UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de ingeniería del Diseño

PROGRAMACIÓN DE UNA TAREA ROBOTIZADA MEDIANTE SIMULACIÓN EMPLEANDO DIFERENTES ENTORNOS DE DESARROLLO Y COMPARATIVA

TRABAJO FINAL DEL

Grado en Ingeniería Eléctrica

REALIZADO POR

Víctor Estruch Martínez

TUTORIZADO POR

Luis Ignacio Gracia Calandin

CURSO ACADÉMICO: 2019/2020

RESUMEN

En el presente Trabajo Fin de Grado se aborda la comparativa entre diferentes entornos de programación mediante el mecanizado de la misma pieza. Este mecanizado va a constar de etapas de fresado y etapas de perforado por igual en ambos entornos de programación.

Para la realización de esta comparativa, se han introducido paso a paso durante la implementación de la tarea ambos programas. Con esto se ha conseguido destacar las ventajas y desventajas de ambos entornos para la ejecución del mismo trabajo.

En este proyecto se ha tratado la programación de un robot industrial, el IRB-140 de ABB y un robot colaborativo, el UR5 de Universal Robots. Durante el proceso se han detectado aspectos diferenciativos en ambas ramas que se han comparado de la misma manera que los entornos de trabajo.

Para la realización de este proyecto se han escogido los entornos de programación de RobotStudio de la marca ABB y de PolyScope de la marca Universal Robots.

Palabras clave TFG: programación, robot industrial, robot colaborativo, modelado, simulación.

ABSTRACT

In the present End of Degree Project, is shown the comparison between different programming environments is approached by machining the same part. This machining will consist of milling stages and drilling stages equally in both programming environments.

To carry out this comparison, both programs have been introduced step by step during the implementation of the task. With this it has been possible to highlight the advantages and disadvantages of both environments for the execution of the same job.

In this project, the programming of an industrial robot, ABB's IRB-140 and a collaborative robot, Universal Robots' UR5, has been dealt with. During the process, some differentiating aspects have been detected in both branches that have been compared in the same way as the working environments.

For this project, the programming environments of ABB's RobotStudio and Universal Robots' PolyScope have been chosen.

Keywords TFG: programming, industrial robot, collaborative robot, modeling, simulation

DOCUMENTOS DEL TFG

- **DOCUMENTO I MEMORIA DESCRIPTIVA**
- **DOCUMENTO II PLANOS**
- **DOCUMENTO III PLIEGO DE CONDICIONES**
- **DOCUMENTO IV PRESUPUESTO**

ÍNDICE DE LA MEMORIA DESCRIPTIVA

ÍNDICE DE PLANOS

ÍNDICE DE PLIEGO DE CONDICIONES

ÍNDICE DE PRESUPUESTO

ÍNDICE DE FIGURAS

DOCUMENTO I

MEMORIA DESCRIPTIVA

1. Objeto

En el presente Trabajo Fin de Grado se va a realizar el estudio y la comparativa de dos entornos de desarrollo para una misma tarea robotizada con el fin de analizar las ventajas y desventajas de cada uno.

En este documento se van a introducir ambos entornos de desarrollo explicando cada uno de los robots que se van a emplear (El IRB-140 de ABB y el UR5 de Universal Robots).

El objetivo de este trabajo es hacer una clara comparativa entre los entornos de RobotStudio de ABB y PolyScope de Universal Robots. Para saber en qué situaciones nos convendría más el uso de un robot industrial como es el IRB-140 o el uso de un robot colaborativo como es el UR5. Ya que los procesos de mecanizado se dan en todo tipo de industrias, pequeñas, medianas y grandes.

Este trabajo también sirve como una guía práctica para ambos entornos ya que se va a documentar paso a paso cada acción de los programas.

La tarea para realizar está compuesta de varios procesos de mecanizado comunes en la industria. Estos procesos son de fresado y perforado. Los vamos a realizar con una misma herramienta la cual lleva dos útiles en cada extremo para realizar cada una de las tareas.

Estos robots constan de 6 grados de libertad gracias a los cuales son capaces de usar y aprovechar ambos útiles al igual que para desplazarse correctamente por los recorridos demandados en la pieza.

Finalmente, otro de los objetivos de este trabajo va a ser comprobar las aptitudes de los entornos y robots para las siguientes características:

- Facilidad y comodidad en el entorno de programación para realizar programas de mecanizado.
- Seguridad a la hora de trabajar con el robot.
- Flexibilidad en su manejo y accesibilidad para cualquier operario en caso de tener que modificar cualquier dato o parámetro en la línea de producción.

2. Antecedentes

El concepto de la robótica tiene su origen desde la antigüedad. Que es cuando se da para el entretenimiento ya que eran máquinas con apariencias humanas o animales que estaban accionadas por mecanismos simples. Con el paso de los siglos se siguen desarrollando estos autómatas y además empiezan a surgir otros que ayudan en tareas simples y repetitivas.

Es en el siglo XX es cuando su desarrollo comienza a tomar importancia. A principios de siglo se usa por primera vez el término de robot en una obra de teatro. La definición que tenía este término era la de servidor o trabajador forzado. Es entonces cuando se empieza a hablar de robots y empiezan a investigar más al respecto.

En 1954 se construye un brazo articulado fabricado para realizar movimientos programables este brazo es considerado el primer robot industrial. Durante los años posteriores se estudian, desarrollan e innovan otros robots de uso industrial o de servicio. Estos robots facilitan la vida en la industria, medicina, agricultura, en el hogar y en un amplio recorrido de sectores. La evolución de estos robots está marcada por distintas generaciones:

- Primera generación: Robots que repiten una misma tarea programada. Carece de sensores, no toma información del exterior.
- Segunda generación: En esta generación los robots ya llevan sensores para adquirir información del entorno. Los movimientos de esta generación son más complejos que la anterior. Su uso principal está enfocado en la industria automotriz
- Tercera generación: Se programan mediante lenguaje natural. Son reprogramables y tienen mayor percepción del entorno. Pueden realizar tareas automáticamente. Son el inicio de la inteligencia artificial.
- Cuarta generación: Dentro de esta generación tenemos a los robots inteligentes, similares a los anteriores. Tienen sensores que envían constantemente información al ordenador de control y esta toma decisiones inteligentes.
- Quinta generación: Está considerada como el futuro de los robots relacionado con la inteligencia artificial. Serán autónomos y podrán satisfacer las necesidades de los humanos. Los nuevos robots serán de percibir y razonar en entornos dinámicos. Podrán simular funciones de la mente humana.

3. Introducción a la robótica

La robótica es una ciencia que acumula varias ramas tecnológicas. Entre las que se encuentran la eléctrica, mecánica, electrónica, Informática… y que se ocupa de aplicaciones como el diseño, manufactura, medicina, transporte, uso doméstico y de las que se van a tratar en este trabajo la industrial y la colaborativa.

3.1 Robótica industrial

Los robots industriales son los más utilizados de todos los tipos de robots. Nacen de la alta exigencia y demanda de productividad en la industria. Son por lo general brazos robots grandes articulados destinados a su uso en la industria. Estos robots tienen como misión realizar una tarea repetitiva para automatizar un proceso. La definición de estos robots según nos marca la norma **ISO 8373:2012** es:

''Manipulador multifuncional, controlado automáticamente, reprogramable en tres o más ejes, que puede estar fijo o móvil para uso en aplicaciones de automatización industrial.''

Figura 1 - Robot Industrial

Los fabricantes más importantes de estos robots industriales en la actualidad son:

- ABB Motors
- Yaskawa
- EPSON Robots
- Kawasaki Heavy industries
- KUKA Robotics

Los robots más utilizados dentro de la robótica industrial son los brazos articulados como se observa en la figura 1.1. Estos robots tienen un gran abanico de utilidades como son:

- Soldadura
- Pintura
- Ensamblado
- Mecanizado (Perforado, fresado…)

- Pick and Place

Dentro del grupo de los robots industriales también nos encontramos los robots colaborativos de los cuales se hablará en el siguiente punto y los vehículos de guiado automático o AGVS estas dos modalidades de robot están tomando gran importancia últimamente en la industria debido a su facilidad en la cooperación con operarios u otros robots.

Estos brazos robóticos por lo general tienen de 3 a 7 grados de libertad y simulan un cuerpo humano de la cintura a la muñeca formado por una serie de eslabones y articulaciones que proporcionan una gran libertad de movimiento.

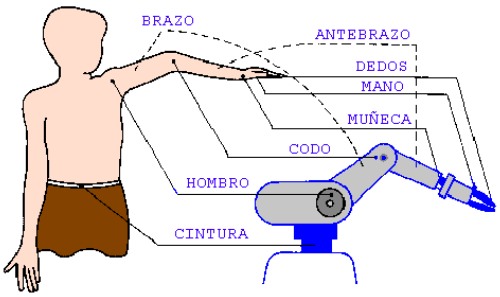


Figura 2- Brazo robótico

El problema principal de los robots industriales tradicionales es que necesitan un área de trabajo cercada y marcada para no colisionar con objetos o personas. Estas áreas de trabajo pueden estar formadas por uno o más robots encargados con una tarea programada fija y repetitiva. Estos robots carecen de sensores de fuerza o visión para detectar a los operarios por ello es necesario que la zona esté acotada.

3.2 Robótica colaborativa

La robótica colaborativa o también conocida como cobots, está diseñada para colaborar e interactuar con los trabajadores humanos. Estos robots pueden trabajar sin problema con operarios en un espacio común, sin necesidad de un espacio de seguridad como en los robots industriales tradicionales.

Existen diferentes tipos de colaboraciones entre estos robots y humanos, las cuales son:

- Cooperación: El robot y el humano trabajan en la misma zona de trabajo, pero no a la vez, es decir, el humano no interactúa con el robot mientras está realizando la operación.

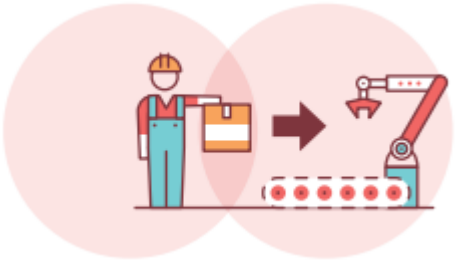


Figura 3 - Cooperación

- Coexistencia: El operario y el robot trabajan a la vez, pero están situados en espacios diferentes dentro de la zona de trabajo y no tienen contacto directo entre ellos.

Figura 4 - Coexistencia

- Colaboración directa: Trabajan ambos en la misma zona de trabajo y a la vez sobre la misma pieza.

Figura 5 - Colaboración

Se ha investigado mucho sobre estos robots ya que su uso es muy interesante en todo tipo de industria, debido a que no se necesita un perímetro especial para ellos y pueden estar junto a operarios. Por lo que la pequeña industria se puede beneficiar de ello. También hay que saber que son muy flexibles en su programación, es decir se puede cambiar con facilidad de un programa a otro y para realizar estos cambios no se requiere de un técnico cualificado a diferencia de los robots tradicionales. Esto los hace idóneos para trabajar en cualquier tipo de tarea.

Otras ventajas por la que se caracterizan los cobots son:

- Pueden trabajar 24h al día, es decir, pueden seguir funcionando a falta de operario debido a su seguridad.

- Debido a que son robots colaborativos y ayudan a los trabajadores, evitan lesiones de sobreesfuerzo o de esfuerzo repetitivo, además pueden estar expuestos a gases nocivos, al contrario que los operarios.
- Por último, destaca como ventaja que al trabajar con los operarios se disminuye el tiempo de hacer productos. Por lo que se pueden producir con mayor velocidad y generar mayores ganancias.

En este proyecto se va a estudiar la robótica colaborativa con los robots de Universal Robots una de las compañías más importantes en el sector de la robótica colaborativa.

Figura 6 Robots Colaborativos Universal Robots

Los fabricantes más importantes de robots colaborativos son:

- Universal Robots
- KUKA
- Bosch Rexroth
- FANUC
- ABB
- Rethink Robotics

Por último, se debe saber que dependiendo de la la aplicación a la que esté destinada el robot colaborativo puede llegar a ser peligroso. Como por ejemplo si transporta una carga pesada y afilada. Aunque el robot al detectar la presencia del humano reduzca su velocidad la carga que lleva puede llegar a ser peligrosa.

Como se ha visto existe una lista de diferencias entre los robots industriales tradicionales y los colaborativos. La finalidad de este trabajo es enumerar las diferencias que tienen tanto los robots como la programación de estos. Además, enumerar las ventajas y desventajas de cada uno de ellos.

4. Implementación en el RobotStudio

4.1 Introducción a RobotStudio

El programa Robotstudio es un entorno de desarrollo basado en Windows que está destinado a la programación fuera de línea, simulación y control de robots de la marca ABB. Esta marca de robots es una de la más importantes a nivel mundial, no solo de robots industriales, sino también de robots colaborativos. En este trabajo se utilizará la herramienta de RobotStudio para realizar el mecanizado con el robot seleccionado IRB-140 de la marca ABB.

Robotstudio es un programa de simulación potente, que presenta diversas características:

- Nos da la opción de elegir la estación con el robot ABB que nosotros deseemos.
- Permite la importación de objetos 3D de cualquier formato y la propia creación de estos objetos dentro del programa.
- Ofrece facilidades para la creación de células robóticas (Robot con su entorno de trabajo)
- Nos permite programar sobre el robot o bien sobre lenguaje RAPID. Una vez programado nos da la opción de simularlo.
- Facilita la opción de exportar las simulaciones obtenidas dentro de la herramienta a la estación del robot real.

Esta herramienta nos presenta una simulación muy realista debido a que le podemos añadir todos los elementos que queramos a la célula de trabajo para hacerla lo más similar posible a la estación real. Este es un punto muy a favor a destacar.

4.2 Introducción a IRB-140

El robot IRB-140 va a ser con el que estudiemos el RobotStudio en este proyecto. Este es un robot compacto y potente. Consta de 6 ejes y puede soportar una carga de hasta 6kg y su alcance es de 810 mm. El fabricante nos indica que tiene posibilidad de instalación en el suelo, en la pared desde cualquier ángulo o de manera invertida.

Este robot presenta una protección IP67. El significado de este número es su grado de protección, el 6 significa que tiene una protección total contra el polvo y el 7 que puede ser sumergido hasta máximo un metro de agua. Por lo que el robot tiene un sistema estanco al polvo y agua.

Es un robot flexible, con cables integrados. Además, cabe destacar que, aunque sea un robot industrial presenta una función de detección de colisiones con retirada completa esto es un punto a favor de estos robots. Ya que es un robot costoso y así evita que se rompa en caso de fallo en la programación o colisión.

Figura 7 - Robot IRB-140

4.3 Puesta a punto del IRB-140 en RobotStudio

Para empezar a trabajar con esta herramienta primero deberemos instalarla correctamente en el ordenador y solicitar una licencia. Una vez se tiene todo esto podemos comenzar con el programa. Al iniciarlo se nos abre una ventana primero se hace clic en Solución con estación vacía, le ponemos nombre al programa que queremos realizar, lo ubicamos en una carpeta y clic en crear.

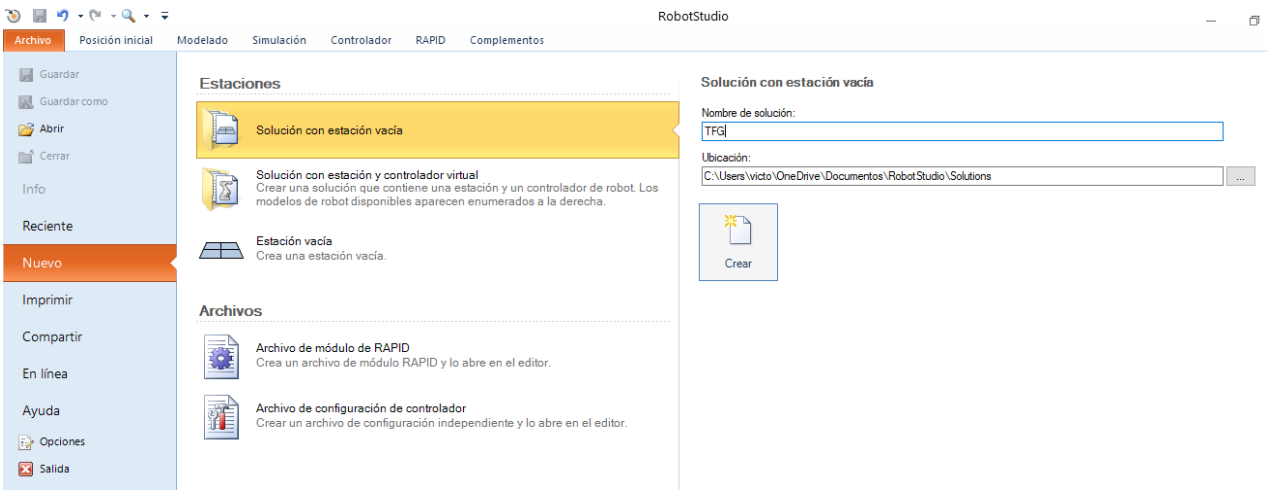


Figura 8 - Creación de estación en RobotStudio

Con la estación vacía podemos ver todas las opciones que nos da el programa como son:

- Posición inicial: Es la primera opción de RobotStudio y es una de las que más se va a usar en este trabajo, por lo que se va a ir explicando con la realización de la programación.

- Modelado: Con el que se va a crear la pieza a mecanizar. Se puede crear cualquier tipo de geometría con las opciones que da el programa. También se pueden importar piezas de otros programas de CAD.
- Simulación: Donde podemos ver la simulación de la programación realizada. También da la opción de seguir el recorrido de la herramienta dejando la trayectoria realizada con el color que deseemos. Por último, también da la opción de grabar la simulación realizada en video.
- Controlador: En esta opción del programa podemos añadir el controlador que deseemos para accionar el robot.
- RAPID: Este es el apartado de programación del robot. Es donde se realiza el código a simular.
- Complementos: En los complementos podemos encontrar los controladores a instalar para que funcione el Robot o bien partes de la estación como cintas transportadoras o herramientas creadas por otros usuarios.

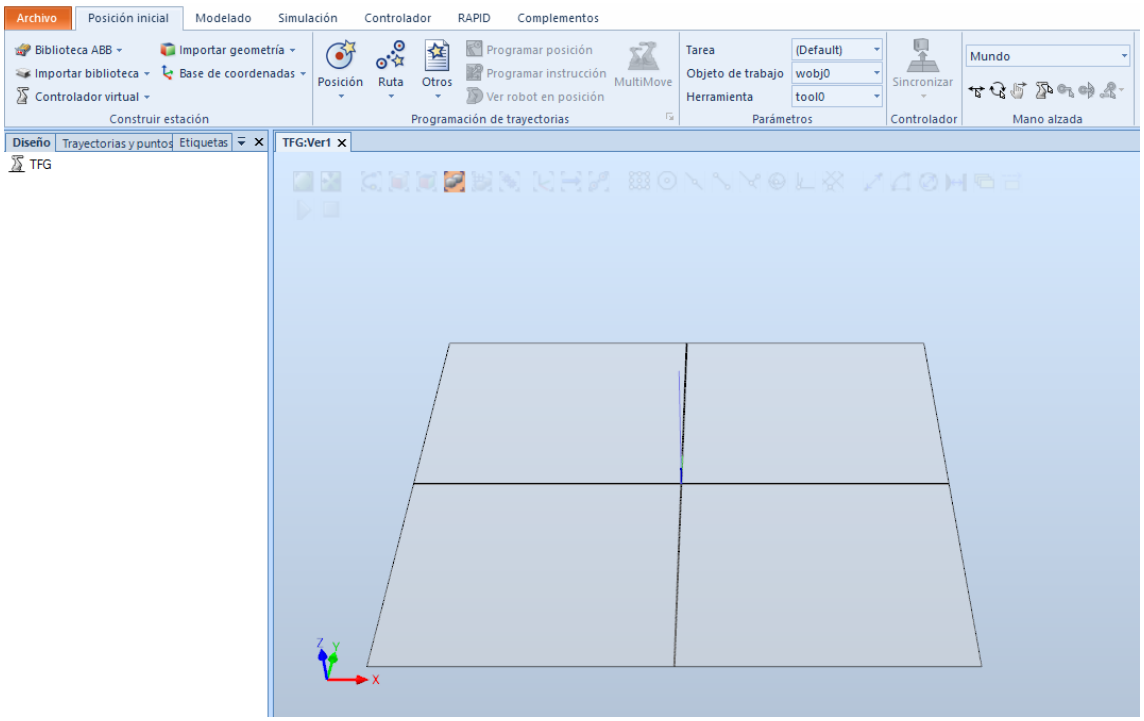


Figura 9 - Opciones principales RobotStudio

En la anterior imagen podemos ver como la estación está vacía, por lo que el siguiente paso en este programa es añadir el robot. En mi caso en este trabajo y como se ha comentado voy a usar el IRB-140. Para esto hay que darle clic a Biblioteca ABB y seleccionar el robot. Una vez seleccionado este se pondrá en el eje de coordenadas de la estación que por defecto es en el medio.

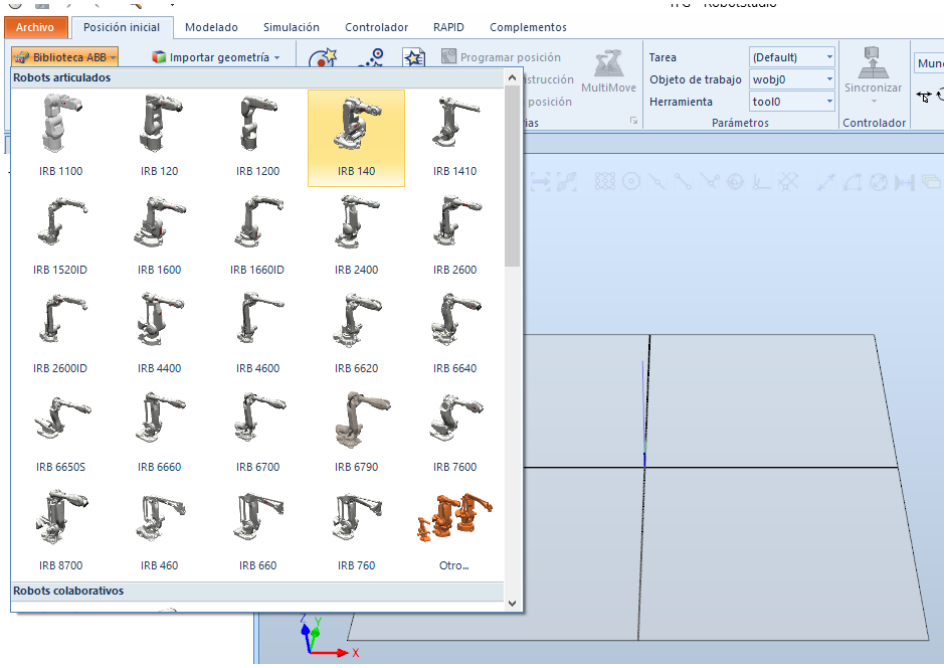


Figura 10 - Selección del robot

El siguiente paso para poder programar con el robot va a ser añadir el controlador. Es el que se va a encargar de dejar operativo al robot para poder moverlo a deseo y poder realizar la programación. El controlador por instalar se encuentra en el apartado de complementos de RobotStudio. En este apartado hay que buscar ‘’RobotWare for IRC5’’ y descargar la última versión.

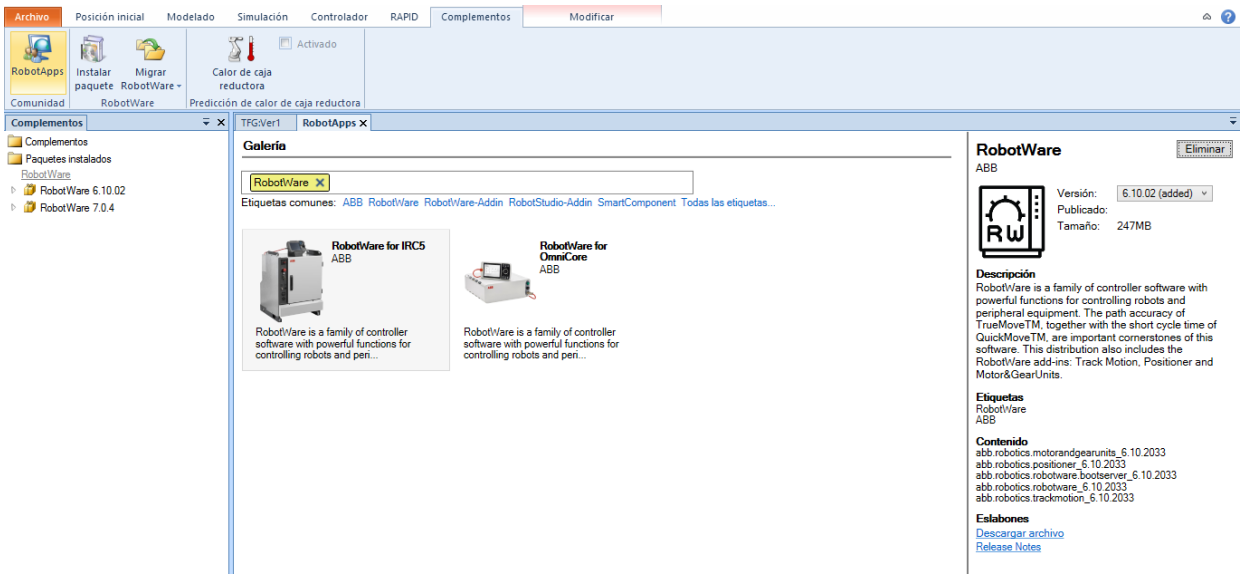


Figura 11 - Instalación del controlador

Por último y para dejar el robot preparado para su programación falta la herramienta con la que se va a realizar el mecanizado. Esta herramienta se puede diseñar con cualquier programa de CAD. En el caso de este trabajo se ha cogido una herramienta ya diseñada, que consta de dos útiles. El primer útil es una fresadora y el segundo una perforadora, con estos vamos a realizar posteriormente el mecanizado.

Para añadir una herramienta o pieza al programa se hace desde la opción de posición inicial. Hay que darle clic a importar biblioteca, buscar la carpeta donde esté situada la pieza a añadir y darle clic a la pieza. Automáticamente esta se añade y se coloca en el centro de coordenadas de la estación.

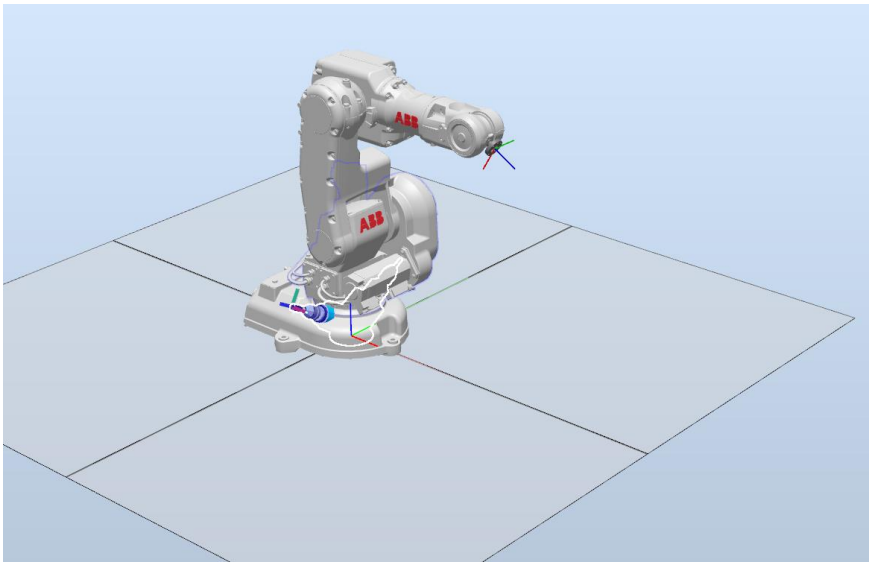


Figura 12 - Añadir herramienta a la estación

Ya estaría añadida la herramienta a la estación, ahora solo queda colocarla al robot. Para ello en el menú de la izquierda de RobotStudio encontramos unas opciones: Diseño, Trayectorias y puntos y Etiquetas. En la de diseño tenemos el robot y la herramienta, para colocar la herramienta en el robot hay que arrastrar la herramienta en este caso (MyNew Tool) al robot (IRB 140_6_81_C_03).

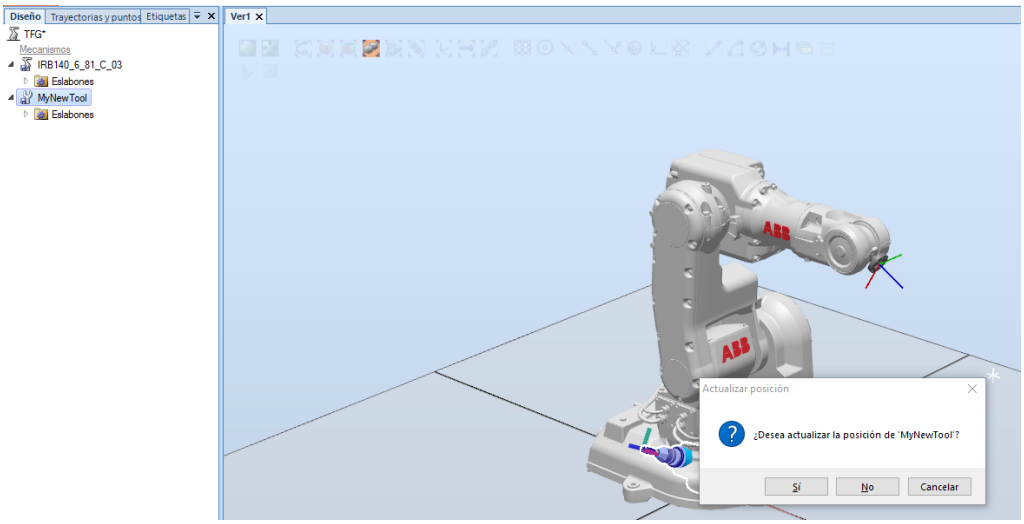


Figura 13 - Ubicación de la herramienta en el robot

Al hacer esto sale un mensaje que te pregunta si se desea actualizar la posición de la herramienta, se le da a Sí y aparece directamente bien colocada la herramienta en el robot.

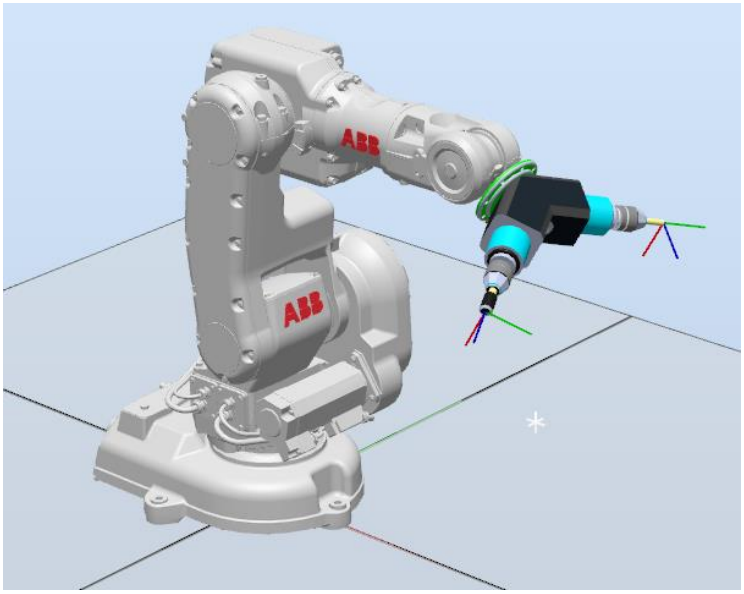


Figura 14 - Herramienta colocada en el robot

Con estos pasos ya está el robot preparado para realizar el mecanizado. Ahora hay que realizar el entorno, en este robot se va a realizar una mesa de trabajo y el objeto a mecanizar. Esto se va a realizar con la opción de modelado.

4.4 Modelado del entorno

Dentro de RobotStudio se encuentra la pestaña de modelado, esta ofrece una gran cantidad de herramientas para crear la célula de trabajo, de las cuales se han utilizado las siguientes:

- Crear un grupo de componentes para la estación, piezas vacías y componentes inteligentes. Estos últimos tienen funciones especiales como por ejemplo una cinta transportadora con movimiento propio, útil para las simulaciones de pick and place.
- Importar geometrías, esta opción es la misma que da estando en la pestaña de posición inicial. Permite importar piezas ya creadas de RobotStudio o importar piezas que tengamos descargadas. Estas piezas también pueden haber sido creadas en un programa de CAD.
- Sólidos, con esta opción se pueden crear piezas geométricas en 3D como son: tetraedro, cono, cilindro, pirámide y esfera. Con esta opción es con la que se va a realizar la mesa de trabajo para el robot y la pieza a mecanizar.
- Superficie, esta opción es similar a la anterior solo que ahora las piezas son planas en 2D.
- Curva, con esta opción podemos crear piezas geométricas sin relleno.
- Ofrece también la opción de sumar o restar las anteriores piezas geométricas entre sí para formar piezas. En este trabajo se mostrará paso a paso la realización de la pieza a mecanizar, por lo que se le explicará el uso de esta función y las anteriores.
- Por último, de este apartado de modelado también se usará las herramientas de medición tanto punto a punto como de diámetro para circunferencias.

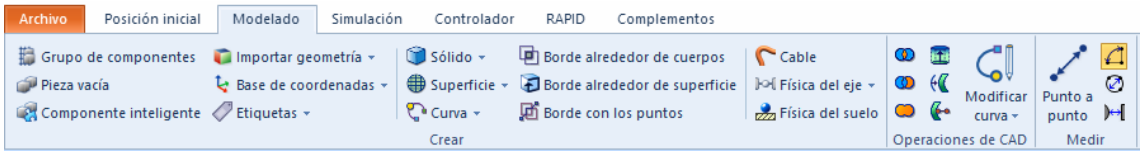


Figura 15 - Herramientas del Modelado

Tras esta introducción a la pestaña de modelado se va a pasar a explicar cómo se ha creado la pieza a mecanizar por el robot y la mesa de trabajo.

Como paso principal y para crear con mayor facilidad los elementos del entorno se va a guardar la anterior estación ya preparada y a abrir una nueva vacía, para no tener ningún elemento que moleste.

El mecanizado se va a realizar sobre una pieza tetraédrica. Este va a constar de operaciones de rebajado con fresadora y de tres operaciones de perforado. El primer paso para realizar la pieza es crear la base. Para ello se selecciona la opción de sólido, se da clic en tetraedro y por defecto se ubicará en el centro de coordenadas que es justo lo que se desea para que se facilite la realización de la pieza. Las medidas de la pieza van a ser de 100 mm de longitud, 225 mm de ancho y 75 mm de alto. El programa crea una vista previa de la pieza con las dimensiones que se quiere realizar como se aprecia en la siguiente imagen. Por último, se le da clic a la opción crear y se crea la pieza.

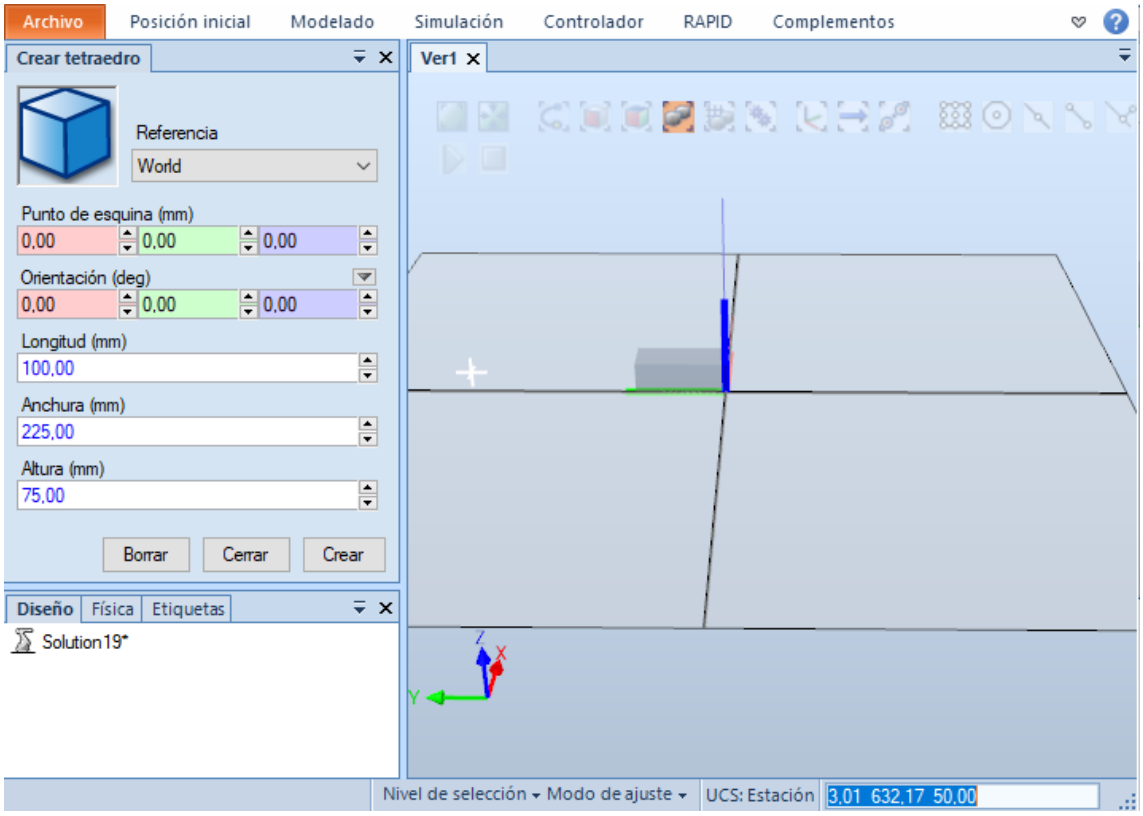


Figura 16 – Vista previa del tetraedro

Esta es la pieza base del mecanizado, para la realización de la simulación hay que dejar la pieza con la forma final. Para ello hay que crear nuevas piezas con el tamaño que se le desea retirar a la pieza principal. Se crea un tetraedro nuevo con las dimensiones del primer fresado que son 100 mm de longitud, 25 mm de anchura y 10 mm de altura. Esta pieza hay que ubicarla en el punto de coordenadas exacto para restarla posteriormente a la principal este punto es el 0,0,65. Una vez creada la pieza se ve superpuesta a la principal como se muestra en la siguiente imagen.

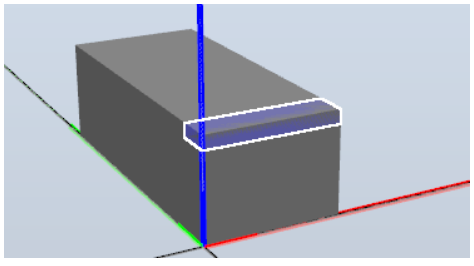


Figura 17 - Superposición de piezas

El siguiente paso es restarle a la pieza base la pequeña. Con esto se consigue dejar el hueco para que pase la herramienta del robot y simule la operación de fresado. Para realizar este paso hay que darle clic en modelado a la opción de restar. Esta operación

nos deja dos huecos uno para la pieza a la que se le quiere restar y otro para la pieza que se resta. Por lo que en el primero se clica en la pieza base y en el segundo hueco en la pieza pequeña y se le da a crear.

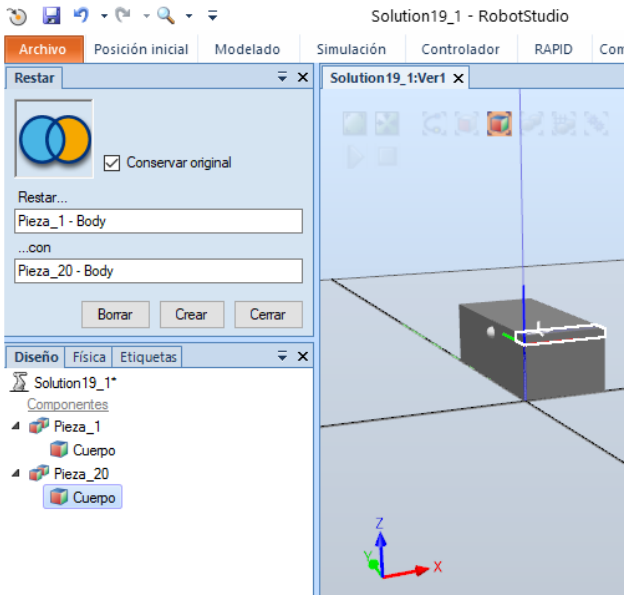


Figura 18 - Operación de restado

El resultado es que se ha creado una nueva pieza con el resultado que se desea. Para verla solo hay que poner en invisible las dos piezas anteriores o borrarlas directamente.

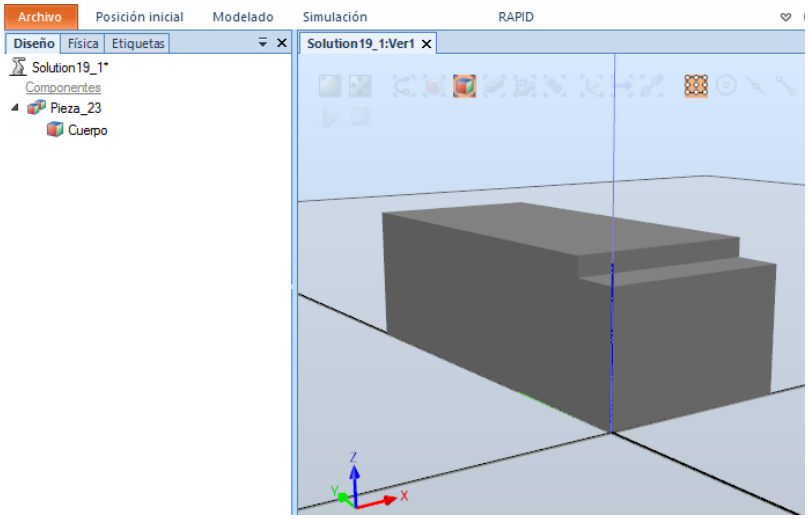


Figura 19 - Pieza tras el restado

Para el rebajado de la pieza mediante fresado se ha seguido usando esta técnica de restar piezas a la principal.

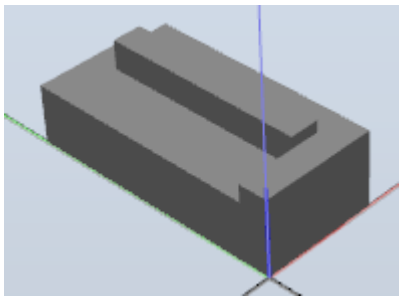


Figura 20 - Pieza después del rebajado

Este es el resultado final del fresado, ahora hay que realizar el modelado de las tres perforaciones que se realizarán posteriormente. Para las perforaciones se han usado cilindros y se le han restado a la pieza principal. Para crear estos cilindros se le da clic a sólido y a cilindro. Pide punto central de la base, en el caso del primero va a ser en el punto 10,69.25,25. Y Con las medidas de 5 mm de radio y 25mm de altura. Con todas estas medidas se da clic a crear y se resta a la pieza grande como en los pasos anteriores.

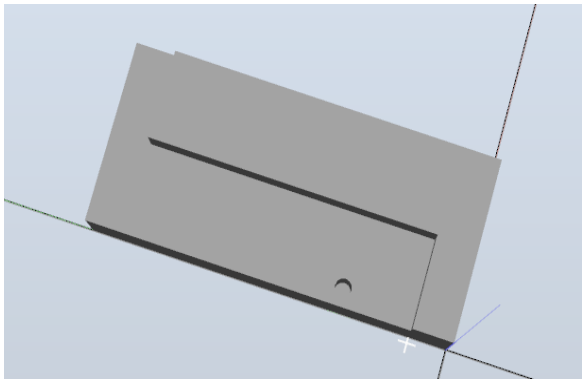


Figura 21 - Operación de perforado

Al tratarse de 3 perforados hay que repetir la operación igual con el siguiente cilindro. Con el ultimo lo único que va a cambiar es que el diámetro es de 21 mm y la altura de 15mm. El último paso va a ser cambiar el color a la pieza para verla con mayor claridad. Para ello clic derecho sobre la pieza, clic en seleccionar color y en este caso va a ser el naranja.

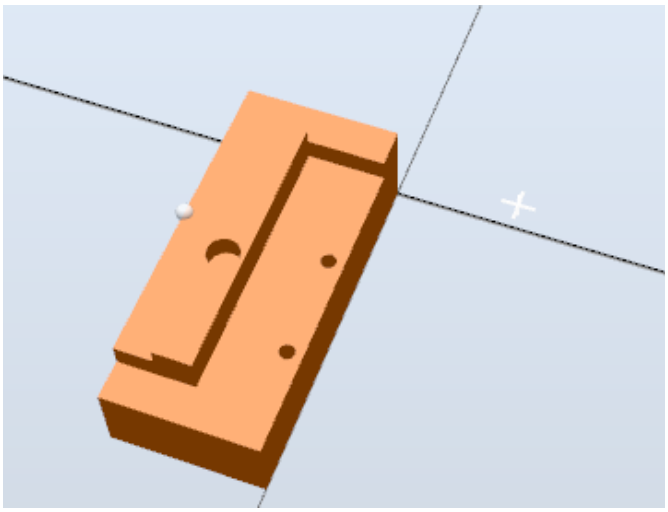


Figura 22 - Pieza final

En la imagen anterior está la pieza final a mecanizar en este proyecto. El siguiente y último paso de modelado va a ser crear la mesa de trabajo para el robot y la pieza. Esta mesa va a costar de dos cilindros de 250 mm de alto y 50 mm de radio y separados entre sí 800 mm. Por lo que para crearla se da clic en sólido, cilindro y se ponen las características descritas. En las coordenadas del primero se pone 0,400,0 y en las del segundo 0,-400,0.

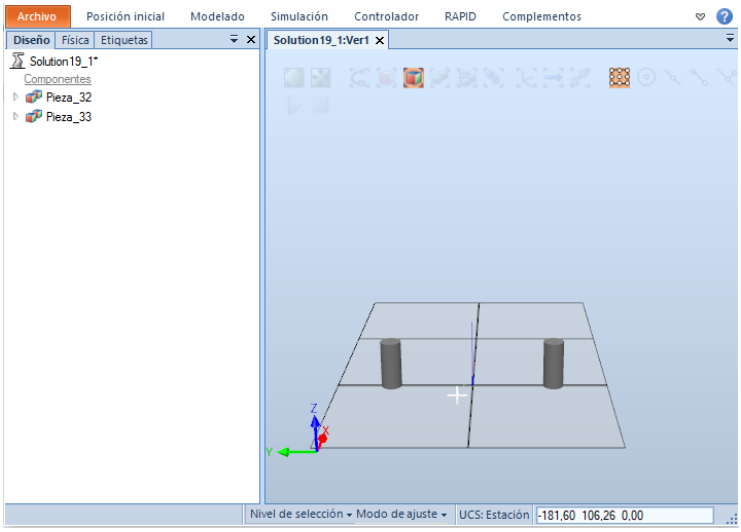


Figura 23 - Base de la mesa

Ahora hay que crear el tablero, este va a tener unas medidas de 500 mm de longitud, 1000 mm de anchura y 50 mm de altura. Para que coincidía con los cilindros y quede bien ajustado se colocara en los siguientes puntos de coordenadas: -250,-700,350.

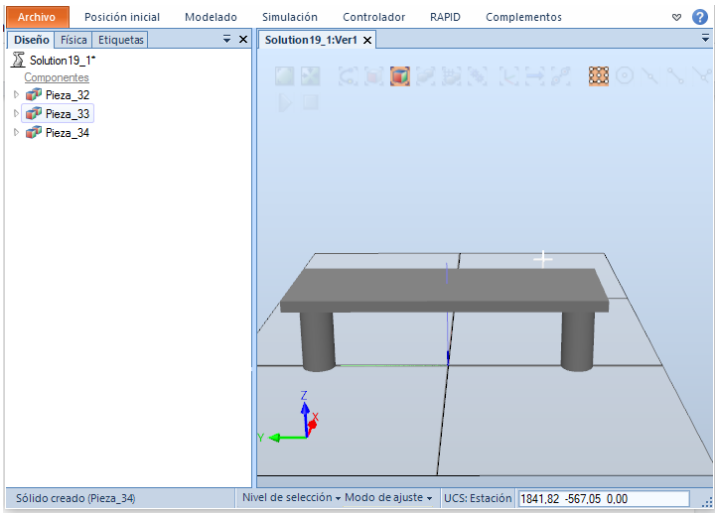


Figura 24 - Mesa de trabajo

Una vez finalizadas ambas piezas solo queda guardarlas e introducirlas en el fichero donde estaba la estación montada para proceder con la fase de programación.

4.5 Ubicación del entorno con el robot

Con el entorno de trabajo finalizado el siguiente paso es ubicarlo en la estación del robot correctamente, para que este pueda moverse con facilidad por la pieza sin ningún tipo de limitación por distancia. Esta limitación de distancia se puede ver en los datos técnicos del robot o dentro del programa.

Dentro de RobotStudio existe la opción de mover los ejes del robot con esto se puede observar de forma visual qué limitaciones se encuentran a la hora de realizar cualquier tarea. Esta opción se encuentra en el apartado de posición inicial del robot junto con las opciones de mover, girar y desplazar objetos las cuales se van a usar para ubicar la mesa y la pieza en la estación.

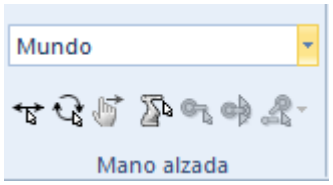


Figura 25 - Opciones de movimiento

Para comprobar el rango de movimientos del robot se usará la opción ‘’Movimiento de eje’’ una vez clicada hay que pulsar sobre cualquier eje del robot y este se puede mover con las limitaciones del robot.

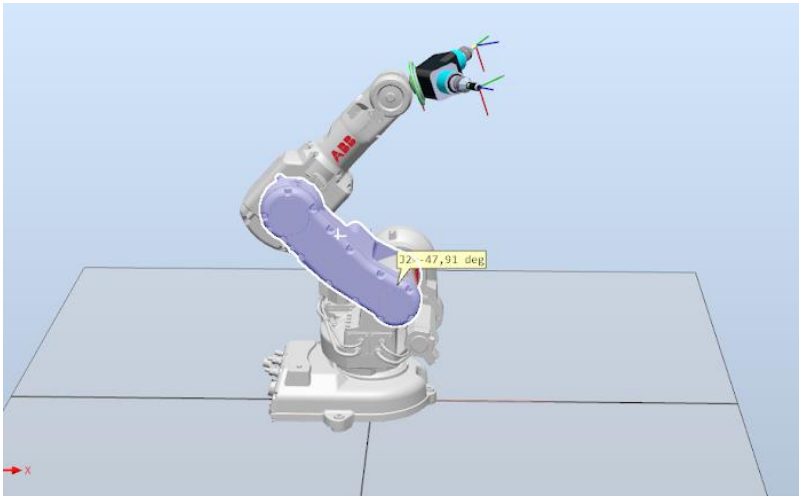


Figura 26 - Movimiento de los ejes del robot

De esta forma se puede mover el robot hacia cualquier posición y comprobar sus zonas de trabajo. Una vez esto se ha realizado el siguiente paso es introducir la mesa a la estación. Tanto la mesa como la pieza a mecanizar están guardadas en dos archivos de RobotStudio, la forma más sencilla de introducirlas en la estación es seleccionar el archivo y arrastrarlo dentro de la estación. Al meter cualquier pieza en RobotStudio esta se ubica donde se haya arrastrado, para moverla hay que darle clic en la opción ‘’Mover’’ y clic en el objeto a mover. Una vez hecho esto se abre un desplegable de 6 ejes, para mover la pieza hay que darle clic sobre el eje que se quiere mover y arrastrar hacia la posición que se quiere.

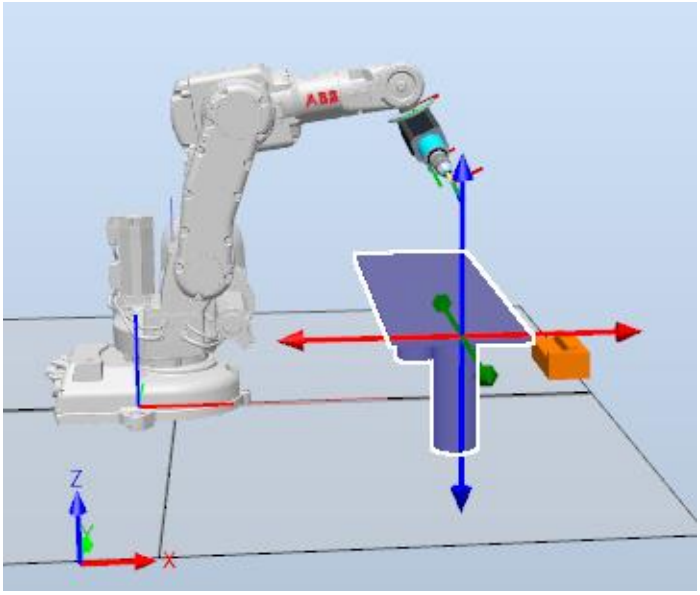


Figura 27 - Movimiento de objetos

Para finalizar hay que ubicar la pieza a mecanizar de la misma forma y encima de la mesa. Una vez esto realizado se pasa a la fase de programación.

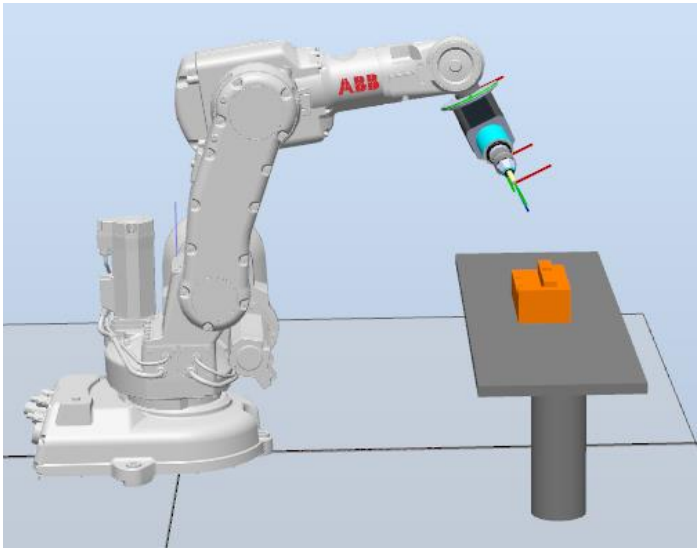


Figura 28 - Estación de mecanizado

4.5 Creación de objetos de trabajo y posiciones

Para empezar con la programación de esta tarea de mecanizado primero hay que establecer las posiciones del robot. Para ello hay que asegurarse de tener el controlador del robot activado ya que sin él no se pueden realizar estos pasos.

Las opciones de objeto de trabajo y posicionado sirven para:

- Crear los grupos de trabajo de fresado y perforado con su correspondiente herramienta.
- Diferenciar las distintas tareas del proyecto.
- Establecer posiciones base para posteriormente poder enlazarlas mediante trayectorias

Las opciones para crear objetos de trabajo y posiciones se encuentran en la opción de posición inicial.

Figura 29 - Opciones de posición y objeto de trabajo

El primer paso es situar el robot en la posición de reposo para poder crear el primer objeto de trabajo. Con la opción ‘’Movimiento de eje’’ se sitúa el robot en dicha

posición y se da clic en crear objeto de trabajo. El programa pide que se le dé un nombre al objeto de trabajo, en este caso ha sido ‘’Inició’’ y se le da clic en crear.

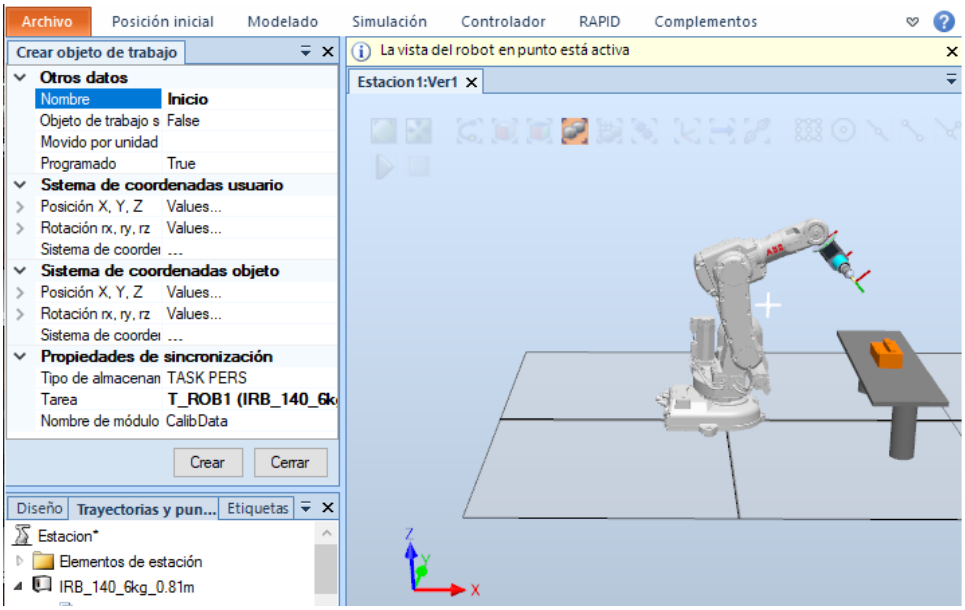


Figura 30 - Objetos de trabajo

Una vez creado el objeto de trabajo este sale por defecto dentro de las opciones de trayectorias y puntos. Aquí se verán los objetos de trabajo que se vayan creando.

El siguiente paso es comprobar que estemos dentro de ese objeto de trabajo y colocar la herramienta que se vaya a usar dentro de este. Esto se puede comprobar en la misma pestaña de posición inicial.

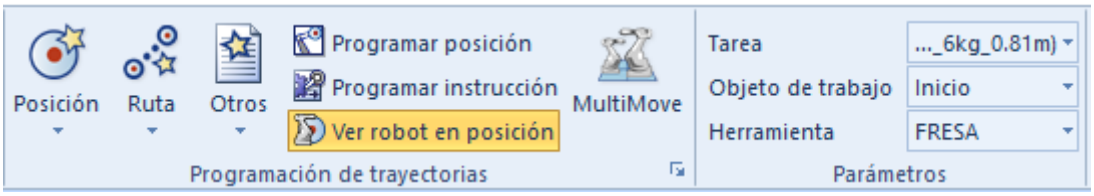


Figura 31 - Parámetros del objeto de trabajo

Este paso es importante porque en este trabajo la herramienta es doble y equivocarse poniendo la herramienta va a significar tener todo el objeto de trabajo mal.

El siguiente paso es declarar la posición del robot. Para ello hay que asegurarse de que el robot está en la posición deseada y darle clic a programar posición. Una vez hecho esto se genera dentro del objeto de trabajo un punto.

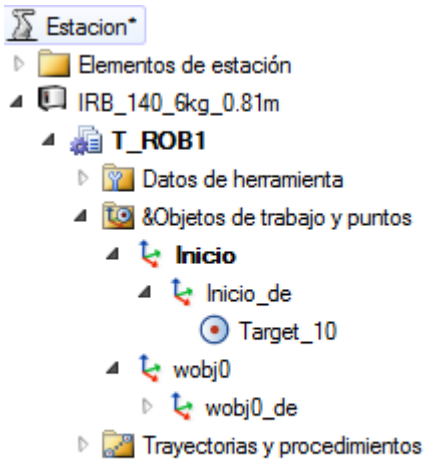


Figura 32 - Creación de posiciones

Con estos pasos ya se tiene la posición inicial del robot y se ha conseguido introducir la creación de posicionado y objeto de trabajo.

Ahora se va a empezar con la primera tarea de mecanizado, el fresado. Para ello primero se crea un nuevo objeto de trabajo. Este va a ser el fresado1. Una vez creado hay que asegurarse de que el objeto de trabajo esté bien puesto y la herramienta a utilizar sea la FRESA. Para comenzar con este objeto primero hay que posicionar el robot en una posición de reposo que sea segura y sin peligro de colisión. Una vez posicionado se le da clic en programar posición y ya estaría el primer punto guardado.

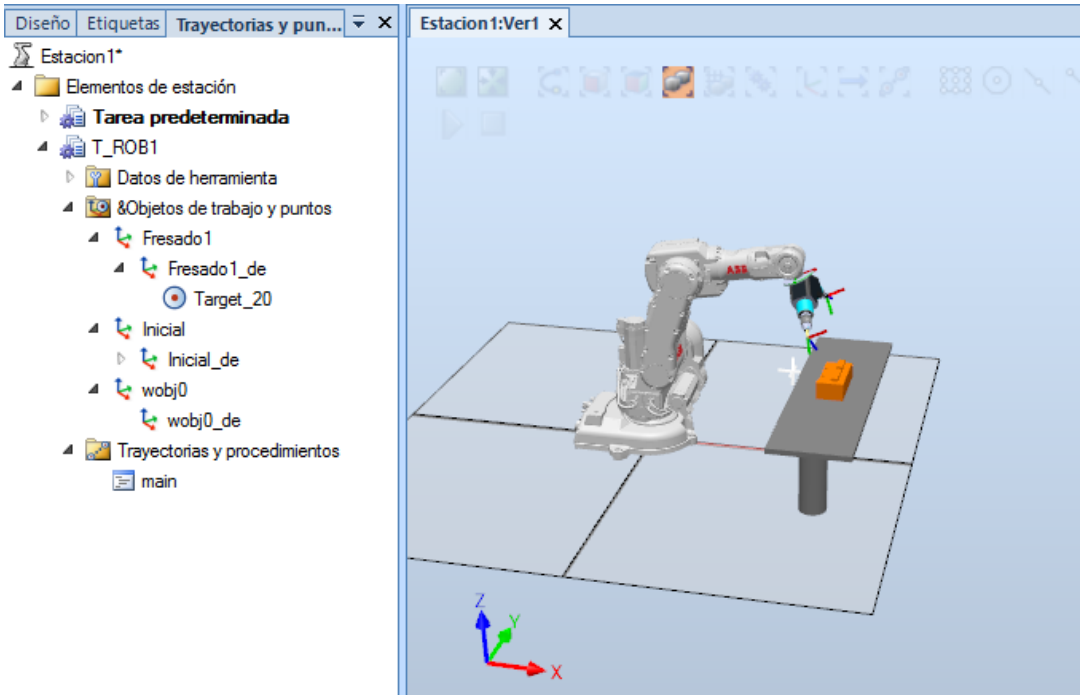


Figura 33 - Posición de reposo

El siguiente paso es bajar el robot a la posición de ataque. Para ello primero de todo hay que mover el eje de la herramienta 90 grados para que esté en perpendicular con la pieza y simule el fresado. Una vez perpendicular a la pieza se va a usar un nuevo comando para que al mover la herramienta el robot se desplace con ella con libertad de movimiento. Esta opción es ‘’Movimiento lineal’’ al darle clic y posteriormente a la herramienta aparecen 6 ejes. La herramienta se puede mover en cualquier de estos ejes y el robot se desplazará con ella en consecuencia.

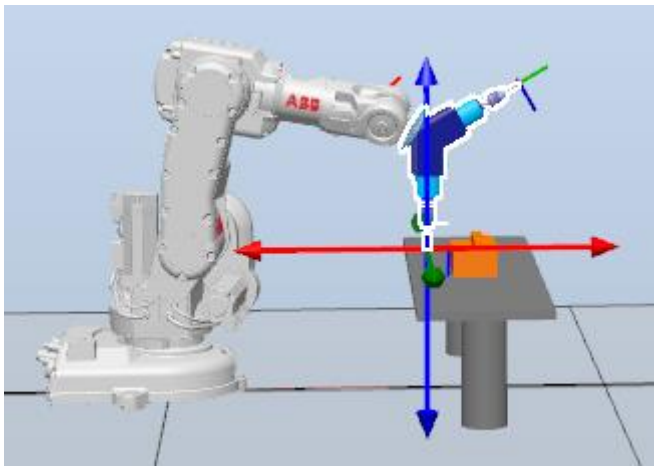


Figura 34 - Movimiento lineal

Para la posición de ataque del fresado se va a dejar una distancia de seguridad de 50 mm respecto a la pieza para evitar posibles colisiones. Esta distancia la dejaremos perfecta una vez se repase el programa con RAPID. El siguiente paso es marcar la posición de la herramienta para el comienzo del fresado. Con este fresado se le va a quitar a una parte de la pieza 10 mm de altura.

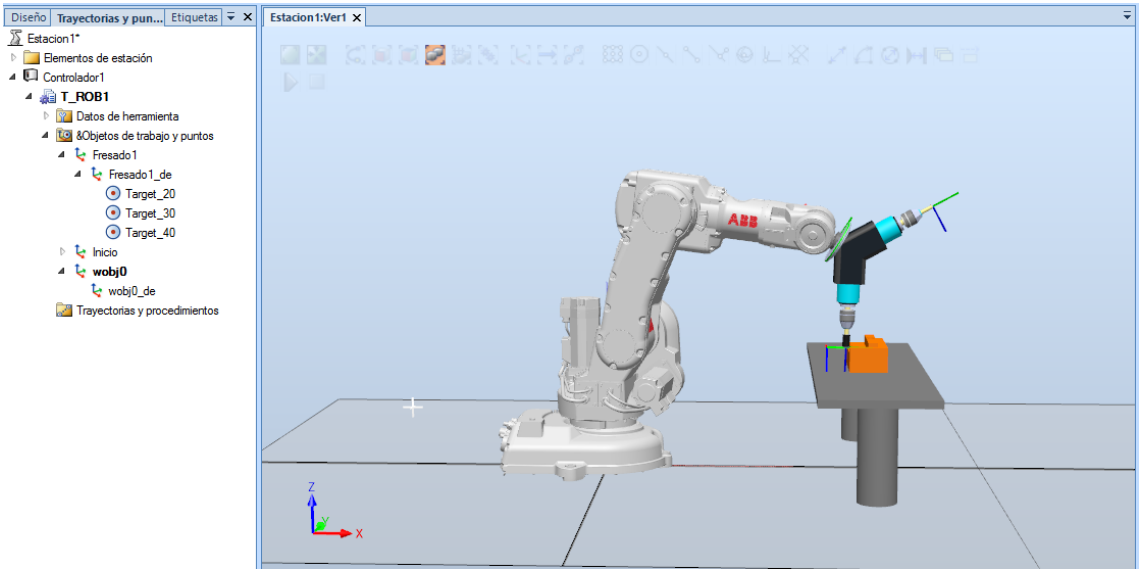


Figura 35 - Posición de ataque e inicial del fresado

Con estos pasos tenemos el principio de la primera tarea de fresado. Lo siguiente sería marcar posiciones en la pieza en las cuales haya giros inminentes para posteriormente crear las trayectorias. En el caso del primer fresado se van a crear 8 posiciones en total. Esto se va a conseguir moviendo la herramienta del robot hasta la posición exacta donde el robot tiene que girar y programando la posición en ese punto. Una vez se haga esto se verá en la pieza los puntos exactos por donde ha pasado la herramienta.

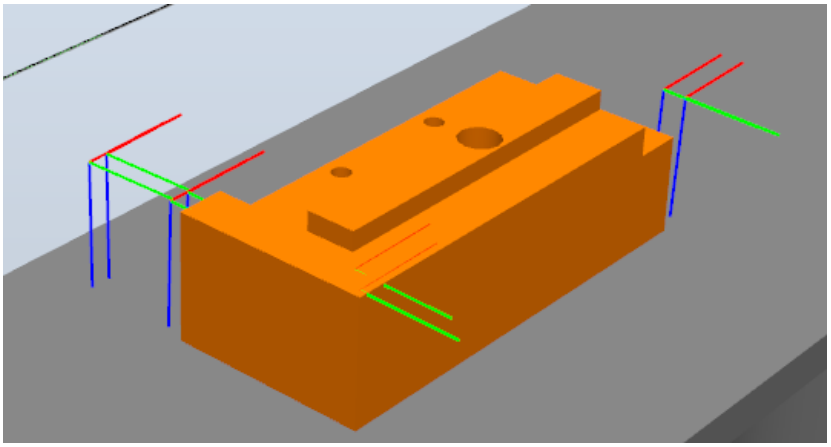


Figura 36 - Posicionado de la herramienta para el fresado

En la figura anterior se puede apreciar con los 3 ejes donde se ha posicionado la herramienta, este es un aspecto muy positivo que presenta Robotstudio. Ya que es un programa muy visual.

Una vez terminado el posicionado se retira la herramienta otra vez 50 mm por seguridad y se mueve hacia la siguiente fase de mecanizado. Esta fase va a ser el segundo fresado.

Para implementar el segundo fresado se crea un nuevo objeto de trabajo con el nombre fresado2. En este caso se le quiere quitar a la pieza 25 mm de altura. Se crea la primera posición con los 50 mm de seguridad y la segunda posición en la primera etapa del mecanizado. Este mecanizado va a constar de 14 posiciones más, ya que la pieza tiene bastante anchura y tiene que dar varias pasadas el robot. Una vez creadas el resultado es el siguiente.

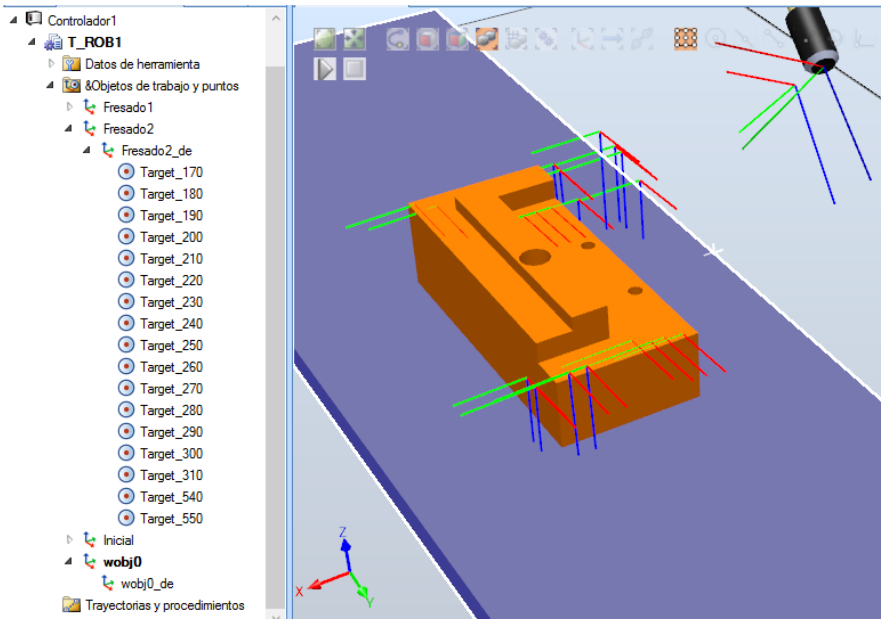


Figura 37 - Segundo fresado

Con esto ya estarían las etapas de rebajado de la pieza. Ahora faltan dos perforaciones con la broca y una última con la fresa. Primero que todo hay que cambiar de herramienta a la broca por lo que se volverá como primer paso a la posición de reposo. El resto de las posiciones en el mecanizado de la perforación van a ser:

- Posición de seguridad a 50 mm de la pieza.
- Herramienta encima de la zona de perforación
- Herramienta perforando la pieza
- Subir la herramienta en recto a la posición de seguridad de 50 mm

Estas posiciones se usarán también en el segundo perforado con broca. En la siguiente imagen se pueden apreciar las posiciones. En este caso se han dejado en invisibles las posiciones del fresado para que se vea con más claridad el posicionado de las perforaciones.

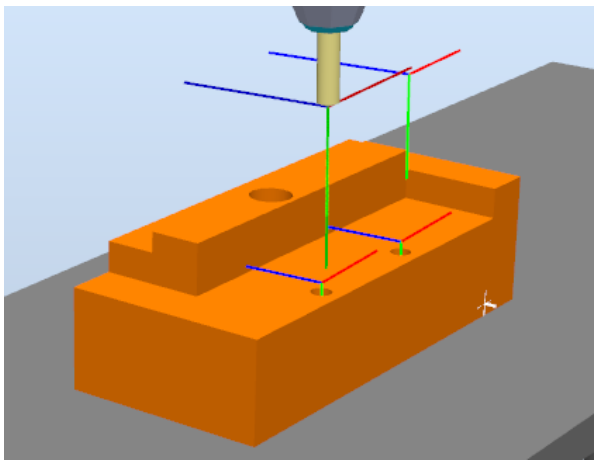


Figura 38 - Posiciones del perforado

Por último, para finalizar con el posicionada, falta el segundo perforado con la fresadora. Primero hay que mover el robot a la posición de reposo para cambiar de herramienta y el resto es idéntico al anterior mecanizado. Posicionado a 50 mm de seguridad, posicionado encima de la pieza, perforado en la pieza y salida a la posición de seguridad.

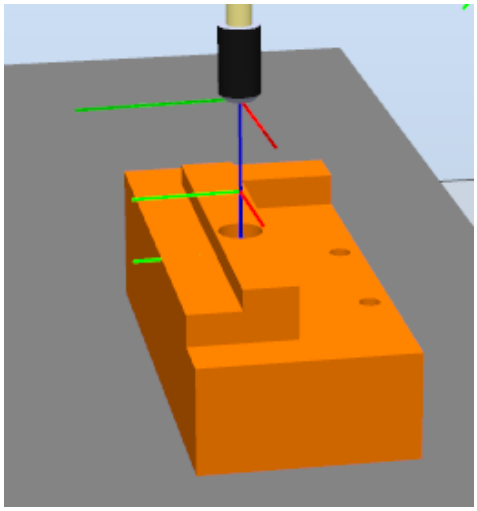


Figura 39 - Posicionado del segundo perforado

Con esto ya estarían los posicionados para todo el mecanizado, el siguiente paso va a ser crear las trayectorias del robot a través de la pieza.

4.6 Creación de las trayectorias del mecanizado

Una vez puesto todos los puntos sobre la pieza es necesario revisarlos al igual que su orden. Si se crean las trayectorias y el orden está mal o el punto no está en el sitio correcto se harán mal las trayectorias y habrá errores en el mecanizado.

El paso siguiente es pasar a crear las trayectorias. Para ello se va a la pestaña de ‘’trayectorias y puntos’’ que está ubicada a la izquierda por defecto en RobotStudio. Se selecciona la opción de ‘’trayectorias y procedimientos’’ y se le da clic derecho y clic en crear trayectoria. Con esto se habrá creado una nueva trayectoria con un nombre predefinido, en este proyecto se le ha cambiado el nombre para evitar confusión.

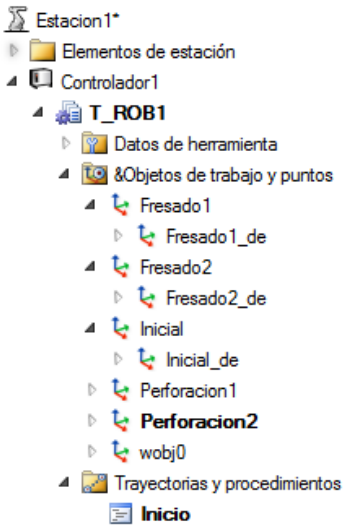


Figura 40 - Creación de trayectorias

Lo siguiente es arrastrar la posición a la trayectoria. La primera posición es la inicial por lo que solo tiene un punto dentro de ella. Se abre este punto y se arrastra hacia la trayectoria inició como se puede observar en la siguiente figura. Esta trayectoria va a servir en las fases de inicio y fin del ciclo para mover el robot a una posición segura.

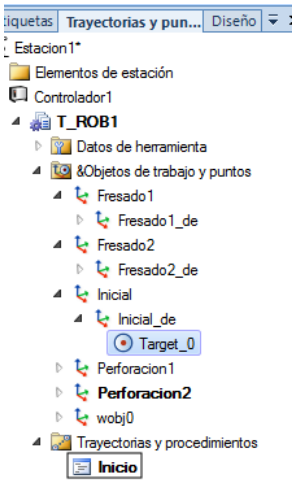


Figura 41 - Primera trayectoria

Para los siguientes mecanizados los pasos son similares a los anteriores:

- Se crea una nueva trayectoria.
- Se cambia el nombre para organizarlo.
- Se arrastran todos los puntos hacia la trayectoria deseada.
- Se comprueba que el robot ha trazado correctamente la trayectoria.

La segunda trayectoria a realizar es el ‘‘fresado1’’ en la siguiente figura se ve como se han trazado las líneas por donde el robot va a simular el mecanizado.

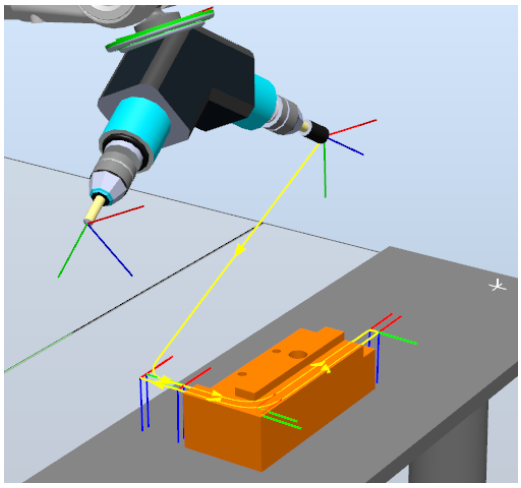


Figura 42 - Trayectoria del primer fresado

Esta es una trayectoria en bruto que posteriormente se va a repasar con la programación en RAPID para pulir errores que se puedan producir. Estos errores se pueden apreciar en las curvas de la trayectoria. Si se dejara así el robot eliminaría partes de la pieza que no se desean y realizaría mal el mecanizado.

Los siguientes pasos son iguales, hay que crear las trayectorias correspondientes, identificarlas como tal y una vez creadas e implementadas revisar paso a paso que el robot sigue las posiciones creadas correctamente.

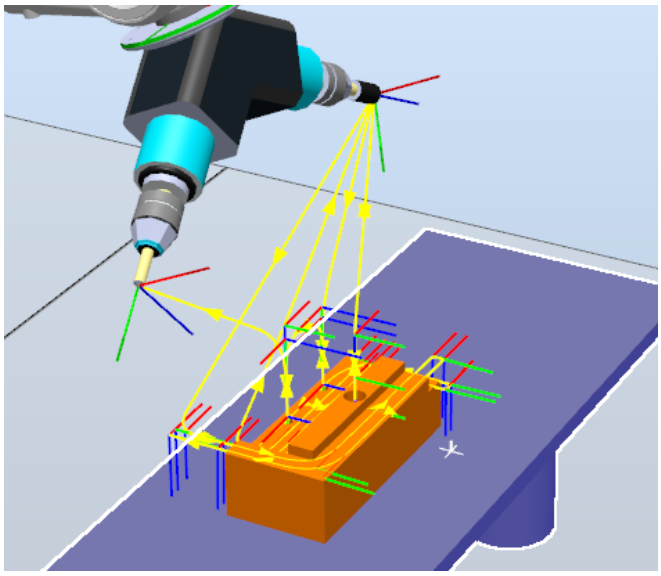


Figura 43 - Pieza con todas las trayectorias

En la figura anterior está el robot finalizado con todas las trayectorias que tiene que realizar. Y en la siguiente figura se van a ver las 4 trayectorias por separado. Para observar mejor las rutas.

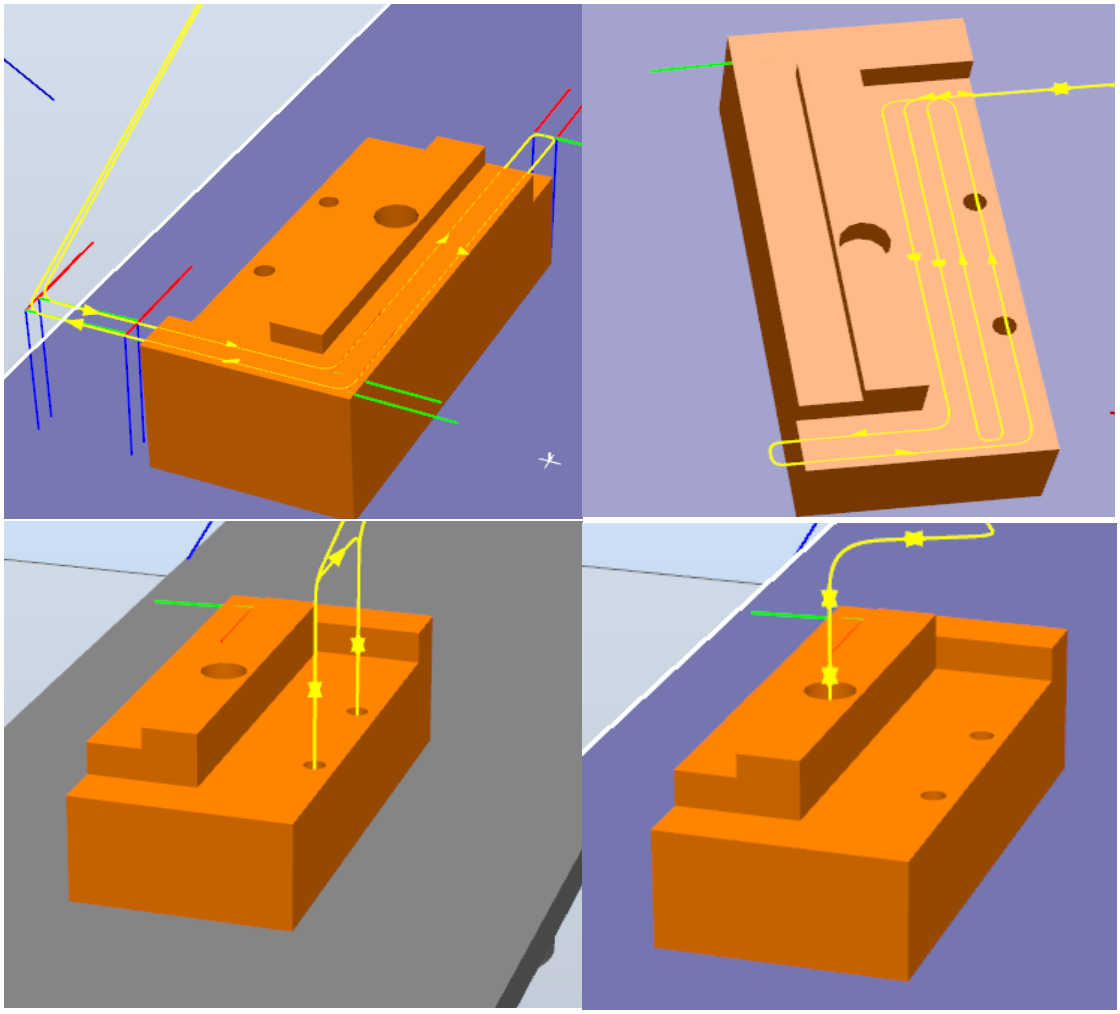


Figura 44 – Trayectorias de los mecanizados

Con el programa hecho así en bruto el robot tiene por defecto unas velocidades y unos errores iguales en todos los puntos. Por lo que el siguiente es pulirlo, esto se puede hacer desde el mismo menú de las trayectorias con opciones que nos da el programa para cambiar velocidades y errores. Pero es mucho más efectivo hacerlo desde la programación con RAPID, ya que se ve toda la programación y se puede modificar todo de forma más precisa.

4.7 Introducción a RAPID

RAPID es el lenguaje de programación creado por ABB. Esta programación consta de un programa principal y módulos secundarios. Programar en RAPID se puede hacer de dos maneras distintas:

- Creando a mano los programas desde cero. Esta opción se ha cursado en la asignatura de ‘’Sistemas Robotizados’’.
- Transfiriendo los datos de una estación con las trayectorias previamente programadas mediante la sincronización del controlador virtual con RAPID. De esta forma se crea el programa con todos los puntos y movimientos realizados como es el caso de este proyecto.

Para esto hay que dirigirse a la pestaña de inicio y seleccionar la opción de sincronizar con RAPID. Cuando se le da a la opción el controlador pregunta que se desea sincronizar, en este caso hay que seleccionarlo todo para que se sincronice de una forma correcta.

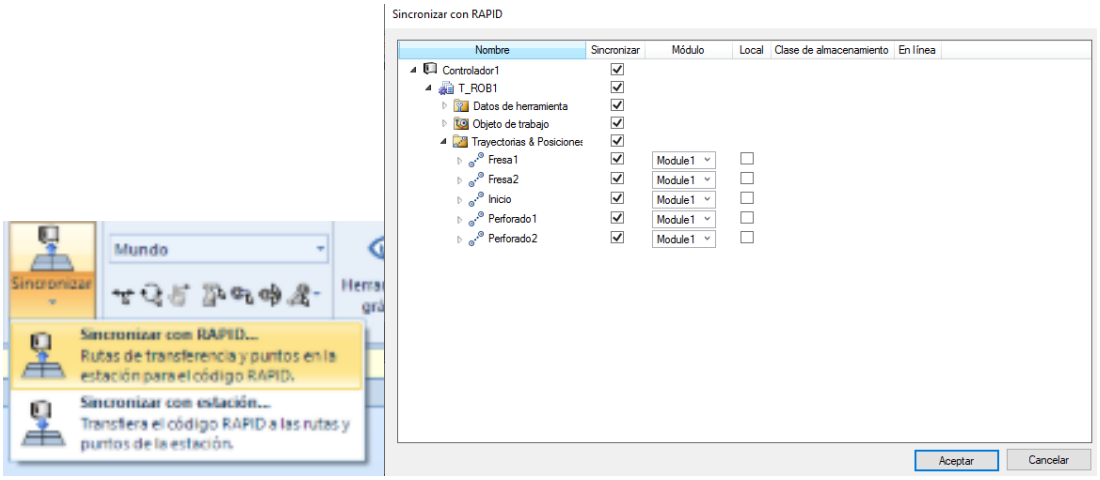


Figura 45 - Sincronizar con RAPID

Una vez ya convertido en RAPID las trayectorias y posiciones anteriores se va a pasar a explicar los comandos utilizados en este programa.

El programa ‘’MODULE1’’ es el archivo que ha creado RAPID al sincronizarlo. Este es el programa principal del proyecto y contiene módulos con cada una de las trayectorias que se han realizado.

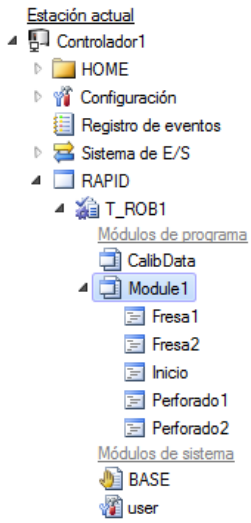


Figura 46 - Programa principal y módulos

Este programa principal se inicializa con el comando de ‘’MODULE Module1’’ y se finaliza con ‘’ENDMODULE’’. Dentro de este programa se encuentran las posiciones programadas previamente. Estas posiciones empiezan con el comando ‘’CONST robtarget’’ y siguen con su nombre que en este caso se ha dejado el de defecto, las coordenadas en la estación y la configuración de los ejes para el punto.

```
CONST robtarget
Target_0:=[[0,0,0],[1,0,0],[0,1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

El programa tiene todas las posiciones realizadas. Lo siguiente son los módulos uno a uno. Estos módulos empiezan en el caso del primero con el comando ‘’PROC fresado1()’’ y finalizan con ‘’ENDPROC’’. Lo que contiene este programa son las trayectorias que se han realizado previamente con el tipo de movimiento que se ha usado, la velocidad del robot, el error que puede tener de un punto a otro y la herramienta que se ha usado.

```
MoveL Target_30,v100,z10,FRESA\WObj:=Fresado1;
```

En RAPID existen tres tipos de movimientos principales:

- MoveL: Este movimiento es lineal, es decir, el robot va de una posición a otra con la mínima distancia, en línea recta.

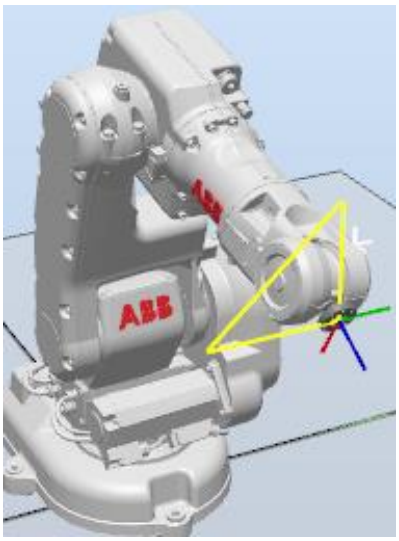


Figura 47 - Ejemplo de MoveL

- MoveJ: El movimiento en este caso no es lineal. Es un movimiento libre. Se puede apreciar la diferencia de movimientos entre la figura anterior y la figura siguiente.

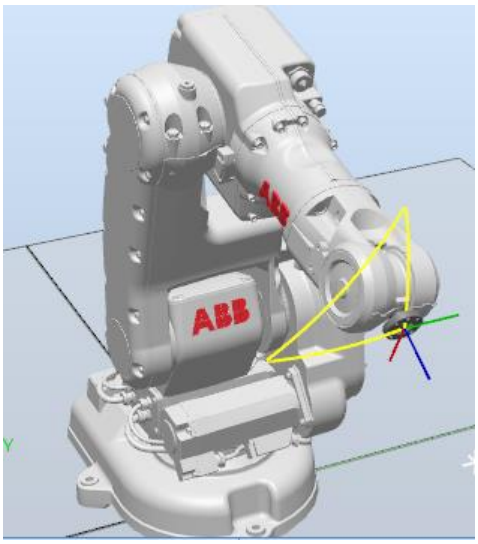


Figura 48 - Ejemplo de MoveJ

- MoveC: Esta instrucción sirve para hacer trayectorias circulares. Para la realización de un MoveC es necesario dos posiciones. Una en el punto intermedio de la circunferencia y otra en el extremo final. En el caso de este proyecto no se ha llevado a cabo ningún MoveC.

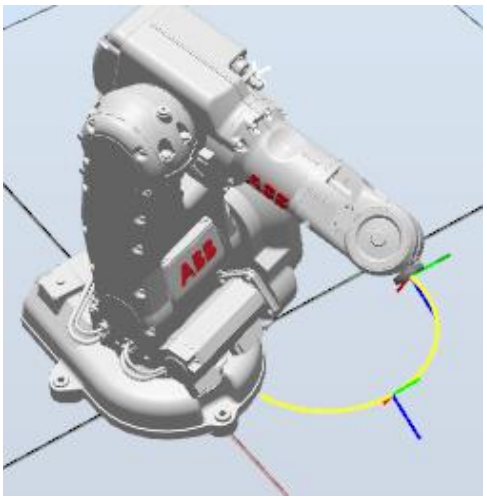


Figura 49 - Ejemplo de MoveC

En cuanto a la velocidad de este robot va desde v5 a v7000. Significando v5 que va a una velocidad de 5 mm/s y v7000 a una velocidad de 7000 mm/s. Esta velocidad máxima va a depender también del robot utilizado.

El siguiente dato para dar es el error del movimiento. Este error va a afectar a la hora de pasar de una posición a otra ya que a mayor error menor precisión tendrá la herramienta y peor hará su trabajo. El error puede ir desde ‘’fine’’ que seria 0 mm de error pasando por z100 que serían 100 mm de error, hasta z200 que son 200 mm de error. También puede elegirse z0 que tiene un error de 0.3 mm aproximadamente. El tipo de error que se elija va a ser relevante ya que a menor error más tardara el robot en realizar el mecanizado. Pero a mayor error mayor probabilidad de fallo.

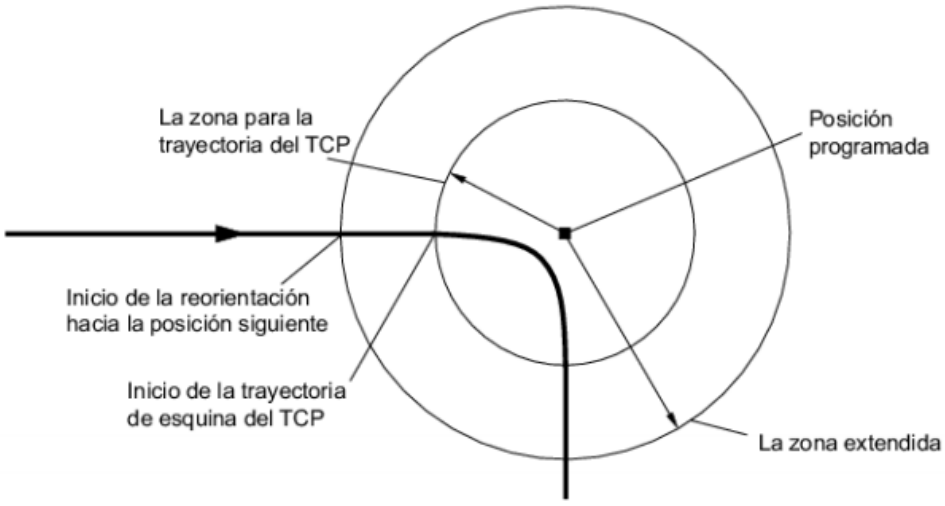


Figura 50 - Error de precisión entre dos posiciones

En la figura anterior se aprecia como la trayectoria viene por la parte izquierda y hace un ángulo de 90º. Inicialmente la trayectoria debería ir de la parte izquierda hacia el

punto programado y luego hacer el giro de 90º hacia el siguiente punto. Pero esto se ve afectado por el error que se le ponga al programa. A mayor error más tiende el programa a acortar el camino y por lo tanto se aleja más del punto programado.

Este error en la precisión se puede ver claramente en la parte del trabajo donde se han trazado las primeras trayectorias. Al hacer un giro de línea a línea la trayectoria hacia un semicírculo. Esto es debido a que el error que genera por defecto el programa es de z10, de 10 mm por esto se produce. Para solucionar este problema se sigue un esquema de movimientos, errores y velocidades:

- Cuando la herramienta está fuera de la zona de seguridad el robot puede tener un movimiento libre ‘’MoveJ’’, un error grande y una velocidad alta. Ya que no tiene nada con lo que pueda colisionar. En el caso de este proyecto esto se da cuando se va de la zona de seguridad al cambio de herramienta o a la posición inicial.
- Cuando la herramienta está bajando de la distancia de seguridad a la zona de trabajo el robot tiene un movimiento lineal ‘’MoveL’’, error y velocidad pequeños. Para que no colisione con ninguna parte de la pieza una vez llegue a ella.
- Cuando la herramienta está en la zona de trabajo. Aquí el robot puede tener dos tipos de movimientos el lineal ‘’MoveL’’ o el circular ‘’MoveC’’ y siempre velocidad baja y error pequeño. Esto es así para no cometer errores en la pieza y tener mayor precisión.
- Por último, cuando la herramienta se retira a la zona de seguridad. Aquí el robot tendrá un movimiento lineal ‘’MoveL’’, velocidad baja y error mediano. Para volver de una manera más rápida a la zona de seguridad, pero sin causar ningún problema a la pieza.

Por último, en la línea de comando también se indica la herramienta que se está utilizando para cada movimiento. Este es un punto importante para revisar porque si se hace con la herramienta equivocada toda la línea de código va a ser errónea.

Una vez todos estos conceptos introducidos el paso siguiente es arreglar el código, ya que hay que ajustar movimientos velocidades y errores en consecuencia con lo que se ha explicado. Las pautas para seguir van a ser las siguientes:

- Fuera de la zona de seguridad siempre va a ser movimiento libre ‘’MoveJ’’ velocidad alta ‘’v300’’ y error grande ‘’z50’’. Ejemplo:

  MoveJ Target_30,v300,z50,FRESA\WObj:=Fresado1;

- De la zona de seguridad a la de trabajo movimiento lineal ‘’MoveL’’, velocidad pequeña ‘’v150’’ y error pequeño ‘’z0’’. Ejemplo:

  MoveL Target_40,v50,z0,FRESA\WObj:=Fresado1;

- En la zona de trabajo igual a la anterior, movimiento lineal ‘’MoveL’’, velocidad pequeña ‘’v150’’ y error pequeño ‘’z0’’. Ejemplo:

  MoveL Target_100,v50,z0,FRESA\WObj:=Fresado1;

- Por último, de la zona de seguridad a la zona de trabajo a la de seguridad, movimiento lineal ‘’MoveL’’, velocidad pequeña ‘’v150’’ y error mediano ‘’z20’’. Ejemplo:

MoveL Target_150,v50,z20,FRESA\WObj:=Fresado1;

Estos van a ser los criterios para todo el programa de RAPID. Con todo esto cambiado en el programa falta por añadir una nuevo módulo al programa. Este va a ser el módulo del mecanizado. Dentro de este módulo tiene que llevar por orden los mecanizados a realizar. Para llamar a un módulo dentro de otro se hace con el comando ''Nombre del módulo;'' por ejemplo en el caso del primer módulo es ''Inicio''.

Una vez añadido todos los módulos hay que sincronizar el RAPID con el controlador virtual. Este es el proceso inverso que se ha dado al principio. Para ello en las opciones de RAPID se encuentra la opción sincronizar, clic sobre la flechita que se encuentra a la derecha de esta opción y clic sobre sincronizar con estación. Saldrá un cuadro con opciones para marcar. Se marcan todas las opciones para transferirlo a la estación.

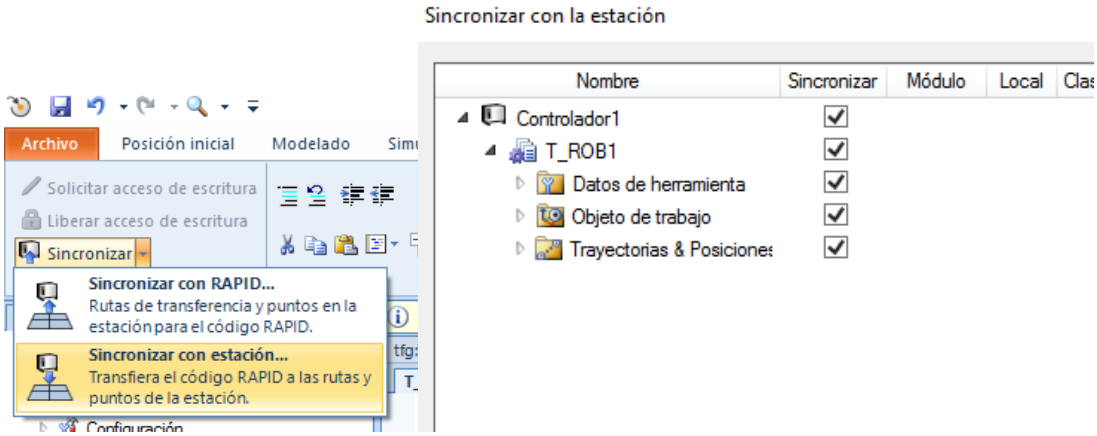


Figura 51 - Sincronización de RAPID con la estación

En cuanto se ha transferido las trayectorias cambian y al recortarse los errores ya no se aprecian los semicírculos en los giros de las curvas de punto a punto. Ahora los giros son de 90º y la herramienta no corre peligro de hacer mal la trayectoria.

El último paso va a ser añadir tiempo de espera en ciertas etapas del mecanizado para hacer más realista el mecanizado. Estas etapas de espera van a colocarse en:

- Posición antes de cada ataque a la pieza.
- En cada giro del fresado.
- En los cambios de herramienta.
- Por último, en las bajadas de las perforaciones.

El comando para las etapas de pausa es ''WaiTime X; siendo la X el tiempo en segundos que se le desea poner.

4.8 Simulación en RobotStudio

Finalizada toda la programación pertinente se pasa al último paso del RobotStudio que es la simulación. Primero se va a ver todas las trayectorias realizadas en una imagen.

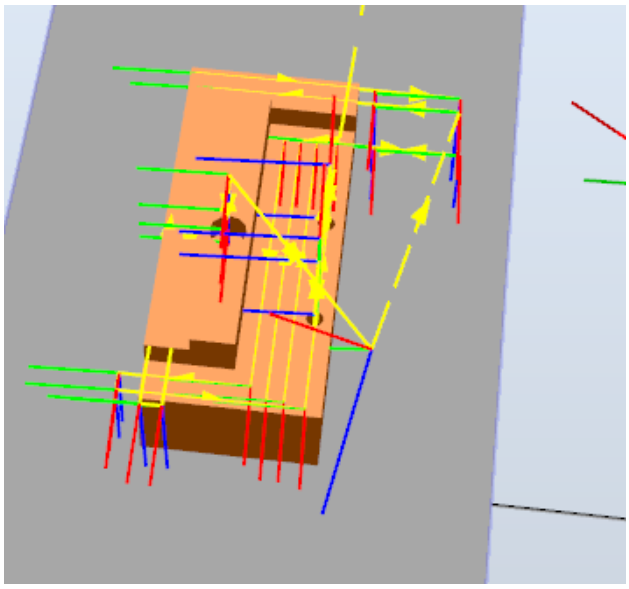


Figura 52 - Resultado final trayectorias

Este es el resultado final de las trayectorias. Ahora se va a ver este resultado trayectoria a trayectoria.

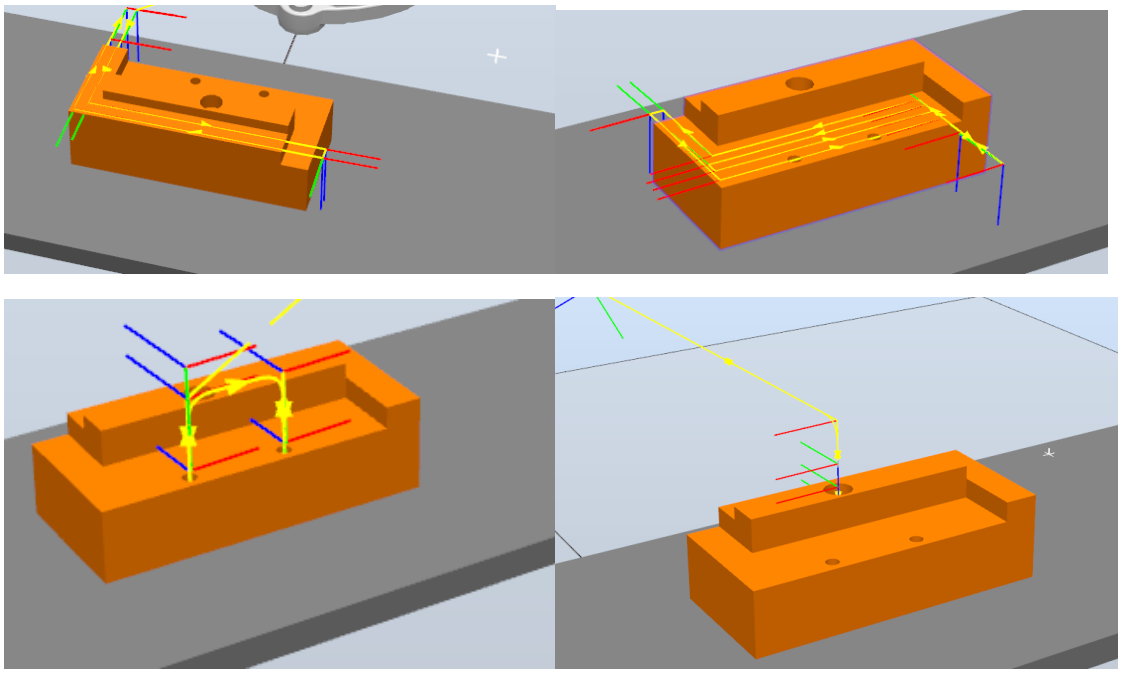


Figura 53 - Trayectorias del mecanizado

Por último, se va a hacer una secuencia de imágenes de la simulación del mecanizado. Para observar más a fondo las trayectorias y el programa. La opción de simular está situada junto al resto. Hay que dar clic en la pestaña de simulación y darle a la opción de reproducir. Si se quiere ver la trayectoria que hace el robot en la simulación existe una herramienta llamada ‘’Rastreo de TCP’’. Se le da clic a la herramienta y clic en activar el rastreo TCP. Una vez dentro del panel se le puede cambiar el color para hacerlo más visual.

Figura 54 - Pestaña simulación y rastreo TCP

Con esta secuencia de imágenes concluye el apartado de RobotStudio.

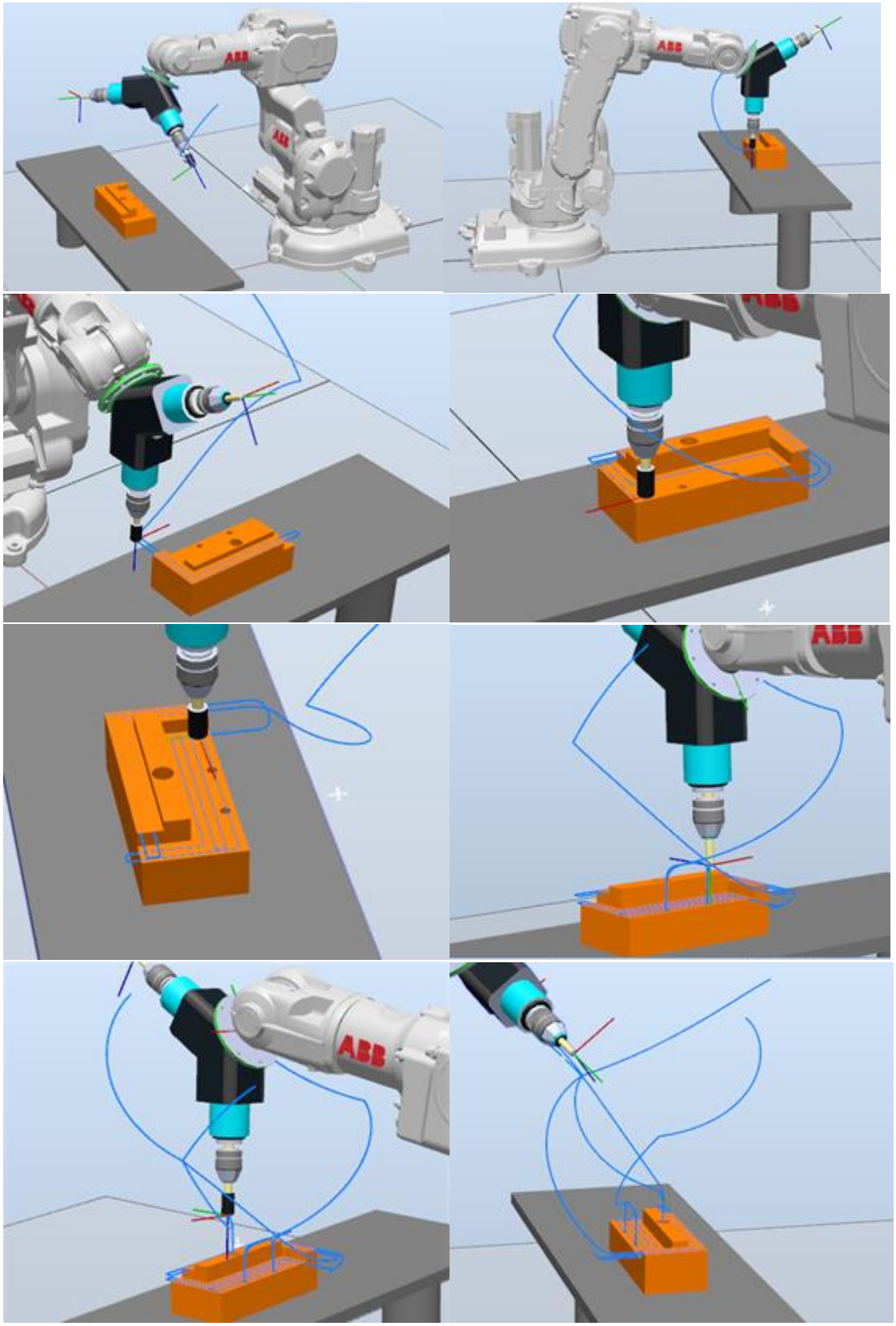


Figura 55 - Simulación de mecanizado en RobotStudio.

5. Implementación en el PolyScope

5.1 Introducción a PolyScope

El programa PolyScope es un entorno de desarrollo gratuito basado en Linux. Esta herramienta permite la simulación offline, fuera de línea, de los robots colaborativos de Universal Robots. Una de las marcas más importantes a nivel mundial en la fabricación de robots colaborativos.

PolyScope abarca tres programas distintos. Cada uno de estos programas está diseñado para cada uno de sus robots colaborativos. Estos son el UR3, UR5 y UR10. Cada robot tiene su programa debido a que cada uno tiene un alcance y medidas distintas, pero con la misma interfaz, PolyScope.

Las características principales de PolyScope son:

- Herramienta de gran utilidad y versátil.
- Entorno muy visual y sencillo de programar.
- Es fácil modificar los datos en la programación del robot.
- Cada robot tiene su propio programa especializado.
- No es necesario usar lenguaje de programación, se puede programar a mano.

Este programa es muy visual y destaca por su sencillez y comodidad a la hora de realizar tareas con el robot.

5.2 Introducción a UR5

El robot UR5 es el robot mediano entre el UR3 y el UR10. Pertenece a la gama de robots colaborativos de Universal Robots y va a ser el elegido para este trabajo. Es flexible, seguro, genial para tareas de automatizado de poco peso y fácil de programar y configurar.

Este robot presenta 6 grados de libertad. Tiene una carga útil de hasta 5kg de ahí su nombre de UR5. Tiene un alcance de 850 mm. Y el robot pesa 20.6 kg.

El UR5 es un robot colaborativo. Tiene dos principales ventajas. La primera es que pueden compartir espacio y tareas con otros cobots mediante métodos comunicativos inteligentes. La otra ventaja es que puede compartir el espacio con trabajadores, además de que no necesitan un espacio de trabajo propio ni jaula, por lo general.

Figura 56 - Robot UR5

5.3 Puesta a punto del robot UR5 en PolyScope

El programa de Universal Robots PolyScope trabaja en el sistema operativo de Linux. Por lo que el primer paso para poder trabajar con el programa ha sido descargarse un simulador de Linux. El propio Universal Robots ofrece los pasos para instalar sus softwares. La página te va preguntando qué necesitas descargar y si indicas que quieres el software en un dispositivo que no utiliza Linux te da las posibles opciones. Estas son instalar una máquina virtual. Las dos que ofrece son ‘’VMWARE Player y VirtualBox’’. La que se ha usado para este proyecto ha sido VirtualBox por la facilidad que ofrece. La propia página de Universal Robots ofrece todos los pasos y el programa listo para instalar.

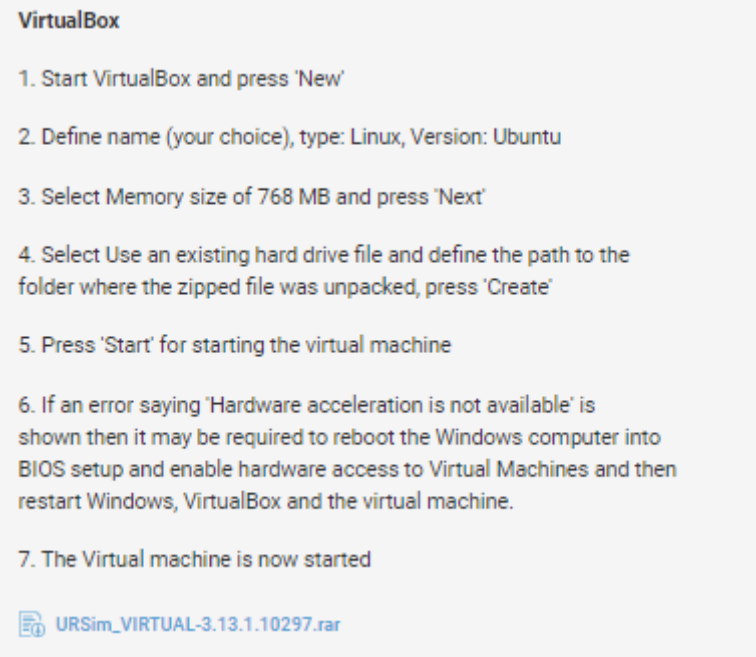


Figura 57 - Instalación de VirtualBox

Una vez instalada la máquina virtual hay que iniciarla. Esta hace la función de un ordenador con el sistema operativo Linux. Al iniciarse, por defecto, tiene instalados los programas de Universal Robots para cada uno de sus tres robots. En la siguiente figura se va a apreciar como es el primer contacto con la máquina virtual y Linux.

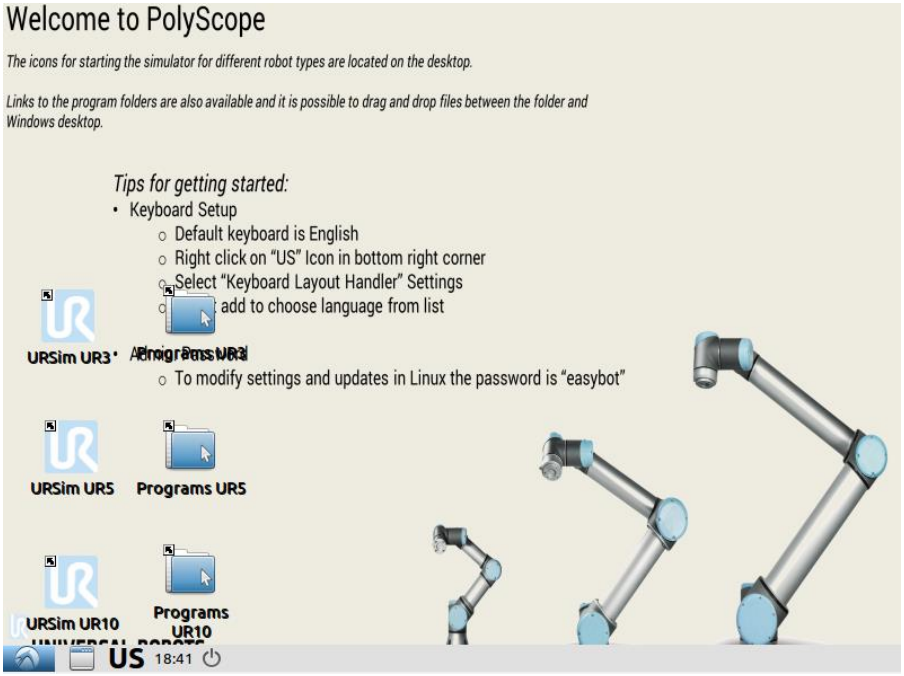


Figura 58 - VirtualBox iniciado

Como ya se ha indicado, se va a utilizar el robot UR5 por lo que habrá que seleccionar el programa URSim UR5. La interfaz que se ve al abrir el programa es la siguiente.

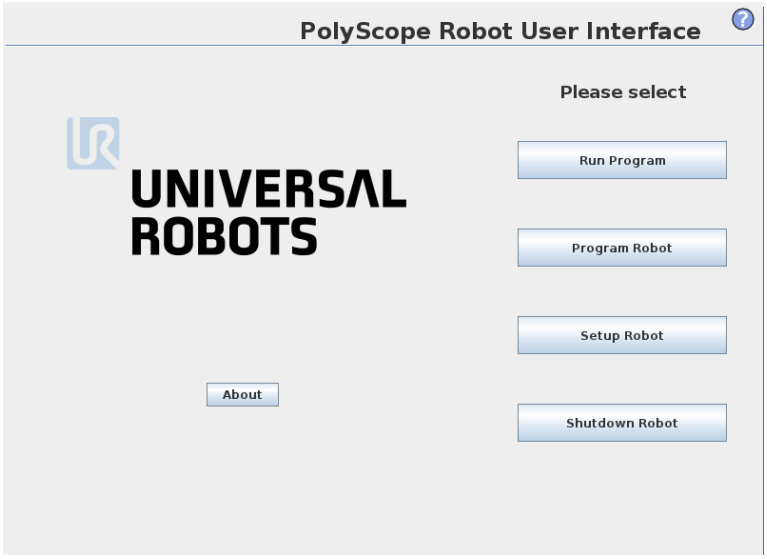


Figura 59 - Interfaz PolyScope.

El primer paso es configurar el robot. Cabe destacar que en este programa no ha sido posible añadirle herramienta ni un entorno al robot de mesa y pieza. Por lo que la configuración del robot va a ser simple. Va a ser una recogida de datos para la hora de programar el robot.

Este es el programa básico de Universal Robots, pero existen otros softwares de programación más complejos que permiten el uso de herramientas, piezas y entornos. Uno de estos softwares es RoboDK. Este tiene funciones similares a RobotStudio y abarca tanto los robots de la marca Universal Robots como los de ABB. El inconveniente es que es un programa con una dificultad mayor y funciona con licencias de pago.

Para configurarlo se le da clic en setup robot. Y se ve la siguiente lista de opciones a cambiar.

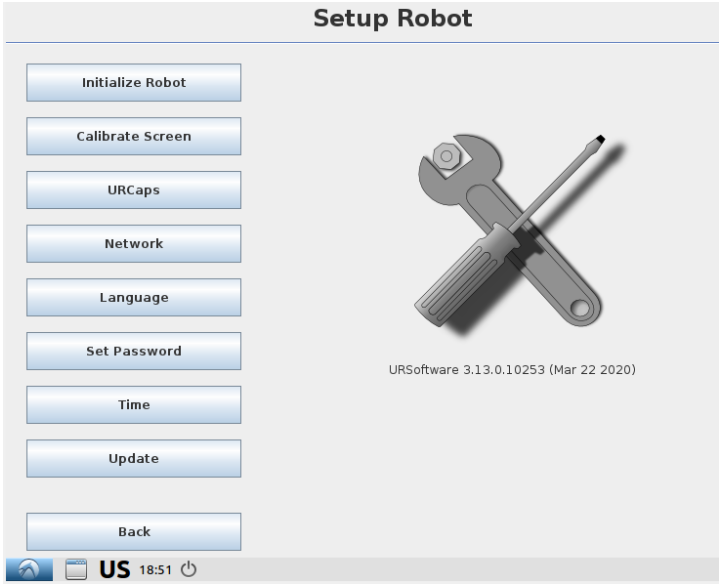


Figura 60 - Configuración del robot

Lo primero va a ser darle clic en inicializar el robot en esta opción se puede enchufar o apagar el robot, por defecto viene encendido. También vienen las opciones del peso de la carga y la herramienta que se le va a poner al robot. Por último, permite configurar el centro de coordenadas de la herramienta.

En el caso de este trabajo se le va a poner una carga activa de 2 kg, este es el peso de la herramienta. Para configurar el centro de coordenadas se le da a ‘’Configure TCP’’ y para simular una única herramienta se le va a poner el centro en la posición 0,0,50 mm. Esto se va a hacer para simular las características de la herramienta en el robot. En el caso de tener una herramienta real y saber su peso y sus medidas se podría calibrar perfectamente en el programa.

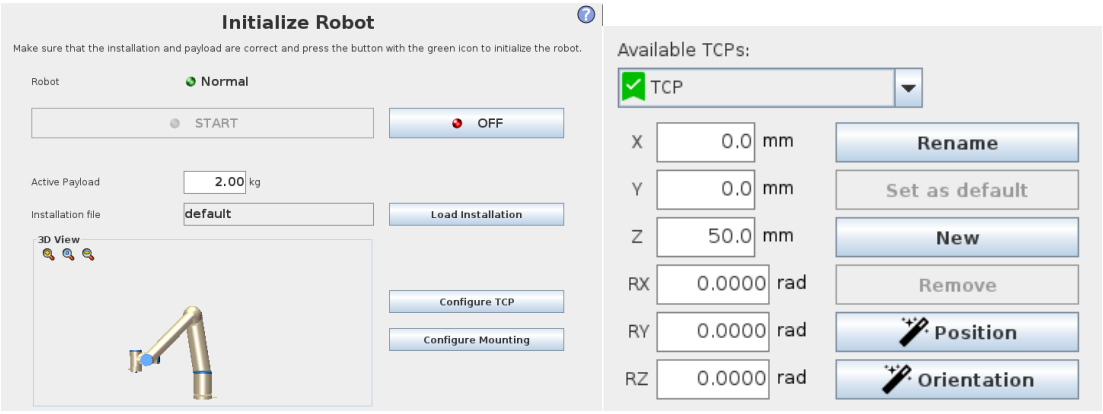


Figura 61 - Puesta a punto del robot UR5

Las siguientes opciones que nos presenta el robot se van a quedar igual. Por lo que el siguiente paso va a ser programar el robot.

5.4 Introducción a la programación en PolyScope

Al crear un programa nuevo en PolyScope se presentan una serie de pestañas. Para el caso de este trabajo y debido a que es un mecanizado se va a hacer uso de las básicas.

Para comenzar con la programación del mecanizado hay que darle clic a ‘’Move’’. Este comando creará un tipo de movimiento para realizar trayectorias.

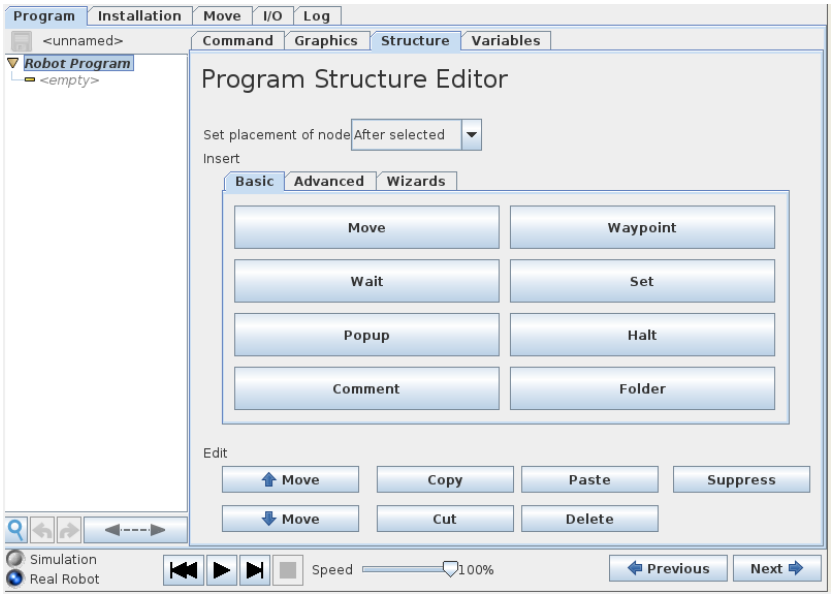


Figura 62 - Comandos básicos para la programación en PolyScope

Por defecto el comando que se crea es un MoveJ al igual que en RobotStudio tenemos los tres tipos de movimientos básicos a elegir.

- MoveJ: Movimiento libre no lineal. Este movimiento es el más rápido de todos y se usa para desplazar el robot cuando la ruta de la herramienta no es importante. Su desplazamiento es en ángulos de las partes del robot por segundo º/s.

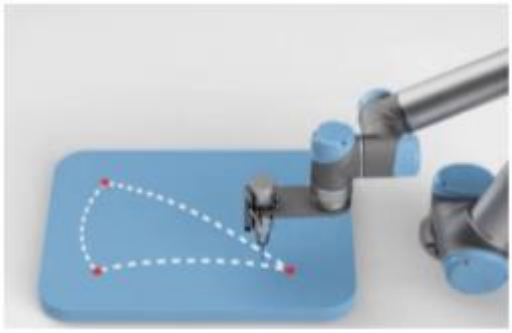


Figura 63 - Movimiento libre en UR

- MoveL: Movimiento lineal. Se utiliza en movimientos donde sea importante la trayectoria de la herramienta.

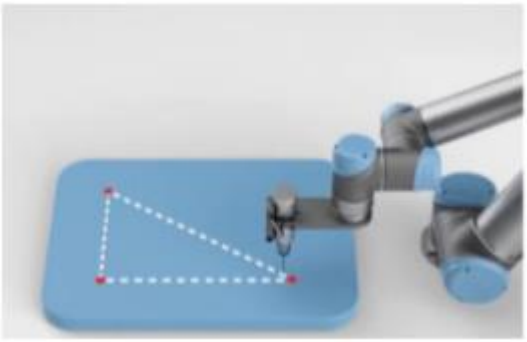


Figura 64 - Movimiento lineal en UR

- MoveP: Movimiento de proceso que mantiene fija la velocidad de la herramienta.

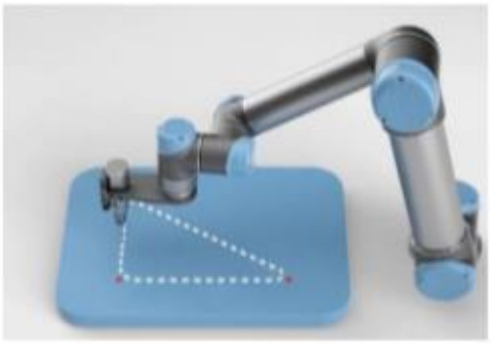


Figura 65 - Movimiento de proceso en UR

Al crear el movimiento pregunta la velocidad que se le desea poner. En el caso de este proyecto y como se estableció en RobotStudio va a ser de 300 para los movimientos libres y de 50 para los lineales.

Al crear un nuevo movimiento, por defecto, se crea una posición en el robot. Esta posición es la que se va a editar punto a punto creando nuevas. Al crear nuevas posiciones el robot va trazando solo las trayectorias punto a punto con el tipo de movimiento que se ha creado.

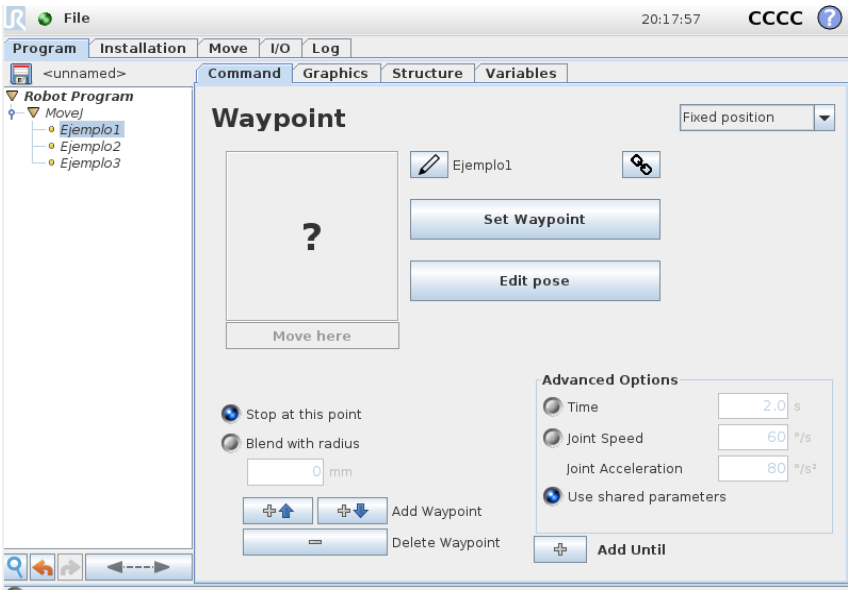


Figura 66 - Creación de posiciones en PolyScope

Para editar las posiciones se puede hacer de varias formas. La primera es dando clic en ‘’Set Waypoint’’.

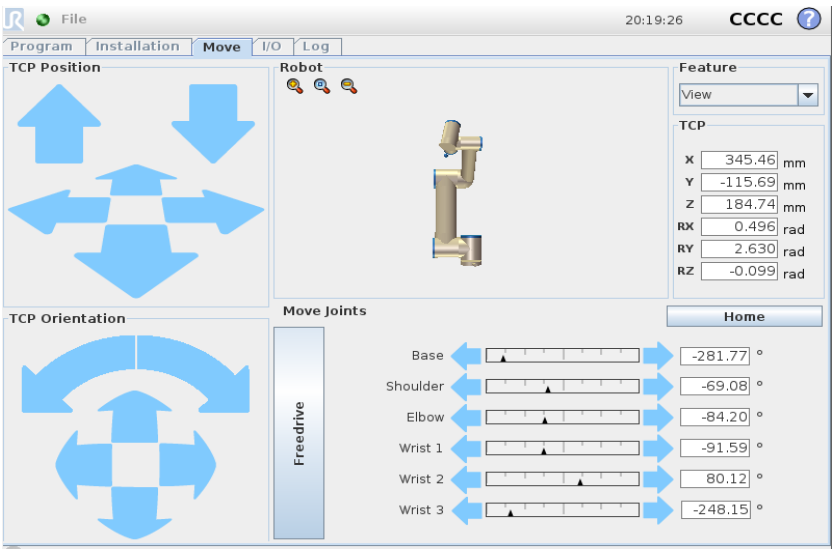


Figura 67 - Primera forma de editar la posición del robot UR5

La primera y más visual es mediante el uso de las flechas. Con las flechas se puede modificar respecto a la herramienta. También se puede modificar eje a eje del robot, con esto podemos mover cada parte del robot por separado y ver el alcance real del robot.

La otra opción es dándole clic a ‘’edit pose’’.

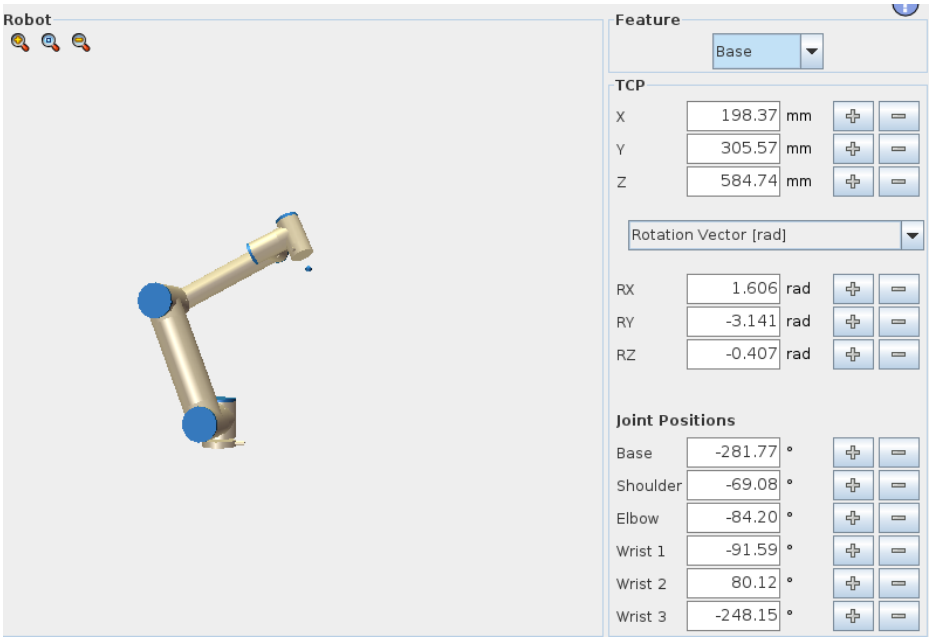


Figura 68 - Segunda forma de editar la posición del robot UR5

Esta opción es la que se ha seguido durante el trabajo. Se trata modificar numéricamente la posición del robot. En este caso se ha hecho con respecto a la herramienta. Ya que sabiendo las medidas de la pieza y la posición de esta es muy sencillo realizar las trayectorias. Se coloca el robot en una posición conocida con respecto a la pieza y para crear las posiciones hay que ir añadiendo la distancia de un punto a otro en coordenadas.

Como se puede comprobar simular el movimiento del robot es sencillo y visual.

5.5 Creación de las posiciones y trayectorias del robot UR5

La primera trayectoria para crear va a ser la posición inicial. Esta consta de un MoveJ con velocidad 150 º/s y de dos puntos. Esta va a ser la posición de inicio y de reposo del robot.

La segunda trayectoria es la del primer fresado. Para realizar esta trayectoria se va a crear un MoveL. Dentro del MoveL se van a crear todas las posiciones del mecanizado. Estas van a ir a una velocidad de 50mm/s. El primer paso va a ser crear las posiciones de bajada, esta va a ser la posición de seguridad previa al ataque de la pieza.

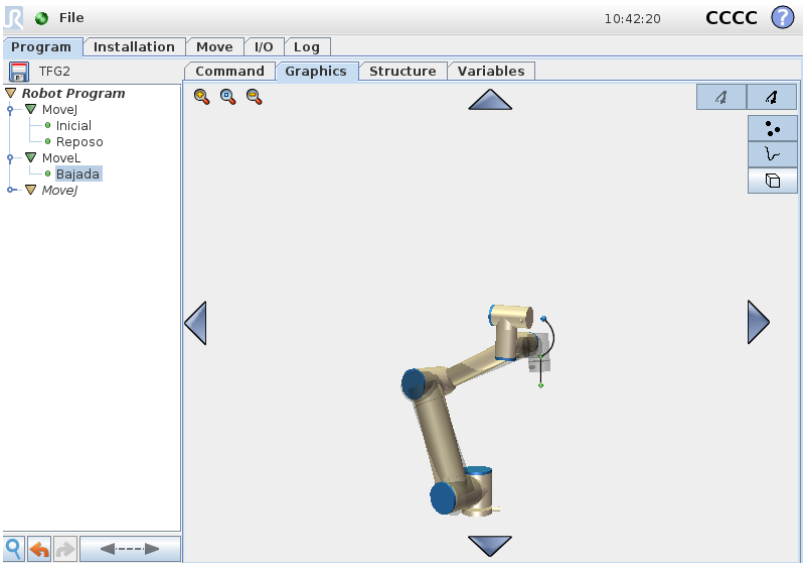


Figura 69 - Primer paso programación en Polyscope

De la anterior figura hay que saber que las posiciones creadas aparecen gráficamente como puntos verdes y las trayectorias que van de un punto a otro son una línea negra.

Una vez estos pasos creados los siguientes son las posiciones de ataque. Para ello sabiendo las medidas de la pieza habrá que ir sumando o restando las medidas en las coordenadas para obtener las posiciones del fresado. Es importante ir actualizando la posición del robot de punto a punto. Una vez se ha creado un punto se le da clic en mover el robot y aparece la siguiente pestaña.

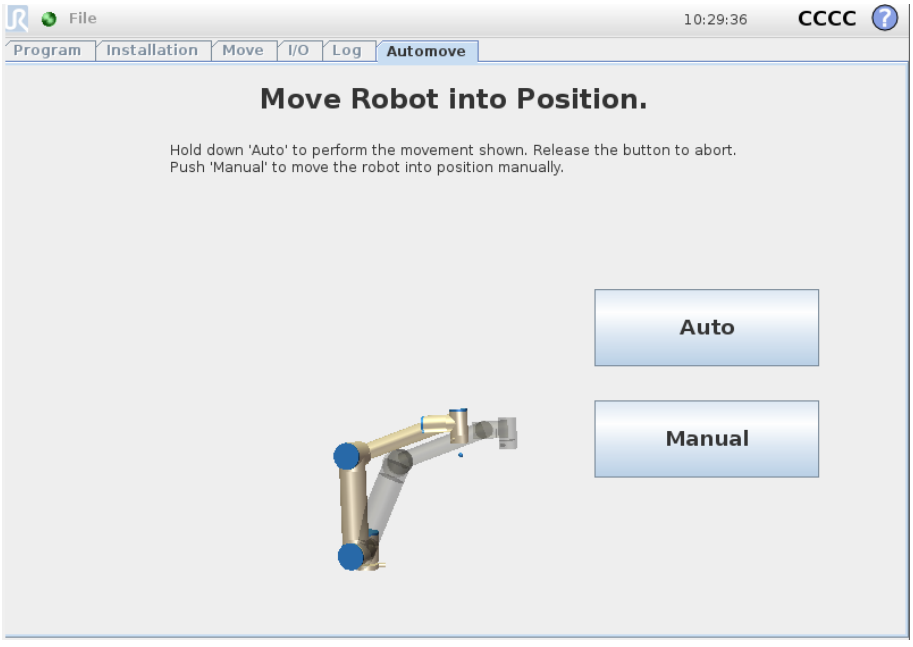


Figura 70 - Actualizar posición del robot UR5

Para actualizarla se deja pulsado el botón de auto hasta que llega al punto. Así conseguimos que al crear la siguiente posición el robot esté en el punto anterior y solo haya que modificar las coordenadas para llegar al siguiente punto. Con se crean todas las trayectorias del primer fresado.

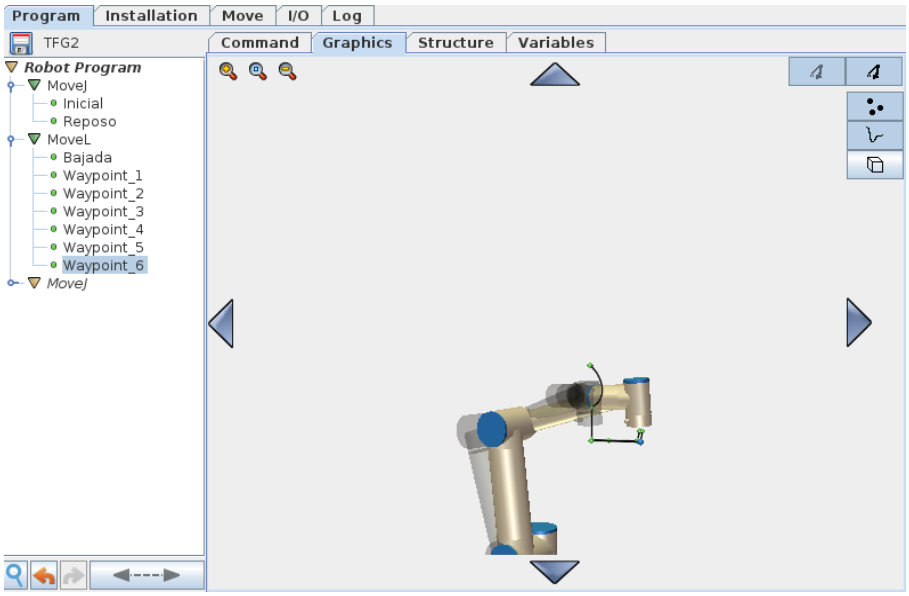


Figura 71 - Primer fresado en PolyScope

Para el segundo fresado habrá que seguir los mismos pasos que en el primero. La primera posición será la de seguridad y las siguientes posiciones serán punto a punto el mecanizado de la pieza sabiendo las coordenadas de la misma. Por último, el último paso será dejar el robot en la posición de reposo ya que el siguiente mecanizado será el de perforación.

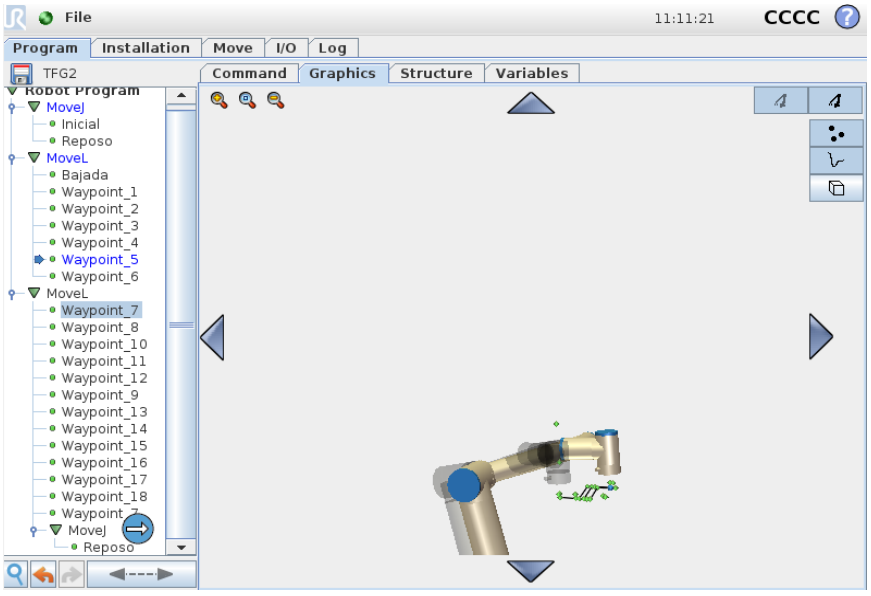


Figura 72 - Segundo Fresado en PolyScope

La siguiente etapa es la de perforación, se va a dividir en dos partes. La primera va a ser una perforación con la fresa sin tener que cambiar la herramienta. La segunda el robot se irá a la posición de reposo con un tiempo de espera para cambiar la herramienta y se realizará el último perforado. En el caso de este trabajo el tiempo de espera va a ser un segundo para que sea más visual, pero se podría implementar que el robot tuviera tiempo de espera hasta el cambio de herramienta.

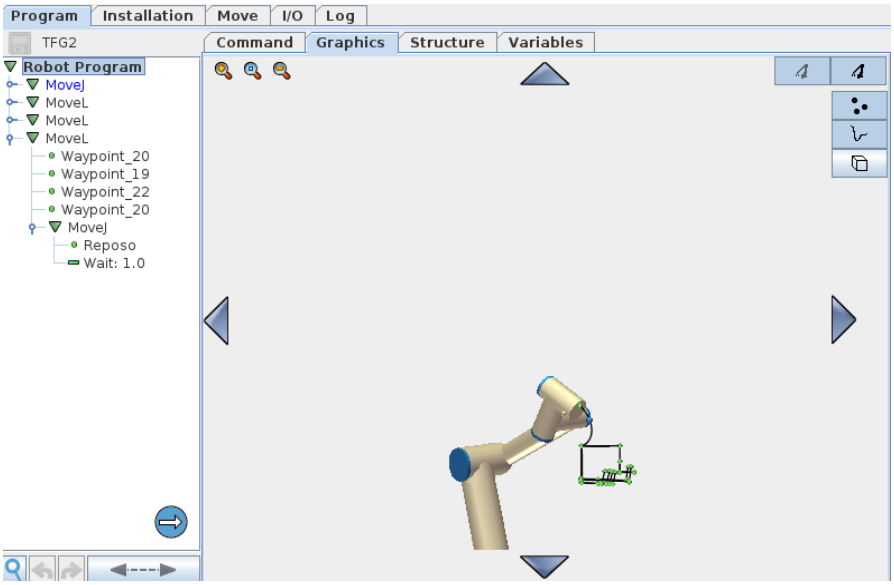


Figura 73 - Primera perforación PolyScope

El último paso del mecanizado son las dos perforaciones restantes que se realizarán con la broca. Después de estas el robot volverá a reposo y posteriormente a la posición inicial.

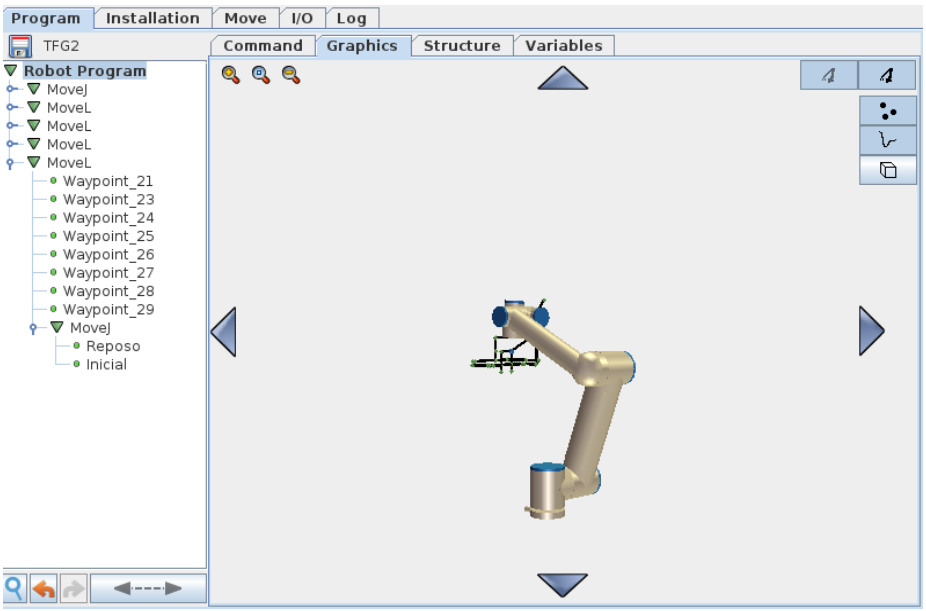


Figura 74 - Segunda perforación en PolyScope

Con esto ya estarían todas las posiciones y trayectorias programadas.

5.6 Simulación en PolyScope

En este programa se han realizado una gran cantidad de puntos, por lo que para que sea más visual se va a realizar la simulación etapa a etapa.

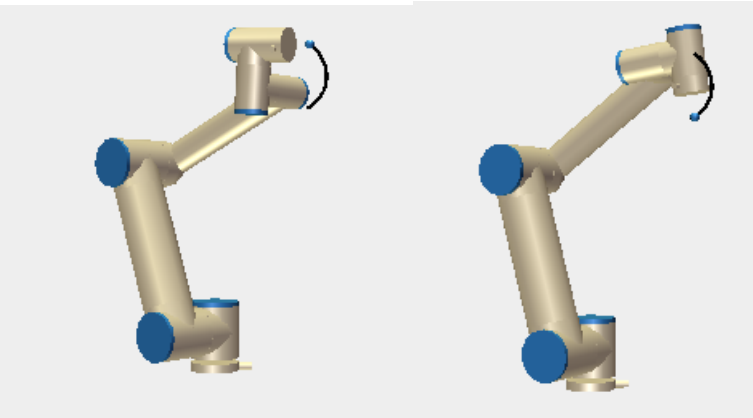


Figura 75- Simulación movimiento inicial PolyScope

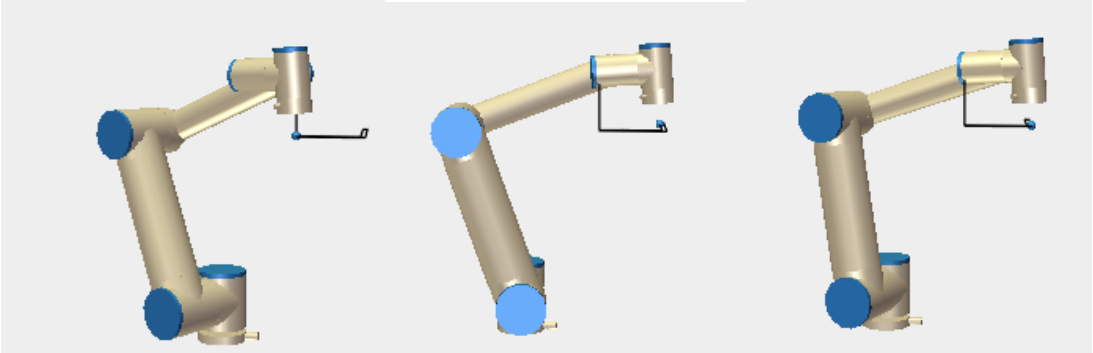


Figura 76 - Simulación primer fresado PolyScope

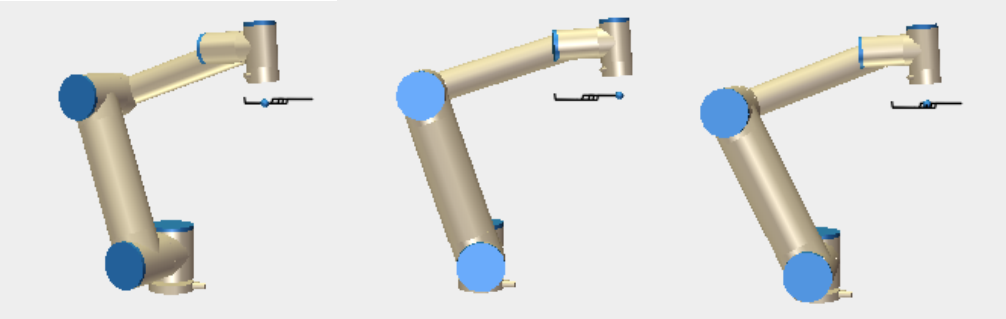


Figura 77 - Simulación segundo fresado Polyscope

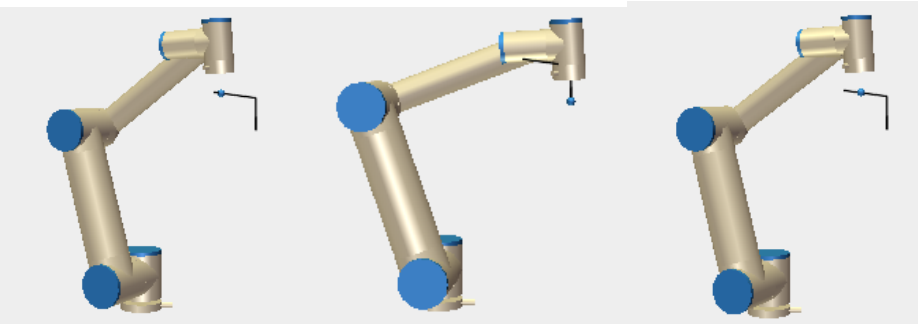


Figura 78 - Simulación primera perforación PolyScope

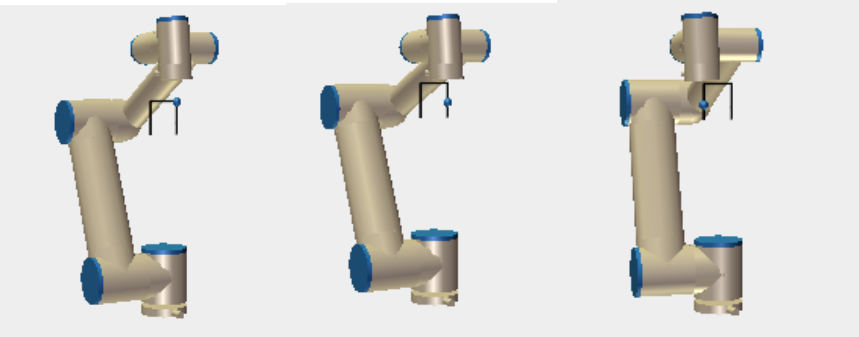


Figura 79 - Simulación segunda perforación PolyScope

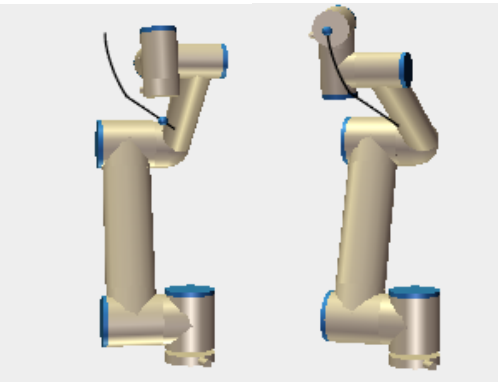


Figura 80 - Simulación movimiento final PolyScope

6. Comparativa entre RobotStudio y PolyScope

En este trabajo se han estudiado desde el principio de la tarea hasta el fin paso a paso cada robot. Por lo que se han visto claras diferencias entre ambos. Estas diferencias se van a plasmar en este punto.

El primer punto de comparaciones se va a realizar de la robótica industrial con la colaborativa.

Eficiencia: Los robots industriales son claramente superiores en este punto. Esto es debido a que la velocidad de movimiento puede llegar a ser hasta cuatro veces superior. No necesitan de la supervisión constante de un operario. Pueden trabajar en entornos más hostiles que los cooperativos.

Flexibilidad: En este punto los robots colaborativos son los más flexibles. Ya que, pueden adaptarse a cualquier trabajo y además pueden realizar operaciones con operarios.

Programación: Los programas de los robots industriales son mucho más complejos que los de los robots colaborativos. De este punto se hablará mejor en la comparativa de ambos entornos.

Almacenamiento e instalación: En cuanto a este punto, los robots industriales necesitan de una zona acotada y vallada para su funcionamiento siendo esta, por lo general, una zona fija. Mientras que los robots colaborativos pueden ser instalados en cualquier zona y el perímetro de seguridad depende de la aplicación, por lo general, no lo necesitan.

Seguridad: En los robots industriales no es un problema porque existe una zona de seguridad. En los colaborativos hay que tener cuidado si el robot transporta piezas o herramientas afiladas.

Peso: Los robots industriales son superiores en este punto porque pueden levantar una cantidad de peso muy superior a los colaborativos.

Precio: Los robots colaborativos tienen un precio más bajo y además, no necesitan componentes adicionales de seguridad por lo que los hacen aún más baratos que la instalación de un industrial.

Esta comparativa deja clara que cada robot es útil para unas aplicaciones determinadas.

El siguiente punto para tratar va a ser la comparativa entre los entornos de RobotStudio y PolyScope. Sabiendo que el primero es el entorno de simulación de robots industriales y el segundo es el entorno de simulación robots colaborativos.

A grandes rasgos el programa de RobotStudio es mucho más complejo que el PolyScope. Tiene la posibilidad de crear un propio entorno dentro del robot con todo tipo de objetos y herramientas, simulando el entorno real del robot. También permite compartir estos objetos y herramientas entre usuarios. Visualmente es mucho más claro de ver una simulación, ya que, se puede ampliar o alejar el zoom. Otro aspecto positivo, es que, nos da la opción de grabar la simulación para verla posteriormente.

PolyScope es una herramienta básica y gratuita para la programación de sus robots, es muy visual. Pero si se requiere de una programación más estricta y con simulación de herramienta y entorno de trabajo, será necesario el uso de otros softwares.

Universal Robots ofrece información sobre ellos en su sección de programas. Uno de estos a mencionar es el RoboDK es un programa de pago pero que habilita el uso de robots de diferentes marcas. Es un software completo que permite el desarrollo al igual que RobotStudio.

Por lo tanto, se puede decir que para la programación del RobotStudio se necesita una cantidad mucho mayor de horas para dominar.

En cuanto a PolyScope, es mucho más sencillo y básico. Se ha visto que la programación de la tarea ha sido más intuitiva que el otro programa. Por lo que, para este, no serían necesarios operarios cualificados.

Los aspectos más relevantes de estos programas a modo de resumen se pueden ver en la siguiente tabla:

RobotStudio	PolyScope
Ventajas: - Programa amplio con multitud de detalles para la programación - Posibilidad de crear estaciones de trabajo. - Simulaciones mucho más realistas. - Posibilidad de compartir objetos entre usuarios. - Posibilidad de modelar las propias herramientas u objetos. - El programa ofrece buena documentación. - Es muy flexible Desventajas: - Curva de aprendizaje lenta. - Para modificar el programa es necesario de un operario con conocimientos. - Es necesaria una licencia para utilizar el programa	Ventajas: - Facilidad a la hora de programar. - Programa muy intuitivo. - Cada robot tiene su propio simulador. - Curva de aprendizaje rápida. - No es necesario un operario cualificado para realizar modificaciones en el programa. - El programa ofrece buena documentación. - Misma interfaz gráfica para todos sus robots - Es un programa gratuito Desventajas: - Poca flexibilidad del programa. - Ofrece pocas opciones visualmente. - No permite crear estaciones de trabajo.

7. Conclusión

Existe una gran cantidad de diferencias entre robots industriales y robots colaborativos, ya que, están pensados para situaciones diferentes por lo general. En consecuencia, sus programas de simulación se acoplan a las necesidades de estos. Por ello se ha observado en la comparación como los robots industriales son robots mas precisos, robustos y estáticos. Y los colaborativos son más flexibles, versátiles y fáciles de programar.

En este trabajo mediante la simulación de una tarea mecanizada se ha plasmado cada diferencia entre ambos entornos de programación. Ha quedado claro que RobotStudio es un programa muy completo y con un gran abanico de herramientas. Y que PolyScope es un programa muy simple, pero que cubre las necesidades que se le demandan.

En resumen, cada programa ofrece ventajas distintas al igual que cada tipo de robot. Por un lado, tenemos que para trabajar en RobotStudio se necesita de un operario con experiencia, pero, ofrece muchas mas utilidades. Y por otro lado que, para trabajar en PolyScope no es necesaria una gran experiencia, pero, las herramientas son muy reducidas.

8. Bibliografía

Manuales

ABB. Manual de instalación de RobotStudio. https://library.e.abb.com/public/5240940e78c27430c1257b67004c6759/3HAC032104-es.pdf (Ultima visita 25/06/2020)

Universal Robots. Manual de PolyScope. https://s3-eu-west-1.amazonaws.com/ur-support-site/16267/Software_Manual_es_Global.pdf (Ultima visita 25/06/2020)

Webs

https://blog.universal-robots.com/es/cobots-vs-robots-industriales (Ultima visita 25/06/2020)

https://www.femeval.es/dam/jcr:fd091e5f-3c97-42fd-9981-680e8232a645/GUIA_ROBOTS.pdf (Ultima visita 25/06/2020)

https://www.interempresas.net/Flipbooks/XM/51/pdf/Libro%201_XM51.pdf (Ultima visita 25/06/2020)

http://www.udesantiagovirtual.cl/moodle2/mod/book/view.php?id=24892&chapterid=178#:~:text=El%20pionero%20en%20la%20rob%C3%B3tica,el%20germen%20del%20robot%20industrial. (Ultima visita 25/06/2020)

https://digital.csic.es/bitstream/10261/12832/1/Historia_robotica.pdf (Ultima visita 25/06/2020)

https://ticnegocios.camaravalencia.com/servicios/tendencias/caminar-con-exito-hacia-la-industria-4-0-capitulo-16-robots/ (Ultima visita 25/06/2020)

http://www.etitudela.com/profesores/rpm/rpm/downloads/robotica.pdf (Ultima visita 25/06/2020)

https://www.cfzcobots.com/wp-content/uploads/2017/03/ur5_user_manual_es_global.pdf (Ultima visita 25/06/2020)

https://movicontrol.es/robots-colaborativos/ (Ultima visita 25/06/2020)

https://www.maqcenter.com/es/news/robots-vs-cobots-ventajas-y-desventajas.html (Ultima visita 25/06/2020)

https://www.gruasyaparejos.com/gruas-industriales/fabricantes-de-robots-industriales/ (Ultima visita 25/06/2020)

Apuntes

Apuntes y documentación extraído de la asignatura de Sistemas Robotizados. Impartida en el Grado de Ingeniera Eléctrica de la ETSID Universidad Politécnica de Valencia.

Normativa

https://www.boe.es/buscar/act.php?id=BOE-A-1997-8671 (Ultima visita 25/06/2020)

9. Anexos

9.1 Código de programación en RobotStudio

```
MODULE Module1

    CONST robtarget Target_30:=[[587.647988537,-
64.22966599,335.014078007],[0.002591202,-0.693148013,-
0.720790155,0.000818723],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Target_40:=[[637.647988537,-
64.22966599,335.014078007],[0.012188829,-0.702837111,-
0.711168295,0.010539648],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Target_100:=[[732.055424258,-
64.229659482,335.014080153],[0.012188707,-0.702837101,-
0.711168308,0.010539559],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget
Target_110:=[[732.055417061,158.358598787,335.014063538],[0.012188754,-
0.702837118,-0.711168291,0.010539505],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget
Target_120:=[[744.101512816,158.358612828,335.014075497],[0.012188775,-
0.702837127,-0.711168282,0.010539491],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Target_130:=[[744.101527243,-
74.767755267,335.014072055],[0.012188758,-0.702837118,-
0.711168292,0.010539457],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Target_140:=[[637.532473097,-
74.767770554,335.014076561],[0.012188743,-0.702837107,-
0.711168303,0.010539477],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Target_150:=[[587.532473097,-
74.767770554,335.014076561],[0.002591216,-0.693148054,-
0.720790115,0.000818719],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget
Target_480:=[[500.062776666,236.946876962,663.704726157],[0.464094781,-
0.818558833,-0.182476859,-0.285095889],[0,-
1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Target_170:=[[584.513024889,-
43.863998205,320.236484386],[0.002591216,-0.693148054,-
0.720790115,0.000818719],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Target_180:=[[634.513024889,-
43.863998205,320.236484386],[0.002591102,-0.693148081,-
0.72079009,0.000818577],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
CONST robtarget Target_190:=[[687.020008417,-
43.864012755,320.23648785],[0.002591046,-0.693148084,-
0.720790087,0.000818558],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget
Target_200:=[[687.020011094,131.165197513,320.236492496],[0.002591026,-
0.693148073,-0.720790098,0.000818534],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget
Target_220:=[[759.70978314,130.996096829,320.236489894],[0.002590983,-
0.693148085,-0.720790087,0.000818487],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget
Target_230:=[[759.709762669,142.083505441,320.236491398],[0.00259103,-
0.693148091,-0.720790081,0.00081847],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget
Target_240:=[[656.223478448,142.083500396,320.236482615],[0.002590993,-
0.693148087,-0.720790084,0.00081844],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Target_250:=[[656.223478264,-
43.88418692,320.236493283],[0.00259095,-0.693148073,-
0.720790098,0.000818391],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Target_260:=[[677.656108463,-
43.884191125,320.236485958],[0.00259096,-0.693148083,-
0.720790089,0.000818385],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget
Target_270:=[[677.656111894,139.105168593,320.236479084],[0.002590918,-
0.693148062,-0.720790109,0.000818424],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget
Target_540:=[[667.9976023,139.105168593,320.236491544],[0.002590856,-
0.693148094,-0.720790079,0.00081839],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Target_280:=[[667.9976023,-
43.714388264,320.236487735],[0.002590845,-0.693148074,-
0.720790098,0.000818448],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Target_530:=[[635.439904981,-
184.255090506,550.842628918],[0.25604369,-0.941170488,-0.073278065,-
0.208014581],[0,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget
Target_330:=[[657.290218249,81.671955207,374.489277127],[0.504245316,-
0.519951891,0.489374995,-0.485694149],[0,-
1,3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
        CONST robtarget
Target_350:=[[657.290216504,81.671943427,302.350486088],[0.504245372,-
0.519951843,0.489374893,-0.485694246],[0,-
1,3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

        CONST robtarget
Target_390:=[[657.290219252,15.069998377,374.489277127],[0.504245605,-
0.519951719,0.489374756,-
0.485694274],[0,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

        CONST robtarget
Target_380:=[[657.290214474,15.069962675,296.477190074],[0.504245612,-
0.519951791,0.489374663,-
0.485694284],[0,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

        CONST robtarget Target_520:=[[635.439904981,-
184.255090506,550.842628918],[0.25604369,-0.941170488,-0.073278065,-
0.208014581],[0,-1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

        CONST robtarget
Target_570:=[[623.636589441,248.554704595,543.989275039],[0.4909121,-
0.842831862,-0.124619309,-0.181960955],[0,-
1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

        CONST robtarget
Target_580:=[[710.345983502,45.996322424,396.749800076],[0.00259111,-
0.693148073,-0.720790098,0.0008185],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

        CONST robtarget
Target_430:=[[710.345990766,45.99632593,364.363181393],[0.002591152,-
0.693148081,-0.72079009,0.000818512],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

        CONST robtarget
Target_440:=[[710.345983914,45.99633992,334.541770557],[0.002591158,-
0.693148089,-0.720790082,0.000818468],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget
Target_340:=[[657.290214381,81.671891219,324.489277127],[0.504245299,-
0.519951949,0.489374858,-0.485694243],[0,-
1,3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget
Target_340_2:=[[657.290214381,81.671891219,324.489277127],[0.504245299,-
0.519951949,0.489374858,-0.485694243],[0,-
1,3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget
Target_370:=[[657.290217954,15.069917935,325.212583422],[0.504245586,-
0.519951882,0.489374572,-
0.485694305],[0,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget
Target_370_2:=[[657.290217954,15.069917935,325.212583422],[0.504245586,-
0.519951882,0.489374572,-
0.485694305],[0,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget
Target_590:=[[710.345989931,45.99634531,346.749800076],[0.002591151,-
0.693148082,-0.720790089,0.000818464],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget
Target_590_2:=[[710.345989931,45.99634531,346.749800076],[0.002591151,-
0.693148082,-0.720790089,0.000818464],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

        PROC Fresa1()

            MoveJ Target_570,v300,z50,FRESA\WObj:=Fresado1;

      WaitTime 1;

      MoveJ Target_30,v300,z50,FRESA\WObj:=Fresado1;

            MoveL Target_40,v50,z0,FRESA\WObj:=Fresado1;

            WaitTime 1;

            MoveL Target_100,v50,z0,FRESA\WObj:=Fresado1;

            WaitTime 0.5;

            MoveL Target_110,v50,z0,FRESA\WObj:=Fresado1;

            WaitTime 0.5;

            MoveL Target_120,v50,z0,FRESA\WObj:=Fresado1;

            WaitTime 0.5;

            MoveL Target_130,v50,z0,FRESA\WObj:=Fresado1;

            WaitTime 0.5;

            MoveL Target_140,v50,z0,FRESA\WObj:=Fresado1;

            MoveL Target_150,v50,z20,FRESA\WObj:=Fresado1;

        ENDPROC

        PROC Inicio()

            MoveJ Target_480,v300,z50,FRESA\WObj:=Inicial;

        ENDPROC

        PROC Fresa2()

            MoveL Target_170,v50,z20,FRESA\WObj:=Fresado2;

            MoveL Target_180,v50,z0,FRESA\WObj:=Fresado2;

            WaitTime 1;
```

```
        MoveL Target_190,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_200,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_220,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_230,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_240,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_250,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_260,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_270,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_540,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_280,v50,z0,FRESA\WObj:=Fresado2;
        WaitTime 0.5;
        MoveL Target_170,v50,z20,FRESA\WObj:=Fresado2;
    ENDPROC
    PROC Perforado1()
        MoveJ Target_530,v300,z50,BROCA\WObj:=Perforacion1;
        WaitTime 1;
        MoveJ Target_330,v300,z50,BROCA\WObj:=Perforacion1;
MoveL Target_340,v50,z0,BROCA\WObj:=Perforacion1;
        WaitTime 0.5;
        MoveL Target_350,v50,z0,BROCA\WObj:=Perforacion1;
        MoveL Target_340_2,v50,z0,BROCA\WObj:=Perforacion1;
        MoveL Target_330,v50,z20,BROCA\WObj:=Perforacion1;
        MoveL Target_390,v50,z20,BROCA\WObj:=Perforacion1;
```

```
            MoveL Target_370,v50,z0,BROCA\WObj:=Perforacion1;
            WaitTime 0.5;
            MoveL Target_380,v50,z0,BROCA\WObj:=Perforacion1;
            MoveL Target_370_2,v50,z0,BROCA\WObj:=Perforacion1;
            MoveL Target_390,v50,z20,BROCA\WObj:=Perforacion1;
        ENDPROC
        PROC Perforado2()
            MoveJ Target_570,v300,z50,FRESA\WObj:=Perforacion2;
            WaitTime 1;
            MoveJ Target_580,v300,z50,FRESA\WObj:=Perforacion2;
            MoveL Target_590,v50,z0FRESA\WObj:=Perforacion2;
            WaitTime 0.5;
            MoveL Target_440,v50,z0,FRESA\WObj:=Perforacion2;
            MoveL Target_590_2,v50,z0,FRESA\WObj:=Perforacion2;
            MoveL Target_580,v50,z20,FRESA\WObj:=Perforacion2;
            MoveL Target_570,v300,z50,FRESA\WObj:=Perforacion2;
        ENDPROC
        PROC Main()
    Inicio;
    Fresa1;
    Fresa2;
    Perforado1;
    Perforado2;
    Inicio;
        ENDPROC
ENDMODULE
```

9.2 Código de programación en PolyScope

```
  set_tcp(p[0.0,0.0,0.05,0.0,0.0,0.0])
  set_payload(1.0)
  set_standard_analog_input_domain(0, 1)
  set_standard_analog_input_domain(1, 1)
  set_tool_analog_input_domain(0, 1)
  set_tool_analog_input_domain(1, 1)
  set_analog_outputdomain(0, 0)
  set_analog_outputdomain(1, 0)
  set_input_actions_to_default()
  set_tool_voltage(0)
  set_safety_mode_transition_hardness(1)
  set_gravity([0.0, 0.0, 9.82])
  step_count_bc1b46bc_ded3_45cb_a0c9_53c70818dd19 = 0
  thread Step_Counter_Thread_b2384d49_270b_4885_bbba_95d19d3b8d00():
    while (True):
      step_count_bc1b46bc_ded3_45cb_a0c9_53c70818dd19 =
step_count_bc1b46bc_ded3_45cb_a0c9_53c70818dd19 + 1
      sync()
    end
  end
  run Step_Counter_Thread_b2384d49_270b_4885_bbba_95d19d3b8d00()
  while (True):
    $ 1 "Robot Program"
    $ 2 "MoveJ"
    $ 3 "Inicial" "breakAfter"
    movej(get_inverse_kin(p[.300000000000, .200000000000, .730000000000, -
.591891555284, 1.641935329376, 1.022512723838], qnear=[-5.427750029502209, -
1.3185923205625816, -1.4563820615955425, -0.4373538167403339,
1.740147251131222, -4.391095747755855]), a=1.3962634015954636,
v=2.6179938779914944)
    $ 4 "Reposo" "breakAfter"
    movej(get_inverse_kin(p[.249997444455, .200000000000, .599997727455,
.972717388202, -2.982595118209, -.111026351948], qnear=[-5.285207125160802, -
```

1.2974765816165785, -1.2190250204576492, -2.153085746316947, 1.6272013620315713, -4.343712599003457]), a=1.3962634015954636, v=2.6179938779914944)

$ 5 "MoveL"

$ 6 "Bajada" "breakAfter"

movel(p[.249997436789, .200000000000, .499997688159, .972717442871, -2.982595073588, -.111026388797], a=1.2, v=0.05)

$ 7 "Waypoint_1" "breakAfter"

movel(p[.249997396826, .250000000000, .499997706840, .972717536998, -2.982595218958, -.111026707785], a=1.2, v=0.05)

$ 8 "Waypoint_2" "breakAfter"

movel(p[.249997396826, .335000000000, .499997706840, .972717536998, -2.982595218958, -.111026707785], a=1.2, v=0.05)

$ 9 "Waypoint_3" "breakAfter"

movel(p[.045000000000, .335000000000, .499997706840, .972717536998, -2.982595218958, -.111026707785], a=1.2, v=0.05)

$ 10 "Waypoint_4" "breakAfter"

movel(p[.044999973141, .345000000000, .499997612175, .972717465892, -2.982595246760, -.111026076953], a=1.2, v=0.05)

$ 11 "Waypoint_5" "breakAfter"

movel(p[.265000000000, .344999971258, .499997673176, .972717391327, -2.982595304616, -.111026523398], a=1.2, v=0.05)

$ 12 "Waypoint_6" "breakAfter"

movel(p[.265000022280, .200000000000, .499997504679, .972717349797, -2.982595096143, -.111026188312], a=1.2, v=0.05)

$ 13 "MoveL"

$ 14 "Waypoint_7" "breakAfter"

movel(p[.240000000000, .200000021267, .484997387299, .972717252151, -2.982594899498, -.111025925988], a=1.2, v=0.05)

$ 15 "Waypoint_8" "breakAfter"

movel(p[.240000004144, .250000000000, .484997494992, .972717414002, -2.982595114880, -.111026351229], a=1.2, v=0.05)

$ 16 "Waypoint_10" "breakAfter"

movel(p[.240000004144, .295000000000, .484997494992, .972717414002, -2.982595114880, -.111026351229], a=1.2, v=0.05)

$ 17 "Waypoint_11" "breakAfter"

movel(p[.065000000000, .294999913837, .484997438610, .972717479253, -2.982595001183, -.111026075386], a=1.2, v=0.05)

$ 18 "Waypoint_12" "breakAfter"

movel(p[.065000000000, .355000000000, .484997438610, .972717479253, -2.982595001183, -.111026075386], a=1.2, v=0.05)

$ 19 "Waypoint_9" "breakAfter"

movel(p[.055000000000, .355000000000, .484997438610, .972717479253, -2.982595001183, -.111026075387], a=1.2, v=0.05)

$ 20 "Waypoint_13" "breakAfter"

movel(p[.055000001097, .260000000000, .484997389067, .972717406052, -2.982595020123, -.111025729897], a=1.2, v=0.05)

$ 21 "Waypoint_14" "breakAfter"

movel(p[.240000004144, .260000000000, .484997494992, .972717414002, -2.982595114880, -.111026351229], a=1.2, v=0.05)

$ 22 "Waypoint_15" "breakAfter"

movel(p[.240000014399, .274999985514, .484997538203, .972717423465, -2.982595115039, -.111026242029], a=1.2, v=0.05)

$ 23 "Waypoint_16" "breakAfter"

movel(p[.055000014399, .274999985514, .484997538203, .972717423465, -2.982595115039, -.111026242029], a=1.2, v=0.05)

$ 24 "Waypoint_17" "breakAfter"

movel(p[.055000013058, .284999994607, .484997465511, .972717579809, -2.982595063883, -.111025754204], a=1.2, v=0.05)

$ 25 "Waypoint_18" "breakAfter"

movel(p[.240000013058, .284999994607, .484997465511, .972717579809, -2.982595063883, -.111025754204], a=1.2, v=0.05)

$ 26 "Waypoint_7" "breakAfter"

movel(p[.240000000000, .200000021267, .484997387299, .972717252151, -2.982594899498, -.111025925988], a=1.2, v=0.05)

$ 27 "MoveJ"

$ 28 "Reposo" "breakAfter"

movej(get_inverse_kin(p[.249997444455, .200000000000, .599997727455, .972717388202, -2.982595118209, -.111026351948], qnear=[-5.285207125160802, -1.2974765816165785, -1.2190250204576492, -2.153085746316947, 1.6272013620315713, -4.343712599003457]), a=1.3962634015954636, v=2.6179938779914944)

$ 29 "MoveL"

$ 30 "Waypoint_20" "breakAfter"

movel(p[.142001510306, .312000296999, .585000474872, .972843599783, -2.982568302989, -.106856129496], a=1.2, v=0.05)

$ 31 "Waypoint_19" "breakAfter"

movel(p[.141998974269, .311999265155, .534997645294, .973092815110, -2.983251895107, -.105951057151], a=1.2, v=0.05)

$ 32 "Waypoint_22" "breakAfter"

movel(p[.142001510306, .312000296999, .500000474872, .972843599783, -2.982568302989, -.106856129496], a=1.2, v=0.05)

$ 33 "Waypoint_20" "breakAfter"

movel(p[.142001510306, .312000296999, .585000474872, .972843599783, -2.982568302989, -.106856129496], a=1.2, v=0.05, r=0.05)

$ 34 "MoveJ"

$ 35 "Reposo" "breakAfter"

movej(get_inverse_kin(p[.249997444455, .200000000000, .599997727455, .972717388202, -2.982595118209, -.111026351948], qnear=[-5.285207125160802, -1.2974765816165785, -1.2190250204576492, -2.153085746316947, 1.6272013620315713, -4.343712599003457]), a=1.3962634015954636, v=1.0471975511965976)

$ 36 "Wait: 1.0"

sleep(1.0)

$ 37 "MoveL"

$ 38 "Waypoint_21" "breakAfter"

movel(p[.108998970806, .264999262439, .534997633415, .973092744637, -2.983251812548, -.105950762377], a=1.2, v=0.05)

$ 39 "Waypoint_23" "breakAfter"

movel(p[.108998986164, .264999329809, .484997510778, .973092596586, -2.983251775924, -.105950335256], a=1.2, v=0.05)

$ 40 "Waypoint_24" "breakAfter"

movel(p[.108998986164, .264999329809, .459997510778, .973092596586, -2.983251775924, -.105950335256], a=1.2, v=0.05)

$ 41 "Waypoint_25" "breakAfter"

movel(p[.108998986164, .264999329809, .534997510778, .973092596586, -2.983251775925, -.105950335256], a=1.2, v=0.05)

$ 42 "Waypoint_26" "breakAfter"

movel(p[.174998986164, .264999329809, .534997510778, .973092596586, -2.983251775924, -.105950335256], a=1.2, v=0.05)

$ 43 "Waypoint_27" "breakAfter"

movel(p[.174998986697, .264999371966, .484997426874, .973092624306, -2.983251707113, -.105950157338], a=1.2, v=0.05)

$ 44 "Waypoint_28" "breakAfter"

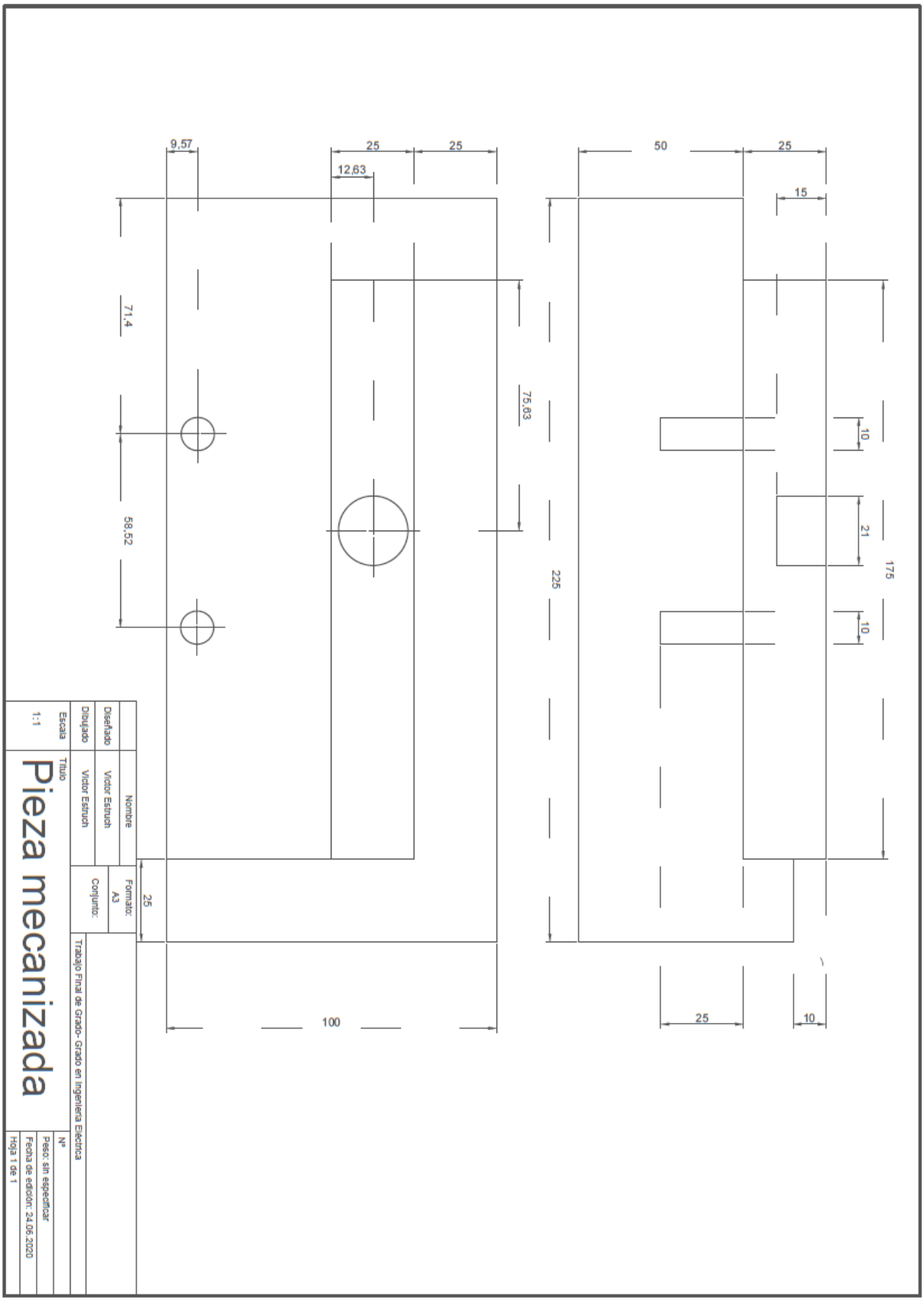
movel(p[.174998986697, .264999371966, .459997426874, .973092624306, -2.983251707113, -.105950157338], a=1.2, v=0.05)

$ 45 "Waypoint_29" "breakAfter"

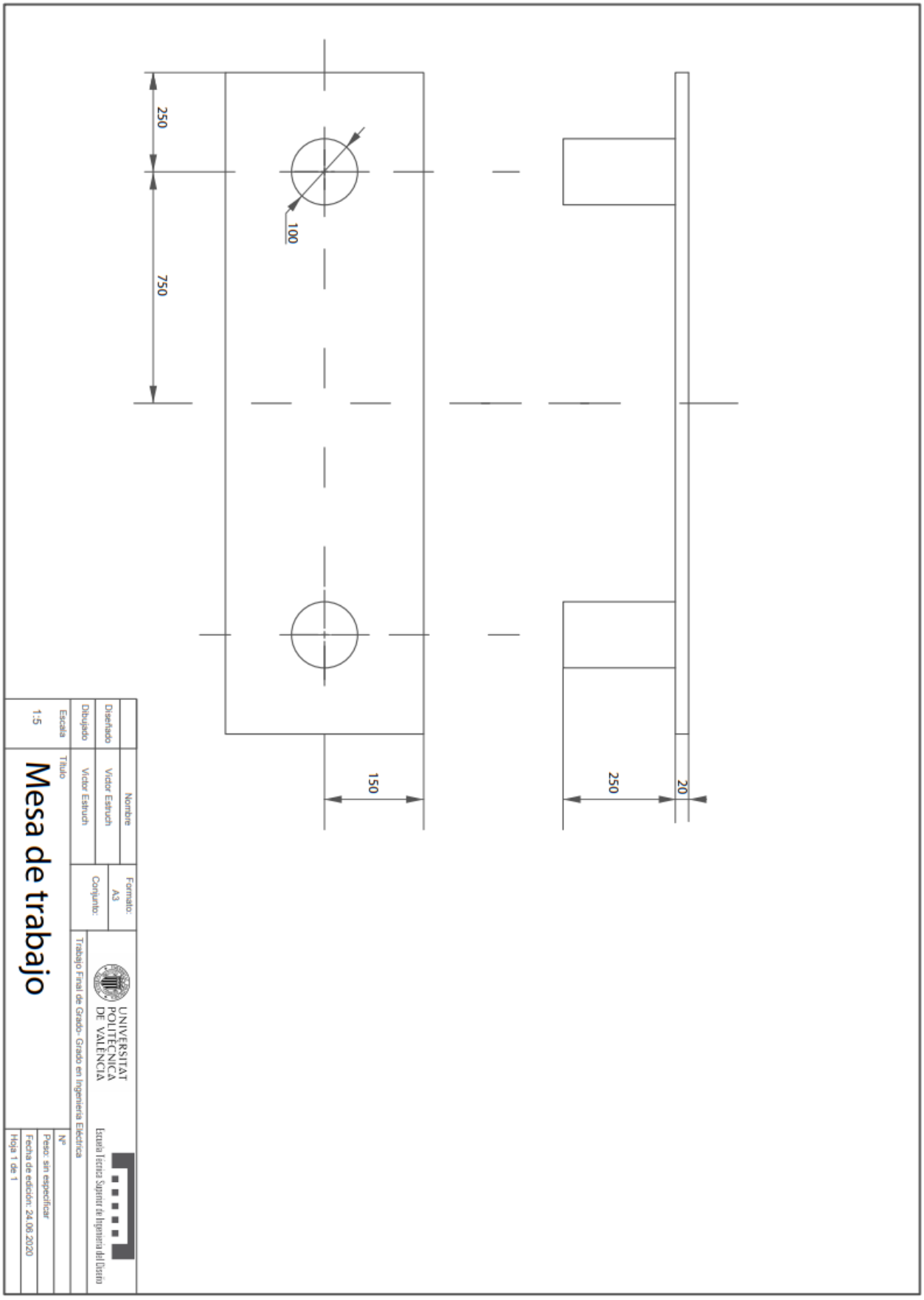
DOCUMENTO II

PLANOS

Plano 1: Pieza Mecanizada



Plano 2: Mesa de trabajo



DOCUMENTO III

PLIEGO DE CONDICIONES

1. Objeto

El objetivo de este pliego de condiciones es establecer las obligaciones que se deben cumplir por las partes contratantes durante la realización de este Trabajo Fin de Grado: Programación de una tarea robotizada mediante diferentes entornos de trabajo y comparativa.

Este documento se aplica a todos los equipos eléctricos que forman parte de este proyecto.

2. Condiciones de los materiales

En el caso de los materiales necesarios para este trabajo están constituidos por un ordenador portátil constituido por pantalla, teclado y ratón. De accesorios ofimáticos y de un asiento y superficie de trabajo.

Todos estos materiales tienen que estar regidos por el **REAL DECRETO 488/1997**, de prevención de riesgos laborales, que determinan las garantías y responsabilidades precisas para establecer un adecuado nivel de protección a los trabajadores frente a riesgos derivados de las condiciones de trabajo. En este proyecto tiene que ver sobre distancias mínimas de seguridad y salud frente a ordenadores.

Descripción de los equipos

1. **Hardware**
   Portátil MSI CX62 6QD-241XES

   Procesador: Intel i7-6700HQ

   RAM: 8 GB

2. **Software**

- Windows 10
- Microsoft Robotics Studio
- Interfaz de programación PolyScope de Universal Robots
- Microsoft Office 2010

Normativa

Regido por el **REAL DECRETO 488/1997.**

1. **Pantalla**

- Los caracteres deberán estar bien definidos y configurados de forma clara, y tener una dimensión suficiente, disponiendo de un espacio adecuado entre los caracteres y los reglones.
- La imagen de la pantalla deberá ser estable, sin fenómenos de destello, centelleos u otras formas de inestabilidad
- El usuario de terminales con pantalla deberá poder ajustar fácilmente la luminosidad y el contraste entre los caracteres y el fondo de la pantalla, y adaptarlos fácilmente a las condiciones de entorno.
- La pantalla deberá ser orientable e inclinable a voluntad, con facilidad para adaptarse a las necesidades del usurario.
- Podrá utilizarse un pedestal independiente o una mesa regulable para la pantalla.

- La pantalla no deberá tener reflejos ni reverberaciones que puedan molestar al usuario.

2. Teclado

- El teclado deberá ser inclinable e independiente de la pantalla para permitir que el trabajador adopte una postura cómoda que no provoque cansancio en los brazos o las manos
- Tendrá que haber espacio suficiente delante del teclado para que el usuario pueda apoyar los brazos y las manos.
- La superficie del teclado deberá ser mate para evitar reflejos.
- La disposición del teclado y las características de las teclas deberán tender a facilitar su utilización
- Los símbolos de las teclas deberán resaltar suficientemente y ser legibles desde la posición normal del trabajo.

3. Mesa o superficie de trabajo

- La mesa o superficie de trabajo deberán ser poco reflectantes, tener dimensiones suficientes y permitir una colocación flexible de la pantalla, del teclado, de los documentos y del material accesorio
- El soporte de los documentos deberá ser estable y regulable y estará colocado de tal modo que se reduzcan al mínimo los movimientos incómodos de la cabeza y los ojos
- El espacio deberá ser suficiente para permitir a los trabajadores una posición cómoda.

4. Asiento de trabajo

- El asiento de trabajo deberá ser estable, proporcionando al usuario libertad de movimientos y procurándole una postura confortable.
- La altura del mismo deberá ser regulable.
- El respaldo deberá ser reclinable y su altura ajustable
- Se pondrá un reposapiés a disposición de quienes lo deseen.

DOCUMENTO IV

PRESUPUESTO

En el siguiente documento se van a analizar los costes de la realización de este proyecto. Estos costes se van a desglosar en los siguientes:

- Costes de amortización del material empleado
- Costes de mano de obra
- Coste total sin IVA y con IVA

1. Coste de amortización del material empleado

Este coste hace referencia a la relación de tiempo que se ha empleado en un dispositivo o programa durante el proyecto con el tiempo de vida total. En este trabajo, este coste va a aparecer en todos los equipos instalados que han permitido el desarrollo del proyecto. El coeficiente aplicado ha sido el siguiente:

$$Coef.Amoritización = \frac{Tiempo\ del\ proyecto}{Tiempo\ de\ amortización}$$

El tiempo de este proyecto ha sido de 90 días y se ha considerado el tiempo de amortización de los dispositivos utilizados de 4 años. La licencia de RobotStudio es gratuita durante 2020 y la de PolyScope es gratuita indefinidamente.

CONCEPTO	DESCRIPCIÓN	FABRICANTE	PRECIO (€)	COEFICIENTE AMORTIZACIÓN	TOTAL(€)
Hardware	Ordenador Portatil	MSI	730	0,06	43,8
Software	Microsoft Windows 10	Microsoft	50	0,06	3
Software	Office Profesional 2010	Microsoft	90	0,06	5,4
Software	Oracle VM VirtualBox	VirtualBox	-----	-----	
Software	RobotStudio	ABB	-----	-----	-----
Software	PolyScope	Universal Robots	-----	-----	-----
COSTE TOTAL DE AMORTIZACIÓN DEL MATERIAL EMPLEADO					52,2

2. Coste de Mano de Obra

En este punto se van a detallar todos los costes de organización, investigación, diseño y desarrollo del software de este proyecto.

ORGANIZACIÓN DEL PROYECTO			
TAREA	HORAS	PRECIO (h)	TOTAL (€)
Planificación	16	25	400
Investigación y documentación	50	25	1250
DISEÑO			
Diseño de piezas en RobotStudio	4	25	100
Desarrollo del Software en RobotStudio	60	25	1500
Desarrollo del Software en PolyScope	25	25	625
Simulación en ambos Softwares	1	25	25
COSTE TOTAL DE MANO DE OBRA			3900

3. Coste Total

Hasta el momento, todos los precios indicados han sido sin IVA. El ultimo coste van a ser los precios sumados sin IVA y con IVA.

CONCEPTO	COSTE (€)
Coste de amortización del material empleado	52,2
Coste de mano de obra	3900
Total sin IVA	3952,2
IVA (21%)	829,96
Coste total del proyecto	4782,16