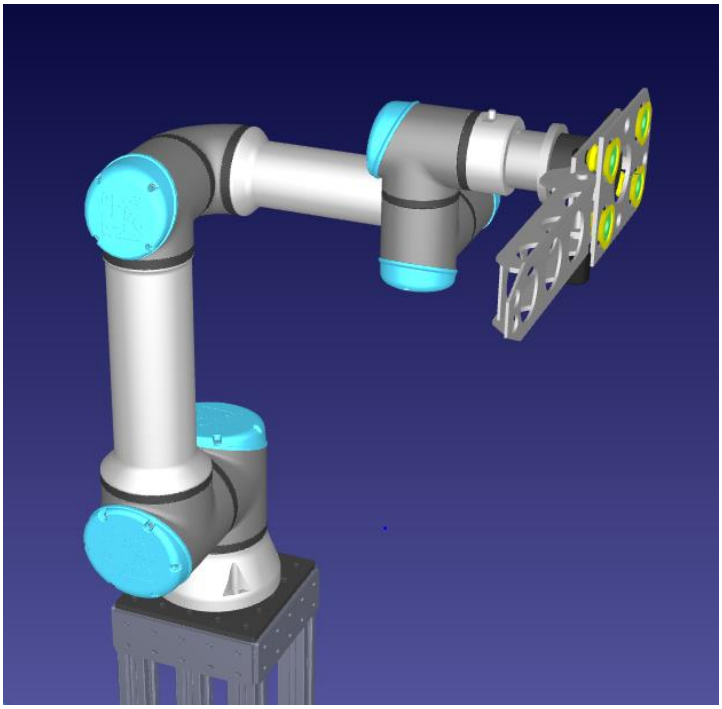


Ilustración 23. Robot UR16e y herramienta en RoboDk.

Robot encargado de coger una caja plana y, mediante el uso de su herramienta, darle forma hasta conseguir la caja deseada. Una vez conseguida la forma final el robot deposita la caja en la cinta de rodillos.

Las características del robot son las siguientes:

Brand	Universal Robots
Model	UR16e
Axes	6
Reach	900 mm
Payload	16 kg
Weight	33 kg
Repeatability	0.050 mm

Ilustración 24. Características del UR16e.

Al igual que en la línea de etiquetado el robot UR16e está situado en un pedestal, objeto que también se ha de tener en cuenta.

Variables o salidas digitales que intervienen en la línea:

NOMBRE DE LA SALIDA DIGITAL	FUNCIÓN
DOBLADO	Asegura que la caja dónde se depositarán las botellas esta debidamente colocada en la cinta de rodillos
RODILLOS_DONE	Asegura que la cinta transportadora de rodillos se ha detenido tras recorrer una distancia determinada.

Tabla 2. Salidas digitales línea de doblado y preparado de cajas.

Funcionamiento secuencial de la línea:

La línea dedicada a la preparación de las cajas funciona de mediante el siguiente comportamiento secuencial:

En primer lugar, el robot UR16e se dirige hacia la zona de las cajas planas situada a su izquierda y coge un cartón de la mesa. Posteriormente, haciendo uso de su herramienta y del "case erector" comienza a darle forma a la caja hasta conseguir el objeto deseado. En este momento, el robot espera a que la cinta transportadora de rodillos termine su recorrido ("RODILLOS_DONE" valga uno).

Una vez la cinta ha avanzado todo lo que debía el robot deposita la caja en la cinta transportadora y la variable "DOBLADO" pasa a valer uno. Permitiendo así, que el robot UR5 introduzca el lote de doce botellas en el interior de la caja.

Finalmente, el robot UR16e vuelve a su posición inicial para seguir con la producción de cajas mediante la secuencia anteriormente descrita.

7.3 Línea de sellado y paletizado de paquetes

En este apartado, de forma similar a los dos anteriores, se procederá a explicar el funcionamiento de la línea de sellado y paletizado de cajas. Haciendo especial hincapié en los diferentes elementos que forman parte de la misma (objetos, mecanismos y robots), las funciones que desempeñan cada uno de estos elementos y el funcionamiento secuencial de la línea.

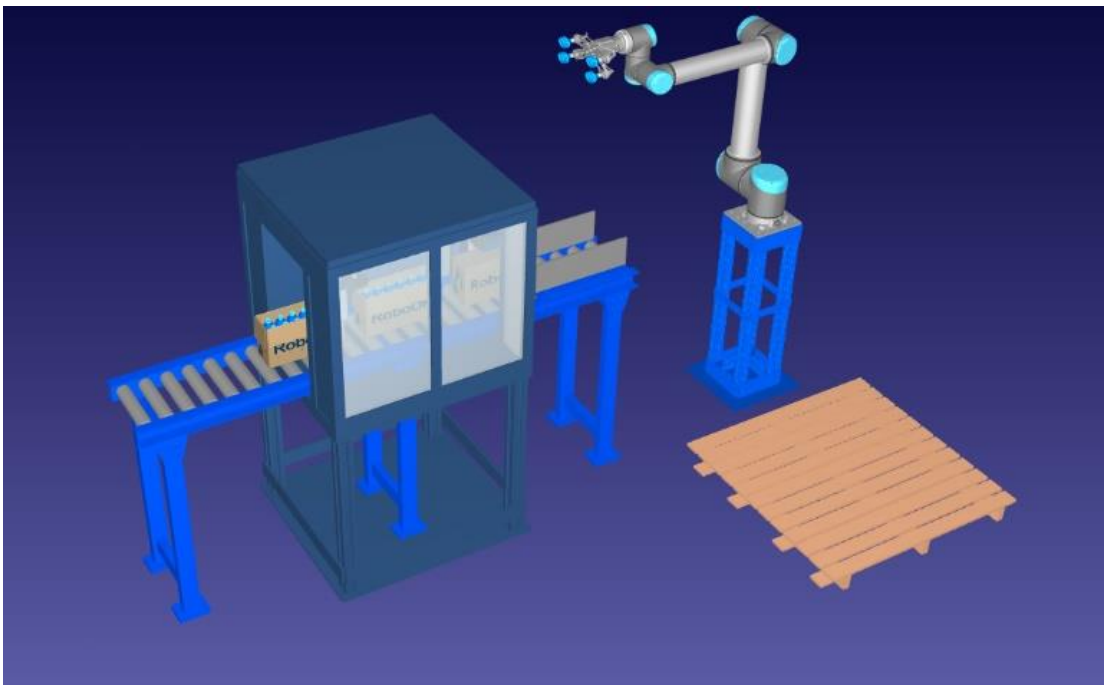


Ilustración 25. Vista general de la estación de sellado y paletizado en RoboDk.

Máquina de sellado:

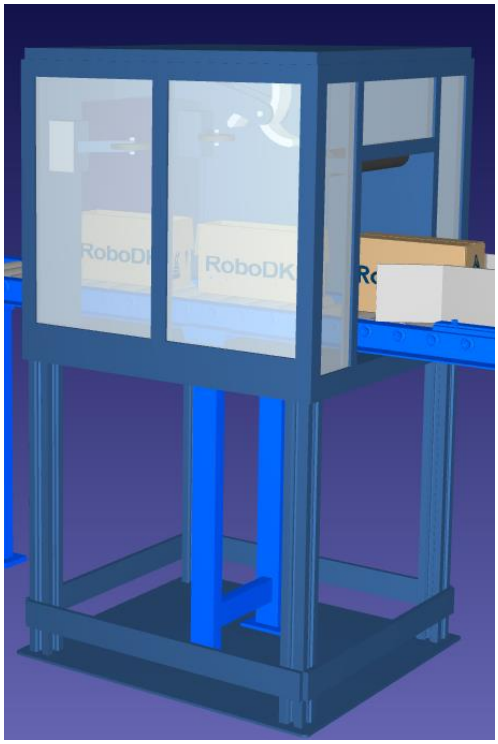


Ilustración 26. Máquina de sellado en RoboDk.

Al igual que otros objetos de este proyecto su función es decorativa ya que lo que realmente "cierra y sella" los paquetes es un fichero escrito en lenguaje Python. Los detalles sobre la programación realizada para llevar a cabo este proceso se explicarán en el punto 8.2 de esta memoria.

Paquete abierto/cerrado:

Ilustración 27. Paquete abierto/cerrado en RoboDk.

Se trata de un objeto con dos versiones diferentes. La versión abierta y la versión cerrada. La primera está formada por dos conjuntos de botellas y una caja abierta (debidamente colocada y escalada) mientras que la segunda es simplemente una caja cerrada. Son los objetos a manipular durante el proceso de sellado y paletizado.

Cinta transportadora de rodillos de 2 m:

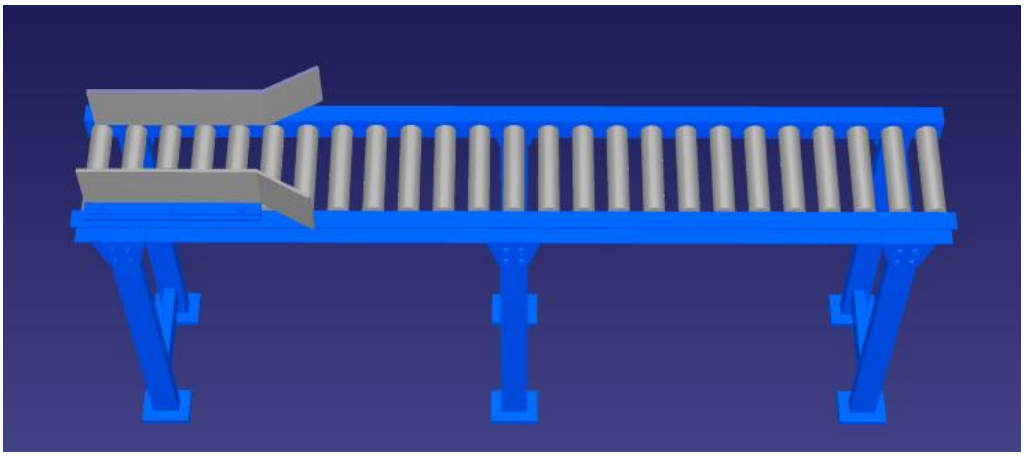


Ilustración 28. Cinta de rodillos 2 m en RoboDk.

Su función principal es trasladar los paquetes de un punto a otro. Al igual que la cinta de la línea de etiquetado y “Pick and place” es un mecanismo de un eje lineal.

Pallet de madera 900 x 900 mm:

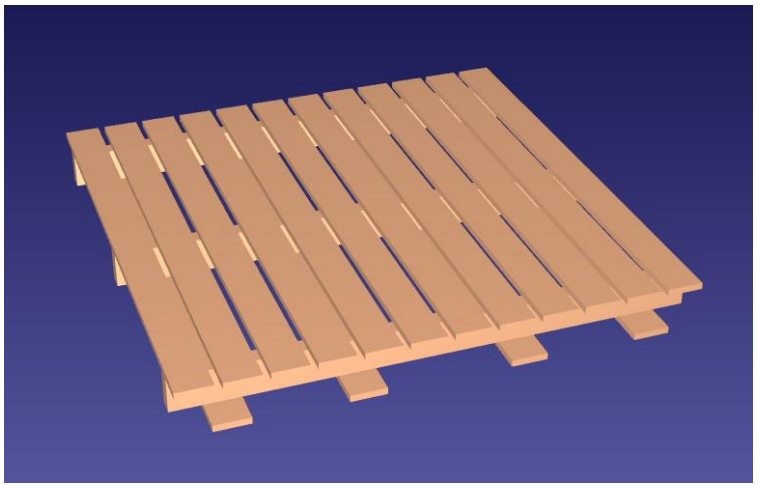


Ilustración 29. Pallet de madera en RoboDk.

Armazón rígido de madera sobre el que se coloca los paquetes una vez están debidamente sellados.

Robot UR10 y herramienta:

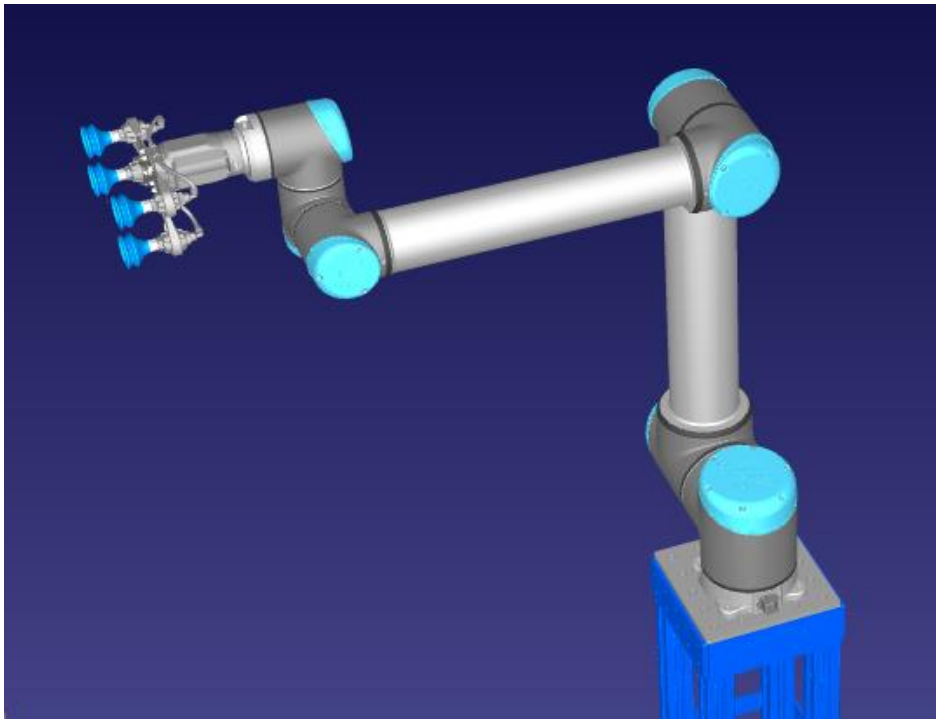


Ilustración 30. Robot UR10 y herramienta en RoboDk.

Robot situado encima de un pedestal encargado de coger los paquetes de la cinta transportadora de rodillos y depositarlos en el pallet situado a su derecha. Cuenta con una herramienta dotada de ventosas para asegurar un agarre firme a la hora de mover los paquetes. Sus características son las siguientes:

Brand	Universal Robots
Model	UR10
Axes	6
Reach	1300 mm
Payload	10 kg
Weight	29 kg
Repeatability	0.100 mm

Ilustración 31. Características del robot UR10.

Variables o salidas digitales que intervienen en la línea:

NOMBRE DE LA SALIDA DIGITAL	FUNCIÓN
PALETIZADO	Asegura que un paquete se encuentra al final de la cinta transportadora de rodillos listo para ser recogido por el robot UR10.
RODILLOS_DONE	Asegura que la cinta transportadora de rodillos se ha detenido tras recorrer una distancia determinada.
PACK_LISTO	Asegura que se han depositado correctamente las botellas en la caja.
DOBLADO	Asegura que la caja dónde se depositarán las botellas está debidamente colocada en la cinta de rodillos.

Tabla 3. Salidas digitales línea de sellado y paletizado.

Funcionamiento secuencial de la línea:

La línea de sellado y paletizado de cajas funciona a partir de la siguiente secuencia de procesos:

Primero de todo, una vez se ha comprobado que tanto la caja vacía (“DOBLADO”) como el lote de doce botellas (“PACK_LISTO”) están listos para ser procesados (sus respectivas variables valen uno), la cinta de rodillos avanza hasta que un paquete alcanza el final de la cinta (“PALETIZADO” vale uno).

Una vez se ha detenido la cinta (“RODILLOS_DONE” vale uno), el robot UR10 coge un paquete y lo deposita en el pallet de manera que no interfiera ni choque con otros paquetes. En ese momento, el robot regresa a su posición inicial y espera a que los robots UR5 y UR16e terminen sus respectivas tareas. Cuando esto ocurre la cinta se vuelve a mover, procediendo así, al sellado de un paquete y al traslado de un nuevo paquete al final de la cinta. Este proceso se repetirá en bucle hasta que el usuario así lo desee.

8 PROGRAMACIÓN DEL PROCESO AUTOMÁTICO

8.1 Programación en RoboDk

Como ya se explicó anteriormente, el software RoboDk cuenta con una gran cantidad de herramientas de programación las cuales han sido fundamentales a la hora de desarrollar este proyecto. En este apartado se explicarán, paso a paso, los diferentes elementos de modelado y programas que han hecho posible la generación del gemelo digital. Para ello, se describirán de forma independiente las tres líneas comentadas en el apartado 7 de esta memoria. Los desarrollos de los programas explicados en este apartado se pueden encontrar en el *Anexo I* de esta memoria.

Línea de etiquetado y "Pick and Place" (elementos de programación empleados)

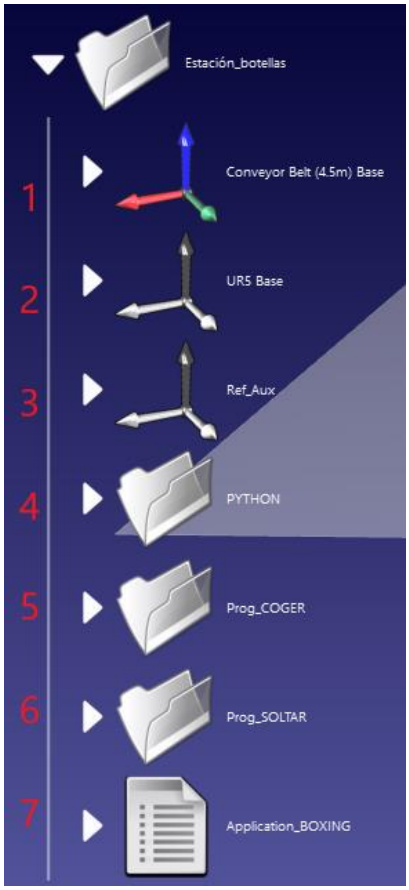


Ilustración 32. Programación línea etiquetado y "Pick and Place".

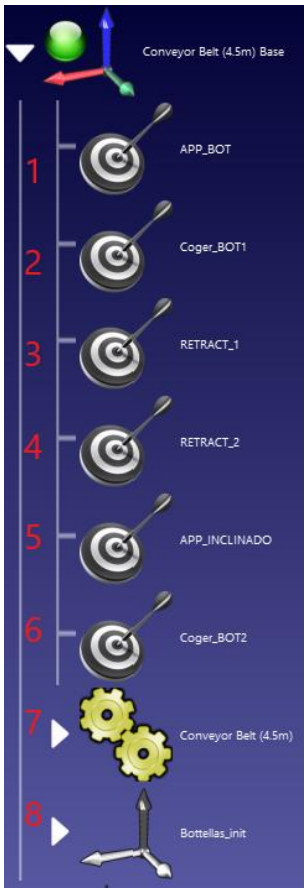


Ilustración 33. Interior del sistema de referencia de la cinta transportadora.

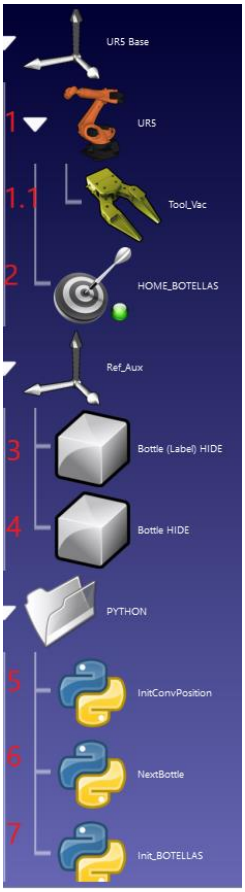


Ilustración 34. Más elementos de programación de la línea.

Con la finalidad de establecer un orden en la explicación se comenzará comentando la *ilustración 32.* Los puntos que aparecen en la imagen son los siguientes:

1. **“Conveyor Belt (4.5m) Base”:** sistema de referencia al que van ligado una gran cantidad de objetos muy importantes en la simulación.

2. **“UR5 Base”:** sistema de referencia del robot UR5.

3. **“Ref_Aux”:** sistema de referencia que contiene dos objetos clave en la utilización del programa “NextBottle.py” (explicado en el punto 8.2 de la memoria).

4. **Carpeta Python:** en su interior se encuentran todos los programas escritos en Python necesarios para el correcto funcionamiento de la línea.

5. **Carpeta con programas de coger elementos:** contiene todos los programas empleados para coger y desplazar las botellas que se encuentran en la cinta.

6. **Carpeta con programas de dejar elementos:** contiene todos los programas necesarios para que el robot deposite correctamente las botellas.

7. **“Application_BOXING”:** Programa mediante el cual se puede empaquetar un lote de doce botellas exitosamente.

En la *ilustración 33.* encontramos los siguientes elementos:

1. **al 6. Conjunto de puntos (coger objetos):** puntos empleados en la simulación del “pick” de las botellas.

7. **“Conveyor Belt (4.5m)”:** mecanismo de la cinta transportadora. En su interior se encuentra un sistema de referencia (que se mueve solidario con la cinta) en el que se pueden observar, anidados, los diferentes lotes de botellas situados encima de la cinta.

8. **“Botellas_init”:** sistema de referencia cuyo interior contiene una copia exacta de la disposición las botellas antes de comenzar la simulación. Se emplea para que, mediante Python, se pueda reiniciar la simulación de forma rápida y eficiente.

En la *ilustración 34* se encuentran los siguientes elementos:

1. **Robot UR5.**

    1.1. **“Tool_Vac”:** herramienta empleada durante el proceso de “Pick and Place” de las botellas.

2. **Punto de regreso a casa del robot UR5.**

3. **Lote de botellas etiquetadas escondido:** objeto que se utiliza para la generación de un lote botellas etiquetadas mediante un programa escrito en Python.

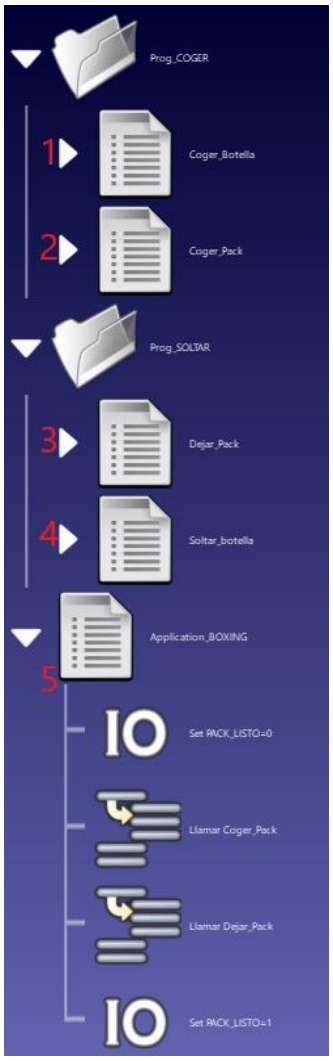
4. **Lote de botellas sin etiquetar escondido:** objeto que se utiliza para la generación de botellas sin etiquetar mediante un programa escrito en Python.

5. **“InitConvPosition.py”:** programa que reinicia la posición de la cinta transportadora.

6. **“NextBottle.py”:** programa que posibilita la generación de un nuevo lote de botellas, el proceso de etiquetado y el avance de la cinta.

7. **“init_BOTELLAS.py”:** utiliza todas las botellas anidadas en el sistema de referencia de inicialización para el reinicio de la simulación.

Línea de etiquetado y "Pick and Place" (programas empleados)



1. **"Coger_Botella":** programa mediante el cual el robot UR5 coge un objeto determinado. En RoboDK cuando un robot coge un objeto éste se anida dentro de la herramienta del robot, haciendo así, que el objeto se mueva solidario a la herramienta.

2. **"Coger_Pack":** su funcionamiento es el siguiente: el robot se dirige a la posición de recogida de un conjunto de botellas y coge un lote. A continuación, el robot retrocede y se mueve a una posición de espera ("APP_BOT") en ese momento se ejecuta el programa "Next_Bottle.py" y recoge otras seis botellas. Acto seguido el robot regresa a la posición inicial y, paralelamente, se vuelve a ejecutar el programa "Next_Bottle.py".

3. **"Dejar_Pack":** este programa permite que el robot deje el objeto que haya cogido en un sistema de referencia determinado (en este caso se deja en la cinta de rodillos).

Ilustración 35. Programas de la línea.

4. "Soltar_botella": su funcionamiento es el siguiente: primero fija el valor de "RODILLOS_DONE" a uno (esto se hace para permitir que, al principio de la simulación, el robot UR16e deposite exitosamente una caja vacía en la cinta de rodillos).

A continuación, el robot se dirige a la cinta de rodillos dónde espera a que el robot UR16e termine de colocar una caja ("DOBLADO" valga uno). Una vez colocada la caja el robot UR5 deposita las doce botellas en su interior. Finalmente, el robot regresa a su posición inicial

5. **“Application_BOXING”:** primero fija el valor de la variable “PACK_LISTO” a cero, avisando así, de que aún no ha comenzado el proceso. Acto seguido, ejecuta los programas “Coger_Pack” y “Dejar_Pack” momento en el que fija el valor de “PACK_LISTO” a uno puesto que ya se ha completado el proceso con éxito.

Línea de doblado y preparado de cajas (elementos de programación empleados)

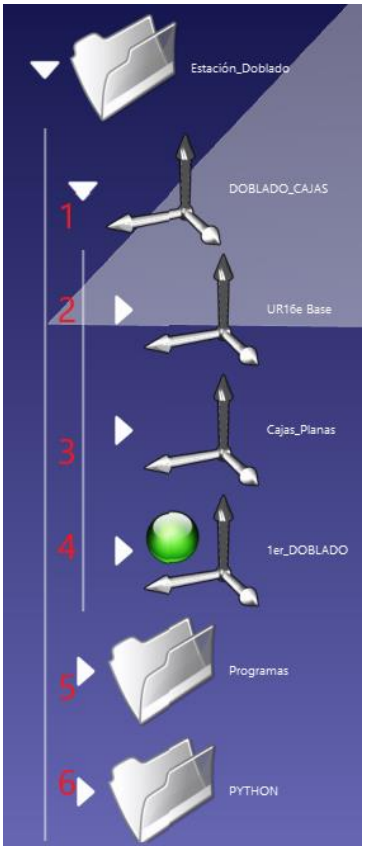


Ilustración 36. Programación línea de doblado y preparado de cajas.

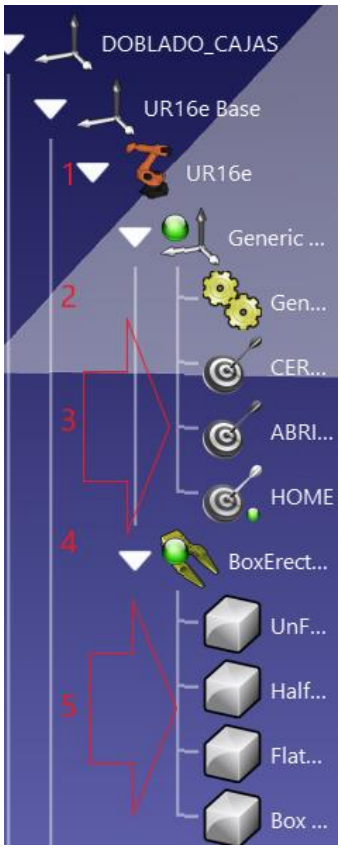


Ilustración 37. Desglosamiento del UR16e Base.

En la *Ilustración 36.* podemos observar todos los elementos que conforman la línea. A continuación, se procederá a realizar una breve descripción de cada uno de ellos:

1. **“DOBLADO_CAJAS”:** sistema de referencia global de la línea. En su interior podemos encontrar la totalidad de objetos, mecanismos y robots que intervienen el proceso.

2. **“UR16e Base”:** sistema de referencia del robot UR16e. En la *ilustración 37.* se puede observar desglosado.

3. **“Cajas_Planas”:** sistema de referencia en el que se encuentran los puntos necesarios para la programación de la aproximación y recogida de los cartones. En su interior también están las propias cajas y la mesa sobre la que descansan.

4. **“1er_DOBLADO”:** sistema de referencia el que se pueden encontrar todos los puntos que intervienen en el proceso de doblado y colocación de la caja.

5. **Carpeta de programas de RoboDk:** lugar en el que se encuentran todos los programas necesarios para la realización del proceso de doblado y preparado de cajas.

6. **Carpeta Python:** contiene al programa “CreateOpenBox.py” utilizado en la generación de cajas vacías cuando el robot se encuentra en posición de dejar la caja.

En la *Ilustración 37.* se pueden apreciar los siguientes elementos:

1. **Robot UR16e**

2. **“Generic Box Erector Gripper”:** mecanismo de un eje lineal rotativo, el cual determina el grado de apertura de la pinza de la herramienta.

3. **Conjunto de puntos de la herramienta:** puntos utilizados en la programación de la apertura o cierre de la pinza de la herramienta.

4. **“BoxErectorToolTcp”:** herramienta que se emplea durante el proceso de doblado y preparado de cajas.

5. **Versiones de una caja:** consiste en un total de cuatro objetos que representan una versión más o menos cercana a la versión final de la caja que será depositada en la cinta. Se usan para dotar de realismo al proceso de transformación de un cartón a una caja funcional.

Línea de doblado y preparado de cajas (programas empleados)

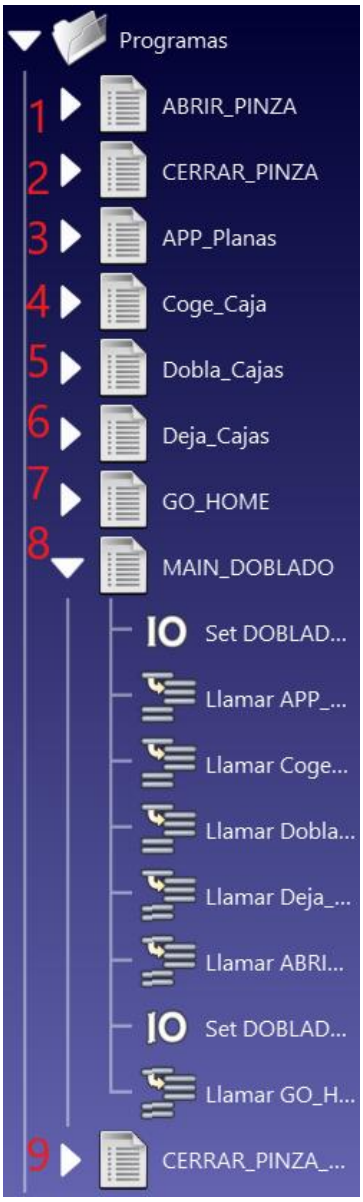


Ilustración 38. Programas de la línea.

1. **"ABRIR_PINZA":** programa ligado al mecanismo de la pinza de la herramienta que permite abrir la pinza.

2. **"CERRAR_PINZA":** al ejecutarlo se cierra la pinza de la herramienta.

3. **"APP_Planas":** hace que el robot se acerque al conjunto de cajas planas mediante una sucesión de movimientos rectilíneos.

4. **"Coge_Caja":** programa mediante el cual el robot coge uno de los cartones situados encima de la mesa de color gris.

5. **"Dobla_Cajas":** programa en el que a partir de la opción de hacer aparecer/desparecer un objeto se consigue una simulación convincente del proceso de doblado de una caja.

6. **"Deja_Cajas":** la sucesión de eventos dentro del programa es la siguiente: primero el robot espera a que la cinta se detenga tras haber avanzado todo lo necesario ("RODILLOS_DONE vale uno). A continuación, deposita la caja encima de la cinta trasportadora y se ejecuta el programa "CreateOpenBox.py" generando así una nueva caja.

7. **"GO_HOME":** Una vez el robot ha dejado la caja éste regresa a su posición inicial y se prepara para producir una nueva.

8. **“MAIN_DOBLADO”:** primero se fija el valor de la variable “DOBLADO” a cero (aún no hay una nueva caja lista en le cinta), a continuación, se ejecutan de forma secuencial varios de los programas anteriormente descritos (desde el 1 al 6 en orden creciente). En ese momento se avisa de que ya hay una caja lista en la cinta (“DOBLADO” pasa a valer uno) y, finalmente, el robot retorna a su posición inicial.

9. **“CERRAR_PINZA_AUX”:** se ejecuta al principio del programa “APP_planas” para poder cerrar la pinza y así evitar colisiones que se daban durante la simulación.

Línea de sellado y paletizado (elementos de programación empleados)

La línea de sellado y paletizado consta de dos ramas claramente diferenciadas, la rama dedicada al transporte y sellado de los paquetes (parte denominada como "CINTA_RODILLOS") y la rama dedicada al proceso de paletizado de los paquetes (en el proyecto aparece como "Estación_paletizado").

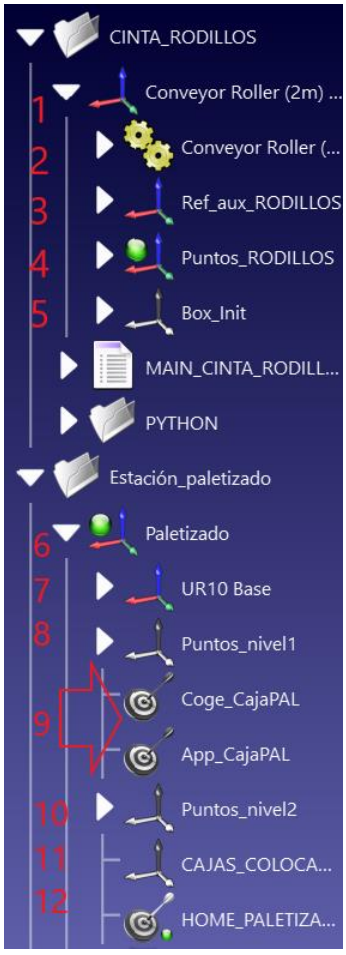


Ilustración 39. Programación de la línea de sellado y paletizado.

1. **"Conveyor Roller (2m) Base":** sistema de referencia principal en el que encontramos todos los elementos de programación empleados en la parte de sellado de paquetes.

2. **"Conveyor Roller (2m)":** mecanismo de un eje lineal empleado para el transporte de los diferentes paquetes. En su interior se encuentra un sistema de referencia al que se van adjuntando los diferentes paquetes a medida que son producidos por los robots UR5 y UR16e.

3. **"Ref_aux_RODILLOS":** sistema de referencia en el que se encuentran dos objetos invisibles utilizados durante la ejecución del programa "nextBOX.py" que se explicará más adelante.

4. **"Puntos_RODILLOS":** en su interior se encuentran todos los puntos empleados en el "place" de las botellas por parte del robo UR5.

5. **"Box_Init":** sistema de referencia en el que se encuentran anidados una serie de objetos empleados en el proceso de reinicio de la simulación.

6. **"Paletizado":** sistema de referencia fundamental en el que se pueden encontrar todos los elementos de programación empleados en la estación de paletizado.

7. **“UR10 Base”:** en su interior se ubican el robot UR10 y su herramienta.

8. **“Puntos_nivel1”:** contiene todos los puntos empleados (objetivos de “approach” y “place”) en el proceso de paletizado del primer nivel de paquetes.

9. **Objetivos para la recogida de paquetes:** dos puntos que se emplean en la simulación cuando el robot recoge un paquete de la cinta transportadora de rodillos.

10. **“Puntos_nivel2”:** conjunto de puntos empleados en el proceso de paletizado del segundo nivel de paquetes.

11. **“CAJAS_COLOCADAS”:** sistema de referencia al que se irán ligando los paquetes procesados una vez han sido paletizados.

12. **“HOME_PALETIZADO”:** punto del cual parte el robot UR10 al principio de la simulación.

Línea de sellado y paletizado (programas empleados)

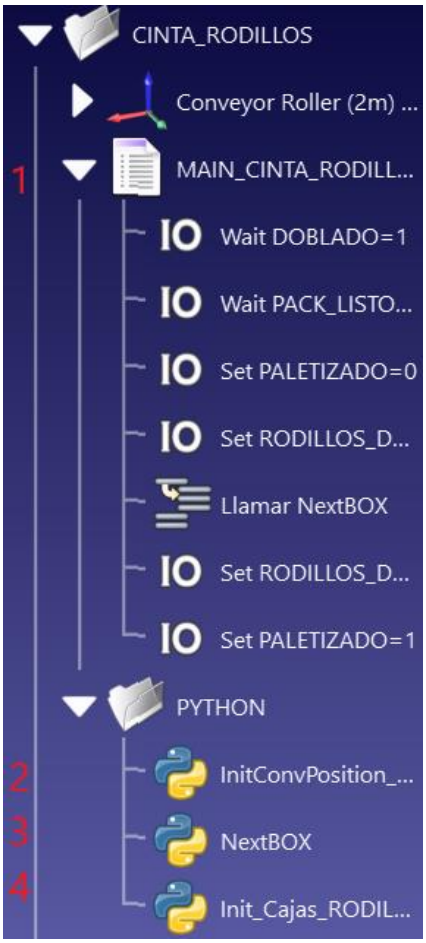


Ilustración 40. Programas empleados en la parte de transporte y sellado de paquetes.

Ilustración 41. Programas empleados durante el proceso de paletizado.

En la *ilustración 40.* se pueden diferenciar los siguientes programas:

1. **"MAIN_CINTA_RODILLOS":** programa diseñado en RoboDk cuya función principal es asegurar que todos los procesos del proyecto están funcionando de forma coordinada antes y después de ejecutar el programa "NextBOX.py". Para ello, primero comprueba que se se ha terminado la preparación de un paquete ("DOBLADO" y "PACK_LISTO" valen uno).

A continuación, fija a cero las variables “PALETIZADO” y “RODILLOS_DONE”, puesto que aún no ha llegado un paquete a la posición de recogida y la cinta transportadora está a punto de moverse (programa “NextBOX.py” a punto de ejecutarse). Acto seguido, se ejecuta el programa “NextBOX.py” y la cinta comienza a moverse.

Finalmente, fija a uno el valor de las salidas digitales de “PALETIZADO” y “RODILLOS_DONE” ya que hay un paquete en posición de ser recogido (el robot UR10 puede paletizarlo) y la cinta se ha vuelto a parar tras haber recorrido la distancia necesaria.

2. **“InitConvPosition_Out.py”:** programa que reinicia la posición de la cinta transportadora.

3. **“NextBOX.py”:** programa que se encarga del avance de la cinta transportora y del proceso de sellado de los paquetes.

4. **“Init_Cajas_RODILLOS.py”:** reinicia la posición de los paquetes a la que tenían antes de empezar la simulación.

En la *ilustración 41.* se pueden pueden observar los siguientes programas:

1. **“AttachBOX”:** programa mediante el cual el robot UR10 coge un paquete y lo anida dentro de su herramineta.

2. **“PlaceBOX”:** programa que permite al robot UR10 depositar un paquete en el pallet (desliga el paquete de su herrmienta y lo anida dentro del sistema de referencia “CAJAS_COLOCADAS”).

3. **“PickBOX”:** permite al robot UR10 coger un paquete de la cinta de rodillos y y moverlo una vez lo ha cogido. El funcionamiento es el siguiente: primero el robot se acerca a la posición de recogida de un paquete y espera a que un paquete haya llegado al final de la cinta transportadora (“PALETIZADO” vale uno).

   A continuación, fija el valor de la variable “PALETIZADO” a cero, puesto que la detección del paquete ya ha concluido y, si no se reinicia su valor a cero, se estaría mandando una señal constante de detección de un paquete, por lo que el robot seguiría paletizando sin que hubiera ningún paquete en posición.

Finalmente, el robot coge un paquete ejecutando “AttachBOX” y levanta el paquete unos cuanto centímetros de la cinta.

4. **“GO_HOME_PALETIZADO”:** mueve el robot a una posición de inicio. Se usa durante el reinicio de la simulación.

5. **“Reset_pallet.py”:** borra todos los paquetes depositados en el pallet durante la simulación. Se usa durante el reincio de la simulación.

6. **“Programas_paletizado_L1”:** carpeta en la que se pueden encontrar un total de veintiseite programas dedicados al paletizado de veintisiete paquetes en el primer nivel. Todos los programas presentan la misma estructura, la única diferencia entre ellos son los objetivos a los que van ligados los movimientos del robot.

   El funcionamiento genérico de estos programas es el siguiente: primero, se ejecuta el programa “PickBOX”. A continuación, el robot se dirige a uno de los puntos definidos en el sitema de referencia “Puntos_nivel1” y deposita el paquete en el pallet (se ejecuta el programa “PlaceBOX). Finalmente, el robot se aleja un poco de la posición en la que ha dejado el paquete.

7. **“Programas_paletizado_L2”:** carpeta en cuyo interior se encuantran otros veintiseite programas que paletizan veintisiete paquetes en el segundo nivel. La idea es exactamente la misma que la descrita el apartado anterior.

8. **“Paletizado_L1”:** consiste en una sucesión de llamadas de los veintisiete programas relacionados con el paletizado del nivel uno. Permite establecer un orden a la hora de paletizar los paquetes.

9. **“Paletizado_L2”:** mismo concepto que el punto anterior pero aplicado al nivel dos del proceso de paletización.

10. **“MAIN_PALETIZADO”:** programa principal del proceso de paletizado. Primero ejecuta el programa “Paletizado_L1”. A continuación, el programa “Paletizado_L2” y, finalmente, ejecuta el programa “Reset_pallet.py”.

    Puesto que entender el funcionamiento de este programa puede resultar un poco complicado en las próximas líneas se intentará explicar de forma más clara.

En primer lugar se ejecuta el programa “Paletizado_L1” y, por tanto, se ejecuta el primer programa de los veintisiete anidados en la carpeta “Programas_paletizado_L1”, es decir, el programa “PROG-1-L1”.

Aquí empieza realmente el proceso ya que se ejecuta el programa “PickBOX” (el robot espera a que pueda coger un paquete, lo coge y lo levanta de la cinta) y, a continuación, lo lleva hasta el pallet y lo deposita en la posición adecuada. En este punto deja de ejcutarse el “PROG-1-L1” y comienza a jecutarse el programa “PROG-2-L2” por lo que vuelve a ejecutarse el programa “PickBOX” y deja el seugundo paquete en la posición que le ha sido asignada. Aquí deja de ejecutarse el programa “PROG-2-L2” y sigue con el siguiente programa hasta completar los veintisiete programas del nivel uno.

Una vez se ha ejecutado el último programa de “Paletizado_L1” comienza el paletizado del nivel dos mediante la llamada del programa “Paletizado_L2” cuyo funcionamiento secuencial es el mismo que el del “Paletizado_L1”. Finalmente, se reinicia el pallet ejecutando el programa “Reset_pallet.py”.

Programas principales del proyecto

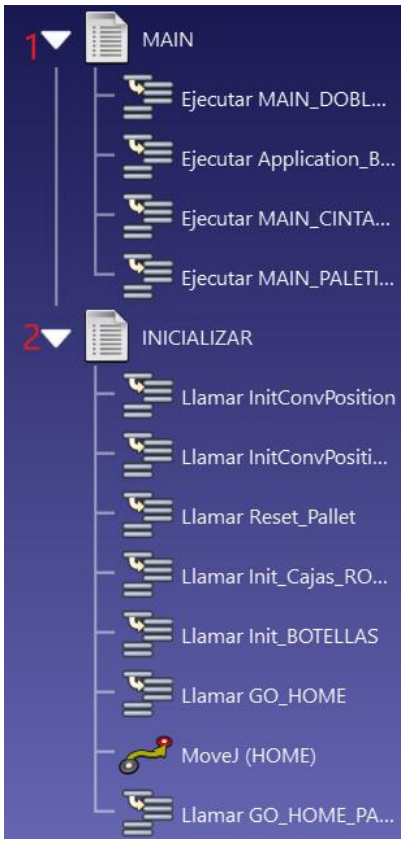


Ilustración 42. Programas principales del proyecto.

1. **“MAIN”:** el programa principal de este proyecto consiste en la ejecución simultánea de los distintos programas principales de cada una de las líneas descritas anteriormente.

   Es decir, se ejecutan al mismo tiempo los programas “MAIN_DOBLADO”, “Application_BOXING”, “MAIN_CINTA_RODILLOS” y “MAIN_PALETIZADO”. Con esto conseguimos llevar a cabo el proceso productivo deseado.

2. **“INICIALIZAR”:** programa que sirve para reiniciar la simulación al completo mediante la ejecución secuencial de varios programas.

8.2 Programación en Python

En este apartado se tiene la intención de explicar, de forma clara y precisa, el funcionamiento de los distintos programas escritos en lenguaje Python que se han empleado a lo largo del proyecto. Se avisa al lector que se evitará, en la medida de lo posible, explicar más de una vez la misma función con la finalidad de conseguir una explicación lo menos redundante posible. Las explicaciones de las funciones empleadas en la programación de este apartado se han realizado a través de la información disponible en la página web de RoboDk (RoboDk API).

InitConvPosition.py/InitConvPosition_Out.py

```
from robolink import *
from robodk import *
RDK = Robolink()

MECHANISM_NAME = 'Conveyor Belt (4.5m)'
mechanism = RDK.Item(MECHANISM_NAME,itemtype=ITEM_TYPE_ROBOT)

if mechanism.Valid():
    mechanism.setJoints([0])
```

Ilustración 43. Programa InitConvPosition.py.

Primero se llama a los dos módulos “robolink” y “robodk” necesarios para poder empezar a programar en lenguaje Python en RoboDk. A continuación, se define la variable “MECHANISM_NAME” para hacer referencia al mecanismo de la cinta transportadora en RoboDk y se llama a la función *Item ()* para poder obtener el objeto de la simulación en RoboDk.

Finalmente, mediante la función *Valid ()* se comprueba que el objeto al que se ha llamado existe y se reinicia la posición de la cinta llamando a la función *setJoints ()*. El programa "InitConvPosition_Out.py" presenta exactamente la misma estructura.

NextBottle.py/NextBOX.py

```
from robodk import robolink
RDK = robolink.Robolink()


MECHANISM_NAME = 'Conveyor Belt (4.5m)'
PART_TRAVEL_MM = 270

mechanism = RDK.Item(MECHANISM_NAME,itemtype=robolink.ITEM_TYPE_ROBOT)


NEW_REF_OBJECT_NAME = 'Bottle (Label) HIDE'
NEW_OBJECT_NAME = 'Bottle (Label)AUX'
REFERENCE_FRAME = 'Botella'
REFERENCE_FRAME_REF = 'Ref_Aux'


frame_conv = RDK.Item('Botella',itemtype=robolink.ITEM_TYPE_FRAME)


obj_list = frame_conv.Childs()

RDK.Render(False)

for item in obj_list:

    if item.PoseAbs()[0,3] > 2000 and item.PoseAbs()[0,3] < 2300 :
        item.Delete()


frame = RDK.Item(REFERENCE_FRAME, robolink.ITEM_TYPE_FRAME)
frame_ref = RDK.Item(REFERENCE_FRAME_REF, robolink.ITEM_TYPE_FRAME)
ref_obj = RDK.Item(NEW_REF_OBJECT_NAME, robolink.ITEM_TYPE_OBJECT)


if frame.Valid() and  ref_obj.Valid():
    ref_obj.Copy()
    new_obj = RDK.Paste()
    new_obj.setVisible(False)
    new_obj.setName(NEW_OBJECT_NAME)
    new_obj.setParent(frame_ref)
    RDK.Update()
    new_obj.setParentStatic(frame)
    new_obj.setVisible(True)
RDK.Render(True)

NEW_REF_OBJECT_NAME1 = 'Bottle HIDE'
NEW_OBJECT_NAME1 = 'BottleAUX'
REFERENCE_FRAME1 = 'Botella'
REFERENCE_FRAME_REF1 = 'Ref_Aux'

frame = RDK.Item(REFERENCE_FRAME1, robolink.ITEM_TYPE_FRAME)
frame_ref = RDK.Item(REFERENCE_FRAME_REF1, robolink.ITEM_TYPE_FRAME)
ref_obj = RDK.Item(NEW_REF_OBJECT_NAME1, robolink.ITEM_TYPE_OBJECT)

if frame.Valid() and  ref_obj.Valid():
    ref_obj.Copy()
    new_obj = RDK.Paste()
    new_obj.setVisible(False)
    new_obj.setName(NEW_OBJECT_NAME1)
    new_obj.setParent(frame_ref)
    RDK.Update()
    new_obj.setParentStatic(frame)
    new_obj.setVisible(True)
RDK.Render(True)


if mechanism.Valid():
    mechanism.MoveJ(mechanism.Joints() + PART_TRAVEL_MM)
```

Ilustración 44. Programa NextBottle.py.

El programa comienza definiendo el nombre del mecanismo que se quiere manipular en RoboDk y la distancia que ha de recorrer la cinta cada vez que se ejecute el programa. A continuación, se definen otras cuatro variables: "NEW_REF_OBJECT_NAME" (nombre del objeto a copiar), "NEW_OBJECT_NAME" (nombre del nuevo objeto creado), "REFERENCE_FRAME" (sistema de referencia dónde será creado el nuevo objeto) y "REFERENCE_FRAME_REF" (sistema de referencia en el que se encuentra el objeto a copiar).

Para poder acceder a las botellas dentro del sistema de referencia "Botella" se llama a la función *Childs ()* y se procede, mediante el uso de la función *Delete ()*, al borrado de todos los objetos que se encuentren entre las coordenadas absolutas [2000, 2300] en dirección del eje X (ya que [0, 3] se refiere a la línea 0 de la columna 3 de la matriz de posición, es decir, la posición X).

Más adelante se procede a la generación de las botellas con etiqueta copiando el aspecto del objeto "Bottle (Label) HIDE", se pone nombre al objeto copiado, se traslada al sistema de referencia deseado y, finalmente, se hace visible. En esta parte del programa se hacen uso de las funciones *Render ()* y *Update ()* utilizadas para renderizar y actualizar la estación cada vez que es editada.

Las próximas líneas de código permiten la generación de botellas sin etiquetar. Las funciones y herramientas de programación que se han empleado son las mismas que las que se emplearon para la generación de botellas etiquetadas y, por tanto, no se explicarán para evitar redundancia.

En las últimas dos líneas de código se procede al movimiento de la cinta mediante el uso de la función *MoveJ ()* hasta haber recorrido la cantidad definida en la variable "PART_TRAVEL_MM". El funcionamiento del programa "NextBOX.py" es similar y, por tanto, no merece la pena explicar lo mismo de nuevo.

Init_BOTELLAS.py/Init_Cajas_RODILLOS.py

```
from robodk import robolink
RDK = robolink.Robolink()


CONV_FRAME_NAME = 'Botella'
NEW_REF_OBJECT_NAME1 = 'Bottle'
NEW_OBJECT_NAME1 = 'Bottle_AUX'
NEW_REF_OBJECT_NAME2 = 'Bottle (Label)'
NEW_OBJECT_NAME2 = 'Bottle (Label)_AUX'
REFERENCE_FRAME = 'Botella'
REFERENCE_FRAME_REF = 'Bottellas_init'

frame = RDK.Item(CONV_FRAME_NAME,robolink.ITEM_TYPE_FRAME)

MECHANISM_LIST = ["Conveyor Belt (4.5m)"]


frame = RDK.Item(REFERENCE_FRAME, robolink.ITEM_TYPE_FRAME)
frame_ref = RDK.Item(REFERENCE_FRAME_REF, robolink.ITEM_TYPE_FRAME)

ref_obj1 = RDK.Item(NEW_REF_OBJECT_NAME1, robolink.ITEM_TYPE_OBJECT)
ref_obj2 = RDK.Item(NEW_REF_OBJECT_NAME2, robolink.ITEM_TYPE_OBJECT)

RDK.Render(False)

for i in range(len(MECHANISM_LIST)):
    MECHANISM_NAME=MECHANISM_LIST[i]
    mechanism = RDK.Item(MECHANISM_NAME,itemtype=robolink.ITEM_TYPE_ROBOT)
    if mechanism.Valid():
        mechanism.setJoints([0])

if frame.Valid():
    for item in frame.Childs():
        if item.Type() == robolink.ITEM_TYPE_OBJECT:
            item.Delete()

RDK.Render(False)

for i in range(len(MECHANISM_LIST)):
    MECHANISM_NAME=MECHANISM_LIST[i]
    mechanism = RDK.Item(MECHANISM_NAME,itemtype=robolink.ITEM_TYPE_ROBOT)
    if mechanism.Valid():
        mechanism.setJoints([0])

if frame.Valid():
    for item in frame.Childs():
        if item.Type() == robolink.ITEM_TYPE_OBJECT:
            item.Delete()

for item in frame_ref.Childs():
    if frame.Valid() and  ref_obj1.Valid() and ref_obj2.Valid():
        item.Copy()
        new_obj1 = RDK.Paste()
        new_obj1.setVisible(False)
        new_obj1.setName(NEW_OBJECT_NAME1)
        new_obj1.setParent(frame_ref)
        RDK.Update()
        new_obj1.setParentStatic(frame)
        new_obj1.setVisible(True)

RDK.Render(True)
```

Ilustración 45. Programa Init_BOTELLAS.py.

En primer lugar, se define una serie de variables que determinarán qué objetos se quieren copiar, dónde se sitúan estos objetos, qué nombre tendrán las copias creadas y dónde se depositarán las copias creadas.

A continuación, se crea el vector "MECHANISM_LIST" y se reinicia la posición de todos los mecanismos que hayan sido incluidos en dicho vector. Posteriormente se borran todos los objetos anidados en el sistema de referencia definido en la variable "CONV_FRAME_NAME" (verificando que son objetos mediante el uso de la función *Type ()*) y se crean todas las copias de los objetos deseados. La forma de generar todas estas copias es la misma que la empleada en el programa "NextBottle.py".

El programa "Init_Cajas_RODILLOS.py" presenta exactamente la misma estructura.

CreateOpenBox.py

```
from robodk import robolink
from robodk import robomath
RDK = robolink.Robolink()

NEW_REF_OBJECT_NAME = 'Box 12x10in HIDE'
NEW_OBJECT_NAME = 'OpenedBoxAUX'
REFERENCE_FRAME = 'CAJAS_RODILLOS'
REFERENCE_FRAME_REF = 'Ref_aux_RODILLOS'

frame = RDK.Item(REFERENCE_FRAME, robolink.ITEM_TYPE_FRAME)
frame_ref = RDK.Item(REFERENCE_FRAME_REF, robolink.ITEM_TYPE_FRAME)
ref_obj = RDK.Item(NEW_REF_OBJECT_NAME, robolink.ITEM_TYPE_OBJECT)

if frame.Valid() and  ref_obj.Valid():
    ref_obj.Copy()
    new_obj = RDK.Paste()
    new_obj.setVisible(False)
    new_obj.setName(NEW_OBJECT_NAME)
    new_obj.setParent(frame_ref)
    RDK.Update()
    new_obj.setParentStatic(frame)
    new_obj.setVisible(True)
RDK.Render(True)
```

Ilustración 46. Programa CreateOpenBox.py.

Tras haber definido cuales van a ser los objetos a copiar, el nombre que tendrán las copias y las localizaciones tanto del objeto a copiar como de la copia. Se ejecuta parte del código explicado en el programa "NextBottle.py" y se crea una nueva caja lista para ser utilizada.

ResetPallet.py

```
from robodk import robolink
RDK = robolink.Robolink()

PALLET_FRAME_NAME = 'CAJAS_COLOCADAS'

frame = RDK.Item(PALLET_FRAME_NAME,robolink.ITEM_TYPE_FRAME)

RDK.Render(False)

if frame.Valid():
    for item in frame.Childs():
        if item.Type() == robolink.ITEM_TYPE_OBJECT:
            item.Delete()
```

Ilustración 47. Programa ResetPallet.py.

Programa muy sencillo que permite borrar todos los objetos depositados en el sistema de referencia ligado a la variable "PALLET_FRAME_REFERENCE". Las funciones que aparecen ya han sido explicadas previamente.

9 OPTIMIZACIÓN DEL TIEMPO DE CICLO

Introducción

Como se comentó al principio de la memoria, uno de los principales objetivos del proyecto es poder optimizar el tiempo de ciclo del proceso de fabricación a través de la simulación en RoboDk. Para ello, RoboDk cuenta con las siguientes herramientas:

- **"Rounding" o redondeo:** instrucción que permite mantener la máxima velocidad al desplazar el robot de un punto a otro. Por ejemplo, si el robot se encuentra en un punto A y, pasando por un punto B, se quiere alcanzar un punto C. Al añadir una instrucción de "rounding" en el movimiento de A-B, se consigue que el robot no llegue a detenerse nunca en B (siempre y cuando esto sea físicamente posible) y, por tanto, no pierda velocidad a la hora de llegar hasta C.

Si se establece un valor positivo de “rounding” se suavizan las transiciones entre punto y punto, consiguiendo así, el efecto anteriormente comentado. Sin embargo, si el valor establecido es negativo, el robot se detendrá en cada punto al que se dirija de manera precisa, aumentando así, el tiempo del proceso.

- **Velocidades y aceleraciones del robot:** en RoboDk existen dos tipos de velocidades y aceleraciones distintas. Velocidad y aceleración lineal (mm/s y mm/s^2) y velocidad y aceleración angular (grados/s y grados/s^2). Las dos primeras influyen en los movimientos lineales programados y las segundas en los movimientos no lineales.

- **Python:** en este apartado se hará uso del fichero de Python “CicleTimeDisplayAll.py” que será útil para estimar la duración en segundos de cada programa.

En las especificaciones técnicas de los robots empleados en la estación disponibles en la *Bibliografía* se puede observar que los valores máximos de velocidad angular están en torno a los 180 grados/s y los de velocidad lineal en torno a unos 1000 mm/s. En cuanto a las aceleraciones se establecerán unos máximos de 800 grados/s^2 y de 2500mm/s^2 puesto que son valores realistas teniendo en cuenta los robots empleados.

A continuación, se van a exponer un total de tres versiones diferentes del proyecto, a través de las cuales se ha ido optimizando el tiempo de ciclo del proceso. La primera versión es la más rudimentaria mientras que la última cuenta con todos los cambios pertinentes para cumplir el objetivo deseado.

1ª versión

Esta es una versión en la que no se le ha prestado atención alguna al tiempo del proceso. La programación del proyecto no cuenta con ninguna instrucción de “rounding”, las velocidades de los robots no están optimizadas para que trabajen de forma coordinada y, además, la disposición de los diferentes elementos en la estación de doblado y preparado de cajas es bastante ineficiente, puesto que cada vez que el robot ha de preparar un paquete ha de realizar un recorrido de 360º, lo que alarga de manera notable el tiempo empleado en esta tarea. En el *Anexo I* se pueden encontrar, desarrollados, los programas empleados en esta versión.

Tras haber ejecutado el fichero de “CycleTimeDisplayAll.py” se obtuvieron los siguientes resultados:

Application_BOXING	100 %	4076.5 mm	8.9 s
Paletizado-L1	100 %	77195.1 mm	196.3 s
Paletizado-L2	100 %	77511.9 mm	201.8 s

Ilustración 48. Tiempos Pick and Place y paletizado 1ª versión.

A estos programas se les ha de sumar el tiempo de ejecución de los programas “NextBottle.py” y “NextBOX.py” puesto que son programas de Python y no se contabilizan al ejecutar el programa “CicleTimeDisplayAll.py”. Se ha medido manualmente que el proceso de etiquetado junto con el movimiento de la cinta donde se sitúan las botellas tarda 7.46 s. mientras que el tiempo del proceso de sellado junto con el movimiento de la cinta de rodillos es de 6.89 s. estos tiempos son aproximados ya que factores el rendimiento de la CPU, la velocidad de procesamiento del equipo o la precisión a la hora de detener el cronómetro influyen en la veracidad de la medida.

APP_Planas	100 %	1372.2 mm	9.0 s
Coge_Caja	100 %	1594.8 mm	10.1 s
Dobla_Cajas	100 %	1918.0 mm	10.5 s
Deja_Cajas	100 %	1367.9 mm	7.1 s
GO_HOME	100 %	4293.7 mm	9.3 s

Ilustración 49. Tiempos estación de doblado 1ª versión.

Para poder calcular de manera correcta el tiempo de procesado de una caja se ha de sumar todos los tiempos de la *ilustración 49* (esto es así porque el fichero “CicleTimeDisplayAll.py” contabiliza el tiempo de ejecución de los programas desde el punto en el que se encuentra el robot al momento de ejecutar el programa de Python y, puesto que el procedimiento de doblado de una caja es secuencial, es

decir, cada programa se empieza a ejecutar en el punto dónde acabó el programa anterior, es necesario hacer la suma de todos los programas empezando en el punto que les corresponde). Al hacer la suma obtenemos que el robot UR16e tarda 46.00 s. en terminar su tarea.

Además, cada paquete del nivel uno tarda de media unos 7.27 s. en ser paletizado y unos 7.47 s. si se trata del segundo nivel.

Haciendo la suma de todos los tiempos se obtiene que el tiempo de procesamiento aproximado de un paquete es de 67.62 s. en el primer nivel y de 67.82 s. si es en el segundo nivel.

Es un tiempo altamente ineficiente desde el punto de vista productivo y es una prueba más de la importancia del tiempo a la hora de automatizar un proceso.

Durante esta simulación se han empleado las siguientes velocidades y aceleraciones:

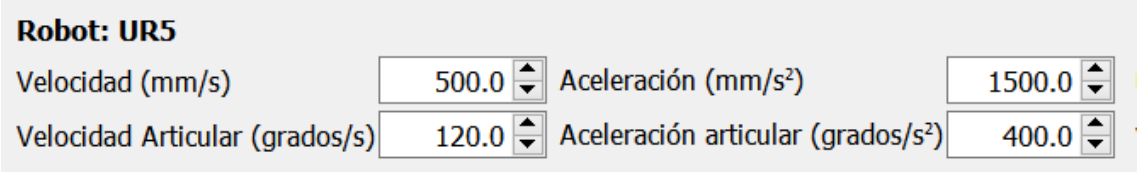


Ilustración 50. Velocidades robot UR5 1ª versión.

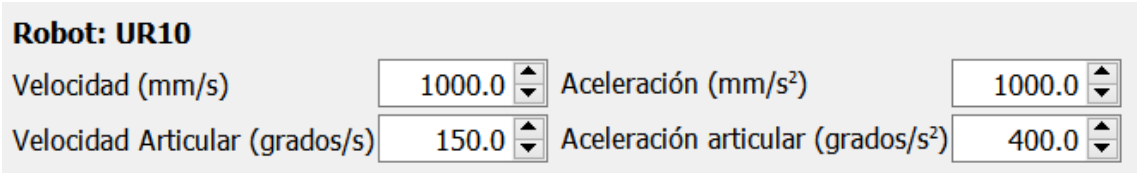


Ilustración 51. Velocidades robot UR16e 1ª versión.

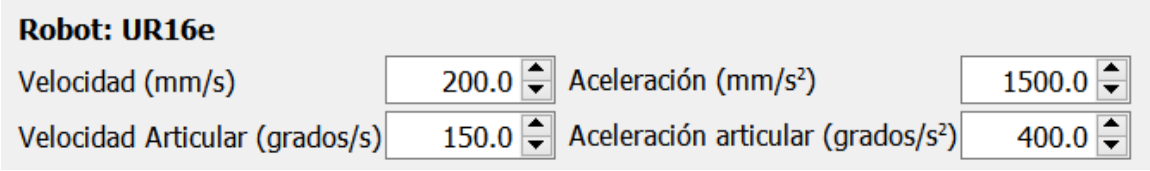


Ilustración 52. Velocidades y aceleraciones robot UR10 1ª versión.

Las velocidades y aceleraciones de las cintas transportadoras empleadas durante la simulación fueron las siguientes:

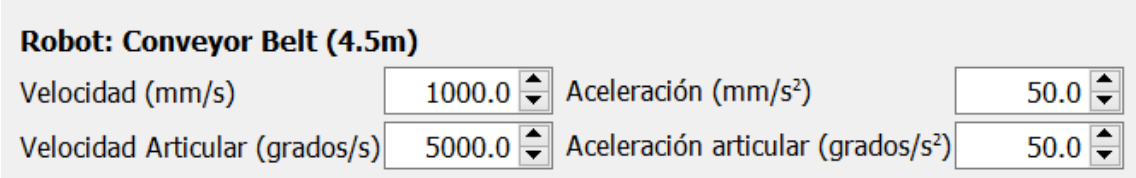


Ilustración 53. Velocidades ya aceleraciones cinta de las botellas 1ª versión.

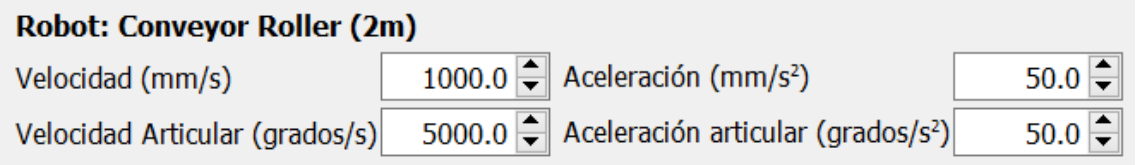


Ilustración 54. Velocidades ya aceleraciones cinta de rodillos 1ª versión.

2ª versión

Analizando la versión anterior es muy fácil entender que el problema principal se encuentra en el proceso de doblado y preparado de las cajas. Este proceso tarda 46.00 s. en realizarse, y puesto que el robot UR5 sólo tarda 16.36 s. en realizar su tarea (ejecución del programa “Application_BOXING” más los 7.46 s. del avance de la cinta), hay casi treinta segundos desperdiciados por ciclo ya que el robot UR5 estará 29.64 s. esperando a que el UR16e deje una caja.

Es por esto por lo que en esta segunda versión del proyecto se ha decidido cambiar la disposición de los elementos de la línea de doblado y preparados de cajas. Con la nueva distribución se ha conseguido reducir notablemente el recorrido del robot UR16e y, por tanto, el tiempo de ciclo del proceso. Además, se ha modificado las velocidades y aceleraciones de los robots con la intención de coordinar y optimizar sus movimientos.

Para conseguir todo lo anterior se han realizado cambios en la parte de programación del proyecto, en el *Anexo II* de la memoria se explica detenidamente cuales han sido los cambios realizados.

Una vez simulado el programa “CycleTimeDisplayAll.py” se obtuvieron los siguientes resultados:

Application_BOXING	100 %	4076.5 mm	7.0 s
Paletizado-L1	100 %	77195.1 mm	196.3 s
Paletizado-L2	100 %	77511.9 mm	201.8 s

Ilustración 55. Tiempos Pick and Place y paletizado 2ª versión.

APP_Planas	100 %	392.3 mm	0.8 s
Coge_Caja	100 %	968.2 mm	1.8 s
Dobla_Cajas	100 %	1867.1 mm	3.4 s
Deja_Cajas	100 %	877.2 mm	1.7 s
GO_HOME	100 %	2507.9 mm	4.8 s

Ilustración 56. Tiempos estación de doblado 2ª versión.

Como se puede observar en la *ilustración 52.* (al sumar todos los tiempos pertinentes) se ha logrado reducir de manera muy notoria el tiempo de ejecución del proceso de doblado y preparado de cajas a 12.5 s y el del “Pick and Place” a 14.46 s. Las velocidades y aceleraciones de las cintas se han mantenido constantes. Tampoco se han modificado las velocidades y aceleraciones del robot UR10.

En la línea de sellado y paletizado de paquetes no se han realizado cambios por lo que los tiempos son los mismos que los de la primera versión. Si hacemos la suma de todos los tiempos necesarios para la preparación de un paquete se obtiene que el tiempo estimado es de 28.62 s. en el primer nivel y de 28.82 s. en el segundo. Este es un tiempo bastante más aceptable.

Las velocidades y aceleraciones que han cambiado en esta versión son las siguientes:

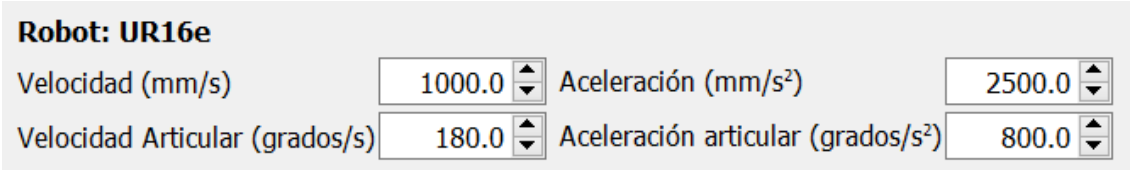


Ilustración 57. Velocidades y aceleraciones robot UR16e 2ª versión.

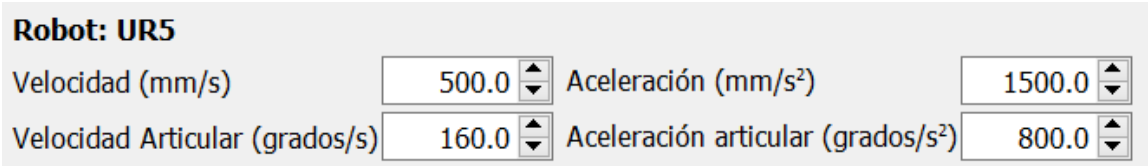


Ilustración 58. Velocidades y aceleraciones robot UR5 2ª versión.

3ª versión

En esta última versión se va a mejorar el tiempo de ciclo añadiendo instrucciones de "rounding" en las partes de los programas en las que la precisión no sea un requisito e incrementando las velocidades y aceleraciones de las cintas transportadoras, de la pinza del robot UR16e y del robot UR10 encargado del proceso de paletizado. En el *Anexo III* se pueden encontrar imágenes de los programas editados en esta versión.

Para calcular el tiempo de ciclo de las líneas de "Pick and Place" de botellas y de doblado de cajas de esta versión no se va a hacer uso del fichero "CicleTimeDisplayAll.py" ya que no tiene en cuenta los cambios de transición "suave" a "precisa" que ocurren dentro de un programa. Por tanto, para conocer el tiempo estimado de un programa se tomará como referencia el valor que RoboDk devuelve al ejecutarlo.

Al haber aumentado la velocidad de las cintas transportadoras el tiempo del proceso de etiquetado y avance de las botellas ahora es de aproximadamente 4.8 s. mientras que el tiempo asociado al proceso de sellado y transporte de los paquetes ronda los 4.1 s. Además, la pinza ahora tarda 1.4 s. en abrirse o cerrarse (casi la mitad de 2.7 s. que era lo que tardaba anteriormente)

Una vez hecho todos los cambios pertinentes se obtuvieron los siguientes resultados:

Coger_Pack tiempo: 2.9s **Dejar_Pack** tiempo: 2.0s

Ilustración 59. Tiempos proceso de "Pick and Place" 3ª versión.

Si a esos 4.9 s. les sumamos 4.8 s. que tarda el proceso de etiquetado junto al avance de la cinta obtenemos que el proceso en total tarda 9.7 s. en completarse exitosamente.

APP_Planas tiempo: 0.4s **Dobla_Cajas** tiempo: 1.9s **Deja_Cajas** tiempo: 1.3s

Coge_Caja	100 %	968.2 mm	1.0 s	GO_HOME	100 %	2507.9 mm	2.9 s

Ilustración 60. Tiempos de la estación de doblado 3ª versión.

La *ilustración 60.* es un "collage" de los tiempos de los programas que intervienen en el doblado y preparado de una caja.

Al sumar todos los tiempos se obtiene que se tarda 7.5 s. en realizar todo el proceso. Por último, en el proceso de paletizado se obtuvieron los siguientes resultados:

Paletizado-L1	100 %	77195.1 mm	146.4 s
Paletizado-L2	100 %	77511.9 mm	149.9 s

Ilustración 61. Tiempos estación de paletizado 3ª versión.

Por tanto, de media se tarda 5.42 s. en paletizar un paquete en el primer nivel y 5.55 s. en el segundo.

Si sumamos todos los tiempos que intervienen en el proceso obtenemos que la preparación de un paquete implica aproximadamente unos 19.22 s. en el primer nivel y unos 19.35 s. en el segundo.

Las velocidades y aceleraciones que se han modificado en esta versión son las siguientes:

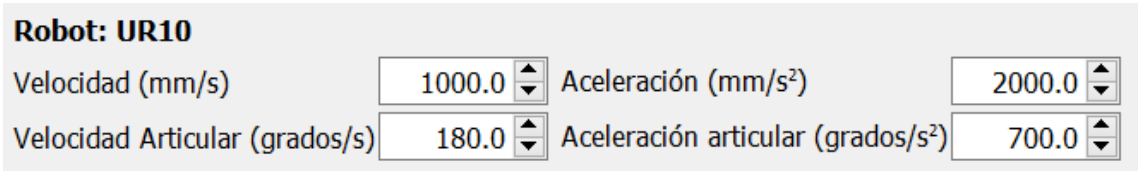


Ilustración 62. Velocidades y aceleraciones robot UR10 3ª versión.

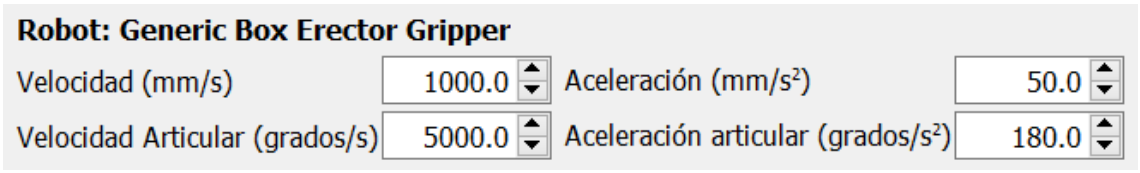


Ilustración 63. Velocidades y aceleraciones pinza 3ª versión.

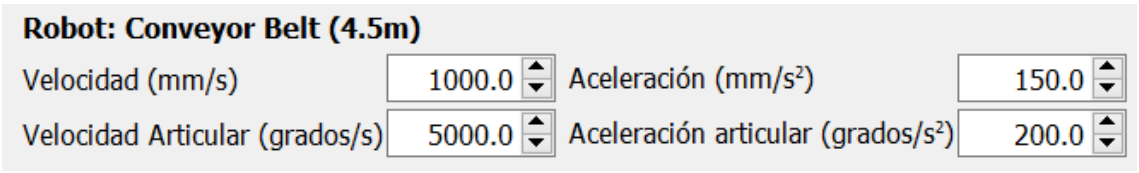


Ilustración 64. Velocidades y aceleraciones cinta de las botellas 3ª versión.

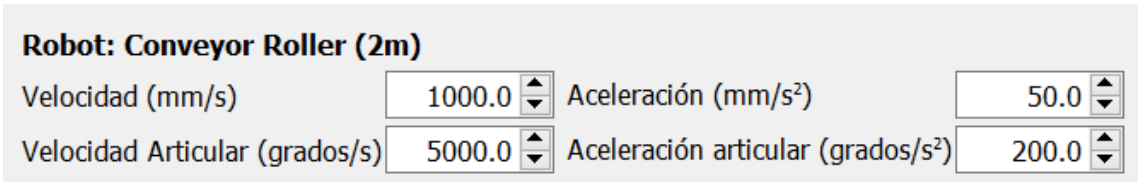


Ilustración 65. Velocidades y aceleraciones cinta de rodillos 3ª versión.

10 POSIBLES MEJORAS

En este proyecto se hubieran podido implementar una serie de funcionalidades que, desde el punto de vista técnico e industrial, hubieran supuesto una mejora interesante.

Algunas de las posibles mejoras que se han valorado son las siguientes:

- **Llenado inicial de las cintas transportadoras**: nada más abrir el archivo de RoboDk del proyecto, aparecen dispuestas, antes de ejecutar el programa, una serie de botellas y paquetes que serán procesados a medida que avance la simulación. Para dotar de un mayor realismo al modelo virtual generado, se valoró la posibilidad de implementar un proceso, antes de arrancar la simulación, de llenado tanto de la cinta transportadora de botellas como de la de paquetes.

- **Implementación de un PLC:** En un momento dado se planteó la posibilidad de realizar la programación de un PLC para controlar el proceso, pero se acabó descartando por falta de soluciones convincentes.

- **Mejora en la programación de los movimientos de los robots:** es probable que, mediante una programación más rigurosa, se hubiera conseguido una mejora en los movimientos de los robots. De esta forma se hubiera podido ahorrar algo más de tiempo y mejorar el alcance de los distintos robots a la hora de realizar sus tareas.

11 CONCLUSIONES

Una vez finalizado este proyecto final de grado me gustaría recalcar la importancia del uso de los gemelos digitales en el mundo de la industria moderna. Mediante el uso de un ordenador con prestaciones limitadas y los softwares adecuados se pueden controlar y mejorar procesos productivos de gran complejidad. El uso de RoboDk permite la programación de un robot sin interrumpir la producción, ahorrando así, una gran cantidad de tiempo y recursos. Esta herramienta resulta muy útil a la hora de diseñar estaciones donde varios robots trabajan coordinadamente y, además, nos permite prevenir errores o implementar mejoras siempre que lo deseemos.

Además, como ya se ha comentado previamente, RoboDk facilita la conversión de modelos CAD en código para robots y admite el uso de más de trescientos tipos de robots corporativos de hasta treinta marcas distintas. Aunque es cierto que presenta una funcionalidad limitada a la hora de la incorporación de sensores, sus prestaciones son potentes a la hora de programar procesos de taladrado, soldadura, impresión 3D, “Pick and Place” o paletizado.

Por último, con la realización de este proyecto he podido aprender cosas nuevas sobre automatización y robótica y, personalmente, considero que RoboDk es una herramienta didáctica muy potente que permite asimilar fácilmente conceptos fundamentales en la programación de robots y procesos industriales.

12 ANEXOS

12.1 Anexo I

En este primer anexo se puede observar, en diferentes ilustraciones, todos los programas que se han desarrollado en RoboDk a lo largo del proyecto. Además, también se puede apreciar la distribución espacial de los objetivos creados para las tres líneas de producción.

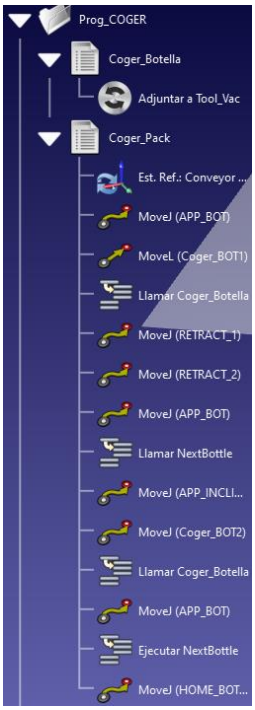


Ilustración 66. Programas 1 “Pick and Place” 1ª versión.

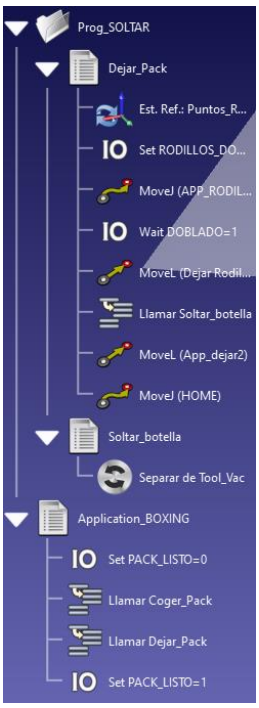


Ilustración 67. Programas 2 “Pick and Place” 1ª versión.

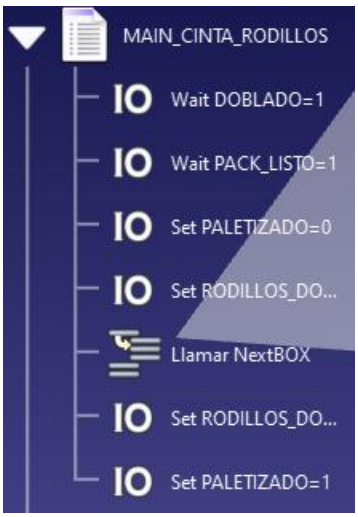


Ilustración 68. Programas cinta de rodillos 1ª versión.

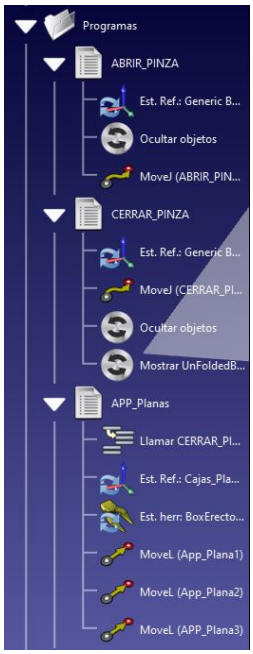


Ilustración 69. Programas estación de doblado 1 1ª versión.

Ilustración 70. Programas estación de doblado 2 1ª versión.

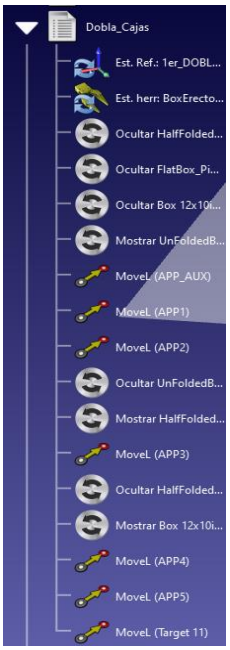


Ilustración 71. Programas estación de doblado 3 1ª versión.

Ilustración 72. Programas estación de doblado 4 1ª versión.

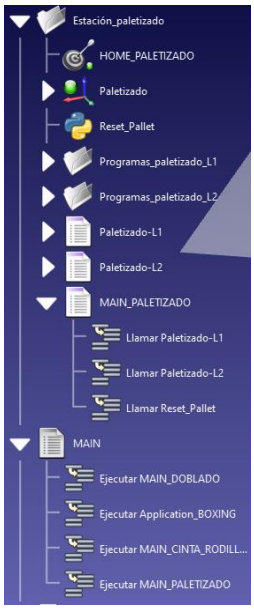


Ilustración 73. Programas estación de paletizado 1 1ª versión.

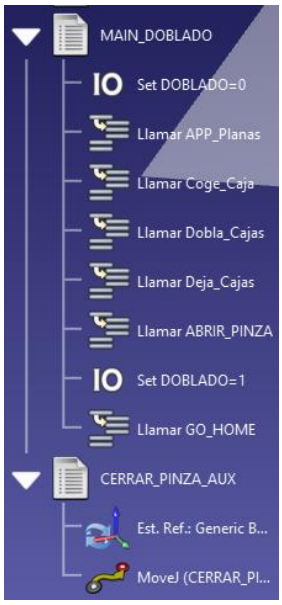


Ilustración 74. Programas principales de la estación 1ª versión.

Ilustración 75. Programas estación de paletizado 2 1ª versión.

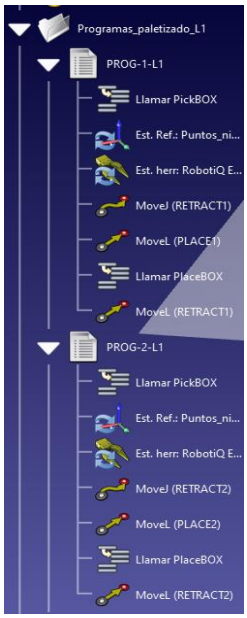


Ilustración 76. Programas estación de paletizado 3 1ª versión.

Cabe recordar que los cincuenta y cuatro programas destinados al paletizado de paquetes presentan la misma estructura, esta se puede observar en la *ilustración* 76.

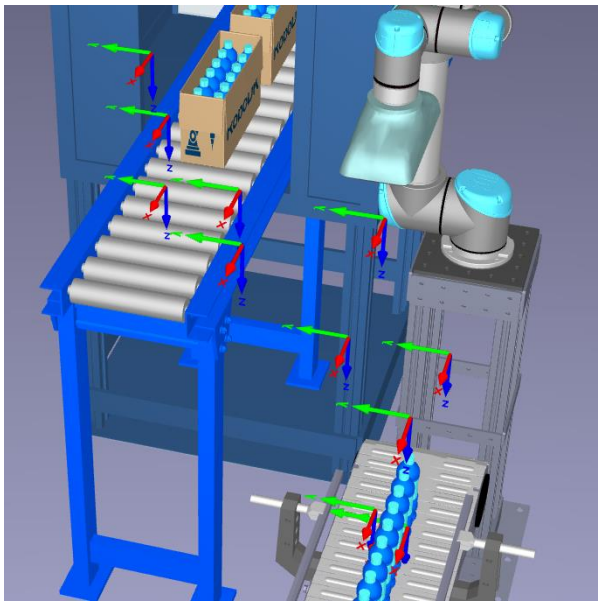


Ilustración 77. Conjunto de puntos de la estación de etiquetado y “Pick and Place” de botellas.

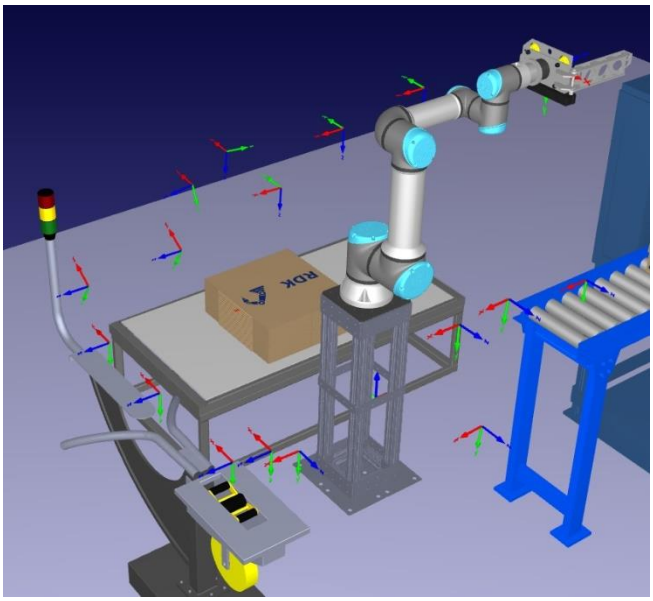


Ilustración 78. Conjunto de puntos de la estación de doblado y preparado de cajas.

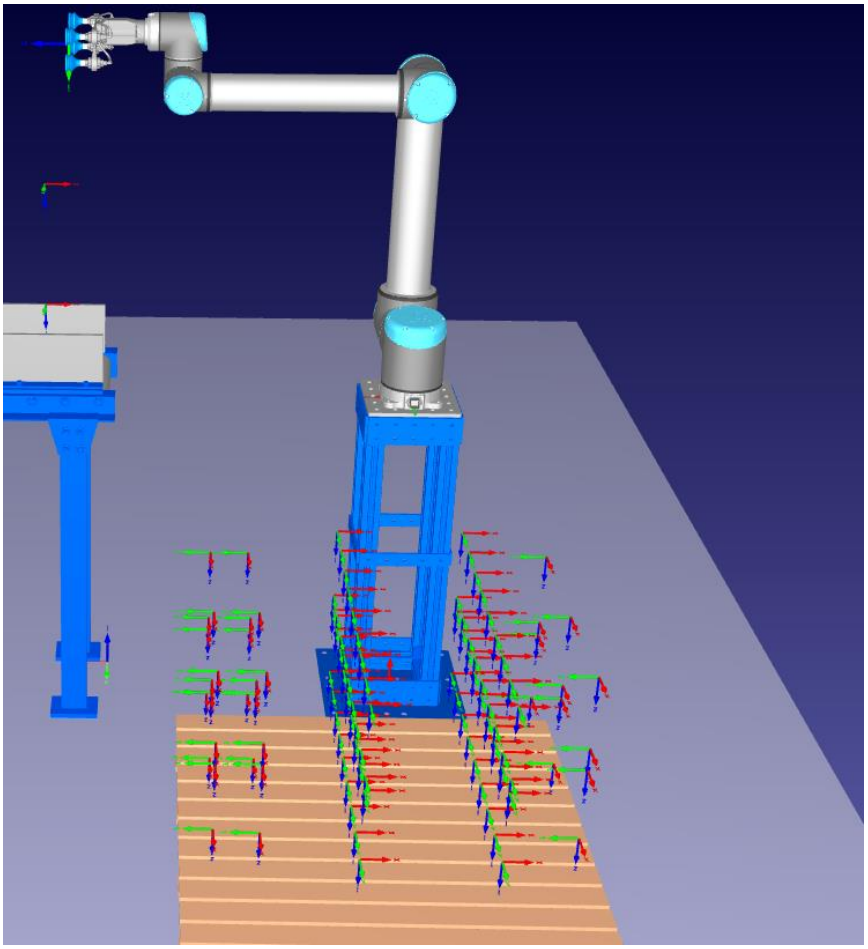


Ilustración 79. Conjunto de puntos de la estación de paletizado.

12.2 Anexo II

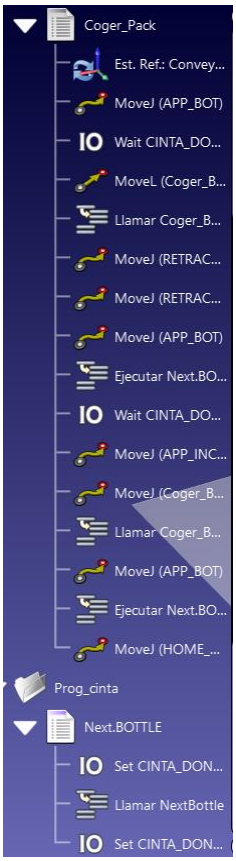


Ilustración 80. Programas estación de "Pick and Place" 2ª versión.

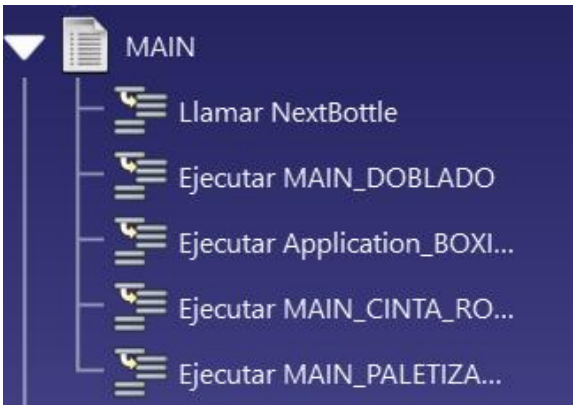


Ilustración 81. Programa principal de la estación 2ª versión.

Para introducir los cambios presentes en la segunda versión del proyecto, primero se definió una nueva variable binaria llamada "CINTA_DONE" (vale uno cuando la cinta dónde se encuentran las botellas ha terminado de hacer su recorrido, es decir, se encuentra en reposo). A continuación, se añadió un nuevo programa llamado "Next.BOTTLE" que permitiera definir cuando "CINTA_DONE" vale uno y cuando no, además, se hicieron algunas modificaciones en el programa "Coger_Pack" para que el robot UR5 esperara a que la cinta terminara su recorrido antes de ir a coger un nuevo lote de botellas.

Finalmente, se ha decidido ejecutar el fichero "NextBottle.py" al principio del programa "MAIN" para solucionar un problema que existía en el reinicio de la simulación.

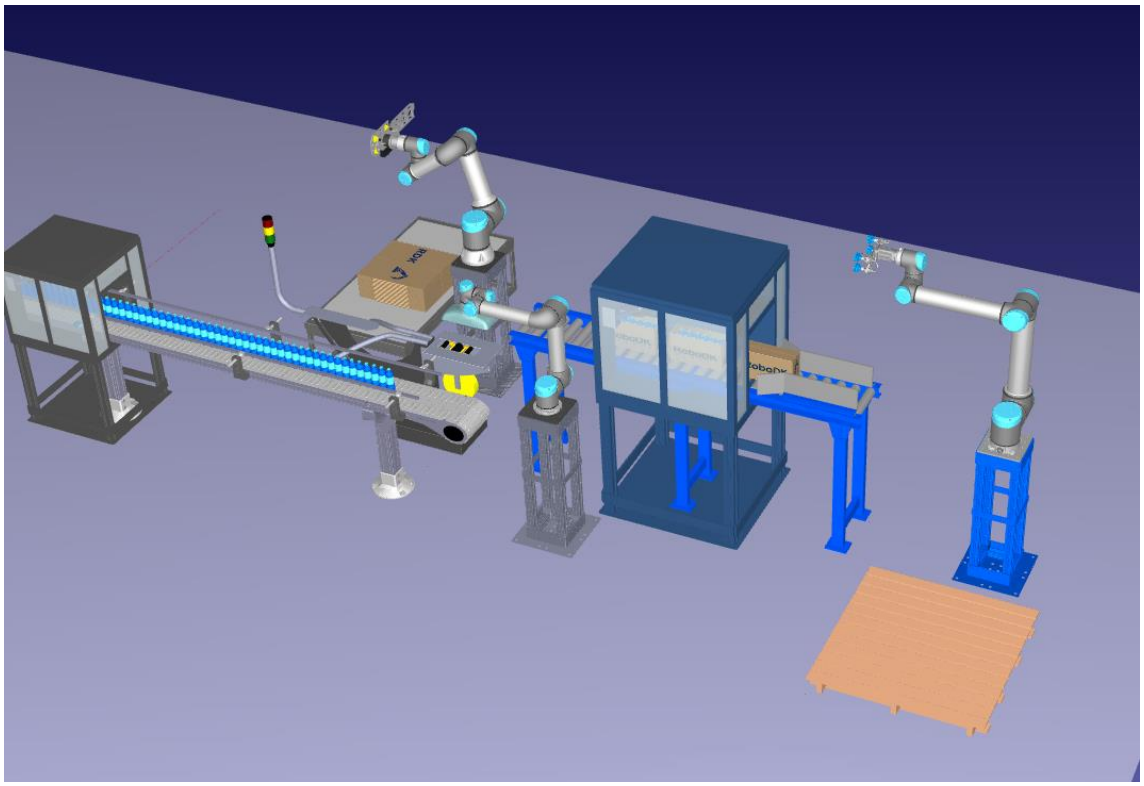


Ilustración 82. Distribución de la estación 2ª versión.

En la *ilustración 82.* se pueden observar los cambios realizados en la distribución de elementos de la estación de doblado y preparado de cajas.

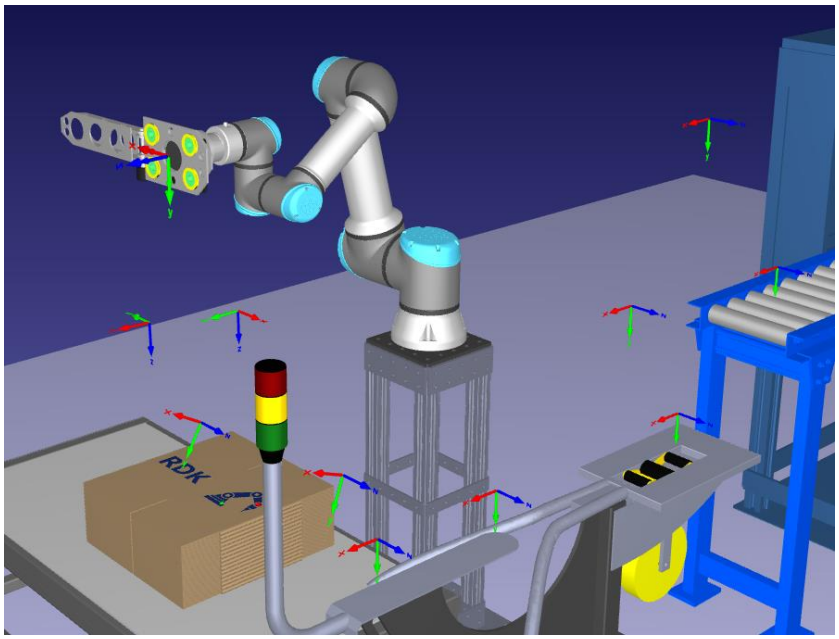


Ilustración 83. Distribución de puntos de la estación de doblado y preparado de cajas 2ª versión.

12.3 Anexo III

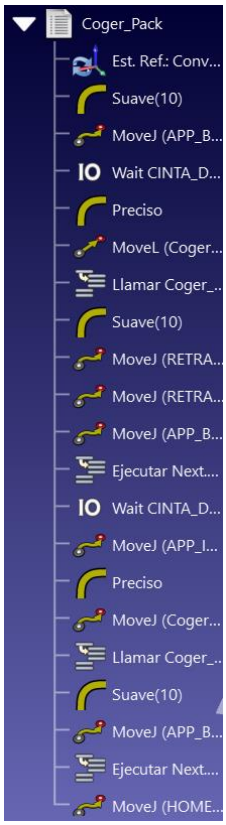


Ilustración 84. Programa estación "Pick and Place" 1, 3ª versión.

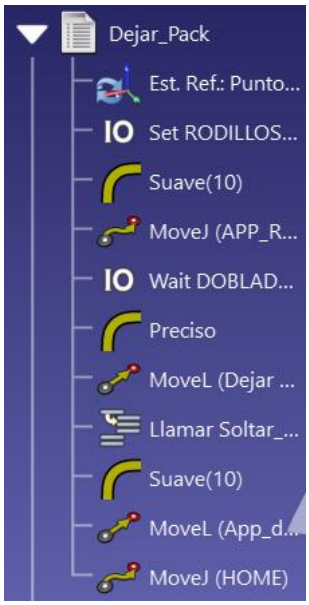


Ilustración 85. Programa estación "Pick and Place" 2, 3ª versión.

Ilustración 86. Programa estación de doblado 1, 3ª versión.

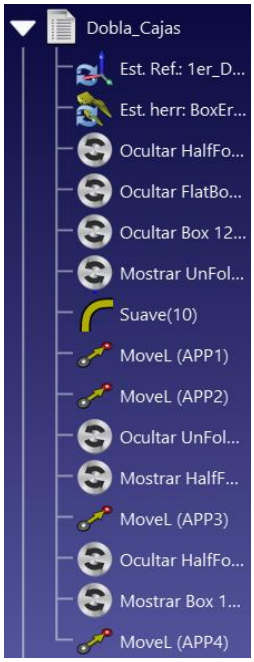


Ilustración 87. Programa estación de doblado 2, 3ª versión.

Ilustración 88. Programa estación de doblado 3, 3ª versión.

13 BIBLIOGRAFÍA

- **Importancia de la robótica en los ODS**
  Sòlene Guenat y otros. (21 de junio de 2022) *Meeting sustainable development goals via robotics and autonomous systems*. Nature. Recuperado el día 16 de mayo de 2024 de https://www.nature.com/articles/s41467-022-31150-5

- **Vídeo del proceso en Baxter Laboratories**
  andrewdonalddesign. (2017, 7 de mayo). *Case Packing and Palletising at Baxter Laboratories* [video]. YouTube. https://www.youtube.com/watch?v=pO09vPELQmM

- **Vídeos necesarios para la preparación previa**
  RoboDk. (2021, 16 de noviembre). *Intro and Installation - Module 1.01 - RoboDK Pro Training* [video]. YouTube. https://www.youtube.com/watch?v=qBSQhG8_HCQ&list=PLjiA6TvRACQexjP3pB1YOaxT-aTLG0GO2

- **Programación con Python en RoboDk**
  RoboDk API. *RoboDK API (robodk package)*. Recuperado el día 20 de abril de 2024 de https://robodk.com/doc/en/PythonAPI/robodk.html

- **Características robot UR16e**
  Universal Robots. (noviembre de 2023). *UR16e Technical Specification* [Archivo PDF]. Recuperado el 07 de junio de 2024 de https://www.universal-robots.com/media/1811483/ur16e-rgb-fact-sheet-landscape-a4.pdf

- **Características UR10**
  Universal Robots. *Technical specifications UR10* [Archivo PDF]. Recuperado el 07 de junio de 2024 de https://www.universal-robots.com/media/50880/ur10_bz.pdf

- **Características UR5**
  Universal Robots. *UR5 Technical specifications* [Archivo PDF]. Recuperado el 07 de junio de 2024 de https://www.universal-robots.com/media/50588/ur5_en.pdf

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN GEMELO DIGITAL CORRESPONDIENTE A UNA CÉLULA ROBOTIZADA PARA EL ETIQUETADO, EMPAQUETADO Y PALETIZADO DE BOTELLAS MEDIANTE ROBODK

PRESUPUESTO

ÍNDICE DEL PRESUPUESTO

1 INTRODUCCIÓN

Con este proyecto se ha conseguido generar una simulación convincente de un proceso de etiquetado, empaquetado y paletizado de botellas. Puesto que se da por hecho que la maquinaria (robots, cintas transportadoras…) y los materiales (botellas, cajas, pallets…) son aportados por un tercero, éstos no serán incluidos dentro del presupuesto. Por lo tanto, tan sólo se incluirán los costes asociados a la programación del proyecto.

2 COSTES UNITARIOS

2.1 Mano de obra

La única mano de obra necesaria para llevar a cabo el proyecto ha sido el ingeniero industrial. Suponiendo que su sueldo mensual ronda unos 4000 euros brutos y que trabaja 160 horas al mes se obtiene que su precio unitario es de 25 euros/hora.

2.2 Material

Para la realización del proyecto se ha hecho uso de un ordenador portátil de un precio de 700 euros, del uso de Microsoft Office 365 con un coste de 69 euros/año y, por supuesto, el software de RoboDk cuyo precio es de 3995 euros el primer año y, a partir del segundo año, se ha de pagar un mantenimiento anual de 1500 euros/año.

Para calcular la vida útil (anual) de los softwares empleados se ha limitado su tiempo de uso a 8 horas diarias, sin tener en cuenta los 30 días de vacaciones con los que se cuentan habitualmente.

A continuación, se presenta una tabla con el coste unitario asociado a cada material empleado:

MATERIAL	VIDA ÚTIL (h)	PRECIO (€)	COSTE UNITARIO (€/h)
Portátil	2000	700	0.35
Microsoft Office 365	2680	69	0.026
RoboDk	2680	3995	1.49

Tabla 1. Cuadro de precios unitarios del coste del material.

3 PRESUPUESTOS PARCIALES

Dentro del proyecto se han decidido definir seis unidades de obra:

- UD01. Preparación previa: dentro de esta unidad se incluye el tiempo empleado para decidir el tema del proyecto y la cantidad de horas empleadas en aprender a usar el software de RoboDk.
- UD02. Modelado y programación de la línea de etiquetado y "Pick and Place" de las botellas.
- UD03. Modelado y programación de la línea de doblado y preparado de cajas.
- UD04. Modelado y programación de la línea de sellado y paletizado de paquetes.
- UD05. Estudio del tiempo de ciclo del proceso: incluye el modelado y programación de las tres versiones del proyecto.
- UD06. Redacción del proyecto.

UD01

DESCRIPCIÓN	UNIDADES	CANTIDAD	PRECIO UNITARIO (€/h)	IMPORTE (€)
Ingeniero industrial	h	30	25	750
Ordenador portátil	h	30	0,35	10,5
RoboDk	h	30	1,49	44,7
Electricidad	h	30	0,05	1,5
			TOTAL	806,7

Tabla 2. Presupuesto preparación previa.

UD02

DESCRIPCIÓN	UNIDADES	CANTIDAD	PRECIO UNITARIO (€/h)	IMPORTE (€)
Ingeniero industrial	h	45	25	1125
Ordenador portátil	h	45	0,35	15,75
RoboDk	h	45	1,49	67,05
Electricidad	h	45	0,05	2,25
			TOTAL	1210,05

Tabla 3. Presupuesto modelado y programación línea de "Pick and Place" y etiquetado de botellas

UD03

DESCRIPCIÓN	UNIDADES	CANTIDAD	PRECIO UNITARIO (€/h)	IMPORTE (€)
Ingeniero industrial	h	55	25	1375
Ordenador portátil	h	55	0,35	19,25
RoboDk	h	55	1,49	81,95
Electricidad	h	55	0,05	2,75
			TOTAL	1478,95

Tabla 4. Presupuesto modelado y programación línea doblado y preparado de cajas.

UD04

DESCRIPCIÓN	UNIDADES	CANTIDAD	PRECIO UNITARIO (€/h)	IMPORTE (€)
Ingeniero industrial	h	50	25	1250
Ordenador portátil	h	50	0,35	17,5
RoboDk	h	50	1,49	74,5
Electricidad	h	50	0,05	2,5
			TOTAL	1344,5

Tabla 5. Presupuesto modelado y programación sellado y paletizado de paquetes.

UD05

DESCRIPCIÓN	UNIDADES	CANTIDAD	PRECIO UNITARIO (€/h)	IMPORTE (€)
Ingeniero industrial	h	20	25	500
Ordenador portátil	h	20	0,35	7
RoboDk	h	20	1,49	29,8
Electricidad	h	20	0,05	1
			TOTAL	537,8

Tabla 6. Presupuesto optimización del tiempo de ciclo.

UD06

DESCRIPCIÓN	UNIDADES	CANTIDAD	PRECIO UNITARIO (€/h)	IMPORTE (€)
Ingeniero industrial	h	60	25	1500
Microsoft Office 365	h	60	0,026	1,56
Ordenador portátil	h	60	0,35	21
RoboDk	h	60	1,49	89,4
Electricidad	h	60	0,05	3
			TOTAL	1614,96

Tabla 7. Presupuesto redacción del proyecto.

4 PRESUPUESTO TOTAL DE EJECUCIÓN

PRESUPUESTOS PARCIALES	IMPORTE (€)
UD01	806,7
UD02	1210,05
UD03	1478,95
UD04	1344,5
UD05	537,8
UD06	1614,96
TOTAL	6992,96

Tabla 8. Presupuesto total de ejecución del proyecto.

5 PRESUPUESTO TOTAL DE EJCUCIÓN POR CONTRATA

PRESUPUESTO TOTAL DE EJECUCIÓN (€)	6.992,96
Gastos generales (13%)	909,08
Beneficio industrial (6%)	419,58
PRESUPUESTO TOTAL DE EJECUCIÓN POR CONTARTA (€)	8.321,62

Tabla 9. Presupuesto total de ejecución por contrata del proyecto.

6 PRESUPUESTO BASE DE LICITACIÓN

PRESUPUESTO TOTAL DE EJECUCIÓN POR CONTARTA (€)	8.321,62
IVA (21%)	1.747,54
PRESUPUESTO BASE DE LICITACIÓN	10.069,16

Tabla 10. Presupuesto base de licitación del proyecto.

Por tanto, el presupuesto base de licitación del proyecto es de **diez mil sesenta y nueve euros con dieciséis céntimos.**

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN GEMELO DIGITAL CORRESPONDIENTE A UNA CÉLULA ROBOTIZADA PARA EL ETIQUETADO, EMPAQUETADO Y PALETIZADO DE BOTELLAS MEDIANTE ROBODK

PLIEGO DE CONDICIONES

ÍNDICE DEL PLIEGO DE CONDICIONES

1 OBJETO DEL PLIEGO

Este documento, tiene como objetivo definir las diferentes condiciones técnicas, facultativas, económicas y legales que influyen en el proyecto. En este pliego de condiciones se tratará de desarrollar un pliego de condiciones generales, en el que se describirán normas de carácter general y de obligado cumplimiento, y un pliego de condiciones particulares, relacionado con una serie de características específicas del proyecto.

2 PLIEGO DE CONDICIONES GENERALES

2.1 Pliego de condiciones facultativas

Dentro de este documento es necesario conocer dos aspectos fundamentales:

- Las leyes.
- La totalidad del proyecto.

2.1.1 Programación de trabajos

El autor de este proyecto, encargado de la programación, se compromete a realizar el proyecto según lo expuesto en esta memoria, entregándolo dentro de los plazos acordados y tras una revisión exhaustiva del mismo.

2.1.2 Corrección de errores y modificaciones

Si, a la hora de valorar el trabajo se apreciara un error o modificación necesaria, es responsabilidad del cliente el comunicar lo que, bajo su punto de vista, es necesario mejorar. Si, en un primer momento, no se detectara ningún error y luego los hubiera, es también responsabilidad del cliente no haberse percatado con anterioridad de la existencia de estos.

Si, al modificar la programación del proyecto, se genera un error. La responsabilidad recae sobre la persona que ha provocado dicho error y no sobre el proyectista.

En caso de que la anomalía detectada se deba a un defecto en la programación, el programador se compromete a subsanar el error detectado, sin coste alguno para el cliente.

2.1.3 Plazo de ejecución del proyecto

El plazo de ejecución del proyecto será de un año hábil desde el momento de inicio de las tareas de programación.

2.1.4 Recepción definitiva

Una vez terminado el plazo de ejecución del proyecto, se procederá con la recepción definitiva del proyecto.

2.1.5 Liquidación final

Una vez realizada la recepción definitiva, se procederá con la liquidación fijada en el presupuesto del proyecto.

2.2 Pliego de condiciones económicas

A continuación, en los próximos apartados, se procederá con la descripción de las relaciones económicas que se establecen entre el cliente y el proyectista.

2.2.1 Base fundamental

Como base fundamental de las condiciones económicas de este proyecto, se establece que el proyectista debe percibir el importe correspondiente a todos los trabajos realizados.

2.2.2 Garantías

El ingeniero encargado del proyecto podrá exigir al comprador la presentación de avales bancarios u otras modalidades de garantía, con la finalidad de garantizar que el comprador reúne todas las condiciones necesarias para el total cumplimientos del pago del proyecto.

2.2.3 Fianza

Se podrá exigir al cliente, en el momento de la adjudicación del proyecto y antes de comenzar las tareas de programación, una fianza del 10% del total presupuesto del proyecto.

2.2.4 Fijación de precios

Los precios fijados en este proyecto son coherentes con los precios establecidos por el mercado. Asimismo, las tarifas de la mano de obra empleada se corresponden a las vigentes para un Ingeniero Industrial llevando a cabo tareas de programación.

2.2.5 Revisión de precios

Puesto que el periodo de tiempo transcurrido entre la ejecución y la entrega del proyecto puede ser largo, se admite la revisión de los precios contratados, bien en alza o en baja, en respuesta a las oscilaciones del precio de mercado.

3 PLIEGO DE CONDICIONES PARTICULARES

3.1 Características mínimas exigidas (Hardware)

- **Memoria:** 2 GB como mínimo. Recomendado 4 GB o más.

- **Configuración gráfica:** pantalla con una resolución mínima de 1024x768 píxeles. El controlador gráfico debe ser compatible con OpenGL 3.0 o posterior. Se recomiendan tarjetas gráficas con OpenGL acelerado por hardware que incluyan memoria interna.

- **Espacio en disco duro**: 15 GB, 1 GB libre.

- **Ratón:** no es imprescindible pero sí muy recomendable. Se recomienda 2 botones (mínimo), 3 botones o 2 botones con la rueda central del ratón.

- **Conexión a red:** necesario para activar las licencias de red (licencias por defecto). Los "firewalls" no deben bloquear los puertos 80 y 443 (sólo licencias de red). Como alternativa, se puede solicitar una licencia de dongle USB.

3.2 Características mínimas exigidas (Software)

- **Sistema operativo**: Windows Vista, Windows 7, Windows 8, Windows 10 o Windows 11 (versión de 32 o 64 bits), Mac OS (64 bits), Ubuntu 16 o Ubuntu 18 (64 bits).

- **RoboDK.**

- **Python:** aunque viene instalada una versión de Python "por defecto" en RoboDk se recomienda tener instalado Python (versión 3.12.4) en el ordenador para facilitar la manipulación del código.

3.3 Instalación del programa

El programa será instalado y puesto a punto bajo la supervisión del programador del proyecto ya que el cliente no tiene por qué conocer el procedimiento requerido.

3.4 Adjudicación del proyecto

La adjudicación del proyecto no da derecho a la adquisición de una licencia de RoboDk. Si se quiere valorar el trabajo realizado por el programador se deberá hacer a través del uso de una licencia gratuita o procediendo a la compra de una nueva licencia, dicha compra correrá a cargo del cliente.

3.5 Condiciones de mantenimiento

En principio el software informático empleado en este proyecto no necesita ser revisado por ningún técnico. En caso de que la configuración del software o del hardware del equipo sufriera algún tipo de cambio, contribuyendo así, a la modificación de alguna característica o ubicación del programa, dicha revisión, sí sería necesaria.

Si se acaba verificando que el programa ha sido modificado voluntariamente y, dicha modificación, ha originado un error inicialmente no existente, el programador queda exento de gestionar la reparación del producto.

3.6 Condiciones de garantía

En caso de pérdida o eliminación por error de algún fichero o programa necesario para el correcto funcionamiento del producto, el programador se compromete a facilitar al cliente una copia del proyecto en un período de tiempo no superior a una semana. Esta garantía pierde su efecto al paso de los cuatro años desde la recepción definitiva del proyecto. Por tanto, si transcurrido este periodo se requiriera de una nueva copia, se deberá pactar con el programador, un nuevo precio por la copia del producto.

Se considera necesario remarcar que cualquier cambio o modificación del programa, así como su transferencia, o instalación no autorizada supondrá la pérdida total de las garantías expuestas anteriormente.