UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE MECATRÓNICA

DESARROLLO DE UN SISTEMA TRADUCTOR DE CÓDIGO G A KRL PARA LA ADAPTACIÓN DE ROBOTS KUKA KR16-2 EN ENTORNOS DE MECANIZADO CNC.

Trabajo de titulación previo a la obtención
del título de Ingeniero en Mecatrónica

AUTOR: ROBERT SEBASTIÁN MURILLO AYORA

TUTOR: ING. PAUL ANDRES CHASI PESANTEZ M.SC.

Cuenca – Ecuador

2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Robert Sebastián Murillo Ayora con documento de identificación N° 0107184418 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 9 de septiembre del 2024

Atentamente,

Robert Sebastián Murillo Ayora
0107184418

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Robert Sebastián Murillo Ayora con documento de identificación N° 0107184418, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto Técnico: "Desarrollo de un sistema traductor de código G a KRL para la adaptación de robots KUKA KR16-2 en entornos de mecanizado CNC.", el cual ha sido desarrollado para optar por el título de: Ingeniero en Mecatrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 9 de septiembre del 2024

Atentamente,

Robert Sebastián Murillo Ayora
0107184418

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Paúl Andrés Chasi Pesantez con documento de identificación N° 0103652095, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UN SISTEMA TRADUCTOR DE CÓDIGO G A KRL PARA LA ADAPTACIÓN DE ROBOTS KUKA KR16-2 EN ENTORNOS DE MECANIZADO CNC, realizado por Robert Sebastián Murillo Ayora con documento de identificación N° 0107184418, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 9 de septiembre del 2024

Atentamente,

Paúl Andrés Chasi Pesantez
0103652095

Dedicatoria

Robert Sebastián Murillo Ayora

Hoy termina un capítulo importante en mi vida académica, quiero dedicar este trabajo a las personas que han estado para mí y me han apoyado de manera incondicional durante mi formación profesional y sobre todo durante mi formación personal.

El presente trabajo se lo dedico a mis padres Robert Murillo y Fabiola Ayora, que en cada etapa de mi vida me han apoyado y me han ayudado a convertirme en la persona que soy, con sus consejos y apoyo nunca hubiera llegado hasta este punto.

A mi hermano Daniel, que ha sido mi mayor apoyo desde que tengo memoria, y ha sido un pilar fundamental para que no me dé por vencido, gracias hermano.

A mi hermano Martín que quiero aprecio mucho, estoy orgulloso de ti y gracias por estar siempre a mi lado.

A toda mi familia, gracias por su apoyo, amor y cariño que me han dado en cada etapa de mi vida.

A mis docentes que me han compartido su conocimiento y sabiduría que han contribuido en mi formación profesional y humana.

Y sobre todo se lo dedicó al señor que ha sido un farol en mis momentos más oscuros y me ha ayudado a seguir adelante siendo mi guía manteniendo mi fe.

Agradecimientos

Robert Sebastián Murillo Ayora

Agradezco a mi papá que me ha dado su consejo y me ha impulsado a ser el mejor hombre que puedo llegar a ser. Gracias por tu esfuerzo, paciencia, tiempo y sobre todo cariño que me has dado, has sido un pilar fundamental para mí, y siempre tendré presenta cada uno de tus sabios consejos. El límite está en el cielo.

A mi madre no tengo nada más que decirle que gracias por acompañarme, por enseñarme compasión, por enseñarme que el amor de una madre es infinito y que su apoyo es invaluable.

A todo mis amigos, en especial mi amigo Alex, que me ha apoyado ayudado durante toda la universidad, gracias por tu amistad, amigo.

A todos mis docentes que me han compartido su sabiduría y conocimiento, que han sido mi guía en el camino de la ingeniería, que me han ayudado a superarme en cada materia que he recibido.

Este documento fue realizado enteramente en LaTeX

Índice

Lista de Tablas

Lista de Figuras

Resumen

El presente trabajo de titulación aborda el diseño de un sistema traductor de codigo G a KRL para el robot KUKA KR16-2 de la Universidad Politécnica Salesiana, con el fin de que el robot pueda ser adaptado a entornos de mecanizado CNC. El objetivo principal es que el sistema traductor de código G a KRL supere las limitaciones del sistema de programación nativo del robot KUKA, que no es compatible con comandos de fresado programados en código G.

El sistema implementado en el robot es capaz de convertir instrucciones de código G en comandos ejecutables por el robot. El proceso incluye el análisis de la dinámica y cinemática del robot, mediante el método recursivo Newton-Euler, se desarrolló un algoritmo para medir los torques en cada posición del robot, también se incluyó el análisis de las limitaciones del robot en entornos de mecanizado CNC.

Los resultados obtenidos incluyen la fabricación de figuras geométricas. Se realizaron pruebas en las que el robot KUKA KR16-2 ejecutó tareas de fresado siguiendo trayectorias previamente diseñadas. Se fabricó círculos, octágonos y dodecágonos en el plano, así como una pirámide escalonada y un fresado adaptativo circular en tres dimensiones. Durante estas pruebas, se midieron las desviaciones y variaciones dimensionales entre las piezas diseñadas y las obtenidas.

Palabras clave: Codigo G, Codigo KRl, Sistema traductor, Cinemática, Spindle, fresado, Control numérico computarizado

Abstract

This thesis addresses the design of a G-code to KRL transcompiler system for the KUKA KR16-2 robot of the Politécnica Salesiana University, so that the robot can be adapted to CNC machining environments. The main objective is for the G-code to KRL translation system to overcome the limitations of the KUKA robot's native programming system, which is not compatible with milling commands programmed in G-code.

The system implemented in the robot is capable of converting G-code instructions into commands executable by the robot. The process includes the analysis of the dynamics and kinematics of the robot, using the Newton-Euler recursive method, an algorithm was developed to measure the torque at each position of the robot, and the analysis of the robot's limitations in CNC machining environments was also included.

The results obtained include the manufacture of geometric figures. Experimental test were conducted in which the KUKA KR16-2 robot performed milling operations along predefined trajectories. Circles, octagons and dodecagons were manufactured on the plane, as well as a stepped pyramid and circular adaptive milling in three dimensions. During these tests, dimensional deviations and variations between the designed and obtained parts were measured.

Keywords: G-code, KRL-code, Transcompiler system, Kinematics, Spindle, Milling, Computerized numerical control

1. Introducción

La robótica industrial ha experimentado un avance significativo en los últimos años. Según la (International Federation of Robotics, 2023), el uso de robots en las empresas ha aumentado un 5 porciento a nivel mundial. Este incremento ha promovido la aplicación de robots, especialmente los antropomórficos, en diversas tareas dentro de la industria manufacturera. Sin embargo, existe un área donde no se ha explotado completamente su potencial y se observan limitaciones en entornos de manufactura CNC. Este desafío se debe al uso del lenguaje de programación KRL (KUKA Robot Language), que enfrenta dificultades de integración directa con los procesos de fresado que emplean código G. Este documento, elaborado en la Universidad Politécnica Salesiana, sede Cuenca, propone el diseño y la implementación de un sistema traductor de código G a KRL. Este sistema busca superar las dificultades de compatibilidad, permitiendo que el robot KUKA KR16-2 ejecute procesos de manufactura CNC que tradicionalmente utilizan código G, mejorando así su aplicación en la producción industrial.

2. Problema

2.1. Descripción del problema

En la Universidad Politécnica Salesiana, el sistema actual de programación para el robot KUKA KR16-2 se limita a reconocer únicamente instrucciones en KRL (KUKA Robot Language), sin embargo, el robot KUKA podría trabajar en manufactura aditiva como extractiva aprovechando sus 6 grados de liberad. Debido a esto se hace necesario la integración de un sistema que se adapte al controlador existente para convertir Código G a KRL, adaptándose al controlador existente.

Se añade a la problemática que el robot KUKA KR16-2 no es compatible con plug-ins como el KUKA.CNC que permite el uso de código G debido a la antigüedad del sistema, esta limitación afecta la programación del robot, ya que el uso de código G es un estándar en la industria para controlar máquinas CNC. La falta de compatibilidad puede requerir métodos alternativos más complejos y menos intuitivos para programar las tareas de fresado (RoboDK, 2022).

El código KRL (KUKA Robot Language) no fue creado con la finalidad de realizar tareas de fresado. Este lenguaje específico de programación, aunque potente para controlar los movimientos y operaciones del robot KUKA, no está optimizada para los patrones complejos y las trayectorias precisas que requiere el fresado. La programación de tareas de fresado implica la definición detallada de rutas y velocidades de herramienta, algo que en código KRL puede resultar tedioso y consumir mucho tiempo. Esto se debe a la necesidad de traducir los diseños y trayectorias del fresado, normalmente expresados en código G por software de CAM (Computer-Aided Manufacturing), a instrucciones que el KRL pueda interpretar (Braumann y Brell-Çokcan, 2011).

2.2. Antecedentes

Estos robots están diseñados para imitar la fisiología y apariencia humana, emulando partes del cuerpo como el brazo, el hombro y la muñeca. debido a esto el robot KUKA se le brinda un amplio espectro de aplicaciones prácticas y también los hace perfectos para tareas que demandan gran precisión y adaptabilidad (Esneca, 2019).

El KUKA KR16-2 es un robot industrial que ofrece seis grados de libertad, lo que le permite realizar movimientos complejos y precisos para tareas de manipulación y montaje. La utilización de código G en el KR16-2 puede estar limitada por su controlador y software, que no están optimizados para las operaciones de fresado que requieren código G. El KUKA KR16-2, ha experimentado limitaciones en sus capacidades debido a los progresos en la robótica. Una de sus restricciones principales radica en la incompatibilidad con plug-ins como el KUKA.CNC, que permite el uso de código G para diferentes aplicaciones como el fresado (KUKA, 2021).

Una restricción del controlador KUKA KR C2 es que su sistema operativo que utiliza es Windows XP que finalizo su soporte en el año 2014. A diferencia de los nuevos modelos como el KUKA KR C4 que opera con Microsoft Embedded Windows 7, reflejando una evolución en la tecnología de control y ofreciendo capacidades mejoradas en términos de procesamiento, seguridad y conectividad en los modelos más recientes como el KR C4 (RoboDK, 2022).

2.3. Importancia y alcances

El proyecto se centrará en el diseño y la implementación de un sistema traductor de Código G a KRL para el robot KUKA KR16-2 en la Universidad Politécnica Salesiana. El objetivo principal es desarrollar un sistema que permita la conversión efectiva de los programas, repotenciando así el robot antropomórfico en procesos específicos de manufactura extractiva. La relevancia de este proyecto reside en asegurar que el producto final cumpla con los estándares de calidad predefinidos y que se ejecute correctamente el programa. El propósito es desarrollar un sistema que habilite al robot antropomórfico para realizar interpolaciones lineales y circulares, manteniendo una alta precisión en sus movimientos.

El sistema propuesto deberá replicar una CNC 2 ejes y medio, asegurando la precisión posicional limitando las cargas a las capacidades del robot. El diseño del sistema debe tomar en cuenta los seis grados de libertad del robot KR16-2 manteniendo un correcto posicionamiento mediante los cálculos de su cinemática y permitiendo su adecuada manipulación durante el fresado.

2.4. Delimitación

El problema de estudio se delimitará en las siguientes dimensiones:

2.4.1. Espacial o geográfica

Este proyecto se desarrollará en la Universidad Politécnica Salesiana en el Campus El Vecino: Calle Vieja 12 - 30 y Elia Liut.

Figura 1

Ubicación Satelital Universidad Politécnica Salesiana

Nota: *Ubicación Universidad Politécnica Salesiana* (2022).

2.4.2. Temporal

El desarrollo del trabajo de titulación se llevará a cabo durante la asignatura de Integración Curricular con un total de 240 horas.

2.4.3. Sectorial o institucional

El presente trabajo de titulación está visto dentro de la Universidad Politécnica Salesiana en el área de robótica industrial.

3. Objetivos

3.1. Objetivo general

- Diseñar un programa traductor de código G a código KRL implementable en el robot KUKA KR16-2.

3.2. Objetivos Específicos

- Analizar la dinámica del robot KUKA KR16-2 aplicada a procesos de manufactura por arranque de viruta CNC en la operación de fresado.

- Diseñar un sistema traductor entre código G y KRL, garantizando la operatividad del robot KR16-2 y permitiendo ejecutar tareas definidas en código G replicando una CNC de dos ejes y medio.

- Implementar el sistema traductor en el robot KUKA KR16-2.

- Realizar pruebas del funcionamiento del traductor en el robot KUKA KR16-2.

4. Marco Teórico

Las máquinas CNC con sus siglas que especifican control numerico computarizado se especializan en manufactura extractiva y aditiva, cuentan con un software especializado capaz de generar instrucciones automatizadas llamado código G. Según Grumeber (2021) este software permite operar una variedad de equipos como tornos, fresadoras, molinos y cortadoras láser, entre otros, permitiendo así procesos de fabricación complejos y precisos.

El código G, dicta acciones específicas a ejecutar, como interpolaciones o funciones de taladrado. Además, gestiona operaciones más complejas que pueden incluir el uso de herramientas motorizadas opcionales. Los códigos G se organizan en grupos numéricos, donde cada grupo engloba comandos destinados a un propósito específico. Por ejemplo, el grupo 1 de la tabla

1 se encarga de dirigir los movimientos punto a punto de los ejes de la máquina, mientras que el grupo 7 de la tabla 1 regula la compensación de la herramienta de corte (Haascnc, 2021).

Tabla 1

Códigos G.

CÓDIGO	DESCRIPCIÓN	GRUPO
G00	Posicionamiento de movimiento rápido	1
G01	Movimiento de interpolación lineal	1
G02	Movimiento de interpolación circular en sentido horario	1
G03	Movimiento de interpolación circular en sentido antihorario	1
G04	Pausa	0
G09	Parada exacta	0
G10	Establecer correctores	0
G12	Fresado circular de cavidades en sentido horario	0
G13	Fresado circular de cavidades en sentido antihorario	0
G17	Selección de plano XY	2
G18	Selección de plano XZ	2
G19	Selección de plano YZ	2
G20	Seleccionar pulgadas	6
G21	Seleccionar sistema métrico	6
G28	Retorno al punto cero de la máquina	0
G29	Retorno desde el punto de referencia	0
G31	Avance hasta salto	0
G35	Medida automática del diámetro de la herramienta	0
G36	Medida automática del corrector de piezas	0
G37	Medida automática del corrector de herramientas	0
G40	Cancelar compensación de la herramienta de corte	7
G41	Compensación de la herramienta de corte izquierda 2D	7
G42	Compensación de la herramienta de corte derecha 2D	7
G43	Compensación de la longitud de la herramienta + (Añadir)	8
G44	Compensación de la longitud de la herramienta - (Restar)	8
G47	Engrabación de texto	0

Tabla 1

Códigos G.

CÓDIGO	DESCRIPCIÓN	GRUPO
G49	G43/G44/G143 Cancelar	8
G50	Cancelar escala	11
G51	Escala	11
G52	Establecer sistemas de coordenadas de trabajo	0 o 12
G53	Selección de coordenadas de la máquina no modal	0
G60	Posicionamiento unidireccional	0
G61	Modo de parada exacta	15
G65	Opción de llamada a subprograma macro	0
G68	Rotación	16
G69	Cancelar G68 Giro	16
G70	Círculo de agujero para tornillos	0
G71	Arco de agujero para tornillos	0
G72	Agujeros para tornillos a lo largo de un ángulo	0
G73	Ciclo fijo de taladrado intermitente de alta velocidad	9
G74	Ciclo fijo de roscado inverso	9
G76	Ciclo fijo de mandrinado fino	9
G77	Ciclo fijo de mandrinado posterior	9
G80	Cancelar ciclo fijo	9
G81	Ciclo fijo de taladrado	9
G82	Ciclo fijo de taladrado de punto	9
G83	Ciclo fijode taladrado intermitente normal	9
G84	Ciclo fijo de roscado	9
G85	Ciclo fijo de mandrinado	9
G86	Ciclo fijo de mandrinado y parada	9
G89	Ciclo fijo de mandrinado hacia dentro, pausa y mandrinado hacia fuera	9
G90	Comando de posicionamiento absoluto	3
G91	Comando de posicionamiento incremental	3
G92	Establecer valor de cambio de sistemas de coordenadas de trabajo	0
G93	Modo de avance de tiempo inverso	5
G94	Modo de avance por minuto	5

Tabla 1

Códigos G.

CÓDIGO	DESCRIPCIÓN	GRUPO
G95	Avance por revolución	5
G98	Retorno al punto inicial de ciclo fijo	10
G99	Retorno al plano R de ciclo fijo	10
G100	Cancelar imagen especular	0
G101	Habilitar imagen especular	0
G103	Limitar almacenamiento de bloques	0
G107	Correlación cilíndrica	0

Nota: Comandos básicos cdigo G (Haascnc, 2021).

El lenguaje de programación para CNC tiene normas específicas como el estándar alemán DIN 66024 y 66025, en el estándar internacional es equivalente a la ISO 1056 (Internacional Organization for Standardization), que rigen la estructura de programa para máquinas de control numérico computarizado(Deans, 2021).

En la programación CNC con código G, las líneas numeradas comienzan con 'N' seguido de un número, como se observa en la figura 2, facilitando la organización del programa. El código G01 se usa para interpolaciones lineales, indicando un movimiento recto a coordenadas específicas, un ejemplo de movimiento en 'x' es G01 X50, con este comando la fresa se mueve 50 milímetros en 'x' positivo. Los comandos G02 y G03 gestionan las interpolaciones circulares en sentidos horario y antihorario, respectivamente, necesitan coordenadas y detalles del centro o radio del arco, por ejemplo, G02 X100 Y50 I20 J25 (Smid, 2003). Después de analizar el código G se va a analizar el código KRL para posteriormente realizar una comparación en la codificación.

4.0.1. Código KRL

El código KRL que es usado en robots KUKA, tiene una estructura básica, que incluye definiciones de movimientos, tareas, y gestión de entradas/salidas. Los programas se escriben en bloques que contienen instrucciones para operar el robot. Comienzas definiendo el programa con DEF, seguido del nombre del programa, y usan comandos como PTP para movimientos

PALABRAS	BLOQUE	PROGRAMA
N5	N5 G01 Z1.5 M08	N5 G01 Z1.5 M08
G01		N10 Z0 X25.
Z1.5		N15 G03 X27. Z-1. R1.
M08		N20 G00 X30.
		N25 G28 W0.
		N30 M30

Figura 2

Diagrama de programación de robot.

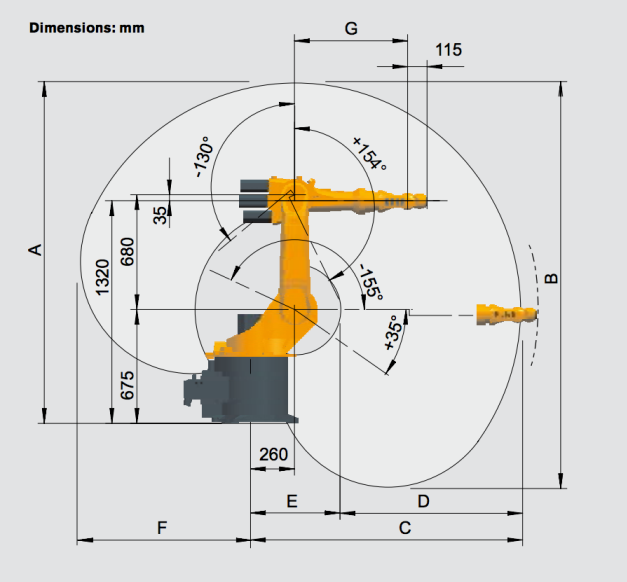


Figura 3

Medidas nominales Robot KR16-2 (KUKA, 2021).

punto a punto o LIN para movimientos lineales (KUKA, 2021).

Sin embargo, para la aplicación de un sistema traductor de código G a KRL resulta conveniente el uso de coordenadas lineales en lugar de comandos PTP, esto debido a que el comando PTP (point to point) determina la ruta más eficiente para que el robot alcance la posición deseada, por lo tanto, no se puede utilizar como sustituto del comando G01 del código G, en este caso el comando LIN en KRL es ideal para realizar interpolaciones lineales, ya que permite programar el plano de trabajo y utilizar coordenadas específicas para controlar los movimientos del robot con precisión.

Para realizar movimientos lineales (LIN), se programa la posición de destino, asegurando un trayecto controlado hacia el objetivo establecido. Por ejemplo, el comando LIN X 500, Y 0, Z 1000 movería al robot de manera lineal al punto especificado en coordenadas cartesianas. Este comando asegura que el robot se mueve en una línea recta desde la posición actual hacia la posición programada en 'x', 'y' y 'z'. Para implementar esto de manera correcta, es necesario definir con precisión los puntos objetivos y configurar las velocidades apropiadas. (KUKA, 2021).

Para realizar movimientos circulares con un robot KUKA en KRL, se utilizan los comandos CIRC o CR. Se debe especificar al menos tres puntos: la posición actual, un punto intermedio que define la curva, y el punto final de la trayectoria. Por ejemplo, CIRC X 100, Y 200, Z 300, X 400, Y 500, Z 600 realizaría un movimiento circular comenzando en la posición actual, pasando por el primer conjunto de coordenadas y terminando en el segundo conjunto (KUKA, 2021).

Para establecer un plano de trabajo en KRL (KUKA Robot Language), se usa el comando BASE seguido de la definición del plano. Por ejemplo, BASE X 500, Y 0, Z 100, A 0, B 0, C 0 definiría un nuevo plano de trabajo en las coordenadas especificadas. Esto permite al robot interpretar las coordenadas subsiguientes de los movimientos en relación con este plano de trabajo establecido (KUKA, 2021).

En similitud al código G, los movimientos lineales se realizan según Smid (2003) con el comando G01, seguido de coordenadas específicas, mientras que los circulares utilizan G02 o G03 para direcciones horaria y antihoraria, respectivamente. El movimiento en el eje Z se indica directamente con la coordenada Z. En KRL, el comando LIN realiza movimientos lineales hacia un punto especificado, CIRC para movimientos circulares definiendo puntos

intermedios y final, y el movimiento en eje Z se especifica dentro de las coordenadas del comando de movimiento.

En KRL, se establece un plano de trabajo usando el comando BASE, donde se define la posición y orientación del plano respecto al sistema de coordenadas del robot. En código G, según Smid (2003) se establece un sistema de coordenadas de trabajo (G54 a G59) para definir el origen y orientación del plano de trabajo en relación con la pieza o la máquina. Ambos métodos permiten al programador definir un contexto de referencia para los movimientos subsiguientes del robot o la herramienta de la máquina.

4.0.2. Comparación de la Codificación en Código G y KRL: Análisis de Similitudes

El Código G y el KUKA Robot Language (KRL), tienen sus propias aplicaciones especializadas, pero con similitudes en su estructura y funcionalidad. El presente análisis se centra en una comparativa de las propiedades de cada lenguaje, proporcionando una base para entender cómo pueden ser utilizados de manera intercambiable en diferentes entornos de producción. Se van a comparar aspectos fundamentales como la sintaxis de programación, la estructura de comandos y los sistemas de coordenadas, al hacer esto se busca facilitar la traducción de interpolaciones generadas. Se van a resaltar las características comunes.

Por ejemplo en la siguiente linea de código se declara interpolaciones lineales con G02 en la dirección de z de 0.1 milímetros, ademas se mueve en dirección de Y en 1.962 milimetros.

N65 G01 Z0.1 F333.

N70 Y1.962 F1000.

El equivalente en KRL es el comando de interpolación lineal para mover el robot en dirección de Y es equivalente a:

DECL E6POS XP5=X 0,Y 1.962,Z 0,A 180.000,B 0.000,C 180.000.

A 180: El ángulo de rotación alrededor del eje X, en grados.

B 0.000: El ángulo de rotación alrededor del eje Y, en grados.

C 180: El ángulo de rotación alrededor del eje Z, en grados.

Los ángulos A, B y C se definen mediante el ángulo deseado del efector final del robot, en base en lo calculado, el algoritmo interpola cada uno de los movimientos tomando en cuenta la posición de las articulaciones del robot en un espacio de trabajo.

4.0.3. Cinemática y dinámica de un robot

La cinemática directa en robótica se utiliza para calcular la posición y orientación de la herramienta o extremo del robot (efector final) en el espacio tridimensional. En el caso de un robot antropomórfico, que imita la estructura y movilidad de un brazo humano, este proceso se basa en la información de las articulaciones, como los ángulos de rotación y desplazamientos lineales. Mediante algoritmos como el método Denavit-Hartenberg se puede determinar la ubicación exacta del efector final (Rotella, 2021).

El método Denavit-Hartenberg es empleado para los cálculos de cinemática directa, proporcionando un esquema sistemático para caracterizar la estructura de un robot. Este método requiere la asignación de sistemas de coordenadas a las articulaciones del robot y la definición de los parámetros de Denavit-Hartenberg, que son la longitud, desplazamiento, ángulo y giro, para los conjuntos de coordenadas adyacentes (Abdolmalaki, 2017).

A diferencia de la cinemática, que se centra exclusivamente en el estudio de los movimientos sin considerar las fuerzas que los causan, la dinámica en robótica aborda tanto los movimientos como las fuerzas que experimentan los robots (Corke, 2017). Este campo analiza cómo interactúan las fuerzas externas e internas con la masa y la inercia de los componentes del robot. El estudio de la dinámica con métodos como el Newton-Euler permiten el diseño de sistemas de control efectivos para la simulación precisa de robots.(Siciliano y Khatib, 2016).

El método de dinámica Newton-Euler se fundamenta en las leyes de Newton, que gobiernan la traslación, y en las leyes de Euler, que describen la rotación. Este enfoque es directo y frecuentemente se aplica a sistemas que incluyen pocos cuerpos o en situaciones donde las fuerzas tienen un impacto más crítico que la energía del sistema. Siciliano y Khatib (2016).

4.0.4. Trabajos previos

La integración de robots antropomórficos en entornos CNC fue objeto de varias investigaciones en el campo de la manufactura, un aspecto fundamental de esta integración radica en la capacidad de traducir G-Code (el lenguaje utilizado tradicionalmente para dirigir las máquinas CNC) a otros lenguajes de robots, ofreciendo un nuevo espectro de capacidades operativas. Esta revisión de la literatura examina este proceso de traducción, con énfasis en las implicaciones para los robots KR16-2, un modelo que ejemplifica la convergencia de la robótica y la tecnología CNC es el desarrollado por Pan y cols. (2022) el cual aborda la programación fuera de línea (OLP) para robots industriales en tareas de fresado, enfocándose en la conversión de comandos G-Code a trayectorias de control para robots. En dicha investigación se presenta un software llamado RobMach que facilita esta conversión y permite la simulación de fresado robótico.

Estudios anteriores se centran en aumentar las capacidades robóticas, sentando así precedentes fundamentales para su implementación en procesos de fabricación sustractivos (Siciliano, Sciavicco, Villani, y Oriolo, 2009). Al adoptar estos hallazgos e integrar conceptos del diseño asistido por computadora (CAD) y la robótica (Klimchik, Ambiehl, Garnier, Furet, y Pashkevich, 2016), permitió realizar complejas tareas de mecanizado, antes inalcanzables por los robots convencionales. Estos avances formativos en la manipulación robótica fomentaron el desarrollo de algoritmos y protocolos que permitirían a los manipuladores industriales como el KR16-2 no solo emular las funciones de las máquinas CNC, sino también superarlas potencialmente en ciertos aspectos (Corke, 2017).

El KR16-2 cuenta con un diseño robusto, propicio para una amplia gama de aplicaciones industriales, proporcionando una capacidad de carga útil de hasta 16 kilogramos, y un amplio alcance de aproximadamente 1610 milímetros (KUKA, 2020). En particular, la versatilidad de seis ejes del robot, permite articulaciones intrincadas, lo que presenta importantes ventajas en términos de capacidad, para ejecutar tareas complejas. Inicialmente, sus aplicaciones se concentraron principalmente en los ámbitos de manipulación, montaje y soldadura por puntos, donde se aprovecha la precisión y repetibilidad del KR16-2, cualidades que son primordiales en tales entornos.

Sin embargo, el giro hacia los contextos CNC, específicamente en el fresado, ha revelado

ciertas limitaciones inherentes al robot. La principal de ellas es la rigidez limitada en comparación con la maquinaria de fresado CNC tradicional, que afecta inherentemente la precisión de las tareas de mecanizado y la estabilidad de la trayectoria de la herramienta bajo las cargas dinámicas experimentadas durante los procesos de eliminación de material (Altintas, 2012). El fresado CNC de alta tolerancia presenta desafíos adicionales, agravados por la optimización limitada basada en software para tales operaciones dentro del conjunto de control estándar del robot.

La ausencia de soluciones de código abierto para la conversión de G-code a KRL, específicamente para el robot KR16-2, representa un obstáculo considerable dentro de las investigaciones existentes. Esta carencia de programas de código abierto obstaculiza las tareas de fresado asistido por robots, ya que los sistemas propietarios restringen tanto la accesibilidad como la expansión colaborativa del conocimiento. Las ventajas de las iniciativas de código abierto son múltiples, incluyendo la promoción de innovaciones rápidas, la reducción de costos y un enfoque dirigido por la comunidad que podría acelerar los ciclos de desarrollo y solucionar problemas de manera más efectiva.

5. Marco metodológico

5.1. Establecimiento del punto de trabajo.

Para configurar el punto de trabajo inicial de un robot KUKA de manera similar a una máquina CNC, donde se define el punto cero al rozar el material, se sigue un proceso que utiliza las coordenadas angulares de cada articulación del robot. Este método permite que el robot establezca un punto de referencia preciso que servirá como base para todos los movimientos posteriores en los ejes X, Y, Z.

Posicionamiento Manual para Establecer el Punto Inicial Primero, se mueve manualmente las articulaciones del robot (A1 a A6) hasta que la herramienta toque ligeramente el material en el punto deseado.

Este contacto simula el proceso de una máquina CNC al establecer el punto cero. Una vez que el robot esté en esta posición, se captura las coordenadas angulares de cada articulación, ya que estas definirán el punto de trabajo inicial.

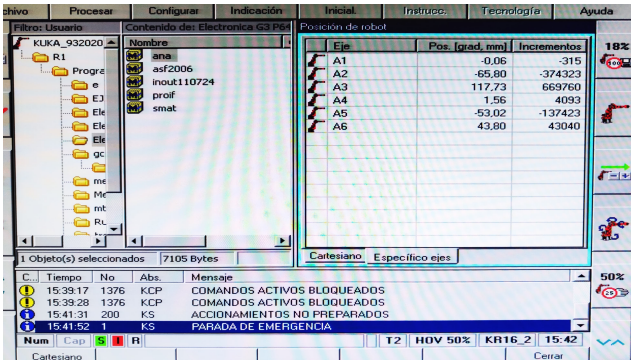


Figura 4

Posiciones angulares KUKA KR16-2 (KUKA, 2021).

Con las coordenadas angulares capturadas, se define el punto inicial en el código KRL del robot. Este punto se representará como una posición de articulación, en el código se expresa como (E6AXIS), lo que permitirá al robot reconocer esta posición como el punto cero. Un ejemplo implementado en el código es:

DECL E6AXIS P0=A1 -14.58420, A2 -56.13140, A3 130.63200, A4 164.89000, A5 -75.01220, A6 45.14.

5.2. Obtención de un sistema de coordenadas mediante la cinemática directa del robot.

Para obtener las coordenadas se aplicó el procedimiento de Denavit-Hartenberg como se observa en la tabla 2 para calcular el modelo cinemático directo del robot. Determinaremos la relación entre las variables articulares y las posiciones/orientaciones del extremo del robot, esto determinará la posición y orientación finales del extremo del robot para una configuración dada de las articulaciones.

i	θ_i	d_i	a_i	α_i
1	$-\theta_1$	l_1	l_2	$-90°$
2	θ_2	0	l_3	$0°$
3	$\theta_3 - 90°$	0	$-l_4$	$-90°$
4	θ_4	$l_6 + l_5$	0	$90°$
5	θ_5	0	0	$-90°$
6	θ_6	l_3	0	$0°$

Tabla 2

Parámetros de Denavit-Hartenberg para el robot KUKA KR16-2

5.2.1. Dimensiones del robot KUKA

l1 = 675 mm que es medido desde el suelo.

l2 = 260 mm Offset de la base con la 2da articulación.

l3 = 680 distancia de la articulación 2 y 3.

l4 = 35 mm Offset de la articulación 3 y 5.

l5 = 500 mm l5 y l6, es medido de la articulación 3 a 5.

l6 = 210 mm.

l7 = 118 mm la distancia de la articulación 4 a 6.

5.2.2. Matrices del robot KUKA

En el contexto de la robótica, las matrices de rotación como R(alpha) se usan para calcular y describir la orientación de las articulaciones o segmentos de un robot. En el caso del robot KUKA KR16-2, que es un robot industrial con múltiples grados de libertad, matrices de este tipo se aplicarían para controlar y simular la orientación de cada uno de sus ejes articulados.

La matriz de rotación sobre el eje X se expresa como:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz de rotación sobre el eje Y se expresa como:

$$R_y = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz de rotación sobre el eje Z se expresa como:

$$R_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz de traslación se expresa como:

$$T = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.3. Obtención de la cinemática inversa

Se desarrolló el modelo cinemático inverso, con un enfoque particular en los primeros tres grados de libertad. Para verificar su validez, se calcularon las variables articulares para diversos puntos y se comprobaron estas soluciones mediante la cinemática directa.

Se resuelve en el plano general el ángulo q1 como se observa en la figura 5.

Esta fórmula representa la distancia radial (r) en un plano horizontal (XY), donde (r) es la hipotenusa que resulta de aplicar el teorema de Pitágoras a las coordenadas (x) (y).

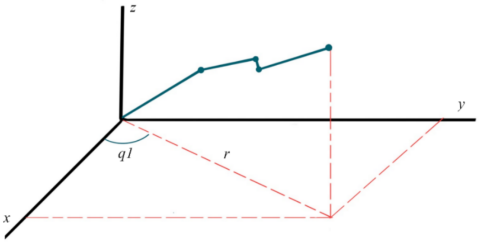


Figura 5

Diagrama KUKA KR16-2 tres últimas articulaciones.

$$r = \sqrt{x^2 + y^2} \tag{1}$$

Se obtiene el ángulo (q1) entre el eje (x) y el vector radial (r) usando la tangente inversa.

$$q_1 = \tan^{-1}\left(\frac{y}{x}\right) \tag{2}$$

Tomando en cuenta el ángulo que se forma entre la continuación del primer eslabón y el segundo, se obtiene el ángulo q2 como se referencia en la figura 6, se forma un nuevo triángulo.

$$L_1 = \sqrt{a_3^2 + a_4^2} \tag{3}$$

$$\alpha = \tan^{-1}\left(\frac{a_3}{a_4}\right) \tag{4}$$

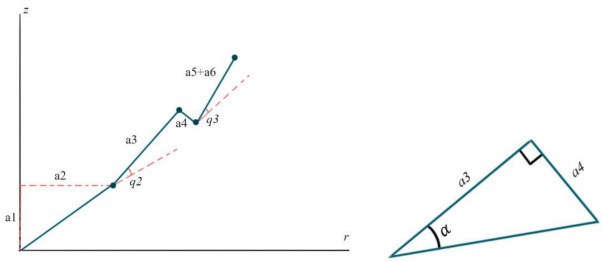


Figura 6

Vista frontal y enfoque en a3 y a4.

Para el último eslabón L2 como se observa en la figura 7, se considera las longitudes a5 y a6.

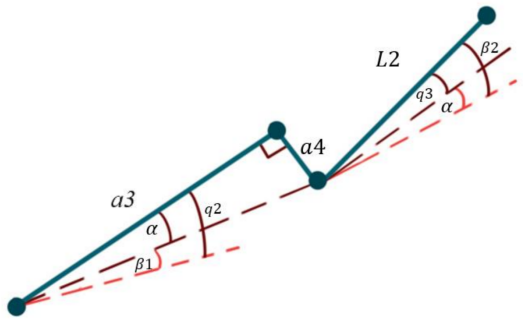


Figura 7

Vista frontal y ángulos.

$$L_2 = a_5 + a_6 \tag{5}$$

La ecuación de q2 índica como se relacionan los q2 y q3 con los ángulos beta1, beta2 y alpha. Los ángulos resultantes dependen de las relaciones geométricas entre los eslabones

$$q_2 = \beta_1 + \alpha \tag{6}$$

$$q_3 = \beta_2 - \alpha \tag{7}$$

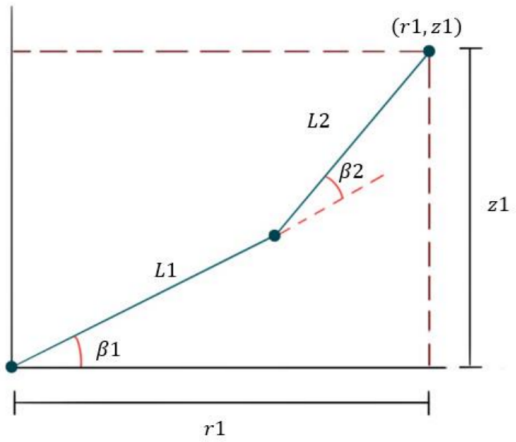


Figura 8

Vista frontal con enfoque en beta1 y beta2.

Para obtener r1 y z1 como se observa en la figura 8 se tiene que ajustar en función de las distancias a1 y a2.

$$r_1 = r - a_2 \tag{8}$$

$$z_1 = z - a_1 \tag{9}$$

Obtenidos z1 y r1 se calcula h mediante Pitágoras.

$$h = \sqrt{r_1^2 + z_1^2} \tag{10}$$

Obtenido h se puede obtener el coseno del ángulo beta2, en donde se incluyen la longitud de los eslabones l1,l2.

$$C2 = \cos(\beta_2) = \frac{h^2 - L_1^2 - L_2^2}{2 \cdot L_1 \cdot L_2} \tag{11}$$

Luego de obtener C2 se obtiene el seno de beta2 usando la relación trigonométrica entre coseno y seno.

$$S2 = \sqrt{1 - C2^2} \tag{12}$$

El ángulo beta2 se obtiene con la tangente inversa, utilizando el cociente y el coseno calculado previamente.

$$\beta_2 = \tan^{-1}\left(\frac{S2}{C2}\right) \tag{13}$$

El ángulo beta1 se calcula combinando la tangente inversa de z1/r1 y en la segunda parte se toma en cuenta beta2 en relación con l1 y l2.

$$\beta_1 = \tan^{-1}\left(\frac{z_1}{r_1}\right) - \tan^{-1}\left(\frac{L_2 \cdot \sin(\beta_2)}{L_1 + L_2 \cdot \cos(\beta_2)}\right) \tag{14}$$

5.4. Dinámica robot KUKA KR16-2

En el modelo dinámico, se incorporan las fuerzas y torques que influyen en el movimiento del robot. Se emplean variables simbólicas para representar las coordenadas articulares q_i, así como sus respectivas velocidades angulares $\dot{q}_i$ y aceleraciones angulares $\ddot{q}_i$. Adicionalmente, se consideran las masas de los eslabones del robot m_i y la aceleración gravitacional g para completar el sistema de ecuaciones que rige el movimiento del robot.

El modelo dinámico del robot se obtiene utilizando el método de Newton-Euler, que permite calcular las fuerzas y torques en las articulaciones del robot en función de las velocidades, aceleraciones y otros parámetros físicos.

5.4.1. Variables Simbólicas

Se definen las siguientes variables:

- q_i: Coordenadas articulares (ángulos de las articulaciones).
- $\dot{q}_i$: Velocidades angulares articulares.
- $\ddot{q}_i$: Aceleraciones angulares articulares.
- m_i: Masa de los eslabones del robot.
- g: Aceleración debido a la gravedad ($9.81\,\mathrm{m/s^2}$).

5.4.2. Matrices de Rotación

Para calcular las posiciones y orientaciones de los eslabones en el espacio, se utilizan las matrices de rotación homogéneas, como por ejemplo la descrita en la tabla 2.

Se definen varias matrices como $A_{01}, A_{12}, A_{23}, \ldots$, que describen la orientación relativa entre eslabones consecutivos.

5.4.3. Velocidad Angular

La velocidad angular de cada eslabón se calcula a partir de las matrices de rotación y las velocidades articulares. Para un eslabón i, se obtiene según Siciliano y cols. (2009) la siguiente ecuación.

$$w_i = A_{i-1,i} \cdot (w_{i-1} + z_{i-1} \cdot \dot{q}_i)$$

Donde w_i es la velocidad angular y z_{i-1} es el eje de rotación del eslabón anterior.

5.4.4. Aceleración Angular

La aceleración angular se calcula de manera similar, tomando en cuenta las aceleraciones articulares $\ddot{q}_i$ y las velocidades angulares anteriores. La fórmula para la aceleración angular es según Siciliano y cols. (2009).

$$\dot{w}_i = A_{i-1,i} \cdot (\dot{w}_{i-1} + z_{i-1} \cdot \ddot{q}_i) + w_{i-1} \times (z_{i-1} \cdot \dot{q}_i)$$

5.4.5. Aceleración Lineal y Fuerza

Se utiliza la siguiente ecuación para calcular la aceleración lineal de los eslabones según Siciliano y cols. (2009).

$$a_i = \dot{w}_i \times p_i + w_i \times (w_i \times p_i) + A_{i-1,i} \cdot a_{i-1}$$

Donde p_i es la posición del eslabón y a_{i-1} es la aceleración del eslabón anterior.

La fuerza sobre cada eslabón se calcula según Siciliano y cols. (2009).

$$f_i = m_i \cdot a_i$$

Posteriormente, esta fuerza se utiliza para calcular el torque en cada articulación.

5.4.6. Método de Newton-Euler

El método de Newton-Euler combina las ecuaciones de movimiento para la dinámica de traslación y rotación. La ecuación general del torque para un eslabón i es según Siciliano y cols. (2009).

$$\tau_i = \frac{d}{dt}(I_i w_i) + w_i \times (I_i w_i)$$

Donde I_i es el tensor de inercia del eslabón y w_i es la velocidad angular.

5.4.7. Torque en las Articulaciones

Finalmente, el torque en cada articulación se calcula con las fuerzas y los momentos de inercia, usando la ecuación según Siciliano y cols. (2009).

$$\tau_i = n_i^T A_{i-1,i} z_0$$

Aquí n_i es el vector de momento resultante y z_0 es el eje de rotación de la articulación.

5.4.8. Ejemplo de Torque

El torque resultante es:

$$\tau_1 = 72.0 \cdot \ddot{q}_1 + 33.0 \cdot \cos(2 \cdot q_2 + q_3) - 49.0 \cdot \cos(2 \cdot q_2) - 50.0 \cdot \cos(2 \cdot q_3) + \ldots$$

Este tipo de ecuaciones se aplica a todas las articulaciones para obtener los torques en función de las posiciones, velocidades y aceleraciones articulares.

El resultado final del torque está en Newton-metros (Nm).

Las gráficas resultantes para cada articulación contada desde A6 desde la base del robot se observa para la articulación 1 en la figura 9, para la articulación 2 en la figura 10, para la articulación 3 en la figura 11, para la articulación 4 en la figura 12, para la articulación 5 en la figura 13 y para la articulación 6 en la figura 14.

Para el análisis es necesario tomar en cuenta la rigidez de las máquinas CNC, según Chen y cols. (2022) la rigidez del robot es considerablemente menor en comparación con los sistemas CNC, siendo esta aproximadamente 50 veces superior en los últimos. Además, hay que tomar en cuenta la carga máxima permisible (en kg). La gráfica inferior muestra varias líneas que indican los diferentes pesos máximos (8 kg, 10 kg, 12 kg, 14 kg y 16 kg) que el robot puede manejar dependiendo de la posición del centro de gravedad de la carga en el eje Lz (el eje longitudinal del robot, que se extiende desde el robot hacia afuera). Esto implica que, mientras más lejos esté el centro de gravedad de la carga desde el robot, menor será el peso máximo

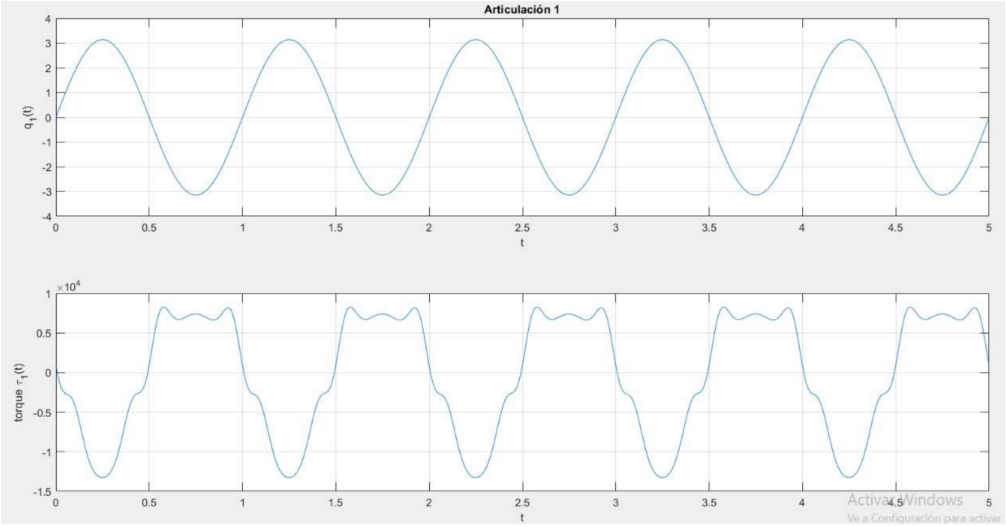


Figura 9

Angulo y torque de la articulación 1.

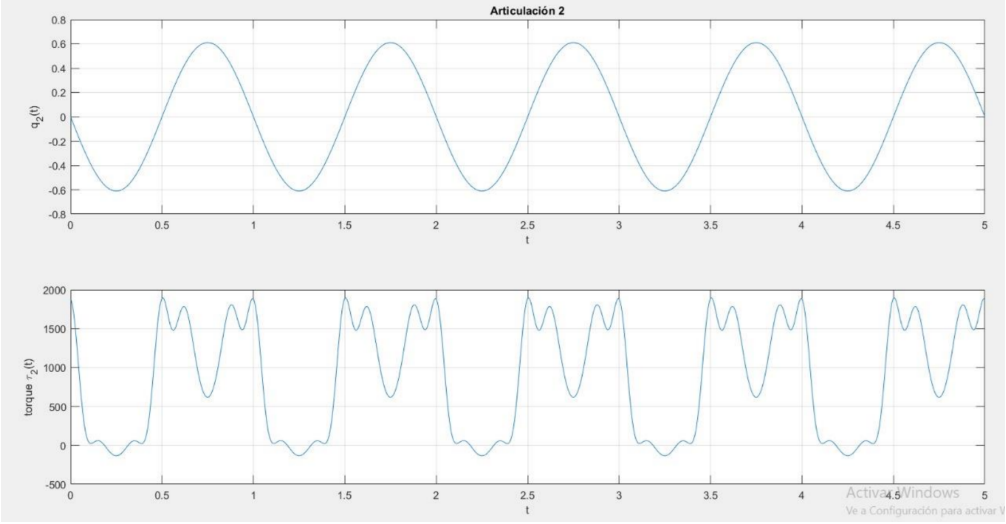


Figura 10

Angulo y torque de la articulación 2.

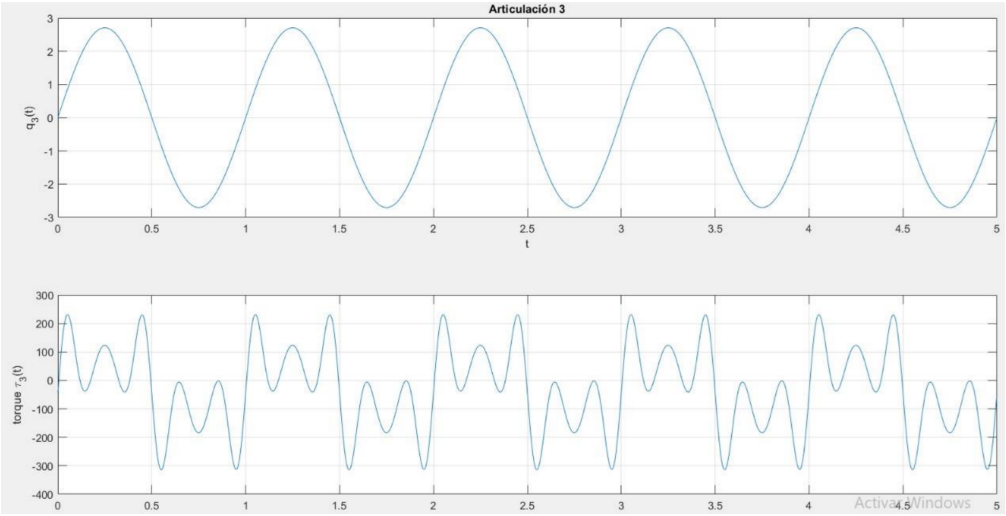


Figura 11

Angulo y torque de la articulación 3.

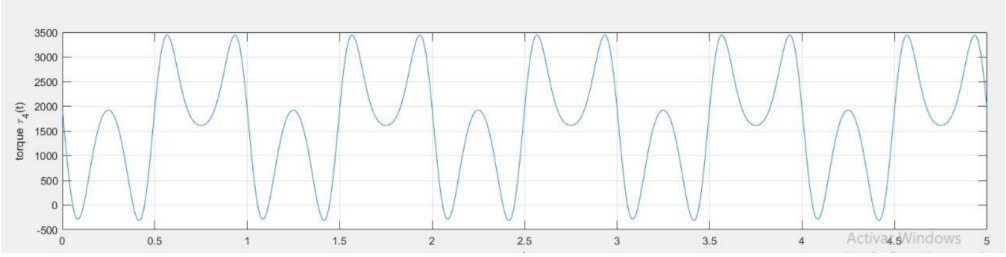


Figura 12

Torque de la articulación 4.

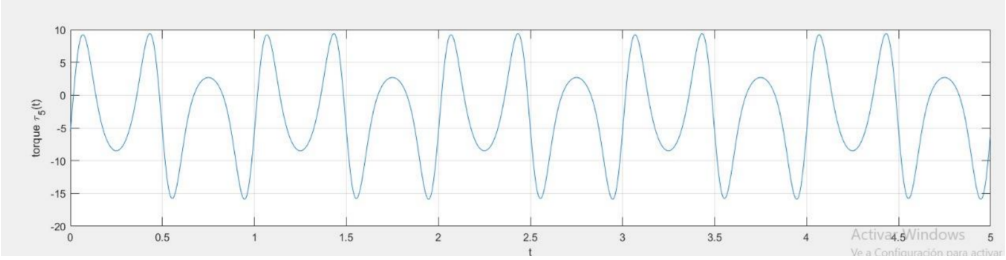


Figura 13

Torque de la articulación 5.

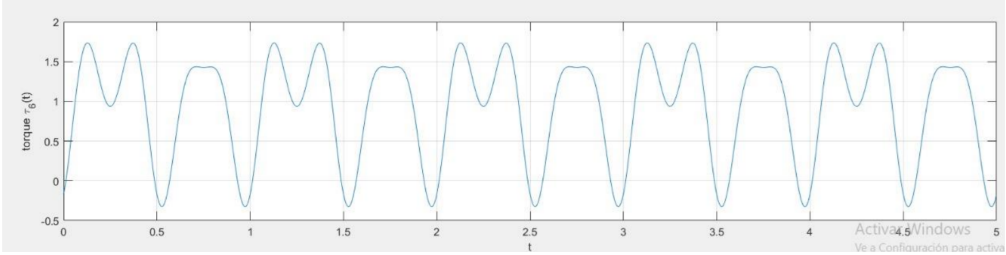


Figura 14

Torque de la articulación 6.

que el robot KUKA KR16-2 puede manejar (KUKA ROBOTICS, 2020).

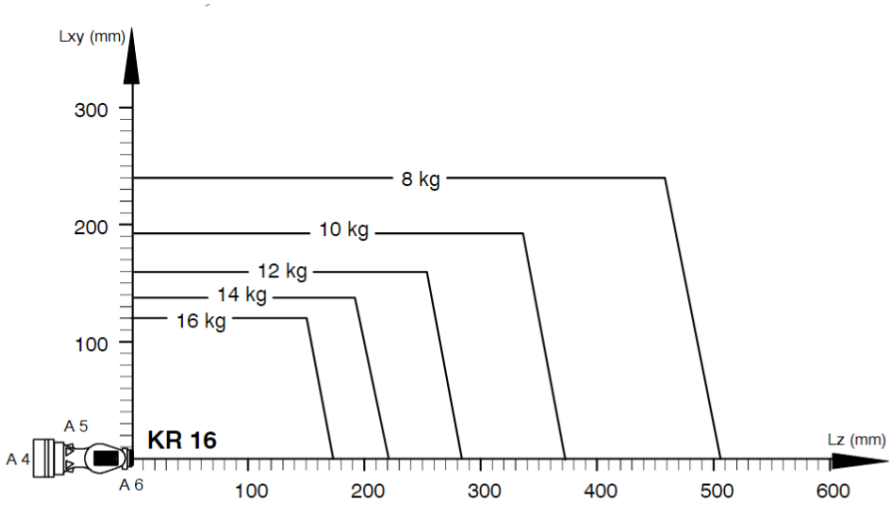


Figura 15

Curvas características para el KR 16-2 KUKA ROBOTICS (2020).

5.5. Programación del sistema traductor

KRL, el lenguaje de programación para los controladores KUKA en su versión KRC2, fue utilizado como base, mientras que todo el código que gestiona las posiciones en el espacio de trabajo y el sistema traductor, ilustrado en la figura 16, se desarrolló mediante programación en Python.

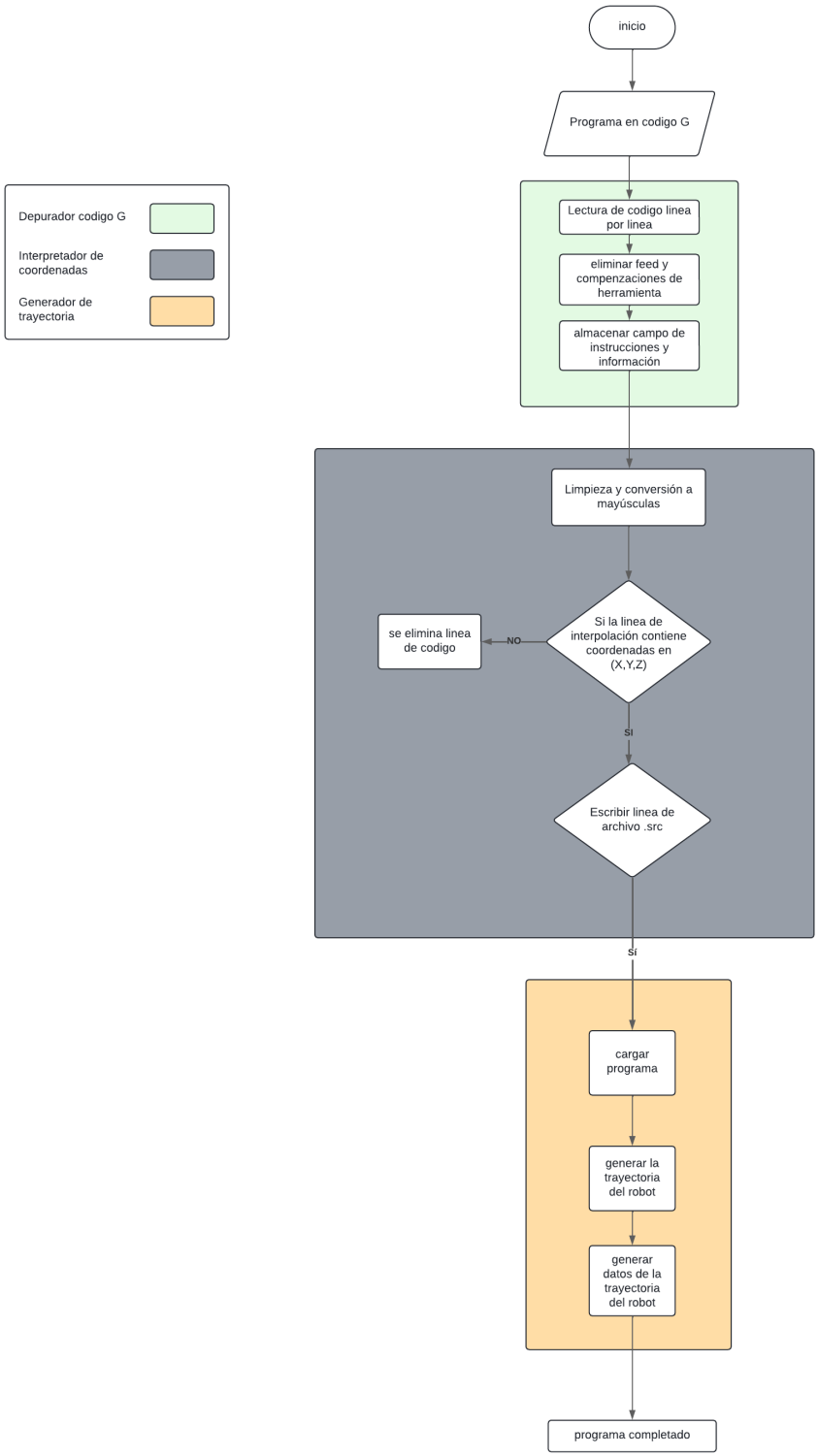


Figura 16

Diagrama de flujo de sistema traductor de codigo G a KRL.

5.5.1. Mapeo de Puntos en el Espacio de Trabajo y Transferencia de Archivos

Se inició programando directamente a través del KCP (KUKA Control Panel), partiendo desde la posición de reposo del robot hasta alcanzar una posición en la que los distintos se situara a la distancia cercana con respecto al plano de trabajo, seguidamente se alineó la herramienta con respecto a la mesa de trabajo, con el robot alineado se obtuvieron los ejes de rotación de cada una de las articulaciones desde el KCP como se observa en la figura 4, con estas posiciones se puede programar un primer comando PTP que va a llevar al robot a la posición deseada.

5.5.2. Sistema de traducción de codigo G a KRL

La primera parte del sistema de traducción de código G a código KRL se encarga de eliminar comandos especiales como los códigos M o códigos F que no van a ser necesarios para generar trayectorias.

Depurador codigo G:

```
import re

# Leer el contenido del archivo entero
with open('/mnt/data/circulomaquinado.txt', 'r') as file:
    file_content = file.read()

# quitar códigos específicos como 'F300', 'M05' usando regex
updated_content = re.sub(r'F\d+|M\d+', '', file_content)

# direccion del archivo
updated_file_path = '/mnt/data/circulomaquinado_updated.txt'

# Escribe el archivo depurado en un nuevo archivo
with open(updated_file_path, 'w') as file:
    file.write(updated_content)

```

Para realizar el sistema traductor de código G a KRL se toma en cuenta las características del código G, se empieza por extraer las coordenadas en 'x', 'y' y 'z'. En la siguiente línea se define un índice para las posiciones que incrementará una línea de programa con cada punto de datos procesado.

Lectura del codigo G:

```
def convert_gcode_lines_to_krl(gcode_lines):
    output_lines = []
    x, y, z = None, None, None
    position_index = 1
```

En el siguiente bloque se itera sobre cada línea de G-code se los divide en partes para extraer las coordenadas 'x', 'y' y 'z', el bloque los convierte en variables flotantes. Se generan líneas de código KRL con la función 'generate krl'. El índice de posición se incrementa tras procesar cada conjunto de coordenadas.

Ingreso de variables 'x', 'y' y 'z':

```
  for line in gcode_lines:
        parts = line.split()
        new_x, new_y, new_z = None, None, None

        for part in parts:
            if 'X' in part:
                new_x = float(part[1:])
            if 'Y' in part:
                new_y = float(part[1:])
            if 'Z' in part:
                new_z = float(part[1:])

        if new_x is not None:
            x = new_x
        if new_y is not None:
            y = new_y
```

```
17
18          if new_z is not None:
19              z = new_z
20              if x is not None and y is not None:
21                  output_lines.extend(generate_krl(position_index, x, y, z))
22                  output_lines.append("")
23                  position_index += 1
```

En este bloque se generan las tres líneas de código necesarias para que el robot KUKA entienda las interpolaciones. La primera declaración que se genera es él (E6POS), es una declaración de datos de posición del efector final, cuenta con las coordenadas 'x', 'y', 'z' y los ángulos de rotación 'A', 'B' y 'C'.

La segunda declaración en generarse es el FDAT, en donde especifica la herramienta a usarse, la base que el robot va a usar para interpretar las posiciones.

La última declaración en generarse es el LDAT, se especifica la velocidad, la aceleración, y variables para que el robot entienda el tipo de movimiento generado.

Líneas de código KRL FDAT, LDat y E6POS:

```
1    for line in gcode_lines:
2          parts = line.split()
3          new_x, new_y, new_z = None, None, None
4
5          for part in parts:
6              if 'X' in part:
7                  new_x = float(part[1:])
8              if 'Y' in part:
9                  new_y = float(part[1:])
10             if 'Z' in part:
11                 new_z = float(part[1:])
12
13         if new_x is not None:
14             x = new_x
```

```
        if new_y is not None:
            y = new_y

        if new_z is not None:
            z = new_z
            if x is not None and y is not None:
                output_lines.extend(generate_krl(position_index, x, y, z))
                output_lines.append("")
                position_index += 1
```

La última parte del sistema de traducción de código G a código KRL convierte las líneas de G-code a formato KRL mediante la función `convert_gcode_lines_to_krl`. Luego, une las líneas convertidas en un solo texto con saltos de línea. A continuación, abre un archivo de texto llamado `"KUKA_Declarations.txt"` en modo de escritura con codificación `UTF-8` y escribe el texto KRL generado en dicho archivo. Finalmente, imprime un mensaje que confirma que el archivo ha sido generado correctamente.

Bloque final sistema traductor de codigo G a codigo KRL:

```
  def generate_krl(index, x, y, z):
    e6pos = f"DECL E6POS XP{index}={{X {x:.3f},Y {y:.3f},
    Z {z:.3f},A 180.000,B 0.000,C 180.000}}"
    fdat = f"DECL FDAT FP{index}={{TOOL_NO 1,BASE_NO 2,
    IPO_FRAME #BASE,POINT2[] \" \",TQ_STATE FALSE}}"
    ldat = f"DECL LDAT LCPDAT{index}={{VEL 1.00000,ACC
    100.000,APO_DIST 1.000,APO_FAC 50.0000,ORI_TYP #VAR,
    CIRC_TYP #BASE,JERK_FAC 50.0000}}"
    return [e6pos, fdat, ldat]

gcode_lines = [
    "N45 G00 X32.2 Y-29.243",
    "N50 G43 Z15.",
    "N55 G00 Z5.",
    "N60 G01 Z0.",
    "N65 X32.01",
```

```
    # Añade más líneas según sea necesario
]

converted_krl_lines = convert_gcode_lines_to_krl(gcode_lines)
output_text = "\n".join(converted_krl_lines)

# Escribir el resultado en un archivo con codificación correcta
with open("KUKA_Declarations.txt", 'w', encoding='utf-8') as file:
    file.write(output_text)

print("Archivo generado correctamente: KUKA_Declarations.txt")
```

6. Resultados

Se desarrolló un programa en código G utilizando coordenadas absolutas y medidas expresadas en milímetros. A partir de este, se depuró el código para extraer las coordenadas precisas en milímetros. Las figuras generadas incluyeron un círculo con un radio de 25 milímetros, como el de la figura 18, un octágono como el de la figura 19 y un dodecágono como el de la figura 20.

Además, se adaptó un spindle al gripper del robot para poder realizar fresados en diferentes materiales, como se ve en la figura 17.

Figura 17

Spindle adaptado al robot KUKA KR16-2.

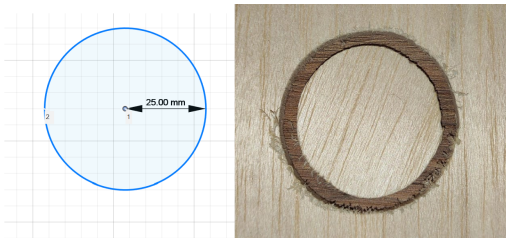


Figura 18

Triángulo generado con código G a la izquierda y fabricado con el robot KUKA KR16-2 a la derecha.

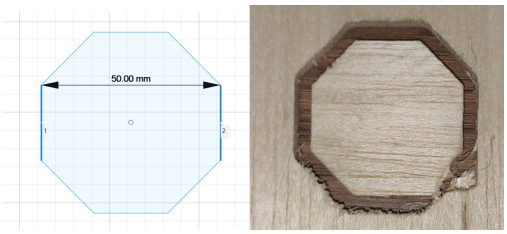


Figura 19

Octágono generado con código G a la izquierda y fabricado con el robot KUKA KR16-2 a la derecha.

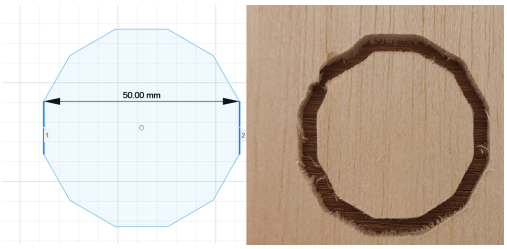


Figura 20

Dodecágono generado con código G a la izquierda y fabricado con el robot KUKA KR16-2 a la derecha.

6.1. Pruebas en el plano XY con broca 4 mm

Se realizó una evaluación estadística de las dimensiones obtenidas con las diferentes interpolaciones. Se calcularon la desviación estándar, el error absoluto y el error relativo, con el objetivo de obtener una respuesta precisa sobre el comportamiento de los distintos movimientos realizados por el robot.

Tabla 3

Dimensiones de las figuras con fresado

Figure	Designed dimensions	Obtained dimensions
circle	r = 25 mm	r = 25.15 mm
Octagon	50 mm between each side	49.62 mm between each side
Dodecagon	50 mm between each side	49.25 mm between each side

Con los datos tabulados en la tabla 3 creó un diagrama de cajas para realizar un análisis

estadístico en la tabla 4 para evaluar la consistencia del proceso de fabricación.

Tabla 4

Resumen estadístico de las pruebas realizadas.

	$\bar{x}$	e_a	e_r (%)
	25.15	0.15	0.6
	49.62	0.38	0.76
	49.25	0.75	1.5
average		0.42	0.95

Nota: Los valores de e_a y e_r corresponden a los errores absoluto y relativo, respectivamente, mientras que σ representa la desviación estándar.

6.2. Pruebas en YZ con broca 3 mm

Para la prueba en YZ con fresa de 3 mm, se efectuó una modificación correspondiente a la herramienta del spindle, se usó una fresa para corte de madera, se realizó un triángulo, como el de la figura 21, un hexágono como el de la figura 22, y un octadecágono como el de la figura 23.

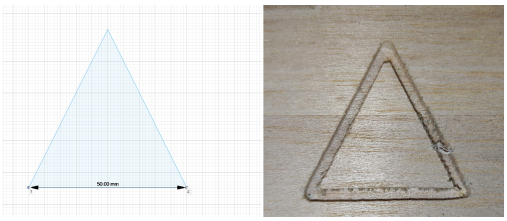


Figura 21

Triángulo generado con código G a la izquierda y fabricado con el robot KUKA KR16-2 a la derecha.

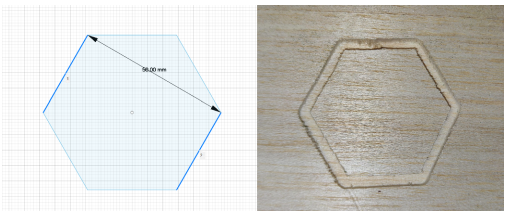


Figura 22

Hexágono generado con código G a la izquierda y fabricado con el robot KUKA KR16-2 a la derecha.

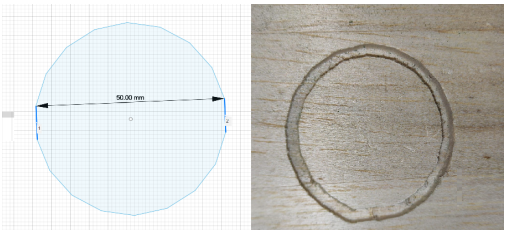


Figura 23

Octadecágono generado con código G a la izquierda y fabricado con el robot KUKA KR16-2 a la derecha.

Tabla 5

Dimensiones de las figuras con fresado

Figure	**Designed dimensions**	**Obtained dimensions**
Triangle	50 mm triangle base	49.15 mm triangle base
Hexagon	50 mm between each side	49.91 mm between each side
Octadecagon	50 mm between each side	50.45 mm between each side

Con los datos tabulados en la tabla 5, se creó un diagrama de cajas para realizar un análisis estadístico para evaluar la consistencia del proceso de fabricación.

Tabla 6

Resumen estadístico de las pruebas realizadas fresa 4 milímetros.

	$\bar{x}$	e_a	e_r (%)
	49.15	0.85	1.7
	49.91	0.09	0.18
	49.25	0.45	0.9
average		0.4633	0.92

Nota: Los valores de e_a y e_r corresponden a los errores absoluto y relativo, respectivamente, mientras que σ representa la desviación estándar.

En este caso el error absoluto en promedio es 0.4633 como se observa en la tabla 6.

6.3. Pruebas 3d pirámide escalonada

La fabricación de la pirámide escalonada se inició con un planeado del material de espuma para establecer una superficie de referencia homogénea. Mediante el software Fusion 360 se realizó un planeado, luego se realizaron varios de contorneados desde la parte superior hasta la base. Después de cada conjunto de contorneados, se ejecutaron operaciones de planeado para refinar las superficies y asegurar la precisión dimensional según las especificaciones del diseño original y el resultado se observa en la figura 24.

Figura 24

Pirámide escalonada fabricada con el robot KUKA KR16-2.

Mediante el software nc viewer se puede observar las trayectorias realizadas con el código G generado en la figura 25.

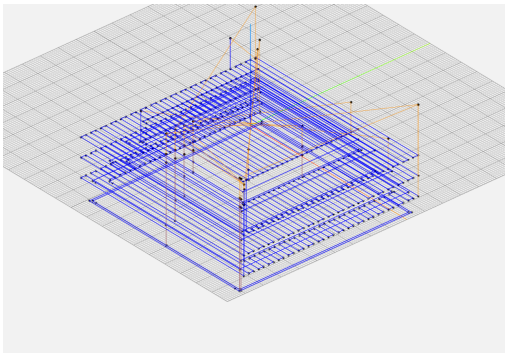


Figura 25

Trayectoria generada por código G vista en el software NC Viewer.

Las instrucciones para el fresado se codificaron en formato G, adaptadas específicamente para la compatibilidad con el robot KUKA KR16-2. Esta codificación permitió la ejecución automatizada y precisa de todas las fases de mecanizado, resultando en una pirámide con escalones claramente definidos y alineación precisa sin mayores desviaciones del diseño planeado, como se observa en la figura 26.

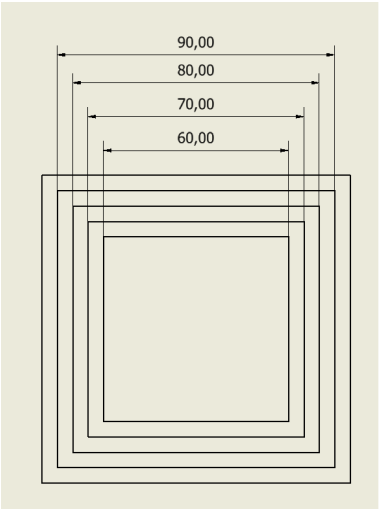


Figura 26

Cotas pirámide escalonada.

Con los datos obtenidos en la tabla 7, se realiza un resumen estadístico en la tabla 8.

Tabla 7

Dimensiones de las figuras con fresado

Figure	Designed dimensions	Obtained dimensions
Step 1	60 × 60 mm	59.6 × 59.4 mm
Step 2	70 × 70 mm	69.56 × 70.05 mm
Step 3	80 × 80 mm	80.5 × 79.56 mm
Step 4	90 × 90 mm	90.06 × 89.22 mm

Tabla 8

Resumen estadístico de las pruebas realizadas.

	$\bar{x}$	e_a	e_r (%)
	59.5	0.5	0.83
	69.8	0.2	0.28
	80.03	0.03	0.0375
	89.64	0.36	0.4
average		0.27	0.38

Nota: Los valores de e_a y e_r corresponden a los errores absoluto y relativo, respectivamente, mientras que σ representa la desviación estándar.

El error absoluto de la pieza fue de 0.27 milímetros, como se observa en la tabla 8.

6.4. Pruebas 3d planeado y fresado adaptativo circular

Para esta prueba se usó el mismo material que el de la pirámide escalonada, las compensaciones se repiten, además se realizó fresados y se midió el radio de la figura fabricada, en este proceso se realizó un planeado con la broca de cuatro milímetros, la trayectoria generada en código G se observa en la figura 27, con las medidas nominales en la figura 28.

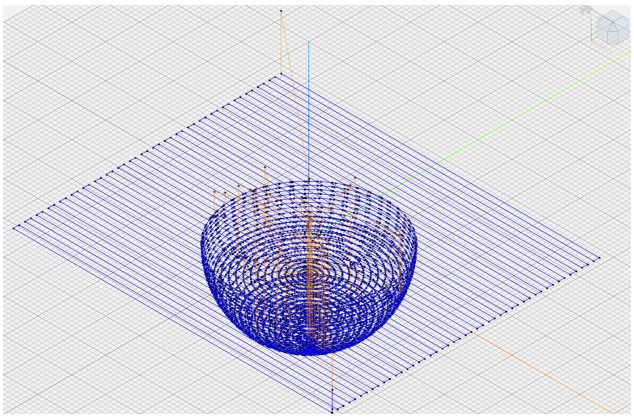


Figura 27

Ruta herramienta en figura con fresado adaptativo circular.

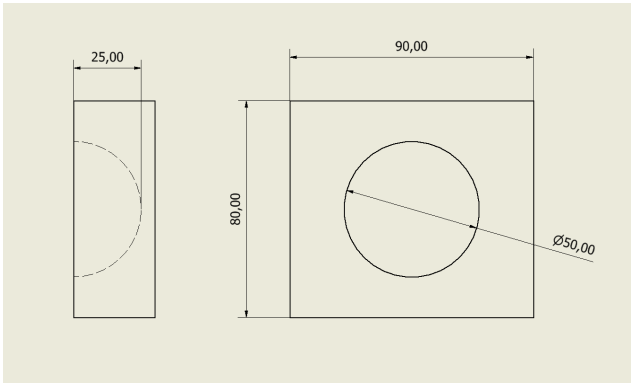


Figura 28

Cotas figura planeado y fresado adaptativo circular.

Realizada la ruta, se generó un programa en código KRL para imitar la ruta con el robot KUKA KR16-2, se fabricó la figura como se observa en la figura 29.

Figura 29

fresado adaptativo circular, fabricado con el robot KUKA KR16-2.

Con la figura fabricada se realizó mediciones de las medidas de la figura Final en la tabla 9.

Tabla 9

Dimensiones de las figuras con fresado

Figure	**Designed dimensions**	**Obtained dimensions**
Outside diameter	50 millimeters	50.53 millimeters
Interior radio	25 millimeters	24.64 millimeters

Con los datos obtenidos se realiza un resumen estadístico visto en la tabla 10.

Tabla 10

Resumen estadístico de las pruebas realizadas.

	$\bar{x}$	e_a	e_r (%)
	50.53	0.53	1.06
	24.94	0.36	0.24
average		0.445	0.65

Nota: Los valores de e_a y e_r corresponden a los errores absoluto y relativo, respectivamente, mientras que σ representa la desviación estándar.

Basándose en las medidas generales, se puede observar que el error en interpolaciones circulares es mayor, esto debido a factores como la perdida de pasos en interpolaciones circulares y vibraciones generadas en la mesa de trabajo.

7. Conclusiones

Mediante el sistema traductor de código G a código KRL el robot KUKA KR16-2 se logró que las trayectorias programadas en un software CAD/CAM (fusion 360) se conviertan en trayectorias en KRL. Se comprobó que el robot puede llevar a cabo tareas de fresado CNC complejas; sin embargo, se tiene que contar con un espacio de trabajo adecuado para evitar vibraciones como se pudo observar en la estadística, las vibraciones ocasionan desfase milimétrico que no superaba los 0.5 milímetros en la mayoría de fresados realizados. Los desfases inevitablemente van a ocasionar una perdida de calidad en los acabados superficiales.

El robot es capaz de mantener la trayectoria ante la aplicación de fuerzas externas, ya que el algoritmo de control nativo del controlador KRC2 compensa automáticamente las desviaciones causadas por dichas cargas. Sin embargo, se ha comprobado que la rigidez de los robots industriales, en general, es inferior a 1 N/µm, mientras que en las máquinas CNC puede ser hasta 50 veces mayor. Esta limitación sugiere el uso de materiales isotrópicos para evitar sobrecargar los motores del robot, priorizando aquellos de baja dureza y fáciles de mecanizar. Es esencial controlar la carga aplicada al robot para evitar exceder sus capacidades mecánicas y asegurar un funcionamiento óptimo.

Con las piezas fabricadas, el robot KUKA KR16-2 ha demostrado la capacidad de aprovechar sus seis grados de libertad, permitiendo realizar trayectorias complejas. A diferencia de las máquinas CNC convencionales, el robot ofrece un espacio de trabajo más amplio, lo que facilita el fresado en un mayor tamaño. Esta ventaja permite realizar trayectorias más variadas y optimizar la disposición del material para conseguir mejores resultados en el mecanizado. Al aprovechar estos grados de libertad, se pueden ejecutar tareas que requieren movimientos más complejos y adaptaciones espaciales, lo que expande las posibilidades de manufactura en comparación con las limitaciones físicas de una máquina CNC tradicional.

8. Recomendaciones

1. **Desalineación de la mesa de trabajo:** La desalineación de la mesa de trabajo provocó errores en los ángulos obtenidos. A pesar de los esfuerzos por alinear la herramienta lo mejor posible, la ausencia de un entorno adecuadamente adaptado para este tipo de mecanizados resultó en errores inevitables.

2. **Precisión de los movimientos:** La capacidad del robot para realizar movimientos precisos está limitada por su resolución, que no permite desplazamientos inferiores a 0.1 milímetros. Esta restricción impide la realización de movimientos micrométricos, lo que compromete la precisión en la creación o mecanizado de piezas.

3. **Vibraciones:** La falta de mesas especializadas durante el proceso de fresado genera vibraciones que afectan la precisión milimétrica del producto final. Además, se detectó que ciertos movimientos del robot KR16-2, especialmente en el eje A1, producen vibraciones involuntarias que contribuyen a las desviaciones observadas. Se recomienda en futuros proyectos diseñar una mesa para fresado.

4. **Posicionamiento de herramienta:** El ángulo del efector final en el robot KUKA KR16-2 esencial para que el trabajo de fresado sea preciso, por lo que es necesario realizar antes de cualquier fresado la calibración del eje final con respecto a la mesa de trabajo.

Referencias

Abdolmalaki, R. Y. (2017). Development of Direct Kinematics and Workspace Representation for Smokie Robot Manipulator the Barret WAM. , 1–7. Descargado de http://arxiv.org/abs/1707.04820

Altintas, Y. (2012). *Manufacturing automation: Metal cutting mechanics, machine tool vibrations, and cnc design* (2.ª ed.). Cambridge University Press.

Braumann, J., y Brell-Çokcan, S. (2011). Parametric robot control: Integrated cad/cam for architectural design. *ACADIA proceedings*. Descargado de https://api.semanticscholar.org/CorpusID:11210942

Chen, Q., Zhang, C., Hu, T., Zhou, Y., Ni, H., y Xue, X. (2022). Posture optimization in robotic machining based on comprehensive deformation index considering spindle weight and cutting force. *Robotics and Computer-Integrated Manufacturing*, *74*, 102290. Descargado de https://www.sciencedirect.com/science/article/pii/S0736584521001708 doi: https://doi.org/10.1016/j.rcim.2021.102290

Corke, P. (2017). *Robotics, Vision and Control*. Cham, Switzerland: Springer International Publishing. Descargado de https://link.springer.com/book/10.1007/978-3-319-54413-7

Deans, M. (2021, mar). *What is CAM (Computer-Aided Manufacturing)? - Fusion Blog.* Descargado 2024-04-09, de https://www.autodesk.com/products/fusion-360/blog/computer-aided-manufacturing-beginners/

Esneca. (2019). *Brazo Robótico: Qué es y en qué Industrias se utiliza?* Descargado 2024-01-25, de https://www.esneca.com/blog/brazo-robotico-industrias/

Grumeber. (2021). *Programación en CNC: ¿Qué es y qué tipos existen? - Grumeber.* Descargado 2024-03-28, de https://grumeber.com/tipos-de-programacion-en-cnc/#:$\sim$:text=Ã¿Qu{é}eslaprogramaci{ó}nCNC,mecanizarpiezasconaltaprecisi{ó}n.

Haascnc. (2021). *Qué son los códigos G.* Descargado 2024-03-28, de https://www.haascnc.com/es/service/service-content/guide-procedures/what-are-g-codes.html

International Federation of Robotics. (2023). *Annual installations of industrial robots 2017-2022 and 2023-2026.* Descargado 2024-09-09, de https://www.aer-automation.com/automation_review/la-instalacion-global-de-robots-industriales-crece-un-5-en-2022/#:\sim:text=La%20instalaci{ó}n%20de%20robots%20industriales%20en%20todo%20el%20mundo%20creci{ó},por%20sus%20siglas%20en%20ingl{é}s).

Klimchik, A., Ambiehl, A., Garnier, S., Furet, B., y Pashkevich, A. (2016, enero). Experimental study of robotic-based machining. *IFAC-PapersOnLine*, *49*(12), 174–179. doi: 10.1016/J.IFACOL.2016.07.591

KUKA. (2020, nov). *KUKA KR C4 | KUKA AG.* Descargado 2024-04-10, de https://www.kuka.com/en-us/products/robotics-systems/robot-controllers/kr-c4

KUKA. (2021, jan). *KUKA.CNC.* Descargado 2024-04-07, de https://www.kuka.com/en-de/products/robot-systems/software/application-software/kuka_cnc

KUKA ROBOTICS. (2020). *KR 16 Manual.* Descargado 2024-09-08, de https://studylib.es/doc/3512009/kr-16âĂŞpdf-.

Pan, J., Fu, Z., Xiong, J., Lei, X., Zhang, K., y Chen, X. (2022, febrero). RobMach: G-Code-based off-line programming for robotic machining trajectory generation. *The International Journal of Advanced Manufacturing Technology*, *118*(7), 2497–2511. Descargado de https://doi.org/10.1007/s00170-021-08082-3 doi: 10.1007/s00170-021-08082-3

RoboDK. (2022, jun). *KUKA robots - RoboDK Documentation.* Descargado 2024-04-10, de https://robodk.com/doc/en/Robots-KUKA.html

Rotella, N. (2021, jan). *Forward Direct Kinematics | Nick Rotella.* Descargado 2024-04-09, de https://nrotella.github.io/journal/forward-direct-kinematics.html

Siciliano, B., y Khatib, O. (2016). *Springer handbook of robotics.* doi: 10.1007/978-3-319-32552-1

Siciliano, B., Sciavicco, L., Villani, L., y Oriolo, G. (2009). *Robotics: Modelling, planning and control.* London: Springer. Descargado de https://doi.org/10.5860/choice.46-6226 doi: 10.5860/choice.46-6226

Smid, P. (2003). *Cnc programming handbook: a comprehensive guide to practical cnc programming.* Industrial Press Inc.

Ubicación universidad politécnica salesiana. (2022).

ANEXOS

Anexo A: Matriz de Consistencia Lógica

Tabla 11

Matriz de consistencia.

MATRIZ DE CONSISTENCIA			
PROBLEMA GENERAL	**OBJETIVO GENERAL**	**VARIABLES**	**MARCO TEÓRICO**
¿Es posible convertir de codigo G a codigo KUKA KRL?	Desarrollar un sistema traductor de codigo G a codigo KRL para adaptar en el robot KUKA KR16-2 en entornos de mecanizado CNC	VD: Diseño de la plataforma	Control numerico computarizado, Robótica, programación, fresado.
PROBLEMAS ESPECÍFICOS	**OBJETIVOS ESPECÍFICOS**	**VARIABLES**	**MARCO TEÓRICO**
¿Cómo influye la dinámica del robot KUKA KR16-2 en los procesos de fresado CNC?	Analizar la dinámica del robot KUKA KR16-2 aplicada a procesos de manufactura por arranque de viruta CNC en la operación de fresado.	VD: Dinamica del robot KUKA KR16-2.	Algoritmos y métodos dinámicos y cinemáticos de analisis de robots.
¿Es posible programar la relacion entre el codigo G y el KRL para tareas de fresado en dos ejes y medio?	Diseñar un sistema traductor entre código G y KRL, garantizando la operatividad del robot KR16-2, permitiendo ejecutar tareas definidas en código G replicando una CNC de dos ejes y medio	VD: Programación. VI: Interfaz entre Python y el KR C2,	Cinemática directa, Cinemática Indirecta, relación entre interpolaciones de codigo G y codigo KRL
¿Cuál es el proceso idóneo para la implementación de un traductor para el robot KUKA KR16-2?	Implementar el sistema traductor en el robot KUKA KR16-2	VD: Mapeo del espacio de trabajo del Robot VI: Codigo KRL resultante	
¿Es comprobable el funcionamiento del traductor en tareas de fresado?	Realizar pruebas del funcionamiento del traductor en el robot KUKA KR16-2.	VD: Interpolaciones del robot	

Nota: La matriz de consistencia presentada facilita identificar la relación que existe entre las variables y los objetivos además de como se relaciona con el marco teórico referencial.

Anexo B: hoja de ruta piramide escalonada

Setup Sheet for Program 1001

Job Description: Configuración3
Document Path: piramide cuadrada v1

Configuración

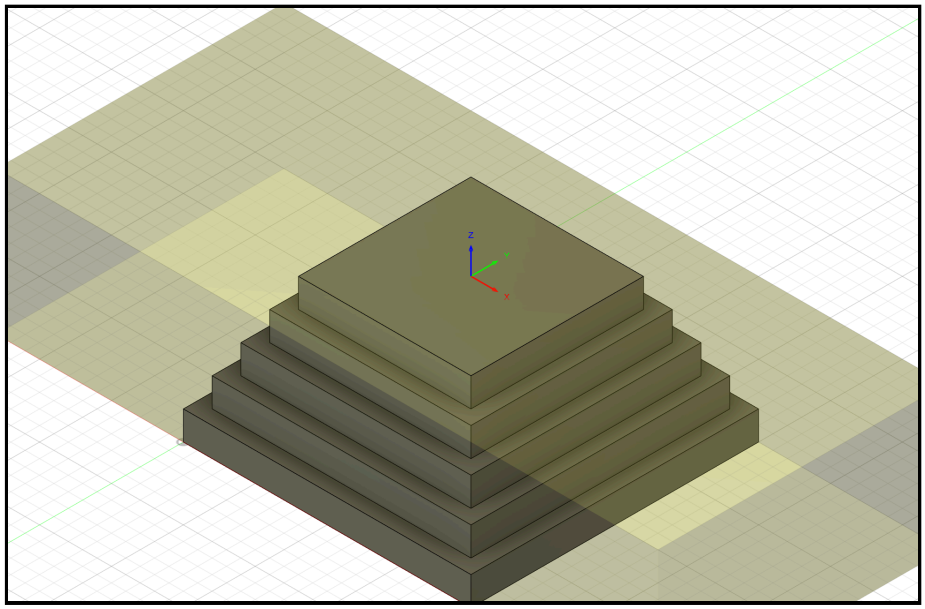
Plano de trabajo: #0

Material:
DX: 230mm
DY: 100mm
DZ: 50mm

Pieza:
DX: 100mm
DY: 100mm
DZ: 50mm

Stock Lower in WCS #0:
X: -115mm
Y: -50mm
Z: -50mm

Stock Upper in WCS #0:
X: 115mm
Y: 50mm
Z: 0mm



Total

Number Of Operations: 10
Number Of Tools: 1
Herramientas: **T1**
Maximum Z: 15mm
Minimum Z: -40mm
Maximum Feedrate: 1000mm/min
Velocidad máxima de husillo: 5000rpm
Cutting Distance: 11192.63mm
Distancia rápida: 1105.95mm
Estimated Cycle Time: 12m:37s

Herramientas

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Minimum Z: -40mm
Maximum Feed: 1000mm/min
Velocidad máxima de husillo: 5000rpm
Cutting Distance: 11192.63mm
Distancia rápida: 1105.95mm
Estimated Cycle Time: 12m:22s (98%)

Operaciones

Operación 1/10
Descripción: Cara2
Estrategia: Facing
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrepasada máxima: 2.8mm

Maximum Z: 15mm
Minimum Z: 0mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 1470.94mm
Distancia rápida: 25mm
Estimated Cycle Time: 1m:29s (11.8%)
Refrigerante: Fluido

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 2/10
Descripción: 2D Contorneado3
Estrategia: Contour 2D
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrematerial: 0mm
Reducción máxima: 5mm
Sobrepasada máxima: 3.8mm

Maximum Z: 15mm
Minimum Z: -10mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 788mm
Distancia rápida: 60mm
Estimated Cycle Time: 50s (6.7%)
Refrigerante: Fluido

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 3/10
Descripción: 2D Contorneado5
Estrategia: Contour 2D
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrematerial: 0mm
Sobrepasada máxima: 3.8mm

Maximum Z: 15mm
Minimum Z: -10mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 279mm
Distancia rápida: 35mm
Estimated Cycle Time: 19s (2.5%)
Refrigerante: Desactivado

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 4/10
Descripción: 2D Contorneado6
Estrategia: Contour 2D
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrematerial: 0mm
Sobrepasada máxima: 3.8mm

Maximum Z: 15mm
Minimum Z: -20mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 632mm
Distancia rápida: 80mm
Estimated Cycle Time: 44s (5.8%)
Refrigerante: Desactivado

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 5/10
Descripción: Cara3
Estrategia: Facing
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrepasada máxima: 2.8mm

Maximum Z: 15mm
Minimum Z: -10mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 2365.17mm
Distancia rápida: 190.61mm
Estimated Cycle Time: 2m:30s (19.8%)
Refrigerante: Desactivado

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 6/10
Descripción: 2D Contorneado7
Estrategia: Contour 2D
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrematerial: 0mm
Sobrepasada máxima: 3.8mm

Maximum Z: 15mm
Minimum Z: -30mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 732mm
Distancia rápida: 100mm
Estimated Cycle Time: 52s (6.9%)
Refrigerante: Desactivado

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 7/10
Descripción: Cara4
Estrategia: Facing
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrepasada máxima: 2.8mm

Maximum Z: 15mm
Minimum Z: -20mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 1913.16mm
Distancia rápida: 174.71mm
Estimated Cycle Time: 2m:3s (16.2%)
Refrigerante: Desactivado

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 8/10
Descripción: 2D Contorneado9
Estrategia: Contour 2D
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrematerial: 0mm
Sobrepasada máxima: 3.8mm

Maximum Z: 15mm
Minimum Z: -40mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 832mm
Distancia rápida: 120mm
Estimated Cycle Time: 1m:1s (8.1%)
Refrigerante: Desactivado

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 9/10
Descripción: Cara5
Estrategia: Facing
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrepasada máxima: 2.8mm

Maximum Z: 15mm
Minimum Z: -30mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 1330.35mm
Distancia rápida: 209.63mm
Estimated Cycle Time: 1m:31s (12%)
Refrigerante: Desactivado

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Operación 10/10
Descripción: 2D Contorneado10
Estrategia: Contour 2D
Plano de trabajo: #0
Tolerancia: 0.01mm
Sobrematerial: 0mm
Sobrepasada máxima: 3.8mm

Maximum Z: 15mm
Minimum Z: -40mm
Velocidad máxima de husillo: 5000rpm
Maximum Feedrate: 1000mm/min
Cutting Distance: 850mm
Distancia rápida: 111mm
Estimated Cycle Time: 1m:3s (8.3%)
Refrigerante: Desactivado

T1 D1 L1
Tipo: fresa con punta plana
Diámetro: 4mm
Longitud: 24mm
Flutes: 4

Recalculado por Fusion CAM 2.0.19994 Thursday, September 05, 2024 13:48:03

Anexo C: Sistema Traductor de código G a KRL

```
import re

def convert_gcode_lines_to_krl(gcode_lines):
    output_lines = []
    x, y, z = None, None, None  # Inicialización de las variables de
    posición
    position_index = 1  # Inicialización del índice para nombres
    secuenciales

    for line in gcode_lines:
        parts = line.split()
        new_x, new_y, new_z = None, None, None

        for part in parts:
            if 'X' in part:
                new_x = float(part[1:])
            if 'Y' in part:
                new_y = float(part[1:])
            if 'Z' in part:
                new_z = float(part[1:])

        if new_x is not None:
            x = new_x
        if new_y is not None:
            y = new_y

        if new_z is not None:
            z = new_z
            if x is not None and y is not None:
                output_lines.extend(generate_krl(position_index, x, y, z))
                output_lines.append("")
                position_index += 1

```

```
    return output_lines

def generate_krl(index, x, y, z):
    e6pos = f"DECL E6POS XP{index}={{X {x:.3f},Y {y:.3f},Z {z:.3f},
    A 180.000,B 0.000,C 180.000}}"
    fdat = f"DECL FDAT FP{index}={{TOOL_NO 1,BASE_NO 2,
    IPO_FRAME #BASE,POINT2[] \" \",TQ_STATE FALSE}}"
    ldat = f"DECL LDAT LCPDAT{index}={{VEL 1.00000,ACC
    100.000,APO_DIST 1.000,APO_FAC 50.0000,ORI_TYP #VAR,CIRC_TYP
    #BASE,JERK_FAC 50.0000}}"
    return [e6pos, fdat, ldat]

gcode_lines = [
    #ingresar lineas de codigo G depurado
    # Añade más líneas según sea necesario
]

converted_krl_lines = convert_gcode_lines_to_krl(gcode_lines)
output_text = "\n".join(converted_krl_lines)

# Escribir el resultado en un archivo con codificación correcta
with open("KUKA_Declarations.txt", 'w', encoding='utf-8') as file:
    file.write(output_text)

print("Archivo generado correctamente: KUKA_Declarations.txt")

```

Anexo D: Codigo sistema traductor de codigo G a codigo KRL depurador

```
import re

# Read the entire content of the file
with open('/mnt/data/circulomaquinado.txt', 'r') as file:
    file_content = file.read()

# Remove specific patterns like 'F300', 'M05' using regex
updated_content = re.sub(r'F\d+|M\d+', '', file_content)

# Path for the updated file
updated_file_path = '/mnt/data/circulomaquinado_updated.txt'

# Write the cleaned content to a new file
with open(updated_file_path, 'w') as file:
    file.write(updated_content)

```

Anexo E: Codigo matlab Dinamica Newton-Euler

OBTENCIÓN DEL MODELO DINÁMICO

- **Cálculo del Modelo Dinámico:**

Utilizamos uno de los métodos revisados en clase para calcular el modelo dinámico del robot. Este modelo considera las fuerzas y torques que afectan el movimiento del robot, proporcionando una comprensión completa de su comportamiento dinámico.

Asignación de Variables y Parámetros, aquí se definen variables simbólicas para las coordenadas articulares q_i, velocidades angulares $\dot{q}_i$, aceleraciones angulares $\ddot{q}_i$, masas m_i, y la aceleración debida a la gravedad g.

```
%
%% definición de variables
%syms q1 q2 q3 q4 q5 q6 %Simbolico
% q1 = deg2rad(0);        %grado de articulaciones en radianes
% q2 = deg2rad(-45);
% q3 = deg2rad(45);
q4 = deg2rad(0);
q5 = deg2rad(0);
q6 = deg2rad(0);
qp4 = deg2rad(0);
qp5 = deg2rad(0);
qp6 = deg2rad(0);
qpp4 = deg2rad(0);
qpp5 = deg2rad(0);
qpp6 = deg2rad(0);

%logitudes de eslabones en [m]

l1 = 0.675;         % [m]    % Medido del Suelo con respecto al suelo
m1 = 70.93848;    % [kg]
l2 = 0.260;         % [m]    % Offset de la base con la 2da articualcion
m2 = 20.07544;    % [kg]
l3 = 0.680;         % [m]    % distancia de la articulacion 2 y 3
m3 = 13.47648;    % [kg]
l4 = 0.035;          % [m]    % Offset de la aticulacion 3 y 5
m4 = 12.05601;    % [kg]
l5 = 0.500;         % [m]    % l5 y l6, es medido de la articulacion 3 a 5
m5 = 2.43852;     % [kg]
l6 = 0.210;         % [m]
l7 = 0.118;         % [m]    % la distanci de la articulacion 4 a 6
m6 = 1.74157;     % [kg]

g = 9.81;

syms q1 q2 q3  qp1 qp2 qp3  qpp1 qpp2 qpp3 %Simbolico
%q4 q5 q6 qp4 qp5 qp6 qpp4 qpp5 qpp6
```

```
disp('Constantes creadas')

%% Matrices Rotacion

    A01 = [cos(q1) 0 -sin(q1);sin(q1) 0 cos(q1); 0 -1 0];
    A12 = RotZ(q2)*RotX(0);
    A23 = [cos(q3-(pi/2)) 0 -sin(q3-(pi/2));sin(q3-(pi/2)) 0 cos(q3-(pi/2)); 0 -1
0];
    A34 = RotZ(q4)*RotX(pi/2);
    A45 = RotZ(q5)*RotX(-pi/2);
    A56 = RotZ(q6)*RotX(0);

A06 = simplify(A01*A12*A23*A34*A45*A56);

% Matrices de Rotación Inversas

A10 = simplify(inv(A01));
A21 = simplify(inv(A12));
A32 = inv(A23);
A43 = inv(A34);
A54 = inv(A45);
A65 = inv(A56);

A60 = simplify(inv(A06));

A67 = eye(3);
A76 = inv(A67);

    disp('Paso 1 ejecutado')
    disp('Paso 2 ejecutado')

    %% Paso 3: Condiciones Iniciales
w0=[0 0 0].';
wp0=[0 0 0].';
v0=[0 0 0].';
vp0=[0 0 -g].';

%Posicion de los sistemas Si

p1 = [0 l2 l1].';
p2 = [l3 0 0].';
p3 = [-l4 0 0].';
p4 = [0 0 (l5+l6)].';
p5 = [0 0 0].';
p6 = [0 0 l7].';

% Coordenadas del centro del masa con respecto observadas en solidworks

S1 = [0 -0.02157 0.27847].';
S2 = [-0.37163 0 0].';
S3 = [0.01284 0 0].';
S4 = [0 0 -0.08135].';
S5 = [0 0 0].';
S6 = [0 0 -0.10598].';

% vector unitario eje de movimiento
```

```
z0=[0 0 1].';
f7=[0 0 0].';
n7=[0 0 0].';

disp('Paso 3 ejecutado')

%% PASO 4: velocidad angular

w1 = A10 * (w0 + z0 * qp1);
w2 = A21 * (w1 + z0 * qp2);
w3 = A32 * (w2 + z0 * qp3);
w4 = A43 * (w3 + z0 * qp4);
w5 = A54 * (w4 + z0 * qp5);
w6 = A65 * (w5 + z0 * qp6);

disp('Paso 4 ejecutado')

%% PASO 5: Aceleración angular

wp1 = A10 * (wp0 + z0 * qpp1) + cross(w0, z0 * qp1);
wp2 = A21 * (wp1 + z0 * qpp2) + cross(w1, z0 * qp2);
wp3 = A32 * (wp2 + z0 * qpp3) + cross(w2, z0 * qp3);
wp4 = A43 * (wp3 + z0 * qpp4) + cross(w3, z0 * qp4);
wp5 = A54 * (wp4 + z0 * qpp5) + cross(w4, z0 * qp5);
wp6 = A65 * (wp5 + z0 * qpp6) + cross(w5, z0 * qp6);

disp('Paso 5 ejecutado')

%% PASO 6: Aceleración lineal

vp1=cross(wp1,p1) + cross(w1,cross(w1,p1)) + A10*vp0;
vp2=cross(wp2,p2) + cross(w2,cross(w2,p2)) + A21*vp1;
vp3=cross(wp3,p3) + cross(w3,cross(w3,p3)) + A32*vp2;
vp4=cross(wp4,p4) + cross(w4,cross(w4,p4)) + A43*vp3;
vp5=cross(wp5,p5) + cross(w5,cross(w5,p5)) + A54*vp4;
vp6=cross(wp6,p6) + cross(w6,cross(w6,p6)) + A65*vp5;

disp('Paso 6 ejecutado')

%% PASO 7: aceleración lineal respecto al centro de gravedad

acc1 = cross(wp1,S1) + cross(w1,cross(w1,S1)) + vp1;
acc2 = cross(wp2,S2) + cross(w2,cross(w2,S2)) + vp2;
acc3 = cross(wp3,S3) + cross(w3,cross(w3,S3)) + vp3;
acc4 = cross(wp4,S4) + cross(w4,cross(w4,S4)) + vp4;
acc5 = cross(wp5,S5) + cross(w5,cross(w5,S5)) + vp5;
acc6 = cross(wp6,S6) + cross(w6,cross(w6,S6)) + vp6;

disp('Paso 7 ejecutado')

%% PASO 8: fuerza ejercida sobre cada eslabón

f6 = simplify(( A76 * f7 ) + ( m6 * acc6));
f5 = simplify(( A65 * f6 ) + ( m5 * acc5));
f4 = simplify(( A54 * f5 ) + ( m4 * acc4));
f3 = simplify(( A43 * f4 ) + ( m3 * acc3));
f2 = simplify(( A32 * f3 ) + ( m2 * acc2));
f1 = simplify(( A21 * f2 ) + ( m1 * acc1));
```

```
disp('Paso 8 ejecutado')

%% PASO 9: torque ejercido sobre cada eslabón
% % los momentos de inercia respecto al centro de gravedad
% % Caso de masas concentradas

% I1 = zeros(3);
% I2 = zeros(3);
% I3 = zeros(3);
% I4 = zeros(3);
% I5 = zeros(3);
% I6 = zeros(3);

I1 = [5.08 0.022 -0.044; 0.022 2.61 -0.92;-0.044 -0.92 4.27];
I2 = [0.96 -0.0069 0.708; -0.0069 1.68 -0.0026; 0.70 -0.0026 0.97];
I3 = [1.87 0.0011 -0.000053098; 0.0011 0.096 0.252; -0.000053098 0.252 1.83];
I4 = [0.061 -0.052 -0.1; -0.052 2.06 0.005; -0.1 0.005 2.06];
I5 = [0.043 0.000002653 -0.000002584; 0.000002653 0.043 -0.0029;-0.000002584 -
0.0029 0.0044];
I6 = [0.012 -0.000022834 -0.004;-0.000022834 0.021 0.000037031;-0.004 0.000037031
0.0114];

n6 = simplify( A76 * ( n7 + cross( A67 * p6 , f7 )) + cross( p6 + S6, m6 *acc6) +
I6 * wp6 + cross(w6,I6*w6));
n5 = simplify( A65 * ( n6 + cross( A56 * p5 , f6 )) + cross( p5 + S5, m5 *acc5) +
I5 * wp5 + cross(w5,I5*w5));
n4 = simplify( A54 * ( n5 + cross( A45 * p4 , f5 )) + cross( p4 + S4, m4 *acc4) +
I4 * wp4 + cross(w4,I4*w4));
n3 = simplify( A43 * ( n4 + cross( A34 * p3 , f4 )) + cross( p3 + S3, m3 *acc3) +
I3 * wp3 + cross(w3,I3*w3));
n2 = simplify( A32 * ( n3 + cross( A23 * p2 , f3 )) + cross( p2 + S2, m2 *acc2) +
I2 * wp2 + cross(w2,I2*w2));
n1 = simplify( A21 * ( n2 + cross( A12 * p1 , f2 )) + cross( p1 + S1, m1 *acc1) +
I1 * wp1 + cross(w1,I1*w1));

disp('Paso 9 ejecutado')

%% Paso 10: Torque sobre cada articulación

tau6 = simplify((n6.')*A65*z0);
tau5 = simplify((n5.')*A54*z0);
tau4 = simplify((n4.')*A43*z0);
tau3 = simplify((n3.')*A32*z0);
tau2 = simplify((n2.')*A21*z0);
tau1 = simplify((n1.')*A10*z0);

disp('Paso 10 ejecutado')
```

- **Obtención de la trayectoria**

```
clc; clear all; close all

%% definición de variables
%syms q1 q2 q3 q4 q5 q6 %Simbolico
% q1 = deg2rad(0);        %grado de articulaciones en radianes
% q2 = deg2rad(-45);
% q3 = deg2rad(45);
q4 = deg2rad(0);
```

```
q5 = deg2rad(0);
q6 = deg2rad(0);
qp4 = deg2rad(0);
qp5 = deg2rad(0);
qp6 = deg2rad(0);
qpp4 = deg2rad(0);
qpp5 = deg2rad(0);
qpp6 = deg2rad(0);


%logitudes de eslabones en [m]

l1 = 0.675;        % [m]    % Medido del Suelo con respecto al suelo
m1 = 70.93848;   % [kg]
l2 = 0.260;        % [m]    % Offset de la base con la 2da articualcion
m2 = 20.07544;   % [kg]
l3 = 0.680;        % [m]    % distancia de la articulacion 2 y 3
m3 = 13.47648;   % [kg]
l4 = 0.035;         % [m]    % Offset de la aticulacion 3 y 5
m4 = 12.05601;   % [kg]
l5 = 0.500;        % [m]    % l5 y l6, es medido de la articulacion 3 a 5
m5 = 2.43852;    % [kg]
l6 = 0.210;        % [m]
l7 = 0.118;        % [m]    % la distanci de la articulacion 4 a 6
m6 = 1.74157;    % [kg]

g = 9.81;

syms q1 q2 q3  qp1 qp2 qp3  qpp1 qpp2 qpp3 %Simbolico
%q4 q5 q6 qp4 qp5 qp6 qpp4 qpp5 qpp6


disp('Constantes creadas')

%% Matrices Homogenea
% tomando encuenta la estencion maxima del robot

    A01 = RotZp63(-q1)*trap63(0,0,l1)*trap63(l2,0,0)*RotXp63(-pi/2);
    A12 = RotZp63(q2)*trap63(0,0,0)*trap63(l3,0,0)*RotXp63(0);
    A23 = RotZp63(q3-(pi/2))*trap63(0,0,0)*trap63(-l4,0,0)*RotXp63(-pi/2);
    A34 = RotZp63(q4)*trap63(0,0,(l6+l5))*trap63(0,0,0)*RotXp63(pi/2);
    A45 = RotZp63(q5)*trap63(0,0,0)*trap63(0,0,0)*RotXp63(-pi/2);
    A56 = RotZp63(q6)*trap63(0,0,l7)*trap63(0,0,0)*RotXp63(0);

    %matrices con respecto al origen

    A02 = simplify(A01*A12);
    A03 = simplify(A01*A12*A23);
    A04 = simplify(A01*A12*A23*A34);
    A05 = simplify(A01*A12*A23*A34*A45);
    A06 = simplify(A01*A12*A23*A34*A45*A56);

    disp('Paso 1 ejecutado')
    disp('Paso 2 ejecutado')
%% definir movimientos
N=1000;                      % numero de puntos por periodo
t = 0:1/N:5;                % tiempo de simulación
amplitud_grados1 = 180;              % amplitud de la onda en grados
amplitud_grados2 = -35;
```

```
amplitud_grados3 = 155;
amplitud_grados4 = 0;
amplitud_grados5 = 0;
amplitud_grados6 = 0;

% Modificación para variar la onda de -60 a 60 grados
% Convertir amplitud a radianes
amplitud_radianes1 = deg2rad(amplitud_grados1);
amplitud_radianes2 = deg2rad(amplitud_grados2);
amplitud_radianes3 = deg2rad(amplitud_grados3);
amplitud_radianes4 = deg2rad(amplitud_grados4);
amplitud_radianes5 = deg2rad(amplitud_grados5);
amplitud_radianes6 = deg2rad(amplitud_grados6);

% Valores variables primera articulación

Q1 = amplitud_radianes1*sin(2*pi*t);   % Q en grados
Qp1 = diff(Q1)*N*1;           % velocidad angular de q1
Qpp1 = diff(Qp1)*N*1;         % aceleración angular de q1

% Valores variables segunda articulación

Q2 = amplitud_radianes2*sin(2*pi*t);   % Q en grados
Qp2 = diff(Q2)*N*1;           % velocidad angular de q2
Qpp2 = diff(Qp2)*N*1;         % aceleración angular de q2

% Valores variables tercera articulación

Q3 = amplitud_radianes3*sin(2*pi*t);   % Q en grados
Qp3 = diff(Q3)*N*1;           % velocidad angular de q3
Qpp3 = diff(Qp3)*N*1;         % aceleración angular de q3

% Valores variables cuarta articulación

Q4 = amplitud_radianes4*sin(2*pi*t);   % Q en grados
Qp4 = diff(Q4)*N*1;           % velocidad angular de q4
Qpp4 = diff(Qp4)*N*1;         % aceleración angular de q4

% Valores variables quinta articulación

Q5 = amplitud_radianes5*sin(2*pi*t);   % Q en grados
Qp5 = diff(Q5)*N*1;           % velocidad angular de q2
Qpp5 = diff(Qp5)*N*1;         % aceleración angular de q2

% Valores variables tercera articulación

Q6 = amplitud_radianes6*sin(2*pi*t);   % Q en grados
Qp6 = diff(Q6)*N*1;           % velocidad angular de q3
Qpp6 = diff(Qp6)*N*1;         % aceleración angular de q3

disp('Variables articulares calculadas')

%%
LQ=length(Q1);
for i=1:LQ-2
q1=Q1(i);
qp1=Qp1(i);
qpp1=Qpp1(i);
```

```
q2=Q2(i);
qp2=Qp2(i);
qpp2=Qpp2(i);

q3=Q3(i);
qp3=Qp3(i);
qpp3=Qpp3(i);

q4=Q4(i);
qp4=Qp4(i);
qpp4=Qpp4(i);

q5=Q5(i);
qp5=Qp5(i);
qpp5=Qpp5(i);

q6=Q6(i);
qp6=Qp6(i);
qpp6=Qpp6(i);

% % %
```

$$tau1(i) = 72.0 * qpp1 + 33.0 * cos(2.0 * q2 + q3) - 2.5e-16 * sin(2.0 * q2 + q3) - 49.0 * cos(2.0 * q2) - 50.0 * cos(2.0 * q3) - 190.0 * sin(2.0 * q2) + 38.0 * sin(2.0 * q3) + 70.0 * cos(q3) - 49.0 * cos(2.0 * q2 + 2.0 * q3) - 2.5e-16 * sin(q3) + 61.0 * sin(2.0 * q2 + 2.0 * q3) + 2.9 * qp1^2 * cos(q2) + 10.0 * qp2^2 * cos(q2) - 0.27 * qp3^2 * cos(q2) - 1.7 * qp1^2 * sin(q2) - 3.7e-3 * qp2^2 * sin(q2) - 0.014 * qp3^2 * sin(q2) - 0.62 * qpp1 * cos(2.0 * q2 + q3) + 4.9 * qpp2 * cos(q2 - 1.0 * q3) - 3.4 * qpp2 * cos(q2 - 2.0 * q3) - 3.4 * qpp2 * cos(q2 + 2.0 * q3) + 0.023 * qpp3 * cos(q2 - 1.0 * q3) + 6.7 * qpp1 * sin(2.0 * q2 + q3) - 0.96 * qpp2 * sin(q2 - 1.0 * q3) - 2.6 * qpp2 * sin(q2 - 2.0 * q3) + 4.2 * qpp2 * sin(q2 + 2.0 * q3) - 0.96 * qpp3 * sin(q2 - 1.0 * q3) - 2.0 * qp1^2 * cos(3.0 * q2 + 2.0 * q3) + 1.8 * qp1 * qp2 + 1.8 * qp1 * qp3 - 1.8 * qp1^2 * sin(3.0 * q2 + 2.0 * q3) + 18.0 * qpp1 * cos(2.0 * q2) - 3.2 * qpp1 * cos(2.0 * q3) - 0.36 * qp1^2 * cos(3.0 * q2 + 3.0 * q3) - 5.2 * qpp1 * sin(2.0 * q2) - 5.2 * qpp1 * sin(2.0 * q3) + 0.091 * qp1^2 * sin(3.0 * q2 + 3.0 * q3) - 0.21 * qp1^2 * cos(q2 - 1.0 * q3) - 1.3 * qp1^2 * cos(q2 - 2.0 * q3) - 3.3 * qp1^2 * cos(q2 + 2.0 * q3) + 0.41 * qp2^2 * cos(q2 - 1.0 * q3) - 2.6 * qp2^2 * cos(q2 - 2.0 * q3) - 4.2 * qp2^2 * cos(q2 + 2.0 * q3) + 0.41 * qp3^2 * cos(q2 - 1.0 * q3) + 2.3 * qpp2 * cos(q2 + q3) - 0.037 * qpp3 * cos(q2 + q3) - 2.0 * qp1^2 * sin(q2 - 1.0 * q3) + 1.7 * qp1^2 * sin(q2 - 2.0 * q3) - 3.5 * qp1^2 * sin(q2 + 2.0 * q3) - 4.6 * qp2^2 * sin(q2 - 1.0 * q3) + 3.4 * qp2^2 * sin(q2 - 2.0 * q3) - 3.4 * qp2^2 * sin(q2 + 2.0 * q3) + 0.29 * qp3^2 * sin(q2 - 1.0 * q3) - 0.94 * qpp2 * sin(q2 + q3) - 0.94 * qpp3 * sin(q2 + q3) + 0.21 * qp1^2 * cos(q2 + 3.0 * q3) - 0.3 * qp1^2 * cos(3.0 * q2 + q3) + 0.052 * qp1^2 * sin(q2 + 3.0 * q3) + 1.1 * qp1^2 * sin(3.0 * q2 + q3) - 1.3 * qpp1 * cos(q3) - 7.1 * qpp2 * cos(q2) - 0.24 * qpp3 * cos(q2) - 0.64 * qpp1 * cos(2.0 * q2 - 1.0 * q3) - 7.2 * qpp1 * cos(2.0 * q2 + 2.0 * q3) + 5.5 * qpp1 * sin(q3) - 5.0 * qpp2 * sin(q2) - 0.016 * qpp3 * sin(q2) + 3.8 * qp1^2 * cos(3.0 * q2) + 0.65 * qpp1 * sin(2.0 * q2 - 1.0 * q3) - 5.2 * qpp1 * sin(2.0 * q2 + 2.0 * q3) - 1.7 * qp1^2 * sin(3.0 * q2) + 0.11 * qp1^2 * cos(q2 + q3) + 0.18 * qp2^2 * cos(q2 + q3) + 0.18 * qp3^2 * cos(q2 + q3) + 3.1 * qp1^2 * sin(q2 + q3) + 2.0 * qp2^2 * sin(q2 + q3) - 0.29 * qp3^2 * sin(q2 + q3) - 3.6 * qp1 * qp2 * cos(q2) + 3.9e-3 * qp1 * qp2 * cos(q3) + 0.96 * qp1 * qp3 * cos(q3) - 0.54 * qp2 * qp3 * cos(q2) + 0.65 * qp1 * qp2 * cos(2.0 * q2 - 1.0 * q3) + 1.8 * qp1 * qp2 * cos(2.0 * q2 + 2.0 * q3) - 1.6 * qp1 * qp3 * cos(2.0 * q2 - 1.0 * q3) + 1.8 * qp1 * qp3 * cos(2.0 * q2 + 2.0 * q3) - 3.1 * qp1 * qp2 * sin(q2) - 0.031 * qp1 * qp2 * sin(q3) - 0.047 * qp1 * qp3 * sin(q3) - 0.028 * qp2 * qp3 * sin(q2) + 0.64 * qp1 * qp2 * sin(2.0 * q2 - 1.0 * q3) - 2.9 * qp1 * qp2 * sin(2.0 * q2 + 2.0 * q3) + 5.4e-4 * qp1 * qp3 * sin(2.0 * q2 - 1.0 * q3) - 2.9 * qp1 * qp3 * sin(2.0 * q2 + 2.0 * q3) + 2.0 * qp1 * qp2 * cos(q2 - 1.0 * q3) - 1.8 * qp1 * qp2 * cos(q2 - 2.0 * q3) - 1.8 * qp1 * qp2 * cos(q2 + 2.0 * q3) - 3.8 * qp1 * qp2 * cos(2.0 * q2 + q3) - 2.4 * qp1 * qp3 * cos(2.0 * q2 + q3) + 0.82 * qp2 * qp3 * cos(q2 - 1.0 * q3) + 0.024 * qp1 * qp2 * sin(q2 - 1.0 * q3) - 0.56 * qp1 * qp2 * sin(q2 - 2.0 * q3) + 3.0 * qp1 * qp2 * sin(q2 + 2.0 * q3) + 0.62 * qp1 * qp2 * sin(2.0 * q2 + q3) + 0.013 * qp1 * qp3 * sin(2.0 * q2 + q3) + 0.59 * qp2 * qp3 * sin(q2 - 1.0 * q3) + 1.8 * qp1 * qp2 * cos(2.0 * q2) + 1.8 * qp1 * qp2 * cos(2.0 * q3) + 1.8 * qp1 * qp3 * cos(2.0 * q2) + 1.8 * qp1 * qp3 *$$

$$cos(2.0 * q3) + 2.2 * qp1 * qp2 * sin(2.0 * q2) - 0.62 * qp1 * qp2 * sin(2.0 * q3) + 1.4 * qp1 * qp3 * sin(2.0 * q2) - 0.62 * qp1 * qp3 * sin(2.0 * q3) + 5.1 * qp1 * qp2 * cos(q2 + q3) + 0.36 * qp2 * qp3 * cos(q2 + q3) - 0.024 * qp1 * qp2 * sin(q2 + q3) - 0.59 * qp2 * qp3 * sin(q2 + q3) - 50.0;$$

$$\begin{aligned} tau2(i) = {} & 2.9 * qpp2 + 0.016 * qpp3 - 0.11 * cos(q2 + 2.0 * q3) - 23.0 * sin(q2 + 2.0 * q3) \\ & - 37.0 * sin(q2 + q3) + 61.0 * cos(q2) - 23.0 * sin(q2) + 1.1 * qp1\text{^}2 * cos(q3) \\ & + 2.8 * qp2\text{^}2 * cos(q3) + 0.23 * qp3\text{^}2 * cos(q3) + 0.79 * qp1\text{^}2 * cos(2.0 * q2 \\ & + 2.0 * q3) - 4.5e-17 * qp1\text{^}2 * sin(q3) + 2.0e-17 * qp2\text{^}2 * sin(q3) + 2.0e \\ & - 17 * qp3\text{^}2 * sin(q3) + 7.1e-3 * qpp1 * cos(q2 - 1.0 * q3) + 4.0 * qpp1 * cos(q2 \\ & + 2.0 * q3) - 0.012 * qp1\text{^}2 * sin(2.0 * q2 + 2.0 * q3) - 1.8e-17 * qpp1 * sin(q2 \\ & - 1.0 * q3) + 0.051 * qpp1 * sin(q2 + 2.0 * q3) + 0.58 * qp1\text{^}2 * cos(2.0 * q2 + 3.0 \\ & * q3) - 0.7 * qp1 * qp2 + 0.028 * qp2 * qp3 - 0.039 * qp1\text{^}2 * sin(2.0 * q2 + 3.0 \\ & * q3) - 7.8e-3 * qpp2 * cos(2.0 * q3) - 1.6 * qpp2 * sin(2.0 * q3) + 0.83 * qp1\text{^}2 \\ & * cos(2.0 * q2 + q3) - 0.56 * qpp1 * cos(q2 + q3) + 9.3e-3 * qp1\text{^}2 * sin(2.0 * q2 \\ & + q3) + 5.3e-17 * qpp1 * sin(q2 + q3) + 0.79 * qp1\text{^}2 * cos(2.0 * q2) + 0.79 \\ & * qp1\text{^}2 * cos(2.0 * q3) + 1.6 * qp2\text{^}2 * cos(2.0 * q3) + 3.7 * qpp1 * cos(q2) + 0.06 \\ & * qpp2 * cos(q3) + 0.06 * qpp3 * cos(q3) + 1.3 * qp1\text{^}2 * sin(2.0 * q2) - 3.9e-3 \\ & * qp1\text{^}2 * sin(2.0 * q3) - 7.8e-3 * qp2\text{^}2 * sin(2.0 * q3) + 3.5 * qpp1 * sin(q2) \\ & - 0.69 * qpp2 * sin(q3) + 1.9 * qpp3 * sin(q3) + 0.79 * qp1\text{^}2 + 1.6 * qp2\text{^}2 \\ & + 0.014 * qp3\text{^}2 + 0.77 * qp1 * qp2 * cos(q2) + 3.5e-17 * qp1 * qp2 * cos(q3) \\ & + 3.2 * qp1 * qp3 * cos(q2) + 0.46 * qp2 * qp3 * cos(q3) + 2.0 * qp1 * qp2 * sin(q2) \\ & + 3.1 * qp1 * qp2 * sin(q3) + 2.0 * qp1 * qp3 * sin(q2) + 4.0e-17 * qp2 * qp3 \\ & * sin(q3) - 1.8e-17 * qp1 * qp2 * cos(q2 - 1.0 * q3) - 3.6e-3 * qp1 * qp2 \\ & * cos(q2 + 2.0 * q3) - 3.3 * qp1 * qp3 * cos(q2 - 1.0 * q3) - 3.6e-3 * qp1 * qp3 \\ & * cos(q2 + 2.0 * q3) - 7.1e-3 * qp1 * qp2 * sin(q2 - 1.0 * q3) + 2.3 * qp1 * qp2 \\ & * sin(q2 + 2.0 * q3) + 1.1e-3 * qp1 * qp3 * sin(q2 - 1.0 * q3) + 2.3 * qp1 * qp3 \\ & * sin(q2 + 2.0 * q3) + 0.059 * qp1 * qp2 * cos(2.0 * q3) - 2.5 * qp1 * qp2 * sin(2.0 \\ & * q3) + 5.3e-17 * qp1 * qp2 * cos(q2 + q3) - 1.4 * qp1 * qp3 * cos(q2 + q3) \\ & - 3.1 * qp1 * qp2 * sin(q2 + q3) - 3.2 * qp1 * qp3 * sin(q2 + q3); \end{aligned}$$

$$\begin{aligned} tau3(i) = {} & 0.21 * cos(q2 + q3) - 1.9 * qpp3 - 1.9 * qpp2 - 0.23 * sin(q2 + q3) + 7.8e-3 \\ & * qp1\text{^}2 * cos(q3) + 0.016 * qp2\text{^}2 * cos(q3) + 0.049 * qp1\text{^}2 * cos(2.0 * q2 + 2.0 \\ & * q3) + 7.1e-3 * qp1\text{^}2 * sin(q3) + 0.014 * qp2\text{^}2 * sin(q3) + 1.0 * qp1\text{^}2 \\ & * sin(2.0 * q2 + 2.0 * q3) - 4.0e-17 * qp2 * qp3 + 7.8e-3 * qp1\text{^}2 * cos(2.0 * q2 \\ & + q3) - 0.047 * qpp1 * cos(q2 + q3) + 7.1e-3 * qp1\text{^}2 * sin(2.0 * q2 + q3) \\ & + 0.28 * qpp1 * sin(q2 + q3) - 3.5e-17 * qpp1 * cos(q2) + 0.014 * qpp2 \\ & * cos(q3) - 0.016 * qpp2 * sin(q3) + 4.5e-17 * qp1\text{^}2 - 2.0e-17 * qp2\text{^}2 - 2.0e \\ & - 17 * qp3\text{^}2 + 0.27 * qp1 * qp2 * cos(q3) + 0.062 * qp1 * qp3 * cos(q2) + 3.5e \\ & - 17 * qp1 * qp2 * sin(q2) + 0.062 * qp1 * qp2 * sin(q3) + 1.9 * qp1 * qp3 * sin(q2) \\ & - 0.27 * qp1 * qp2 * cos(q2 + q3) - 0.27 * qp1 * qp3 * cos(q2 + q3) - 0.062 * qp1 \\ & * qp2 * sin(q2 + q3) - 0.062 * qp1 * qp3 * sin(q2 + q3); \end{aligned}$$

$$\begin{aligned} tau4(i) = {} & 0.54 * qp2 * qp3 - 0.013 * qpp3 - 100.0 * cos(q2 + q3) - 0.13 * qp1\text{^}2 * cos(2.0 * q2 \\ & + 2.0 * q3) - 3.6 * qp1\text{^}2 * sin(q3) - 7.2 * qp2\text{^}2 * sin(q3) - 6.4e-3 * qp1\text{^}2 \\ & * sin(2.0 * q2 + 2.0 * q3) - 0.013 * qpp2 - 0.048 * qpp1 * cos(q2 + q3) - 3.6 \\ & * qp1\text{^}2 * sin(2.0 * q2 + q3) - 16.0 * qpp1 * sin(q2 + q3) - 7.2 * qpp2 * cos(q3) \\ & + 0.23 * qp1\text{^}2 + 0.27 * qp2\text{^}2 + 0.27 * qp3\text{^}2 - 8.9 * qp1 * qp2 * cos(q3) + 0.048 \\ & * qp1 * qp3 * cos(q2) + 0.048 * qp1 * qp2 * sin(q3) + 0.013 * qp1 * qp3 * sin(q2) \\ & + 4.9 * qp1 * qp2 * cos(q2 + q3) + 4.9 * qp1 * qp3 * cos(q2 + q3) + 0.048 * qp1 \\ & * qp2 * sin(q2 + q3) + 0.048 * qp1 * qp3 * sin(q2 + q3); \end{aligned}$$

$$tau5(i) = 0.064 * qpp2 + 0.064 * qpp3 + 0.21 * cos(q2 + q3) - 3.6e-3 * qp1\text{^}2 * cos(2.0 * q2 + 2.0 * q3) + 7.1e-3 * qp1\text{^}2 * sin(q3) + 0.014 * qp2\text{^}2 * sin(q3) + 0.02 * qp1\text{^}2 * sin(2.0 * q2 + 2.0 * q3) - 1.5e-3 * qp2 * qp3 + 2.0e-5 * qpp1 * cos(q2 + q3) + 7.1e-3 * qp1\text{^}2 * sin(2.0 * q2 + q3) + 0.026 * qpp1 * sin(q2 + q3) + 0.014 * qpp2 * cos(q3) - 3.7e-4 * qp1\text{^}2 - 7.3e-4 * qp2\text{^}2 - 7.3e-4 * qp3\text{^}2 + 0.012 * qp1 * qp2 * cos(q3) - 2.0e-5 * qp1 * qp3 * cos(q2) - 2.0e-5 * qp1 * qp2 * sin(q3) - 0.064 * qp1 * qp3 * sin(q2) - 0.012 * qp1 * qp2 * cos(q2 + q3) - 0.012 * qp1 * qp3 * cos(q2 + q3) + 2.0e-5 * qp1 * qp2 * sin(q2 + q3) + 2.0e-5 * qp1 * qp3 * sin(q2 + q3);$$

$$tau6(i) = 4.6e-5 * qp2 * qp3 - 3.7e-5 * qpp3 - 1.1e-5 * qp1\text{^}2 * cos(2.0 * q2 + 2.0 * q3) - 1.9e-5 * qp1\text{^}2 * sin(2.0 * q2 + 2.0 * q3) - 3.7e-5 * qpp2 - 4.0e-3 * qpp1 * cos(q2 + q3) - 0.011 * qpp1 * sin(q2 + q3) - 1.1e-5 * qp1\text{^}2 + 2.3e-5 * qp2\text{^}2 + 2.3e-5 * qp3\text{^}2 - 0.011 * qp1 * qp2 * cos(q3) + 4.0e-3 * qp1 * qp3 * cos(q2) + 4.0e-3 * qp1 * qp2 * sin(q3) + 3.7e-5 * qp1 * qp3 * sin(q2) - 9.0e-3 * qp1 * qp2 * cos(q2 + q3) - 9.0e-3 * qp1 * qp3 * cos(q2 + q3) + 4.0e-3 * qp1 * qp2 * sin(q2 + q3) + 4.0e-3 * qp1 * qp3 * sin(q2 + q3);$$

```
end
disp('Simulación ejecutada')
%% Generación de gráficas resultados
figure(1)
subplot(2,1,1)
plot(t,Q1)
grid on; xlabel('t');ylabel('q_1(t)'); title('Articulación 1')
subplot(2,1,2)
plot(t(1,1:LQ-2),tau1)
grid on; xlabel('t');ylabel('torque \tau_1(t)');

%%
figure(2)
subplot(2,1,1)
plot(t,Q2)
grid on; xlabel('t');ylabel('q_2(t)'); title('Articulación 2')
subplot(2,1,2)
plot(t(1,1:LQ-2),tau2)
grid on, xlabel('t');ylabel('torque \tau_2(t)');

%%
figure(3)
subplot(2,1,1)
plot(t,Q3)
grid on; xlabel('t');ylabel('q_3(t)'); title('Articulación 3')
subplot(2,1,2)
plot(t(1,1:LQ-2),tau3)
grid on, xlabel('t');ylabel('torque \tau_3(t)');

%%
figure(4)
subplot(2,1,1)
plot(t,Q4)
grid on; xlabel('t');ylabel('q_4(t)'); title('Articulación 4')
subplot(2,1,2)
plot(t(1,1:LQ-2),tau4)
grid on, xlabel('t');ylabel('torque \tau_4(t)');

%%
figure(5)
```

```
subplot(2,1,1)
plot(t,Q5)
grid on; xlabel('t');ylabel('q_5(t)'); title('Articulación 5')
subplot(2,1,2)
plot(t(1,1:LQ-2),tau5)
grid on, xlabel('t');ylabel('torque \tau_5(t)');

%%
figure(6)
subplot(2,1,1)
plot(t,Q6)
grid on; xlabel('t');ylabel('q_6(t)'); title('Articulación 6')
subplot(2,1,2)
plot(t(1,1:LQ-2),tau6)
grid on, xlabel('t');ylabel('torque \tau_6(t)');

%% Ecuaciones Cinematicas
Ecu=vpa(A06,2)*[0;0;0;1];
x=260.*cos(Q1) + 0.00000000000005070037748679935243239016995353 6.*sin(Q1) +
680.*cos(Q1).*cos(Q2) +
0.00000000000004163799116809835943451602313647 4.*sin(Q1).*sin(Q2) - 35.*cos(Q3 -
1.570796326734125614166259765625).*(cos(Q1).*cos(Q2) +
0.00000000000000006123233995384609899612009622 5396.*sin(Q1).*sin(Q2)) -
828.*cos(Q3 - 1.570796326734125614166259765625).*(cos(Q1).*sin(Q2) -
0.00000000000000006123233995384609899612009622 5396.*cos(Q2).*sin(Q1)) -
828.*sin(Q3 - 1.570796326734125614166259765625).*(cos(Q1).*cos(Q2) +
0.00000000000000006123233995384609899612009622 5396.*sin(Q1).*sin(Q2)) + 35.*sin(Q3
- 1.570796326734125614166259765625).*(cos(Q1).*sin(Q2) -
0.00000000000000006123233995384609899612009622 5396.*cos(Q2).*sin(Q1));
y=0.00000000000005070037748679935243239016995353 6.*cos(Q1) - 260.*sin(Q1) -
35.*cos(Q3 -
1.570796326734125614166259765625).*(0.00000000000000006123233995384609899612009622
5396.*cos(Q1).*sin(Q2) - 1.*cos(Q2).*sin(Q1)) +
0.00000000000004163799116809835943451602313647 4.*cos(Q1).*sin(Q2) -
680.*cos(Q2).*sin(Q1) - 828.*sin(Q3 -
1.570796326734125614166259765625).*(0.00000000000000006123233995384609899612009622
5396.*cos(Q1).*sin(Q2) - 1.*cos(Q2).*sin(Q1)) + 828.*cos(Q3 -
1.570796326734125614166259765625).*(0.00000000000000006123233995384609899612009622
5396.*cos(Q1).*cos(Q2) + sin(Q1).*sin(Q2)) - 35.*sin(Q3 -
1.570796326734125614166259765625).*(0.00000000000000006123233995384609899612009622
5396.*cos(Q1).*cos(Q2) + sin(Q1).*sin(Q2));
z=35.*sin(Q2).*sin(Q3) - 35.*cos(Q2).*cos(Q3) - 828.*cos(Q2).*sin(Q3) -
828.*cos(Q3).*sin(Q2) - 680.*sin(Q2) + 675;
disp('X,Y,Z definidos')
figure(7)
% ecuaciones cinemáticas del robot planar 6DOF
plot3(x,y,z)
grid on; xlabel('x(t)'); ylabel('y(t)'); zlabel('z(t)');
title('Trayectoria efector')
```

Anexo F: Adaptación de spindle de 500 watts al robot KUKA KR16-2.

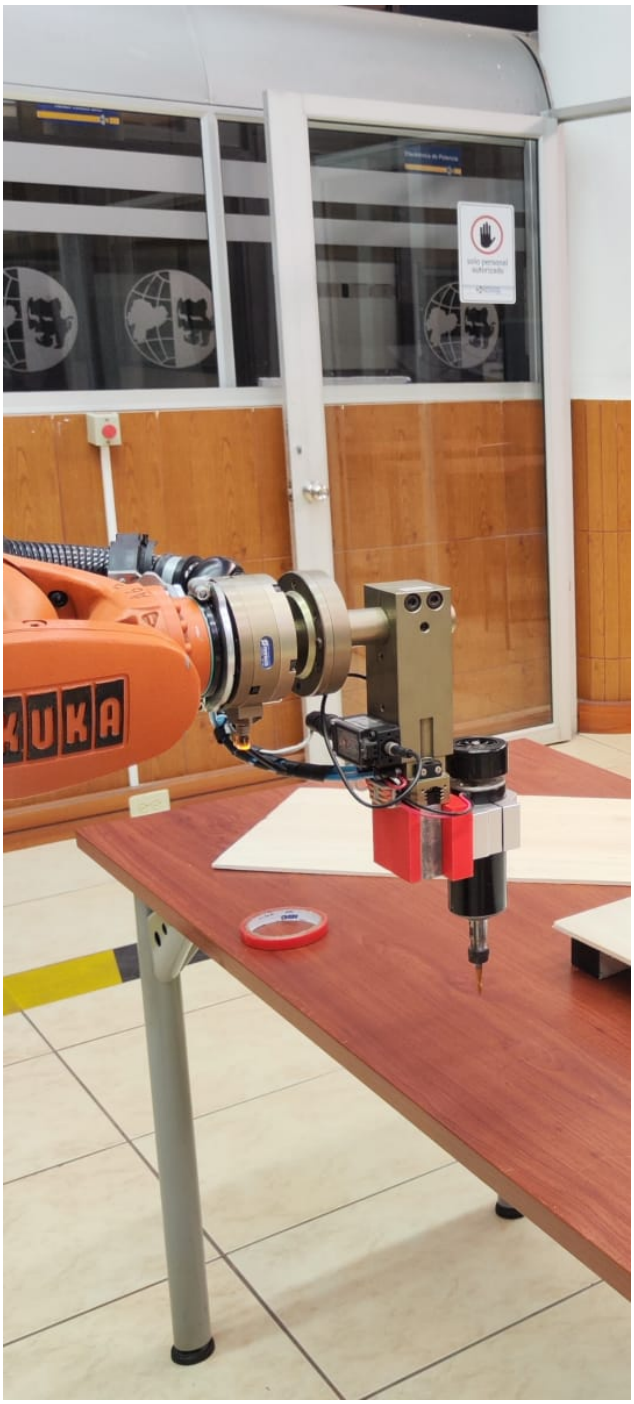


Figura 30

Adaptación de spindle de 500 watts al robot KUKA KR16-2.

Anexo G: hoja de ruta fresado adaptativo

Setup Sheet for Program 1001

Job Description: Configuración1
Document Path: vaciado v3

Configuración

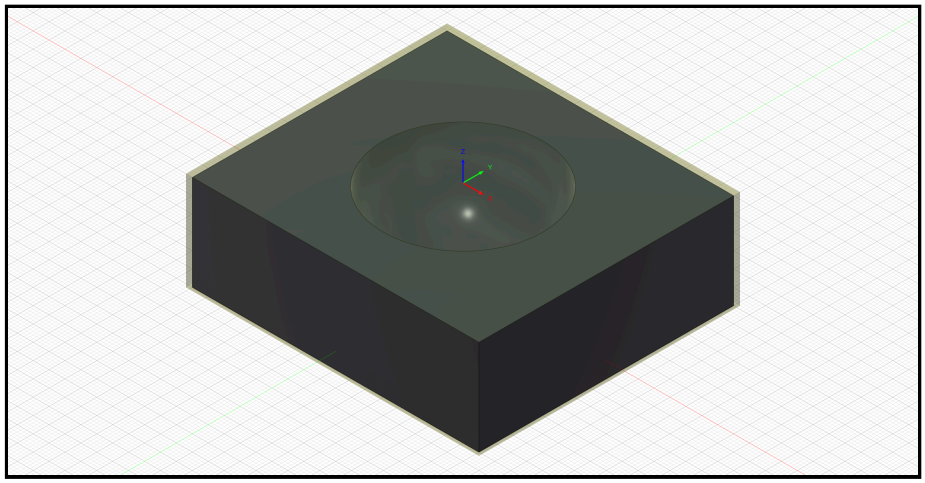
Plano de trabajo: #0

Material:
DX: 92mm
DY: 82mm
DZ: 31mm

Pieza:
DX: 90mm
DY: 80mm
DZ: 30mm

Stock Lower in WCS #0:
X: -46mm
Y: -41mm
Z: -31mm

Stock Upper in WCS #0:
X: 46mm
Y: 41mm
Z: 0mm



Total

Number Of Operations: 2
Number Of Tools: 1
Herramientas: **T1**
Maximum Z: 15mm
Minimum Z: -25.36mm
Maximum Feedrate: 1000mm/min
Velocidad máxima de husillo: 5000rpm
Cutting Distance: 9009.79mm
Distancia rápida: 1606.79mm
Estimated Cycle Time: 9m:40s

Herramientas

T1 D1 L1
Tipo: bullnose end mill
Diámetro: 4mm
Radio de esquina: 0.75mm
Longitud: 24mm
Flutes: 4

Minimum Z: -25.36mm
Maximum Feed: 1000mm/min
Velocidad máxima de husillo: 5000rpm
Cutting Distance: 9009.79mm
Distancia rápida: 1606.79mm
Estimated Cycle Time: 9m:25s (97.4%)

Operaciones

Operación 1/2 Descripción: Cara1 Estrategia: Facing Plano de trabajo: #0 Tolerancia: 0.01mm Sobrepasada máxima: 1.75mm	Maximum Z: 15mm Minimum Z: -1mm Velocidad máxima de husillo: 5000rpm Maximum Feedrate: 1000mm/min Cutting Distance: 4504.17mm Distancia rápida: 26mm Estimated Cycle Time: 4m:31s (46.8%) Refrigerante: Desactivado	**T1** D1 L1 Tipo: bullnose end mill Diámetro: 4mm Radio de esquina: 0.75mm Longitud: 24mm Flutes: 4	
Operación 2/2 Descripción: Adaptativo2 Estrategia: Adaptativo Plano de trabajo: #0 Tolerancia: 0.1mm Sobrematerial: 0.5mm Reducción máxima: 10mm Carga óptima: 1.6mm Desviación de la carga: 0.16mm	Maximum Z: 15mm Minimum Z: -25.36mm Velocidad máxima de husillo: 5000rpm Maximum Feedrate: 1000mm/min Cutting Distance: 4505.62mm Distancia rápida: 1580.79mm Estimated Cycle Time: 4m:53s (50.6%) Refrigerante: Desactivado	**T1** D1 L1 Tipo: bullnose end mill Diámetro: 4mm Radio de esquina: 0.75mm Longitud: 24mm Flutes: 4	

Recalculado por Fusion CAM 2.0.19994 Sunday, September 08, 2024 23:04:02