ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"IMPLEMENTACIÓN DE UNA FRESADORA CNC DE 3 EJES USANDO EL BRAZO ROBOT KAWASAKI RS03N"

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN ELECTRÓNICA Y AUTOMATIZACÓN INDUSTRIAL

KEVIN GEOVANNY BALAREZO TRIVIÑO
DIEGO ANTONIO LUZURIAGA BARROS

GUAYAQUIL – ECUADOR

AÑO: 2017

TRIBUNAL DE EVALUACIÓN

Ph.D. Ángel Domingo Sappa	**Msc. Janeth Carolina Godoy Ortega**
PROFESOR EVALUADOR	PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Sr. Kevin Geovanny Balarezo Triviño

Sr. Diego Antonio Luzuriaga Barros

RESUMEN

El presente trabajo consistió en la implementación de una fresadora CNC de 3 ejes mediante la utilización de un brazo robot Kawasaki RS03N con el propósito de combinar la robótica industrial con el control numérico computarizado (CNC) y facilitar la compresión del funcionamiento de estas máquinas a los estudiantes mediante una aplicación real.

En el capítulo 1 se detalla la problemática del proyecto, en lo que destaca la falta de conocimientos de parte de los estudiantes y profesionales acerca de las nuevas de tecnologías en las industrias, para lo cual se presenta el desarrollo de una base de conocimientos prácticos a los estudiantes politécnicos sobre el uso, diseño, implementación y programación de máquinas CNC y equipos de robótica industrial, para así generar nuevos conocimientos sobre estas tendencias tecnológicas en el perfil de los futuros profesionales.

El capítulo 2 trata del marco teórico, en lo que destaca las características principales de cada máquina, sus protocolos de comunicación, propiedades físicas de trabajo y los lenguajes de programación, el código G que es lenguaje de programación de la CNC y el código AS que es el lenguaje de programación del brazo robot.

El capítulo 3 describe la metodóloga usada, así con el fin de generar los mismos movimientos y trayectorias de una CNC con el brazo robot, se tuvo que adaptar el código G mediante los códigos AS, con esto se consiguió realizar los movimientos del brazo robot mediante los códigos G.

El capítulo 4 presenta los resultados y pruebas realizadas, para lo cual se logró obtener el tallado en madera de imágenes en 2D de cualquier tipo de trayectoria que pueda ser generado por el código G, piezas 3D en madera de cualquier tipo de modelo que pueda ser generado por una CNC de 3 ejes y como resultado final se logró hacer el tallado de circuitos impresos en PCB.

ÍNDICE GENERAL

CAPÍTULO 1

1. DELIMITACIÓN DEL PROBLEMA.

En el presenta capítulo se detalla el planteamiento del problema y su respectiva justificación, se muestran los objetivos generales y específicos del proyecto y se define el alcance de la solución propuesta.

1.1 Planteamiento del problema.

La necesidad de disminuir los costos de operación, tiempos de producción y huella de carbono en las industrias resulta en una creciente demanda de automatización de los procesos de manufacturación. La alta competitividad obliga a los ingenieros a enfocarse en técnicas de producción cada vez más eficientes. Esta necesidad de mejorar la eficiencia en la producción ha dado paso a la utilización de máquinas CNC debido a que su implementación cubre en gran medida los requerimientos actuales, tales como: disminución de tiempos de producción, minimización de errores humanos, entre otros. Se proyecta que el mercado global de máquinas CNC crezca de un valor de US $52.600 millones en 2015 a US $93.400 millones en 2024 y se espera que registre un CARG (tasa de crecimiento anual compuesto) de 6.3% durante este período [1].

El pensum académico de la carrera de Ingeniería en Electrónica y Automatización de la Escuela Superior Politécnica del Litoral no contempla en su contenido la enseñanza del funcionamiento, uso y programación de máquinas CNC, lo que ocasiona que los ingenieros graduados no puedan competir en el mercado laboral. Dado que las máquinas CNC son muy versátiles, es muy común encontrarlas en industrias de todo tipo. Es por esto que se considera necesaria la incorporación de contenido académico relacionado a máquinas CNC en las materias que conforman la malla curricular de la carrera.

Por otro lado, en el laboratorio de Control de Procesos Industriales se dispone de un brazo robótico (Kawasaki RS03N), adquirido en el año 2015, con el que no se han desarrollado suficientes aplicaciones que aporten en la enseñanza e investigación a favor de los estudiantes. Dado el potencial que tienen estos equipos en la industria, se considera urgente la implementación de nuevas aplicaciones reales en los laboratorios para realizar pruebas que permitan preparar a los estudiantes en el área.

1.2 Justificación.

Debido al gran campo de aplicación de máquinas CNC en la industria se tiene la necesidad de preparar a los futuros ingenieros politécnicos en el área. Este proyecto tiene la finalidad de crear una base para el aprendizaje práctico de programación de máquinas CNC. Adicionalmente el estudiante deberá aprender a utilizar programas de diseño y manufactura asistidos por computador (CAD/CAM software). Con esto se beneficiará a los estudiantes para que tengan mayor competencia en el campo laboral.

Para el proyecto se hará uso del brazo robótico disponible en el laboratorio de Control de Procesos Industriales. Mediante un programa que interprete códigos G&M y envíe comandos al brazo robot para que éste realice las trayectorias generadas por el programa CAM. De esta manera se dará una aplicación poco convencional al brazo robótico que a su vez servirá para mecanizar piezas y elaborar tarjetas PCB para prototipos y proyectos propios de los estudiantes o aplicaciones académicas.

1.3 Objetivos

1.3.1 Objetivo General

- Implementar las funciones de una máquina de control numérico computarizado (CNC) a partir de los servomecanismos y controladores de un brazo robot (Kawasaki RS03N).

1.3.2 Objetivos Específicos

- Elaborar un programa en Visual Basic para procesar códigos G&M y generar los códigos AS del brazo robot con el fin de realizar los movimientos de una máquina CNC de 3 ejes.
- Automatizar una fresadora con el brazo robot a fin de mecanizar modelos en tres dimensiones diseñados por medio de un ordenador.
- Implementar las condiciones que necesita la máquina CNC para que el brazo robot sea capaz de realizar el fresado de manera segura.

1.4 Alcance

Se desarrollará un programa en Visual Basic que procese código G&M y genere código AS. Adicionalmente se acoplará al brazo robot un motor de fresado, el cual trabajará con una herramienta para tallado en madera y otra para tallado de PCB. Con esto se podrá emplear el brazo robot como una máquina CNC de 3 ejes.

En los programas se validarán las condiciones de seguridad y se preverán colisiones para asegurar el buen funcionamiento de la máquina CNC y evitar daños en los equipos utilizados, así como también lesiones personales.

CAPÍTULO 2

2. ESTADO DEL ARTE

En el presente capítulo se expondrán los antecedentes del proyecto y se revisarán los aspectos más importantes sobre los elementos que formarán parte del sistema de mecanizado de 3 ejes.

2.1 Antecedentes

El robot Kawasaki RS03N fue adquirido en el año 2015 junto con la planta Lucas Nülle IPA 26 para el laboratorio de Control de Procesos Industriales de la Escuela Superior Politécnica del Litoral. El robot forma parte del proceso de envase y vaciado de botellas; específicamente en la etapa de vaciado. La función del robot en la planta es la de transportar botellas individuales desde un pallet en una banda transportadora hacia la máquina de destapado para posteriormente tomar la botella y vaciar el líquido en un reservorio. La programación del robot fue realizada por los desarrolladores de la planta y vino incluida en la compra. A pesar de que se tienen los recursos necesarios para utilizar el robot en las prácticas de control de procesos, no se las realiza debido a que la etapa de vaciado no se encuentra operativa.

En el segundo término del período 2016-2017 dos estudiantes realizaron el proyecto de materia integradora haciendo uso del brazo robot. Con ayuda de un sistema de visión, el brazo robot clasifica frutas con ayuda de la herramienta de agarre incorporada. El robot fue programado por los estudiantes usando comandos de movimiento y comunicación.

El brazo robot disponible es un brazo robótico de propósito general y hasta ahora las aplicaciones que se le han dado son del tipo “pick and place”, es decir, manipulación de objetos, la cual es una aplicación básica. Para esta aplicación el robot cuenta con una herramienta terminal con dos ventosas neumáticas que

sujetan objetos y son alimentadas por un compresor. Esta herramienta terminal es la única disponible para usar con el brazo robot.

2.2 Marco teórico

En este subcapítulo se describirán conceptos básicos sobre las máquinas CNC de 3 ejes junto con el proceso para a partir de un modelo en 3D obtener un código de programación ISO y las características y programación del Robot Kawasaki RS03N.

2.2.1 Descripción del sistema

Para la implementación del sistema se desarrollarán dos programas: el primero se programará en lenguaje AS y servirá para configurar al robot como servidor TCP/IP para que pueda recibir los comandos que describan las trayectorias necesarias para mecanizar objetos desde el computador; en el segundo programa, que se desarrollará en Visual Basic, se configurará un servidor TCP/IP para comunicarse con el robot. También se encargará de recibir el archivo .NC que contiene programación ISO para CNC generada por el CAM y convertirla en un formato que sea compatible con los comandos AS de movimientos y configuraciones del robot.

El diagrama de la figura 2.1 muestra el flujo de información del sistema. El programa desarrollado en Visual Basic envía una línea de código a la vez y espera por la respuesta del robot antes de enviar la línea siguiente.

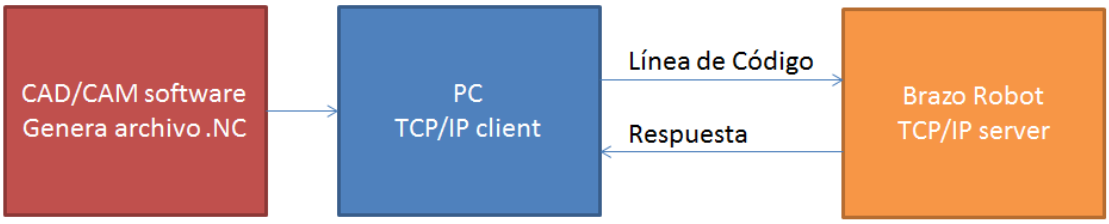

Figura 2.1: Flujo de información del sistema.

2.2.2 Control Numérico Computarizado

El control numérico computarizado es un medio para controlar máquinas a través de un conjunto de instrucciones que describen los ajustes y movimientos requeridos por el programador. Este conjunto de instrucciones posteriormente es procesado por el controlador de la máquina, el cual se encarga interpretar los códigos y llevar a cabo las instrucciones programadas.

Las instrucciones describen los movimientos que tiene que realizar la herramienta de trabajo en relación a al sistema de referencia de la máquina para lograr el producto deseado por el programador [2].

2.2.3 Máquinas de Control Numérico Computarizado

Las máquinas de control numérico computarizado se utilizan para fresar, tornear, cortar, taladrar, prensar, entre otros. La gran versatilidad de estas máquinas ha hecho que sea cada vez más común encontrarlas en todo tipo de industrias. Actualmente la mayoría de las máquinas CNC cuentan con lazos de retroalimentación para controlar velocidad y posición, con lo que se obtiene un mayor rendimiento. Por lo general cuentan con un computador dedicado, el cual se encarga del almacenamiento y procesamiento de datos. El computador trabaja en conjunto con un sistema de interfaz para activar los servomecanismos de movimiento de la máquina. A continuación, se muestran los principales elementos de una Máquina CNC:

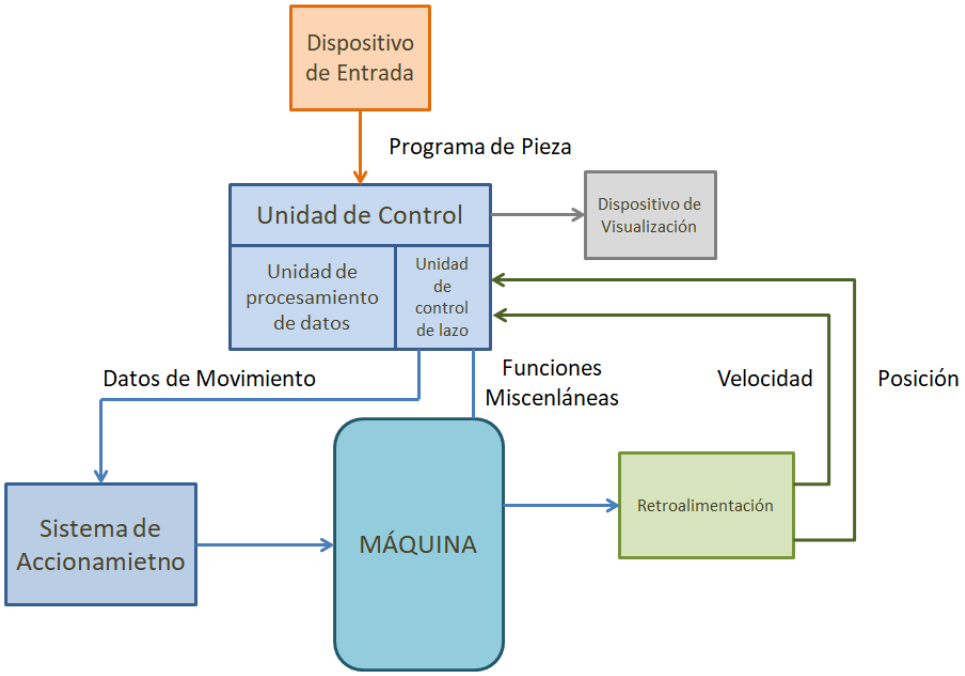

Figura 2.2: Principio de funcionamiento de Máquinas CNC.

2.2.3.1 Dispositivo de Entrada

Los dispositivos de entrada son medios por los cuales se pueden transferir programas de pieza a la máquina CNC. Por lo general las máquinas cuentan con varias maneras de realizar este procedimiento.

Dispositivo de Memoria USB

Es un dispositivo compacto extraíble y regrabable, de gran capacidad en donde se almacena el programa pieza desde el computador para luego transferirlo a la computadora de la máquina CNC. Ambos equipos deben contar con puertos USB para la transferencia de datos.

Comunicación Serial

Otra manera de transferir programas de pieza a la memoria del computador de la máquina CNC es a través del puerto de comunicación serial. Para esto ambos terminales deben contar con una interfaz de comunicación

serial. La mayoría de máquinas CNC y computadores cuentan con un puerto RS232 y se conectan a través de un cable RS232 estándar.

Además de la transferencia programas de pieza, la conexión serial con la máquina CNC permite.

- Monitorización del programa de pieza
- Monitorización de recursos.
- Ejecución de Comandos
- Entre otros.

Por lo general es necesario contar con un software desarrollado por el fabricante para poder acceder a todas las opciones disponibles.

Comunicación Ethernet

La comunicación mediante conexión Ethernet es más eficiente y confiable. Por otro lado, si se dispone de varias máquinas CNC con conexión a Ethernet, es más conveniente construir una red LAN para la interconexión ya que desde un ordenador conectado a la red LAN se puede acceder a las máquinas CNC que sean parte de la red.

Programación Conversacional

Los programas de pieza pueden ser ingresados al controlador a través del teclado de la propia máquina CNC. Se utiliza para programar piezas que no exigen estrategias de mecanizado complejas ya que el programa debe ser ingresado paso a paso por el programador.

2.2.3.2 Unidad de Control (MCU)

La unidad de control es donde se concentra la información de entrada, se procesa y se generan señales eléctricas para accionar dispositivos eléctricos como la velocidad del motor, posición de servomotores, entre otros.

Unidad de procesamiento de datos

Recibe el programa de pieza, lo interpreta y lo codifica en un código de máquina interno. El interpolador calcula las posiciones intermedias del movimiento en términos de la unidad de tamaño básica, que es la unidad más pequeña de movimiento que puede ser manejada por el controlador. Posteriormente los datos calculados son enviados a la unidad de control de laso cerrado. La unidad de procesamiento también es llamada post procesador [3].

Unidad de control de Lazo

La información obtenida de la unidad de control es transformada en señales eléctricas para controlar el sistema de accionamiento y realizar los movimientos requeridos. También se encarga de controlar otras funciones tales como, la velocidad del motor, activar o desactivar el refrigerante, las ventosas de sujeción de la pieza, entre otros [4].

2.2.4 Máquina CNC de 3 ejes

El movimiento de una máquina CNC se basa en el sistema de coordenadas cartesianas. Los desplazamientos se dan a través de los ejes X, Y, Z. Dependiendo de la construcción física de la máquina la herramienta se desplazará en uno o más ejes, mientras que la mesa de soporte se desplazará a través del o de los ejes restantes. Por ejemplo, en un centro de mecanizado de 3 ejes vertical, la herramienta se desplaza verticalmente en el eje Z, mientras que la mesa de soporte se desplaza en el plano que conforman los ejes X, Y.

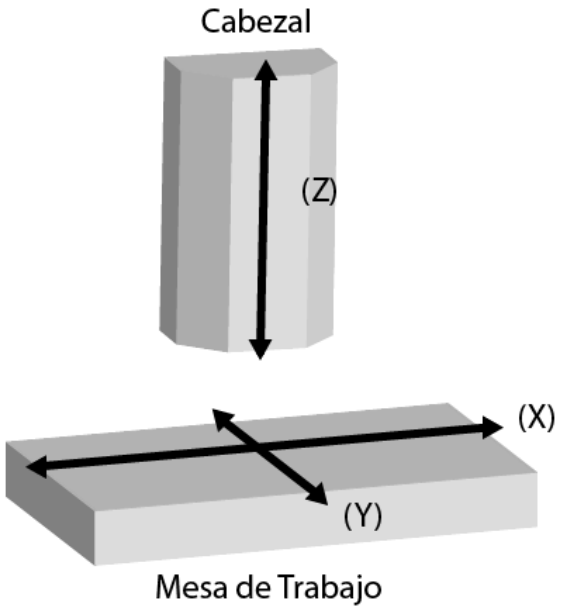

Figura 2.3: Máquina CNC de 3 Ejes.

2.2.4.1 SISTEMA DE COORDENADAS DE MÁQUINA

Las máquinas CNC cuentan con un punto de origen establecido por el fabricante que sirve al momento de inicializar el sistema. Al iniciar el encendido de una máquina CNC, los desplazamientos de la herramienta con respecto a los ejes de coordenadas pueden variar dependiendo de la última posición en la que se apagó el equipo, por lo tanto, el sistema inicia una secuencia para ubicar la herramienta en el Origen de la máquina, de esta manera el sistema sabrá exactamente en qué punto se encuentra. Por lo general el Origen, en inglés llamado Home position, se ubica en los límites de los ejes –X, +Y, +Z. Una vez que se ha llegado al punto de origen, se dice que la herramienta está en posición HOME.

2.2.4.2 SISTEMA DE COORDENADAS DE TRABAJO

Por motivos prácticos, tanto en programación como en calibración, se establece un sistema de coordenadas de trabajo para cada programa de

pieza. Este sistema de coordenadas es un punto sobre la pieza a trabajar establecido por el programador. Por lo general se establece de igual manera que el origen establecido en el CAD. Esto puede variar, pero es importante tener ciertas consideraciones [5].

- El sistema de coordenadas de trabajo se debe poder encontrar por medios mecánicos, es decir, que se pueda situar la herramienta de trabajo en el sistema de coordenadas de trabajo deseado de manera manual
- Debe ser localizado con gran precisión.
- Debe ser repetible: las piezas a mecanizar se deben colocar en la misma posición cada vez.

Figura 2.4: Ejemplo de posible selección de cero de pieza.

2.2.5 Control Numérico Distribuido (DNC)

Es un sistema jerárquico para distribuir información entre un computador de gestión de manufactura y un sistema de control numérico. Las máquinas se vinculan a un computador central de manera directa o a través de otra computadora, para descargar programas de piezas. El programa de pieza que será manufacturada es almacenado en la memoria del computador. Cuando la máquina necesita comandos de control, estos son transmitidos desde el computador de forma inmediata y una vez que se ejecuta el

comando, éste es borrado de la memoria de la máquina CNC. Esta forma de operar resulta particularmente necesaria cuando se tienen limitaciones en la memoria de la máquina.

Existen dos configuraciones para vincular la máquina con la computadora central. En la primera configuración cada máquina es vinculada directamente con la computadora. En este modo puede haber retardos en la computadora central al enviar información requerida por las máquinas mientras ésta procesa otra información. En la segunda configuración, la computadora principal es conectada a la máquina a través de una pequeña computadora llamada computador satélite. La computadora central almacena el programa de la pieza que será mecanizada por una máquina en particular. La computadora satélite recibe y almacena el programa de la pieza. Luego la computadora satélite controla las operaciones de la máquina. La ventaja de este sistema es que la máquina puede ser usada independientemente de si la computadora principal está disponible para realizar dicha operación [6].

2.2.6 Programación de Máquinas CNC – Códigos G&M

Los códigos G&M son un conjunto de instrucciones estandarizadas para la programación de máquinas CNC. Un programa escrito con códigos G&M describe las estrategias programadas para el mecanizado del componente diseñado. Este programa también es llamado programa de pieza.

2.2.6.1 Estructura

Un programa de pieza está formado por un conjunto bloques (líneas). Cada línea contiene palabras (o códigos) que corresponden a funciones o movimientos. Los códigos G corresponden a funciones preparatorias mientras que los códigos M corresponden a funciones misceláneas [3].

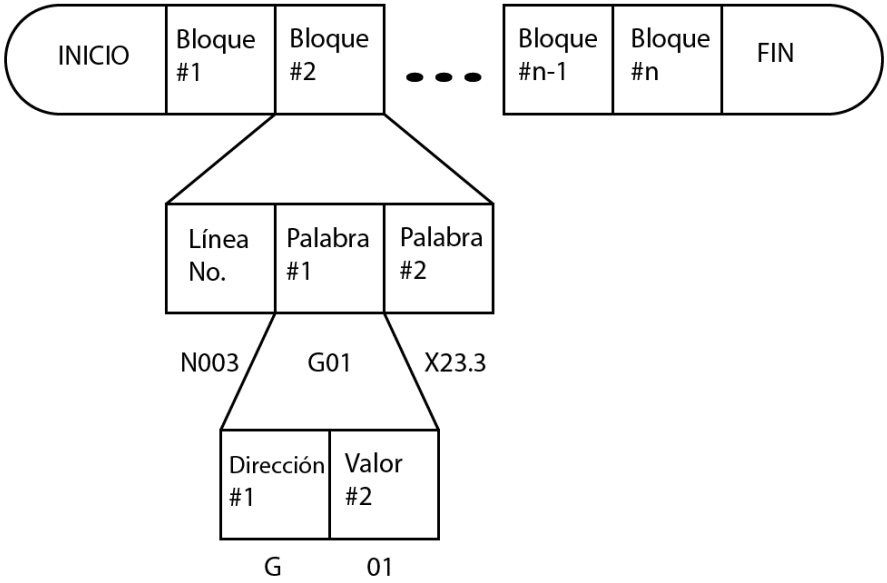

Figura 2.5: Estructura de programa de pieza en código G.

La dirección corresponde a la letra de identificación al comienzo de cada palabra seguido de un número que corresponde al parámetro que se le asigna a dicha función. A continuación, se listan las direcciones más utilizadas:

Dirección	Descripción
N	Número de bloque.
G	Funciones preparatorias.
X, Y, Z	Coordenadas.
F	Velocidad de corte.
M	Funciones misceláneas.
S	Velocidad de husillo.
T	Función de herramienta.

Tabla 1: Principales direcciones utilizadas en código G.

2.2.6.2 Número de Bloque

Este número se asigna a cada bloque con la función de dirección N. Sirve para identificar cada bloque y no interfiere con el orden de ejecución, es decir, los bloques consecutivos no necesariamente tienen un número de identificación consecutivo. Por lo general se utiliza cuando el tipo de programa de pieza se programó como subrutinas, de lo contrario no es absolutamente necesario.

2.2.6.3 Funciones Preparatoria

Las funciones preparatorias sirven para describir cómo se realizarán los movimientos hacia las coordenadas programadas.

Aproximación Rápida

Este comando sirve para desplazar la herramienta a la máxima velocidad y se utiliza para aproximarse al material a trabajar o posicionar la herramienta en un punto deseado. Existe un nivel máximo seguro, que se configura utilizando el CAM, al que se puede aproximar la herramienta utilizando este comando para evitar colisiones que comprometan el equipo. Como parámetros se utilizan las coordenadas de destino X, Y, Z. Al omitir uno de estos parámetros, se considera que el valor anterior de la coordenada se conserva.

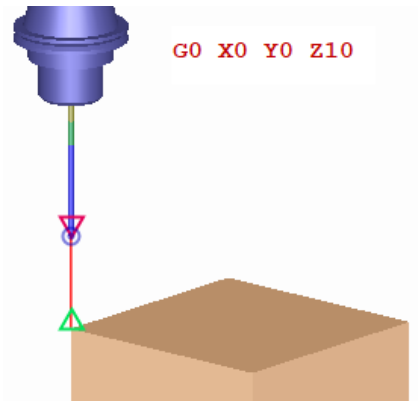

Figura 2.6: Movimiento de Aproximación Rápida.

Interpolación Lineal

Con este comando la herramienta se mueve a través de una línea recta desde el punto en el que se encuentre inicialmente hasta las coordenadas programadas.

Los parámetros de la función son las coordenadas de destino X, Y, Z. Al omitir uno de estos parámetros, se considera que el valor anterior de la coordenada se conserva. Este movimiento se realiza a la velocidad de corte ajustada con la función F.

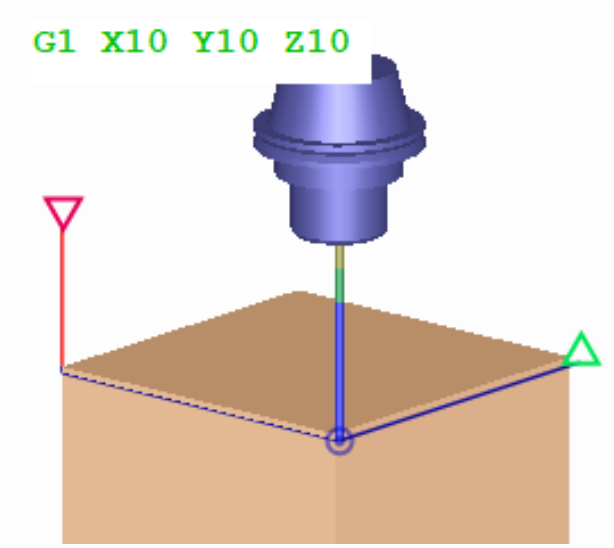

Figura 2.7: Movimiento de Corte Lineal.

Interpolación Circular

Los comandos G2 y G3 sirven para realizar trayectorias circulares a velocidad de corte. La trayectoria circular se realiza en sentido horario y anti horario respectivamente desde el punto que se encuentra inicialmente la herramienta hasta las coordenadas de destino X, Y, Z. Al omitir uno de estos parámetros, se considera que el valor anterior de la coordenada se conserva. Otro parámetro adicional requerido es el radio de la trayectoria circular deseada.

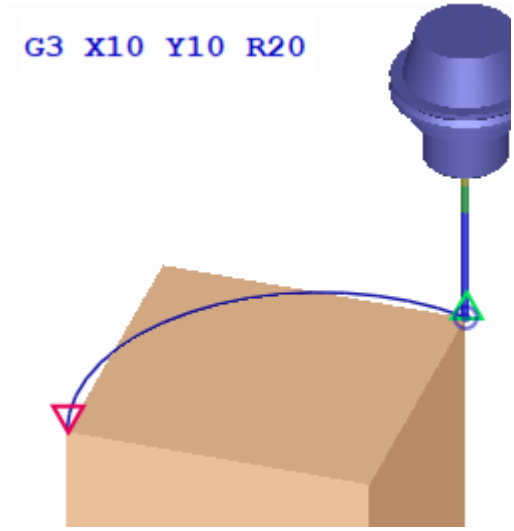

Figura 2.8: Movimiento de Corte Circular.

2.2.7 Brazo Robot Kawasaki RS03N

Los brazos robóticos son máquinas de control numérico computarizado. Al contrario de las máquinas CNC convencionales, los lenguajes de programación de brazos robóticos no se han estandarizado por lo que cada fabricante desarrolla su propio lenguaje de programación válido para sus equipos.

El brazo robótico Kawasaki RS03N, de propósito general, posee 6 grados de libertad y se programa utilizando el lenguaje AS de robots industriales Kawasaki. Se utiliza en aplicaciones de manipulación de objetos, ensamblaje, inspección, etc. Al ser un robot de propósito general, cuenta con un amplio rango de movimientos y varias opciones de comunicación lo que lo hace versátil a la hora de desarrollar aplicaciones.

2.2.7.1 Características Generales

El brazo robot RS03N es el robot más pequeño de la familia R de Kawasaki. A continuación, se muestran las características físicas principales:

Carga	3 Kg
Grados de Libertad	6
Repetitividad	±0.02mm
Velocidad máxima	6000 mm/s
Peso	20 Kg
Grado de Protección	IP54

Tabla 2: Características físicas del brazo robot Kawasaki RS03N [7].

2.2.7.2 Zona de Trabajo

El brazo robot RS03N posee 6 seis articulaciones que son capaces de rotar independientemente una de la otra y se identifican con el prefijo JT seguido por el número de la articulación. La combinación de los movimientos de las articulaciones resulta en una zona de trabajo que se muestra a continuación:

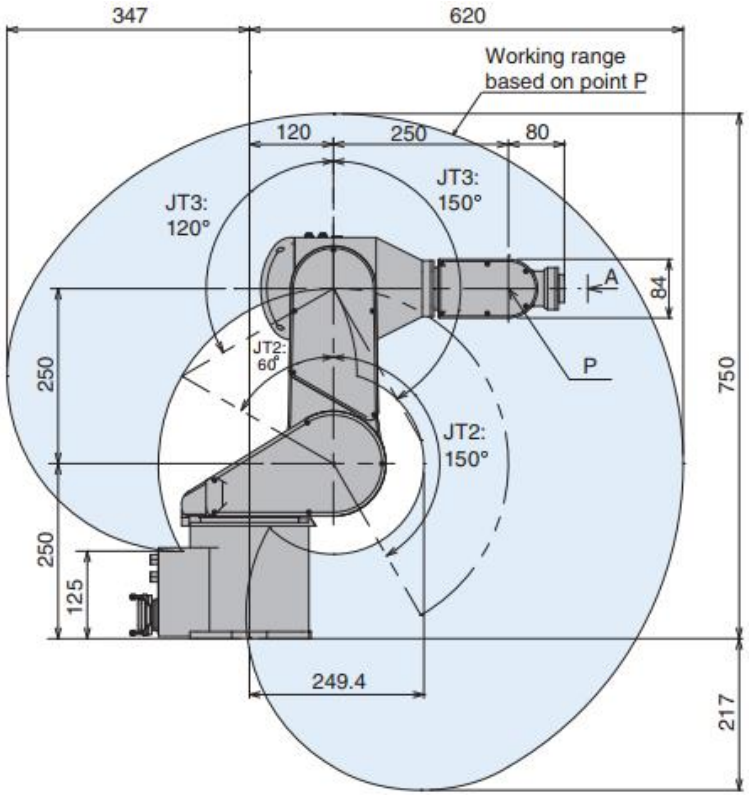

Figura 2.9: Límites físicos del brazo robot Kawasaki RS03N [7].

2.2.7.3 Movimientos

El brazo robot dispone de tres tipos de movimientos que se utilizan dependiendo de la aplicación y tipo de trayectoria deseadas. Para una misma secuencia se pueden utilizar los tres tipos de movimiento de acuerdo a lo requerido.

Movimiento Articulado Interpolado

Con este tipo de movimiento, se puede controlar cada articulación del brazo robot de manera independiente. Se puede ajustar la posición y velocidad de cada articulación, siempre teniendo en cuenta los rangos de movimiento de las articulaciones que se enlistan a continuación:

Rango de movimiento	JT1	±160°
	JT2	+150° y -60°
	JT3	+120° y -150°
	JT4	±360°
	JT5	±135°
	JT6	±360°
Velocidad máxima	JT1	360°/s
	JT2	250°/s
	JT3	2250°/s
	JT4	540°/s
	JT5	225°/s
	JT6	540°/s
Torque máximo	JT4	5.8 N·m
	JT5	5.8 N·m
	JT6	2.9 N·m

Tabla 3: Especificaciones de Articulaciones del brazo robot Kawasaki RS03N [8].

Movimiento Respecto a las Coordenadas Base

En este modo, el brazo robot mueve el TCP (punto central de herramienta) a lo largo de los ejes X, Y, Z de la referencia de la base del robot.

Movimiento Respecto a las Coordenadas de Herramienta

En este modo, el brazo robot mueve el TCP (punto central de herramienta) a lo largo de los ejes X, Y, Z de la referencia de la terminal de herramienta del robot.

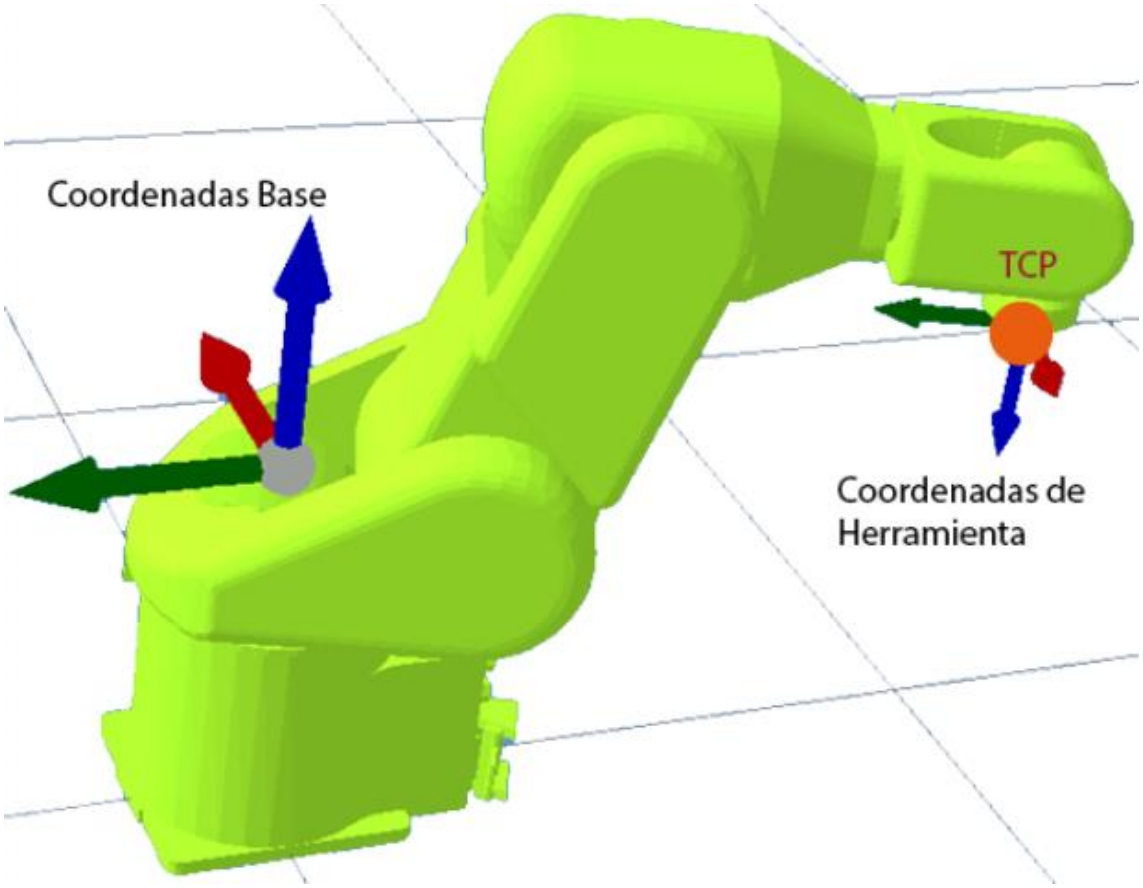

Figura 2.10: Ejes de Coordenadas Base, Coordenadas de Herramienta y ubicación del TCP.

2.2.7.4 Lenguaje AS

El lenguaje de programación AS es un lenguaje de programación de bajo nivel que es usado para comunicación y programación del robot. El lenguaje AS se puede dividir en dos tipos de comandos: comandos de monitoreo e instrucciones de programa.

Comandos de monitoreo: Son usados para escribir, editar y ejecutar programas. Estos códigos se ingresan a través del software de programación KRterm.

Instrucciones de programa: son usadas para describir movimientos del robot, monitoreo de señales internas y externas, comunicación y posee instrucciones para controlar el flujo de ejecución del programa [9].

2.2.7.5 Protocolo de Comunicación

El brazo robot se comunica con la PC a través de una red TCP/IP y se pueden conectar de dos maneras:

- Conexión del controlador con el PC usando un servidor central.
- Conexión directa del controlador con el PC.

Se usará el primer modo de conexión ya que así se podrá conectar al robot desde cualquier PC del laboratorio en donde esté instalada la aplicación sin necesidad de realizar conexiones extra.

En el lenguaje AS existen dos tipos de comandos de comunicación:

- Instrucciones de comunicación TCP.
- Instrucciones de comunicación UDP.

En comunicación TCP los mensajes son automáticamente reenviados cuando ocurren errores de comunicación. En la comunicación UDP los mensajes no son reenviados en caso de errores. Se utilizará la comunicación TCP debido a que es importante que los datos enviados desde la PC lleguen sin errores.

Los comandos AS de comunicación TCP se basan en la arquitectura cliente/servidor. Se configurará al robot como servidor TCP ya que de esta manera se ejecutará el programa del robot y éste esperará la conexión desde la PC [10], [11].

Los comandos AS de comunicación TCP disponibles para servidor son:

Comando	Descripción
TCP_LISTEN	Crea un socket y espera por una petición de conexión.
TCP_ACCEPT	Verifica si una petición de conexión se ha recibido.
TCP_SEND	Envía datos al cliente conectado.
TCP_RECV	Recibe datos del cliente.
TCP_CLOSET	Cierra el socket de comunicación.
TCP_END_LISTEN	Finaliza la espera de petición de conexión.

Tabla 4: Comandos de comunicación TCP/IP de lenguaje As.

CAPÍTULO 3

3. METODOLOGÍA DE TRABAJO

En el presente capítulo se describe la metodología utilizada para el proceso de mecanizado de piezas a partir de un archivo con códigos G. Las dos primeras etapas se requieren para obtener el archivo .NC con los códigos G del modelo que se desea mecanizar. Las dos etapas siguientes sirven para generar el archivo de texto que contendrá los comandos que serán enviados al brazo robot.

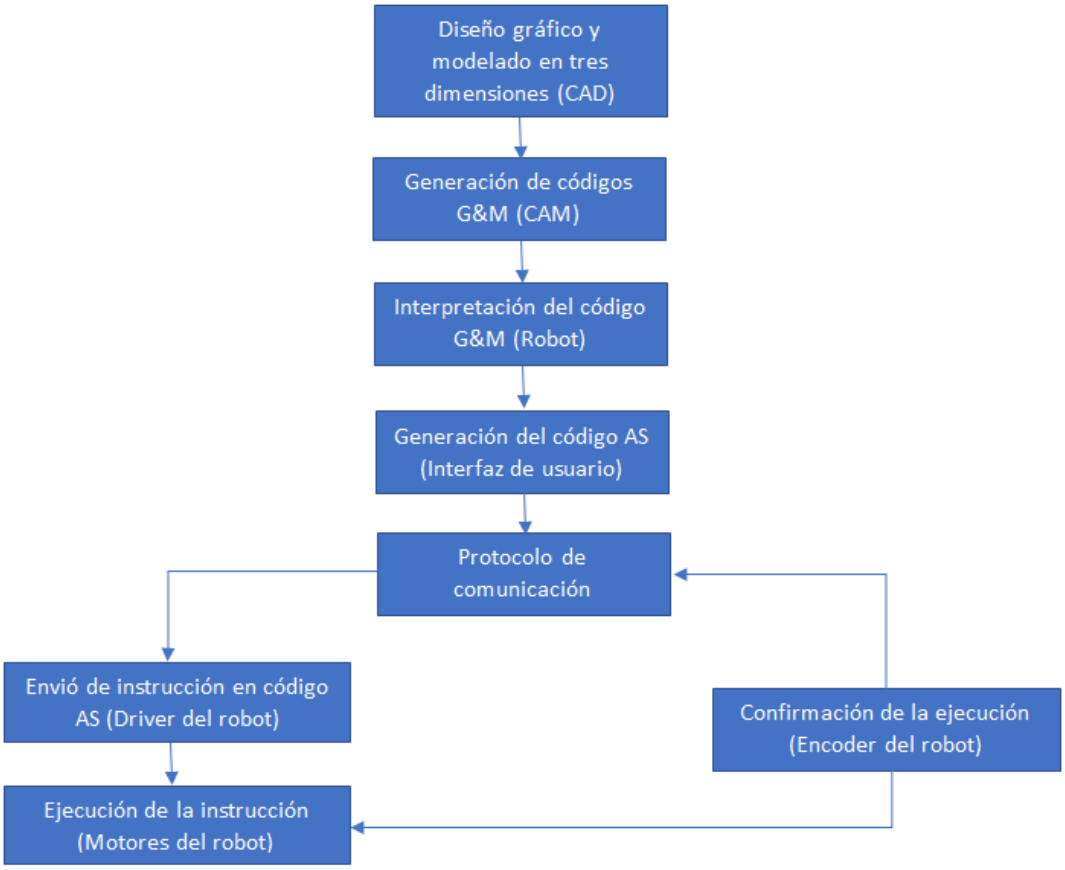

Figura 3.1: Diagrama del proceso para mecanizado.

3.1 Código AS Simplificado

Debido a que los códigos G, dependiendo del post procesador seleccionado al momento de generar el archivo de pieza, varían en el formato y que no todos los códigos de movimiento G tienen su equivalente en lenguaje AS, se estableció un formato para los comandos que posteriormente serán enviados al brazo robot.

```
N70 G43 Z10. H72 M09
N80 G1 Z-0.25 F800
N90 X42.179 Y7.136 Z0.
N100 X42.325 Y7.067 Z-0.25
N110 X43.153 Y7.246
N120 X43.372 Y7.43
N130 X43.339 Y7.493 Z0.
N140 X43.369 Y7.525 Z10.
N150 X43.477 Y7.548 Z-0.25
N160 X43.742 Y8.065
```

```
N40 G0 Z10
N50 G0 X42.085 Y7.07 S1500 M3
N60 G1 Z-0.25 F500
N70 G1 X42.179 Y7.136 Z0
N80 G1 X42.325 Y7.067 Z-0.25
N90 G1 X43.153 Y7.246
N100 G1 X43.372 Y7.43
N110 G1 X43.339 Y7.493 Z0
N120 G1 X43.369 Y7.525 Z10
N130 G1 X43.477 Y7.548 Z-0.25
```

```
N20G0Z10.
N30X42.085Y7.07
N40G1Z-0.25F500
N50X42.179Y7.136Z0.
N60X42.325Y7.067Z-0.25
N70X43.153Y7.246
N80X43.372Y7.43
N90X43.339Y7.493Z0.
N100X43.369Y7.525Z10.
N110X43.477Y7.548Z-0.25
```

Figura 3.2: Diferencia entre formatos de código G según post procesador.

Dado que cada línea de código consta de un comando con sus respectivos parámetros y que al ser enviadas al brazo robot llegan a éste como cadena de caracteres que hay que manipular para separar el comando y sus parámetros, se usó un formato simple debido a que los comandos de manejo de cadena de caracteres son limitados en lenguaje AS y a su vez la ejecución del programa del brazo robot se agiliza. A continuación, se muestra un ejemplo del formato simplificado:

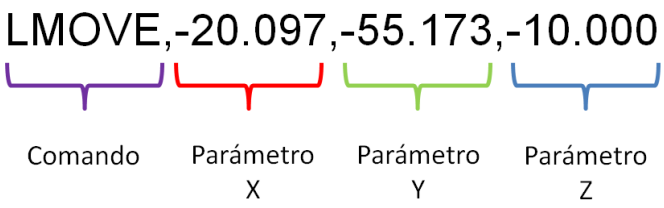

Figura 3.3: Ejemplo del formato de código AS simplificado.

Como se observa en la figura 3.3, la línea de código simplificado consiste en columnas separadas por una coma. La primera columna es el comando de

movimiento que el brazo robot deberá ejecutar y a continuación en las columnas restantes se encuentran los parámetros de coordenadas X, Y, Z respectivamente.

3.2 Programa AS del Brazo Robot

El programa AS que se desarrolló para el brazo robot sirve para configurar al robot como servidor TCP/IP y así poder lograr establecer la comunicación con la aplicación en Visual Basic. Además, el programa se encarga de manipular las cadenas de caracteres recibidas mediante la comunicación TCP/IP y ejecutar los comandos de movimiento. A continuación, se muestra el diagrama de flujo del programa:

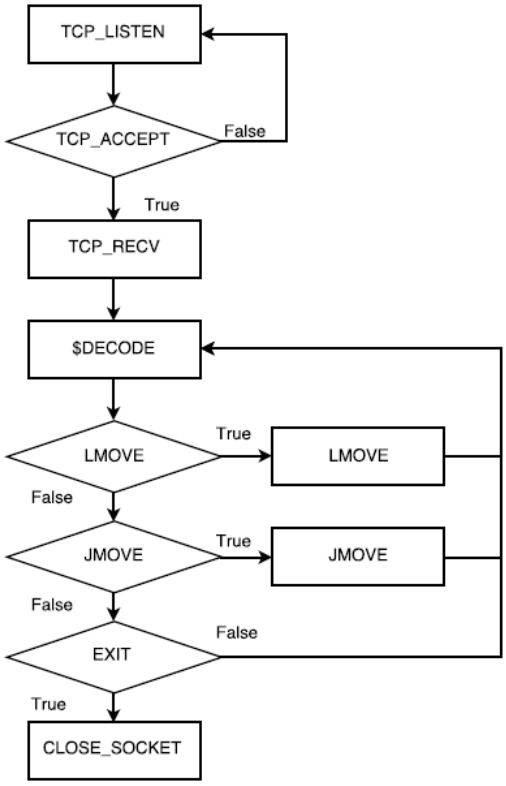

Figura 3.4: Diagrama de Flujo del Programa del Brazo Robot.

El programa inicialmente espera una petición de conexión por parte de la aplicación cliente TCP/IP programada en Visual Basic. Una vez establecida la conexión el

programa espera una línea de comando por parte del cliente. Esta línea llega al servidor como una cadena de caracteres y consta de un comando y sus parámetros separados por una coma. El programa separa el comando y los parámetros en variables independientes usando la instrucción “$DECODE” y posteriormente realiza la validación correspondiente para ejecutar el comando de movimiento con las coordenadas recibidas como parámetros.

La velocidad de avance rápido es fija y se configura antes de ejecutar el movimiento. Esta velocidad es de 10 mm/s. Se programó de esta manera para que el programa no admita configuraciones de velocidad por parte del cliente para evitar que estas velocidades sean manipuladas de forma indebida y así evitar daños en los equipos.

La velocidad de corte se configura desde la ventana principal de la aplicación de pc. El usuario ingresa el valor deseado y ésta se envía al brazo robot al iniciar el proceso de mecanizado. La velocidad está restringida a un rango de 5 a 10 mm/s.

Una vez que el movimiento requerido se ha ejecutado, se envía una respuesta al cliente y espera la siguiente línea de código. Se optó por la comunicación en doble sentido ya que se consideró necesario informar al cliente cuando se ha finalizado un movimiento para que éste a su vez pueda enviar la siguiente línea y así optimizar el tiempo de envío y recepción de líneas de código.

3.3 Programa desarrollado en Visual Basic

La aplicación que se programó en Visual Basic se encarga de leer el archivo .NC para generar un archivo de texto AS.txt con las líneas de código simplificadas obtenidas a partir de las líneas de código G. También establece comunicación con el brazo robot para enviar línea por línea los comandos del archivo AS.txt.

Cada vez que la aplicación envía una línea de código hacia el brazo robot, espera por una respuesta que indica que el brazo robot ha realizado dicho movimiento. Una vez que recibe la respuesta, la aplicación envía la línea siguiente y así sucesivamente hasta finalizar. De esta manera si existe algún fallo en la

comunicación, se sabe exactamente qué línea reenviar y la mecanización no se ve afectada.

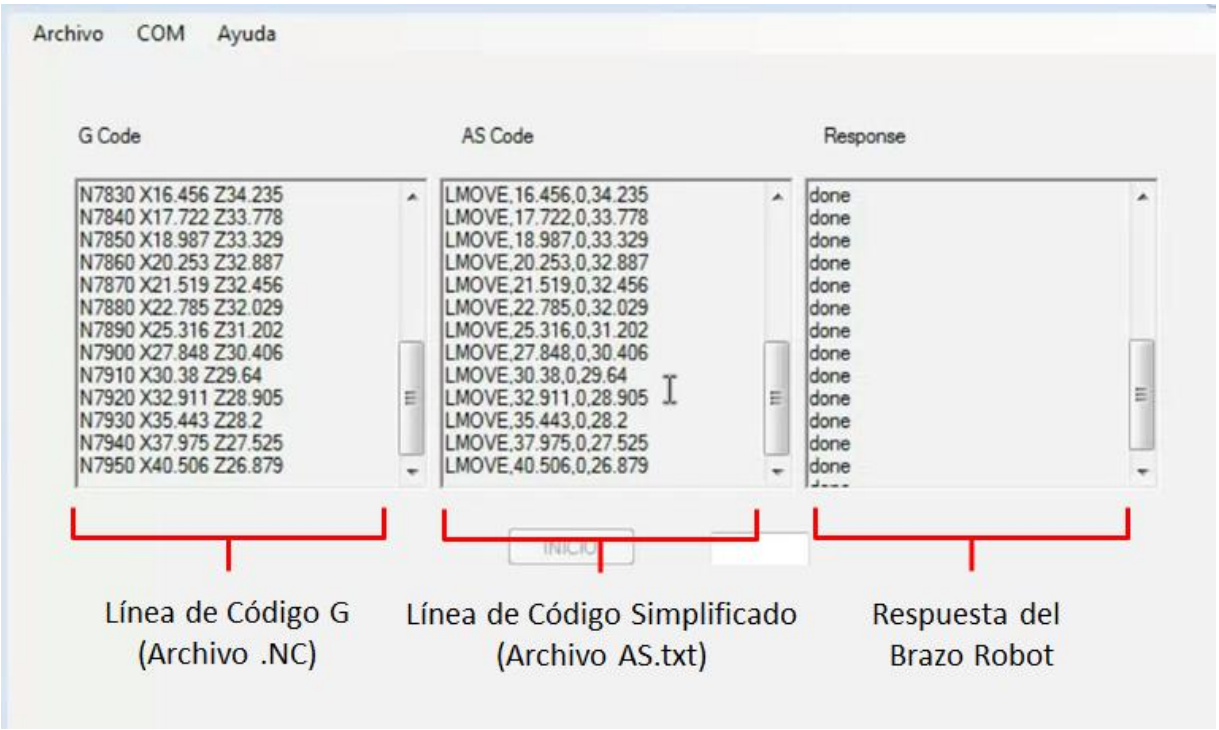

Figura 3.5: Ventana Principal de Aplicación Desarrollada en Visual Basic.

3.3.1 Conexión TCP/IP con el Servidor

En la figura 3.6 se muestra el código para establecer la comunicación TCP/IP con el servidor (brazo robot). Para esto se establece la variable ***_Connection*** del tipo ***ConnectionInfo*** con los atributos necesarios para establecer la conexión. También se emplea la sentencia ***Try-Catch*** para manejar errores que puedan ocurrir mientras se intenta establecer la comunicación.

Una vez establecida la comunicación se pude desconectar usando el mismo botón. Para esto se usa el método ***close*** de la clase ***ConnectionInfo*** y se asigna un valor ***NULL*** a la variable ***_Connection*** [12].

```
    Private Sub ConectarToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles ConectarToolStripMenuItem.Click
        If ConectarToolStripMenuItem.Text = "Connect" And IsNothing(_Connection) Then
            Try
again:          _Connection = New ConnectionInfo(mIpadd(), CInt(mPort()), AddressOf xUpdate)
                _Connection.AwaitData()
                ConectarToolStripMenuItem.Text = "Disconnect"
                'Button1.Enabled = True
                MessageBox.Show("Now you are connected to server.", "My 3 axis Milling Robot",
                MessageBoxButtons.OK, MessageBoxIcon.Information)
            Catch ex As Exception
                If MessageBox.Show("Cannot connect to server.", "My 3 axis Milling Robot",
                MessageBoxButtons.RetryCancel, MessageBoxIcon.Error) _
                = DialogResult.Retry Then
                    GoTo again
                End If
            End Try
        Else
            If _Connection IsNot Nothing Then _Connection.Close()
            _Connection = Nothing
            ConectarToolStripMenuItem.Text = "Connect"
            'Button1.Enabled = False
        End If
    End Sub
```

Figura 3.6: Código para Establecer Comunicación TCP/IP con el Brazo Robot.

3.3.2 Procesamiento de código G y Generación de archivo AS.txt

En esta parte del proceso, el programa diseñado se encarga de convertir línea a línea los códigos G del archivo de pieza seleccionado y genera el archivo AS.txt que posteriormente será leído para enviar su contenido al brazo robot.

La conversión del código G se basa en la lectura del fichero que contiene la lista de códigos G de la pieza a mecanizar. Esta lectura se realiza mediante un puntero el cual realiza un barrido cada línea del programa de pieza y separa los diferentes tipos de funciones, coordenadas y valores numéricos respectivos del código G.

Por cada función de código G que el puntero detecta del programa de pieza, el programa procesa los valores numéricos para posteriormente generar las funciones equivalentes en código simplificado con los parámetros de coordenadas. Posteriormente se escribe en el archivo AS.txt las líneas de código simplificado que se obtienen como resultado la conversión del código G. A continuación, en la tabla se detalla la lista funciones de código G usadas:

Tipo de función	Código G	Código Simplificado
Movimiento lineal rápido	G00 X0 Y0 Z0	JMOVE,0,0,0
Movimiento de corte lineal.	G01 X100 Y100 Z100	LMOVE,100,100,100
Movimiento de corte circular con sentido horario	G02 X100 Y100 R100	INTERPOLACIÓN DE PUTNOS CON LMOVE
Movimiento de corte circular con sentido anti horario	G03 X100 Y100 R100	INTERPOLACIÓN DE PUTNOS CON LMOVE

Tabla 5: Tabla de Equivalencias entre Código G y Código Simplificado.

Una vez terminada la lectura del programa de pieza, el archivo AS.txt contiene las líneas de código equivalentes para mecanizar el modelo deseado. La primera línea de este archivo es de configuración de la velocidad de corte. El resto de líneas de código corresponden cada una a un movimiento, los cuales a su vez corresponden a los movimientos generados por el CAM.

3.3.2.1 Velocidad de Avance

En código G, el comando de velocidad de avance es definido por medio del parámetro F seguido del valor numérico en milímetros por segundo (mm/s) y corresponde a la velocidad al que debe de desplazarse la herramienta al cortar. Al procesar la función y convertirla a código simplificado, el comando cambia por SPEED seguido con el respectivo valor numérico de velocidad en mm/s separados por una coma.

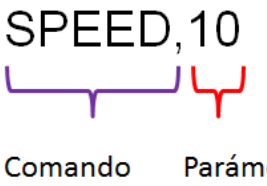

Figura 3.7: Código Simplificado para Ajuste de Velocidad de Avance.

Los ajustes de velocidad depende de factores como el tipo de material a usar, la herramienta de corte utilizado, la técnica de fresado, estos parámetros son definidos previamente en el CAD/CAM del modelo a diseñar y se ajustan automáticamente por defecto en el software o de manera manual.

3.3.2.2 Movimiento lineal rápido

El procesamiento de esta función en el programa consiste en interpretar el comando G00 del código G y dar como resultado la función JMOVE del código AS.

El comando G00 está acompañado de valores numéricos que son las coordenadas hacia donde se debe desplazar la herramienta. El movimiento G00 en las CNC son movimientos rápidos en donde la herramienta no tiene contacto con la pieza por lo tanto no hay mecanizado.

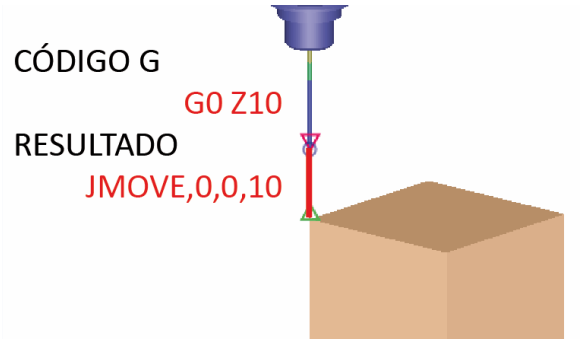

Figura 3.8: Conversión de comando de Movimiento de Aproximación Rápida.

Al interpretar la función G00, las variables de coordenadas X Y Z se guardan en una variable temporal para luego escribirlas en el formato simplificado en el archivo AS.txt.

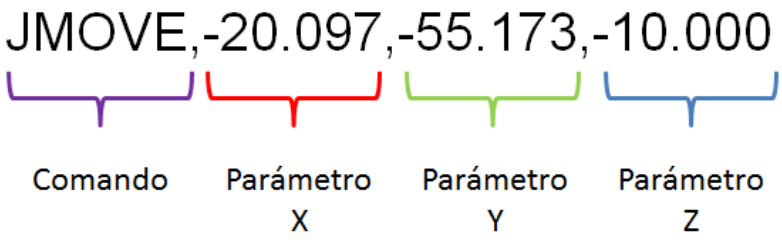

Figura 3.9: Código Simplificado para Comando de Movimiento Lineal Rápido.

3.3.2.3 Movimiento de corte lineal.

El procesamiento de esta función en el programa consiste en interpretar el comando G01 del código G y dar como resultado la función LMOVE del código AS.

El comando G01 está acompañado de valores numéricos que son las coordenadas hacia donde se debe desplazar linealmente la herramienta. El movimiento G01 en las CNC son movimientos de corte y mecanizado en donde la herramienta trabaja y tiene contacto con la pieza.

Figura 3.10: Simulación de Movimientos Lineales en Código G.

Al interpretar la función G01, las variables de coordenadas X Y Z se guardan en una variable temporal para imprimir como resultado estas mismas coordenadas, pero en la función LMOVE. Se tiene en cuenta que LMOVE representa un movimiento lineal en el brazo robot y simula exactamente el mismo comportamiento de la función G01 de las CNC.

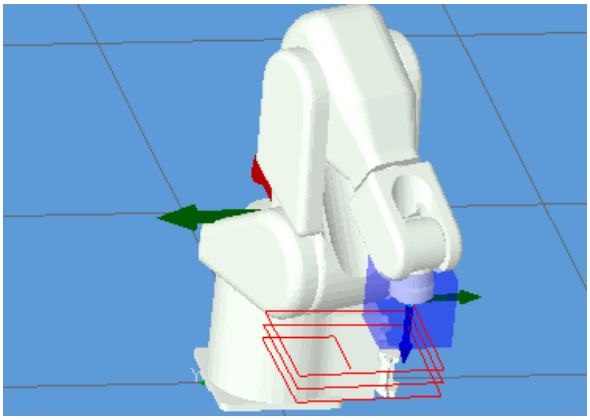

Figura 3.11: Simulación de Movimientos Lineales en Lenguaje AS.

Algunos movimientos del comando G01 son muy pequeños, sin embargo, la capacidad de los motores de paso del robot Kawasaki permiten realizar este tipo de movimientos. Debido a que este movimiento es de corte y fresado, donde existe contacto con la pieza, puede ocurrir que en algunos pasos exista un pequeño desvió o deslizamiento milimétrico debido a la propia estabilidad física del robot.

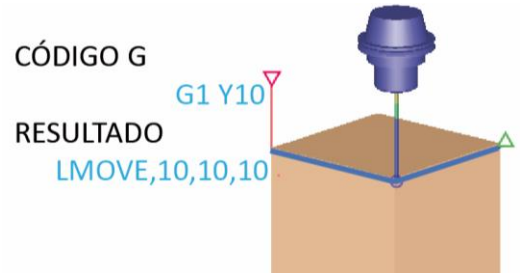

Figura 3.12: Conversión de comando de Corte Lineal.

3.3.2.4 Movimiento de corte circular con sentido horario y anti horario

El procesamiento de esta función en el programa consiste en interpretar el comando G02 para movimiento circular horario o el comando G03 para el movimiento circular anti horario. Como resultado se obtiene una interpolación de puntos con la función LMOVE del código simplificado para obtener una aproximación del movimiento circular.

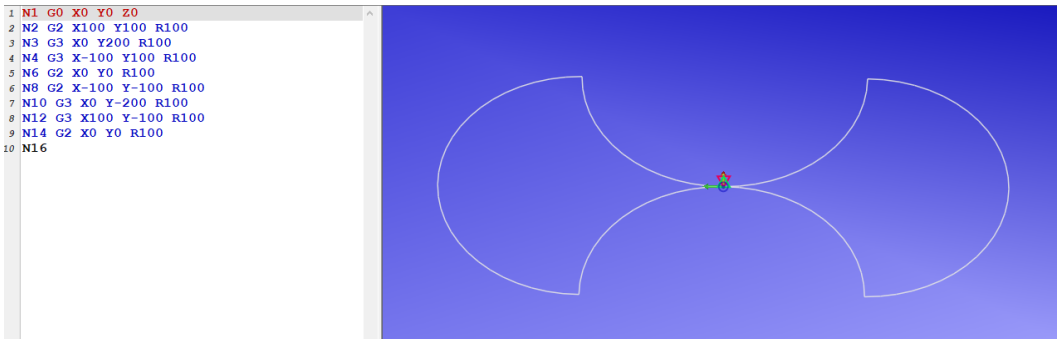

Figura 3.13: Simulación de Movimientos Circulares en Código G.

Para lograr la interpolación de los puntos a través de la trayectoria circular se requiere información de la posición actual de la herramienta, así como también la posición de destino y el radio de la trayectoria circular. Esta información se procesa mediante un algoritmo matemático para obtener las coordenadas de los puntos que componen el arco y así poder interpolar linealmente dichos puntos con la función LMOVE del brazo robot dando como resultado una aproximación del movimiento circular.

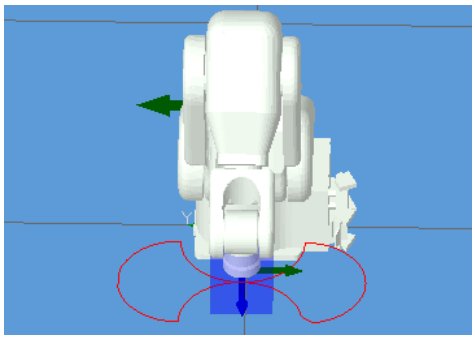

Figura 3.14: Simulación de Movimientos Circulares en Lenguaje AS.

La aproximación gráfica del arco dependerá del número de pasos que la serie de puntos sea procesada. A continuación, se muestra un diagrama de flujo del algoritmo matemático que interpola los puntos de la trayectoria circular para recrear los movimientos circulares de las maquina CNC en el brazo robot Kawasaki.

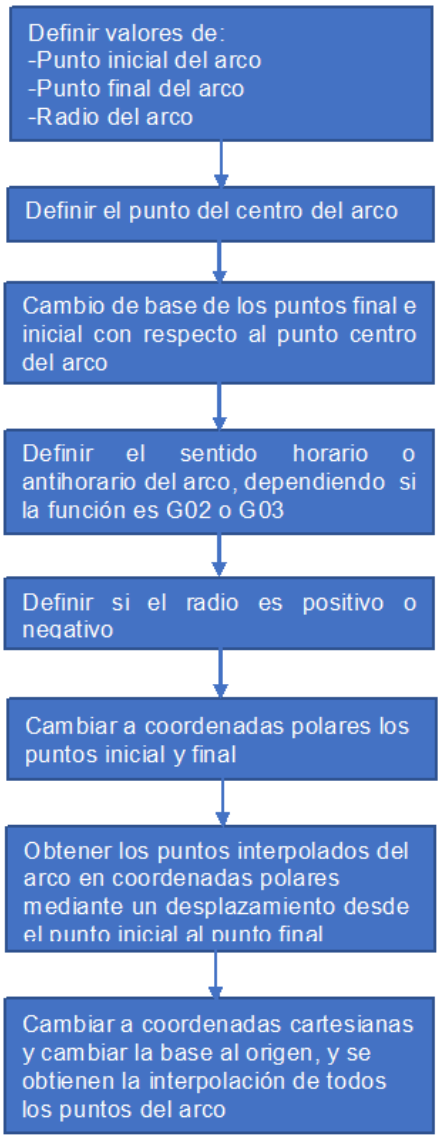

Figura 3.15: Algoritmo para Conversión de Comandos de Movimiento Circular.

El diagrama de flujo de la figura 3.15 representa el algoritmo matemático usado para convertir los comandos G02 y G03 al código simplificado. El primer paso es definir las variables del arco tanto punto inicial, punto final y radio. Obteniendo estos datos se calcula el centro de la circunferencia o arco, dicho centro será la nueva referencia de todos los puntos. Luego se realiza un cambio de base de los puntos iniciales y finales con respecto al nuevo origen que es el centro del arco. Posteriormente se cambia de coordenadas cartesianas a coordenadas polares con el fin de obtener una función en forma polar para obtener los puntos del arco a representar. Luego se realiza un desplazamiento desde el punto inicial al punto final en forma polar para obtener los puntos intermedios y se obtiene una serie de puntos en forma polar con un paso de 1 grado correspondiente a los puntos del arco. Posteriormente transforma a coordenadas cartesianas y se vuelve a realizar el cambio de base al origen de los puntos obtenidos. Los puntos de la trayectoria circular calculados se escriben en código simplificado con el comando de movimiento LMOVE en el archivo AS.txt. Por último, al ser enviados uno a uno estos puntos, el brazo robot se desplazará a través de la trayectoria circular programada.

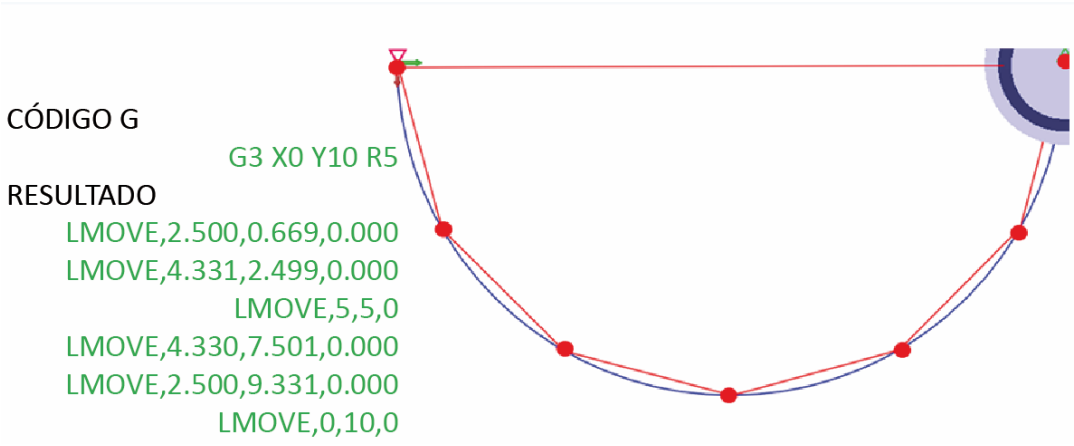

Figura 3.16: Interpolación de puntos a través de trayectoria circular con un paso de 30 grados.

3.4 Delimitación del Área de trabajo

Unas de las limitaciones que presentan tanto las maquinas CNC como el robot Kawasaki es el volumen de trabajo. Para ello se define estas condiciones en el programa antes de iniciar cualquier proceso.

El robot Kawasaki puede moverse libremente dentro de una región establecidas por los grados de libertad que este mismo presenta, sin embargo, los límites del área total de movimiento del robot tienden a ser esféricas y no presentan limites lineales. Las maquinas CNC por defecto trabajan en una región defina por limites lineales en los ejes X Y Z, definidos por el tamaño estructural de la máquina.

Se establece un prisma rectangular circunscrito en la región esférica del robot. Los lados del prisma son los límites del área de trabajo de la CNC, límites de movimientos del robot y el tamaño máximos de la pieza a modelar. Los lados del prisma rectangular se definieron de manera que se obtenga un mayor aprovechamiento del espacio de trabajo del brazo robot.

Es importante la asignación de estos límites en el programa para evitar errores en el robot como movimientos fuera del rango, o piezas que excedan el tamaño máximo permitido.

Los puntos de los límites definidos para la región de trabajo son:

3.4.1 Región de trabajo referida a la base del robot.

$X_{superior}$ = -250 mm $Y_{superior}$ = 250 mm $Z_{superior}$ = 40 mm

$X_{inferior}$ = -550 mm $Y_{inferior}$ = -250 mm $Z_{inferior}$ = -100 mm

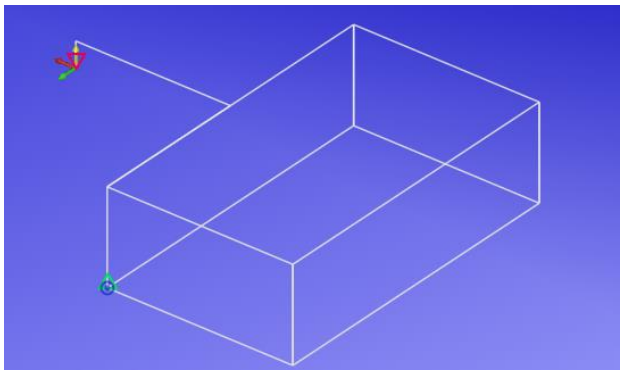

Figura 3.17: Región de Trabajo Referida a la Base del Robot.

3.4.2 Región de trabajo referida al centro de la CNC.

$X_{superior}$ =150 mm $Y_{superior}$ = 250 mm $Z_{superior}$ = 70 mm

$X_{inferior}$ = -150 mm $Y_{inferior}$ = -250 mm $Z_{inferior}$ = -70 mm

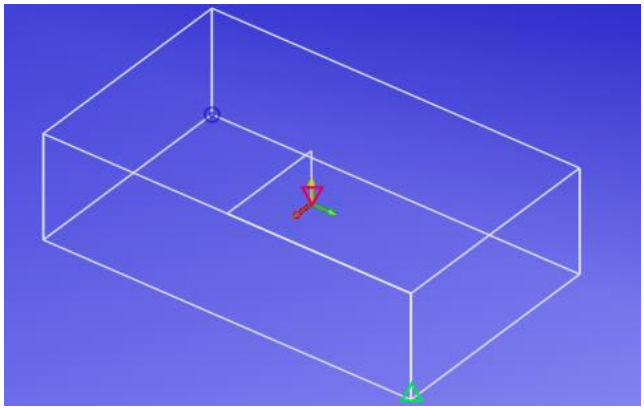

Figura 3.18: Región de Trabajo Referida al Centro de la CNC.

3.4.3 Cambio de base del área de trabajo.

La referencia de bases que tienen tanto el robot y la maquinas CNC deben ser acoplados de tal manera que coincidan en el mismo punto de referencia en ambas máquinas. Una máquina CNC tiene dos puntos de referencia que son el cero máquina (que es el centro de la región de trabajo) y el cero pieza (que es el punto de referencia de la pieza). Estos puntos son

indicados en el robot mediante el programa para poder referenciar los comandos que posteriormente que se vayan a generar en el lenguaje AS.

3.4.3.1 Cero de Máquina

En CNC este punto hace referencia al centro de toda la región de trabajo y es el punto (X = 0, Y = 0, Z = 0), como se tiene definida los límites de la región de trabajo el cero maquina estará definida en el centro del prisma rectangular, este punto central del prisma está en el punto (X = -400, Y = 0, Z = -30) según la referencia del robot.

Se tiene que el punto cero máquina de la CNC (X = 0, Y = 0, Z = 0) está definido en el punto (X = -400, Y = 0, Z = -30) del robot, por lo tanto, se hace un cambio de base mediante programa en el robot de tal manera que su nueva base sea el punto (X = -400, Y = 0, Z = -30).

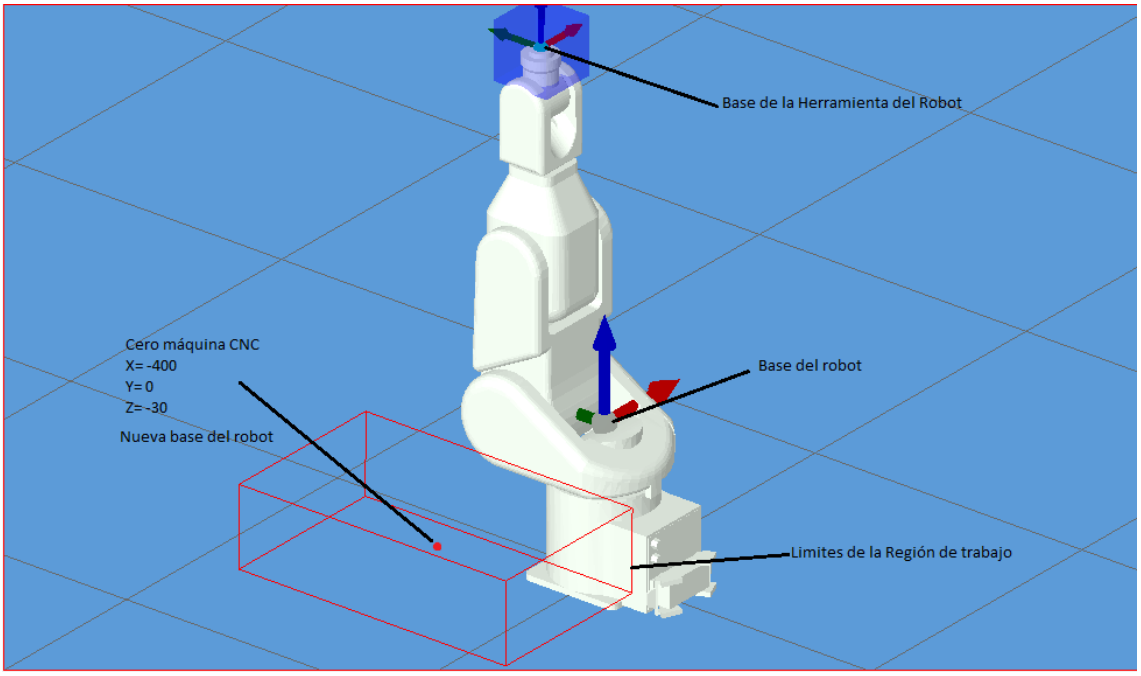

Figura 3.19: Regiones de trabajo del brazo robot como CNC

3.4.3.2 Cero de Pieza

En CNC este es el punto que define la posición geométrica de la pieza, de tal manera que la máquina reconozca la posición del plano en que se ubica la pieza y posteriormente empezar el mecanizado.

Para ubicar este punto en el robot se realiza un procedimiento manual, tal como se hace en las CNC, que consiste en llevar la punta de la herramienta a un punto de referencia de la pieza que por facilidad suele ser un vértice superior. Una vez que el robot detecta la ubicación del punto vértice de la pieza, por programa se espera la posición actual del robot para tener la ubicación geométrica de la pieza en el plano. Posteriormente obtenido está posición el robot procede a realizar todo el proceso de mecanizado.

3.4.4 Procedimiento para referir el punto cero pieza.

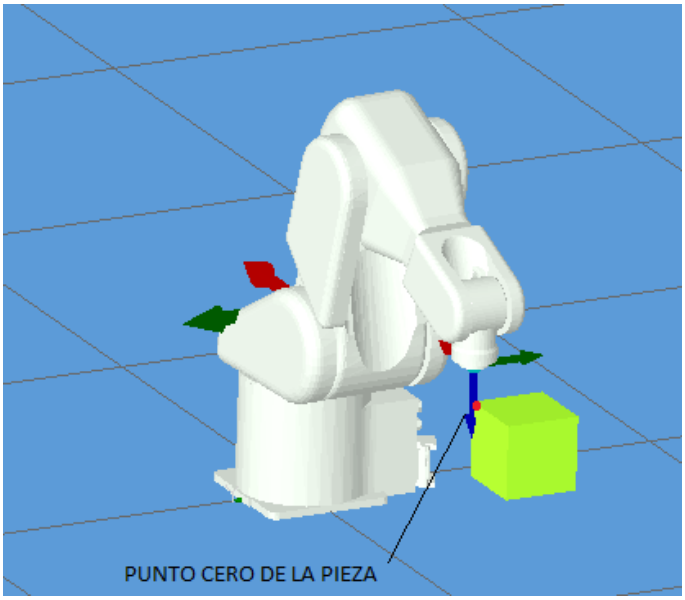

Figura 3.20: Representación Gráfica del Punto Cero de la Pieza.

Consiste en definir un punto origen que se sitúe en la pieza a mecanizar y enviar las coordenadas de este punto al robot para posteriormente procesar

todos los movimientos referidos a este nuevo origen. Usualmente si la pieza a mecanizar es un cubo, se escoge como referencia un vértice superior.

Considerar que dicho punto en la pieza debe ser definido previamente en el software CAD/CAM de la pieza a diseñar, para que el código G generado y posteriormente código AS se refieran a dicho punto.

El procedimiento consiste el manipular manualmente el robot con el TEACH PENDANT y llevar la herramienta hasta el punto en donde será el nuevo origen. Una vez llegada la punta de la herramienta al dicho punto, se obtiene la posición actual mediante el comando HERE del robot y se define esta posición como la nueva base u origen al que van a estar referidas todas las coordenadas del robot.

3.4.5 Validación de Coordenadas

Para asegurarse de que las coordenadas necesarias para mecanizar la pieza no exceden los límites del volumen de trabajo se realiza la validación de cada una de las coordenadas del programa de pieza. Este procedimiento se realiza al momento de la generación del archivo AS.txt y para realizarlo se requiere las coordenadas del cero de la pieza con respecto a las coordenadas base del robot. Estas coordenadas se obtienen al momento de establecer la comunicación con el brazo robot. De esta manera se suma cada coordenada del archivo de pieza a este punto cero y se verifica que el resultado esté dentro del volumen de trabajo. Si existe alguna coordenada que esté fuera de rango, el programa muestra un mensaje de error y posteriormente finaliza la conexión con el brazo robótico.

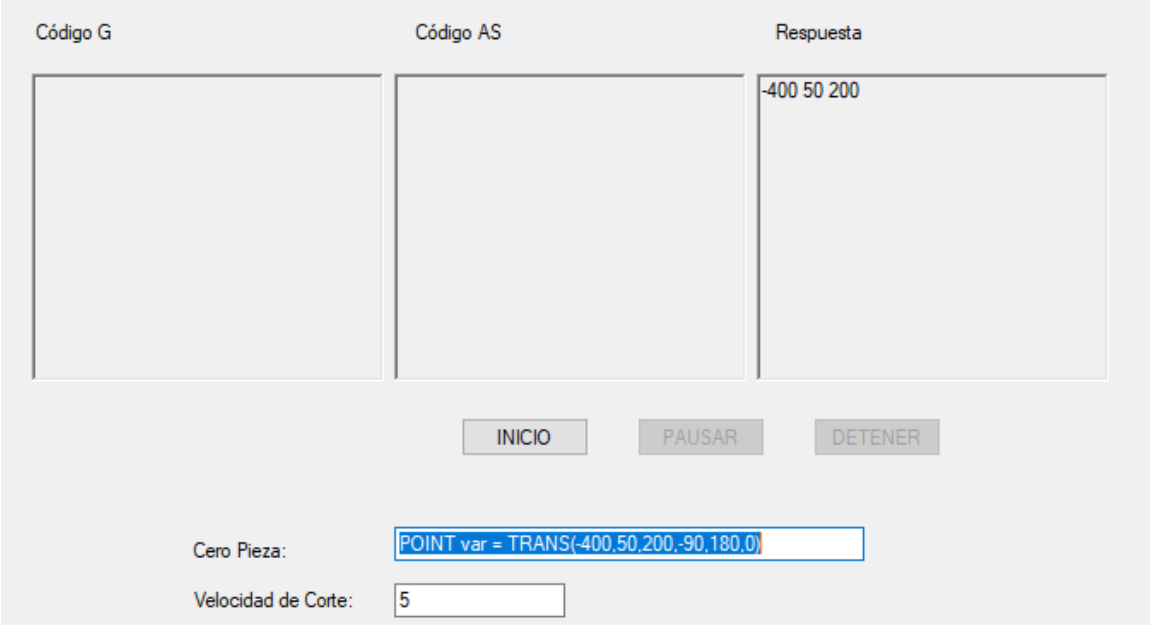

Figura 3.21: Recepción de coordenadas de cero de pieza con respecto a las coordenadas base del brazo robot.

3.5 Motor de Fresado

Para realizar el fresado es necesario un motor de husillo. Se adquirió un motor que fue seleccionado principalmente por sus características físicas como dimensiones, tipo de montaje y peso. Estas características fueron consideradas en primer lugar debido a las limitaciones de carga y el tipo de montaje disponible en el brazo robot. Por lo tanto, se consideraron las siguientes características físicas:

Longitud	<220 mm
Peso	<1.2 Kg
Montaje	Abrazadera

Tabla 6: Tabla de Características Físicas Requeridas del Motor.

La velocidad mínima requerida para el motor es de 5000 rpm debido a los materiales que se van trabajar y además se requiere que la fuga del rotor sea mínima para disminuir el error. Por lo tanto, las características del motor seleccionado son:

Longitud	200 mm
Peso	1.1 Kg
Montaje	Abrazadera
Fuga del Eje	0.01-0.03 mm
Velocidad	Hasta 12000 rpm
Voltaje de Operación	100 Vdc
Potencia	400 W
Diámetro	52 mm

Tabla 7: Tabla de Características del Motor Adquirido.

Adicionalmente se adquirió una fuente de alimentación para el motor. La fuente seleccionada cumple los requerimientos eléctricos para energizar el motor y además cuenta con un potenciómetro para regular la velocidad del motor. La tabla 8 muestra las características eléctricas de la fuente de alimentación.

Voltaje de Entrada	**110 Vac – 50/60 Hz**
Voltaje de Salida	**0-100 Vdc**
Potencia	**500 W**
Potenciómetro de Control	**0-10 Vdc**

Tabla8: Tabla de Características Eléctricas de la Fuente de Voltaje.

3.5.1 Montaje del Motor en el Brazo Robot

Para realizar el montaje del motor sobre el brazo robot se adquirió una abrazadera de 52 mm para luego acoplarla a un ángulo y a su vez éste se acopló al terminal del brazo robot. Se realizaron las perforaciones correspondientes en el ángulo para atornillar la abrazadera y para fijarlo a la superficie de montaje en el robot. En la siguiente figura se muestra el montaje realizado en el brazo robot:

Figura 3.22: Montaje del Motor en el Brazo Robot.

3.6 Estructura del Área de Trabajo

La estructura implementada consiste en una base que permita mantener firme y alineada la pieza a mecanizar. La estructura permite sostener piezas de madera o PCB de tal manera que permita al robot poder moverse libremente dentro de los límites establecidos.

Se considera que la estructura debe de estar dentro del rango de los límites del robot definidos anteriormente, y además de permanecer alineado a los ejes X Y Z de la base del robot.

Figura 3.23: Mesa de Mecanizado.

CAPÍTULO 4

4. RESULTADOS

El presente capítulo muestra los resultados obtenidos en la implementación del sistema de fresadora CNC usando el brazo robot, se detalla el comportamiento del sistema y el desempeño obtenido en base a la metodología utilizada y alcance establecido previamente.

Para el análisis de los resultados se contrastan los modelos gráficos digitales diseñados en el ordenador y la pieza mecanizada por el brazo robot. Se comparan medidas, trayectorias lineales y curvas, acabado, velocidad de corte y similitudes generales entre ambas piezas.

Como un análisis secundario, se detalla los resultados obtenidos durante cada etapa de del proceso, los errores que se presentaron y las soluciones para corregirlos.

4.1 Procesamiento de Códigos

En esta etapa del proceso se obtiene como resultado la lista de códigos AS que posteriormente son enviados al controlador de comunicación mediante comunicación TCP/IP. El programa interpreta el código G a diseñar y lo transforma al código AS simplificado. La conversión de códigos no utiliza muchos recursos del computador y se realiza de manera rápida.

Entre las limitaciones del programa están una lista de comandos G y M de carácter especial, que son usados para trabajo en tornos CNC, CNC de 5 eje y 7 ejes, dicha limitación está definida previamente en el alcance del proyecto, sin embargo el programa diseñado permite su funcionamiento ante estas posibles fallos, cualquier comando no reconocido, ya sea por error humano o porque que el tipo de CNC no es de 3 ejes, el programa detecta el comando y no realiza función alguna continuando con la ejecución total del programa. Cabe recalcar que el programador

debe asegurarse de que el código que genera con el programa CAD/CAM debe ser para fresadora CNC de 3 ejes.

4.1.1 Tiempo de procesamiento de datos y tamaño de archivo generado

El programa tarda cierto tiempo en convertir los códigos G en código AS simplificado. Este tiempo depende del computador que se utilice y del tamaño del programa de pieza a procesar. Un mismo archivo convertido en dos computadores distintos puede tardar tiempos diferentes.

A continuación, se muestra un ejemplo de tiempo de procesamiento de un archivo de programa de pieza relativamente extenso (más de 33 mil líneas).

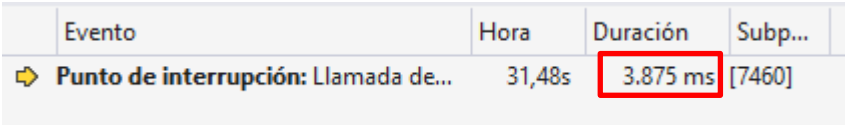

Figura 4.1: Ejemplo de duración de tiempo de procesamiento de datos.

Como se observa en la figura el tiempo que tarda la conversión de códigos es aproximadamente 4 segundos. Por otro lado, continuando con el ejemplo anterior, el archivo AS.txt generado es de mayor tamaño dado que en códigos G las trayectorias circulares se representan en una sola línea mientras que en código AS, se realiza una interpolación de puntos lo que implica más de una línea. A continuación, se muestra la comparación entre el programa de pieza y el archivo AS.txt generado.

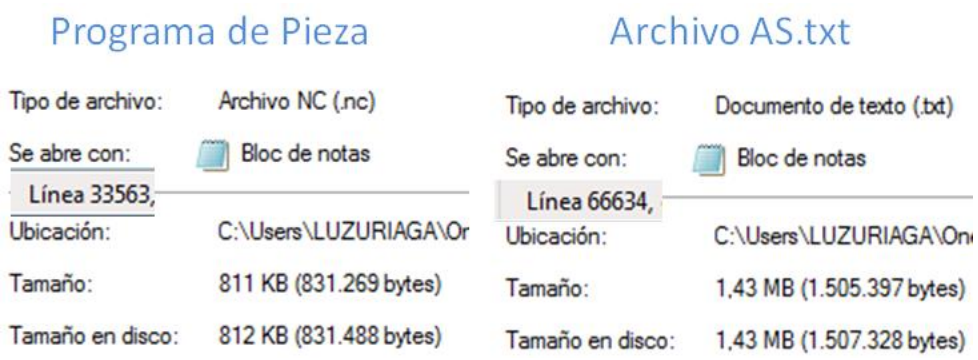

Figura 4.2: Comparación entre archivo de Control Numérico (.NC) y archivo generado (AS.txt).

Como se observa, las extensiones de los archivos son distintas. Por un lado, se tiene una extensión de archivo generado por el CAD/CAM .NC (numeric control) y luego el archivo de texto .TXT generado por el programa desarrollado para el proyecto.

Además, otra diferencia es el tamaño de los archivos. Como se mencionó anteriormente, el archivo generado tiende a contener más líneas que el archivo original debido a la interpolación de puntos de trayectorias circulares. Sin embargo, el archivo de texto generado puede ser desechado una vez que se ha terminado la mecanización debido a que éste se volverá a generar si se pretende realizar nuevamente el proceso.

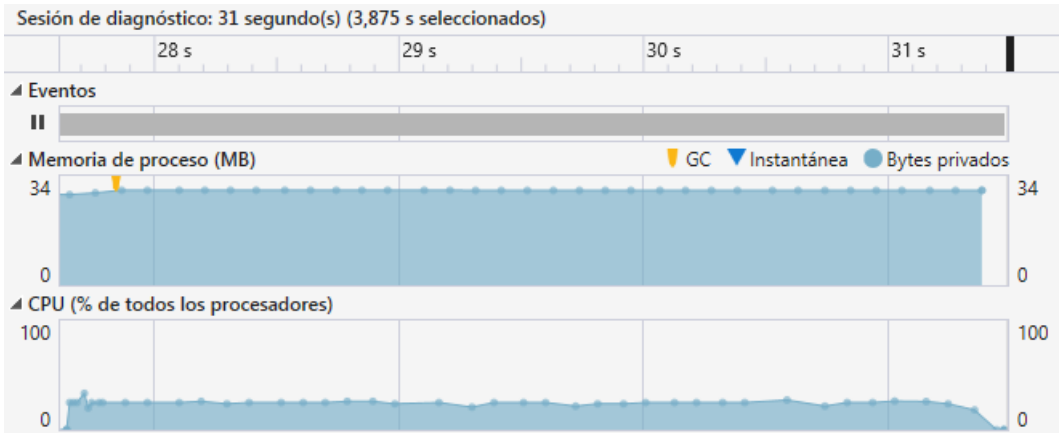

Figura 4.3: Recursos de PC utilizados por la aplicación desarrollada.

La figura muestra la memoria y el porcentaje de capacidad de procesamiento del computador que utilizó el programa para realizar la conversión de códigos para el ejemplo dado. El pico máximo de memoria es de 34 MB mientras que el pico de porcentaje de procesamiento es de 33%.

4.1.2 Códigos Generados

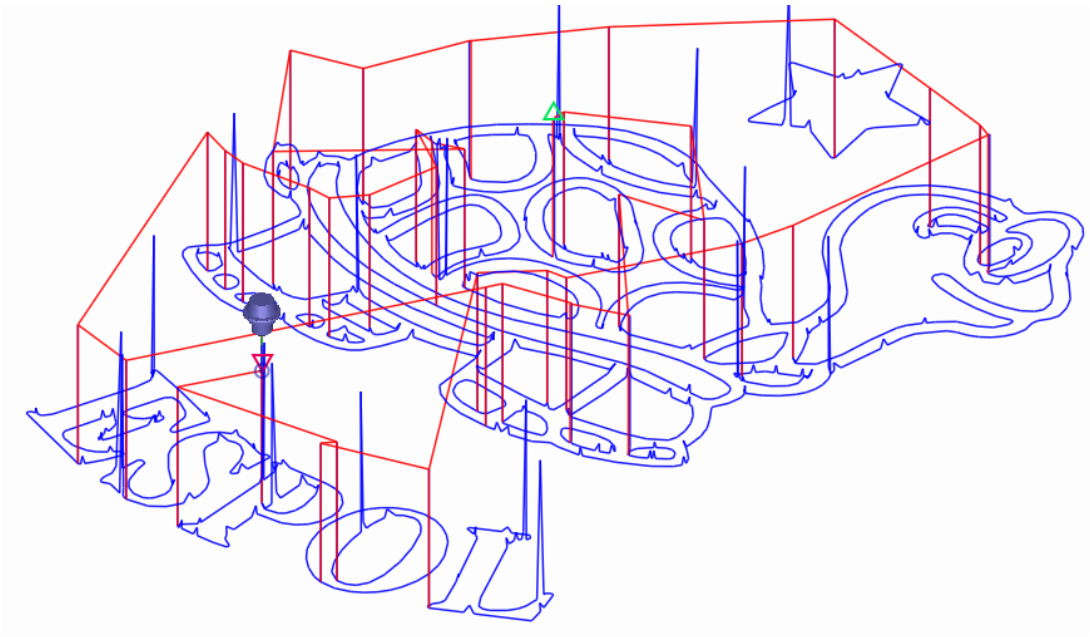

Figura 4.4: Trayectorias de CNC (logo ESPOL)

Como prueba para el mecanizado del proyecto se escogió una imagen del logo de la ESPOL, la cual se tuvo que procesar previamente, en cuanto a cuestiones de tamaño, vectorizado de trayectorias y tipo de herramienta mediante el software CAD/CAM. El resultado de esta etapa del proyecto es la lista códigos AS simplificados correspondiente a las trayectorias de la imagen en un archivo de texto. En la figura tal se muestra la lista de códigos G y la lista de código AS simplificado equivalente.

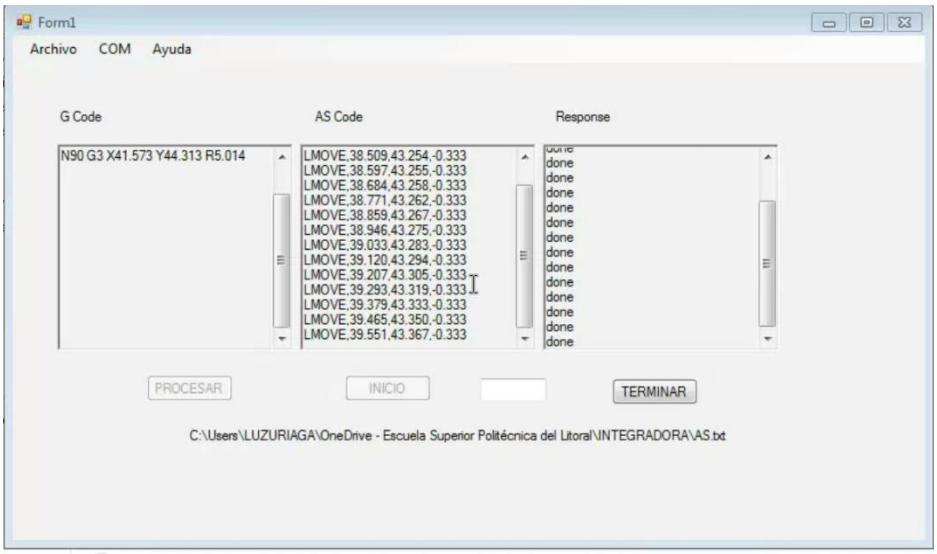

Figura 4.5: Interfaz de usuario del programa.

Al realizar la conversión de códigos surgieron inconvenientes en las trayectorias curvas. Se presentaron problemas al realizar la conversión de los comandos G2 y G3 para cuando las trayectorias circulares era un semicírculo o había un arco de 90 grados, lo que matemáticamente presentaría en la formulas un error de división por “0”. Para solucionar este problema se sumó un pequeño valor de “0.0001” a las variables X Y Z quedando como resultado final X+0.0001, Y+0.0001, Z+0.0001. Considerando este cambio el programa funciona aproximándose a las trayectorias indicadas con un pequeño margen de error. Esta desviación no interfiere en el resultado final debido a que es 1 micrómetro de diferencia y el robot no detecta este cambio por lo que se conserva la exactitud de las trayectorias. Sin embargo, el programa lo detecta necesariamente para su correcto funcionamiento.

El cambio de la referencia del robot o cambio de base, por defecto en el robot no era compatible con las coordenadas del código CNC, entre estos inconvenientes se presentó que el eje Z del robot estaba invertido con respecto al eje Z del código de la CNC, los ejes X Y del robot estaba

intercambiados entre sí por Y X respectivamente con respecto al código de la CNC, para solucionar este problema, se tuvo que intervenir el código del programa y cambiar a las variables que guardaban de estas coordenadas. Para su compresión matemática se aplicó la siguiente conversión en el programa.

$$X_{robot} = X_{cnc}$$

$$Y_{robot} = -Y_{cnc}$$

$$Z_{robot} = -Z_{cnc}$$

Se tiene en cuenta la importancia de definir este punto base o de referencia, debido a que de este punto van a depender todas las coordenadas del programa, el punto de referencia debe de coincidir en el objeto con el mismo punto con el cual se hizo el código G de la pieza en el software de mecanizado. En caso de no concordar su posición pueden surgir problemas tales como: tener una figura desalineada, trayectorias fueras de los límites del área de trabajo, trayectorias de la imagen que se han perdido fuera de la pieza, entre otros. Es importante definir este punto de manera correcta al inicio del programa.

Otro problema que se presentó y que pueden afectar el resultado del código AS del programa, fuel al momento de tratar el archivo .txt del código G, ya sea por error humano por manipulación del archivo, ocurra el caso de que se ingresen caracteres sin ningún sentido en el programa, lo cual puede cambiar los valores numéricos de las coordenadas y lo interpretara como error. Para ello se debe evitar manipular los datos de los códigos generados. En caso de que se necesiten agregar una línea adicional de comandos G, se debe tener conocimientos de programación en CNC para evitar posibles fallos en la programación.

4.2 Implementación y fresado de piezas

Como resultado final del proyecto se obtiene la pieza o modelo mecanizado. Entre los infinitos tipos de piezas y formas que se pueden diseñar, el brazo robot no tiene limitación alguna y puede realizar la trayectoria de fresado de la imagen que se le presente.

Los resultados obtenidos entre las figuras realizadas muestran que no surgieron problemas al momento de convertir cualquier tipo de trayectoria de código G, el programa y el protocolo de comunicación funcionan con fluidez y exactitud, pudiendo realizar las trayectorias de cualquier índole sin problema alguno.

En la figura 4.6 se muestra el proceso de fresado del logo de ESPOL. La imagen se detalló previamente en el proceso de conversión de código G a código AS. Se observa cómo el robot realiza las mismas trayectorias que se señalan en la figura 4.4.

Figura 4.6: Ejemplo de grabado sobre madera (Logo ESPOL).

Entre los problemas surgidos en la parte final del proceso, se presentó un desbalance en la herramienta por el tipo de agarre que tenía con el brazo, lo cual era crítico para la fuerza de corte del tallado, lo que suponía desviaciones graves en la trayectoria. Para solucionarlo se implementó un nuevo tipo de agarre más sólido para mantener totalmente fija la herramienta al brazo sin desviación alguna como se muestra en la figura 4.7.

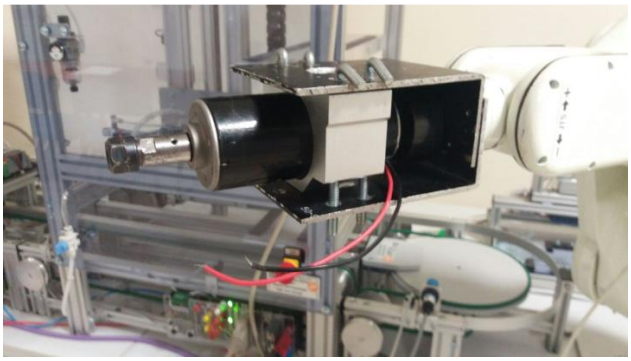

Figura 4.7: Montaje de la fresadora en el brazo robot.

4.3 Comunicación

Como se observa en la figura 4.6 la media del tiempo de respuesta del brazo robot es de 1ms. El tiempo de envío y recepción de comandos es suficiente para que el brazo robot siempre esté realizando un movimiento. Si existen retardos mayores en la comunicación, el mensaje no se pierde debido a que el protocolo TCP/IP se asegura de que el mensaje llegue de manera correcta.

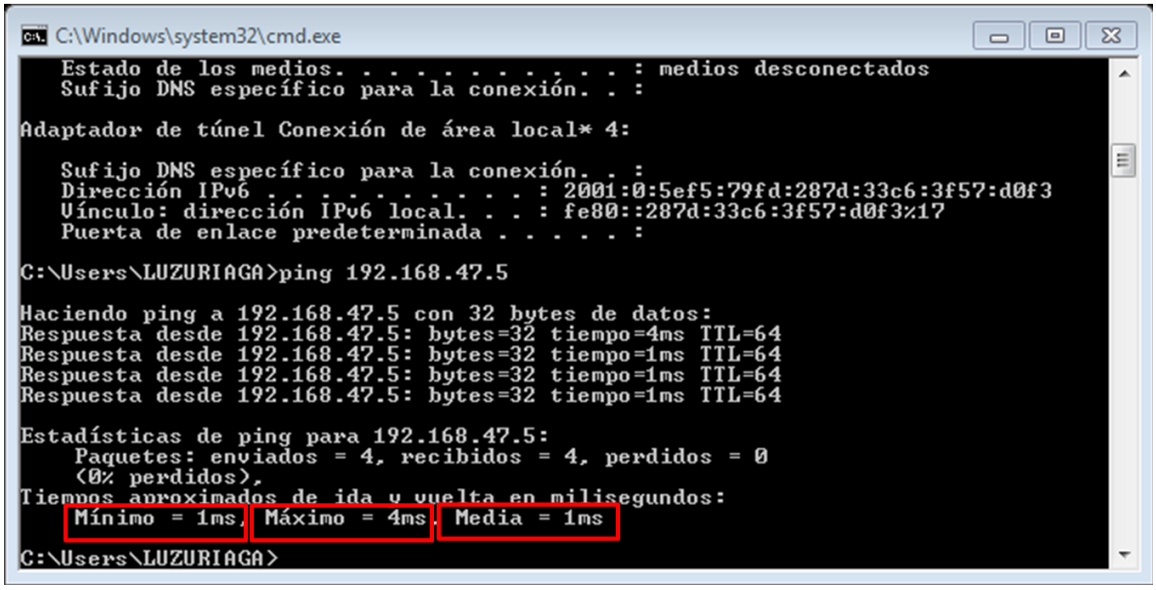

Figura 4.8: Prueba de Comunicación TCP/IP con el Brazo Robot.

Uno de los problemas, en cuanto a comunicación, se presentó al recibir datos desde el brazo robot debido a que cada vez que de recibía información, ésta se almacenaba en un buffer distinto, es decir, cada vez que se recibía un dato, se creaba un buffer de lectura y la información se almacenaba de manera separada. Esta opción resultó ser no compatible con el brazo robot, por lo que se optó por utilizar un único buffer de entrada de tamaño fijo. El tamaño seleccionado fue de 64 bytes, lo que es suficiente para la información que se recibe desde el brazo robot.

```
Public Sub AwaitData()
    _Stream.BeginRead(_Buffer, 0, _Buffer.Length, AddressOf DoReadData, Me)
    '_Stream.BeginRead(New Byte() {0}, 0, 0, AddressOf DoReadData, Me)
End Sub
```

Figura 4.9: Segmento de Código utilizado para Recepción de Datos.

En la figura se observa en verde la línea de código que cada vez que se recibe información, crea un buffer de tamaño indeterminado. Esta línea fue sustituida por el buffer de tamaño fijo _Buffer. De esta manera se logró recibir la información enviada por el robot.

Cuando se tiene un único buffer de entrada, es necesario asegurarse de que no llegue más de un mensaje a la vez ya que, dado el caso, los mensajes se guardarán en el buffer no se podrá distinguir un mensaje del otro. Por este motivo, el robot envía la respuesta y no vuelve a enviar más mensajes hasta haber recibido un comando. De esta manera el buffer de entrada contendrá un solo mensaje a la vez.

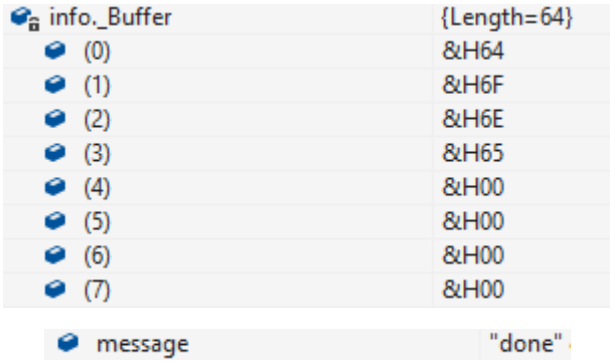

Figura 4.8 Verificación del Contenido del Buffer de Entrada.

La figura 4.8 muestra un mensaje recibido en el buffer de entrada. El mensaje tiene una longitud de 4 bytes, es por esto que los 60 bytes restantes del buffer de entrada están vacíos. Luego se muestra la variable *message* con el mensaje codificado en código ASCII.

CONCLUSIONES Y RECOMENDACIONES

Un brazo robot de seis ejes, al ser una máquina de control numérico computarizado, puede realizar los movimientos de una fresadora CNC de tres ejes. Para esto se requiere conocer el comportamiento de ambos equipos con el fin de entender las analogías que existen entre sus distintos movimientos, así como también los sistemas de referencia y sus lenguajes de programación. A pesar de que el brazo robot no cuenta con comandos de movimiento circular equivalentes a los comandos de las CNC de tres ejes, se logró implementar dichos movimientos utilizando interpolación de puntos a lo largo de las trayectorias. De esta manera se logra programar los movimientos del brazo robot usando lenguaje estándar de máquinas CNC, lo que permite a los estudiantes el aprendizaje práctico de los procesos de mecanizado básicos además de la oportunidad de utilizar programas informáticos de diseño y manufactura asistidos por computadora.

La fresadora acoplada al brazo robot puede trabajar en distintos planos dependiendo de la orientación que se la programe. Para cada orientación será necesario establecer nuevos límites para el volumen de trabajo. Esta ventaja se presenta debido a que el brazo robot utilizado posee 6 grados de libertad, lo que permite rotar los ejes X, Y, Z en distintas direcciones brindando la opción de adaptar los ángulos de giro de los ejes para nivelar la fresadora con respecto a la mesa de trabajo. Por otro lado, utilizando la rotación de los ejes, se puede seguir la misma metodología de trabajo para aumentar los ejes de la fresadora y de esta manera aprovechar al máximo la capacidad del brazo robot y a su vez poder mecanizar piezas más complejas en cuanto a diseño y desarrollo de estrategias de mecanizado.

Mediante la conversión de códigos e interpolación de trayectorias se logró utilizar archivos con programación estándar de máquinas CNC generados utilizando programas CAD/CAM. El archivo resultante de la conversión de códigos G a códigos AS simplificados es de mayor tamaño debido a que incluye las líneas de código G,

además de que por cada línea con comandos de trayectorias circulares en código G, se genera más de una línea de código AS simplificado. Se utilizó comunicación TCP/IP para el envío de comandos de movimiento, de esta manera se supera la limitación de memoria del brazo robot, lo que permite ejecutar programas de pieza extensos ya sea por el tamaño de la pieza, el nivel de detalle de los acabados deseados e incluso ambas.

Para obtener mejores resultados, se recomienza fijar el brazo robot a la base de mecanizado de manera que estén nivelados entre sí y adicionalmente asegurarse de que los equipos estén sujetos de manera firme a las estructuras para evitar vibraciones que interfieran en las trayectorias de mecanizado.

Se validaron movimientos dentro de los límites establecidos para la CNC para evitar colisiones de la fresadora con el brazo robot y para evitar errores de coordenadas fuera de rango del robot. Sin embargo, aún existe la posibilidad de colisiones entre la fresadora y las herramientas de sujeción o la pieza a mecanizar, por lo que se recomienda aprender previamente sobre el proceso de mecanización y adicionalmente utilizar herramientas informáticas de simulación para verificar que las trayectorias no generen colisiones.

Mientras el proceso de mecanizado está en marcha, se generan virutas del material que se está trabajando. En el laboratorio existen equipos que podrían verse afectados por estos desperdicios así que se recomienda utilizar una aspiradora para extraer el material residual de manera segura y evitar posibles afectaciones en los equipos.

BIBLIOGRAFÍA

[1] Transparency Market Research, “Computer Numerical Controls (CNC) Market: Global Industry, Size, Share, Growth, Trends and Forecast, 2016-2024,” TMR., New York, NY, Tech. Rep., Nov. 2016.

[2] Denford, Computerised Machines and Systems, “G and M Programming for CNC Milling Machines”, Denford Limited, England, 2012.

[3] International Standard ISO2806 Industrial Automation System – Numerical Control of Machines – Vocabulary, International Organization for Standardization.

[4] The Hong Kong Polytechnic University, Industrial Centre, “Computer Numerical Control (CNC)”, IC Learning Series, Hong Kong, Marzo 2012, pp. 1-8.

[5] Autodesk CAM, “Fundamentals of CNC Machining”, 2012.

[6] B.S. Pabla, M. Adithan, “CNC Machines”, New Age International (P) Limited, New Delhi, 1995, pp. 10-23.

[7] Kawasaki Heavy Industries, Ltd, “RS003 robot datasheet,” 2013. [En línea]. Available: https://robotics.kawasaki.com/userAssets1/productPDF/RS003N.pdf.

[8] Kawasaki Heavy Industries, Ltd, “Kawasaki Robot Controller E Series Operation Manual,” Estados Unidos, 2013.

[9] Kawasaki Heavy Industries, Ltd, “Kawasaki Robot Controller E Series AS language Reference Manual,” Estados Unidos, 2010.

[10] Kawasaki Heavy Industries, Ltd, “Kawasaki Robot Controller E Series TCP/IP Communication Manual,” Estados Unidos, 2013.

[11] K. Gaughan. (2003) *Client Server Programming with TCP/IP Sockets* [Online]. Avaible FTP: stereochro.me/assets/uploads/notes/dcom3/sockets.pdf

[12] Microsoft. (2010) *TcpClient Class* [Online]. Avaible: https://msdn.microsoft.com/en-us/library/system.net.sockets.tcpclient(v=vs.110).aspx

ANEXOS

Aplicación desarrollada en Visual Basic

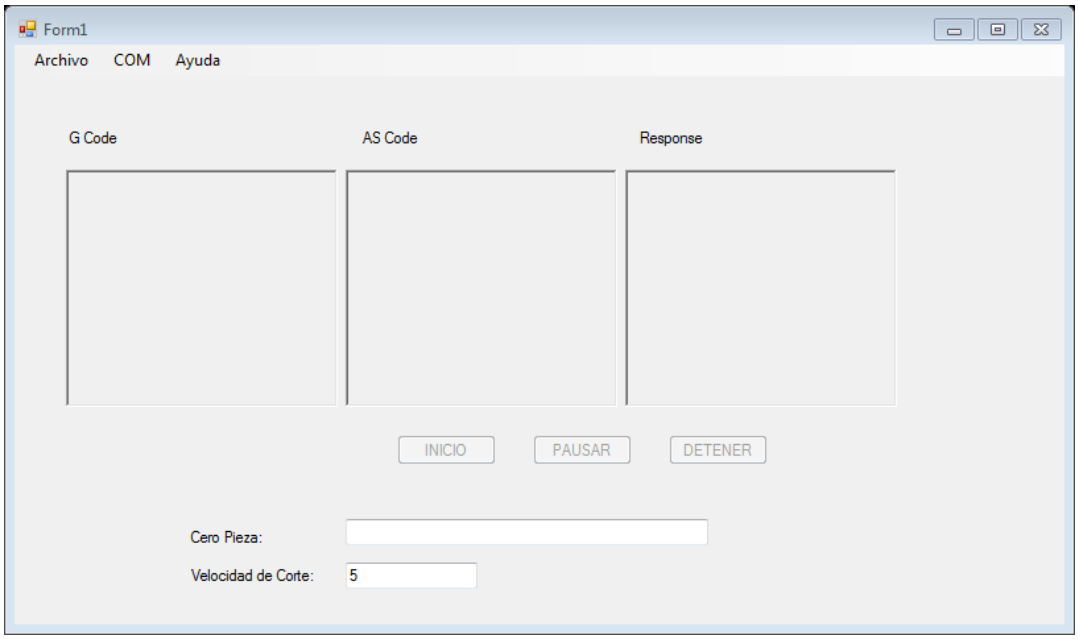

Figura 1: Ventana principal de aplicación desarrollada en Visual Basic.

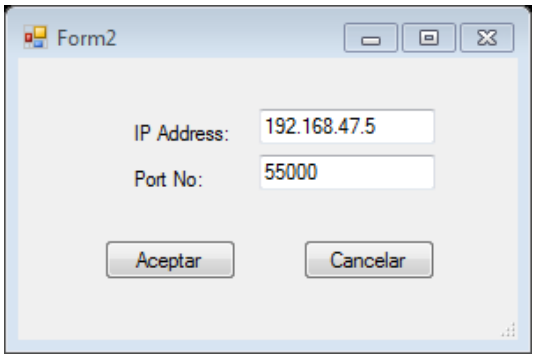

Figura 1: Ventana principal de aplicación desarrollada en Visual Basic.

A continuación se enlista la programación:

```
Imports System.Net
Imports System.IO
Imports System.Net.Sockets
Imports System.Windows
Imports System.Windows.Input
Imports WindowsApplication2.Form2
Public Class Form1 8
    Private _Connection As ConnectionInfo
    Dim port As String
    Dim ipadd As String
    Private Shared response As String = String.Empty
    Private Shared sendDone As New ManualResetEvent(False)
    Private Shared readDone As New ManualResetEvent(False)
    Delegate Sub _xUpdate(ByVal str As String)
    Private fileReaderOUT As StreamReader
    Dim _enable_Send As Boolean = False
    Dim commandFile As NCparse
    Dim _on_proccess As Boolean = False 20
    Dim Xbase As Double
    Dim Ybase As Double
    Dim Zbase As Double
    Dim coord(2) As String
    Dim c_speed As Double = 5 26
    Private Sub hndCadenaRecibida(str As String) 28
        If RichTextBox3.TextLength > 0 Then
            RichTextBox3.AppendText(ControlChars.NewLine)
        End If
        RichTextBox3.AppendText(str)
        RichTextBox3.ScrollToCaret() 34

        If str.Contains(" ") Then 37
            If PlayButton.Enabled = False Then
                PlayButton.Enabled = True
            End If
            coord = str.Split(" ") 42
            Xbase = Val(coord(0))
            Ybase = Val(coord(1))
            Zbase = Val(coord(2))
            TextBox1.Text = "POINT var = TRANS(" & Xbase.ToString & "," & Ybase.ToString & ","
              & Zbase.ToString & ",-90,180,0)"
        End If

        Select Case str 50
            Case "Esperando inicio"
```

```
52                      If PlayButton.Enabled = False Then
53                            PlayButton.Enabled = True
54                      End If Case
55                "done"
56                      If on_proccess() Then
57                            If enable_Send() Then  sendCommand()
58                            End If End
59                      If
60          End Select
61
62
63
64      End Sub
65      Public Sub SetUpEventHandler()
66            AddHandler CadenaRecibida, AddressOf hndCadenaRecibida End Sub
67       Public Sub RaiseCadenaRecibidaEvent(str As String) RaiseEvent
68            CadenaRecibida(str)
69      End Sub
70       Private Sub appendOutputs(appendedLine As String, who As Boolean)  If who Then
71                  If RichTextBox1.TextLength > 0 Then
72                        RichTextBox1.AppendText(ControlChars.NewLine)
73                  End If  RichTextBox1.AppendText(appendedLine)
74                  RichTextBox1.ScrollToCaret()
75             Else
76                  If RichTextBox2.TextLength > 0 Then
77                        RichTextBox2.AppendText(ControlChars.NewLine)
78                  End If  RichTextBox2.AppendText(appendedLine)
79                  RichTextBox2.ScrollToCaret()
80            End If  End
81      Sub
82       Private Sub sendCommand()  Dim
83            toSend As String  Dim gLine As
84            String  Dim line As String
85            line = fileReaderOUT.ReadLine()
86            If String.IsNullOrEmpty(line)  Then
87                  appendOutputs("END",  False)
88                  appendOutputs("END",  True)  toSend = "EXIT"
89                  send(toSend)
90                  _Connection.Close()
91                  _Connection = Nothing  ConectarToolStripMenuItem.Text  = "Connect"
92                  fileReaderOUT.Close()
93                  fileReaderOUT = Nothing
94                  commandFile.close()  commandFile =
95                  Nothing
96            Else
97                  gLine = line.Remove(line.IndexOf("'"), line.Length() - line.IndexOf
98
99
100
101
102
103
```

```
                    ("'"))
                    toSend = line.Substring(line.IndexOf("'") + 1)
104                 appendOutputs(gLine, True) appendOutputs(toSend, False)
105                 If enable_Send() Then send(toSend)
106                 End If End
107           If
108           If toSend = "EXIT" Then
109                 MessageBox.Show("Fin. ", "My 3 axis Milling Robot", MessageBoxButtons.OK,
110                 MessageBoxIcon.Information)
111         End If End
112     Sub
113      Sub xUpdate(ByVal str As String) If
114           InvokeRequired Then
115                 Invoke(New _xUpdate(AddressOf xUpdate), str) Else
116                 Dim i As Integer = 0 response = ""
117                 If Not String.IsNullOrEmpty(str) Then While
118                       str.Chars(i) <> vbNullChar
119                             i = i + 1 End
120                       While
121
122
123
124                       response = str.Remove(i, 64 - i)
125                       RaiseCadenaRecibidaEvent(response)
126                 End If End
127           If
128     End Sub
129
130
131
132      Private Sub send(ByVal str As String) Try
133                 _Connection.sWriter.WriteLine(str)
134                 _Connection.sWriter.Flush() Catch ex As
135           Exception
136                 MessageBox.Show("Error al enviar. " & ex.ToString, "My 3 axis Milling  Robot",
137                 MessageBoxButtons.OK,  MessageBoxIcon.Information)  End Try
138     End Sub
        Public Property mIpadd() As String Get
                    Return Me.ipadd End
139           Get
140           Set(value As String)
141                 Me.ipadd = value
142         End Set End
143     Property
144      Public Property mPort() As String Get
145                 Return Me.port End Get
146
147
148
149
150
151
152
153
```

```
154             Set(value As String) Me.port
155                   = value
156             End Set End
157         Property
158          Public Property enable_Send() As Boolean Get
159                   Return Me._enable_Send End Get
160             Set(value As Boolean) Me._enable_Send = value
161             End Set End
162         Property
163          Public Property on_proccess() As Boolean Get
164                   Return Me._on_proccess End Get
165             Set(value As Boolean) Me._on_proccess = value
166             End Set End
167         Property
168         Public Sub ExitApplication()
169               If MessageBox.Show("Salir?", "My 3 axis Milling Robot", MessageBoxButtons.YesNo,
170                MessageBoxIcon.Question) _
171                = DialogResult.Yes Then
172                   If ConectarToolStripMenuItem.Text = "Desconectar" Then    If _Connection
173                         IsNot Nothing Then _Connection.Close()
174                         _Connection = Nothing ConectarToolStripMenuItem.Text =
175                         "Conectar"
176                   End If
177                   Application.Exit()
178             End If End
179         Sub
180
181
182
183          Private Sub SalirToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
184             SalirToolStripMenuItem.Click
185              ExitApplication() End Sub
186
187
188          Private Sub OpcionesToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
189              OpcionesToolStripMenuItem.Click
190               Dim dlgCom As New Form2 dlgCom.ShowDialog()
191         End Sub
             Private Sub AbrirToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
                AbrirToolStripMenuItem.Click
192               Dim dlg As New OpenFileDialog
193               dlg.FileName = "" dlg.DefaultExt =
194               ".nc"
195               dlg.Filter = "NC documents (.nc)|*.nc" & "|ANC documents (.anc)|*.anc" Dim result As Boolean =
                  dlg.ShowDialog()
                  If result = True Then Try
196
197
198
199
200
201
202
```

```
203                         If Not IsNothing(commandFile) Then commandFile.close()
204                         End If
205                         If Not IsNothing(fileReaderOUT) Then fileReaderOUT.Close()
206                         End If
207                         commandFile = New NCparse(dlg.FileName, dlg.FileName.Remove
208                            (dlg.FileName.IndexOf(dlg.SafeFileName), dlg.SafeFileName.Length) + "AS.txt", coord)
209                         Label4.Text = dlg.FileName.Remove(dlg.FileName.IndexOf (dlg.SafeFileName),
                               dlg.SafeFileName.Length) + "AS.txt"
                            MessageBox.Show("Archivo seleccionado: " & dlg.SafeFileName, "My
                               3 axis Milling Robot",
                            MessageBoxButtons.OK, MessageBoxIcon.Information) Catch Ex As
                      Exception
210                         If result = False Then
                                  MessageBox.Show("No se puede leer el archivo. Error: " & Ex.Message, "My 3 axis
                                   Milling Robot",
211                         MessageBoxButtons.OK, MessageBoxIcon.Information) End If
                      End Try Else


212
213               End If End
214           Sub
215


              Private Sub ConectarToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
216               ConectarToolStripMenuItem.Click
217                If ConectarToolStripMenuItem.Text = "Conectar" And IsNothing(_Connection) Then
218                     Try
219   again:                  _Connection = New ConnectionInfo(mIpadd(), CInt(mPort()), AddressOf xUpdate)
220                           _Connection.AwaitData() ConectarToolStripMenuItem.Text =
221                           "Desconectar"
222                           MessageBox.Show("Conexión establecida.", "My 3 axis Milling Robot",
223                           MessageBoxButtons.OK, MessageBoxIcon.Information) Catch ex As
224                     Exception
                              If MessageBox.Show("No se puede establecer la conexión.", "My 3 axis Milling Robot",
                              MessageBoxButtons.RetryCancel, MessageBoxIcon.Error) _
225                           = DialogResult.Retry Then GoTo again
                              End If
                      End Try
                   Else
226                     If _Connection IsNot Nothing Then _Connection.Close()
227                     _Connection = Nothing ConectarToolStripMenuItem.Text =
                        "Conectar" Button1.Enabled = False
                        Button2.Enabled = False
228
229
230



231
232
233



234
235
236
237
238
239
240
241
242
243
244
```

```
245                     PlayButton.Enabled = False End If
246         End Sub
247
248
249
250          Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load  ipadd = "192.168.47.5"
251               port = "55000"
252               TextBox2.Text = "5"  PlayButton.Enabled =
253               False SetUpEventHandler() Button2.Enabled =
254               False Button1.Enabled = False
255         End Sub
256
257
258
259          Private Sub PlayButton_Click(sender As Object, e As EventArgs) Handles  PlayButton.Click
               If commandFile Is Nothing Then
                    MessageBox.Show("Debe seleccionar un archivo.", "My 3 axis Milling  Robot",
                                    MessageBoxButtons.OK,  MessageBoxIcon.Error)
260
261             Else
                    If Not on_proccess Then
                         If commandFile.parseDoc() = "error" Then send("EXIT")
                              _Connection.Close()
262                           _Connection = Nothing ConectarToolStripMenuItem.Text =
263                           "Conectar"  fileReaderOUT = Nothing     commandFile.close()
264                           commandFile = Nothing
265                           MessageBox.Show("Las coordenadas están fuera de rango.", "My
266                            3 axis Milling Robot",
267                                     MessageBoxButtons.OK,  MessageBoxIcon.Error)
268                      Else
269                           fileReaderOUT  =  My.Computer.FileSystem.OpenTextFileReader  (Label4.Text)
270                           MessageBox.Show("Se ha procesado el archivo exitosamente.",  "My 3 axis Milling
271                            Robot",
272                                     MessageBoxButtons.OK,  MessageBoxIcon.Information)  on_proccess() = True
273                           enable_Send = True Button2.Enabled = True
                              Button1.Enabled = True send("SPEED," &
                              c_speed.ToString)
274                      End If
275                      PlayButton.Enabled = False Else
276                      enable_Send = True sendCommand()
                         Button2.Enabled = True
                    End If  End
               If
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
```

```
292         End Sub
293
294
            Private Sub Button1_Click(sender As Object, e As EventArgs) Handles  Button1.Click
                  on_proccess     =     False
295               enable_Send     =     False
296               send("EXIT")
297               _Connection.Close()
298               _Connection = Nothing ConectarToolStripMenuItem.Text  =  "Conectar"
299               fileReaderOUT.Close()
300               fileReaderOUT = Nothing
301               commandFile.close() commandFile =
302               Nothing PlayButton.Enabled = False
303               Button1.Enabled = False Button2.Enabled
304               = False
305         End Sub
306
307
308
            Private Sub TextBox2_TextChanged(sender As Object, e As EventArgs) Handles  TextBox2.LostFocus
309               If Val(TextBox2.Text) > 15 Then  TextBox2.Text =
                        "5"
                        MessageBox.Show("La velocidad de corte no puede ser mayor a 15 mm/  s .", "My 3 axis
310                        Milling Robot",
311                                       MessageBoxButtons.OK,  MessageBoxIcon.Error)  ElseIf
312               Val(TextBox2.Text) <= 0 Then
313                     TextBox2.Text = "5"
                        MessageBox.Show("La velocidad de corte no puede ser menor o igual a 0  mm/s .", "My 3 axis Milling
314                         Robot",
315                                       MessageBoxButtons.OK,  MessageBoxIcon.Error)
316               Else
317                     c_speed  =  Val(TextBox2.Text)  End If
            End Sub
318
319         Private Sub Button2_Click(sender As Object, e As EventArgs) Handles  Button2.Click
320               enable_Send = False  PlayButton.Enabled = True
321               Button2.Enabled = False
322       End Sub  End
323   Class
324
            Public  Class  ConnectionInfo
325               Private _xUpdateMethod As Action(Of String)
326               Public ReadOnly Property xUpdateMethod As Action(Of String)  Get
327                         Return  _xUpdateMethod  End Get
328               End Property
329
330
331
332
333
334
335
336
337
338
```

```
339         Private _Client As TcpClient
340          Public ReadOnly Property Client As TcpClient Get
341                     Return _Client End Get
342         End Property
343
344
345
346         Private _Stream As NetworkStream
347          Public ReadOnly Property Stream As NetworkStream Get
348                     Return _Stream End Get
349         End Property
350
351
352
353         Private _LastReadLength As Integer
354          Public ReadOnly Property LastReadLength As Integer Get
355                     Return _LastReadLength End Get
356         End Property
357
358
359
360         Private _sWriter As StreamWriter
361          Public ReadOnly Property sWriter As StreamWriter Get
362                     Return _sWriter End
363               Get
364         End Property
365
366
367         Private _Buffer(63) As Byte Public Property
368
369
370         Buffer As Byte()
371               Get
372                     Return _Buffer End Get
373               Set(value As Byte())
374                     _Buffer = value End
375               Set
376         End Property
377
378


             Public Sub New(address As String, port As Integer, xupdate As Action(Of  String))
379               _xUpdateMethod = xupdate
380               _Client = New TcpClient
381               _Client.Connect(address,  port)
382               _Stream = _Client.GetStream
383               _sWriter = New StreamWriter(_Stream)
384              _sWriter.NewLine() = "" End Sub
385
386
387         Public Sub AwaitData()
388              _Stream.BeginRead(_Buffer, 0, _Buffer.Length, AddressOf DoReadData, Me) End Sub
389
```

```
390
391
392
393         Public Sub Close()
394               If _Client IsNot Nothing Then _Client.Close()
395               _xUpdateMethod = Nothing
396               _Client = Nothing
397               _Stream = Nothing End Sub
398
399
400
401         Private Sub DoReadData(result As IAsyncResult)
402               Dim info As ConnectionInfo = CType(result.AsyncState, ConnectionInfo) Try
403                     If info._Stream IsNot Nothing AndAlso info._Stream.CanRead Then info._LastReadLength =
404                           info._Stream.EndRead(result)
                            If info._LastReadLength > 0 Then
                                  Dim message As String = System.Text.Encoding.ASCII.GetString (info._Buffer)
                                  info._xUpdateMethod(message)
405                         End If info.AwaitData()
406                   End If
407               Catch ex As Exception info._LastReadLength = -1
408               End Try End
409         Sub
410   End Class
411
412
413
414   Public Class NCparse
415         Private fileReaderIN As StreamReader Private
416         fileWriterOUT As StreamWriter Private lastG As
417         String = "0"    Private newGcode As String = "0"
418         Private X As String = "0"
419         Private Y As String = "0" Private Z As
420         String = "10" Private R As String = "0"
421         Private lastX As String = "0" Private
422         lastY As String = "0" Private lastZ As
423         String = "0" Private line As String
424         Private LastLine As String Private
425         Xbase As Double Private Ybase As Double
426         Private Zbase As Double
427         Private Xmin As Double = -550 ' Private
428         Xmax As Double = -250 ' Private Ymin As
429         Double = -120 ' Private Ymax As Double =
430         100 ' Private Zmin As Double = 170 '
431         Private Zmax As Double = 275 '
432
433
434
435
436
437
438         Public Sub close()
439
440
```

```
441                 fileReaderIN.Close()
442                 fileWriterOUT.Close() lastG =
443                 "0"
444                 newGcode = "0"
445                 X = "0"
446                 Y = "0"
447                 Z = "0"
448                 R = "0"
449                 lastX = "0"
450                 lastY = "0"
451               lastZ = "0" End
452         Sub
453
454
455
456          Public ReadOnly Property getGLine() As String Get
457                     If String.IsNullOrEmpty(line) Then  Return
458                         "END"
459                     Else
460                         Return line End
461                     If
462               End Get End
463         Property
464          Private Sub writeLineOut(line As String)
465               fileWriterOUT.WriteLine(line)
466         End Sub
467
468
469
470          Public Sub New(fileName As String, filename2 As String, coord_b() As String)  Xbase = Val(coord_b(0))
471               Ybase = Val(coord_b(1)) Zbase =
472               Val(coord_b(2))
                  fileReaderIN  =  My.Computer.FileSystem.OpenTextFileReader(fileName) fileWriterOUT  =
                  My.Computer.FileSystem.OpenTextFileWriter(filename2,                                    ↵
473                  False)
474               Do
475                     line = fileReaderIN.ReadLine() Loop Until
476               (getGcode(line))
477         End Sub
478          Private Function WrittingCommandLine() As String Dim ASline As
479               String = ""
480               Dim end_of_doc As Boolean = False
                  Dim nfi As NumberFormatInfo = New CultureInfo("en-US",                                  ↵
                     False).NumberFormat nfi.NumberDecimalSeparator = "."
481
482
483
484               If Not String.IsNullOrEmpty(line) Then  LastLine = line
485                     lastX = X lastY =
486                     Y lastZ = Z
487                     findCoord("X")
488                     findCoord("Y")
489                     findCoord("Z")
490
```

```
491                     findCoord("R")
492                     getGcode(line)
493               Else
494                     fileWriterOUT.Close()
495                     fileReaderIN.Close() Return "fin"
496               End If
497
498
499
500               line = fileReaderIN.ReadLine() If lastG =
501               "G1" Then
502                     If IsOnRange(Val(X), Val(Y), Val(Z)) Then
503                           writeLineOut(LastLine & "'" & "LMOVE," & X & "," & Y & "," & Z) Else
504                           Return "error" End If
505               ElseIf lastG = "G0" Then
506                     If IsOnRange(Val(X), Val(Y), Val(Z)) Then
507                           writeLineOut(LastLine & "'" & "JMOVE," & X & "," & Y & "," & Z) Else
508                           Return "error" End If
509               ElseIf lastG = "G2" Then
510                     If G_2_3(Val(X), Val(Y), Val(Z), Val(lastX), Val(lastY), Val(lastZ), Val(R), True) = "error"
511                        Then
512                           Return "error"
513                     End If
                        writeLineOut("'" & "LMOVE," & X & "," & Y & "," & Z) ElseIf lastG = "G3" Then
                        If G_2_3(Val(X), Val(Y), Val(Z), Val(lastX), Val(lastY), Val(lastZ), Val(R), False) = "error"
514                        Then
515                           Return "error"
516                     End If
517                     writeLineOut("'" & "LMOVE," & X & "," & Y & "," & Z) End If
518               Return "continua" End
            Function
             Private Function getGcode(a As String) As Boolean Dim i As Integer
519               newGcode = "0"
520               For i = 0 To a.Length - 1
521                     If i + 1 < a.Length - 1 And i + 2 < a.Length - 1 Then
522
523
524
525                           If a.Chars(i) = "G" Then
526                                 If a.Chars(i + 1) = "1" Then
527                                       If a.Chars(i + 2) = " " Then newGcode =
528                                             "G1"
529                                       End If
530                                 ElseIf a.Chars(i + 1) = "0" Then If a.Chars(i
531                                       + 2) = "0" Then
532                                             newGcode = "G1"
533                                       ElseIf a.Chars(i + 2) = "1" Then newGcode = "G1"
534
535
536
537
538
539
540
```

```
541                                     ElseIf a.Chars(i + 2) = "2" Then newGcode = "G3"
542                                     ElseIf a.Chars(i + 2) = " " Or a.Chars(i + 2) = "X" Or a.Chars(i + 2) = "Y"
543                                Or a.Chars(i + 2) = "Z" Then
                                        newGcode = "G1" End
                                    If
544                                ElseIf  a.Chars(i  +  1)  =  "2"  Then  If
545                                    a.Chars(i + 2) = "0" Then Else
546                                        newGcode = "G0" End
547                                    If
548                                ElseIf  a.Chars(i  +  1)  =  "3"  Then  If
549                                    a.Chars(i + 2) = "0" Then Else
550                                        newGcode = "G3" End
551                                    If
552                                End If End
553                            If
554                        End If Next
555                If lastG = newGcode Then If lastG
556                    = "0" Then
557                        Return False Else
558                        Return True End
559                    If
560                Else
561                    If newGcode = "0" Then Return
562                        False
563                    Else
564                        lastG = newGcode Return True
565                    End If End
566                If
567        End Function
568         Private Sub findCoord(coord As String) Dim index As
569                Integer = 1
570                Dim newCoord As String = "" If
571                line.Contains(coord) Then
572                    While exOutOfRange(index, coord)
573                        newCoord = newCoord & line.Chars(line.IndexOf(coord) + index) index = index + 1
574                    End While
575                    Select Case coord Case "X"
576                                X = newCoord Case
577                        "Y"
578                                Y = newCoord Case
579                        "Z"
580                                Z = newCoord Case
581                        "R"
582                                R = newCoord
583
584
585
586
587
588
589
590
591
```

```
592                     End Select End If
593         End Sub
594          Private Function exOutOfRange(index As String, coord As String) As Boolean  Dim result As Boolean
595               Dim a As String
596               If line.IndexOf(coord) + index <= line.Length - 1 Then  a =
597                     line.Chars(line.IndexOf(coord) + index)    result = validChar(a)
598                     Return result Else
599                     Return False End
600               If
601         End Function
602          Private Function IsOnRange(X As Double, Y As Double, Z As Double) As Boolean  Return True
603               If (X + Ybase) >= -120 And (X + Ybase) <= 100 And (-Y + Xbase) >= -425  And (-Y + Xbase) <= -250
604                  And (Z + Zbase) >= 170 And (Z + Zbase) <= 275  Then
605                     Return True Else
606                     Return False End
607               If
608         End Function
             Private Function validChar(a As String) As Boolean Select Case a
                        Case "-", ".", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"
                              Return True Case
                        Else
609                           Return False End
610               Select
611         End Function
612
613
614
615          Public Function parseDoc() As String Dim aux As
616               String = "continua"  While aux = "continua"
617                     aux = WrittingCommandLine() End While
618               Return aux End
619         Function
620          Private Function G_2_3(x As Double, y As Double, z As Double, x_tempi As  Double, y_tempi As Double,
621             z_tempi As Double, radio As Double, G2 As  Boolean) As String
622
623
624
625               Dim  x_tempf  As  Double  =  0  Dim
626               y_tempf  As  Double  =  0  Dim  z_tempf
627               As Double = 0
628
629
630
                  Dim  x_centro  As  Double  =  0  Dim
                  y_centro As Double = 0 Dim z_centro
                  As Double = 0


631
632
633
634
635
636
637
638
639
```

```
640             Dim  x_tempi_base  As  Double  =  0  Dim
641             y_tempi_base   As   Double   =   0   Dim
642             x_tempf_base   As   Double   =   0   Dim
643             y_tempf_base As Double = 0
644
645
646
647             Dim p_cruz As Double = 0
648
649
650
651             Dim  alpha_i  As  Double  =  0  Dim
652             alpha_f  As  Double  =  0  Dim  alpha_m
653             As Double = 0
654
655
656
657
658
                Dim paso As Double = 1
659
660
661             x_tempf = x + 0.001  y_tempf
662             = y + 0.001  z_tempf = z
663
664
665
666              punto_centro(x_tempi, y_tempi, x_tempf, y_tempf, radio, True, x_centro, y_centro)
667
                x_tempi_base = x_tempi - x_centro  y_tempi_base
668             = y_tempi - y_centro  x_tempf_base = x_tempf -
669             x_centro  y_tempf_base = y_tempf - y_centro
670
671
672
673             p_cruz = (x_tempi_base * y_tempf_base) - (x_tempf_base * y_tempi_base)
674              If (G2 And p_cruz >= 0) Or ((Not G2) And p_cruz <= 0) Then punto_centro(x_tempi, y_tempi, x_tempf,
675                   y_tempf, radio, False,
676                      x_centro, y_centro) x_tempi_base = x_tempi
677                   - x_centro  y_tempi_base = y_tempi - y_centro
678                   x_tempf_base = x_tempf - x_centro
679                   y_tempf_base = y_tempf - y_centro
680
681
682             End If
683             alpha_i = arcotan(x_tempi_base, y_tempi_base) alpha_f =
684             arcotan(x_tempf_base, y_tempf_base)
685
686
                 If cond(radio, alpha_f, alpha_i) Then If alpha_f
                       >= alpha_i Then
                             Return G_2_3_Writer(alpha_i, alpha_f, paso, True, x_centro, y_centro, z_tempi, z_tempf,
                                radio)
                       Else
                             Return G_2_3_Writer(alpha_i, alpha_f, paso, False, x_centro, y_centro, z_tempi, z_tempf,
                                radio)
                     End If Else
                       If alpha_f >= alpha_i Then
                             If G_2_3_Writer(alpha_i, 0, paso, False, x_centro, y_centro, z_tempi, z_tempf, radio) =
                                "continua" And
                             G_2_3_Writer(360, alpha_f, paso, False, x_centro, y_centro,
```

```
                        z_tempi, z_tempf, radio) = "continua" Then Return
                           "continua"
687                     Else
688                           Return "error" End If
689               Else
690                     If G_2_3_Writer(alpha_i, 360, paso, True, x_centro, y_centro, z_tempi, z_tempf, radio) =
691                        "continua" And
692                     G_2_3_Writer(0, alpha_f, paso, True, x_centro, y_centro, z_tempi, z_tempf, radio) =
                           "continua" Then
                           Return "continua" Else
                           Return "error" End If
693               End If End
            If


694
695
696
697
698
699
700
701
702
703
704     End Function
705      Private Function cond(ByRef radio As Double, ByRef alpha_f As Double, ByRef  alpha_i As Double) As
           Boolean
              If radio > 0 Then
                    If Abs(alpha_f - alpha_i) < 180 Then Return True
706                 Else
707                       Return False End
708                 If
709           Else
710                 If Abs(alpha_f - alpha_i) > 180 Then Return True
711                 Else
712                       Return False End
713                 If
714          End If End
715     Function
716      Private Function arcotan(ByVal a As Double, ByVal b As Double) As Double  If a > 0 Then
717                 If b >= 0 Then
718                       Return (Atan((b / a)) * 180) / PI Else
719                       Return ((Atan((b / a)) * 180) / PI) + 360 End If
720           ElseIf a < 0 Then
721                 Return ((Atan((b / a)) * 180) / PI) + 180 ElseIf a = 0
722           Then
723                 If b >= 0 Then Return
724                       270
725                 Else
726                       Return ((Atan((b / a)) * 180) / PI) + 360 End If
727
728
729
730
731
732
733
734
```

```
735             End If End
736         Function
737          Private Sub punto_centro(ByVal a_tempi As Double, ByVal b_tempi As Double,  ByVal a_tempf As Double,
                ByVal b_tempf As Double, ByVal r As Double, ByRef  op As Boolean, ByRef a_centro As Double, ByRef
                b_centro As Double)
                  Dim m2 As Double = 0 Dim a_m
                  As Double = 0 Dim b_m As
                  Double = 0
738               Dim segmento_c As Double = 0 Dim
739               segmento_d As Double = 0
740
741
742
743               a_m = ((a_tempi + a_tempf) / 2)  b_m =
744               ((b_tempi + b_tempf) / 2)
745
746
747
                  segmento_d = Sqrt((((a_m - a_tempi) * (a_m - a_tempi)) + ((b_m - b_tempi)
                      * (b_m - b_tempi))))
748               segmento_c = Sqrt((r * r) - (segmento_d * segmento_d))  m2 = -1 * (a_m -
749
750               a_tempi) / (b_m - b_tempi)
751
752               If op Then
753                    a_centro = a_m + Sqrt((segmento_c * segmento_c) / (1 + (m2 * m2)))
754               Else
755                    a_centro = a_m - Sqrt((segmento_c * segmento_c) / (1 + (m2 * m2)))
756               End If
757              b_centro = m2 * (a_centro - a_m) + b_m End Sub
758
759
760
          Private Function G_2_3_Writer(ByVal begin As Double, ByVal to_ As Double,  ByVal paso As Double, ByVal
              less_than As Boolean, ByVal x_centro As Double,
                                                ByVal y_centro As Double, ByVal z_tempi As Double, ByVal z_tempf
761                           As Double, ByVal R As Double) As String
                  Dim x_interpol As Double = 0 Dim
                  y_interpol As Double = 0
                  Dim z_interpol As Double = z_tempi
762               Dim nfi As NumberFormatInfo = New CultureInfo("en-US",
763                  False).NumberFormat
764               nfi.NumberDecimalSeparator = "." If less_than
765               Then
                       For alpha_m As Double = begin + paso To to_ Step paso  x_interpol = x_centro + (R *
                            Cos((alpha_m * PI) / 180))  y_interpol = y_centro + (R * Sin((alpha_m * PI) /
766                         180))  If IsOnRange(x_interpol, y_interpol, z_interpol) Then
767                           If alpha_m = begin + paso Then  writeLineOut(LastLine & "'" &
768                                "LMOVE," &
769                            x_interpol.ToString("N3", nfi) & "," & y_interpol.ToString  ("N3", nfi) & "," &
770                            z_interpol.ToString("N3", nfi))
771                           Else
772                                writeLineOut("'" & "LMOVE," & x_interpol.ToString("N3",  nfi) & "," &
773                            y_interpol.ToString("N3", nfi) & "," &  z_interpol.ToString("N3",  nfi))
                              End If
774
775
776
```

```
777                         Else
778                               Return "error" End If
779
780
781
782                   Next
783                   Return "continua" Else
784                   For alpha_m As Double = begin - paso To to_ Step -paso x_interpol = x_centro + (R
785                         * Cos((alpha_m * PI) / 180)) y_interpol = y_centro + (R * Sin((alpha_m * PI)
786                         / 180)) If IsOnRange(x_interpol, y_interpol, z_interpol) Then
787                               If alpha_m = begin - paso Then writeLineOut(LastLine & "'" &
788                                     "LMOVE," &
789                                x_interpol.ToString("N3", nfi) & "," & y_interpol.ToString ("N3", nfi) & "," &
                                 z_interpol.ToString("N3", nfi))
                                Else
                                     writeLineOut("'" & "LMOVE," & x_interpol.ToString("N3", nfi) & "," &
                                 y_interpol.ToString("N3", nfi) & "," & z_interpol.ToString("N3", nfi))
                                End If
790                         Else
791                               Return "error" End If
                      Next
                      Return "continua" End If
            End Function End

      Class
792
793
794
795
796
797
798
799
800
801
802
```

```
 1
 2  Imports System.Windows ' Window, RoutedEventArgs, IInputElement, DependencyObject
 3  'Imports System.Windows.Controls ' Validation
 4  Imports System.Windows.Input ' Keyboard
 5  Imports System.Net
 6  Imports System.IO
 7  Imports  System.Net.Sockets
 8  Imports  WindowsApplication2.Form1
 9  Public Class Form2
10      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
          Button1.Click
11          Form1.mIpadd() = TextBox1.Text
12          Form1.mPort() = TextBox2.Text
13          Close()
14      End Sub
15
16      Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
          Button2.Click
17          Close()
18      End Sub
19
20      Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
21          TextBox2.Text = Form1.mPort()
22          TextBox1.Text = Form1.mIpadd()
23      End Sub
24
25  End Class
```

Código del robot en lenguaje AS

```
.PROGRAM FRESADORA() #87
;********************BINDING_SOCKET************************************
  port = 55000
  max_length = 255
  tout_open = 60
  tout_rec = 60
  CALL open_socket
  IF sock_id<0 THEN
    GOTO exit_end
  END
;********************INITIALIZING_VARs*********************************
  text_id = 0
  tout = 60
  eret = 0
  rret = 0
  $sdata[1] = ""
  $comm = ""
  $temp = ""
  $pose[1] = "000"
  x = 0
  y = 0
  z = 0
  offset = 0
  index = 0
  path.length = 0
  CP ON
;********************MOVING_AWAY_FROM_PART_ZERO****************************
  BASE NULL
  HERE var
  BASE var
;********************MOVING_AWAY_FROM_PART_ZERO*******************************
  LDEPART 50
;********************WAITING_FOR_INICIO****************************************
  $sdata[1] = "Esperando inicio"
  CALL send(eret,$sdata[1])
  IF eret<0 GOTO error_al_enviar
espera_inicio:
  CALL recv
  IF rret<0 GOTO error_al_recibi
  IF ($recv_buf[1]=="inicio\r") OR ($recv_buf[1]=="inicio") THEN
```

```
;LDEPART -25
    SPEED 1 ALWAYS
    GOTO inicio
  END
  GOTO espera_inicio
inicio:
  $pose[1] = "000"
  $sdata[1] = "done"
  CALL send(eret,$sdata[1])
  IF eret<0 GOTO error_al_enviar
;********************WAITING_FOR_COMMAND*************************************
recv_cmd:
  CALL recv
  IF rret<0 GOTO error_al_recibi
  $comm = $DECODE($recv_buf[1],",",0) ;EXTRACTS THE COMAND FROM THE RECIEVED S
;********************EXIT_COMMAND*************************************
  IF ($comm=="EXIT\r") OR ($comm=="EXIT") GOTO exit
;********************JMOVE_COMMAND*************************************
  IF ($comm=="JMOVE\r") OR ($comm=="JMOVE") THEN
    $temp = $DECODE($recv_buf[1],",",1)
    CALL xyz
    POINT move_pose = TRANS(xx,-1*yy,-1*zz,0,0,0)
    SPEED 10 MM/S ALWAYS
    LMOVE move_pose
    $sdata[1] = "done"
    CALL send(eret,$sdata[1])
    IF eret<0 GOTO error_al_enviar
    GOTO recv_cmd
  END
;********************LMOVE_COMMAND*************************************
  IF ($comm=="LMOVE\r") OR ($comm=="LMOVE") THEN
    $temp = $DECODE($recv_buf[1],",",1)
    CALL xyz
    POINT move_pose = TRANS(xx,-1*yy,-1*zz,0,0,0)
    SPEED 15 MM/S ALWAYS
    LMOVE move_pose
    $sdata[1] = "done"
    CALL send(eret,$sdata[1])
    IF eret<0 GOTO error_al_enviar
    GOTO recv_cmd
  END
  END
  GOTO recv_cmd
```

```
error_al_enviar:
  PRINT "Error al enviar. Error no.:",eret
  GOTO exit
error_al_recibi:
  PRINT "Error al recibir comando de Inicio. Error no.:",eret
exit:
  CALL close_socket
exit_end:
  BASE NULL
.END
```