UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DE GRADO

Simulador de un prototipo de Control Numérico por Computador

Autor:
Jordi Burriel Valencia

Director:
Rubén Puche Panadero

11 de Julio de 2020

TRABAJO FINAL DE GRADO

Memoria

Planos

Pliego de condiciones

Presupuesto

Anexos

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MEMORIA

Simulador de un prototipo de Control Numérico por Computador

Autor:
Jordi Burriel Valencia

Director:
Rubén Puche Panadero

10 de Julio de 2020

TRABAJO FINAL DE GRADO

ÍNDICE

1.- Objeto del trabajo.

Dado las ventajas que produce industrialmente usar sistemas programables de mecanizado de piezas, es importante la enseñanza del lenguaje de Control Numérico por Computador (CNC) usado en la mayoría de dichos sistemas. Esto es válido tanto en titulaciones de formación profesional como en grados universitarios relacionados con la ingeniería.

No obstante, el alto coste de los equipos de mecanizado limita la capacidad de los centros de enseñanza de disponer de maquinaria de mecanizado para todos los alumnos. Como alternativa, es posible recrear de forma virtual la funcionalidad de dichas máquinas mediante software en un ordenador de propósito general, abaratando notablemente los costes.

Por ello, el principal objetivo de este proyecto es desarrollar un software de simulación específica de una máquina de control numérico por computador, para proyectos de mecanizado bidimensional.

Como este simulador está orientado para ser usado en el ámbito de la enseñanza. Tanto los elementos de la interfaz como la funcionalidad estarán diseñados para un uso didáctico más que para un uso profesional. En consecuencia, las características básicas de este constarán de:

Una Interfaz sencilla donde los controles estén más orientados a facilitar al alumno la comprensión de la programación por CNC que al ámbito de producción industrial.

Un área virtual de trabajo donde el simulador mostrará el resultado de los programas de mecanizado desarrollados por los alumnos.

Un editor del código del programa para desarrollar, modificar y depurar los programas de mecanizado.

Varios modos de funcionamiento según las necesidades del usuario. Un modo de edición del programa de mecanizado que active los controles de edición. Un modo de simulación destinado a que el usuario pueda observar y depurar el mecanizado de su programa de mecanizado.

Este modo de simulación a su vez tendrá dos modos. Se dispondrá de simulación en tiempo real para que el usuario pueda analizar la evolución de su programa de mecanizado a lo largo del tiempo. Y por otro lado se dispondrá de simulación por línea de programa controlado por el propio usuario para que pueda desarrollar labores de depuración.

Las propiedades paramétricas del simulador serán personalizables para que el usuario pueda adaptarlos a las especificaciones del hardware requerido.

Verificación de la correctitud e integridad del código de los programas de mecanizado. Necesario para que los usuarios puedan detectar y corregir fallos correspondientes a la nomenclatura del lenguaje CNC o límites del hardware de la máquina simulada.

Capacidad de almacenamiento y apertura de proyectos existentes para que los usuarios puedan mantener la persistencia de sus programas de mecanizado.

2.- Antecedentes.

Como ya se ha comentado en el punto anterior, los sistemas programables de mecanizado de piezas juegan un papel muy importante en la industria. Existe una gran variedad de máquinas de este tipo orientado a múltiples campos de aplicación, como por ejemplo la metalurgia o la carpintería.

Conforme avanza la eficiencia y se reducen costes, el uso de estos sistemas CNC se expande desde la gran industria a otros tipos de industria mediana e incluso a pequeños talleres.

La automatización de los procesos de elaboración de piezas ha hecho aumentar enormemente la productividad y la perfección en la fabricación de piezas. En contraposición al fresado manual, con el mecanizado automático programado se pueden desarrollar piezas cuyas formas eran difíciles de recrear, con precisiones milimétricas imposibles de alcanzar con destreza humana o poder usar materiales que por sus características intrínsecas no se podían modelar.

Ilustración 1. Fresado de alta precisión de una pieza.

El tipo de programación de las máquinas de mecanizado es dependiente del fabricante y del modelo desarrollado. No obstante, hay una clara tendencia usar un lenguaje de programación concreto denominado G-code o RS-274. Este lenguaje G-code es implementado por multitud de fabricantes de maquinaria industrial, como por ejemplo Siemens, Mazak, Sinumeric o FANUC.

A pesar de que existen varios estándares (ISO6983, DIN66025, PN-73M-55256, PN-93/M-55251, BGL, etc.), algunos fabricantes añaden ciertas particularidades que limitan la compatibilidad entre máquinas.

Aunque normalmente se entiende como sistema CNC una máquina con una multiherramienta de fresado, existe multitud de máquinas distintas de mecanizado CNC que usan G-code. Por ejemplo tornos, rectificadoras, de corte por láser, de corte por chorro de agua, electroerosión, estampación, etc.

Incluso dentro del mismo ámbito de máquinas de fabricación de piezas que usan G-code, se encuentran las impresoras 3D por extrusión. En este caso singular, las piezas no se crean a partir del mecanizado y retirada del material, sino que se construye mediante la deposición de capas de material termofusible.

Abarcando a un campo de uso incluso más amplio de la programación por G-code, se encuentran máquinas para otros usos distintos a la fabricación de piezas, como por ejemplo brazos robóticos e impresoras de impresión lineal (plotter).

Por todo esto expuesto anteriormente, queda clara la gran importancia de estos sistemas en la industria y la necesidad de que los futuros ingenieros sean capaces de desarrollar competencias propias para la programación de estos sistemas.

El desarrollo de programas para estos sistemas CNC se realiza principalmente con entornos de desarrollo que suelen incluir un sistema previo de simulación con el que probar el programa de mecanizado antes de ejecutarlo sobre la máquina de verdad.

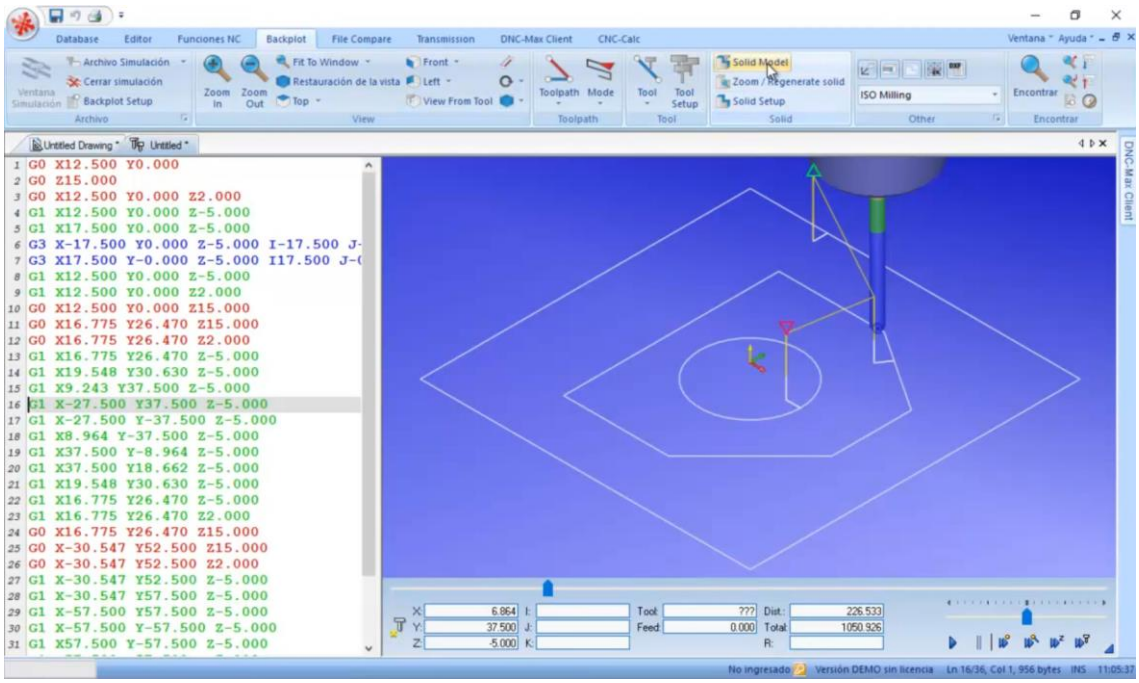


Ilustración 2. Cimco Edit 2015 Simulación de CNC

Gracias a este entorno de simulación previo, se permite a los desarrolladores realizar multitud de pruebas y obtener resultados de una forma rápida sin la necesidad de malgastar tiempo y material hasta llegar a obtener el resultado deseado. Además, de todo esto, el simulador aporta otras ventajas que no posee la máquina real, como la ejecución de la misma simulación en múltiples ordenadores en paralelo, la simulación

inversa de “desmecanizado”, el posicionamiento concreto en cualquiera de los estados de la máquina y la corrección del programa al vuelo durante la simulación.

2.1.- Sistema CNC de mecanizado.

La maquinara disponible para realizar pruebas de programación CNC con G-code, se corresponde con la que se muestra en la siguiente ilustración.

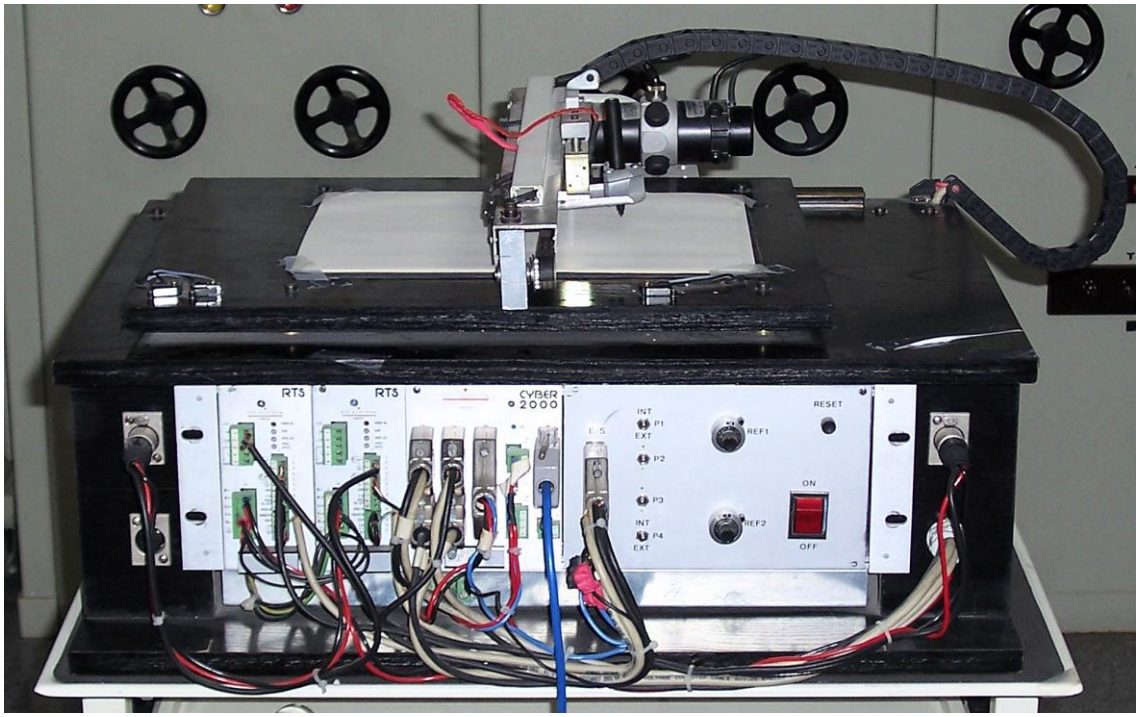


Ilustración 3. Máquina de mecanizado de tipo plotter

Como se observa, se corresponde con un sistema de tipo plotter donde la herramienta usada es un marcador que dibuja sobre la mesa de trabajo. Esta máquina es muy similar a los sistemas CNC que se pueden encontrar para el serigrafiado o cortado de piezas (sistemas de cuchilla, por láser, por chorro de arena, etc.). El sistema de coordenadas es 2.5D, teniendo en cuenta que los dos movimientos principales se suceden en el eje X e Y, usándose el eje Z únicamente como eje de dos posiciones (Herramienta en elevación, herramienta en posición de marcado).

En este caso, surge la necesidad de un sistema simulado similar y compatible con el sistema CNC mostrado en la Ilustración 3. Es decir, las características mínimas del simulador deben simular los aspectos físicos de la máquina, como los límites de mecanizado de la herramienta sobre la mesa de trabajo, o el límite de velocidad de movimiento del cabezal herramienta. También, debe ser capaz de ejecutar de interpretar la misma versión del lenguaje G-code usado en la máquina real. Y por supuesto, el resultado simulado de ejecutar el programa tiene que ser el mismo que el que se obtiene con la máquina de verdad.

Aprovechando las peculiaridades de usar un simulador, también se pueden ampliar funcionalidades del propio sistema de simulación, facilitando las herramientas necesarias para que el usuario pueda realizar el desarrollo de los programas CNC desde el propio simulador, validación de errores del programa y posibilidad de depuración continua y paso a paso.

3.- Justificación.

El desarrollo de este trabajo es justificable desde diversos ámbitos distintos, donde todos ellos aportan motivaciones relevantes para el desarrollo del trabajo en cuestión.

3.1.- Motivación académica.

Una de las motivaciones importantes de este trabajo final de grado (TFG), recae en la justificación de que el autor muestra que ha adquirido los conocimientos, habilidades y competencias para desarrollar un trabajo técnico dentro del campo de la ingeniería eléctrica; con un nivel correspondiente al de un Ingeniero Eléctrico.

Por ello, según la regulación vigente de la Universitat Politècnica de València (UPV) del 7 de marzo del 2013 (y sus siguientes modificaciones); con el desarrollo y defensa satisfactoria del trabajo se habilita al autor a obtener el título de Graduado en Ingeniería Eléctrica. Esta titulación está adscrita a la Escuela Técnica Superior de Ingeniería del Diseño.

El trabajo se desarrolla con la tutorización del profesor D. Rubén Puche Panadero, en el Departamento de Ingeniería Eléctrica de esta misma universidad.

3.2.- Motivación funcional.

Además de las motivaciones académicas descritas, existen varias motivaciones funcionales que imperan al autor a profundizar en los conocimientos necesarios para el desarrollo del trabajo.

La motivación funcional que afecta directamente al autor recae en los siguientes aspectos:

- Ampliación de los conocimientos en el ámbito de los sistemas de Control Numérico por Computador, los diversos sistemas funcionales, la representación CAD/CAM, los diversos lenguajes de producción y diversidad de soluciones dentro la industria.

- Ampliación de los conocimientos de simuladores físicos, análisis desde diversas perspectivas (mecánica, eléctrica, etc.).

- Ampliación de los conocimientos en desarrollo de aplicaciones industriales y educativas, normativas, requisitos, hardware.

- Ampliación de los conocimientos sobre el fundamento físico del mecanizado de piezas, la diversidad de materiales, producción de piezas.

- Ampliación de los conocimientos de electrónica, sistemas de alimentación, sensores y actuadores, motores de posición, variadores de frecuencia, etc.

- Ampliación de los conocimientos de herramientas informáticas de desarrollo de software, plataformas, entornos, lenguajes de programación, metodologías de diseño.

Otra motivación funcional recae en los beneficios a futuros usuarios del simulador, que pueden ser tanto docentes, alumnos, o profesionales que del sector. Esta herramienta responde a las necesidades prácticas de simulación y desarrollo, para ampliación de conocimientos o como fin práctico, en programación de sistemas CNC y lenguaje G-code.

4.- Estudio de viabilidad.

El estudio de la viabilidad de cualquier trabajo consiste en realizar una valoración previa para determinar la posibilidad y la conveniencia de desarrollar el trabajo. En todo caso, es importante analizar los aspectos más relevantes para poder realizar una deducción correcta acerca de la viabilidad del trabajo.

Estos aspectos más relevantes son, el técnico, el académico y el económico, los cuales nos van a indicar los límites del desarrollo del trabajo.

4.1.- Viabilidad técnica.

Las herramientas que se requieren para el desarrollo de este trabajo son bastante genéricas (computadora de prestaciones medias) y con variedad de opciones posibles (abundante diversidad de software para el desarrollo de simuladores).

Por todo ello, desde el aspecto técnico no hay ninguna limitación que imposibilite o implique inconveniente alguno en el desarrollo del trabajo.

4.2.- Viabilidad económica.

Todas las herramientas que se requieren igualmente para el desarrollo tienen un coste asumible. En el caso del hardware necesario no se requiere de un equipo especial de simulación, sino que se puede usar cualquier computadora con unas prestaciones medias.

Respecto del coste del software necesario para el desarrollo de simuladores, existen una amplia variedad con diversas prestaciones, tanto gratuitas como de pago. Siendo las opciones de pago en su mayoría asumibles en términos de salario medio.

En consecuencia, desde el aspecto económico no hay ninguna limitación que imposibilite o implique inconvenientes sobre el desarrollo del trabajo.

4.3.- Viabilidad académica.

Dado el carácter de este trabajo como TFG, el aspecto más relevante a evaluar es su viabilidad académica.

Como ya se ha indicado previamente en el apartado 2, el conocimiento sobre sistemas CNC son de gran importancia en múltiples campos, por lo que el desarrollo de este trabajo aporta un gran valor académico.

Además, como también se ha indicado en el apartado 3, la elaboración del trabajo implica el análisis, uso y ampliación de múltiples conocimientos y capacidades adquiridos a lo largo del desarrollo del grado de Ingeniería Eléctrica.

Por tanto, desde este aspecto académico el trabajo cumple con las necesidades académicas requeridas para ser evaluado como un trabajo final de grado.

5.- Estudio de necesidades, limitaciones y condicionantes.

Previamente al análisis de las posibles soluciones a adoptar para desarrollar el trabajo, se ha de analizar previamente las necesidades, limitaciones y condicionantes que se van a tener.

Aunque el desarrollo del simulador se centra en el sistema CNC presentado en el apartado 2.1, las posibilidades de personalización de las propiedades físicas de simulación permiten que dicho simulador se pueda adaptar a cualquier otro sistema CNC que trabaje en un sistema de coordenadas 2.5D, englobando así directamente a bastantes sistemas CNC.

Estos sistemas por ejemplo pueden ser una máquina de corte por láser, de corte por chorro de agua, de corte por chorro de arena, de serigrafiado de materiales (madera, metales, etc.), de tratamiento de superficies, de diseño lineal, etc.

Igualmente, dicho sistema de simulación se podría adaptar para trabajar en un sistema de coordenadas puramente 3D ampliando las posibilidades de uso en otros sistemas CNC que trabajan con este sistema de coordenadas. No obstante, la complejidad del desarrollo y del uso posterior del simulador crecería notablemente. Dado el carácter académico de este trabajo, esta adaptación no es recomendable ya que sobrepasa los requisitos especificados, y no añade ningún valor docente que sea notable respecto a usar un sistema 2.5D.

Esto es así porque la única diferencia al usar 2.5D o 3D en el lenguaje G-code sería el uso de 2 o tres coordenadas en el espacio, siendo lo menos relevante para su aprendizaje.

5.1.- Especificaciones del encargo.

El resultado final del trabajo tiene que ser una aplicación que no requiera mayores prestaciones que un ordenador con características de gama media/baja similares a los que se pueden encontrar en aulas informáticas universitarias o de institutos de formación profesional.

En la fecha actual de desarrollo de este trabajo, las especificaciones de un ordenador de gama media/baja son las siguientes:

- Procesador Intel o AMD con fecha de lanzamiento posterior al 2012.
- 128 GB de disco duro.
- 2048 Mb de memoria RAM
- Tarjeta gráfica con resolución 1024x768 y soporte DirectX 9
- Windows 7, Windows 8 o Windows 10.

Las especificaciones que se requieren por parte de la aplicación de simulación se dividen en los siguientes conjuntos:

- **Requisitos de simulación**. La simulación debe tener un carácter conceptual del funcionamiento de la máquina. Es decir, debe simular los movimientos físicos del cabezal sobre una superficie de mecanizado con posiciones límites, interpretando las instrucciones en G-code; pero no debe simular aspectos concretos de los componentes de los que está construida la máquina física a la que se hace referencia (apartado 2.1), como por ejemplo el movimiento del motor que empuja el eje o la variación eléctrica que genera la activación/desactivación del sensor mecánico de fin de carrera.

  Los elementos físicos del simulador se deben componer de los siguientes:

  - *Superficie de trabajo*. Una superficie bidimensional donde se irá desarrollando el diseño mediante la actuación del cabezal herramienta sobre esta superficie.

  - *Cabezal herramienta*. Panel desde donde se podrán seguir las posiciones del cabezal y el estado en que se encuentra.

  Además, la simulación de la máquina se debe implementar de dos formas:

  - *Simulación física*. Simulación temporal donde se observa el cabezal herramienta actuando sobre la superficie bidimensional.

  - *Simulación dividida en pasos*. Estado de simulación representado por el estado actual del punto de control fijado por el usuario dentro del programa de mecanizado.

  Se requiere de controles y parámetros que actúen sobre la simulación para que el usuario pueda controlar esta. En este caso los parámetros son las dimensiones de la superficie de trabajo y la velocidad de posicionamiento y de elevación del cabezal herramienta. Los controles para controlar la simulación serán el de avance paso a paso, avance continuo, pausa y retroceso.

- **Requisitos de desarrollo del programa de mecanizado**. El software debe ser capaz de permitir la visualización y edición del programa de mecanizado desarrollado por el usuario.

- **Requisitos de soporte del lenguaje G-code (Estándar ISO 6983)**. El software debe comprender e interpretar los programas de mecanizado

respecto de un subconjunto de los comandos más usados del lenguaje G-code conforme al estándar ISO 6983.

Los resultados de interpretar dicho lenguaje en la simulación deben ser comparables a los que serían obtenidos mediante un sistema CNC físico.

Se debe poder validar la correctitud del programa de mecanizado, tanto en su codificación, en su funcionamiento o en su correcta secuenciación.

- **Requisitos de información adicional**. El software debe mostrar información del estado actual del sistema, del programa de mecanizado y del estado del cabezal herramienta.

  La información más relevante a recopilar es información de la ejecución de la simulación (iniciada, en pausa), de la advertencia de posicionamiento fuera de la superficie de mecanizado, posición actual del cabezal herramienta según la posición actual del programa de mecanizado y la distancia recorrida por el cabezal hasta la posición actual del programa de mecanizado.

Prosiguiendo con las especificaciones concretas del propio software (propósito, ámbito, funciones, características, restricciones, dependencias, etc.); toda esta información se ha realizado conforme al estándar ISO/IEC/IEEE 29148 en el documento de *Especificación de Requisitos que* se puede visualizar completamente en el adjunto del anexo 2. En consecuencia, no se va a incluir a continuación para que no haya redundancia de la misma información.

5.2.- Normativa y estandarización

En este trabajo no se aplica ninguna normativa específica respecto de la parte eléctrica o electrónica, dado que no existe (físicamente hablando) ninguna al ser una simulación. Por tanto, no existe ni se requiere tampoco reglamento necesario para la seguridad.

No obstante, sí que se han tenido en cuenta varios estándares y recomendaciones de buenas prácticas sobre este trabajo.

- **ISO 6983-1:2009.** Sistemas de automatización e integración. Control numérico de máquinas. Formato de programas y definiciones.
- **ISO/IEC/IEEE 29148:2011.** Ingeniería de requisitos.
- **ISO/IEC/IEEE 12207:2008.** Desarrollo del ciclo de vida.
- **ISO/IEC/IEEE 15288:2008.** Desarrollo del ciclo de vida.
- **IEEE Std. 1233.** Vocabulario de ingeniería del software.

5.3.- Estudio de necesidades propias

En esta sección se desarrollan las necesidades, limitaciones y condiciones que se han advertido durante el desarrollo del trabajo. En este caso se ha de intentar optimizar los requisitos del simulador.

Se debe de disponer de todo el equipamiento necesario para la programación del simulador, así como de las mejores condiciones del mismo en cuanto a: capacidad del disco duro, velocidad de ejecución, fiabilidad del programa y prestaciones.

En cuanto al desarrollo del simulador, no se ha encontrado ninguna limitación significativa que no se haya podido solucionar con ingenio.

En cuanto al funcionamiento del simulador, el mayor problema recae en su ejecución en un equipo con prestaciones menores a las mínimas requeridas, lo cual conllevará únicamente una velocidad de simulación en tiempo real deficiente.

No obstante, como anotación posterior después del desarrollo del software, se observa cómo se ha conseguido reducir los requisitos mínimos (Anexo 1) respecto de los que se definieron inicialmente en la sección 5.1.

Para un funcionamiento mínimo del simulador CNC sin uso de simulación real, se requiere de los siguientes requisitos mínimos:

- Procesador Intel o AMD con fecha de lanzamiento posterior al 2010.
- 50 MB de espacio libre en el disco duro.
- 1GB Mb de memoria RAM
- Tarjeta Gráfica con resolución 800x600 o superior y soporte DirectX 9
- Windows 7/8/10 o version Windows Server equivalente.

Para un funcionamiento mínimo del simulador CNC con uso de simulación real, se requiere de los siguientes requisitos mínimos:

- Procesador Intel o AMD con fecha de lanzamiento posterior al 2010.
- 50MB de espacio libre en el disco duro.
- 2GB de memoria RAM
- Tarjeta Gráfica con resolución 800x600 o superior y soporte DirectX 9
- Windows 7/8/10 o version Windows Server equivalente

6.- Planteamiento de soluciones alternativas y justificación de la solución adoptada.

Dada las características del trabajo correspondientes a un sistema virtual simulado, las alternativas que se pueden contemplar corresponden únicamente a las diversas herramientas de software que serán necesarias para desarrollar el simulador.

6.1.- Herramientas de diseño y modelado.

En todo trabajo, previamente al desarrollo del producto, se ha de analizar y diseñar el mismo con las características más adecuadas al menor coste. Al igual que ocurre en un producto físico, en el caso del software se tiene que desarrollar de igual forma.

En este caso existe una gran variedad de metodologías y herramientas de diseño específicas. Por tanto, para saber que herramienta será la más idónea es necesario antes seleccionar la metodología de programación más apropiada.

Existen varias metodologías de programación:

- Programación estructurada.
- Programación modular.
- Programación orientada a objetos.
- Programación concurrente.
- Programación funcional
- Programación lógica.

Considerando el tipo de aplicación que se demanda, la mejor forma de implementar un simulador es desarrollar funcionalmente un símil de los objetos físicos y sus interacciones dentro del software de simulación.

Para ello la mejor metodología de programación es la programación orientada a objetos (POO), ya que permite construir un programa en base a la definición de diversos objetos que interaccionan entre ellos.

Las herramientas de diseño que permiten el diseño de aplicaciones desarrolladas con POO, soportan el lenguaje unificado de modelado (UML).

En este caso se ha optado por usar la herramienta CASE MOSkitt con dicho soporte. No se han valorado alternativas para este caso ya que este tipo de herramientas está muy estandarizado y no hay diferencia apreciable entre una y otra herramienta.

Se ha optado directamente por el software MOSkitt, ya que es un software con soporte integral de modelado y diseño UML, es gratuito y está financiado por la Generalitat Valènciana.

Ilustración 4. Logotipo de la aplicación CASE MOSkitt

6.2.- Entorno de desarrollo y lenguaje de programación.

Actualmente existe una gran variedad de entornos de desarrollo (IDE) en los que se puede implementar soluciones de programación. Para este caso se ha valorado los siguientes elementos
:

- Simplicidad de la programación en ese IDE.
- Documentación y manuales existentes sobre el lenguaje.
- Tamaño de la comunidad de programadores que lo utilizan.
- Que sea soportado por el Sistema Operativo Microsoft Windows 7, Windows 8 o Windows 10, tal como se especifica en los requisitos.
- Número y utilidad para simulación de las librerías que le dan soporte.
- Facilidad para usar funciones gráficas y de control de tiempo. Necesario para el simulador el cual requiere desarrollar gráficamente y con simulación temporal.
- Las ventajas de depuración que ofrece el IDE que soporta el lenguaje.

Se han valorado varias alternativas de entornos de desarrollo, pero solamente pocos programas cumplen con la valoración de características demandada:

- **Eclipse IDE**. Este IDE está desarrollado por parte de la comunidad de software GPL. No obstante, aunque soporta varios lenguajes de programación, el lenguaje nativo usado es Java, el cual requiere de funcionar bajo la máquina virtual JVM, lo que lo hace más lento que otras alternativas,

- **NetBeans IDE.** Exactamente la misma problemática que en el caso del IDE Eclipse. Software GPL con soporte nativo de Java y por tanto ejecución bajo máquina Virtual JVM.

- **Dev-C++.** Buen entorno de desarrollo destinado al lenguaje C/C++. No obstante, el soporte de librerías gráficas es más limitado ya que no tiene librerías implementadas para ello, sino que tiene que hacer uso de librerías externas de OpenGL.

- **QTCreator.** Entorno de desarrollo multiplataforma y multilenguaje. Dispone de librerías gráficas nativas específicas.

- **Visual Studio.** Entorno de desarrollo multiplataforma y también multilenguaje. Dispone de librerías gráficas específicas que hacen uso de la aceleración por hardware mediante DirectX.

En base a los puntos anteriores me he decantado por el uso del entorno de desarrollo **Visual Studio Enterprise 2019**, ya que cumple con todos los criterios. Además, contiene un vasto paquete de complementos de todo tipo y soporta programación en varios lenguajes (**Visual Basic**, **C++**, **Java**, **C#)**.

Se podría haber tenido en cuenta como alternativa el IDE de **QTCreator**, pero el IDE de **Visual Studio** es ya un software bastante maduro (con bastantes años de desarrollo y mejoras) y respaldado por una comunidad de programadores bastante grande.

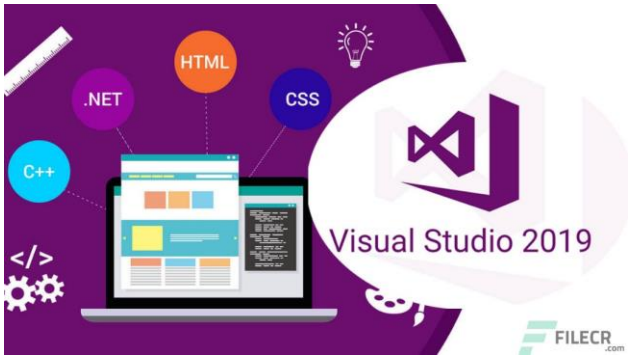


Ilustración 5. Logotipo del IDE Visual Studio 2019

Finalmente, el lenguaje de programación que he elegido en este IDE es **C#**, ya que a diferencia de los otros lenguajes en que se puede programar **Visual Studio**, este ha sido creado nativamente para esta plataforma por lo que no posee elementos heredados por antiguas revisiones de este lenguaje y por lo tanto se encuentra más depurado siendo más fácil e intuitivo de programar.

7.- Descripción de la solución adoptada.

Para el desarrollo del simulador se ha optado por aplicar una metodología de ciclo de vida de producción. Existen varias metodologías de ciclo de vida, por lo que se tiene que elegir la más adecuada.

7.1.- Ciclo de vida de producción

En este caso, teniendo en cuenta el tipo de producto en concreto (software), los requisitos del producto (y por tanto el diseño) pueden variar conforme avanza el desarrollo del producto. En consecuencia, la metodología más acertada se centra en el ciclo de vida clásico con realimentación de las fases anteriores.

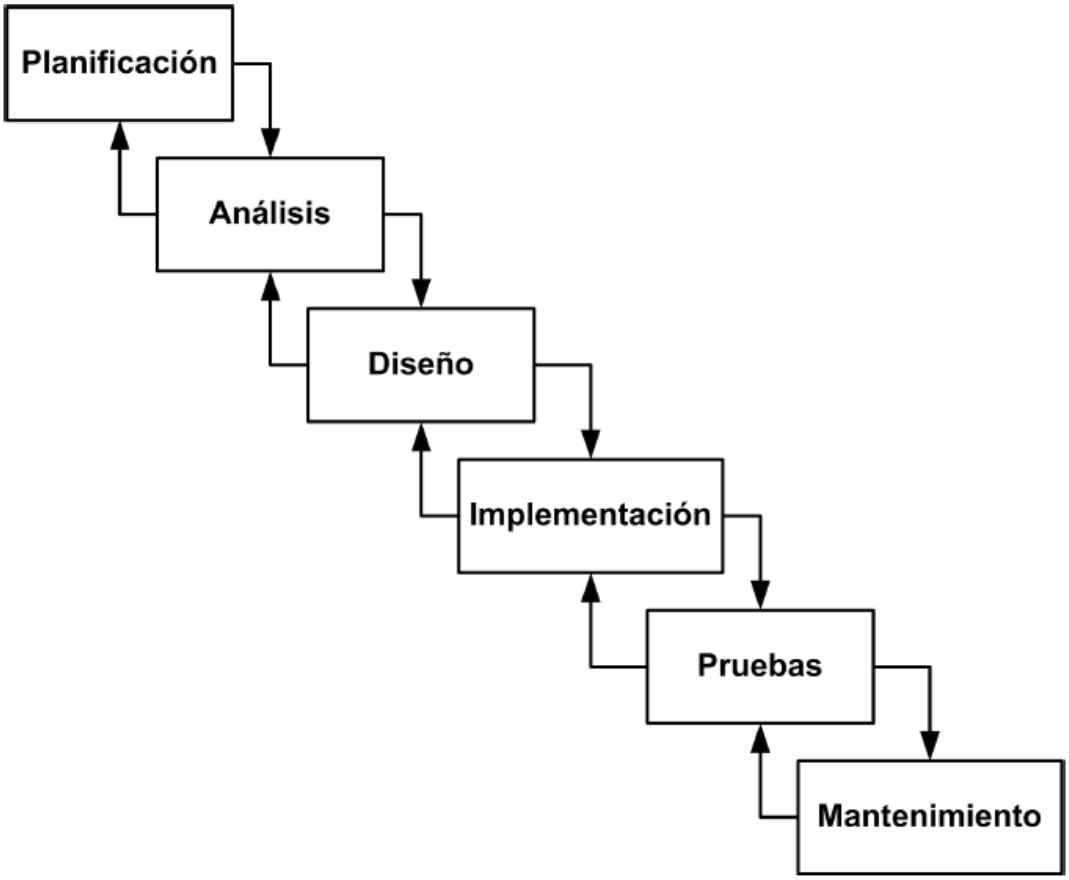


Ilustración 6. Ciclo de vida clásico con realimentación

Las fases más importantes de este ciclo de vida se desarrollan con un mayor detalle en las siguientes secciones de esta memoria.

- **Sección 8.** Se desarrollan las fases de **análisis y especificación de requisitos del producto.**
- **Sección 9.** Se desarrolla la fase de **diseño y modelado de la aplicación.**
- **Sección 10.** Se desarrolla la **implementación del programa.**
- **Sección 11.** Se desarrolla la fase de pruebas sobre posibles **casos de uso.**

7.2.- Arquitectura multicapa abierta.

Para simplificar el desarrollo de la aplicación, dicho simulador se desarrolla partiendo de una arquitectura multicapa.

Una **Arquitectura de Sistema** contempla el desarrollo de los diversos elementos agrupándolos por propiedades comunes que comparten. El tipo de Arquitectura más usada es la llamada **Arquitectura Multicapa** la cual agrupa los elementos de un Sistema agrupándolos en subsistemas que se encuentran ordenados por capas, o sea, uno encima del otro.

En este caso se ha optado por una arquitectura de 3 capas, agrupadas como se muestran en la Ilustración 7. Cada uno de estas capas se corresponde con:

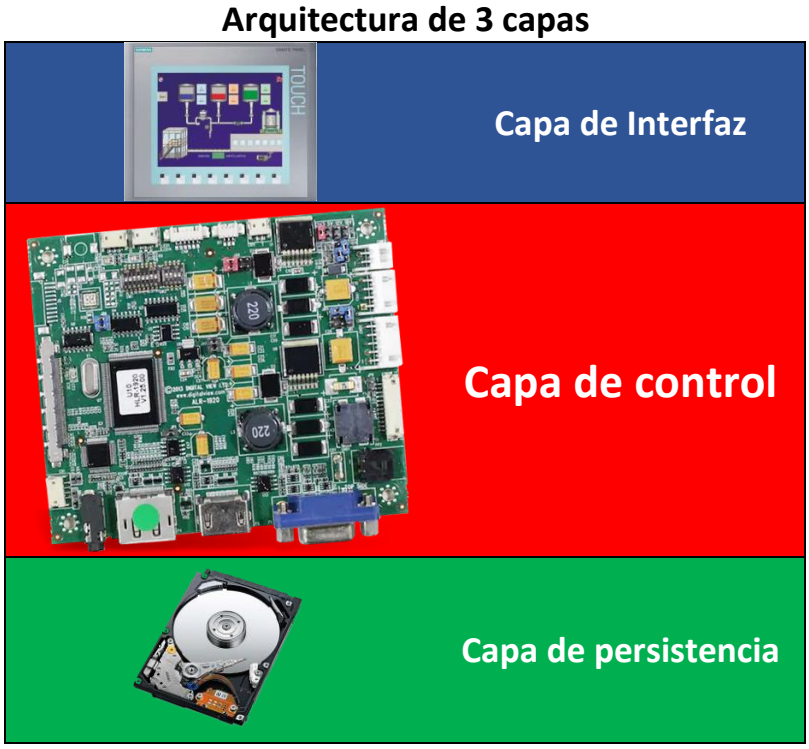


Ilustración 7. Arquitectura de 3 capas

- **Capa de interfaz.** En esta capa se agrupan todos los elementos que tienen que interactuar con los usuarios a través del interfaz. Se suele componer de elementos que realizan representaciones visuales o interacción a través de un terminal de texto. Los elementos de este nivel no contienen funcionalidades de la operativa interna del programa.

- **Capa de control.** En esta capa se agrupan todos los elementos que conforman el control interno del simulador.

.

- **Capa de Persistencia.** Esta capa por lo general no suele desarrollarse por los programadores, sino que es proporcionado por el Sistema Operativo o la Base de Datos. El nivel ofrece los servicios necesarios para la gestión del almacenamiento permanente de datos, como puede ser el acceso a los datos de un archivo o a los de la Base de Datos.

La arquitectura de estas capas es abierta, o sea, que por ejemplo el objeto abrir del nivel de presentación puede comunicarse directamente con las funciones de abrir un archivo del nivel de persistencia sin tener que pasar a través del nivel de aplicación.

Aunque se recomienda que la arquitectura sea cerrada (una capa solo se comunica con sus capas vecinas), en este caso esta forma de comunicación necesitaría complicar bastante el nivel de aplicación con objetos dedicados solo a la comunicación entre capas, acción que tampoco es recomendable.

Además de todos los niveles anteriores, hay que tomar en cuenta que la aplicación que se está desarrollando se basa en un sistema de simulación dinámica.

Por esto mismo, se incorporará un mecanismo que aplique un control de temporización sobre los mecanismos que interactúen en la simulación, los cuales son los objetos que constituyen el nivel de aplicación.

7.3.- Programación orientada a objetos (POO)

Aunque existen muchos lenguajes de programación distintos, en este caso el simulador se basa en la imitación de elementos físicos del mundo real, por tanto, la mejor opción es aplicar un desarrollo de la aplicación orientada a objetos.

Este tipo de programación orientado a objetos es dominante en los actuales controladores industriales PLC (Programmable Logic Controller). Ya que permite la conceptualización dentro del programa de los distintos elementos a controlar.

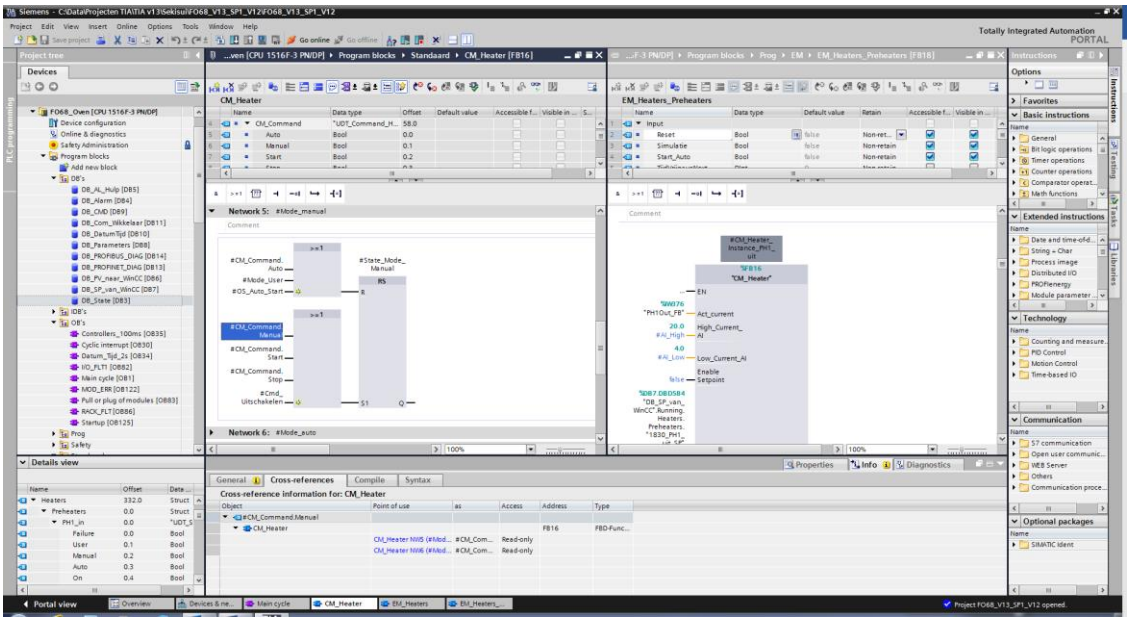


Ilustración 8.Siemens TIA Portal 13. Programación en bloques de funciones

8.- Análisis y Especificación de Requisitos del producto.

La fase de análisis es la primera fase del Ciclo de Vida de la creación de cualquier producto en donde se recogen todos los requerimientos para su posterior desarrollo.

En este caso, es el tutor el que ha aportado los requisitos que debe cumplir el simulador; que elementos tiene que simular, la funcionalidad de este, el objetivo de su uso, restricciones, diseño…

Para la documentación formal de los diversos requisitos del software del simulador se ha aplicado especificación estándar ISO/IEC/IEEE 29148, la cual es la más usada por los desarrolladores para describir una Especificación de Requisitos de calidad.

La creación y el uso de esta clase de documentos como referencia permitirá reducir el tiempo de desarrollo, de modificación, de verificación y validación del software, ya que fuerza a profundizar en lo que realmente se solicita y se tiene más claro lo que se ha de implementar y cuál es la mejor forma.

En el Anexo 1 se presenta el documento de Especificación de Requisitos final generado para este proyecto. En este se puede observar los detalles del mismo.

9.- Diseño y modelado del simulador (UML).

Una vez documentados todos y cada uno de los requisitos y restricciones que va a requerir y cumplir nuestro software dentro de la ERS (Especificación de Requisitos del Software), proseguimos a realizar un modelado de estructurado conceptual sobre las diferentes clases que hemos definido de la aplicación.

La construcción de Modelos visuales para representar los diferentes atributos, funciones y relaciones internos de la aplicación, es el segundo paso para obtener un simulador de alta calidad.

Gracias a las especificaciones que hemos definido anteriormente en la ERS hemos mejorado notablemente el entendimiento del sistema que nos piden construir. Sin embargo, este elemento no cubre todos los aspectos que puede abarcar un modelado conceptual de la aplicación.

Al ser un Modelo un sistema gráfico es más fácil e intuitivo la comprensión de este que teniendo todas las especificaciones desarrolladas sobre un listado, por lo que es más estable. Igualmente facilita la comprensión de sistemas mucho más complejos.

Los Modelos generados anteriormente pueden ser reutilizables para el modelado de otras aplicaciones o problemas similares, pudiéndose variar ligeramente; ya que un Modelado se puede comprender y razonar sobre él sin necesidad de revisar la implementación de su respectiva aplicación.

Además, a partir de un Modelado se puede generar las plantillas para la estructura básica para el simulador. Actualmente existen muchas herramientas de creación de modelado que son capaces de generar estos esqueletos de código automáticamente.

Como se va a desarrollar una programación orientada a objetos, se usará el sistema de modelado que es más complejo para el desarrollo de modelos orientados a objetos, y el más aceptado y usado por los desarrolladores, el Lenguaje Unificado de Modelado (UML).

UML es un "lenguaje de modelado" para especificar y describir métodos o procesos. Se utiliza para definir un sistema complejo, para resaltar los diversos atributos de ese sistema y poderlos documentar y desarrollar posteriormente.

Aunque puede definir muy bien sistemas de modelados conceptuales sobre objetos, no se encuentra limitado a estos modelos, pudiéndose desarrollar bajo éste otros tipos de modelado distinto.

La versión actual en que se encuentran las especificaciones UML es la 2.5 que es reconocida como estándar de Modelado y está respaldada por el consorcio OMG (Object Management Group).

9.1.- Diagrama de clases (Control)

Para la creación del modelado conceptual orientado a objetos de la aplicación que se va a desarrollar, se ha optado por el software de diseño de modelos UML MOSKitt, software gratuito para diseñar modelos UML promovido por la Generalitat Valenciana.

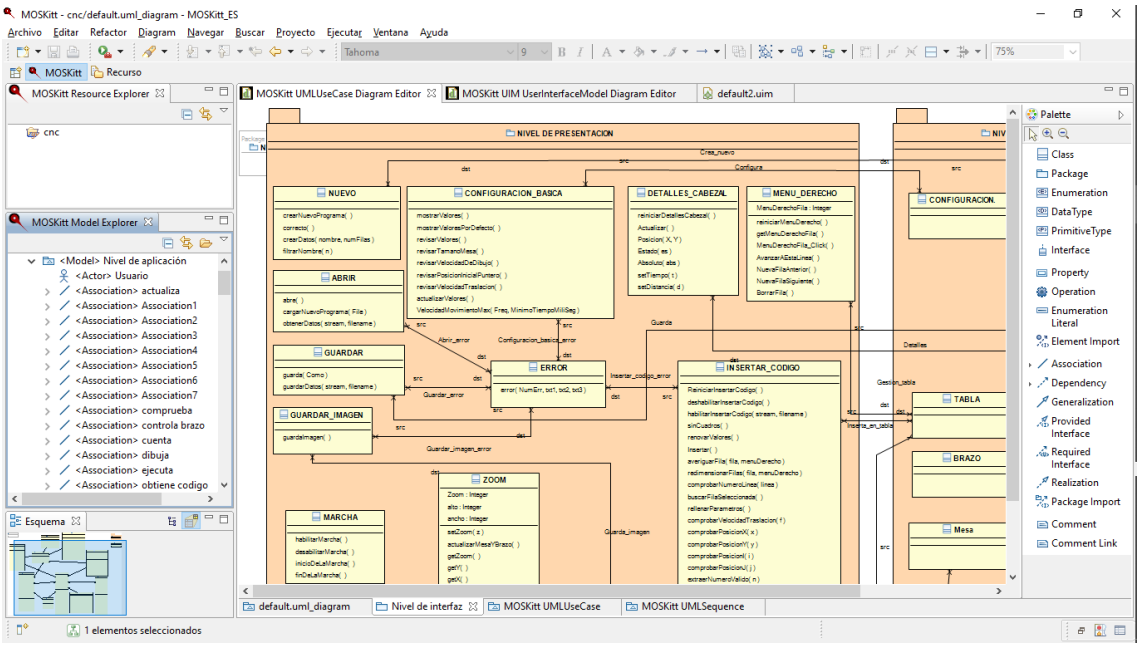


Ilustración 9. IDE de la aplicación CASE MOSkitt

El **diagrama de clases UML** desarrollado para el simulador se encuentra en el **Anexo 2**.

En este se observa claramente que faltan clases que se han definido anteriormente en el ERS, esto se debe a que los diagramas conceptuales de clases solo representan el modelado del funcionamiento interno (conceptual) de la aplicación ya que se da por hecho que es independiente de la interfaz y por tanto no se modelan los elementos de esta.

Por tanto, el diseño de este modelado UML de la aplicación puede implementarse sobre cualquier sistema de interfaz externa sin que tenga que adaptarse. Da igual que sea sobre una interfaz gráfica, de texto, sensorial, etc.

Por otra parte, también se observa en el diagrama UML que se podría modificar las clases MOVIMIENTO y ESTATICO para heredar de la clase BRAZO.

En un principio sería buena la correspondencia de herencia, sin embargo, como se observa en el modelado, la clase ESTATICO y MOVIMIENTO difieren bastante en su funcionalidad, por lo que crear herencia de estos a BRAZO no haría más que aumentar las clases MOVIMIENTO y ESTÁTICO con atributos y funciones que posiblemente no se usarían nunca.

Además, como requisito no evitable, el programa está pensado para poder pasar del modo de simulación dinámica a simulación estática dentro del mismo proceso de ejecución, por eso es mejor tener la clase BRAZO como directora de los movimientos y a las otras dos como extensiones de atributos y funciones de la clase BRAZO.

9.2.- Diagrama de clases (Interfaz)

Para tener una mejor concepción de los objetos de la interfaz dentro de la capa de interfaz, se hace uso del mismo software de modelado UML para modelar un Diagrama de Clases que represente convenientemente los elementos de la interfaz.

Aunque se puede hacer el modelado UML, este no está contemplado dentro del desarrollo normativo UML para modelados de interfaces, ya que no forma parte de su concepción de modelado.

Se ha usado el software de MOSkitt para modelar un Diagrama de Clases sobre los elementos de la interfaz ya que, como objetos, son modelables por el sistema de UML.

El resultado del **diagrama de clases de la interfaz** se encuentra disponible en el **Anexo 3**.

9.3.- Modelos de prueba (Casos de uso)

Esta parte del modelado desarrolla algunos posibles casos de uso del Simulador CNC. Estos casos de uso son varios escenarios distintos del uso de la aplicación que se deben poder realizar.

Estos casos de uso se usarán como modelos de prueba para desarrollar las pruebas de funcionamiento del programa Simulador CNC una vez se encuentre desarrollado.

El autor de los casos de uso va a ser siempre el mismo agente, en este caso el usuario del programa (Ilustración 10).

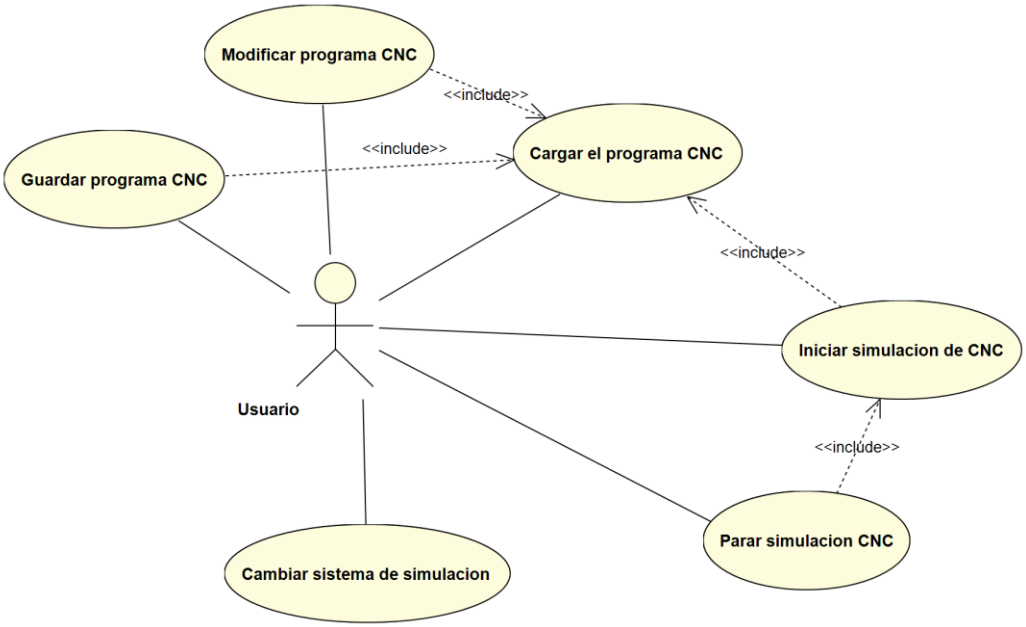


Ilustración 10. Diagrama de casos de uso

Se han definido en total tres escenarios distintos:

- **Escenario 1:**

Autores	Usuario
Objetivo asociado	Abrir y ejecutar un programa de mecanizado
Descripción	El usuario abre un nuevo programa de mecanizado y lo ejecuta para comprobar el resultado en un momento intermedio en el que el usuario detiene la simulación.
Precondición	
Secuencia normal	**Paso** / **Acción** 1 / El usuario abre el programa CNC 2 / Si el programa es correcto se queda la aplicación preparada, si no es correcto *pasa al Paso 2 de Excepciones* 3 / El usuario inicia la simulación del sistema, si no se cumple el *Paso 3 de Excepciones* continúa 4 / El sistema simula la ejecución, si no se cumple el *Paso 4 de Excepciones* continúa 5 / El usuario para la simulación del sistema y el Caso de uso termina.
Postcondición	

<table>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td></td></tr>
<tr><td>2</td><td>Si el programa no es correcto se aborta la preparación del programa, se elimina y se reinicia la máquina virtual y termina el Caso de uso</td></tr>
<tr><td>3</td><td>Si el programa no es correcto aborta el inicio de la simulación y:
Si el usuario quiere cargar un nuevo programa pasa al Paso 1 de Secuencia normal
Si el usuario no carga un nuevo programa el Caso de uso termina.</td></tr>
<tr><td>4</td><td>Si la simulación del sistema da un fallo en ejecución del programa CNC sobre la máquina virtual aborta la simulación y:
Si el usuario quiere cargar un nuevo programa pasa al Paso 1 de Secuencia normal
Si el usuario no carga un nuevo programa el Caso de uso termina.</td></tr>
</table>
</td></tr>
<tr><td>Rendimiento</td><td>
<table>
<tr><th>Paso</th><th>Cola de tiempo</th></tr>
<tr><td>1</td><td>20 segundos</td></tr>
<tr><td>2</td><td>< 1 segundo</td></tr>
<tr><td>3</td><td>< 1 segundo</td></tr>
<tr><td>4</td><td>15 segundos</td></tr>
<tr><td>5</td><td>< 1 segundo</td></tr>
</table>
</td></tr>
<tr><td>Frecuencia esperada</td><td>1 vez por cada programa cargado. 1 * 38 segundos</td></tr>
<tr><td>Importancia</td><td>Importante</td></tr>
<tr><td>Urgencia</td><td>Puede esperar</td></tr>
<tr><td>Comentarios</td><td></td></tr>
</table>

- **Escenario 2:**

Autores	Usuario
Objetivo asociado	Modificación del sistema de simulación
Descripción	El usuario abre un nuevo programa de mecanizado y lo ejecuta, en un momento intermedio de repente cambia el sistema de simulación y espera a que termine todo el programa para comprobar el resultado.
Precondición	

<table>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario abre el programa de mecanizado</td></tr>
<tr><td>2</td><td>Si el programa es correcto se queda la aplicación preparada, si no es correcto pasa al Paso 2 de Excepciones</td></tr>
<tr><td>3</td><td>El usuario inicia la simulación del sistema, si no se cumple el Paso 3 de Excepciones continúa</td></tr>
<tr><td>4</td><td>El sistema simula la ejecución del programa, si no se cumple el Paso 4 de Excepciones continúa</td></tr>
<tr><td>5</td><td>El usuario cambia el sistema de simulación a estático</td></tr>
<tr><td>6</td><td>El sistema adapta el tipo de simulación al nuevo definido</td></tr>
<tr><td>7</td><td>El sistema reanuda la ejecución del programa, si no se cumple el Paso 4 de Excepciones continúa</td></tr>
<tr><td>8</td><td>El sistema simula la ejecución de todo el programa hasta su fin</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td></td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td></td></tr>
<tr><td>2</td><td>Si el programa no es correcto se aborta la preparación del programa, se elimina y se reinicia la máquina virtual y termina el Caso de uso</td></tr>
<tr><td>3</td><td>Si el programa no es correcto aborta el inicio de la simulación y:
Si el usuario quiere cargar un nuevo programa pasa al Paso 1 de Secuencia normal
Si el usuario no carga un nuevo programa el Caso de uso termina.</td></tr>
<tr><td>4</td><td>Si la simulación del sistema da un fallo en ejecución del programa CNC sobre la máquina virtual aborta la simulación y:
Si el usuario quiere cargar un nuevo programa pasa al Paso 1 de Secuencia normal
Si el usuario no carga un nuevo programa el Caso de uso termina.</td></tr>
<tr><td>5</td><td></td></tr>
<tr><td>6</td><td></td></tr>
<tr><td>7</td><td></td></tr>
<tr><td>8</td><td></td></tr>
</table>
</td></tr>
<tr><td>Rendimiento</td><td>
<table>
<tr><th>Paso</th><th>Cola de tiempo</th></tr>
<tr><td>1</td><td>20 segundos</td></tr>
<tr><td>2</td><td>< 1 segundo</td></tr>
<tr><td>3</td><td>< 1 segundos</td></tr>
<tr><td>4</td><td>15 segundos</td></tr>
<tr><td>5</td><td>< 1 segundo</td></tr>
</table>
</td></tr>
</table>

	6 | < 1 segundo 7 | < 1 segundo 8 | 22 segundos
Frecuencia esperada	1 vez por cada programa cargado. 1 * 62 segundos
Importancia	Importante
Urgencia	Puede esperar
Comentarios	

- **Escenario 3:**

Autores	Usuario
Objetivo asociado	Edición de un programa abierto
Descripción	El usuario abre un nuevo programa de mecanizado y lo ejecuta, en un momento intermedio parará el programa, realizará un cambio directo sobre la tabla de códigos, reanudará el programa hasta el final y finalmente guardará el programa y saldrá de la aplicación
Precondición	
Secuencia normal	**Paso** | **Acción** 1 | El usuario abre el programa de mecanizado 2 | Si el programa es correcto se queda la aplicación preparada, si no es correcto *pasa al Paso 2 de Excepciones* 3 | El usuario inicia la simulación del sistema, si no se cumple el *Paso 3 de Excepciones* continúa 4 | El sistema simula la ejecución del programa, si no se cumple el *Paso 4 de Excepciones* continúa 5 | El usuario detiene la simulación del sistema 6 | El sistema termina de procesar el código CNC en ejecución y para la simulación. 7 | El usuario realiza una pulsación doble sobre una fila de la tabla 8 | El sistema habilita la celda para edición 9 | El usuario modifica la celda con el nuevo argumento 10 | El sistema deshabilita la edición de la celda. 11 | El usuario reanuda la simulación del sistema, si no se cumple el *Paso 11 de Excepciones* continúa 12 | El sistema reanuda la ejecución del programa, si no se cumple el *Paso 4 de Excepciones* continúa 13 | El sistema simula la ejecución de todo el programa hasta el final 14 | El sistema para la simulación 15 | El usuario accede al menú de Guardar.

<table>
<tr><td></td><td><table>
<tr><td>16</td><td>El sistema guarda el programa modificado en el archivo.</td></tr>
<tr><td>17</td><td>El usuario finaliza la aplicación terminando el Caso de uso</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td></td></tr>
<tr><td>Excepciones</td><td><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td></td></tr>
<tr><td>2</td><td>Si el programa no es correcto se aborta la preparación del programa, se elimina y se reinicia la máquina virtual y termina el Caso de uso</td></tr>
<tr><td>3</td><td>Si el programa no es correcto aborta el inicio de la simulación y:
Si el usuario quiere cargar un nuevo programa pasa al Paso 1 de Secuencia normal
Si el usuario no carga un nuevo programa el Caso de uso termina.</td></tr>
<tr><td>4</td><td>Si la simulación del sistema da un fallo en ejecución del programa CNC sobre la máquina virtual aborta la simulación y:
Si el usuario quiere cargar un nuevo programa pasa al Paso 1 de Secuencia normal
Si el usuario no carga un nuevo programa el Caso de uso termina.</td></tr>
<tr><td>5</td><td></td></tr>
<tr><td>6</td><td></td></tr>
<tr><td>7</td><td></td></tr>
<tr><td>8</td><td></td></tr>
<tr><td>9</td><td></td></tr>
<tr><td>10</td><td></td></tr>
<tr><td>11</td><td>Si el programa no es correcto aborta el inicio de la simulación y:
Si el usuario quiere cargar un nuevo programa pasa al Paso 1 de Secuencia normal
Si el usuario quiere modificar nuevamente el programa pasa al Paso 7 de Secuencia normal
Si el usuario no carga un nuevo programa o no modifica el actual el Caso de uso termina.</td></tr>
<tr><td>12</td><td></td></tr>
<tr><td>13</td><td></td></tr>
<tr><td>14</td><td></td></tr>
<tr><td>15</td><td></td></tr>
<tr><td>16</td><td></td></tr>
<tr><td>17</td><td></td></tr>
</table></td></tr>
<tr><td>Rendimiento</td><td><table>
<tr><th>Paso</th><th>Cola de tiempo</th></tr>
<tr><td>1</td><td>20 segundos</td></tr>
<tr><td>2</td><td>< 1 segundo</td></tr>
</table></td></tr>
</table>

	3	< 1 segundos
	4	20 segundos
	5	< 1 segundo
	6	5 segundos
	7	2 segundos
	8	< 1 segundo
	9	18 segundos
	10	< 1 segundo
	11	< 1 segundo
	12	< 1 segundo
	13	34 segundos
	14	< 1 segundo
	15	7 segundos
	16	2 segundos
	17	4 segundos
Frecuencia esperada	1 vez por cada programa cargado. 1 * 120 segundos	
Importancia	Importante	
Urgencia	Puede esperar	
Comentarios		

9.4.- Diagramas de secuencias de los modelos de prueba.

En esta última sección del modelado UML, se desarrollan los diagramas de secuencias correspondientes a los tres casos de uso desarrollados en la sección 9.3.

- **Diagrama de secuencia del escenario 1:**

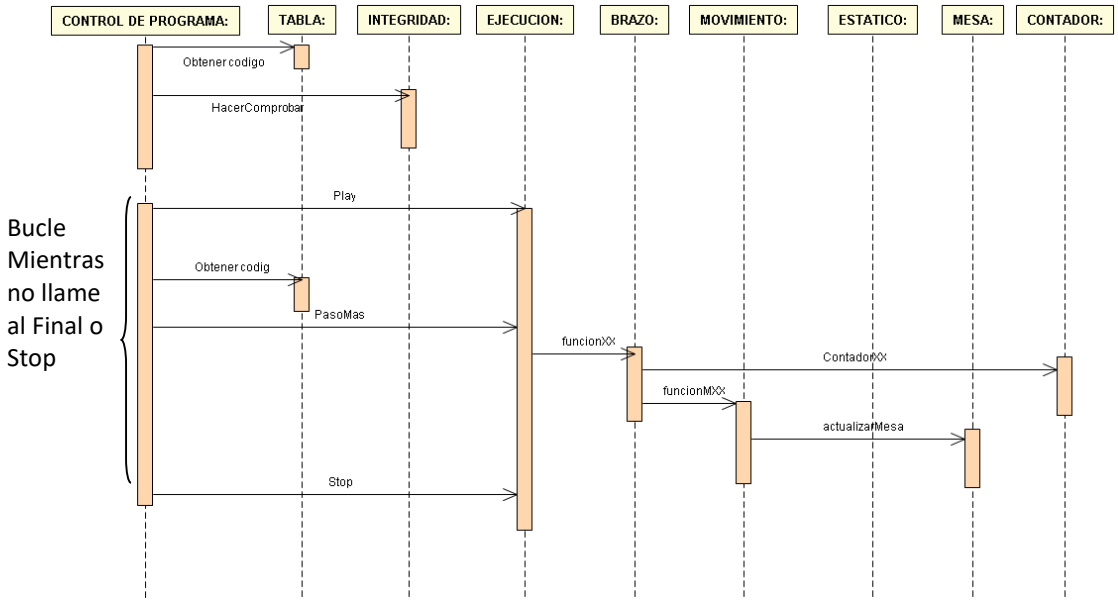


Ilustración 11. Diagrama de secuencia del escenario 1

- **Diagrama de secuencia del escenario 2:**

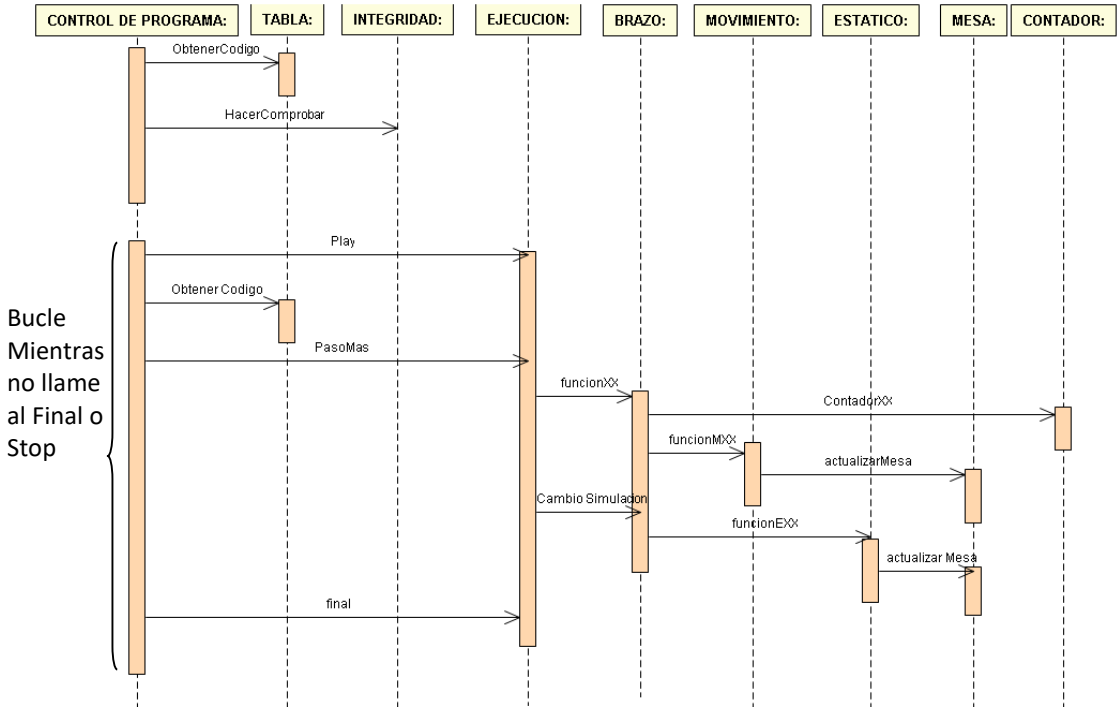


Ilustración 12. Diagrama de secuencia del escenario 2

- **Diagrama de secuencia del escenario 3:**

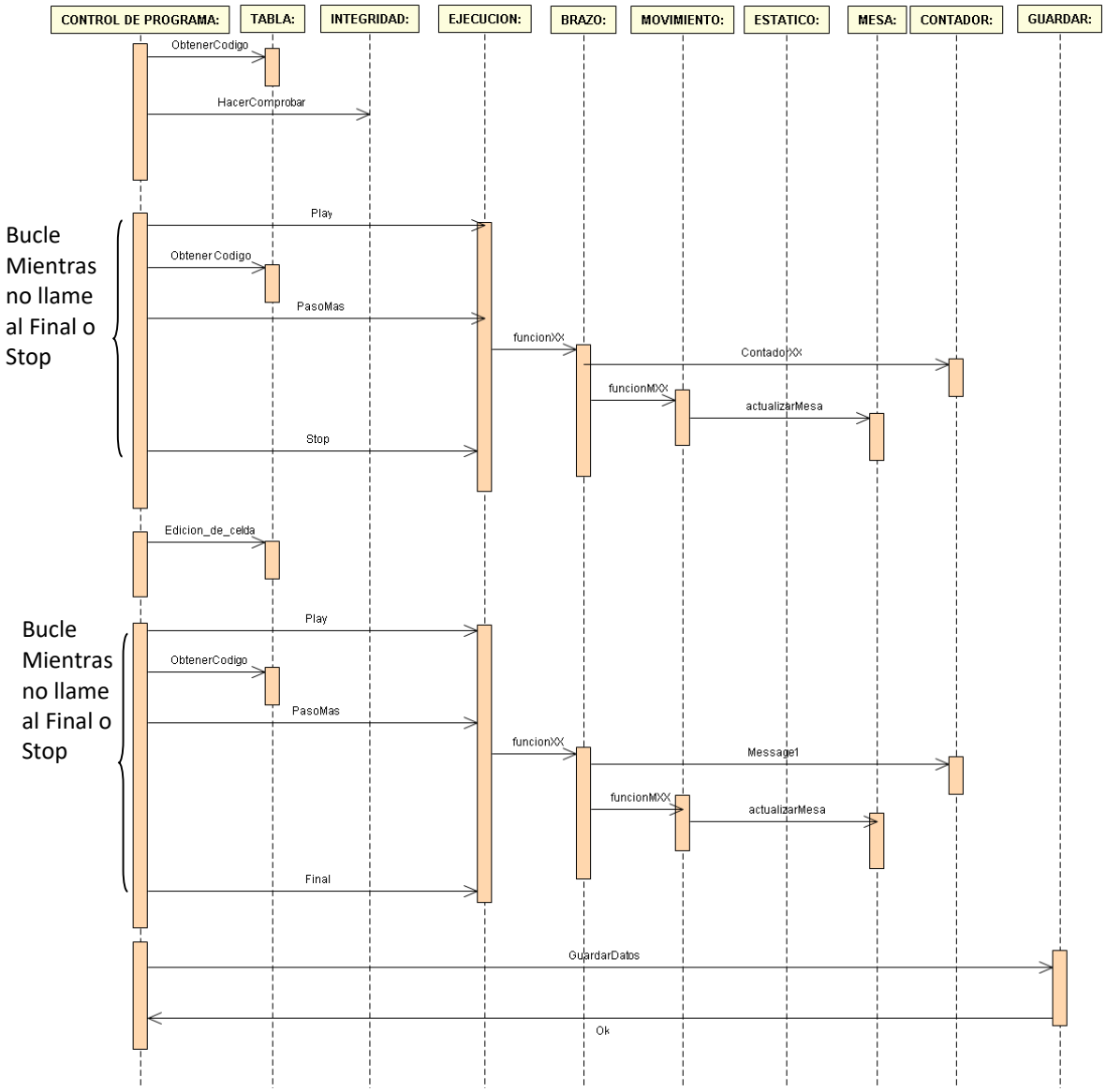
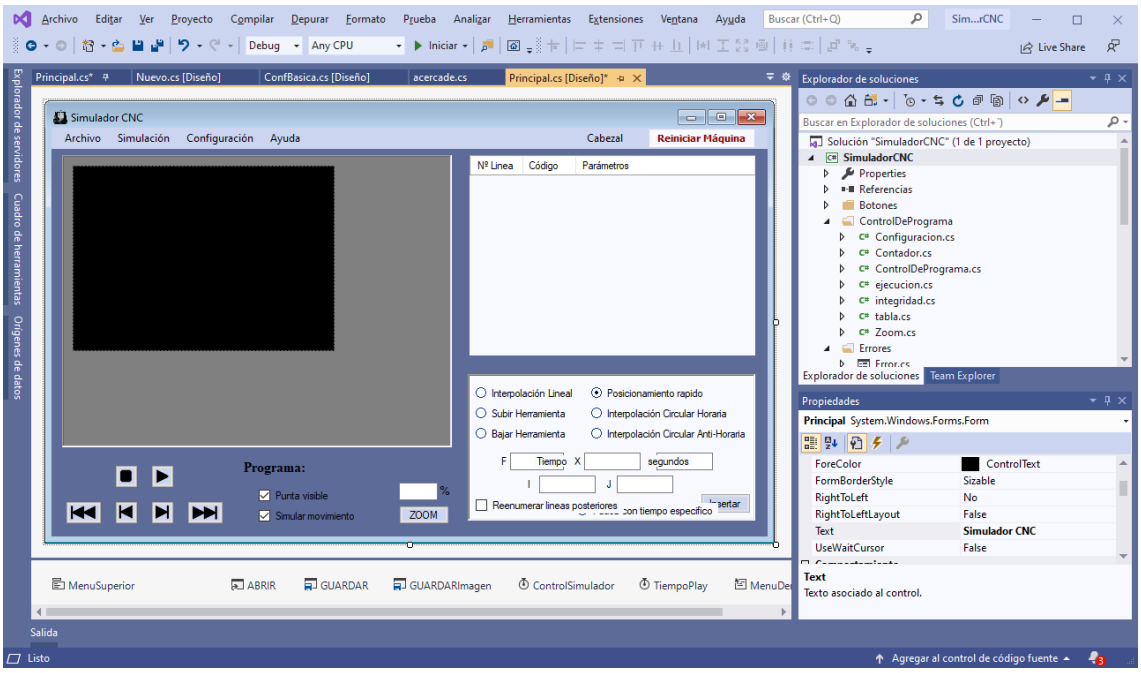


Ilustración 13. Diagrama de secuencia del escenario 3

Con estos escenarios de Casos de Uso se comprobará posteriormente si la aplicación del Simulador CNC funciona correctamente y dentro de las restricciones que se han definido.

10.- Implementación del simulador.

Como ya se ha expuesto previamente al analizar las posibles alternativas de entornos de desarrollo y lenguajes de programación para implementar el simulador, se ha optado por realizarlo mediante **Visual Studio Enterprise 2019**.



.

El lenguaje de programación usado es **C#**. La sintaxis de **C#** es muy parecida a la de lenguajes de programación **C++** o **Java**, con el soporte de un amplio número de librerías funcionales, pero manteniendo una relativa sencillez como la de **Visual Basic**.

La implementación de los objetos en código **C#** se ha desarrollado usando fielmente los modelos y diagramas UML.

En todo caso, la conversión del modelo en su símil de codificación en código no conlleva mayor problema, ya que el UML también define la metodología para derivar desde un elemento del modelado a una plantilla en formato de pseudocódigo ajustable a cualquier lenguaje de programación que esté orientado a objetos.

Con esta plantilla del programa creada en **C#** solo hace falta implementar internamente la funcionalidad dentro de cada una de las funciones de los objetos.

Las funcionalidades de los objetos de la implementación en código **C#** que se ha realizado en el simulador se han implementado cumpliendo con la función descrita en cada objeto dentro del documento de Especificaciones de Requisitos (Anexo 1).

Esta implementación en código **C#** está disponible en el **Anexo 5** de **Implementación del Simulador CNC**.

11.- Pruebas de funcionamiento.

Durante la implementación del Simulador CNC y posteriormente a la finalización de este software, se han realizado diversas pruebas de verificación de su correcto funcionamiento con respecto a los requisitos definidos en la Especificación de Requisitos (Anexo 1).

Por cada implementación de un objeto se han aplicado pruebas de caja blanca para comprobar la respuesta de los métodos del objeto ante datos de entrada normales, límites y datos no válidos.

Una vez finalizado totalmente la implementación del Simulador CNC, se ha verificado todos los casos de uso conceptuales modelados previamente en el diseño UML, validando el cumplimiento de cada uno de ellos en la implementación del software.

11.1.- Verificación de los modelos de prueba.

La comprobación de los Casos de uso, al ser estos escenarios imaginarios en los que se indica cómo tiene que interactuar la aplicación con el usuario, nos va a informar sobre la calidad del software implementado.

Si un Caso de uso no se cumple, nos está mostrando que, o bien el sistema no funciona correctamente o que la forma de funcionamiento de este no se ajusta a las especificaciones y restricciones definidas.

Por ejemplo, para la verificación de los Casos de uso que se han definido en la sección 9.3 de este documento, se ha comprobado que para cada paso explicado en este existe una situación o acción dentro del software que corresponde con este mismo paso y que permite además realizar las diversas acciones que se describen en los siguientes pasos a los que se puede derivar el paso anterior.

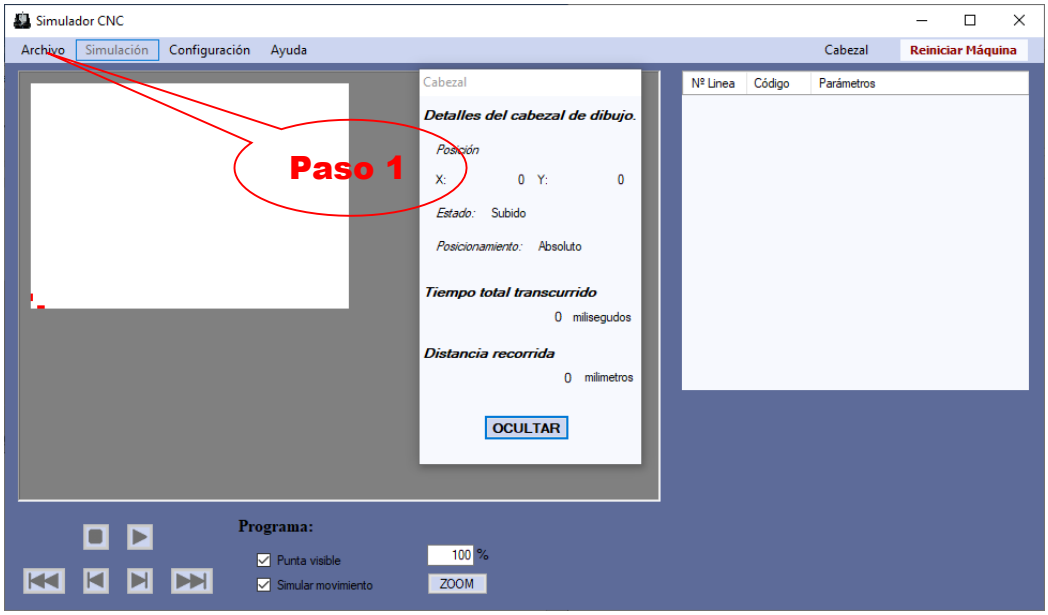
En conclusión, se trata de verificar sobre la implementación la secuencia de pasos que describe el escenario del Caso de uso. La forma de verificación resultante es la siguiente:

Nota informativa: Los pasos a los que puede derivar el interfaz del programa en ese momento se han enmarcado dentro de un globo de color rojo.

○ Verificación del escenario 1:

Autores	Usuario
Objetivo asociado	Cargar y abre un programa de mecanizado
Descripción	El usuario cargará un nuevo programa de mecanizado y lo ejecuta para comprobar el resultado en un momento intermedio en el que el usuario detiene la simulación.

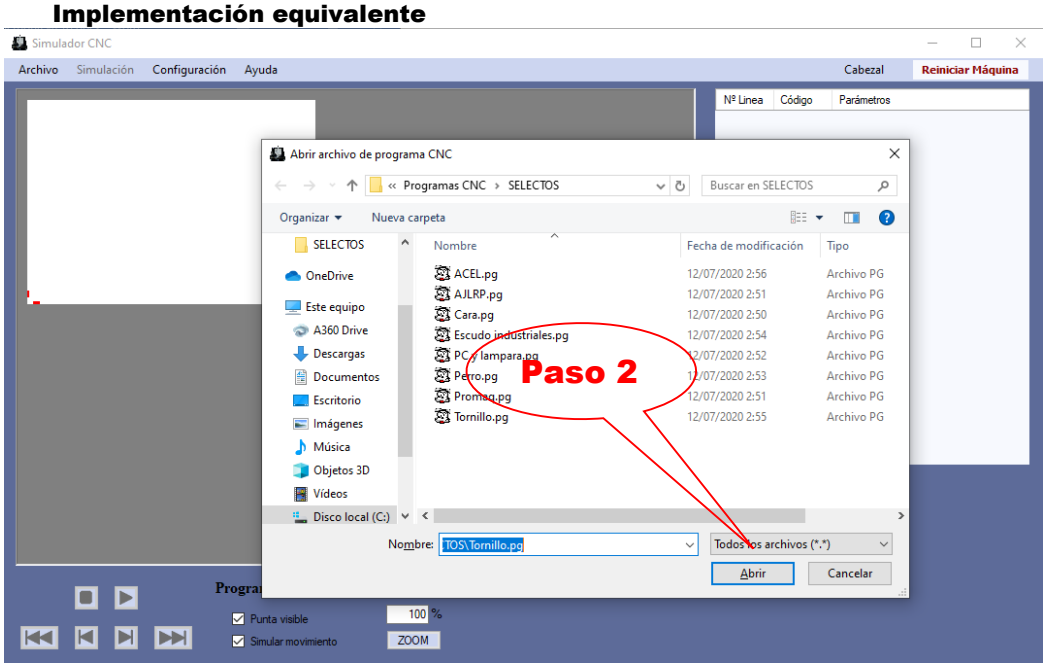
Estado inicial de la aplicación:



Paso 1 Secuencia normal:

El usuario carga el programa CNC.

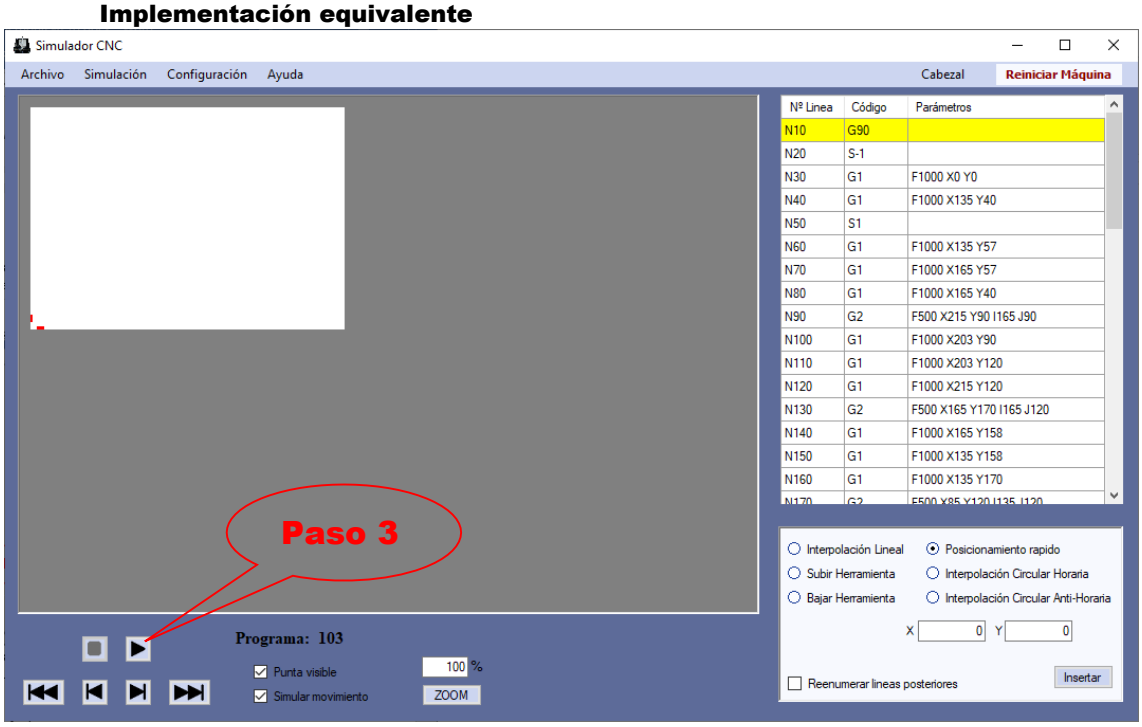
Implementación equivalente



Paso 2 Secuencia normal:

Si el programa es correcto se queda la aplicación preparada, si no es correcto *pasa al Paso 2 de Excepciones*.

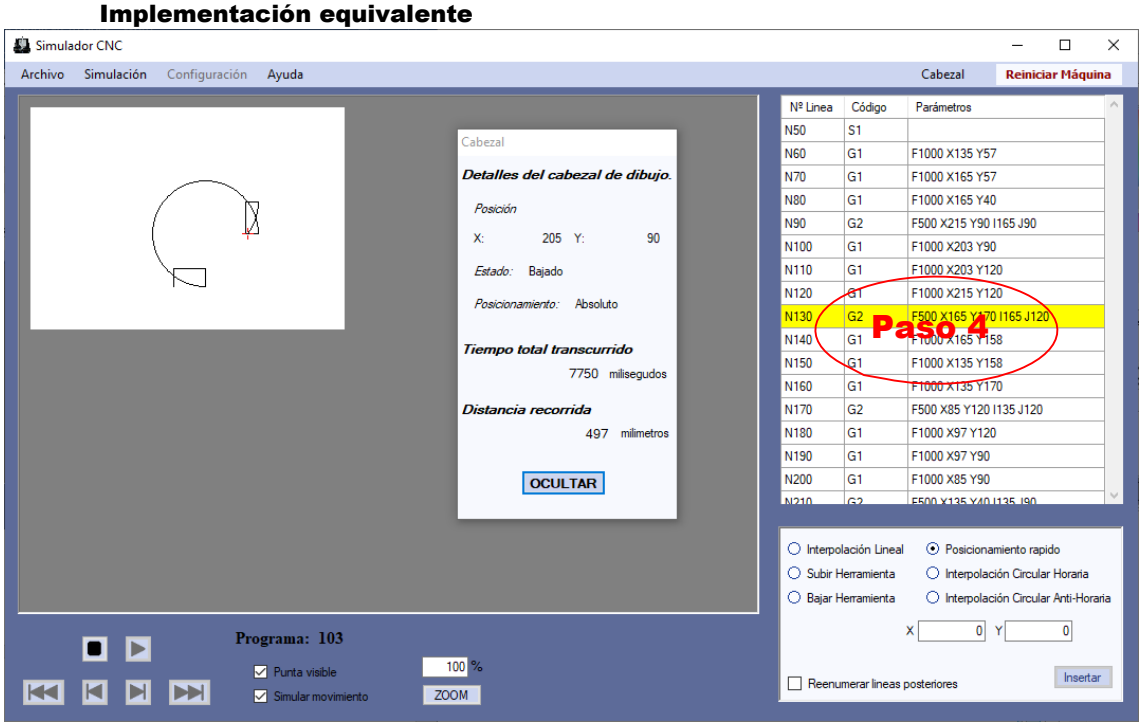
Implementación equivalente



Paso 3 Secuencia normal:

El usuario inicia la simulación del sistema, si no se cumple el *Paso 3 de Excepciones* continúa.

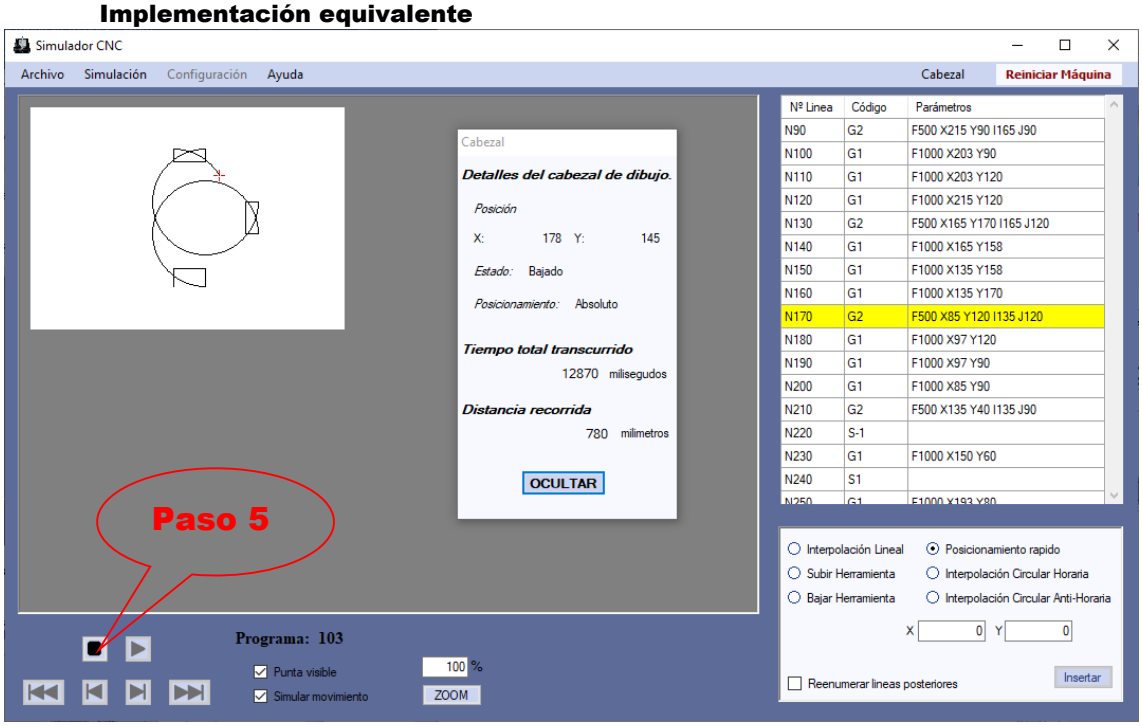
Implementación equivalente



Paso 4 Secuencia normal:

El sistema simula la ejecución, si no se cumple el *Paso 4 de Excepciones* continúa.

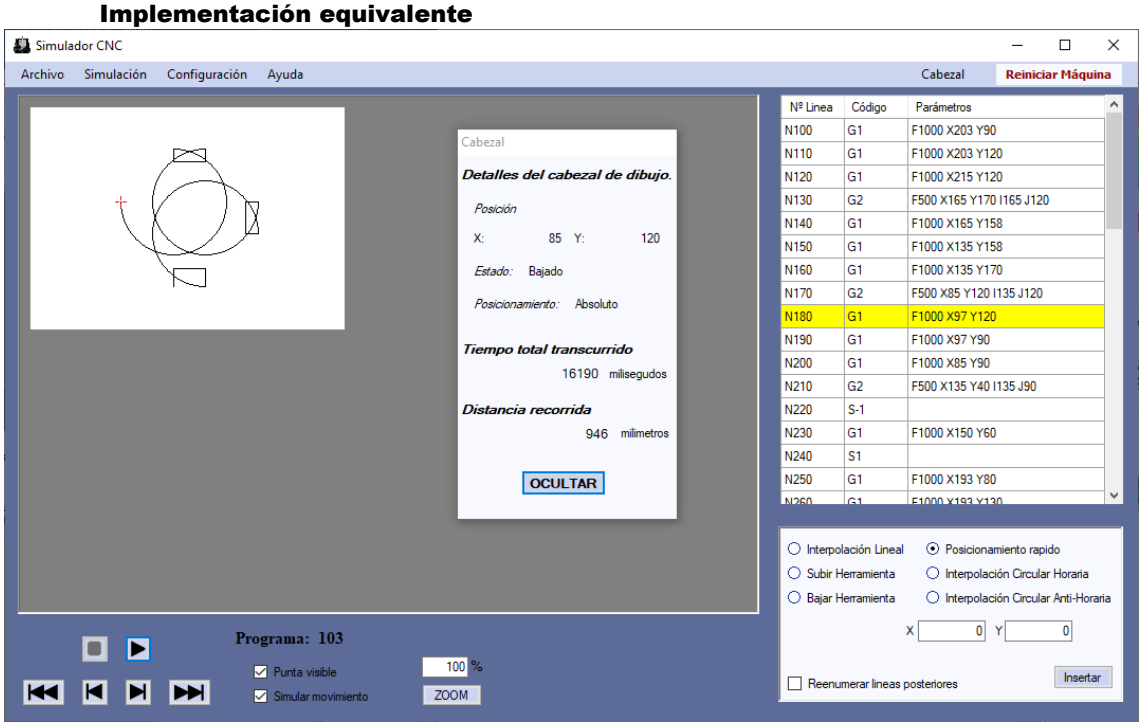
Implementación equivalente



Paso 5 Secuencia normal:

El usuario para la simulación del sistema y el Caso de uso termina.

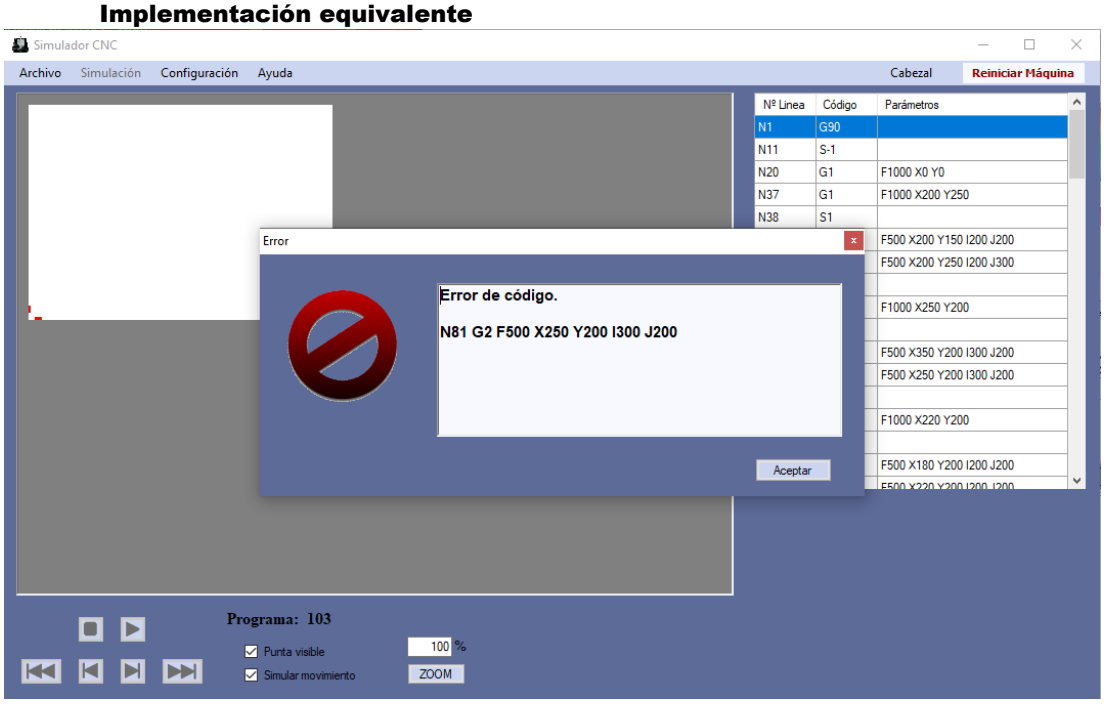
Implementación equivalente



Paso 2 Excepción:

Si el programa no es correcto se aborta la preparación del programa, se elimina y se reinicia la máquina virtual y termina el Caso de uso

Implementación equivalente

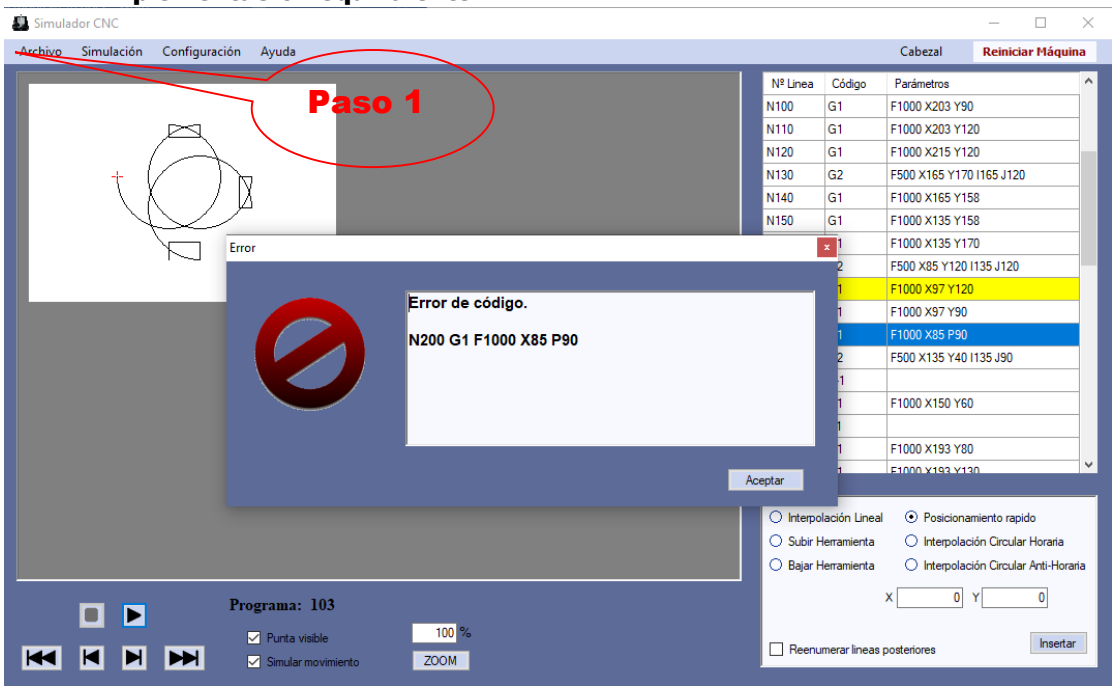


Paso 3 Excepción:

Si el programa no es correcto aborta el inicio de la simulación y:

- Si el usuario quiere cargar un nuevo programa *pasa al Paso 1 de Secuencia normal.*
- Si el usuario no carga un nuevo programa el Caso de uso termina.

Implementación equivalente

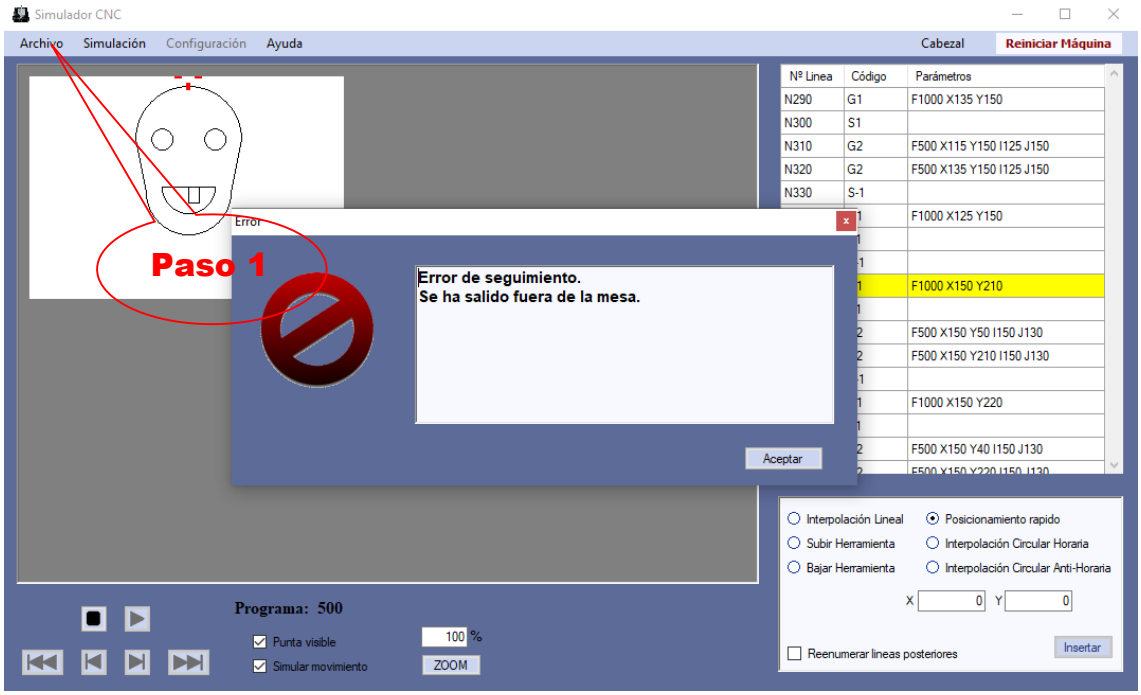


Paso 4 Excepción:

Si la simulación del sistema da un fallo en ejecución del programa CNC sobre la máquina virtual aborta la simulación y:

- Si el usuario quiere cargar un nuevo programa *pasa al Paso 1 de Secuencia normal*
- Si el usuario no carga un nuevo programa el Caso de uso termina.

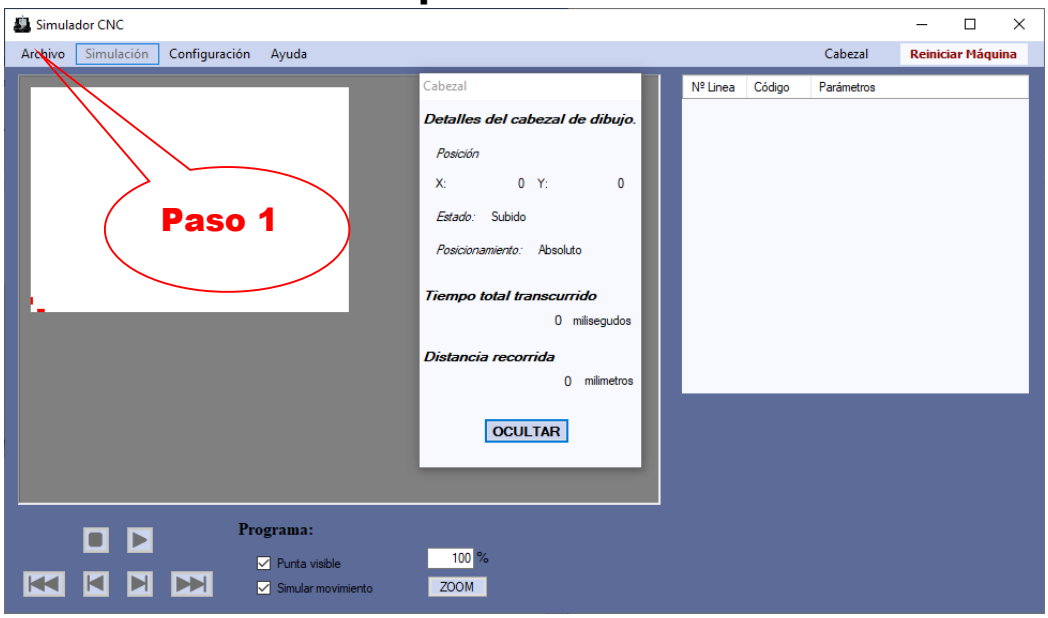
Implementación equivalente



○ Verificación del escenario 2:

Autores	Usuario
Objetivo asociado	Modificación del sistema de simulación
Descripción	El usuario abre un nuevo programa de mecanizado y lo ejecuta, en un momento intermedio de repente cambia el sistema de simulación y espera a que termine todo el programa para comprobar el resultado.

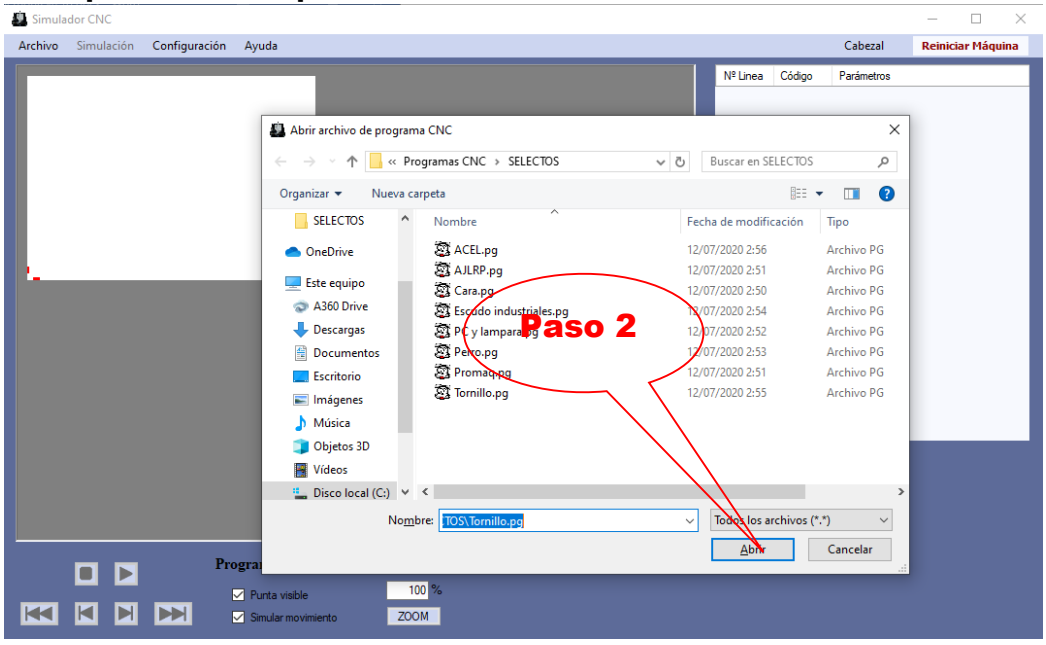
Estado inicial de la aplicación:



Paso 1 Secuencia normal:

El usuario carga el programa CNC.

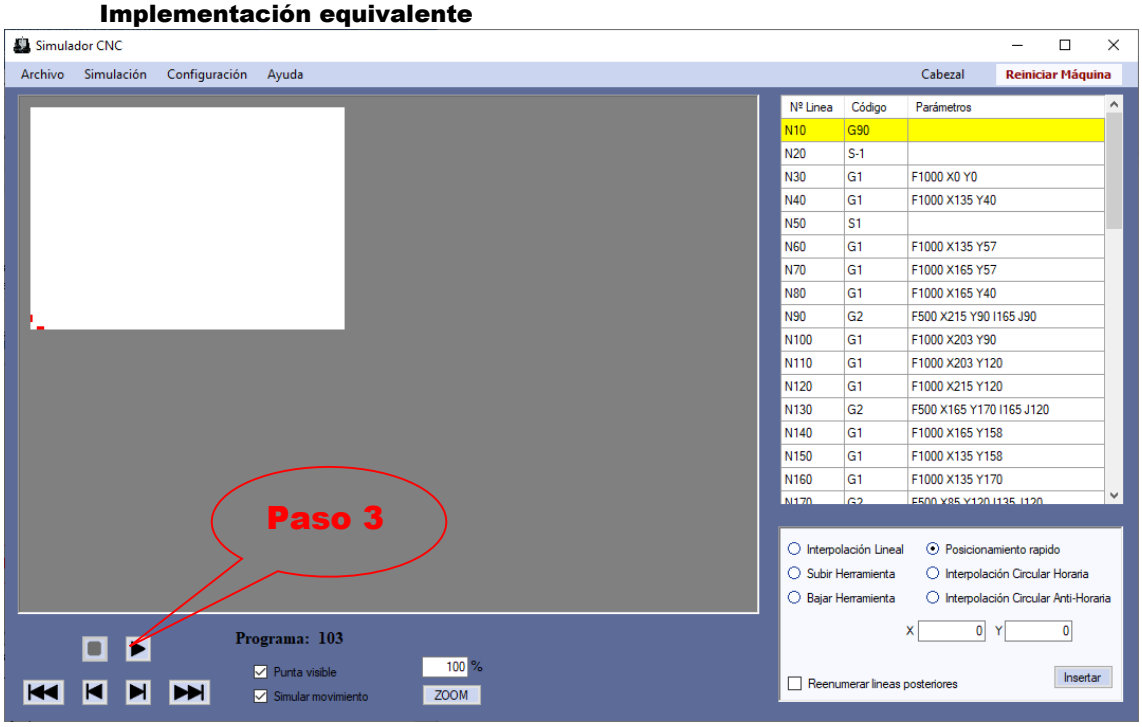
Implementación equivalente



Paso 2 Secuencia normal:

Si el programa es correcto se queda la aplicación preparada, si no es correcto *pasa al Paso 2 de Excepciones*.

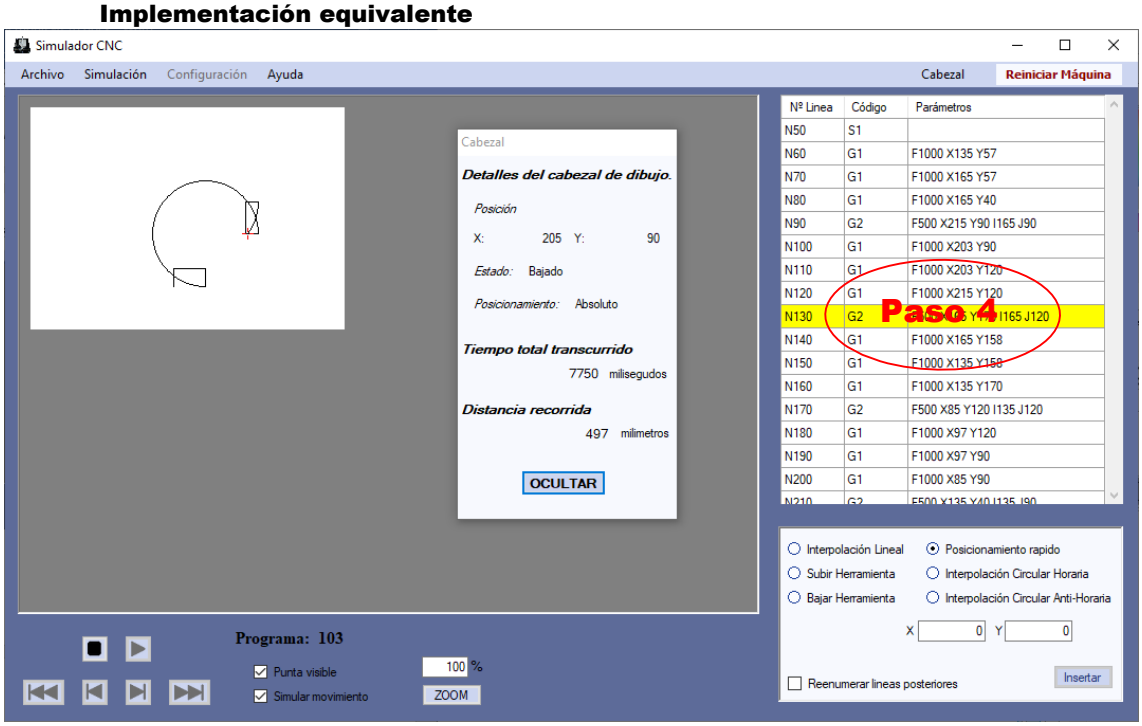
Implementación equivalente



Paso 3 Secuencia normal:

El usuario inicia la simulación del sistema, si no se cumple el *Paso 3 de Excepciones* continúa

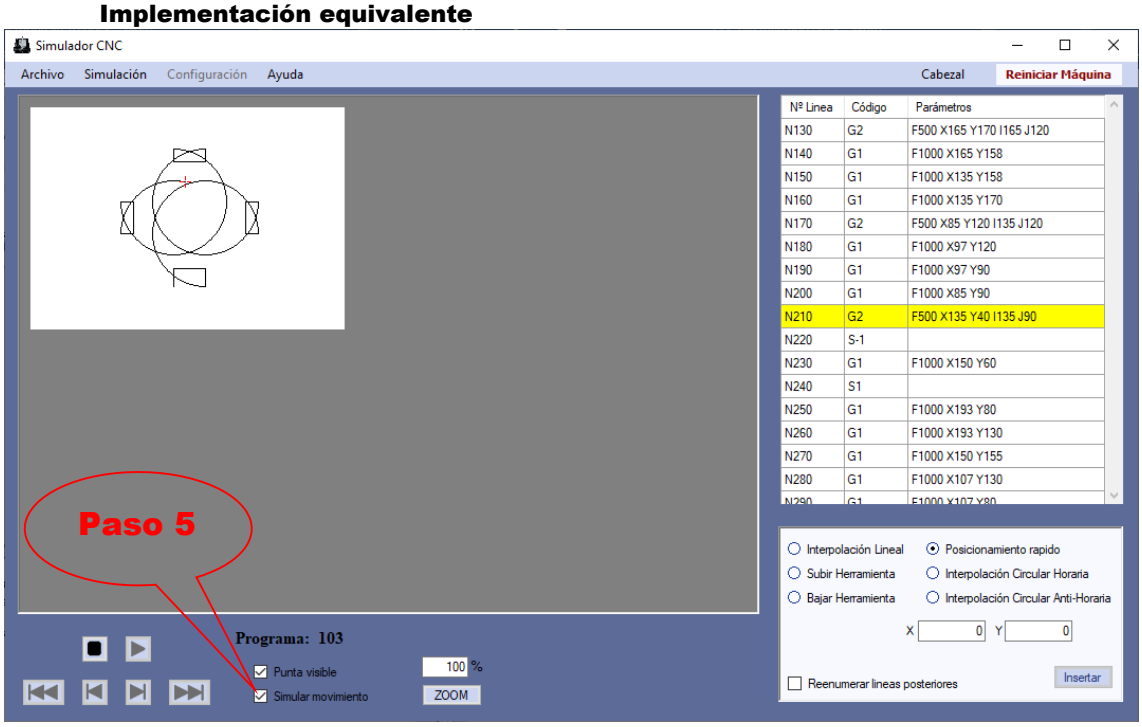
Implementación equivalente



Paso 4 Secuencia normal:

El sistema simula la ejecución del programa, si no se cumple el *Paso 4 de Excepciones* continúa

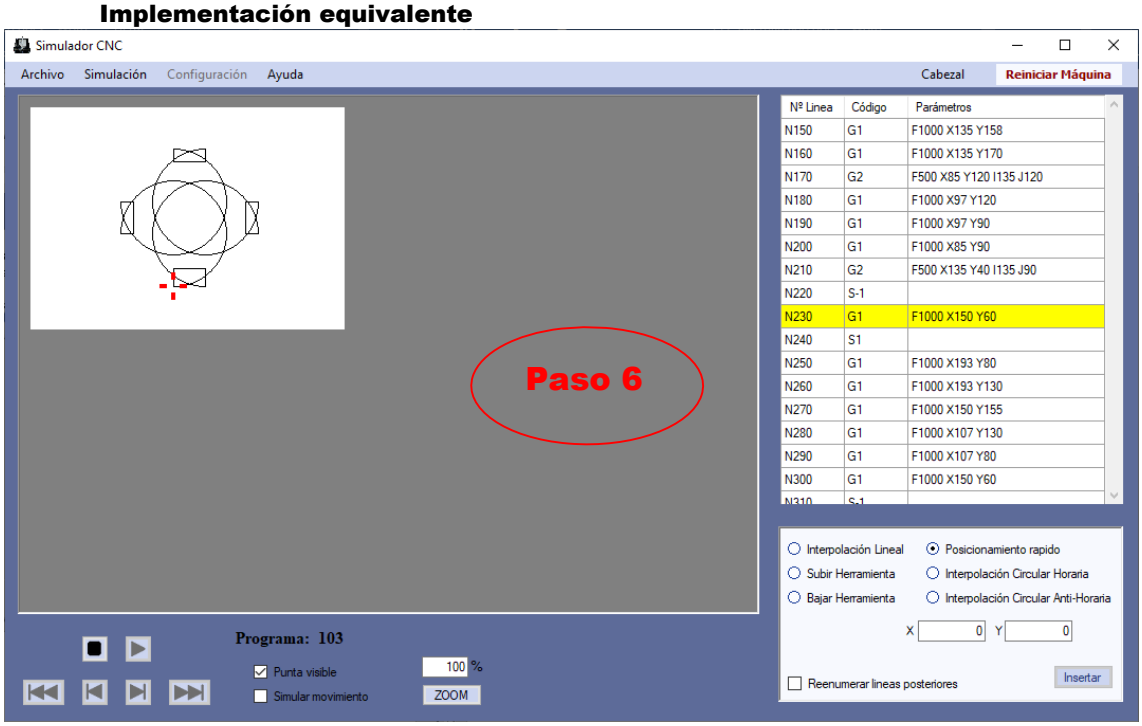
Implementación equivalente



Paso 5 Secuencia normal:

El usuario cambia el sistema de simulación a estático.

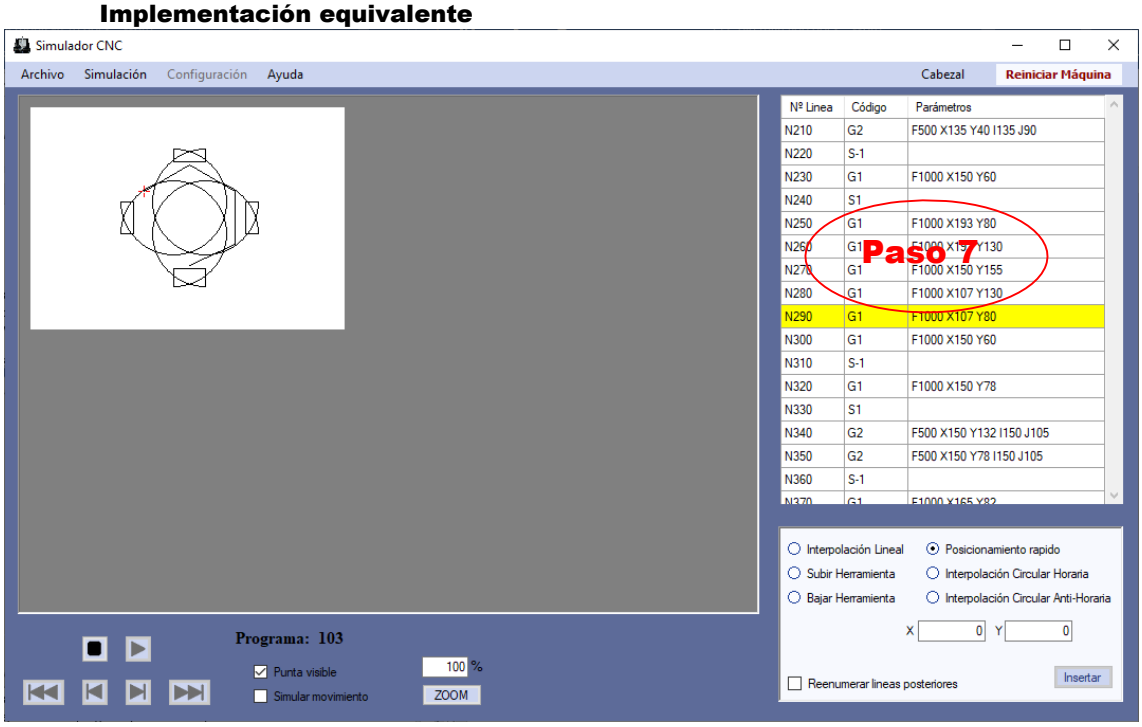
Implementación equivalente



Paso 6 Secuencia normal:

El sistema adapta el tipo de simulación al nuevo definido.
[Este paso no es observable desde la interfaz]

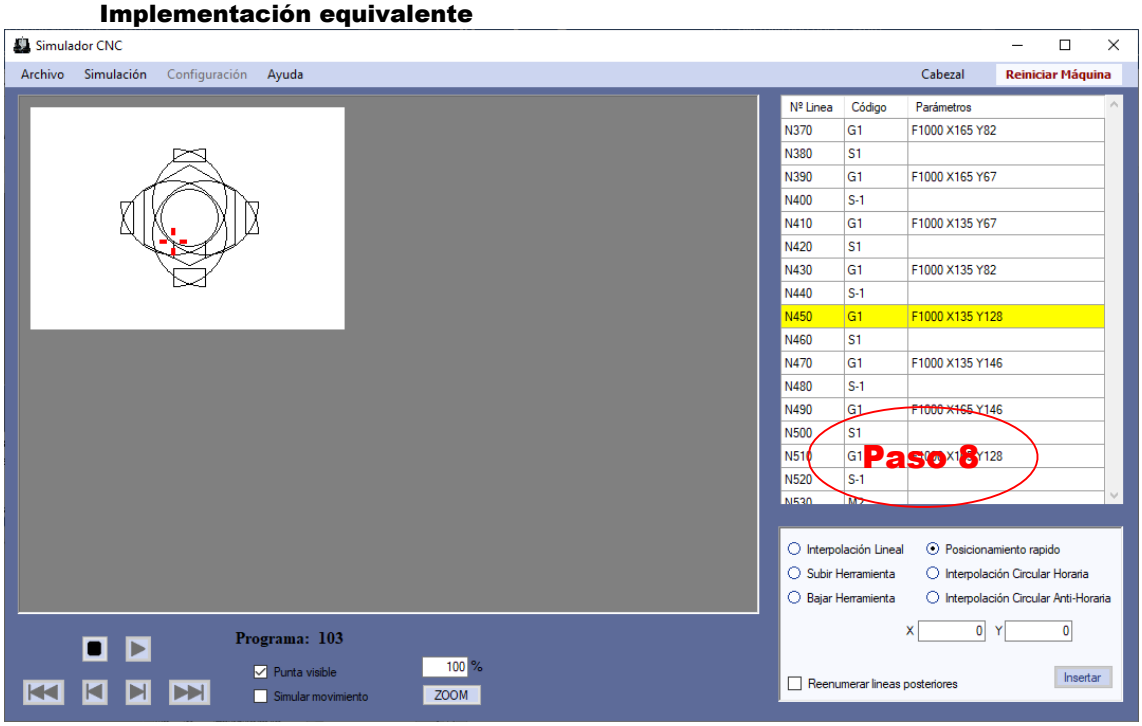
Implementación equivalente



Paso 7 Secuencia normal:

El sistema reanuda la ejecución del programa, si no se cumple el *Paso 4 de Excepciones* continúa.

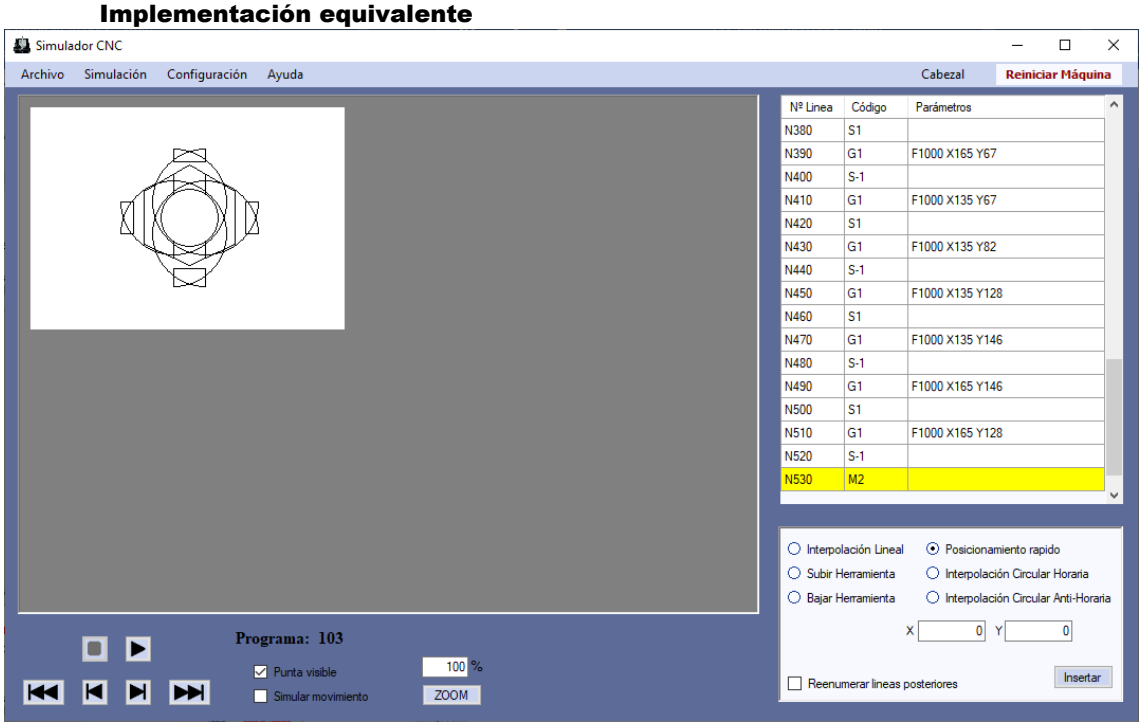
Implementación equivalente



Paso 8 Secuencia normal:

El sistema simula la ejecución de todo el programa hasta su fin.

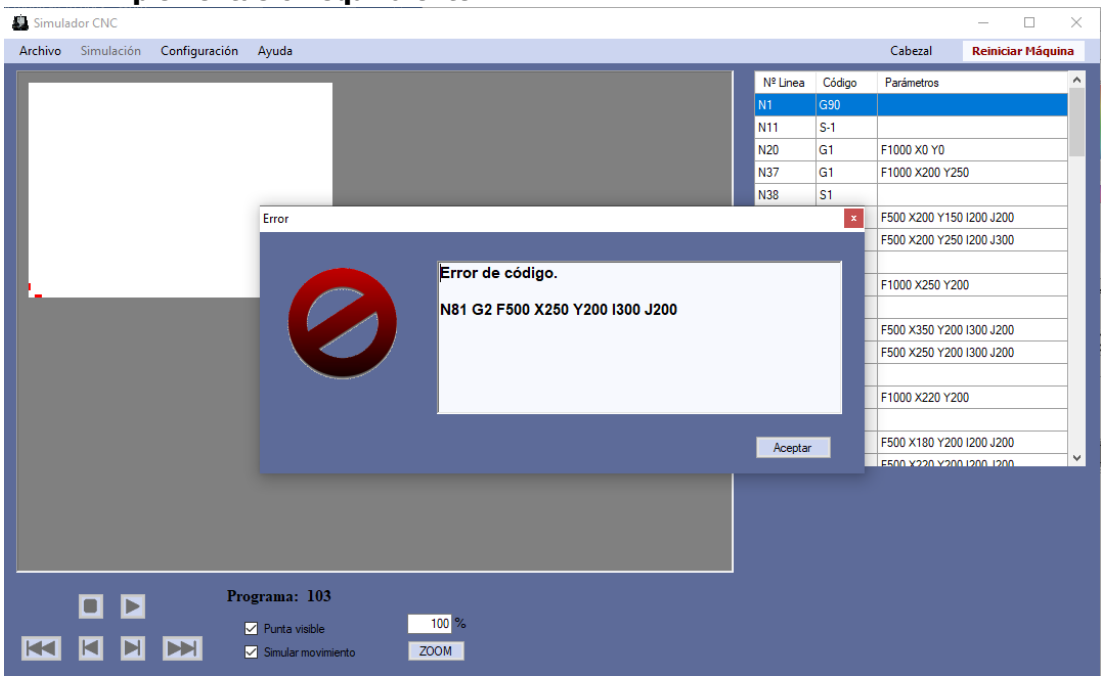
Implementación equivalente



Paso 2 Excepción:

Si el programa no es correcto se aborta la preparación del programa, se elimina y se reinicia la máquina virtual y termina el Caso de uso

Implementación equivalente

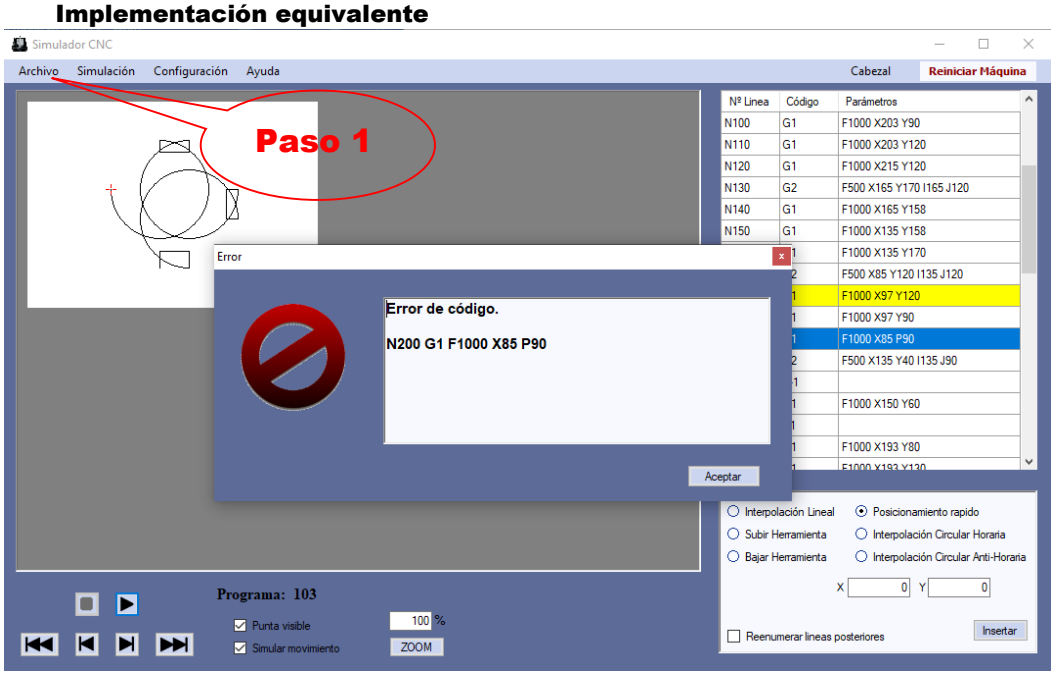


Paso 3 Excepción:

Si el programa no es correcto aborta el inicio de la simulación y:

- Si el usuario quiere cargar un nuevo programa *pasa al Paso 1 de Secuencia normal.*
- Si el usuario no carga un nuevo programa el Caso de uso termina.

Implementación equivalente

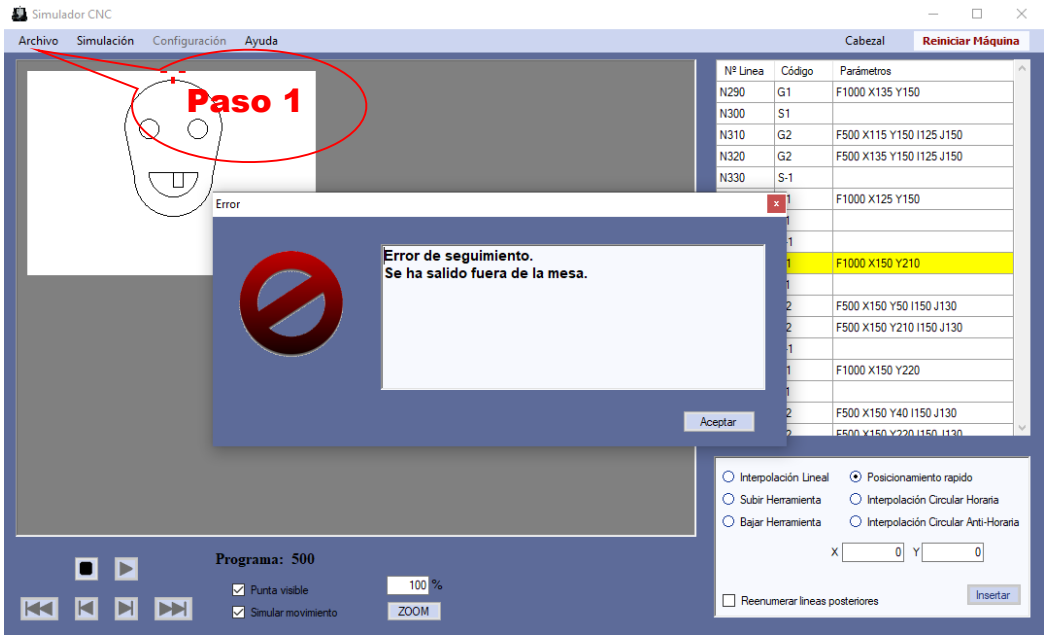


Paso 4 Excepción:

Si la simulación del sistema da un fallo en ejecución del programa CNC sobre la máquina virtual aborta la simulación y:

- Si el usuario quiere cargar un nuevo programa *pasa al Paso 1 de Secuencia normal*
- Si el usuario no carga un nuevo programa el Caso de uso termina.

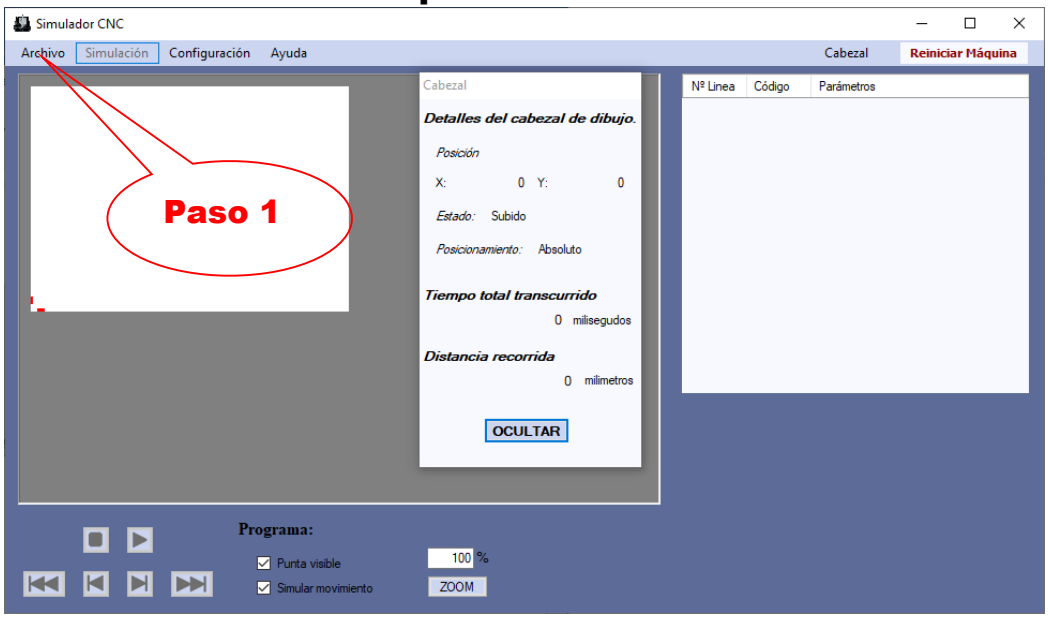
Implementación equivalente



○ Verificación del escenario 3:

Autores	Usuario
Objetivo asociado	Edición de un programa abierto
Descripción	El usuario abre un nuevo programa de mecanizado y lo ejecuta, en un momento intermedio parará el programa, realizará un cambio directo sobre la tabla de códigos, reanudará el programa hasta el final y finalmente guardará el programa y saldrá de la aplicación

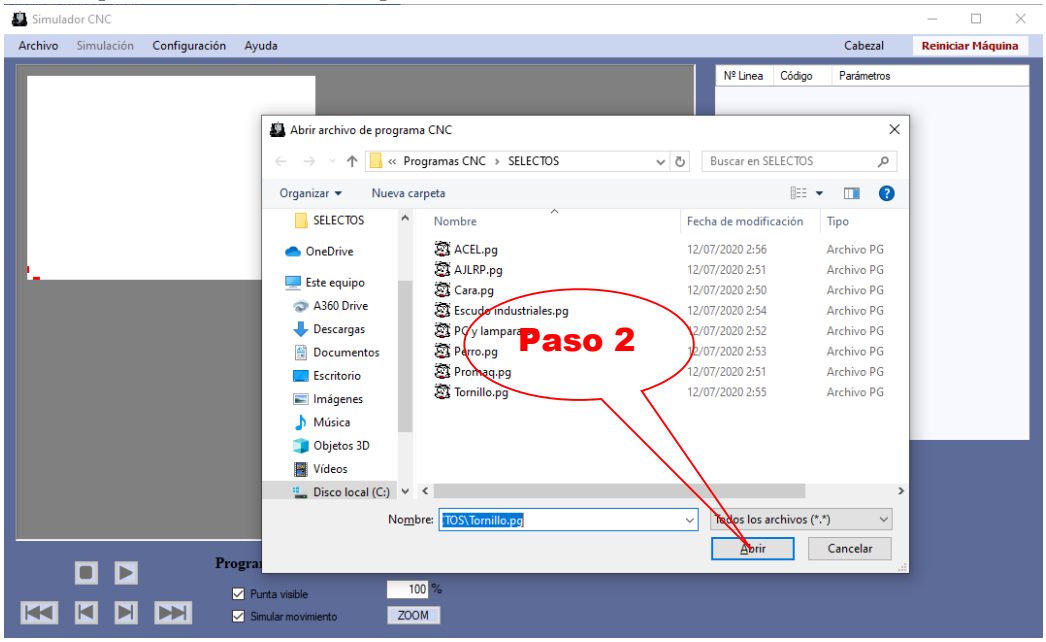
Estado inicial de la aplicación:



Paso 1 Secuencia normal:

El usuario carga el programa CNC.

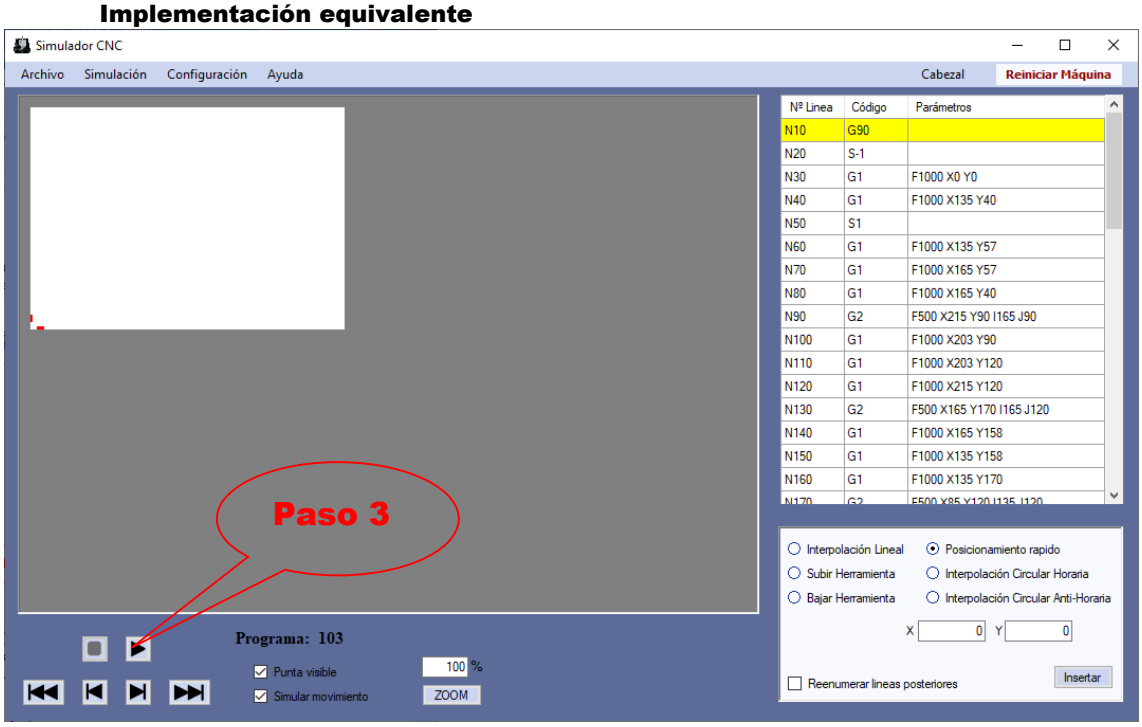
Implementación equivalente



Paso 2 Secuencia normal:

Si el programa es correcto se queda la aplicación preparada, si no es correcto *pasa al Paso 2 de Excepciones*.

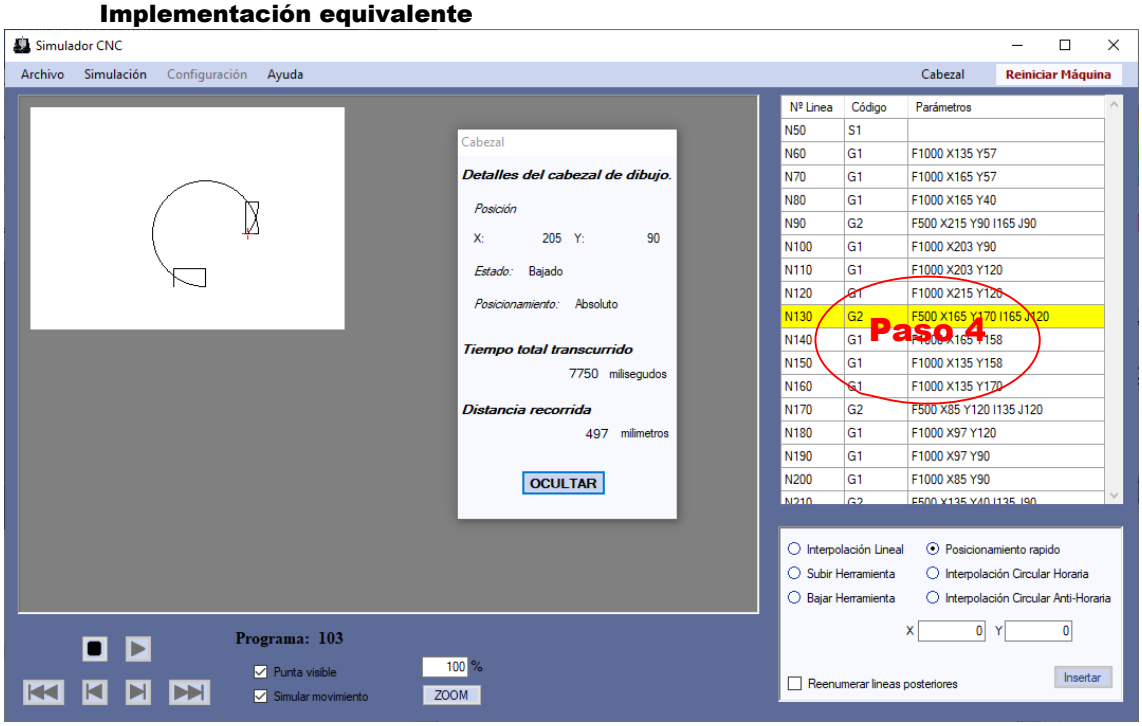
Implementación equivalente



Paso 3 Secuencia normal:

El usuario inicia la simulación del sistema, si no se cumple el *Paso 3 de Excepciones* continúa.

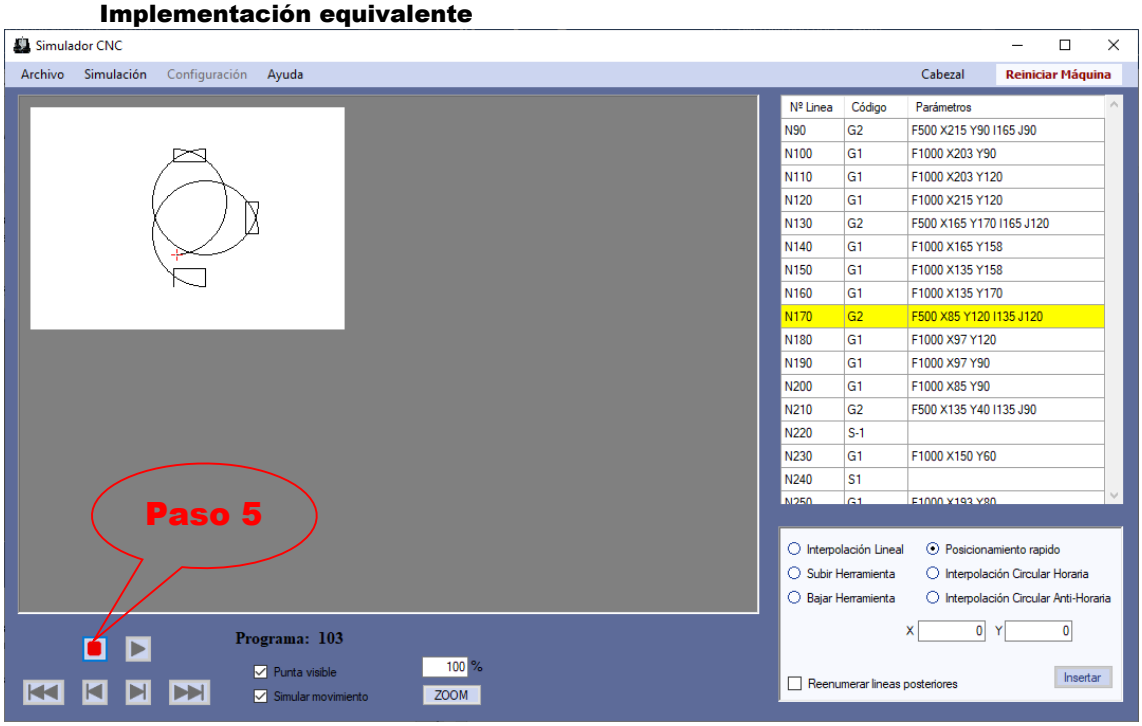
Implementación equivalente



Paso 4 Secuencia normal:

El sistema simula la ejecución, si no se cumple el *Paso 4 de Excepciones* continúa.

Implementación equivalente



Paso 5 Secuencia normal:

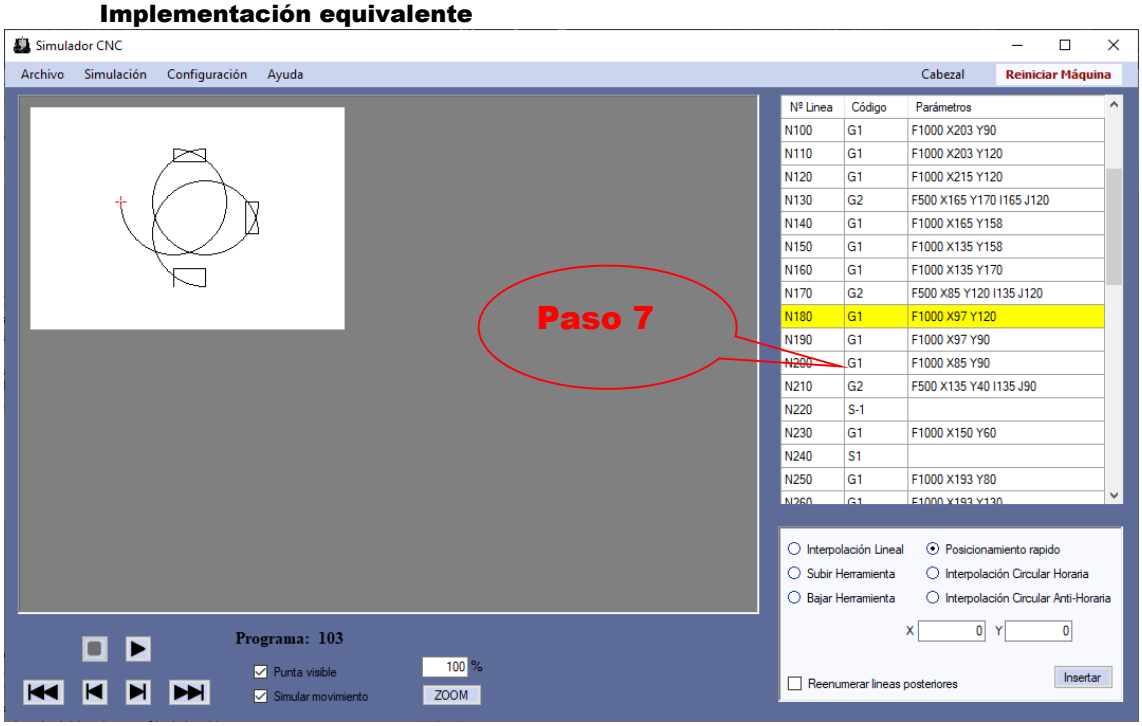
El usuario detiene la simulación del sistema.

Implementación equivalente

Paso 6 Secuencia normal:

El sistema termina de procesar el código CNC en ejecución y para la simulación.

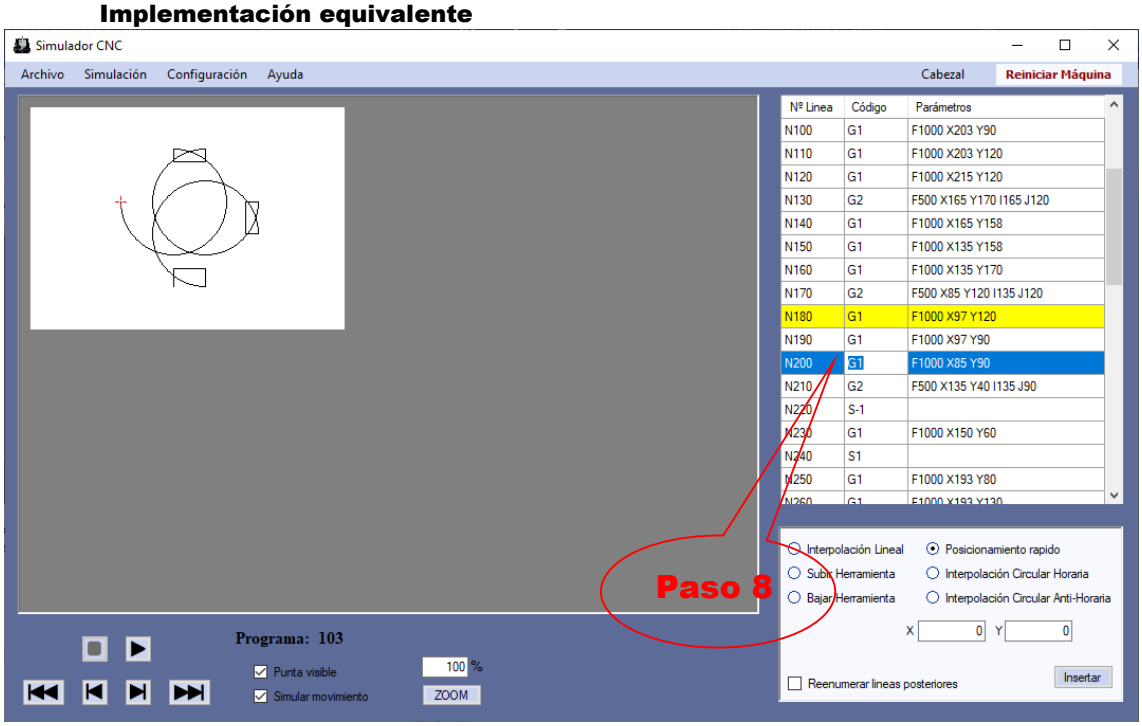
Implementación equivalente



Paso 7 Secuencia normal:

El usuario realiza una pulsación doble sobre una fila de la tabla.

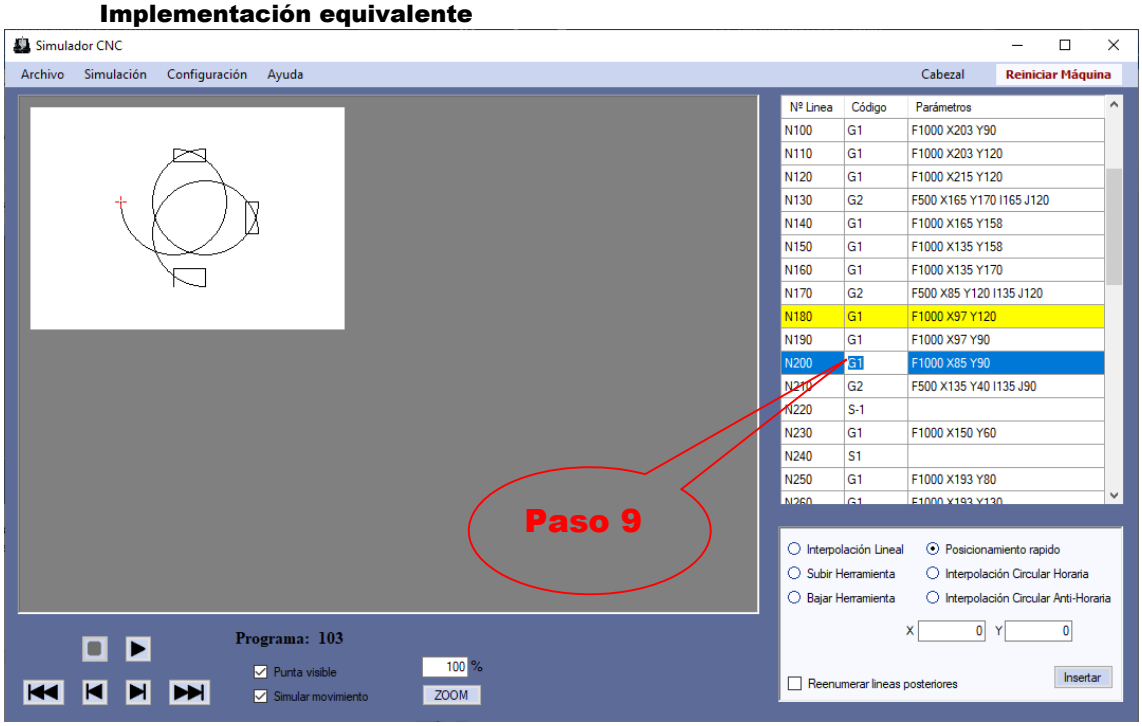
Implementación equivalente



Paso 8 Secuencia normal:

El sistema habilita la celda para edición

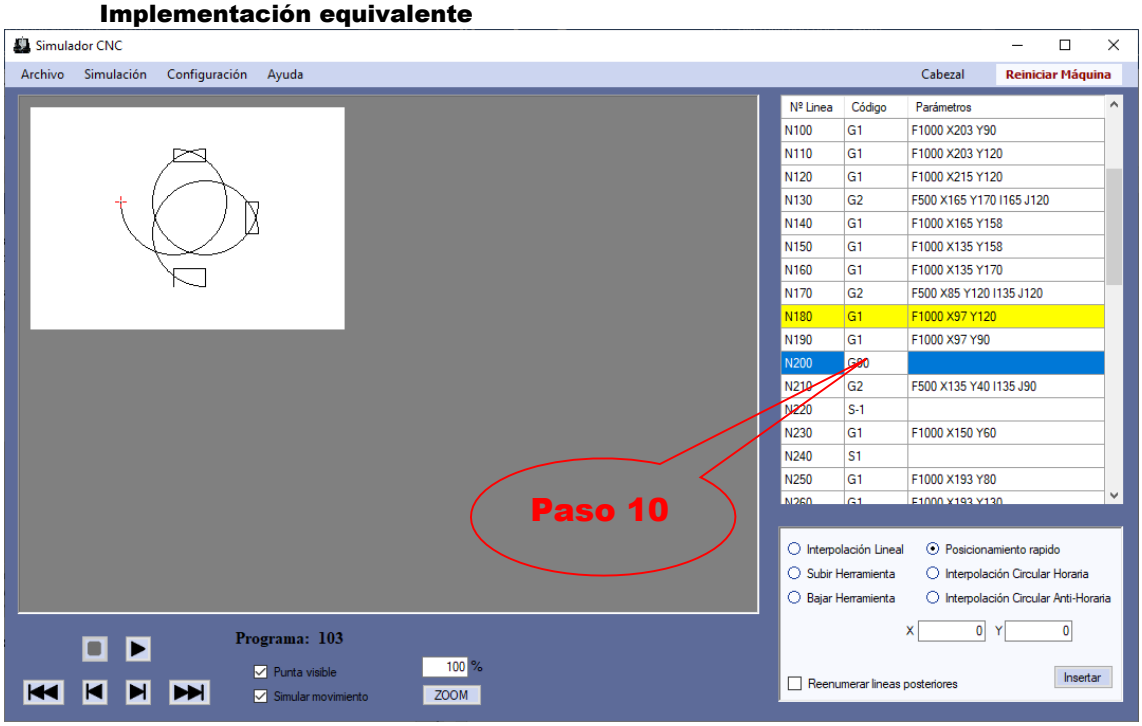
Implementación equivalente



Paso 9 Secuencia normal:

El usuario modifica la celda con el nuevo argumento

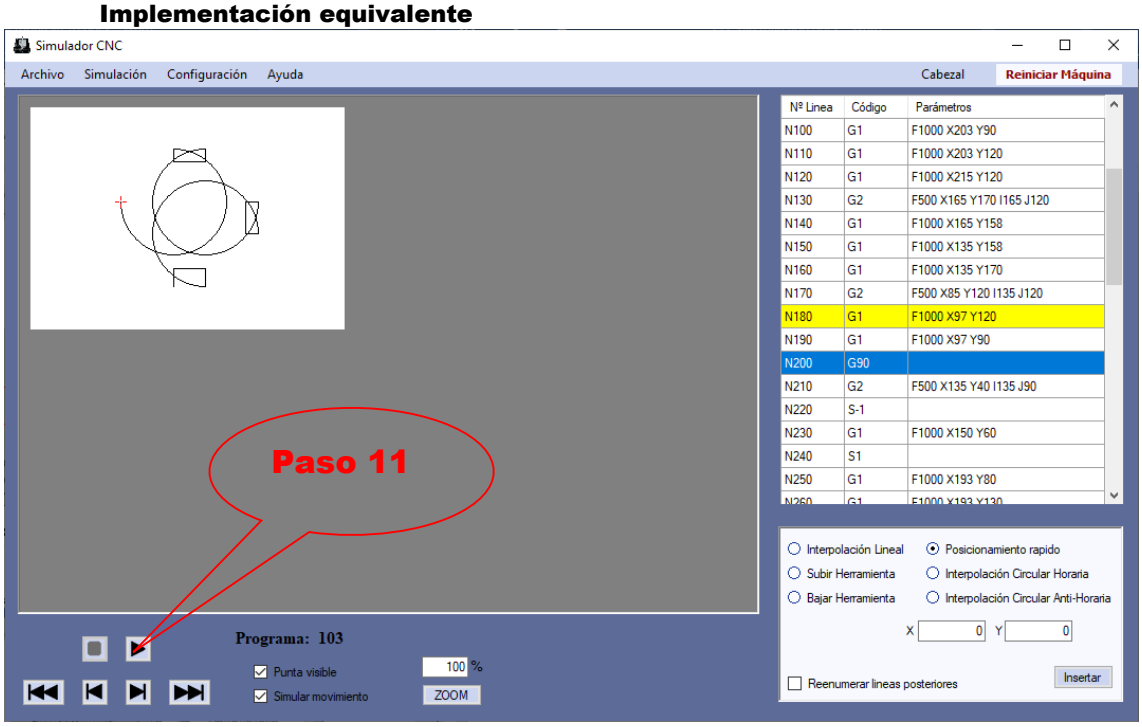
Implementación equivalente



Paso 10 Secuencia normal:

El sistema deshabilita la edición de la celda.

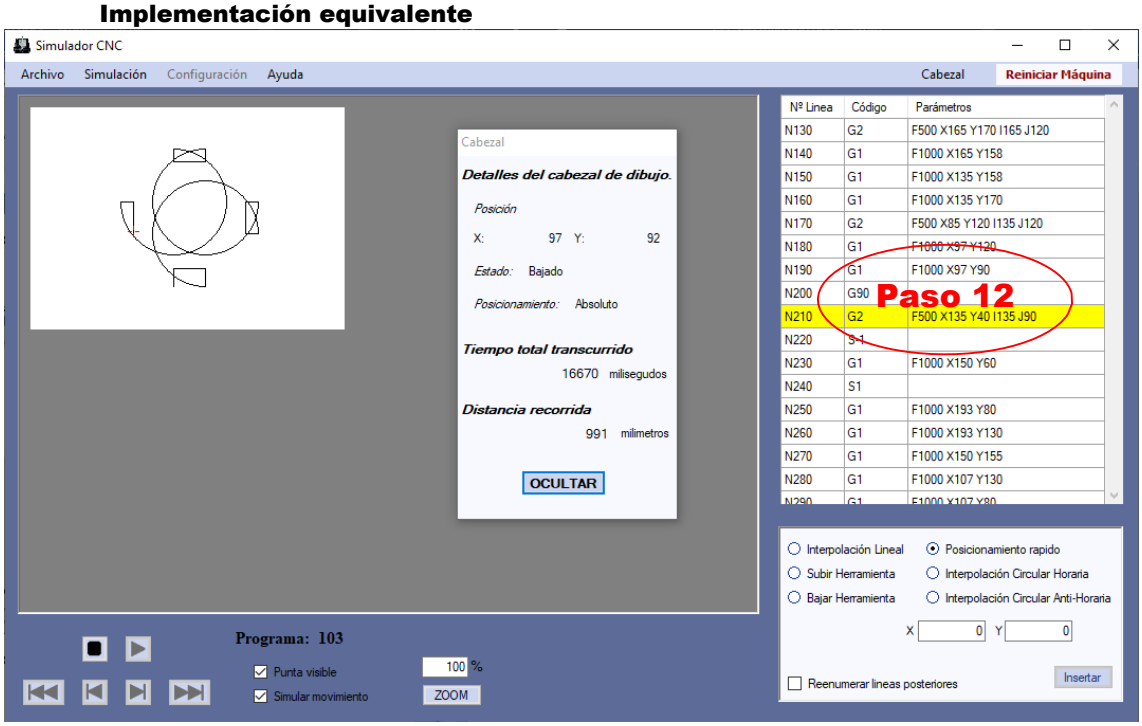
Implementación equivalente



Paso 11 Secuencia normal:

El usuario reanuda la simulación del sistema, si no se cumple el *Paso 11 de Excepciones* continúa

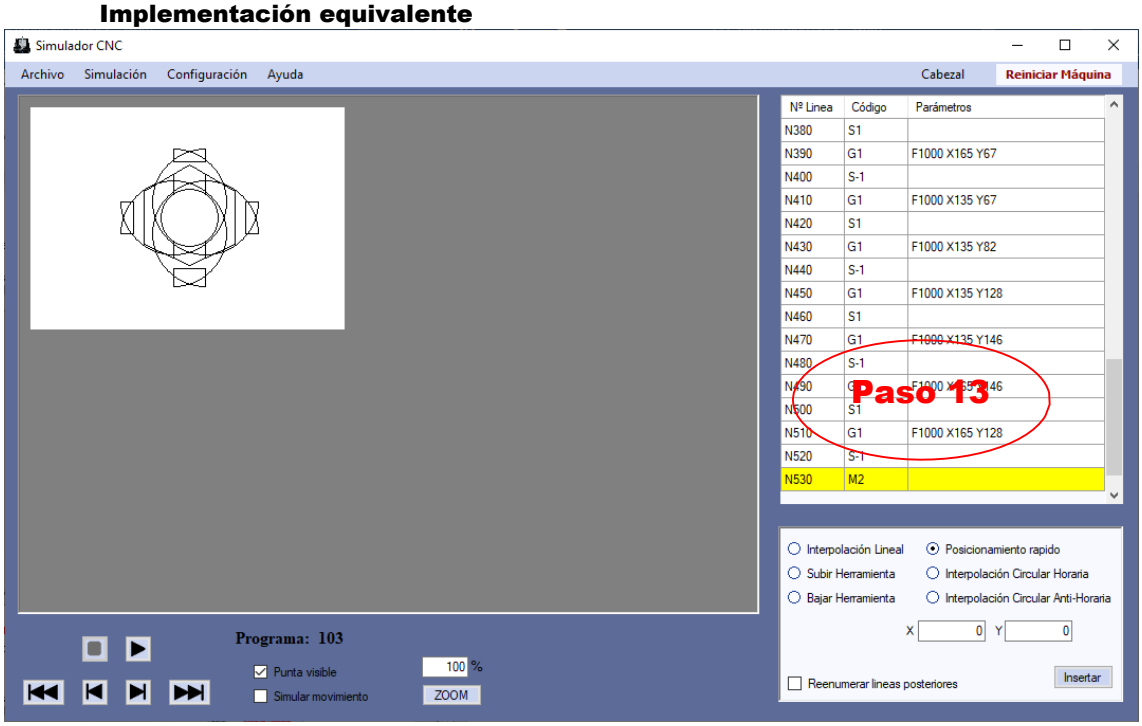
Implementación equivalente



Paso 12 Secuencia normal:

El sistema reanuda la ejecución del programa, si no se cumple el *Paso 4 de Excepciones* continúa

Implementación equivalente



Paso 13 Secuencia normal:

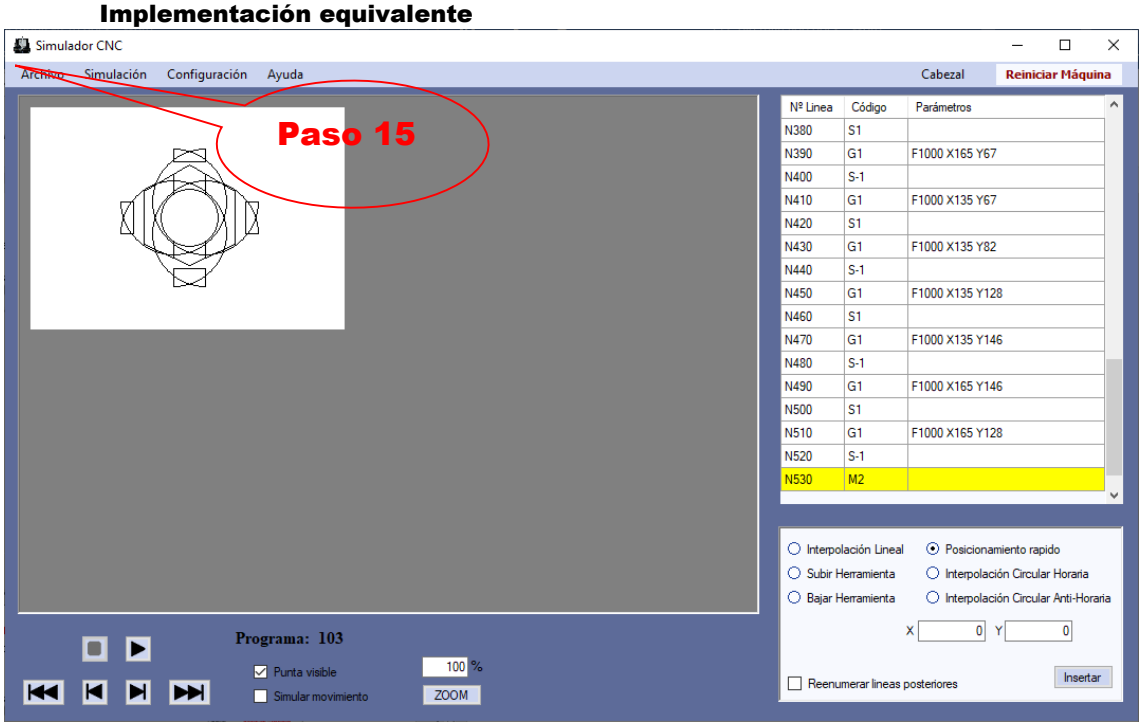
El sistema simula la ejecución de todo el programa hasta el final

Implementación equivalente

Paso 14 Secuencia normal:

El sistema para la simulación

Implementación equivalente



Paso 15 Secuencia normal:

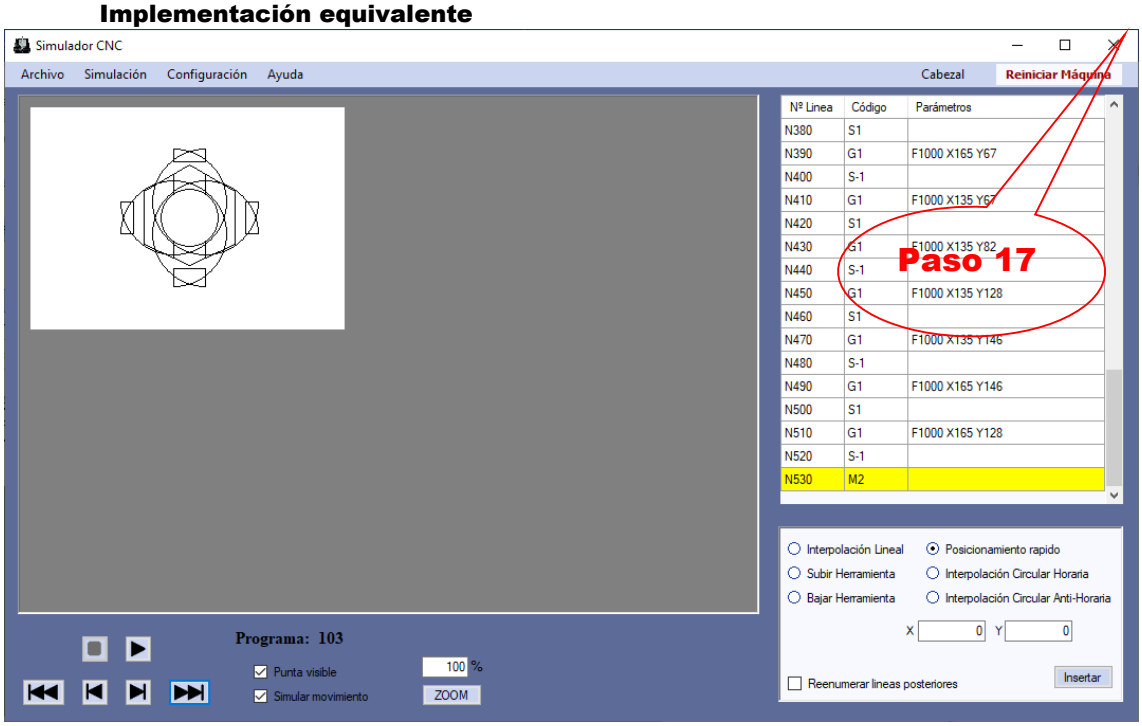
El usuario accede al menú de Guardar.

Implementación equivalente

Paso 16 Secuencia normal:

El sistema guarda el programa modificado en el archivo.

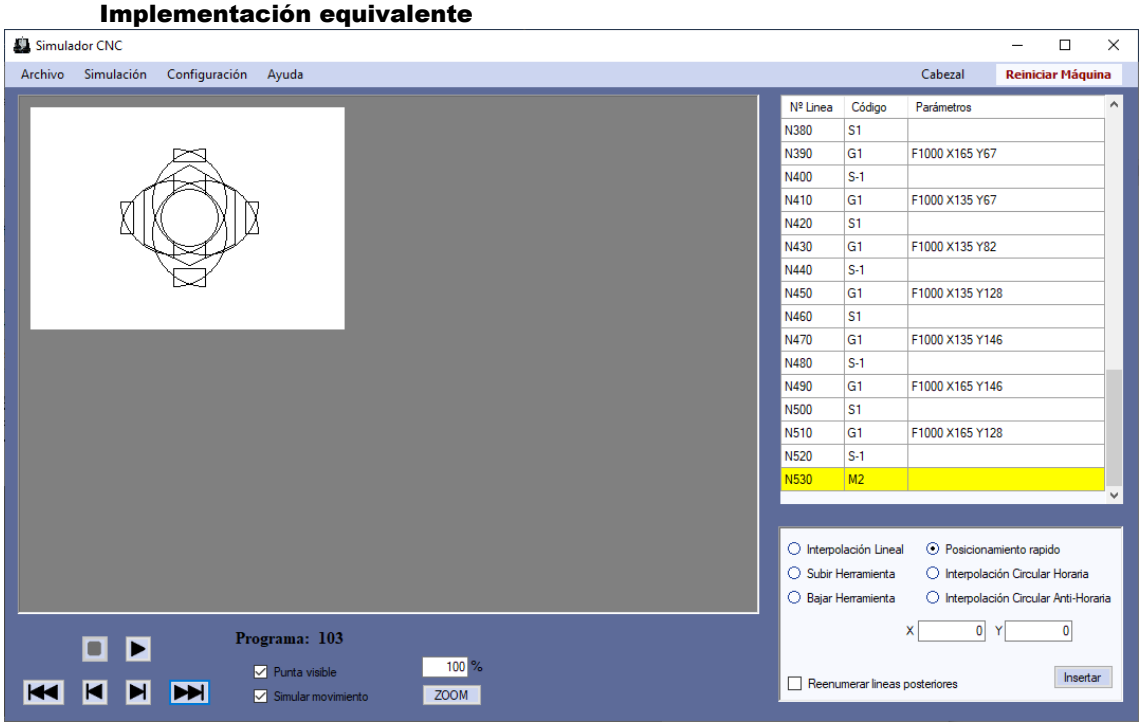
Implementación equivalente



Paso 17 Secuencia normal:

El usuario finaliza la aplicación terminando el Caso de uso.

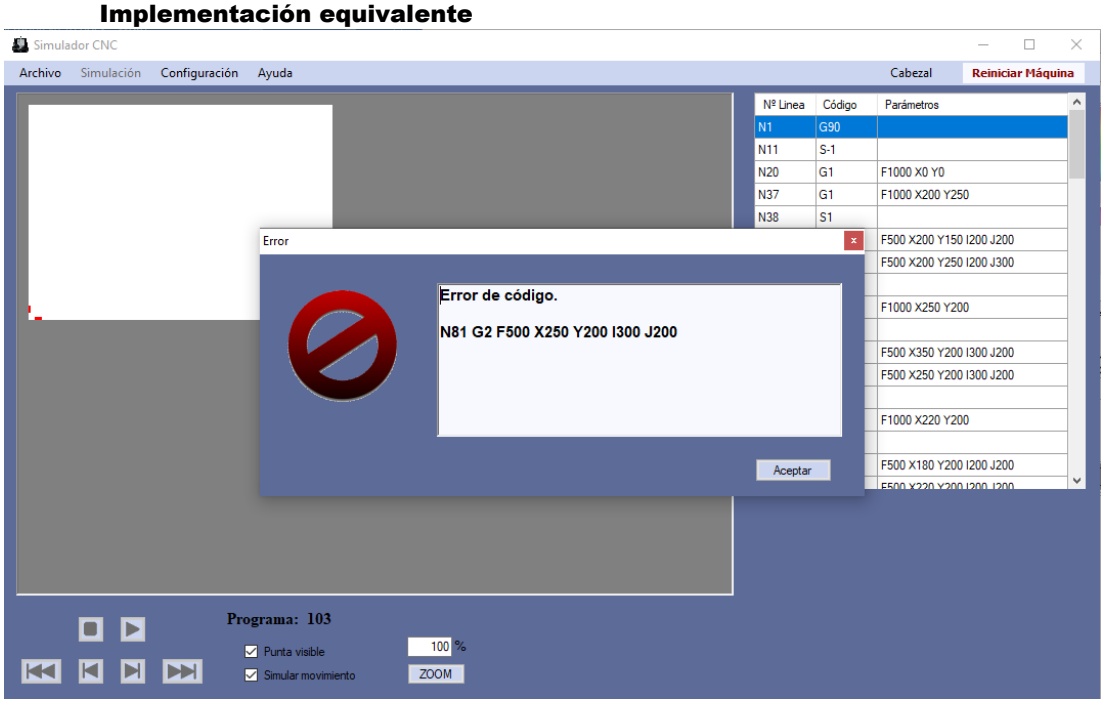
Implementación equivalente



Paso 2 Excepción:

Si el programa no es correcto se aborta la preparación del programa, se elimina y se reinicia la máquina virtual y termina el Caso de uso

Implementación equivalente

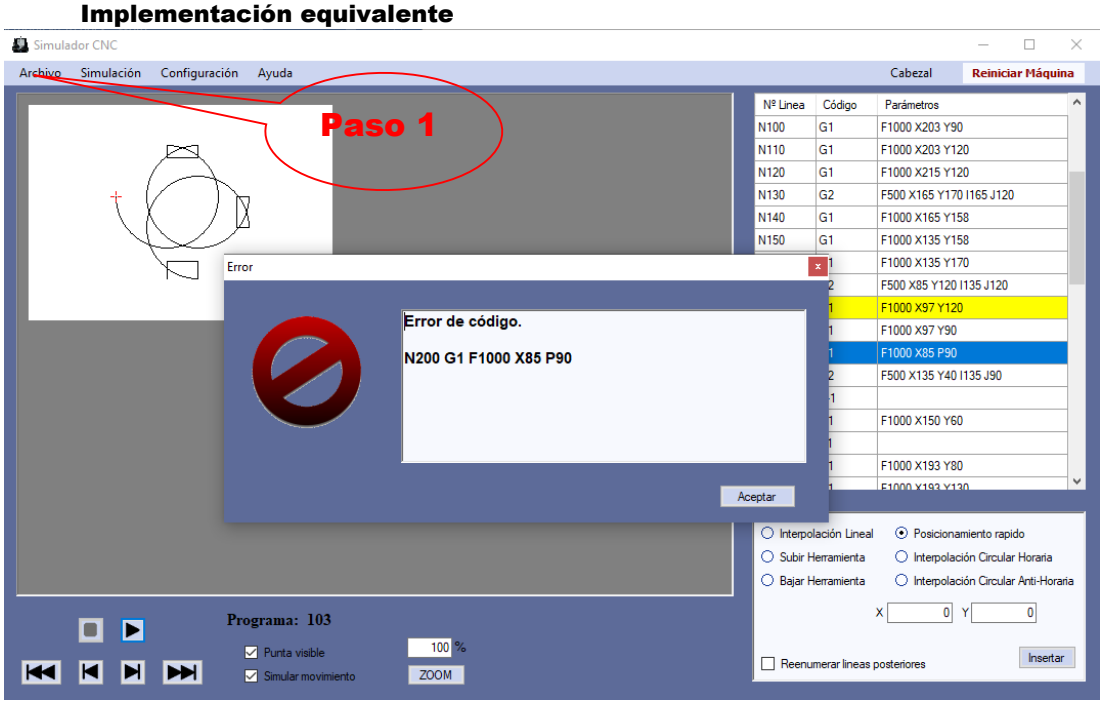


Paso 3 Excepción:

Si el programa no es correcto aborta el inicio de la simulación y:

- Si el usuario quiere cargar un nuevo programa *pasa al Paso 1 de Secuencia normal.*
- Si el usuario no carga un nuevo programa el Caso de uso termina.

Implementación equivalente

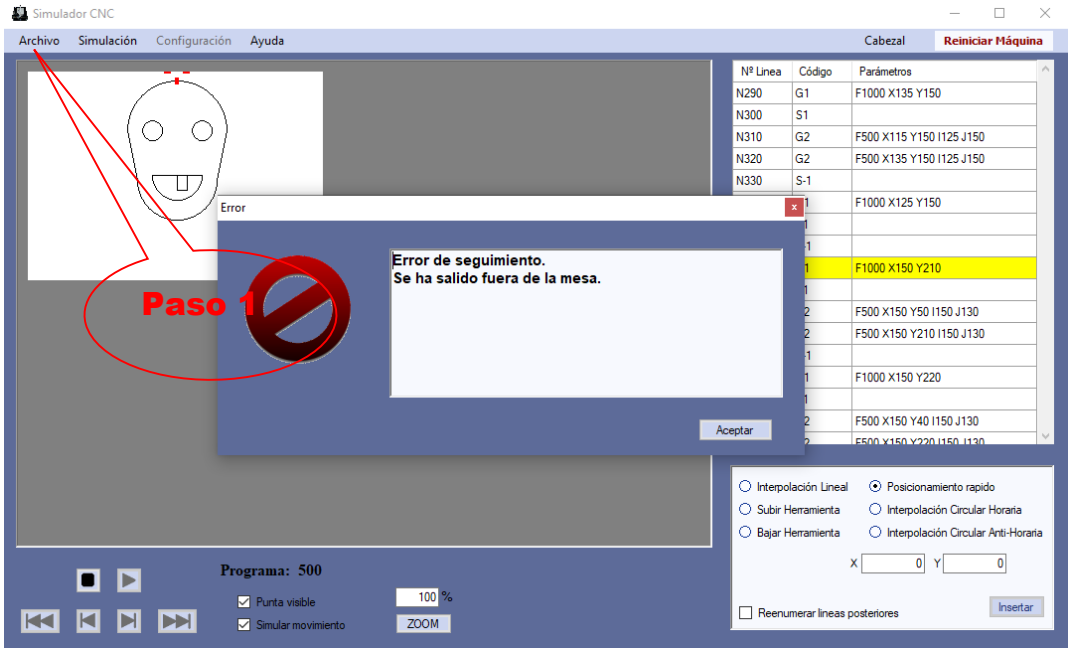


Paso 4 Excepción:

Si la simulación del sistema da un fallo en ejecución del programa CNC sobre la máquina virtual aborta la simulación y:

- Si el usuario quiere cargar un nuevo programa *pasa al Paso 1 de Secuencia normal*
- Si el usuario no carga un nuevo programa el Caso de uso termina.

Implementación equivalente



Paso 11 Excepción:

Si el programa no es correcto aborta el inicio de la simulación y:

- Si el usuario quiere cargar un nuevo programa *pasa al Paso 1 de Secuencia normal*
- Si el usuario quiere modificar nuevamente el programa *pasa al Paso 7 de Secuencia normal*
- Si el usuario no carga un nuevo programa o no modifica el actual el Caso de uso termina.

Implementación equivalente

Como se ha observado, la verificación a partir de los escenarios de los Casos de uso es muy fiable. Como en los ejemplos anteriores, que una vez verificados ya sabemos que por lo menos para estos Casos el programa funcionará correctamente.

Con este capítulo acabado finalmente ya tenemos un excelente programa totalmente funcional para que pueda ser usado por todos los que lo necesiten.

12.- Gestión de fallos.

Aunque esta aplicación es capaz de reconocer una gran variedad de errores distintos (48 tipos de errores diferentes) al usuario del programa solamente se le muestra unos cuantos.

Esto es debido a que la aplicación debe simular lo mejor posible la máquina real que representa y mostrar los mismos errores que esta. Además, esta aplicación está pensada para que los alumnos puedan practicar y crear programas en código G-code, por lo que la aplicación no puede ofrecer demasiados detalles sobre los errores que contiene el código, ya que en este caso no se esforzarían por encontrarlos.

Sin embargo, el mensaje de error que se muestra sí indica la línea en la que ha ocurrido el fallo, pero no comenta sobre que fallo se trata.

12.1.- Plantilla de fallos.

A continuación, se exponen las plantillas de Errores de todos los mensajes que se pueden mostrar al usuario de la aplicación y su número de error representativo en la aplicación.

1: "Error al procesar la línea " + txt1 + ".\nLinea de programa duplicada:\n\n" + txt2
2: "Error al procesar la línea " + txt1 + ".\nLinea no valida: \n\n" + txt2
3: "Error al procesar la línea " + txt1 + ".\nLinea no reconocida: \n\n" + txt2
4: "Error al procesar el fichero.\nNo existe código final \"M2\"."
5: "Error de código.\n\n" + txt1 + " " + txt2 + " " + txt3
6: “Error al procesar el programa.\nEl programa no es válido."
7: "El archivo no se puede abrir."
8: "Error de código.\nLa línea no posee numeración inicial.\n\n" + txt1 + " " + txt2 + " " + txt3
9: "Terminación no esperada del código."
10: "Falta código de terminación del programa."
11: "Error de seguimiento.\nDestino no encontrado."
12: "Error de seguimiento.\nSe ha salido fuera de la mesa."
14: "Error de código.\nSe ha detectado por lo menos una posición negativa dentro de una sección de código configurada en modo absoluto.\n\n" + txt1 + " " + txt2 + " " + txt3
20: "Error al procesar la velocidad de traslación.\nLa velocidad " + txt1 + " es menor que la velocidad mínima (" + txt2 + ")."
21: "Error al procesar la velocidad de traslación.\nLa velocidad " + txt1 + " es mayor que la velocidad máxima ("+txt2+")."
30: "Para poder configurar los parámetros de la máquina no puede tener ningún programa cargado."
40: "Error en el ancho de la mesa: " + txt1 + "\nEl valor tiene que ser un número entero positivo mayor que 0."

41: "Error en la altura de la mesa: " + txt1 + "\nEl valor tiene que ser un número entero positivo mayor que 0."
42: "Error en la velocidad de movimiento: " + txt1 + "\nEl valor tiene que ser un número entero positivo mayor que 0."
43: "Error en la posición inicial del puntero: " + txt1 +" X "+ txt2 +" Y\nLos valores tienen que ser números enteros positivos."
44: "Error en la posición inicial del puntero: " + txt1 + " X " + txt2 + " Y\nLos valores tienen que ser menores al tamaño de la mesa."
45: "Error en la velocidad inicial de traslación del puntero: " + txt1 + "\nEl valor tiene que ser un número entero positivo mayor que 0."
46: "Error en la velocidad máxima de traslación del puntero: " + txt1 + "\nEl valor tiene que ser un número entero positivo mayor que 0."
47: "Error en la velocidad mínima de traslación del puntero: " + txt1 + "\nEl valor tiene que ser un número entero positivo mayor que 0."
48: "Error en la velocidad inicial de traslación del puntero: " + txt1 + "\nEl valor tiene que ser mayor o igual que la velocidad mínima de traslación y menor o igual que la velocidad máxima de traslación."
49: "Error en la velocidad de movimiento: " + txt1 + "\nEl valor es demasiado grande, valor máximo "+txt2+"."

50: "Error al guardar el archivo de configuración.\nLos cambios en la configuración solo serán efectivos hasta que cierre el programa."
60: "Error al guardar archivo.\nEl nombre del archivo no es válido."
61: "Error al guardar archivo.\nEl archivo no se ha podido guardar."
70: "Error al insertar el código.\nDebe seleccionar en la tabla la fila anterior a donde se insertará el nuevo código CNC."
71: "Error al generar el código.\n"+txt1+" debe ser un número."
72: "Error al generar el código.\n"+txt1+" debe ser menor que la anchura de la mesa."
73: "Error al generar el código.\n" + txt1 + " debe ser menor que la altura de la mesa."
74: "Error al generar el código.\nNo se puede determinar el número de linea consecutivo."
75: "Error al generar el código.\n" + txt1 + " debe ser un número positivo mayor que 0."
76: "Error al generar el código.\n" + txt1 + " debe ser un número positivo."
80: "Debe darle un nombre al programa."
81: "El número inicial de lineas debe ser un número positivo."
82: "El zoom debe ser un número positivo mayor que cero."

Otro valor distinto: "Error desconocido."

13.- Conclusiones sobre el trabajo.

Finalmente, cabe decir que los pasos que se han tenido que ir desarrollando hasta llegar al final de todo este compendio han sido largos y pesados.

Sobre todo, ha habido alguna que otra dificultad con alguno de los parámetros gráficos y constructores de objetos que complementa el Visual Studio. Por lo que me he visto obligado varias veces a remodelar parte de la implementación.

Finalmente, este software incluye todos los objetivos que se pretendían implementar previstos al principio del documento. Posee una interfaz con todos los elementos, pero a su vez no es compleja sino simple.

Soporta dos modos de simulación, la simulación real y paso a paso. Permite la edición del programa CNC abierto. Permite la reconfiguración de la aplicación para adaptar los parámetros. Permite el almacenamiento del programa CNC abierto y exportación de imágenes del diseño. Es capaz de verificar la integridad del código del programa CNC abierto.

Por lo que se puede validar claramente que esta aplicación ha alcanzado la meta que se pretendía conseguir posibilitando a los alumnos realizar con mayor facilidad modelos programados por CNC mucho más complejos y creativos, mejorando su experiencia con la programación de este lenguaje.

Como aspecto futuro a implementar, se puede dotar al Simulador CNC de un interfaz por puerto serie o USB, para comunicarse y poder programar directamente sistemas CNC reales.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PLANOS

Simulador de un prototipo de Control Numérico por Computador

Autor:
Jordi Burriel Valencia

Director:
Rubén Puche Panadero

10 de Julio de 2020

TRABAJO FINAL DE GRADO

ÍNDICE

1.- Justificación.

Dado al tipo de trabajo desarrollado de Simulador CNC en este **Trabajo Final de Grado**, no se requiere el desarrollo de ningún tipo de plano.

Se han desarrollado varios tipos de diagramas en UML, no obstante, este tipo de diagramas no están normativizados por ninguna norma ISO, por tanto, se ha visto más adecuado no incluirlos en este documento, sino como anexos en el documento de la memoria del TFG.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PLIEGO DE CONDICIONES

Simulador de un prototipo de Control Numérico por Computador

Autor:
Jordi Burriel Valencia

Director:
Rubén Puche Panadero

10 de Julio de 2020

TRABAJO FINAL DE GRADO

ÍNDICE

1.- Introducción.

El pliego de condiciones es un documento contractual que fija Las condiciones técnicas, económicas y legales que se han de tener en cuenta en la ejecución y desarrollo del trabajo al que está asociado.

Las condiciones de este pliego se pueden agrupar en condiciones generales (comunes a cualquier pliego de condiciones) y particulares (específicas del propio trabajo).

2.- Condiciones generales.

En este apartado se desarrollan los objetivos del pliego de condiciones, así como la normativa vigente requerida.

2.1.- Objetivo del documento.

El objetivo de este documento se centra en definir cuatro tipos básicos de condiciones que se deben cumplir.

- **Condiciones técnicas**: Se corresponde con el tipo de trabajo a realizar, las características técnicas, controles y ensayos que hay que tener en cuenta durante la ejecución del trabajo y posteriormente para mantener la calidad del producto.
- **Condiciones facultativas:** Especifican los derechos y obligaciones de cada una de las partes (incluyendo a sus respectivos representantes) hábiles al comienzo de la ejecución del trabajo.
- **Condiciones económicas:** Especifican las garantías que correspondan al tipo de trabajo, la formación de precios, de los abonos y las posibles indemnizaciones por el incumplimiento del contrato.
- **Condiciones legales**: Especifican las condiciones que son reguladas por ley y que dependen de la tipología del trabajo contratado.

Con todo ello, se especifica en este documento detalladamente toda la normativa legal y de seguridad que estén sujetas a cumplir para el trabajo desarrollado.

Aquellos eventos imprevistos durante la ejecución o a la finalización del trabajo, que no estén recogidos en este pliego de condiciones, se concertará entre ambas partes (contratante y desarrollador)

Como en este caso, el objetivo de este Trabajo Final de Grado tiene un objetivo educacional, la definición en este pliego de las normas a cumplir solo tienen un carácter informativo para mostrar las capacidades del autor alumno del Grado de Ingeniería Eléctrica. Por tanto, estas normas aquí desarrolladas no tienen carácter vinculante real para el desarrollo del trabajo.

2.2.- Normativa y estandarización

Aunque el trabajo a desarrollar versa en el funcionamiento de máquinas de control numérico computarizado, al ser simulado no se requiere aplicar ninguna normativa específica respecto de la parte eléctrica o electrónica, dado que no existe (físicamente hablando) ninguna al ser una simulación. Por tanto, no existe ni se requiere tampoco reglamento necesario para la seguridad en este aspecto. Por tanto, no hay normativa vinculante.

No obstante, se han definido el uso de los siguientes estándares y recomendaciones de buenas prácticas:

- **ISO 6983-1:2009.** Sistemas de automatización e integración. Control numérico de máquinas. Formato de programas y definiciones.
- **ISO/IEC/IEEE 29148:2011.** Ingeniería de requisitos.
- **ISO/IEC/IEEE 12207:2008.** Desarrollo del ciclo de vida.
- **ISO/IEC/IEEE 15288:2008.** Desarrollo del ciclo de vida.
- **IEEE Std. 1233.** Vocabulario de ingeniería del software.

3.- Alcance del proyecto.

El programa Simulador CNC debe abarcar las siguientes funcionalidades.

3.1.- Modo de simulación.

Se implementarán dos modos de simulación:

- **Simulación real.** Simulación con movimiento controlado según la velocidad de traslación configurada en la herramienta.

- **Simulación por pasos.** Simulación sin movimiento real controlado cada paso por el usuario.

3.2- Modo de creación y edición.

Se implementarán las siguientes formas para que los usuarios puedan crear y modificar programas de mecanizado. Los cambios deberán ser directamente verificables y visibles en el momento de finalizar la modificación.

- **Edición directa**. Debe permitir la edición directa de los códigos G-code del programa.

- **Edición directa**. Debe permitir la edición directa de los códigos G-code del programa.

- **Inserción de nuevas instrucciones G-code.** Se debe facilitar la construcción de nuevas instrucciones por parte del usuario. Este tipo de inserción deberá generar código correcto siempre.

3.3- Persistencia de datos.

Se implementan opciones para que el alumno pueda almacenar el código de su programa resultante, e igualmente exportar el modelo creado en el área de mecanizado.

3.4- Detección y validación de fallos.

El simulador deberá verificar y gestionar los fallos de los programas de mecanizado.

Los fallos correspondientes a errores en el programa impedirán la ejecución del mismo más allá del fallo.

Los fallos detectados en la apertura del programa de mecanizado cancelaran su apertura.

Los fallos detectados en el transcurso de la simulación se mostrarán en dicho momento y requerirá al simulador volver al último estado anterior correcto.

3.5- Configuración de parámetros.

Se deberá poder personalizar los siguientes parámetros de la simulación:

- **Dimensiones del área de trabajo**.
- **Velocidad de la simulación**.
- **Parámetros del cabezal herramienta.** Velocidad máxima y mínima de desplazamiento. Altura del Cabezal. Tipo de posicionamiento inicial. Posicionamiento inicial

4.- Condiciones de ejecución y finalización.

4.1- Metodología de las propuestas técnicas.

Las propuestas técnicas descritas en la sección 3 deberán contener una descripción detallada de las metodologías propuesta para cada caso:

Se requerirá el uso de metodologías que promuevan los siguientes objetivos:

- Adopción de cambios a solicitud del cliente.
- Comunicación fluida entre los equipos de modelado y desarrollo.
- Generación de casos de prueba y validación.
- Atención a la calidad y al buen diseño.
- Definición de estándares o buenas prácticas de codificación.

4.2- Planificación.

Se presentará un programa de planificación temporal del proyecto acorde a las condiciones descritas en este pliego.

En esta planificación se definirán las tareas que sean necesarias en su respectivo momento temporal y el uso lógico de los recursos humanos en cada una de ellas.

4.3- Validación del funcionamiento.

Se desarrollarán los casos de uso necesarios para la verificación del correcto funcionamiento de todas las funcionalidades descritas en la sección 3 del pliego de condiciones.

Si hay funcionalidades añadidas al Simulador CNC que no figuran en este pliego de condiciones deberán ser igualmente validadas

4.4- Documentación.

Durante el desarrollo del proyecto y conforme se vaya completando cada estadio de la metodología seguida, la documentación generada en sus diferentes versiones (que se encuentre en la siguiente lista) será entregada para su revisión:

- Especificaciones de requisitos.
- Diagramas de modelado.
- Especificación técnica.
- Plan de planificación del proyecto.
- Plan detallado de los casos de prueba.
- Manuales de usuario.

En todos los documentos entregados constarán, como mínimo, el responsable, la fecha de modificación y la versión del documento.

4.5- Garantía de la solución.

Se facilitará, mediante acuerdo previo, una garantía de mantenimiento de la solución posterior a su entrega final.

5.- Condiciones económicas.

5.1- Mejoras sobre el proyecto.

En este caso, al ser un proyecto centrado en un Trabajo Final de Grado, no se contempla ninguna condición sobre la posibilidad de mejora del proyecto.

En otros casos distintos a este, cualquier mejora sobre el proyecto o su desarrollo deberá ser consultada previamente con el contratante.

En caso de llegar a un acuerdo entre desarrollador y contratante para la implementación de las mejoras, se renegociarán las condiciones del contrato.

5.2- Condiciones del pago del producto.

Se establecen a continuación las condiciones a seguir por todas las partes implicadas en el caso de los pagos realizados sobre el producto.

Si el pago se produce entre los cinco días naturales posteriores a la recepción del producto definitivo, inclusive con antelación a la entrega, el importe no sufrirá recargo alguno.

Si el pago se produce después de los cinco días naturales desde la entrega del producto definitivo, por cada mes natural el pago del producto sufrirá un recargo del 2% sobre el precio inicial hasta el sexto mes de recargo.

Superado el sexto mes de recargo, se considerará incumplimiento del pago y el desarrollador tendrá derecho a demandar judicialmente por dicho incumplimiento de pago.

En caso de que el contratante solicite el fraccionamiento del pago, el desarrollador establecerá el número de plazos e intervalo de tiempo entre cada pago fraccionado. Cada pago fraccionado estará condicionado de acuerdo a las mismas condiciones de pago especificadas previamente para el pago único.

6.- Condiciones legales.

6.1- Condiciones del contrato.

El contrato deberá realizarse por escrito, cumplir todos los requisitos legales y estar firmado por todas las partes implicadas. Cualquier otra forma de contrato distinta a la escrita será inválida en sí.

En este contrato se deberán especificar, como mínimo, los siguientes elementos:

- Precio inicial del desarrollo del producto y su coste unitario.
- Tarifas adicionales.
- Clausulas a cumplir por cada una de las partes.

6.2- Arbitraje y jurisdicción.

En el caso de producirse desentendimiento entre alguna de las partes, antes de proceder por vía judicial se intentará llegar a consenso mediante arbitraje proporcionado por un tercero independiente con nivel técnico suficiente.

Si después del arbitraje sigue sin haber consenso, se podrá recurrir a los tribunales de justicia para resolver el conflicto.

6.3- Impuestos.

Para la entrega de la solución se aplicarán los correspondientes impuestos requeridos por ley sobre este, que se encuentren en vigor en la fecha de entrega de la solución.

6.4- Rescisión del contrato.

El contrato se podrá rescindir en las siguientes situaciones:

- Retraso de la entrega de la solución, con un tiempo mayor a un mes natural desde la fecha de finalización del contrato. Siempre y cuando dicho retraso no se deba a causas de fuerza ajena inevitables.
- Mediante acuerdo entre el desarrollador y contratante.

7.- Derechos y deberes del desarrollador.

Debe conocer y cumplir todas las leyes referentes a su actividad empresarial.

Deberá realizar las pruebas necesarias para asegurarse de todos los elementos y el sistema en general funcione correctamente ofreciendo una buena calidad del proceso finalizado.

El contratante no podrá reclamar por retrasos producidos en el proceso de fabricación por causas justificadas ajenas al desarrollador. En el caso de que estos retrasos no se justifiquen el contratante deberá abonar un importe del 10% del importe de fabricación.

La solución cumplirá con los requisitos especificados en el proyecto y solo se podrán modificar con la aprobación del desarrollador.

El contratante deberá facilitar al desarrollador todos los elementos e información necesaria para una realización eficiente y rápida de la solución. Además, deberá entregarle por escrito todas las especificaciones requeridas en el proyecto y cualquier otro material relevante para facilitar el desarrollo de la solución.

El trabajo del desarrollador finaliza después de validar el correcto funcionamiento de la solución en todos los casos funcionales requeridos por el contratante, incluyendo funcionalidades añadidas por el desarrollador.

No será competencia del desarrollador dar soporte posterior a la entrega de la solución final sin acuerdo previo con el contratante.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PRESUPUESTO

Simulador de un prototipo de Control Numérico por Computador

Autor:
Jordi Burriel Valencia

Director:
Rubén Puche Panadero

10 de Julio de 2020

TRABAJO FINAL DE GRADO

ÍNDICE

1.- Introducción.

El siguiente documento se centra en el desarrollo de un presupuesto considerando todos los costes asociados al trabajo.

En este caso el presupuesto refleja las capacidades del autor, como alumno de grado, de desarrollar en su futuro profesional presupuestos reales. Por tanto, lo que refleja este presupuesto es el hipotético caso del análisis económico de este trabajo como si hubiera sido proyectado por un profesional del campo de la Ingeniería Eléctrica.

En el presupuesto se estiman tanto el coste de los recursos materiales necesarios para el desarrollo del trabajo, como el coste los recursos humanos teniendo en cuenta su implicación dentro de la planificación temporal del trabajo. Además, se tienen en cuenta otros tipos de costes, como los de amortización y de mantenimiento.

La suma total de todos los costes muestra el coste total del trabajo. Por tanto, con el desarrollo de este presupuesto se puede analizar la viabilidad del trabajo de cara a la economía disponible para su desarrollo

2.- Costes de los recursos materiales.

En este apartado se muestran los costes de los recursos de componentes y software necesarios para desarrollar el proyecto.

Los costes que aparecen sobre los distintos componentes se corresponden con los precios de venta al público en la fecha actual en que se ha desarrollado este presupuesto (Julio 2020)

2.1.- Recursos de componentes.

Dado que el simulador se puede desarrollar íntegramente con aplicaciones de software, los únicos recursos necesarios de componentes se centran en las características técnicas del puesto de ordenador requeridas por el entorno de desarrollo utilizado, que en este caso es el Visual Studio 2019, y por las necesidades del desarrollador en su puesto.

Tabla 1. Costes de los recursos de componentes del trabajo

Recursos de componentes				
ID	**Elemento**	**Cantidad (Uds)**	**Precio Unitario (€)**	**Precio (€)**
Co1	**Torre PC alta gama**	1	**1200.00**	**1200.00**
Co2	**Monitor 120Hz 23”**	2	**128.95**	**257.90**
Co3	**Teclado mecánico**	1	**39.95**	**39.95**
Co4	**Ratón 4000DPI**	1	**15.99**	**15.99**
	Precio total componentes (€)			**1384.89**

2.2.- Recursos de software.

Los recursos de software se corresponden con el coste de todas las aplicaciones y licencias necesarias para el desarrollo del trabajo.

Tabla 2. Costes de los recursos de software del trabajo

Recursos de software				
ID	**Elemento**	**Cantidad (Uds)**	**Precio Unitario (€)**	**Precio (€)**
So1	**Windows 10 Professional 64Bits**	1	**144.99**	**144.99**
So2	**Microsoft Office 365**	1	**69.00**	**69.00**
So3	**Software CASE MOSKitt**	1	**0.00**	**0.00**
So4	**Visual Studio 2019 Pro**	1	**399.99**	**399.99**
	Precio total software (€)			**613.98**

2.3.- Costes de todos los recursos materiales.

El coste total de todos los recursos necesarios tanto en componentes como en software será el siguiente:

Tabla 3. Costes totales de los recursos

Coste total de los recursos materiales		
ID	**Elemento**	**Precio (€)**
Re1	**Recursos de componentes**	**1384.89**
Re2	**Recursos de software**	**613.98**
Precio total recursos (€)		**1998.87**

3.- Costes de la mano de obra.

En este apartado se muestran los costes de la mano de obra que se requiere para desarrollar el producto.

Este coste viene calculado por el precio de la mano de obra por hora multiplicado por el número de horas trabajadas por el personal implicado.

3.1.- Planificación de recursos humanos.

Para el desarrollo del proyecto es necesaria la siguiente mano de obra correspondiente a los perfiles profesionales que se muestran en la Tabla 4.

Tabla 4. Mano de obra requerida para el desarrollo del trabajo

Recursos humanos		
ID	**Descripción**	**Cantidad (Horas)**
Hu1	**Dirección**	150
Hu2	**Analista y diseñador**	170
Hu3	**Programador**	200

La asignación de horas es una estimación desarrollada en base al ciclo de vida del producto planteada para este trabajo. Esta distribución de horas se muestra en el siguiente diagrama de Gantt de la Ilustración 1.

Ilustración 1. Diagrama de Gantt

3.2.- Costes total de la mano de obra.

El coste total de la mano de obra se muestra a continuación en la Tabla 5.

Hay que tener en cuenta que el coste por mantenimiento se desarrolla por una cuantía aparte de la mano de obra para el desarrollo del proyecto previo al mantenimiento.

Tabla 5. Coste total de mano de obra del trabajo.

Coste total de la mano de obra				
ID	**Descripción**	**Cantidad (Horas)**	**Precio por Hora (€)**	**Precio (€)**
Hu1	**Dirección**	150	**50.00**	**7500.00**
Hu2	**Analista y diseñador**	170	**20.00**	**3400.00**
Hu3	**Programador**	200	**12.00**	**2400.00**
	Precio total mano de obra (€)			**13300.00**

4.- Costes adicionales.

Además de los recursos humanos y los recursos materiales, existe una serie de costes que hay que tener en cuenta además dentro del presupuesto.

- **Costes de mantenimiento del producto**. Una vez finalizada la aplicación, el desarrollador ofrece un tiempo de mantenimiento (opcional) por el que se concuerda un precio estipulado con el comprador. En este presupuesto se ofrece un coste adicional de mantenimiento de 3 meses por una **cuantía fija de 3000€**

- **Gastos generales**. En este caso existen además otros costes generales necesarios y que deben tenerse en cuenta en el presupuesto, como son los gastos de electricidad, de agua, de teléfono, de aclimatación, etc. Como la mayoría de estos gastos son generados por la mano de obra, el coste de estos gastos se calcula sobre un **15%** respecto del **coste total de mano de obra.**

- **Margen de beneficio.** Como en cualquier compañía, la finalidad de desarrollar el trabajo es conseguir un beneficio por ello. Este beneficio depende de muchos factores (demanda del producto, competencia, etc.). En este caso se ha considerado un **margen de beneficio del 10%** respecto al coste total de desarrollo sin aplicar impuestos.

5.- Coste total del trabajo.

Finalmente, el precio total del trabajo se obtiene al sumar los costes de recursos materiales, de mano de obra, gastos generales, beneficio e impuestos (21% I.V.A.).

Tabla 6. Coste total del trabajo

Coste total del trabajo		
ID	**Elemento**	**Precio (€)**
Re0	Coste total de los recursos materiales	1998.87
Hu0	Coste total de la mano de obra	13300.00
Ge0	Gastos generales (15% Hu0)	1995.00
Precio total desarrollo (€)		**17293.87**
Be0	Margen de beneficio (10%)	1729.39
Precio total trabajo (€)		**19023.26**
IVA	Impuesto de Valor Añadido (21%)	3994.89
Precio total trabajo + IVA (€)		**23018.15**

Por tanto, el **presupuesto total** del presente trabajo asciende a **23018.15 euros.**

Como alternativa, se puede ofrecer un presupuesto añadiendo los costes por mantenimiento antes de aplicar los impuestos.

Tabla 7. Coste total del trabajo con 3 meses de mantenimiento

Coste total del trabajo con 3 meses de mantenimiento		
ID	**Elemento**	**Precio (€)**
Precio total trabajo (€)		**19023.26**
Ma0	Mantenimiento del producto (3 meses)	3000.00
Precio total trabajo (€)		**22023.26**
IVA	Impuesto de Valor Añadido (21%)	4624.89
Precio total trabajo + IVA (€)		**26648.15**

Si el contratante opta por la solución con mantenimiento, el **presupuesto total** asciende a **26648.15 euros.**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXOS

Simulador de un prototipo de Control Numérico por Computador

Autor:
Jordi Burriel Valencia

Director:
Rubén Puche Panadero

12 de Julio de 2020

TRABAJO FINAL DE GRADO

ÍNDICE

ÍNDICE (continuación)

ÍNDICE (continuación)

ANEXO 1.- Especificación de requisitos (ISO/IEC/IEEE 29148).

1. Introducción.

1.1 Propósito.

El propósito de este software es la simulación completa del funcionamiento de un sistema de máquina de Control Numérico Computarizado controlado mediante programas de mecanizado escritos en lenguaje interpretado G-code, así como la posible edición del programa desde la aplicación.

Este software tiene como finalidad el uso educativo con destino a mejorar el aprendizaje al alumnado.

1.2 Ámbito.

- La aplicación se identificará con el nombre Simulador CNC.

- La aplicación debe estar preparada para:

  Reconocer e interpretar una subversión del lenguaje G-code conforme al estándar ISO 6983.
  Desarrollar el resultado de cada uno de estos códigos sobre un área de mecanizado destinado a ello.
  Recrear una simulación real basada en el tiempo.
  Permitir la ejecución del programa de mecanizado sin necesidad de una simulación real.
  Permitir la edición del programa de mecanizado, así como la validación de errores en estos.
  Posibilidad de almacenamiento del resultado obtenido y del programa de mecanizado.

- El benefició de esta aplicación es el de ofrecer un mejor sistema de pruebas y de desarrollo de programas de mecanizado para alumnado universitario y de formación profesional.

1.3 Definiciones, acrónimos y abreviaturas.

CNC: Control Numérico por Computador

Códigos interpretados G-code: Secuencias de comandos con parámetros definidos sobre situación espacial y/o traslación.

Máquina herramienta: Máquina de propósito industrial, comprendida por una zona de trabajo o área de mecanizado y un brazo móvil con una o varias herramientas.

Estándar ISO 6983: Es el estándar en vigor que define la forma y función de cada Código CNC.

Área de mecanizado: Zona bidimensional donde se representa dinámicamente el resultado.

Simulación: Imitar o recrear sobre un medio virtual (no físico) un sistema o acción del mundo real.

2. Descripción General.

2.1 Perspectiva del Producto.

La aplicación del Simulador CNC se ha creado con el propósito de realizar funciones propias de una máquina herramienta del tipo CNC. Se pretende que esta aplicación sea capaz de interpretar cada uno de los códigos G-code de un programa suministrado y realizar los diversos trazos de mecanizado que contiene el programa (puntos, líneas, circunferencias, …) con el fin de poder desarrollar y depurar programas de mecanizado para sistemas CNC.

2.2 Funciones del Producto.

Las funciones que realiza la aplicación las podemos catalogar de la siguiente forma.:

Funciones de Simulación: Configuración temporal para reproducir la situación del sistema y los movimientos del cabezal en un espacio de tiempo determinado.

Funciones de verificación: Implementación de un sistema interno de comprobación del funcionamiento anómalo de la máquina y de la codificación del programa de mecanizado.

Funciones de información: Visualización del estado del sistema, del programa de mecanizado y del cabezal mediante los paneles y tablas oportunos.

Funciones de configuración: Implementación de un sistema de personalización de la aplicación con posibilidad de memoria.

Funciones de carga y almacenamiento: Implementación que permite acceder al sistema de ficheros para abrir o almacenar el programa de mecanizado o el área de trabajo.

Funciones de edición: Implementación de un sistema que interacciona con el usuario permitiendo la modificación dinámica del programa en ejecución.

2.3 Características del usuario.

Los usuarios que han de controlar y configurar el programa es mayormente alumnado procedente de entornos universitarios o de formación profesional, por lo que se espera que tengan un nivel mínimo que sea el básico para uso de aplicaciones informáticas.

Estos usuarios son los que van a configurar la aplicación por lo que la parte de configuración del simulador debe estar centrada en las propiedades de los elementos de la máquina real que queremos configurar (tamaño del área de trabajo, velocidad del cabezal herramienta, etc.).

2.4 Restricciones generales.

No se contempla incluir en el software del Simulador CNC funciones de seguridad o de control de acceso a la información.

Tampoco se contempla actualmente la interoperación con otros sistemas o aplicaciones por lo que no es necesario ningún interfaz de comunicaciones.

Para un funcionamiento mínimo del simulador CNC sin uso de simulación real, se requerirá disponer de un ordenador con los siguientes requisitos:

- Procesador de la familia Intel dual-Core o superior (Incluyendo procesadores compatibles)
- 50 MB de espacio libre en el disco duro.
- 1GB Mb de memoria RAM
- Tarjeta Gráfica con resolución 800x600 o superior y soporte DirectX 9
- Windows 7/8/10 o version Windows Server equivalente.

Para un funcionamiento normal del simulador CNC con posibilidad de simulación real, se requerirá disponer de un ordenador con los siguientes requisitos:

- Procesador de la familia Intel i-Core o superior (Incluyendo procesadores compatibles)
- 50MB de espacio libre en el disco duro.
- 2GB de memoria RAM
- Tarjeta Gráfica con resolución 800x600 o superior y soporte DirectX 9
- Windows 7/8/10 o version Windows Server equivalente

2.5 Supuestos y dependencias.

Para equipos con Windows 7 es posible que la representación de elementos del interfaz sea distinta que en Windows 8/10.

3. Modelo organizado por Objetos.

3.1 Requisitos de interface externo.

3.1.1 *Interfaz de usuario.*

Soporte de Sistema Operativo funcionando en modo gráfico con ventanas con una resolución mínima para una correcta visualización del Simulador CNC de 800 x 600 pixeles y 256 colores.

3.1.2 *Interfaz hardware.*

Procesador	Memoria	Disco duro	T. grafica
x686 1’6 GHz	2048 Mbyte	50 Mbyte	800 x 600 DirectX9

Es necesaria además una interface humana de teclado alfanumérico y un puntero de posicionamiento (ratón).

3.1.3 *Interfaz software.*

Sistema Operativo Windows 7/8/10 con las librerías gráficas necesarias y configuradas de DirectX. No se contempla que pueda existir incompatibilidad para futuras versiones del sistema operativo.

3.1.4 *Interfaz de comunicaciones.*

En este caso no se requiere especificar ninguna interface de comunicaciones, ya que actualmente no se ha planteado la conexión del programa con ningún sistema de Control Numérico por Computador.

3.2 Requisitos funcionales. Clases/Objeto.

Este apartado define el comportamiento de todas las clases de la aplicación, la cual es una lista muy extensa.

En este caso solo he añadido las clases y funciones de más relevancia para la aplicación, descartando clases genéricas que menos influyen en la operativa interna y las funciones con sobrecarga.

3.2.1 *Modo simulación.*

CONTADOR

<table>
<tr><td>Atributos</td><td>distanciaRecorrida, tiempoGastado, tiempoMaxBrazo, distancia, tiempo, error.</td></tr>
<tr><td>Funciones</td><td>
contadorG00
<table>
<tr><td>Entradas</td><td>X, Y.</td></tr>
<tr><td>Función</td><td>Calcula la distancia y el tiempo virtual en realizar la función G00 del código CNC</td></tr>
<tr><td>Salidas</td><td>ninguna</td></tr>
</table>
contadorG01
<table>
<tr><td>Entradas</td><td>Freq, X, Y.</td></tr>
<tr><td>Función</td><td>Calcula la distancia y el tiempo virtual en realizar la función G01 del código CNC</td></tr>
<tr><td>Salidas</td><td>ninguna</td></tr>
</table>
contadorG02
<table>
<tr><td>Entradas</td><td>Freq, X, Y, I, J.</td></tr>
<tr><td>Función</td><td>Calcula la distancia y el tiempo virtual en realizar la función G02 del código CNC</td></tr>
<tr><td>Salidas</td><td>ninguna</td></tr>
</table>
contadorG03
<table>
<tr><td>Entradas</td><td>Freq, X, Y, I, J.</td></tr>
<tr><td>Función</td><td>Calcula la distancia y el tiempo virtual en realizar la función G03 del código CNC</td></tr>
<tr><td>Salidas</td><td>ninguna</td></tr>
</table>
contadorS1
<table>
<tr><td>Entradas</td><td>ninguna</td></tr>
<tr><td>Función</td><td>Calcula el tiempo virtual en realizar la función de bajar el cabezal (S1 del código CNC)</td></tr>
<tr><td>Salidas</td><td>ninguna</td></tr>
</table>
contadorS_1
<table>
<tr><td>Entradas</td><td>ninguna</td></tr>
<tr><td>Función</td><td>Calcula el tiempo virtual en realizar la función de subir el cabezal (S-1 del código CNC)</td></tr>
<tr><td>Salidas</td><td>ninguna</td></tr>
</table>
contadorVacio
<table>
<tr><td>Entradas</td><td>ninguna</td></tr>
<tr><td>Función</td><td>Pone el contador a cero.</td></tr>
<tr><td>Salidas</td><td>ninguna</td></tr>
</table>
</td></tr>
</table>

actualizarDistanciaTiempo

Entradas	freq, iniX, iniY, finX, finY
Función	Actualiza con los valores de entrada la distancia y el tiempo
Salidas	ninguna

contarDistancia

Entradas	iniX, iniY, finX, finY
Función	Actualiza con los valores de entrada la distancia.
Salidas	distancia

contarTiempo

Entradas	Freq, pix
Función	Actualiza con los valores de entrada el tiempo.
Salidas	tiempo

actualizarContador

Entradas	ninguna
Función	Actualiza el panel visual del contador.
Salidas	ninguna

retrocederContador

Entradas	ninguna
Función	Retroceden todos los contadores a los últimos datos antes de su actualización.
Salidas	ninguna

Mensajes

CONTROL DE PROGRAMA

Atributos	programaModificado, haceFaltaComprobar.
Funciones	sinPrograma

Entradas	ninguno
Función	Indica al simulador CNC que no existe ningún programa de mecanizado cargado actualmente y deshabilita los controles de ejecución.
Salidas	ninguno

conPrograma

Entradas	ninguno
Función	Indica al simulador CNC que existe un programa de mecanizado cargado actualmente y habilita los controles de ejecución.
Salidas	ninguno

	reiniciarMaquina

Entradas	CargarNuevoPrograma
Función	Reinicia todas las configuraciones del Simulador CNC y elimina el programa de mecanizado cargado (si existe alguno)
Salidas	ninguno

HacerComprobar

Entradas	ninguno
Función	Indica que en la próxima ejecución del programa antes de ejecución es necesario comprobar la integridad del código.
Salidas	ninguno

Correcto

Entradas	ninguno
Función	Devuelve si el programa de mecanizado abierto es válido o no.
Salidas	correcto

Mensajes	

EJECUCION

Atributos	stop, play, NecesarioReiniciar.
Funciones	extraerXY

Entradas	parametros,absoluto.
Función	Obtiene las coordenadas X e Y a partir de los parámetros de entrada.
Salidas	x, y, correcto.

extraerFXY

Entradas	parametros,absoluto.
Función	Obtiene las coordenadas X e Y la frecuencia a partir de los parámetros de entrada.
Salidas	f, x, y, correcto.

extraerFXYIJ

Entradas	parametros,absoluto.
Función	Obtiene las coordenadas X,Y iniciales y las coordenadas X(I), Y(J) finales y la frecuencia a partir de los parámetros de entrada.
Salidas	f, x, y, i, j, correcto.

extraerFS

Entradas	parametros.
Función	Obtiene de los parámetros el sistema de pausa y el tiempo definido.
Salidas	tipo, valor, correcto.

Stop

Entradas	ninguna
Función	Obliga al simulador CNC a parar la ejecución del programa
Salidas	correcto

Play

Entradas	ninguna
Función	Reanuda la ejecución del programa de mecanizado abierto.
Salidas	correcto

TiempoPlay

Entradas	ninguna
Función	Ejecuta en una unidad de tiempo, definido por defecto de un segundo, un nuevo código del programa de mecanizado abierto.
Salidas	ninguna

Inicio

Entradas	ninguna
Función	Actualiza el estado del programa al comienzo de este.
Salidas	correcto

Final

Entradas	ninguna
Función	Actualiza el estado del programa al final de este.
Salidas	distancia

PasoMas

Entradas	ninguna
Función	Ejecuta el próximo código del programa de mecanizado devolviendo el código ejecutado.
Salidas	Codigo, correcto

PasoMenos

Entradas	ninguna
Función	Ejecuta el código desde el inicio del programa de mecanizado hasta el paso anterior al actual.
Salidas	correcto

MismoPaso

Entradas	ninguna
Función	Ejecuta el código desde el inicio del programa de mecanizado hasta el paso actual.
Salidas	ninguna

<table>
<tr><td></td><td>pasoHasta
<table>
<tr><td>Entradas</td><td>linea</td></tr>
<tr><td>Función</td><td>Ejecuta el código desde el inicio del programa de mecanizado hasta el paso que se encuentra en la línea de la entrada.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
marchaHasta
<table>
<tr><td>Entradas</td><td>final</td></tr>
<tr><td>Función</td><td>Ejecuta el código desde el inicio del programa de mecanizado en modo simulación hasta el paso que se encuentra en la posición que indica la variable final.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
evaluar_sentencia
<table>
<tr><td>Entradas</td><td>linea</td></tr>
<tr><td>Función</td><td>Ejecuta el código de la entrada línea devolviendo el nombre del código por la salida.</td></tr>
<tr><td>Salidas</td><td>Codigo</td></tr>
</table>
Reiniciado
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Solicita el reiniciado de la máquina cuando un programa está en ejecución.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

INTEGRIDAD

<table>
<tr><td>Atributos</td><td>Error, absoluto.</td></tr>
<tr><td>Funciones</td><td>comprobarIntegridadCodigo
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Comprueba en todos los códigos del programa de mecanizado abierto en busca de fallos.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
pruebaPrimerValor
<table>
<tr><td>Entradas</td><td>Cod</td></tr>
<tr><td>Función</td><td>Comprueba en el atributo de entrada Cod si el número de línea del código está bien construido.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
pruebaSegundoValor
<table>
<tr><td>Entradas</td><td>cod, param, posicion.</td></tr>
<tr><td>Función</td><td>Comprueba con los atributos de entrada si el tipo de código es correcto y soportado.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table></td></tr>
</table>

pruebaTercerValor

Entradas	cod, param.
Función	Comprueba si los parámetros del atributo de entrada param son correctos para el código del valor de entrada cod.
Salidas	correcto

comprobarNumeroEnValor

Entradas	coord, num.
Función	Comprueba si el valor del atributo num está dentro de los límites del tipo de coordenada que contiene el código del valor de entrada coord.
Salidas	correcto

comprobarTraslacion

Entradas	tras
Función	Comprueba si el valor del atributo de entrada tras se encuentra dentro de los márgenes de la velocidad de traslación mínima y máxima.
Salidas	correcto

Mensajes

AREA DE TRABAJO

Atributos	Dibujo, plano, panel,

Funciones

reiniciarMesa

Entradas	ninguno
Función	Reinicia el área de trabajo para dejarlo en su estado inicial.
Salidas	ninguno

limpiarMesa

Entradas	ninguno
Función	Elimina cualquier diseño que se haya realizado sobre el área de trabajo
Salidas	ninguno

actualizarMesa

Entradas	ninguno
Función	Actualiza el estado del área de trabajo en la pantalla de la aplicación.
Salidas	ninguno

<table>
<tr><td></td><td>actualizarZoomMesa<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Actualiza el estado del área de trabajo en la pantalla de la aplicación efectuando el redimensionado del Zoom.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>destruirMesa<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Elimina el plano virtual donde se encuentra el área de trabajo inutilizando este.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

BRAZO HERRAMIENTA

<table>
<tr><td>Atributos</td><td>dibujoOculto, dibujoPuntero, planoOculto, planoPuntero, refrescoDelPuntero, brazoSubido, X, Y, traslacion, absoluto, iPen, lPen.</td></tr>
<tr><td>Funciones</td><td>reiniciarBrazo<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Reinicia el brazo para dejarlo en su estado inicial.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>reiniciarBrazoSinRefresco<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Reinicia el brazo para dejarlo en su estado inicial, pero sin actualizar visualmente su movimiento.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>destruirBrazo<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Elimina el plano virtual donde se mueve el brazo inutilizando este.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>FuncionG00<table><tr><td>Entradas</td><td>X, Y.</td></tr><tr><td>Función</td><td>Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función G00 en consecuencia.</td></tr><tr><td>Salidas</td><td>codigo</td></tr></table>FuncionG01<table><tr><td>Entradas</td><td>F, X, Y.</td></tr><tr><td>Función</td><td>Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función G01 en consecuencia.</td></tr><tr><td>Salidas</td><td>codigo</td></tr></table></td></tr>
</table>

FuncionG02

Entradas	F, X, Y, I, J.
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función G02 en consecuencia.
Salidas	codigo

FuncionG03

Entradas	F, X, Y, I, J.
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función G03 en consecuencia.
Salidas	codigo

FuncionG04

Entradas	aux, F.
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función G04 en consecuencia.
Salidas	codigo

FuncionG90

Entradas	ninguno
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función G90 en consecuencia.
Salidas	codigo

FuncionG91

Entradas	ninguno
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función G91 en consecuencia.
Salidas	codigo

FuncionFxxx

Entradas	F.
Función	Actualiza la traslación del brazo a la que indica el atributo de entrada F. Devuelve el valor de la nueva traslación en formato estándar de G-code.
Salidas	codigo

FuncionS1

Entradas	ninguno
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función S1 en consecuencia.
Salidas	codigo

FuncionS_1

Entradas	ninguno
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función S-1 en consecuencia.
Salidas	codigo

FuncionVacia

Entradas	ninguno
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función Vacía en consecuencia.
Salidas	codigo

FuncionM2

Entradas	ninguno
Función	Determina si el brazo se encuentra en simulación dinámica o no y ejecuta la función M2 en consecuencia.
Salidas	codigo

puntero

Entradas	X, Y, lPen, iPen
Función	Actualiza la posición de la punta del brazo herramienta en pantalla si el brazo es visible.
Salidas	ninguno

absolutizarX

Entradas	X
Función	Devuelve el valor absoluto de X en la mesa de trabajo para una posición relativa de entrada del atributo X.
Salidas	X

absolutizarY

Entradas	Y
Función	Devuelve el valor absoluto de Y en la mesa de trabajo para una posición relativa de entrada del atributo Y.
Salidas	Y

<table>
<tr><td></td><td>setRefrescoPuntero
<table>
<tr><td>Entradas</td><td>refresco</td></tr>
<tr><td>Función</td><td>Habilita o deshabilita el refresco del puntero en pantalla</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
actualizarZoomBrazo
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Actualiza la visualización del puntero del brazo en pantalla aplicando la proporción del Zoom indicado.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

ESTATICO

<table>
<tr><td>Atributos</td><td>Ninguno.</td></tr>
<tr><td>Funciones</td><td>FuncionG00
<table>
<tr><td>Entradas</td><td>X, Y.</td></tr>
<tr><td>Función</td><td>Calcula la distancia existente en línea recta entre la posición actual del brazo y las nuevas coordenadas X e Y. Calcula el tiempo en recorrer dicha distancia a la velocidad máxima de movimiento del brazo y actualiza el brazo con la nueva velocidad y posición, y su interacción en la mesa de trabajo, sin simular un desplazamiento virtual del brazo. Devuelve el código de función ejecutado (G00) o un código de error.</td></tr>
<tr><td>Salidas</td><td>codigo</td></tr>
</table>
FuncionG01
<table>
<tr><td>Entradas</td><td>F, X, Y.</td></tr>
<tr><td>Función</td><td>Calcula la distancia existente en línea recta entre la posición actual del brazo y las nuevas coordenadas X e Y. Calcula el tiempo en recorrer dicha distancia con la velocidad de movimiento del brazo F actualiza el brazo con la nueva velocidad y posición, y su interacción en la mesa de trabajo, sin simular un desplazamiento virtual del brazo. Devuelve el código de función ejecutado (G01) o un código de error.</td></tr>
<tr><td>Salidas</td><td>codigo</td></tr>
</table></td></tr>
</table>

FuncionG02

Entradas	F, X, Y, I, J.
Función	Calcula la distancia existente realizando una curva entre la posición actual del brazo y las nuevas coordenadas X e Y con un centro de rotación I(X) y J(Y) en sentido horario. Calcula el tiempo en recorrer dicha distancia con la velocidad de movimiento del brazo F y actualiza el brazo con la nueva velocidad y posición, y su interacción en la mesa de trabajo, sin simular un desplazamiento virtual del brazo. Devuelve el código de función ejecutado (G02) o un código de error.
Salidas	codigo

FuncionG03

Entradas	F, X, Y, I, J.
Función	Calcula la distancia existente realizando una curva entre la posición actual del brazo y las nuevas coordenadas X e Y con un centro de rotación I(X) y J(Y) en sentido anti horario. Calcula el tiempo en recorrer dicha distancia con la velocidad de movimiento del brazo F y actualiza el brazo con la nueva velocidad y posición, y su interacción en la mesa de trabajo, sin simular un desplazamiento virtual del brazo. Devuelve el código de función ejecutado (G03) o un código de error.
Salidas	codigo

FuncionG04

Entradas	aux, F.
Función	La función G04 crea una pausa con el tiempo que se ha determinado en el atributo F, pero como esta función pertenece al modo estático sin simulación virtual no se simula la pausa de la máquina. Devuelve el código de función ejecutado (G04) o un código de error.
Salidas	codigo

FuncionG90

Entradas	ninguno
Función	Modifica el sistema de posicionamiento del brazo al modo absoluto. Devuelve el código de función ejecutado (G90) o un código de error.
Salidas	codigo

FuncionG91

Entradas	ninguno
Función	Modifica el sistema de posicionamiento del brazo al modo relativo. Devuelve el código de función ejecutado (G90) o un código de error.
Salidas	codigo

FuncionS1

Entradas	ninguno
Función	Calcula el tiempo del brazo en bajar a la mesa de trabajo y actualiza la nueva dimensión del brazo y su interacción en la mesa de trabajo, sin simular un acercamiento virtual del brazo. Devuelve el código de función ejecutado (S1) o un código de error.
Salidas	codigo

FuncionS_1

Entradas	ninguno
Función	Calcula el tiempo del brazo en subir desde la mesa de trabajo y actualiza la nueva dimensión del brazo, sin simular un alejamiento virtual del brazo. Devuelve el código de función ejecutado (S-1) o un código de error.
Salidas	codigo

FuncionM2

Entradas	ninguno
Función	Determina que ha terminado la ejecución del programa de la mesa y desaparece la punta del brazo para que sea totalmente visible el diseño. Devuelve el código de función ejecutado (M2) o un código de error.
Salidas	codigo

FuncionVacia

Entradas	ninguno
Función	El brazo no realiza ninguna acción. Devuelve una cadena vacía o un código de error.
Salidas	codigo

crearCirculo

Entradas	Freq, X, Y, I, J, posIniX, posIniY, mesa, color
Función	Crea sobre la mesa del atributo de entrada (area de trabajo) una curva entre la posición posIniX, posIniY y las nuevas coordenadas X e Y con un centro de rotación I(X) y J(Y) y del color definido en el atributo color.
Salidas	correcto

<table>
<tr><td></td><td>errorDelPuntero<table><tr><td>Entradas</td><td>Error</td></tr><tr><td>Función</td><td>Gestiona la acción no permitida que se indica que ha realizado el puntero en el atributo de entrada error y devuelve el puntero a su posición anterior.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>
pasoParaAtras<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Descarta las modificaciones realizadas por el brazo en su última ejecución y lo devuelve al estado anterior a esta.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

MOVIMIENTO

<table>
<tr><td>Atributos</td><td>iniX, iniY, finX, finY, Freq, FreqMin, FreqMax, tiempoSBrazo, tiempo, UsarEstatico, Instrucción, Movimiento, llamada.</td></tr>
<tr><td>Funciones</td><td>FuncionG00<table><tr><td>Entradas</td><td>X, Y.</td></tr><tr><td>Función</td><td>Calcula la distancia existente en línea recta entre la posición actual del brazo y las nuevas coordenadas X e Y. Calcula el tiempo en recorrer dicha distancia a la velocidad máxima de movimiento del brazo y prepara el modo simulación para un desplazamiento virtual del brazo sobre el área de trabajo. Devuelve el código de función ejecutado (G00) o un código de error.</td></tr><tr><td>Salidas</td><td>codigo</td></tr></table>
FuncionG01<table><tr><td>Entradas</td><td>F, X, Y.</td></tr><tr><td>Función</td><td>Calcula la distancia existente en línea recta entre la posición actual del brazo y las nuevas coordenadas X e Y. Calcula el tiempo en recorrer dicha distancia con la velocidad de movimiento del brazo F y prepara el modo simulación para un desplazamiento virtual del brazo sobre el área de trabajo. Devuelve el código de función ejecutado (G01) o un código de error.</td></tr><tr><td>Salidas</td><td>codigo</td></tr></table></td></tr>
</table>

FuncionG02

Entradas	F, X, Y, I, J.
Función	Calcula la distancia existente realizando una curva entre la posición actual del brazo y las nuevas coordenadas X e Y con un centro de rotación I(X) y J(Y) en sentido horario. Calcula el tiempo en recorrer dicha distancia con la velocidad de movimiento del brazo F y prepara el modo simulación para un desplazamiento virtual del brazo sobre el área de trabajo. Devuelve el código de función ejecutado (G02) o un código de error.
Salidas	codigo

FuncionG03

Entradas	F, X, Y, I, J.
Función	Calcula la distancia existente realizando una curva entre la posición actual del brazo y las nuevas coordenadas X e Y con un centro de rotación I(X) y J(Y) en sentido anti horario. Calcula el tiempo en recorrer dicha distancia con la velocidad de movimiento del brazo F y prepara el modo simulación para un desplazamiento virtual del brazo sobre sobre el área de trabajo. Devuelve el código de función ejecutado (G03) o un código de error.
Salidas	codigo

FuncionG04

Entradas	aux, F.
Función	Calcula el tiempo de pausa con el tiempo que se ha determinado en el atributo F y prepara el modo simulación para realizar la pausa. Devuelve el código de función ejecutado (G04) o un código de error.
Salidas	codigo

FuncionG90

Entradas	ninguno
Función	Modifica el sistema de posicionamiento del brazo al modo absoluto. Devuelve el código de función ejecutado (G90) o un código de error.
Salidas	codigo

FuncionG91

Entradas	ninguno
Función	Modifica el sistema de posicionamiento del brazo al modo relativo. Devuelve el código de función ejecutado (G90) o un código de error.
Salidas	codigo

FuncionS1

Entradas	ninguno
Función	Calcula el tiempo del brazo en bajar al área de trabajo de trabajo y prepara el modo simulación para un acercamiento virtual de la punta del brazo sobre el área de trabajo. Devuelve el código de función ejecutado (S1) o un código de error.
Salidas	codigo

FuncionS_1

Entradas	ninguno
Función	Calcula el tiempo del brazo en subir desde el área de trabajo de trabajo y prepara el modo simulación para un alejamiento virtual de la punta del brazo desde el área de trabajo. Devuelve el código de función ejecutado (S-1) o un código de error.
Salidas	codigo

FuncionM2

Entradas	ninguno
Función	Determina que ha terminado la ejecución del programa del área de trabajo y desaparece la punta del brazo para que sea totalmente visible el diseño. Devuelve el código de función ejecutado (M2) o un código de error.
Salidas	codigo

FuncionVacia

Entradas	ninguno
Función	El brazo no realiza ninguna acción, prepara el modo simulación para realizar una pequeña pausa de 0,25 segundos. Devuelve una cadena vacía o un código de error.
Salidas	codigo

ControlSimuladorTimer

Entradas	ninguno
Función	Controla la simulación del brazo que se encuentra en ejecución en ese momento.
Salidas	correcto

<table>
<tr><td></td><td>comprovarPosicion<table><tr><td>Entradas</td><td>laX, laY</td></tr><tr><td>Función</td><td>Verifica en cada posición durante el desplazamiento del brazo que éste continua dentro de los límites del área de trabajo.</td></tr><tr><td>Salidas</td><td>correcto</td></tr></table>nuevaCoordenada<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Devuelve la siguiente coordenada a la que tiene que desplazarse el brazo en una simulación virtual.</td></tr><tr><td>Salidas</td><td>iniX, iniY.</td></tr></table>tiempoTimer<table><tr><td>Entradas</td><td>Freq</td></tr><tr><td>Función</td><td>Calcula y devuelve el tiempo que tarda en desplazar el brazo en un pixel de posición a una frecuencia (Freq) de entrada determinada.</td></tr><tr><td>Salidas</td><td>tiempo</td></tr></table>actualizarRapidoContador<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Descarta las modificaciones realizadas por el brazo en su última ejecución y lo devuelve al estado anterior a esta.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

CONFIGURACION

<table>
<tr><td>Atributos</td><td>MesaX, MesaY, PunteroXIni, PunteroYIni, PunteroTraslacionIni, PunteroTraslacionMax, PunteroTraslacionMin, PunteroAbsolutoIni, PixelesPorFrecuencia, Simular, BrazoVisible, AlturaPuntero, MilisPorIterSubidaPuntero, TiempoPasoEnEstatico, PasoDeTabla, ColorPuntero, ColorLinea, ColorMesa, ruta.</td></tr>
<tr><td>Funciones</td><td>guardarConfiguracion<table><tr><td>Entradas</td><td>MesaX, MesaY, PunteroXIni, PunteroYIni, PunteroTraslacionIni, PunteroTraslacionMax, PunteroTraslacionMin, PunteroAbsolutoIni, PixelesPorFrecuencia, Simular, BrazoVisible, AlturaPuntero, MilisPorIterSubidaPuntero, TiempoPasoEnEstatico, PasoDeTabla, ColorPuntero, ColorLinea, ColorMesa.</td></tr><tr><td>Función</td><td>Actualiza los atributos del objeto de la clase con los nuevos atributos</td></tr><tr><td>Salidas</td><td>correcto</td></tr></table></td></tr>
</table>

cargarConfiguracion

Entradas	archivo
Función	Recoge los parámetros desde el archivo de entrada y actualiza los atributos.
Salidas	correcto

getMesaX

Entradas	ninguno
Función	Envia el atributo MesaX por la salida.
Salidas	MesaX

getMesaY

Entradas	ninguno
Función	Envia el atributo MesaY por la salida.
Salidas	MesaY

PunteroXIni

Entradas	ninguno
Proceso	Envia el atributo PunteroXIni por la salida.
Salidas	PunteroXIni

PunteroYIni

Entradas	ninguno
Función	Envia el atributo PunteroYIni por la salida.
Salidas	PunteroYIni

PunteroTraslacionIni

Entradas	ninguno
Función	Envia el atributo PunteroTraslacionIni por la salida.
Salidas	PunteroTraslacionIni

PunteroTraslacionMax

Entradas	ninguno
Función	Envia el atributo PunteroTraslacionMax por la salida.
Salidas	PunteroTraslacionMax

PunteroTraslacionMin

Entradas	ninguno
Función	Envia el atributo PunteroTraslacionMin por la salida.
Salidas	PunteroTraslacionMin

PunteroAbsolutoIni

Entradas	ninguno
Función	Envia el atributo PunteroAbsolutoIni por la salida.
Salidas	PunteroAbsolutoIni

Mensajes

3.2.2 *Modo Usuario.*

CONFIGURACION BASICA

<table>
<tr><td>Atributos</td><td>ninguno</td></tr>
<tr><td>Funciones</td><td>mostrarValores
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Muestra un cuadro de dialogo con los valores contenidos dentro de la clase CONFIGURACION, con la posibilidad de modificarlos.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
mostrarValoresPorDefecto
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Restaura en el cuadro de dialogo los valores actuales por los valores por defecto definidos en la clase CONFIGURACION.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
revisarValores
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Comprueba que todos los valores de configuración modificados sean correctos.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
revisarTamanoMesa
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Comprueba que el tamaño del área de trabajo sea correcto.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
revisarVelocidadDeDibujo
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Proceso</td><td>Comprueba que el tiempo definido para el dibujo por el área de trabajo sea correcto .</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
revisarPosicionInicialPuntero
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Comprueba que la posición inicial definida de la punta del brazo sea válida.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
revisarVelocidadTraslacion
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Comprueba que el tiempo definido para la velocidad de movimiento del brazo sea correcto .</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td>actualizarValores<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Actualiza los valores de configuración en la clase CONFIGURACION.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>VelocidadMovimientoMax<table><tr><td>Entradas</td><td>Freq, MinimoTiempoMiliSeg.</td></tr><tr><td>Función</td><td>Calcula y devuelve la velocidad máxima a la que se puede desplazar el brazo a partir de los atributos de entrada Freq y MinimoTiempoMiliSeg</td></tr><tr><td>Salidas</td><td>velocidad</td></tr></table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

TABLA

<table>
<tr><td>Atributos</td><td>CeldaSeleccionadaFila, CeldaSeleccionadaColumna, paso.</td></tr>
<tr><td>Funciones</td><td>programaCargado<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Se prepara la tabla para mostrar el paso de ejecución desde el principio.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>reiniciarTabla<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Se restaura la tabla a sus valores iniciales antes de la apertura de un programa de mecanizado.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>setPaso<table><tr><td>Entradas</td><td>Paso, llamarAtencion</td></tr><tr><td>Función</td><td>Marca el nuevo paso en ejecución desplazando la tabla hasta dicha línea si llamarAtencion está activado.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>setPasoSinActualizarTabla<table><tr><td>Entradas</td><td>Paso</td></tr><tr><td>Función</td><td>Se traslada al nuevo paso en ejecución pero sin marcar ni actualizar la tabla.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table></td></tr>
</table>

insertarEnTabla

Entradas	partes
Proceso	Inserta en la tabla en una nueva línea el código CNC dentro del atributo de entrada partes.
Salidas	ninguno

marcaDeLinea

Entradas	linea
Función	Marca la línea del atributo de entrada línea en la tabla.
Salidas	ninguno

remarcarLinea

Entradas	linea
Función	Marca la línea del atributo de entrada línea en la tabla y desplaza la tabla hasta dicha línea.
Salidas	ninguno

edicionDeCelda

Entradas	Linea, columna
Función	Cambia el estado de la celda que indican los atributos de las entradas para que sea posible la edición del contenido por el usuario.
Salidas	ninguno

desedicionDeCelda

Entradas	ninguno
Función	Cambia el estado de la celda en edición al estado de solo lectura.
Salidas	ninguno

cambiarSeleccion

Entradas	antiguo, nuevo.
Función	Cambia la selección desde la fila antiguo a la fila nuevo.
Salidas	ninguno

llamarAtencionFila

Entradas	x
Función	Desplaza la tabla hasta la línea que indica el atributo de entrada x.
Salidas	ninguno

fila

Entradas	numFila
Función	Elimina de la fila que indica el atributo numFila, los espacios en blanco y los caracteres no válidos.
Salidas	ninguno

numeroDeFilas

Entradas	antiguo, nuevo.
Función	Devuelve el número de filas totales que contiene la tabla.
Salidas	numero

obtenerValoresFila

Entradas	x
Función	Devuelve los valores de la fila a la que apunta el atributo de entrada x.
Salidas	numero, código, parámetros.

insertarNuevaFilaEnFila

Entradas	x, numero, codigo, parametros.
Función	Modifica la fila a la que apunta el atributo de entrada x sustituyendo sus columnas por la de los atributos número, codigo, parámetros.
Salidas	ninguno

insertarFilaVacia

Entradas	fila, encima.
Función	Inserta una nueva fila en la tabla, si encima es falso la inserta después de donde indica el atributo de entrada fila, si es verdadero la inserta antes.
Salidas	ninguno

eliminarFila

Entradas	fila.
Función	Elimina de la tabla la fila a la que apunta el atributo de entrada fila.
Salidas	ninguno

Mensajes	

ZOOM

Atributos	Zoom, alto, ancho.
Funciones	setZoom

Entradas	z
Función	Actualiza el porcentaje de ampliación del área de trabajo de la máquina al especificado por el atributo de entrada z.
Salidas	ninguno

<table>
<tr><td></td><td>actualizarMesaYBrazo<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Actualiza el tamaño del área de trabajo a los nuevos valores de ampliación.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table>getZoom<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Devuelve el valor del porcentaje de ampliación actual.</td></tr><tr><td>Salidas</td><td>zoom</td></tr></table>getY<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Devuelve el tamaño del eje Y del área de trabajo con el porcentaje de ampliación aplicado.</td></tr><tr><td>Salidas</td><td>y</td></tr></table>getX<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Devuelve el tamaño del eje X del área de trabajo con el porcentaje de ampliación aplicado.</td></tr><tr><td>Salidas</td><td>x</td></tr></table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

ERROR

<table>
<tr><td>Atributos</td><td>ninguno.</td></tr>
<tr><td>Funciones</td><td>error<table><tr><td>Entradas</td><td>NumErr, txt1, txt2, txt3.</td></tr><tr><td>Función</td><td>Examina con el atributo de entrada NumErr el tipo de error que es y complementando con los atributos de entrada txt1, txt2 y txt3 construye un mensaje de error que devuelve por la salida.</td></tr><tr><td>Salidas</td><td>msgError</td></tr></table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

DETALLES_CABEZAL

<table>
<tr><td>Atributos</td><td>ninguno</td></tr>
<tr><td>Funciones</td><td>reiniciarDetallesCabezal<table><tr><td>Entradas</td><td>ninguno</td></tr><tr><td>Función</td><td>Sustituye en el cuadro de dialogo de los detalles del cabezal los valores actuales a los valores iniciales del cabezal.</td></tr><tr><td>Salidas</td><td>ninguno</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td>
Actualizar
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Actualiza en el cuadro de dialogo de los detalles del cabezal los valores actuales del cabezal.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
Posicion
<table>
<tr><td>Entradas</td><td>X, Y</td></tr>
<tr><td>Función</td><td>Muestra en la posición actual del cabezal los atributos X e Y de la entrada.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
Estado
<table>
<tr><td>Entradas</td><td>es</td></tr>
<tr><td>Función</td><td>Muestra como estado actual del brazo el que se define en el atributo de entrada es.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
Absoluto
<table>
<tr><td>Entradas</td><td>abs</td></tr>
<tr><td>Función</td><td>Muestra como modo de posicionamiento actual del brazo el que se define en el atributo de entrada abs.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
setTiempo
<table>
<tr><td>Entradas</td><td>t</td></tr>
<tr><td>Función</td><td>Muestra como tiempo de ejecución del brazo el que se define en el atributo de entrada t.</td></tr>
<tr><td>Salidas</td><td>Ninguno</td></tr>
</table>
setDistancia
<table>
<tr><td>Entradas</td><td>d</td></tr>
<tr><td>Función</td><td>Muestra como distancia recorrida del brazo el que se define en el atributo de entrada d.</td></tr>
<tr><td>Salidas</td><td>Ninguno</td></tr>
</table>
</td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

NUEVO

<table>
<tr><td>Atributos</td><td>ninguno</td></tr>
<tr><td>Funciones</td><td>crearNuevoPrograma
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Reinicia la aplicación, crea el nuevo programa a partir del nombre y del número inicial de líneas seleccionado y lo abre en la aplicación.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>correcto
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Comprueba que los datos del nombre y del número inicial de líneas introducidos por el usuario sean válidos.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
crearDatos
<table>
<tr><td>Entradas</td><td>nombre, numFilas</td></tr>
<tr><td>Función</td><td>Crea el esqueleto del programa de mecanizado a partir de los atributos de entrada y los carga en el Simulador CNC</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
filtrarNombre
<table>
<tr><td>Entradas</td><td>n</td></tr>
<tr><td>Función</td><td>Sustituye los caracteres del nombre del programa que no son aceptados para nombre de fichero por un guion bajo _ y devuelve el resultado.</td></tr>
<tr><td>Salidas</td><td>nombre</td></tr>
</table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

ABRIR

<table>
<tr><td>Atributos</td><td>ninguno</td></tr>
<tr><td>Funciones</td><td>abre
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Visualiza la ventana para seleccionar un fichero y una vez seleccionado el fichero por el usuario, este se envía para abrir.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
cargarNuevoPrograma
<table>
<tr><td>Entradas</td><td>File</td></tr>
<tr><td>Función</td><td>Abre el archivo del atributo de entrada File, lo lee y comprueba si los códigos son correctos, y en caso afirmativo reinicia el simulador y lo prepara con el nuevo programa.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table>
obtenerDatos
<table>
<tr><td>Entradas</td><td>stream, filename.</td></tr>
<tr><td>Función</td><td>Recoge los datos del archivo especificado en el atributo de entrada filename.</td></tr>
<tr><td>Salidas</td><td>correcto</td></tr>
</table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

GUARDAR

Atributos	ninguno
Funciones	guarda *Entradas*: Como *Función*: Si el atributo de entrada Como es verdadero, muestra un cuadro de dialogo para que insertemos un nombre para el archivo a guardar, si es falso no muestra ningún cuadro de dialogo y pasa al siguiente paso. En el siguiente paso recopila los datos del programa y los almacena en el archivo a guardar especificado. *Salidas*: correcto guardarDatos *Entradas*: stream, filename *Función*: Abre o crea el archivo que se especifica en el atributo de entrada filename, y se recogen los códigos del programa de la clase TABLA para almacenarlos en este archivo. *Salidas*: correcto
Mensajes	

GUARDAR_IMAGEN

Atributos	ninguno
Funciones	guardaImagen *Entradas*: ninguno *Función*: Muestra un cuadro de dialogo para que insertemos un nombre para el archivo a guardar, obtiene los datos gráficos de la mesa de trabajo y los almacena en el archivo especificado. *Salidas*: correcto
Mensajes	

INSERTAR_CODIGO

Atributos	ninguno
Funciones	ReiniciarInsertarCodigo *Entradas*: ninguno *Función*: Reinicia y oculta el panel de inserción de nuevos códigos. *Salidas*: ninguno deshabilitarInsertarCodigo *Entradas*: ninguno *Función*: Oculta el panel de inserción de nuevos códigos. *Salidas*: ninguno

habilitarInsertarCodigo

Entradas	stream, filename.
Función	Muestra el panel de inserción de nuevos códigos.
Salidas	correcto

sinCuadros

Entradas	ninguno
Función	Oculta los cuadros de valores de inserción.
Salidas	ninguno

renovarValores

Entradas	ninguno
Función	Restaura los cuadros de valores de inserción a cero.
Salidas	ninguno

Insertar

Entradas	ninguno
Función	Recoge el tipo de código G-code y los valores definidos, los procesa, convierte y los inserta en la posición de la tabla indicada antes.
Salidas	ninguno

averiguarFila

Entradas	fila, menuDerecho
Función	A partir de la fila seleccionada en la tabla, estudia la numeración de las filas siguiente o anterior y devuelve por la salida la media aritmética entre la fila seleccionada y una de las anteriores.
Salidas	fila

redimensionarFilas

Entradas	fila, menuDerecho
Función	A partir de la fila seleccionada en la tabla, estudia la numeración de las filas siguientes o anteriores, mide su distancia numérica con la fila seleccionada, desplaza las siguientes filas a partir de su numeración original más la distancia desarrollada y devuelve por la salida la fila seleccionada sumada a la distancia calculada.
Salidas	fila

comprobarNumeroLinea

Entradas	linea
Función	Comprueba que el número del atributo de entrada línea sea válido como numeración para la especificación de una numeración de línea de código CNC.
Salidas	correcto

buscarFilaSeleccionada

Entradas	ninguno
Función	Busca y devuelve la línea que ha sido seleccionada por el usuario en la tabla de códigos CNC.
Salidas	fila

rellenarParametros

Entradas	ninguno
Función	Recoge la selección del código y los parámetros para este indicados por el usuario, los procesa y los devuelve por la salida.
Salidas	codigo, parametros

comprobarVelocidadTraslacion

Entradas	f
Función	Comprueba que la velocidad de traslación F del atributo de la entrada es válido y lo devuelve por la salida.
Salidas	freq

comprobarPosicionX

Entradas	x
Función	Comprueba que la posición x de la entrada es válida y está dentro del margen horizontal del área de trabajo. Si es correcto lo devuelve por la salida.
Salidas	X

comprobarPosicionY

Entradas	y
Función	Comprueba que la posición y de la entrada es válida y está dentro del margen vertical del área de trabajo. Si es correcto lo devuelve por la salida.
Salidas	Y

comprobarPosicionI

Entradas	i
Función	Comprueba que la posición i de la entrada es válida y está dentro del margen horizontal del área de trabajo. Si es correcto lo devuelve por la salida.
Salidas	I

comprobarPosicionJ

Entradas	j
Función	Comprueba que la posición j de la entrada es válida y está dentro del margen vertical del área de trabajo. Si es correcto lo devuelve por la salida.
Salidas	J

	extraerNumeroValido *Entradas*: n *Función*: Comprueba que el valor del atributo de entrada n sea un valor numérico válido y lo devuelve por la salida. *Salidas*: N
Mensajes	

MARCHA

Atributos	ninguno
Funciones	habilitarMarcha *Entradas*: ninguno *Función*: Habilita los controles que permiten ejecutar el programa de mecanizado abierto. *Salidas*: ninguno desabilitarMarcha *Entradas*: ninguno *Función*: Deshabilita los controles que permiten ejecutar el programa de mecanizado abierto. *Salidas*: Ninguno inicioDeLaMarcha *Entradas*: ninguno *Función*: Habilita los controles que permiten la terminación de la ejecución del programa de mecanizado, deshabilita los controles destinados a iniciar la ejecución del programa de mecanizado. *Salidas*: ninguno finDeLaMarcha *Entradas*: ninguno *Función*: Habilita los controles que permiten la ejecución del programa de mecanizado, deshabilita los controles destinados a terminar la ejecución del programa de mecanizado. *Salidas*: ninguno
Mensajes	

MENU_DERECHO

<table>
<tr><td>Atributos</td><td>MenuDerechoFila</td></tr>
<tr><td>Funciones</td><td>reiniciarMenuDerecho
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Reinicia los valores del menú derecho del ratón en la tabla de códigos G-code del Simulador CNC.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
getMenuDerechoFila
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Devuelve por la salida el número de la fila en la que se ha hecho clic con el botón derecho del ratón.</td></tr>
<tr><td>Salidas</td><td>fila</td></tr>
</table>
MenuDerechoFila_Click
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Muestra el menú derecho de la tabla de códigos G-code.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
AvanzarAEstaLinea
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Ejecuta los códigos G-code cargados en la tabla hasta la línea en la que se ha desplegado el menú derecho.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
NuevaFilaAnterior
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Inserta una nueva fila vacía una línea anterior a la línea en la que se ha desplegado el menú derecho.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
NuevaFilaSiguiente
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Inserta una nueva fila vacía una línea siguiente a la línea en la que se ha desplegado el menú derecho.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table>
BorrarFila
<table>
<tr><td>Entradas</td><td>ninguno</td></tr>
<tr><td>Función</td><td>Elimina la línea en la que se ha desplegado el menú derecho.</td></tr>
<tr><td>Salidas</td><td>ninguno</td></tr>
</table></td></tr>
<tr><td>Mensajes</td><td></td></tr>
</table>

3.3 Requisitos de eficiencia.

En torno a la memoria RAM y a la necesidad de cálculos complejos que usará el Simulador CNC, dado que solo se usará un renderizado en 2D, no se prevé un uso excesivo de recursos si se le aplica un uso normal a la aplicación.

Para el modo simulado es necesaria cierta potencia gráfica por lo que es en esta parte donde se tiene que optimizar mayormente el rendimiento del Simulador CNC.

3.4 Restricciones de diseño.

3.4.1 *Estándares cumplidos.*

La aplicación debe ser compatible con el núcleo del sistema de Windows, sus librerías básicas y con el entorno de gráficos de este.

La aplicación debe poder comunicarse con las librerías de aceleración gráfica de DirectX 9 o superior.

3.4.2 *Limitaciones hardware.*

El sistema gráfico del equipo debe de tener soporte hardware para aceleración gráfica por DirectX 9 o superior.

Otras limitaciones a determinar.

3.5 Atributos.

3.5.1 *Seguridad.*

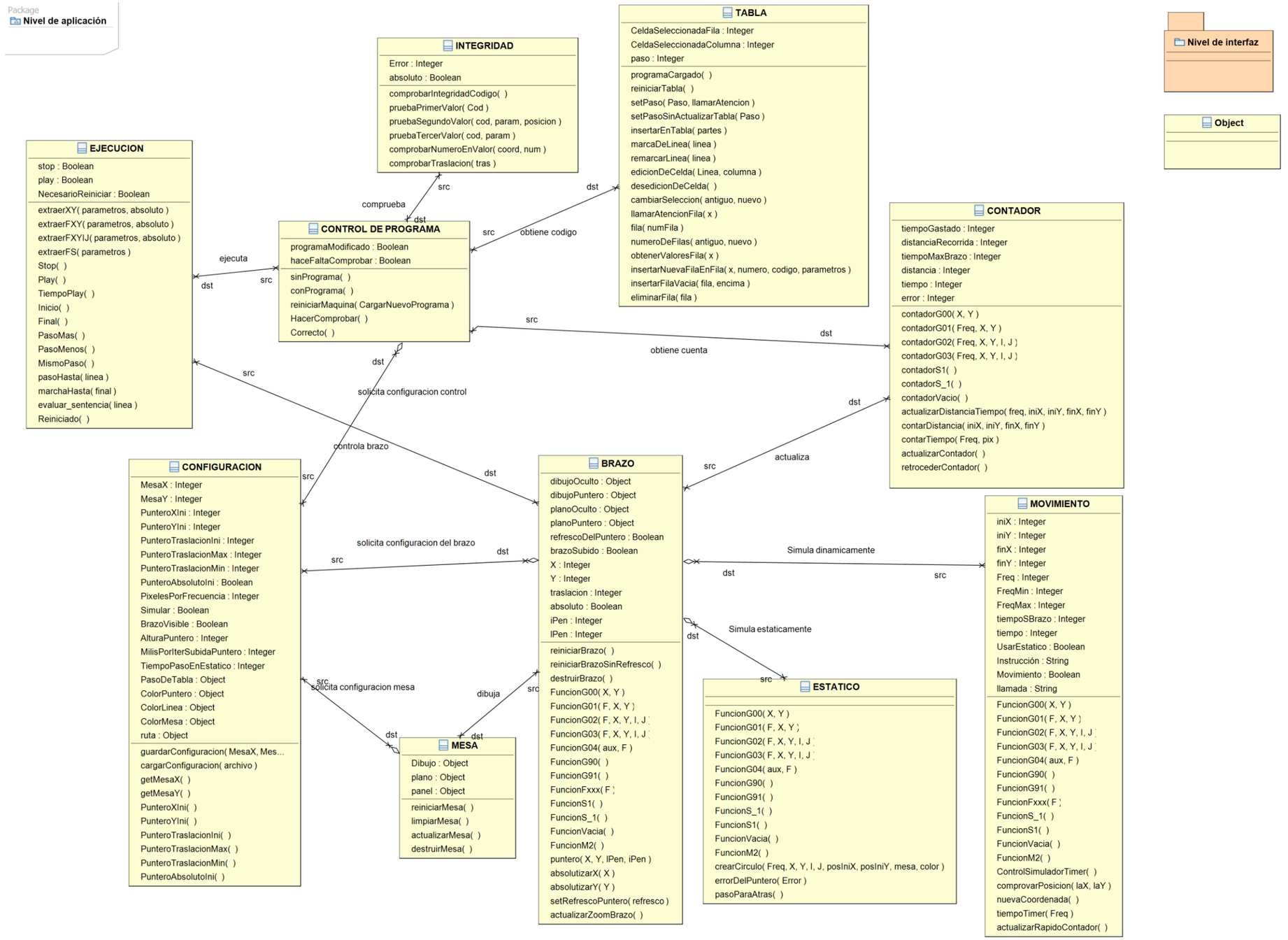
No se definen.

3.5.2 *Mantenimiento.*

El mantenimiento de la aplicación se facilitará en lo posible, incluyendo referencias y comentarios en el código fuente de la aplicación, disponiendo del documento de ERS y con la ayuda de los diagramas del diseño.

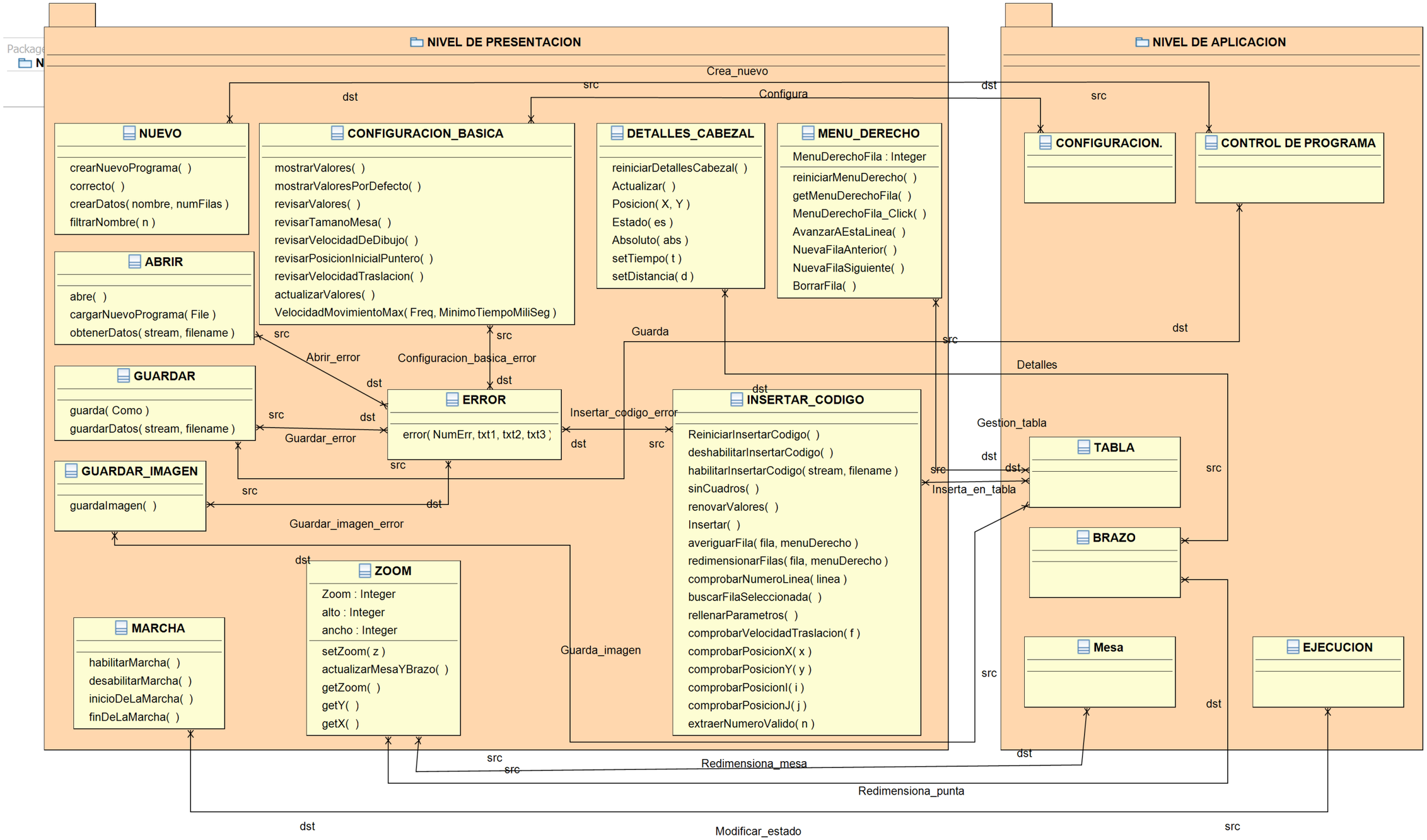
ANEXO 2.- Diagrama UML de la capa de control

Diagrama UML de la capa de control del programa Simulador CNC desarrollado con la aplicación CASE MOSkitt.



ANEXO 3.- Diagrama UML de la capa de interfaz

Diagrama UML de la capa de interfaz del programa Simulador CNC desarrollado con la aplicación CASE MOSkitt, mostrando además su interacción con la capa de control.



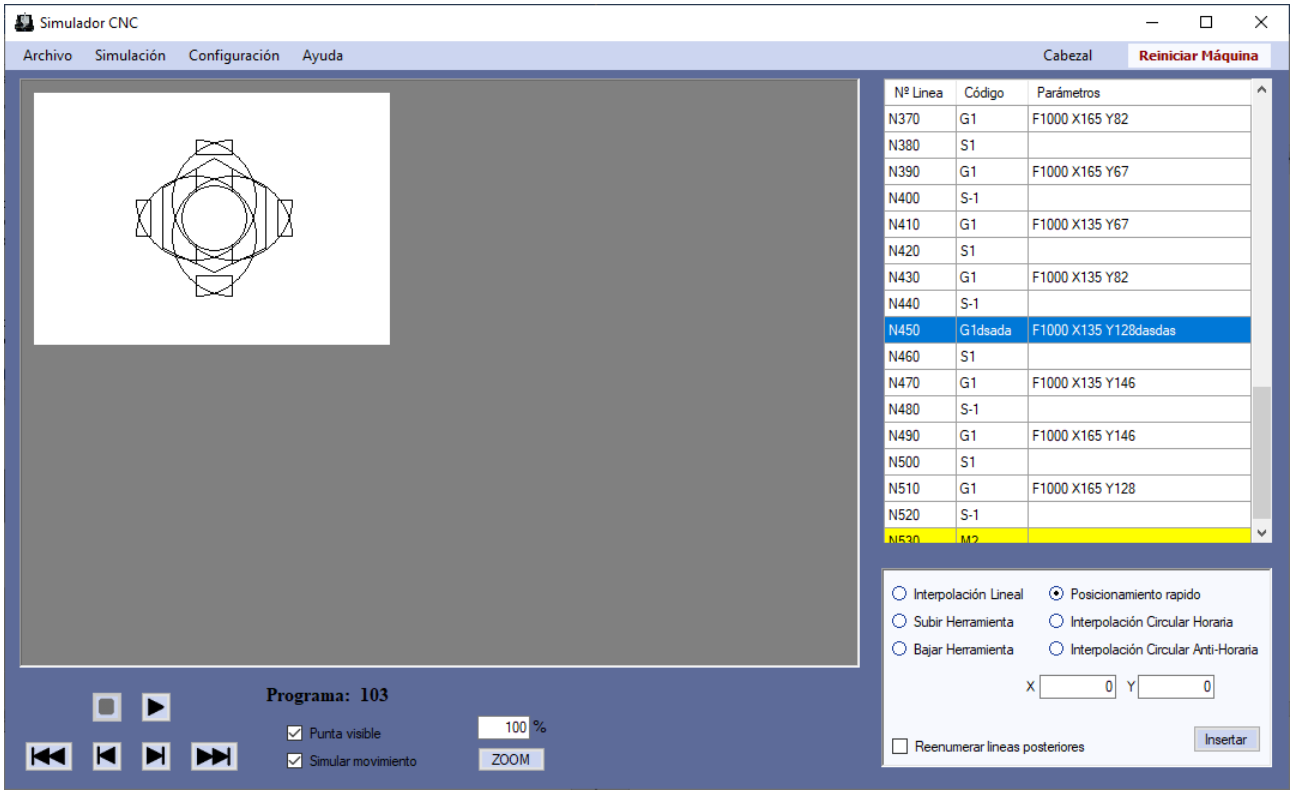
ANEXO 4.- Manual de usuario

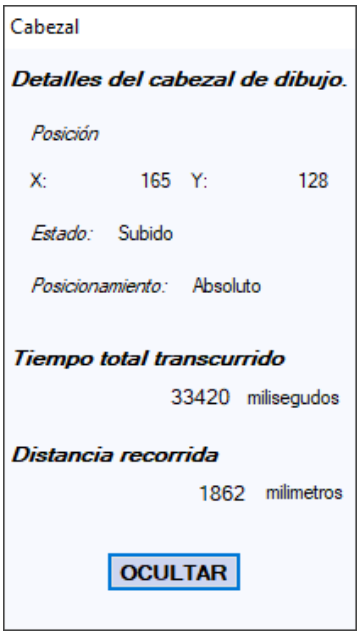
En este anexo se desarrollan las diversas funciones del programa Simulador CNC.

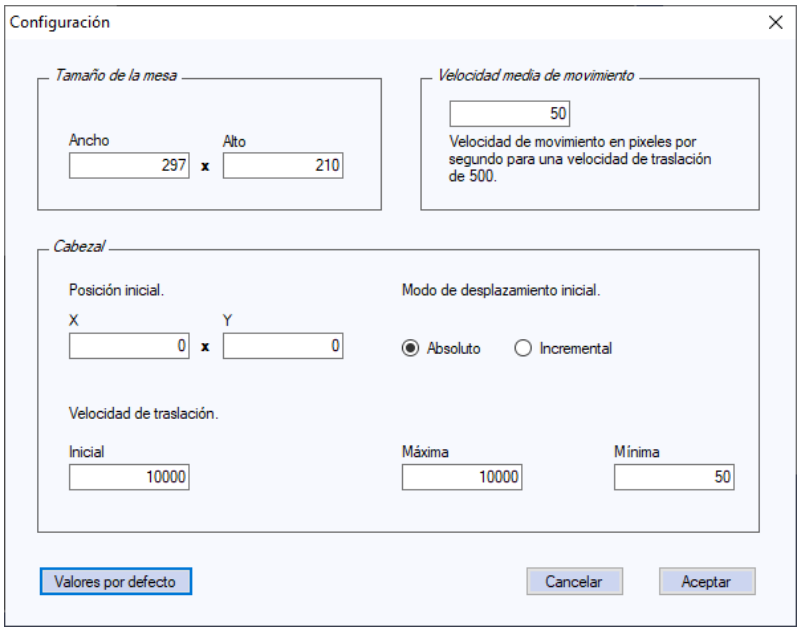
En primer caso se describen los controles de la interfaz, en la siguiente sección se especifica el funcionamiento de los sistemas de simulación disponibles, prosiguiendo con la creación, edición y almacenamiento de programas de mecanizado, finalizando con las opciones de configuración.

A4.1.- Interfaz.

En este capítulo se desarrollan todos los elementos que posee la interfaz, explicando la finalidad y si es posible la modificación del elemento.





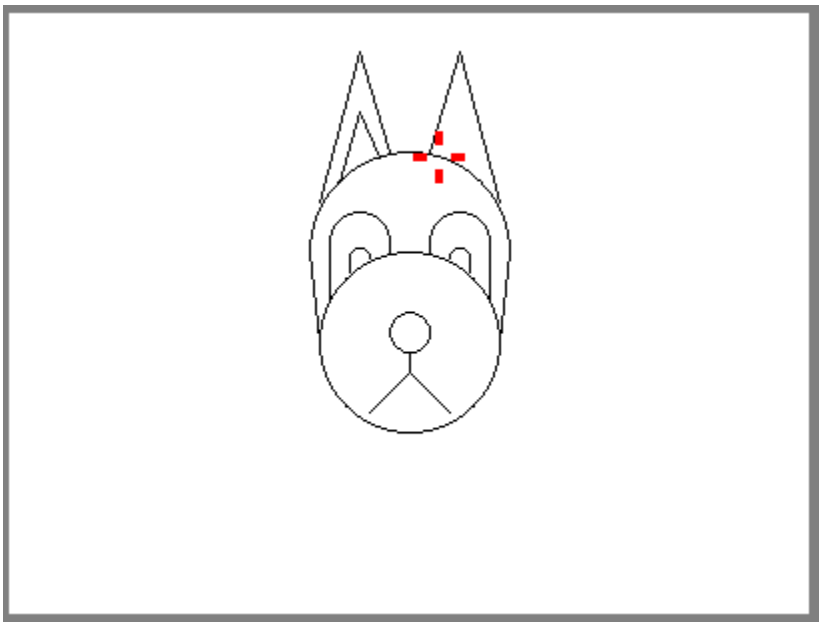


Cada uno de los siguientes apartados corresponde a una sección de la interfaz que compone varios elementos con propiedades comunes.

A4.1.1.- Área de mecanizado.

Esta zona se compone de dos elementos fácilmente diferenciables:

- **Mesa de trabajo.**



La mesa de trabajo es la zona que simula, como su nombre indica, la mesa en la que crearía su diseño una máquina CNC real. La simulación del dibujo sobre la mesa se puede crear de dos formas según el tipo de simulación.

En tiempo real dibujándose pixel a pixel al paso del cabezal del brazo apoyado sobre la mesa. En tiempo no real dibujando directamente sobre la mesa el diseño correspondiente a cada uno de los códigos CNC del programa.

- **Brazo Herramienta.**

El brazo herramienta es representado en la aplicación como una especie de aspa que sobrevuela por encima de la mesa.

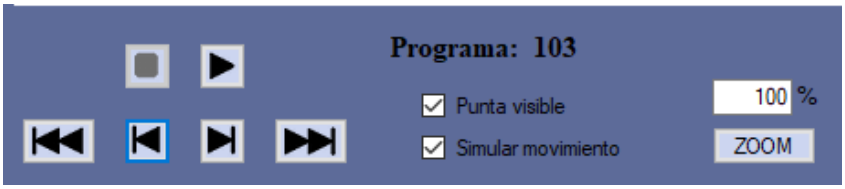
El diseño de este depende de en qué posición se encuentre, si el brazo está elevado el aspa será grande como en el dibujo de arriba izquierda, si el brazo está apoyado sobre la mesa se observa un aspa pequeña como el dibujo de arriba derecha.

Únicamente cuando el brazo esté posado sobre la mesa es cuando se realizará el diseño sobre ella, si el brazo sobrevuela la mesa estando elevado no creará diseño alguno a su paso.

Al igual que la mesa, el brazo herramienta funciona de forma distinta según el modo de simulación.

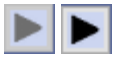
En tiempo real el movimiento del brazo se realiza pixel a pixel, incluyendo la animación de subir o bajar el brazo, actualizándose al mismo tiempo la posición de este. En tiempo no real, en la ejecución de cada código G-code el brazo se posiciona en la posición final en que queda al ejecutar dicho código.

A4.1.2.- Zona de control.



Esta zona comprende los elementos que nos permiten controlar la ejecución de la simulación y la forma de visualización de esta.

- **Controles de simulación.**

 ***PLAY***

Cuando se activa continúa la simulación desde el código actual donde se halla el programa hasta el final del programa (a no ser que se le dé la orden de parar). Si la simulación

del programa no está activa el play estará habilitado, si el programa está en ejecución el play se desactivará hasta que acabe la ejecución.

 ***STOP***

Si la simulación del programa no está activa este elemento se encuentra desactivado, en el caso de que se inicie la ejecución del programa este se activa permitiendo ser usado. El uso de este es el de finalizar la ejecución del programa, en el caso de que se pulse sobre él, no se finaliza directamente la ejecución, sino que termina cuando acaba de ejecutar el código G-code en curso (mientras acaba el botón cambiará a color rojo).

 ***Avanzar un paso***

Este control permite ejecutar el próximo código G-code para pasar al siguiente estado de la máquina. Mientras el programa se encuentra en ejecución este control se desactiva.

 ***Avanzar hasta el final***

Al activar este control, la aplicación ejecuta todos los códigos G-code a partir del siguiente al actual hasta el final del programa (o hasta que encuentre algún fallo del programa). El avance hasta el final lo realiza directamente sin simulación, actualizando el resultado del Brazo herramienta y la Mesa de trabajo a este estado final.

 ***Retroceder un paso***

Este control permite deshacer la ejecución del último código G-code quedando la máquina en el estado anterior a este. La activación de este control no activa la simulación del sistema, sino que pasa directamente al último paso anterior. Mientras el programa se encuentra en ejecución este control se desactiva.

 ***Retroceder hasta el inicio***

Al activar este control, la aplicación deshace la ejecución de todos los códigos y vuelve al estado inicial del programa. Como es obvio, no realiza ninguna simulación, sino que deshace directamente todas las modificaciones.

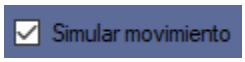
A4.1.3.- Indicador de programa.

Programa: Esta etiqueta muestra el nombre del programa CNC que se encuentra actualmente abierto en el simulador. En el caso de que no haya programa abierto no aparecerá ningún nombre.

A4.1.4.- Controles de visualización.

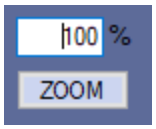
Representar Brazo Herramienta

Si este control se encuentra activado, se observa el brazo herramienta descrito anteriormente. Si está desactivado el brazo herramienta no se observa, no obstante, la simulación de un programa se realiza igual que si se representara el brazo herramienta (pero sin verse este).



Simulación activa / Simulación pasiva

Durante la ejecución de un programa G-code, si este control se encuentra activado la simulación se representa con movimiento simulado del Brazo herramienta sobre la mesa. Si el control se encuentra desactivado la simulación se representa paso por paso sin movimiento simulado.



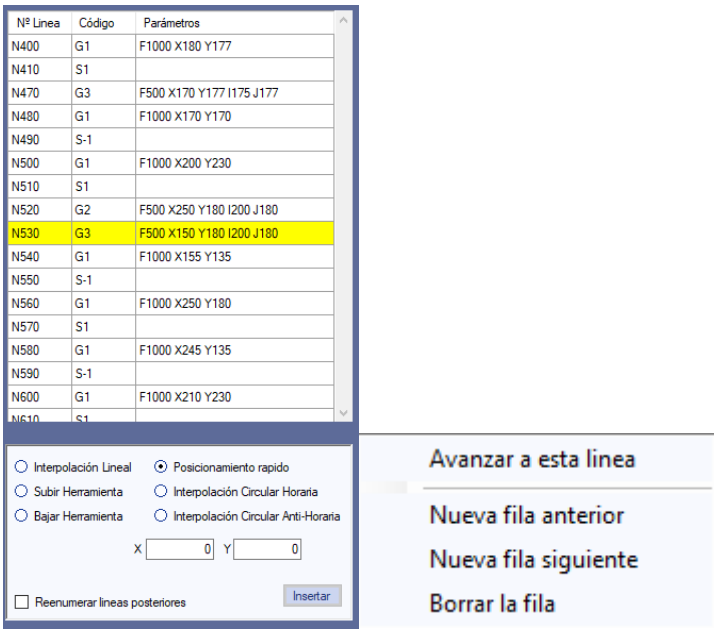
ZOOM

Este control permite aumentar o reducir porcentualmente el tamaño de los elementos del área de mecanizado

Aunque se redimensiona su representación visual, internamente cada elemento (Brazo herramienta, Mesa de trabajo) funciona para el programa G-code con las mismas dimensiones configuradas en la aplicación.

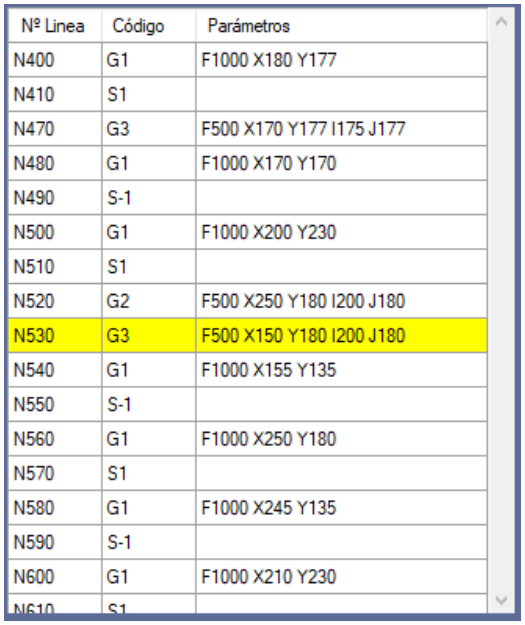
Este control no tiene limitación para el aumento de la visualización, solamente queda limitado por los recursos que se encuentren disponibles en el ordenador en el que se ejecuta.

A4.1.5.- Zona del programa de mecanizado.



Esta zona incluye los códigos que contiene el programa G-code y los elementos necesarios para su edición.

A4.1.6.- Tabla de códigos G-code.



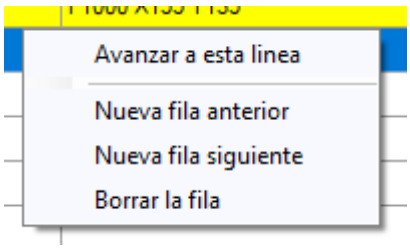
Nº Linea	Código	Parámetros
N400	G1	F1000 X180 Y177
N410	S1	
N470	G3	F500 X170 Y177 I175 J177
N480	G1	F1000 X170 Y170
N490	S-1	
N500	G1	F1000 X200 Y230
N510	S1	
N520	G2	F500 X250 Y180 I200 J180
N530	G3	F500 X150 Y180 I200 J180
N540	G1	F1000 X155 Y135
N550	S-1	
N560	G1	F1000 X250 Y180
N570	S1	
N580	G1	F1000 X245 Y135
N590	S-1	
N600	G1	F1000 X210 Y230
N610	S1	

La tabla de códigos representa sobre una tabla todas las líneas de código que componen el programa abierto actualmente. La tabla se divide en tres columnas.

La primera columna contiene el número de línea que se ha adjudicado al código, la segunda columna contiene la clave del código que ha de interpretar y la tercera columna contiene los parámetros que son necesarios para el código de la segunda columna.

La próxima línea que se tiene que ejecutar del programa se muestra en color amarillo. La aplicación permite editar directamente una línea realizando una doble pulsación con el ratón sobre la línea deseada.

A4.1.7.- Menú de ratón (botón derecho) en tabla de códigos.



Para que este submenú se despliegue es necesario realizar una pulsación con el botón derecho del ratón sobre la fila de la tabla de códigos a la que queremos que afecte la opción.

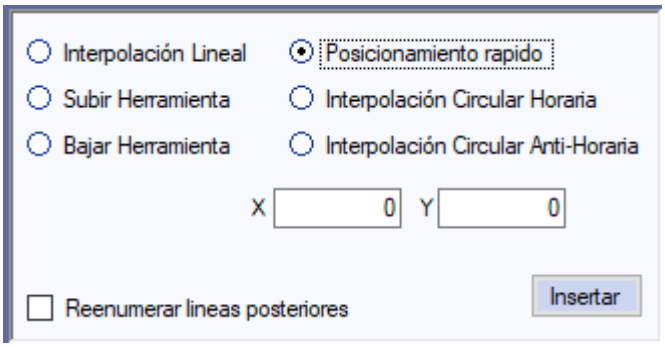
Este submenú contiene varias opciones:

Avanzar a esta línea. Realiza la ejecución del programa CNC abierto desde el principio hasta la línea seleccionada.

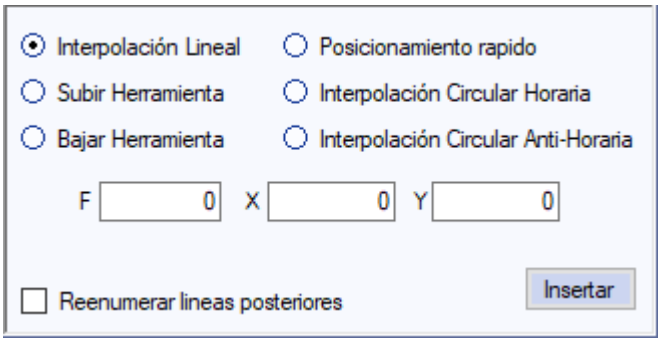
Nueva fila anterior. Inserta una nueva fila antes de la fila seleccionada.
Nueva fila siguiente. Inserta una nueva fila después de la fila seleccionada.
Borrar la fila. Elimina de la tabla y del programa CNC la fila seleccionada.

A4.1.8.- Cuadro de programación de G-code.

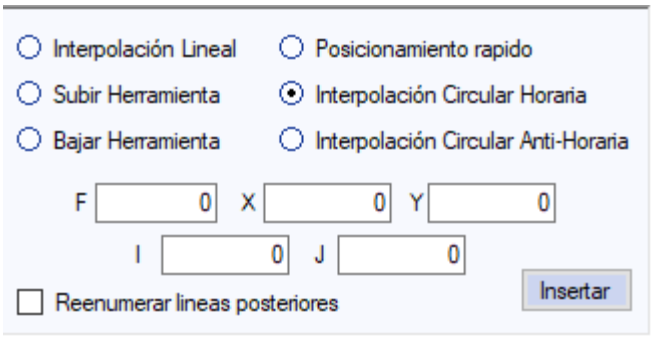
Este panel posee una serie de opciones que nos permiten insertar nuevas filas con códigos G-code configurados por nosotros sobre el programa abierto. Las posibles opciones son:



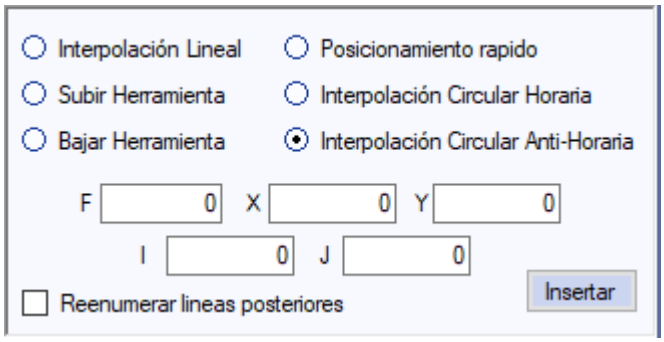
Posicionamiento rápido. Inserta en la tabla de códigos un nuevo código que obliga al Brazo herramienta a posicionarse en unas coordenadas a la máxima velocidad. El parámetro de este código es la posición final de destino.



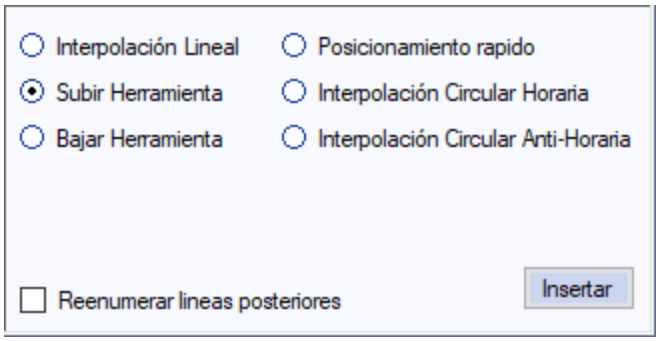
Interpolación lineal. Inserta en la tabla de códigos un nuevo código para dibujar una línea recta. Los parámetros de este código son la velocidad de traslación y posición final de la línea.



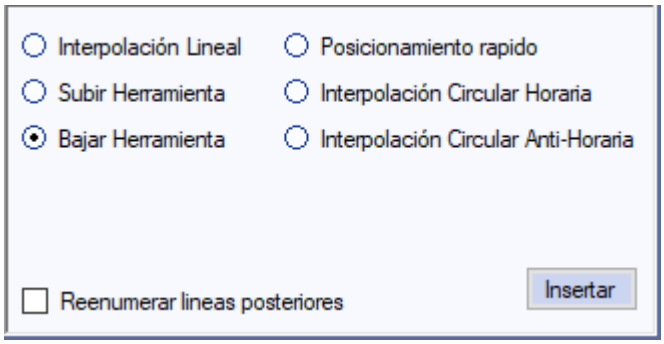
Interpolación Circular Horaria. Inserta en la tabla de códigos un nuevo código para dibujar una curva en sentido horario. Los parámetros de este código son la velocidad de traslación, la posición final de la curva y la posición del centro de rotación.



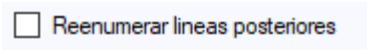
Interpolación Circular Antihoraria. Inserta en la tabla de códigos un nuevo código para dibujar una curva en sentido antihorario. Los parámetros de este código son la velocidad de traslación, la posición final de la curva y la posición del centro de rotación.



Subir herramienta. Inserta en la tabla de códigos un nuevo código que obliga al Brazo herramienta a elevarse en caso de que esté posado sobre la mesa.



Bajar herramienta. Inserta en la tabla de códigos un nuevo código que obliga al Brazo herramienta a posarse sobre la mesa en caso de que esté elevado.

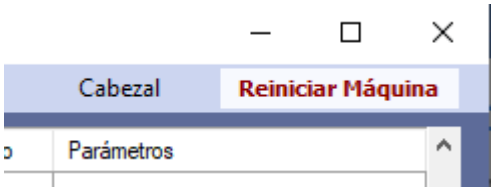


Método de inserción de la opción

Este control complementa a los anteriores. Si está activado además de insertarse el código en la tabla de códigos, se reenumerarán todos los números de línea de los comandos de las filas posteriores.

.

A4.1.9.- Menú superior izquierdo.

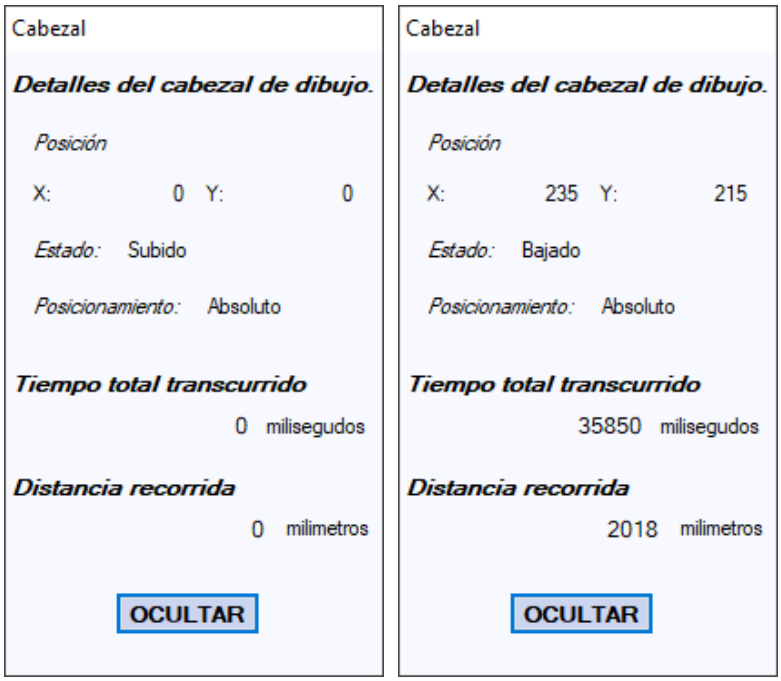


Posee un par de controles que por sus características no se pueden agrupar con los demás controles principales de la aplicación.

Reiniciar Máquina: Al accionar este control se obliga a volver al estado inicial a la aplicación (o sea, a reiniciar la máquina) eliminando tanto el programa que se tenga abierto como las modificaciones en la mesa de trabajo.

Cabezal: Este control hace visible el panel de detalles del Brazo herramienta.

A4.1.10.- Información del cabezal



Como se observa en las dos imágenes de ejemplo de arriba, el panel de detalles del Brazo herramienta se compone de las siguientes zonas de información:

Posición: Detalla la posición actual del Brazo herramienta sobre la mesa.

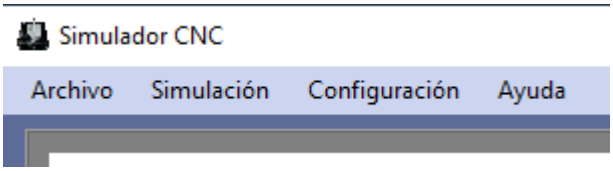
Estado: Indica la situación del Brazo (bajado o subido).

Posicionamiento: Define la configuración interna que tiene actualmente el brazo para el posicionamiento de este. Si es Absoluto, el Brazo herramienta tratará las coordenadas como coordenadas absolutas. Si es Relativo, tratará las coordenadas de entrada como valores a sumar a los de la coordenada anterior.

Tiempo total transcurrido: Corresponde al tiempo supuesto (en milisegundos) en que tardaría una máquina real en ejecutar todos los pasos hasta el actual.

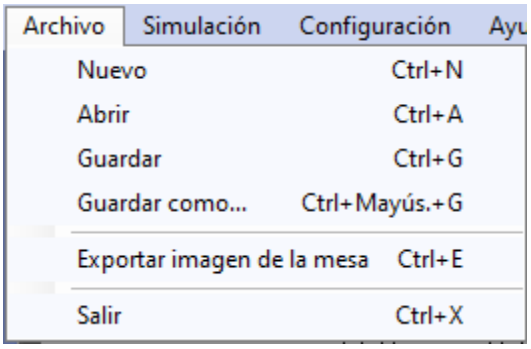
Distancia recorrida: Corresponde a la distancia total que ha ido recorriendo el Brazo herramienta en el transcurso de la ejecución de los códigos G-code hasta el actual.

A4.1.11.- Menú superior derecho

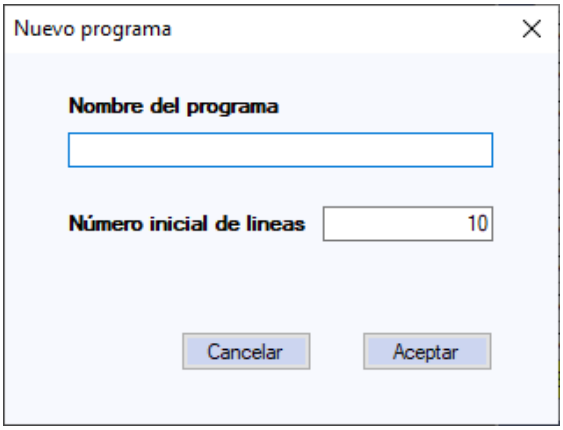


El menú superior derecho es aquel que contiene la mayor parte de las opciones de la aplicación y las típicas de cualquier programa que no son mayormente usadas.

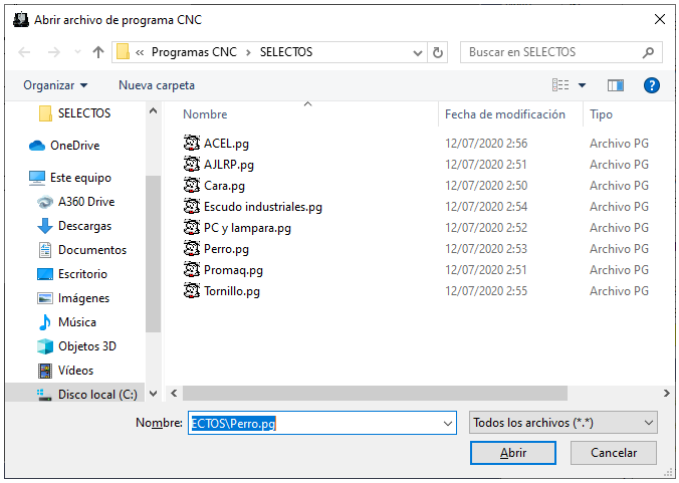
A4.1.12.- Menú Archivo



El menú Archivo contiene todas las opciones que derivan en interacción con el sistema para la lectura o escritura de archivos.

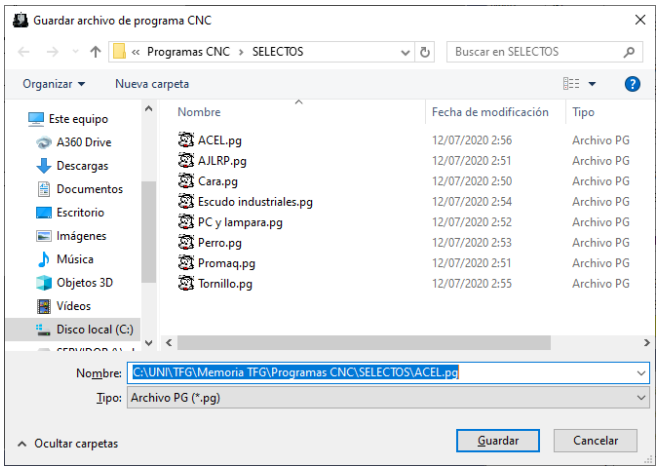


Nuevo: Permite al usuario crear un nuevo programa definiendo el nombre y el número de filas iniciales. Durante la ejecución de un programa este control se encuentra desactivado. También se puede llamar al control presionando al mismo tiempo las teclas Ctrl y N.

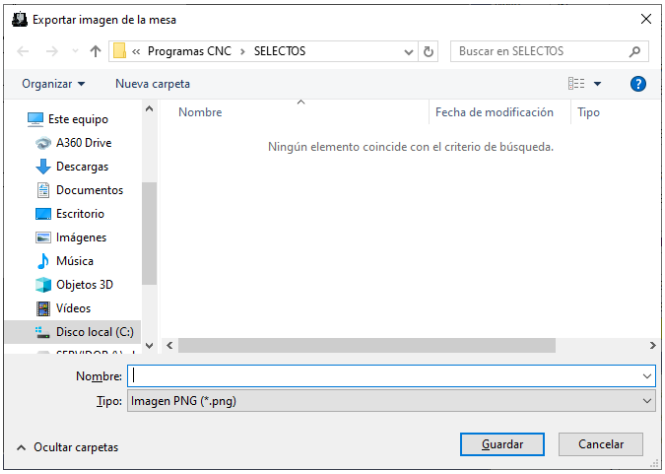


Abrir: Nos da la opción de seleccionar un fichero para abrirlo con el programa. Durante la ejecución de un programa este control se encuentra desactivado. También se puede llamar al control presionando las teclas Ctrl y A.

Guardar: Guarda el programa abierto en la máquina directamente sobre el archivo del que fue cargado inicialmente, si era un programa nuevo, guarda un nuevo archivo con el nombre del programa y la extensión PG. Se puede llamar al control directamente presionando al mismo tiempo las teclas Ctrl y G.



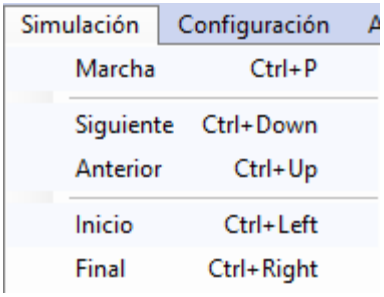
Guardar como: Muestra un cuadro de dialogo en el que nos permite insertar el nombre del archivo en el que queremos guardar el programa CNC abierto. Las extensiones permitidas para guardar el archivo son PG y TXT. Se puede llamar al control directamente presionando al mismo tiempo las teclas Ctrl, mayúsculas y G.



Exportar imagen de la mesa: Muestra un cuadro de dialogo en el que nos permite insertar el nombre del archivo en el que queremos guardar una captura del diseño actual que se encuentra sobre la mesa de trabajo. Los formatos de imagen soportados son PNG, GIF y BMP. Se puede llamar al control directamente presionando al mismo tiempo las teclas Ctrl y E.

Salir: La funcionalidad de este control está bastante clara, obliga a la aplicación a finalizar y liberar la memoria ocupada en el sistema operativo.

A4.1.13.- Menú Simulación



Este menú contiene los mismos controles de ejecución que se han explicado anteriormente. solo que se encuentran en formato de menú, estos controles son equivalentes a los anteriores mencionados.

Marcha: Equivale al control ***PLAY***. Cuando la aplicación se encuentra en ejecución de un programa de mecanizado esta opción desaparece y surge en su lugar la opción ***Paro***. Este control puede ser lanzado pulsando las teclas Ctrl y P.

Paro: Equivale al control ***STOP***. Esta opción solo se muestra cuando la aplicación se encuentra en ejecución de un programa de mecanizado, desapareciendo cuando se interrumpe la ejecución del programa surgiendo en su lugar la opción ***Marcha***. Este control puede ser lanzado pulsando las teclas Ctrl y S.

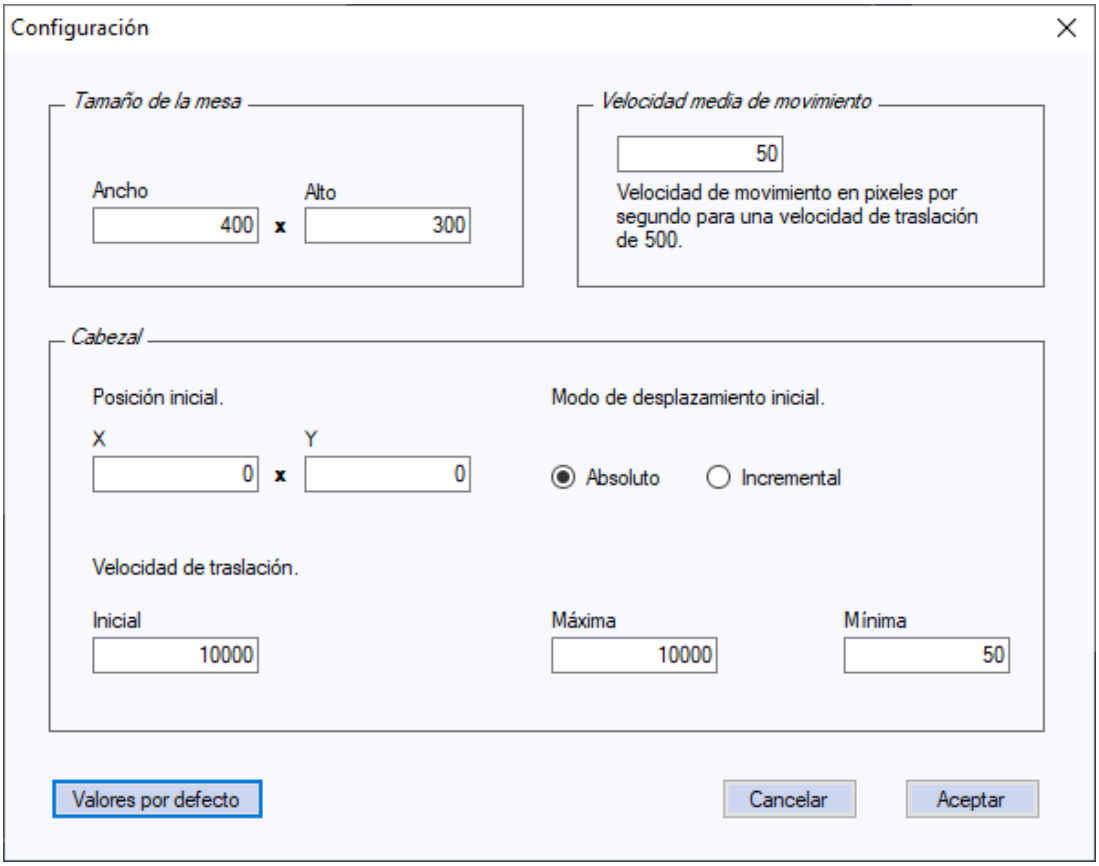
Siguiente: Equivale al control ***Avanzar un paso***. Este control puede ser lanzado pulsando las teclas Ctrl y Flecha para abajo.

Anterior: Equivale al control ***Retroceder un paso***. Este control puede ser lanzado pulsando las teclas Ctrl y Flecha para arriba.

Inicio: Equivale al control ***Retroceder hasta el inicio***. Este control puede ser lanzado pulsando las teclas Ctrl y Flecha hacia la izquierda.

Final: Equivale al control ***Avanzar hasta el final***. Este control puede ser lanzado pulsando las teclas Ctrl y Flecha hacia la derecha.

A4.1.14.- Menú de Configuración



Este menú no despliega ningún submenú, sino que abre una nueva ventana desde donde vamos a poder configurar diversas propiedades de la máquina virtual. Los distintos grupos de elementos que se pueden configurar desde esta nueva ventana son:

Tamaño de la mesa: Permite configurar el ancho y el alto de la mesa en pixeles.

Velocidad media de movimiento: Permite configurar la velocidad media simulada que supuestamente tendría la máquina real a la que se intenta imitar.

Cabezal: Permite configurar los elementos propios del Brazo herramienta, como la posición inicial, el modo de desplazamiento, etc.

A4.2.- Modos de simulación.

En este apartado se desarrolla los dos sistemas de simulación que posee la aplicación y que he ido comentando a lo largo de todo el documento.

Actualmente esta aplicación posee dos sistemas de simulación, uno en tiempo real y otra funcionando paso por paso.

Por defecto se encuentra habilitada la simulación en tiempo real, pero se puede cambiar a la simulación paso a paso a partir del control ***Simulación activa / Simulación pasiva***

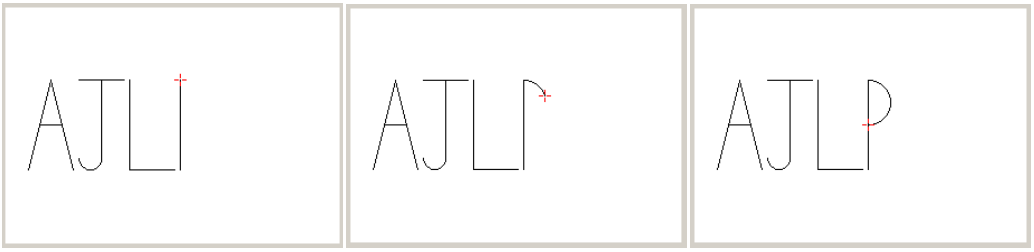
.

El cambio de un sistema a otro puede realizarse incluso cuando se encuentra en fase de simulación. El funcionamiento de cada uno de estos sistemas de simulación es:

A4.2.1.- Modo de simulación en tiempo real

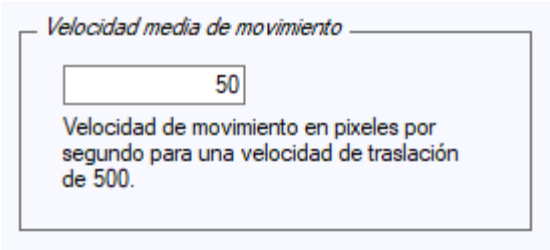
Aunque se denomine simulación en tiempo real, solamente se cumple esto cuando el ordenador donde se ejecuta la aplicación posee los recursos suficientes para que no se vea mermado el tiempo que necesita para realizar la simulación.

Principalmente, este tipo de simulación permite al usuario observar al Brazo herramienta desplazándose dinámicamente a través de la mesa de forma idéntica a como lo haría un Brazo herramienta de una máquina real. Igualmente, también es dinámico el efecto de subir o posar el brazo sobre la mesa.



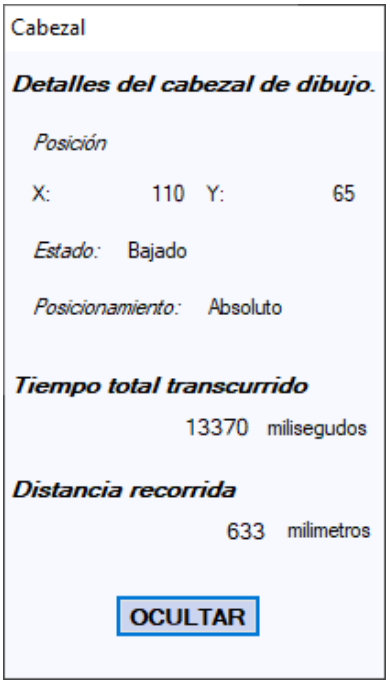
Al mismo tiempo que se desplaza, también se va actualizando la mesa de trabajo dinámicamente según vaya dibujando el Brazo.

La velocidad de traslación del Brazo herramienta es ajustable desde el panel de configuración.



Por defecto esta configuración se encuentra ajustada a 50 pixeles (o milímetros) por segundo cuando la velocidad de traslación que se define es de 500, por tanto, un código que solicite una velocidad de traslación del Brazo herramienta de 10000 obligará al Brazo de la simulación a que avance a 1000 pixeles por segundo.

Al mismo tiempo que se va realizando la simulación, se van actualizando dinámicamente los valores de la ventana de Detalles del Cabezal.



Aunque se ralentice la simulación, el cálculo del ***Tiempo total transcurrido*** no se verá afectado ya que se obtiene el tiempo a partir de cálculos que se realizan sobre el tiempo que tardaría en realizar ese mismo movimiento en la máquina real.

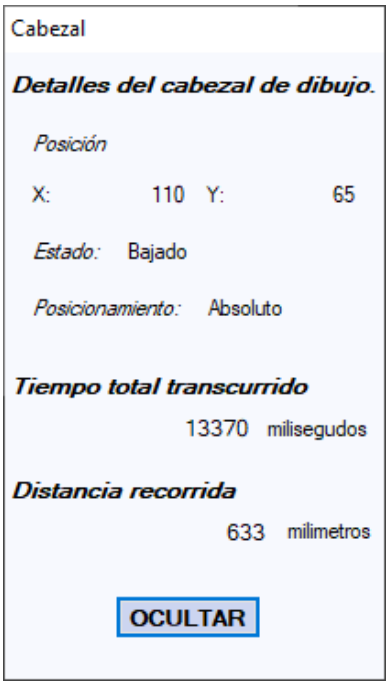
A4.2.2.- Modo de simulación por pasos (depuración)

Este otro sistema de simulación, aunque no muestra el mismo dinamismo que el anterior sistema, necesita de muchos menos recursos gráficos para realizar la simulación que el anterior, por lo que este sistema es preferible a utilizar en ordenadores con recursos muy escasos.

Además, al no realizar simulación con movimiento, el tiempo medio que tarda este sistema también es menor, por lo que si se quiere ir observando el progreso del programa sin necesidad de tener que observar los movimientos del Brazo herramienta, este sistema es el mejor.

Con la simulación paso a paso, el usuario observa directamente el resultado de la ejecución de cada código CNC en cada paso, con intervalos entre paso y paso de un segundo. En cada actualización el Brazo herramienta y la mesa de trabajo se actualizan con el resultado de tal código ejecutado en ese momento.

Por cada paso que se da en la simulación, igualmente se actualizan los valores de la ventana de Detalles del Cabezal.

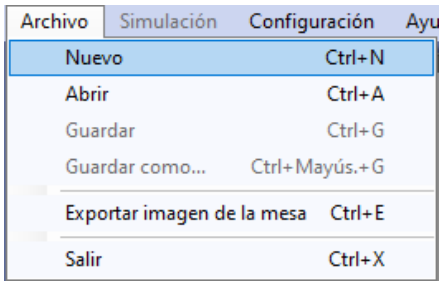


A4.3.- Creación edición y almacenamiento de programas de mecanizado.

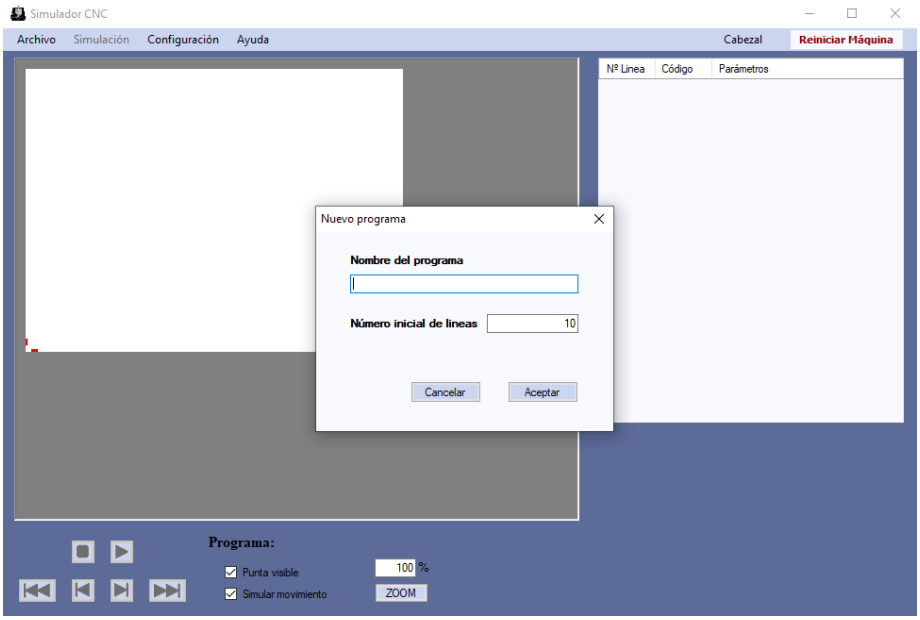
Este apartado profundiza en todas las funciones que posee la aplicación que ayudan a la edición o creación de nuevo código G-code directamente desde el Simulador CNC.

A4.3.1.- Creación de un nuevo programa

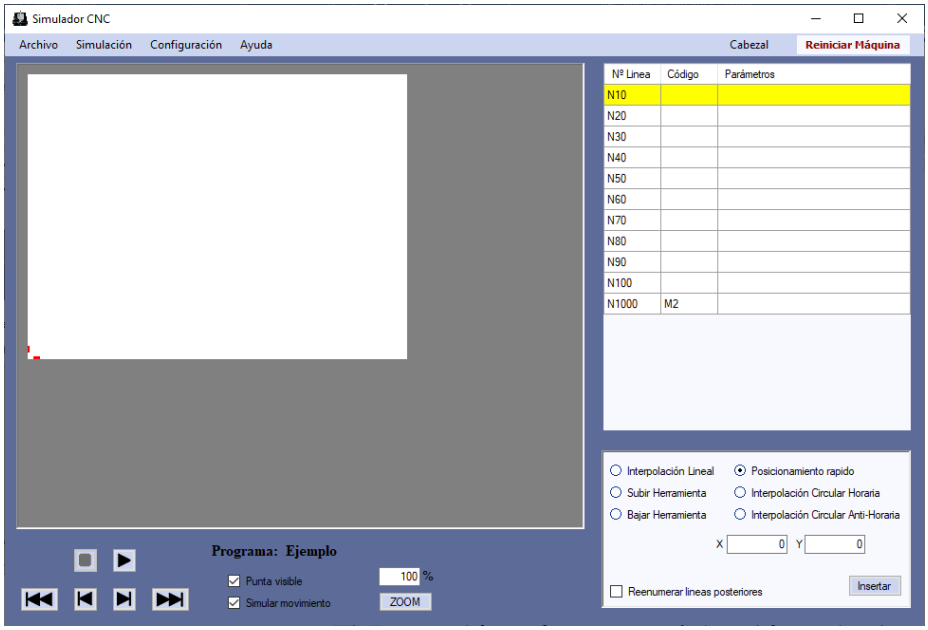
Para crear un nuevo programa se utiliza el sistema que se ha desarrollado en el submenú ***Nuevo*** dentro del menú Archivo.



Accediendo a este submenú se nos muestra la siguiente página solicitando un nombre para nuestro nuevo programa y el número de líneas iniciales que queremos que posea.



Una vez especificado, al aceptar esta ventana ya se puede observar el nuevo programa preparado en la máquina virtual.



Al generar las líneas solicitadas estas siempre son numeradas en incrementos de diez unidades. Además, después de estas líneas siempre se va inserta una línea más con el código G-code M2 que pertenece al código de finalización del programa de mecanizado.

A4.3.2.- Edición de un programa de mecanizado

Existen varios métodos que posibilitan el realizar modificaciones e insertar nuevos códigos al programa CNC. Alguno de estos métodos se asemeja al que utiliza el programa de la máquina real (a la que simula nuestra aplicación), como por ejemplo el panel de inserción de códigos.

Siempre que se realicen modificaciones sobre el programa abierto en la aplicación, se activará el sistema de comprobación de la integridad del código, inhabilitando la ejecución del programa si este no pasa correctamente la verificación de que es correcto.

A4.3.3.- Edición directa en la tabla de códigos G-code.

Nº Linea	Código	Parámetros
N10		
N20		
N30		
N40		
N50		
N60		
N70		
N80		
N90		
N100		
N1000	M2	

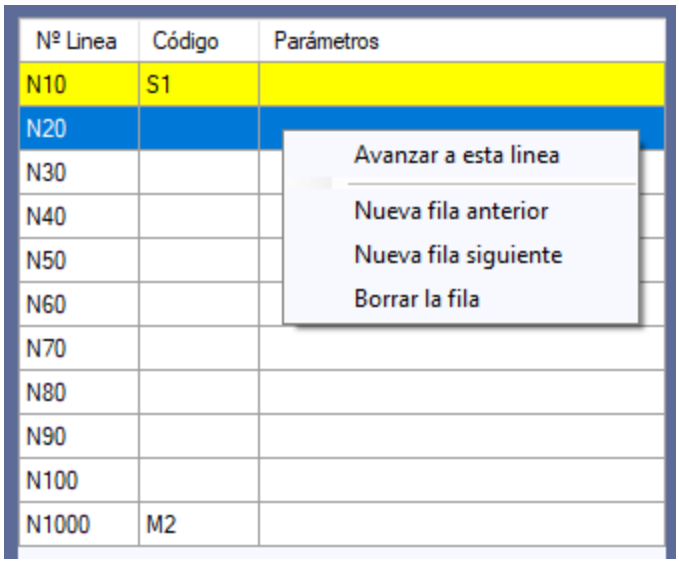
La tabla de códigos permite la edición directa sobre sus filas. Solamente hay que realizar una doble pulsación del ratón sobre la celda que queremos editar y esta misma se pondrá directamente en modo edición. Si queremos que esta misma celda vuelva a cambiar al estado de solo lectura no hay más que seleccionar otro elemento distinto sea cual sea.

Nº Linea	Código	Parámetros
N10	S1	
N20		
N30		
N40		
N50		
N60		
N70		
N80		
N90		
N100		
N1000	M2	

Si queremos que esta misma celda vuelva a cambiar al estado de solo lectura no hay más que seleccionar otro elemento distinto.

Cuando se selecciona una fila de la tabla de códigos se puede hacer uso del comando de teclado Ctrl y C para copiar el contenido de esta al portapapeles.

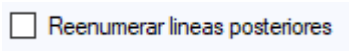
A4.3.4.- Menú derecho del ratón en la tabla de códigos G-code.



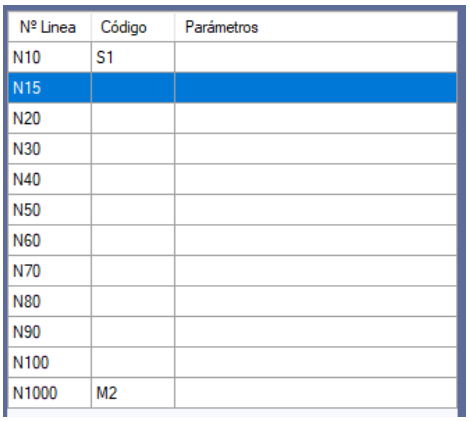
El menú derecho se ha desarrollado dentro del apartado A4.1.7.

Además de la opción de ***Avanzar a esta línea***, ofrece varias opciones interesantes para la creación o eliminación de filas.

Nueva fila anterior: Inserta una fila justo encima de la fila seleccionada desde donde se ha lanzado el menú derecho. Cuando se crea la nueva fila, el número de línea que se le asigna depende de la opción ***Método de inserción de la opción*** del panel de inserción de códigos.

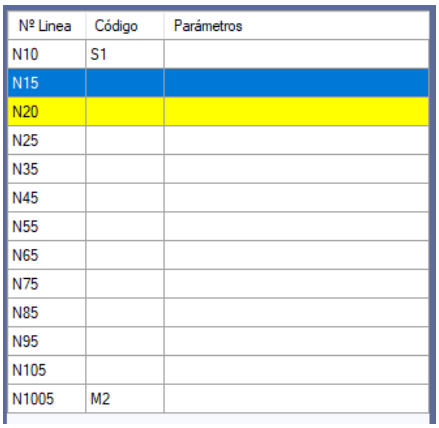


Si este se encuentra desactivado, el número de línea que se le asigna a la fila corresponde a la media entre la fila actual y la fila anterior.



Nº Linea	Código	Parámetros
N10	S1	
N15		
N20		
N30		
N40		
N50		
N60		
N70		
N80		
N90		
N100		
N1000	M2	

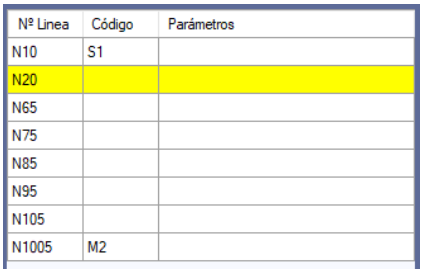
Si se encuentra activado, la aplicación primero comprueba la distancia numérica que tiene la fila seleccionada con la fila anterior, y una vez hallado esta, inserta la nueva fila con el número de línea de la línea anterior más la suma de la distancia numérica calculada. Además, todas las filas posteriores a la nueva fila son modificadas en su número de línea sumándoseles la distancia numérica hallada.



Nº Linea	Código	Parámetros
N10	S1	
N15		
N20		
N25		
N35		
N45		
N55		
N65		
N75		
N85		
N95		
N105		
N1005	M2	

Nueva fila siguiente: Esta opción es exactamente igual a la anterior excepto en la diferencia en que la nueva fila se crea debajo de la fila seleccionada donde se ha desplegado el menú derecho y no en la parte superior.

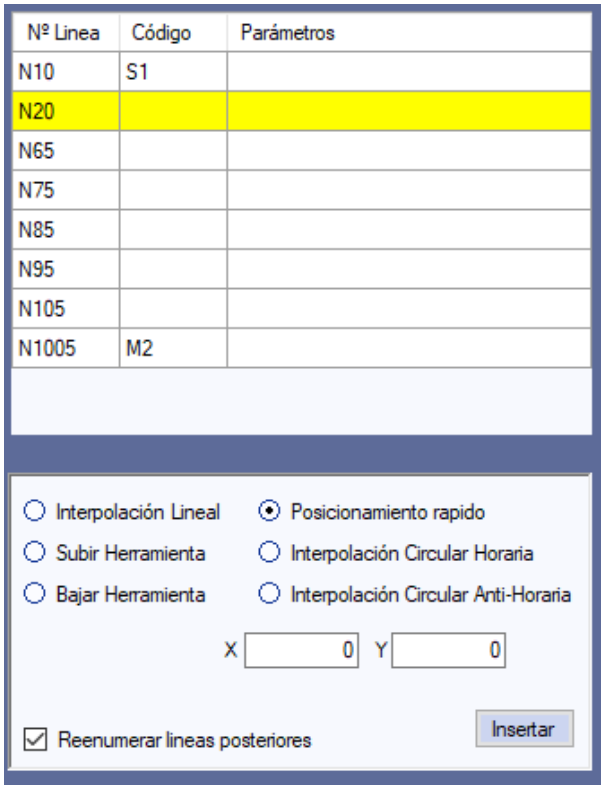
Borrar la fila: Esta opción elimina la fila seleccionada en la que se ha desplegado el menú derecho. Simplemente la elimina, no necesita modificar ningún elemento como el número de línea de las demás filas.



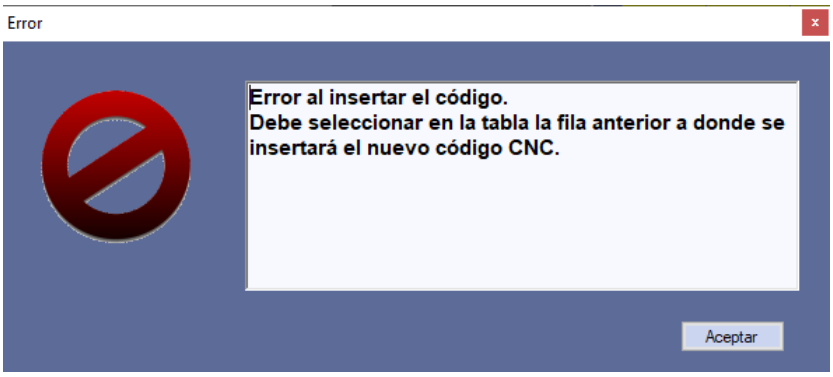
Nº Linea	Código	Parámetros
N10	S1	
N20		
N65		
N75		
N85		
N95		
N105		
N1005	M2	

A4.3.5.- Control de inserción de códigos G-code.

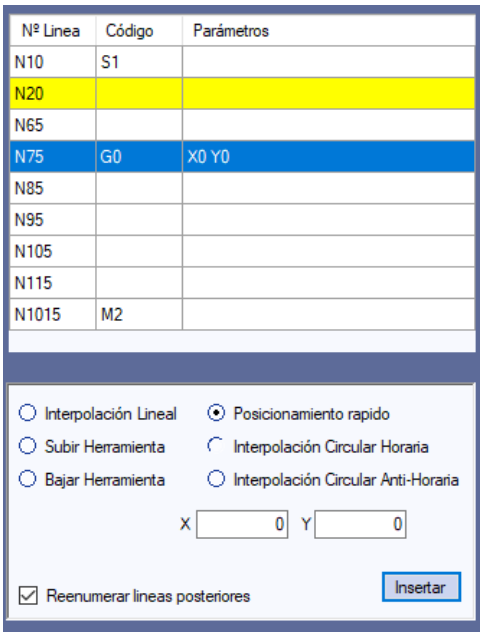
En el apartado A4.1.8 se ha desarrollado todas las opciones de este control. En este apartado se explicará cómo se insertan los códigos que hemos configurado desde este al programa de mecanizado.



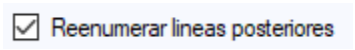
Para poder insertar en la tabla un código que hemos configurado en el panel de inserción es necesario haber seleccionado antes en esta tabla la línea en donde queremos insertarlo. En caso contrario nos mostrará un aviso indicándolo.



Cuando ya se encuentre todo correcto se insertará la nueva fila en el lugar correspondiente, justo después de la fila que se ha seleccionado.

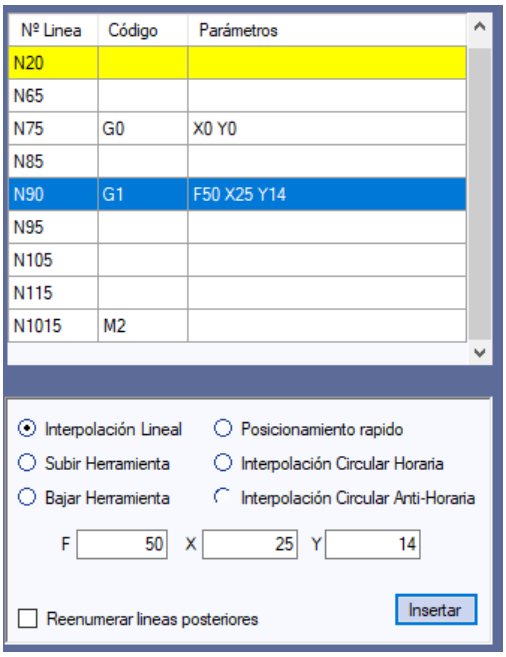


La forma que tiene de calcular el número de línea para la nueva fila es el mismo que he explicado en el apartado anterior del menú derecho. El método de cálculo depende de la opción ***Método de inserción de la opción*** del panel de inserción de códigos.



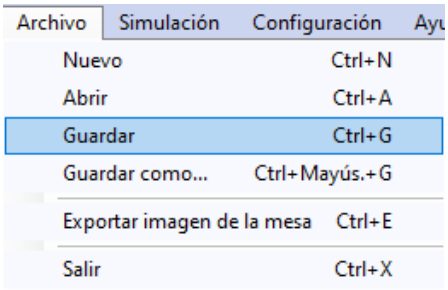
Si este se encuentra desactivado, el número de línea que se le asigna a la fila corresponde a la media entre la fila actual y la fila anterior.

Si este se encuentra activado, la aplicación primero comprueba la distancia numérica que tiene la fila seleccionada con la fila anterior, y una vez hallado esta, inserta la nueva fila con el número de línea de la línea anterior más la suma de la distancia numérica calculada. Además, todas las filas posteriores a la nueva fila son modificadas en su número de línea sumándoseles la distancia numérica hallada.



A4.3.6.- Almacenamiento del programa de mecanizado

Como es normal, si esta aplicación tiene funciones que permiten la modificación del programa igualmente tiene que poseer algún sistema que permita poder almacenar el resultado.



Desde el menú superior Archivo se pueden observar los dos comandos que nos van a permitir guardar el programa CNC abierto en la aplicación. Estos comandos se pueden lanzar incluso cuando la aplicación se encuentra en plena ejecución del programa.

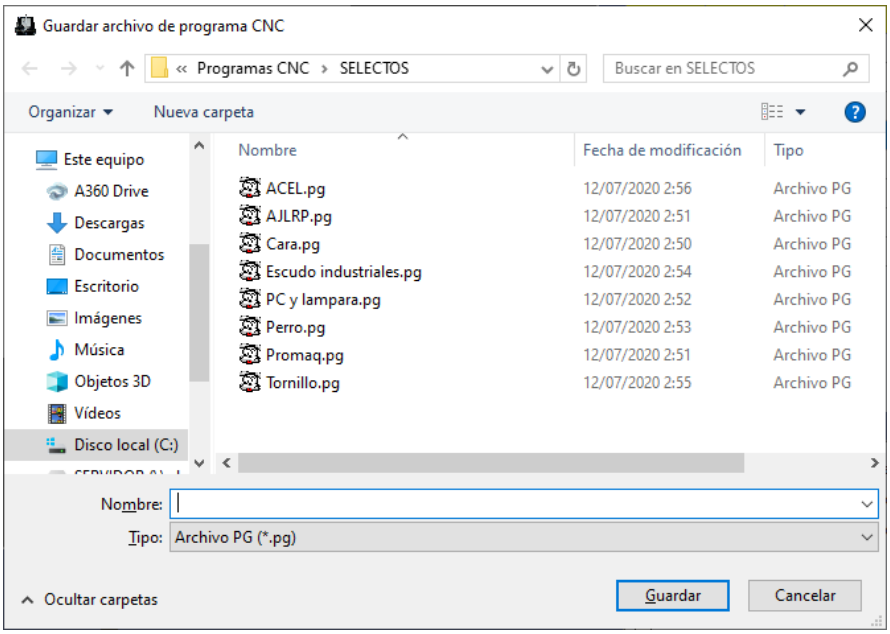
Guardar: Esta opción se ha definido en el apartado A4.1.12, realiza la operación de almacenamiento sobre el mismo archivo que fue abierto por la aplicación.

Si no se abrió ningún archivo, sino que se creó con la opción ***Nuevo***, la aplicación crea un nuevo archivo con el nombre del programa y extensión PG, guardando los datos del programa dentro de este.

En este segundo caso, la ruta del sistema en donde se almacenará el archivo directamente será la última a la que se ha accedido desde la aplicación. Si no se ha accedido al sistema de archivos la ruta será la misma que aquella en la que se encuentra la aplicación.

A esta opción también se puede acceder directamente pulsando simultáneamente las teclas Ctrl y G.

Guardar como…: Esta opción muestra el cuadro de dialogo típico para que podamos seleccionar en donde, con que nombre y en que formato quiero que se guarde el archivo.

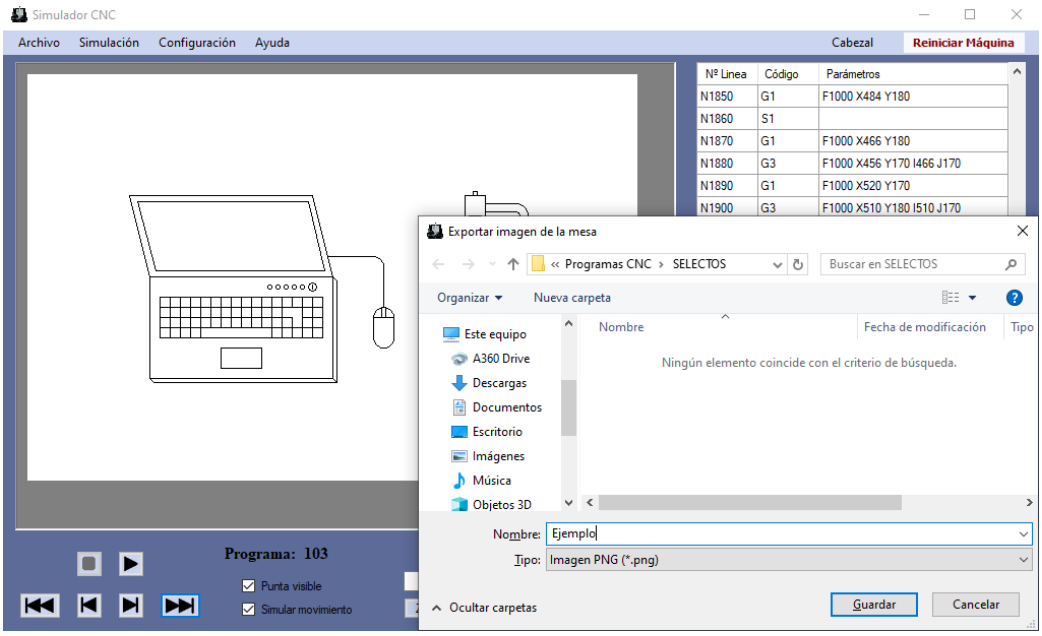


El tipo de formato que se puede configurar en este caso es PG o TXT, sin embargo, internamente estos dos tipos tienen los mismos datos en tipo ASCII, lo único que los diferencia son las siglas de la extensión.

A4.3.7.- Exportar el área de mecanizado a imagen.

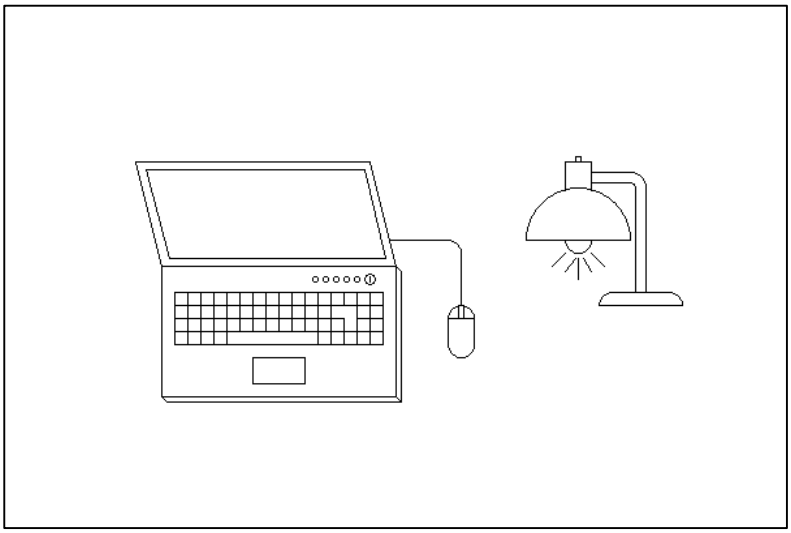
Además de poder guardar el programa CNC modificado, la aplicación también ofrece la posibilidad de guardar una copia de la imagen actual de la mesa de trabajo, incluso si la aplicación se encuentra en ejecución del programa CNC.

Exportar imagen de la mesa: Esta opción muestra también un cuadro de dialogo en el que puedo seleccionar en donde, con que nombre y en que formato quiero que se guarde el archivo.



Los tipos de formatos que pueden ser elegidos son PNG, GIF y BMP. El formato PNG soporta hasta 16 millones de colores, el formato GIF soporta hasta 256 colores y el formato BMP soporta 16 millones de colores.

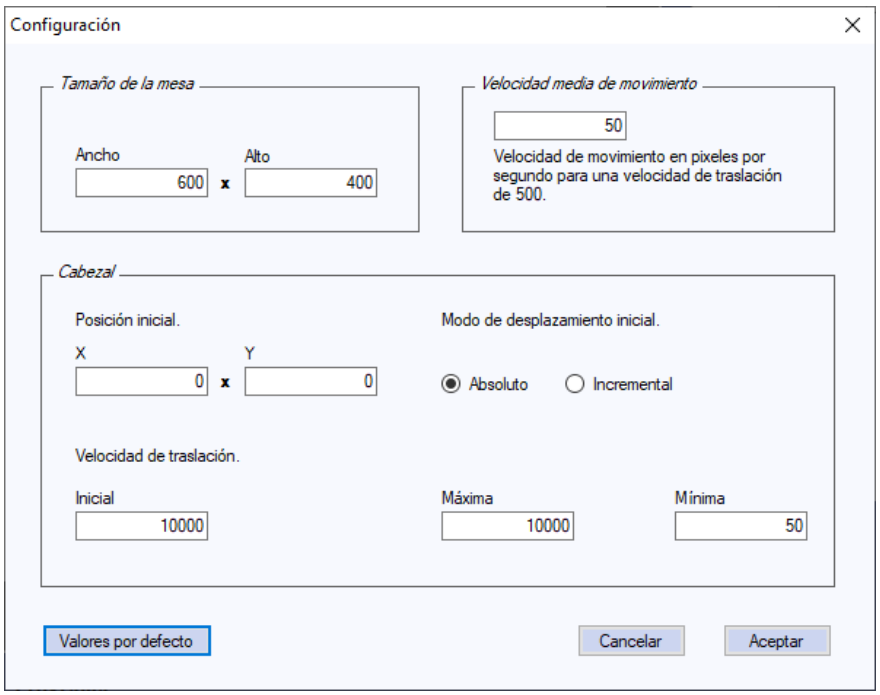
Un ejemplo de imagen exportada de la mesa puede ser este (el recuadro delimitador no es parte de la imagen, se ha añadido para que se aprecie los límites de la misma):



A4.4.- Configuración del programa Simulador CNC.

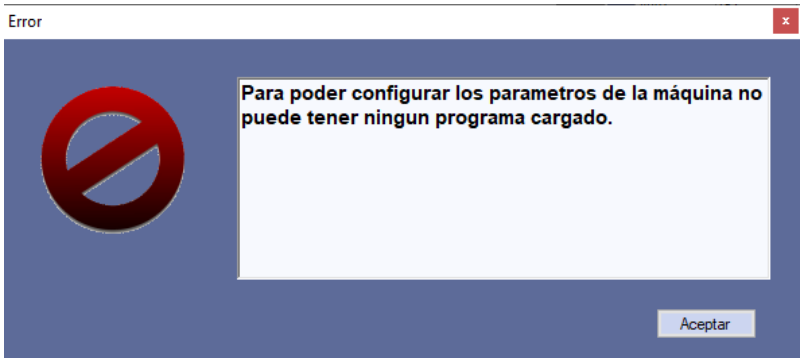
Para que el Simulador CNC pueda ser más versátil se ha dotado de distintos elementos de configuración que permiten modificar los diversos parámetros de la máquina virtual, además de otros.

A4.4.1.- Panel de configuración del programa



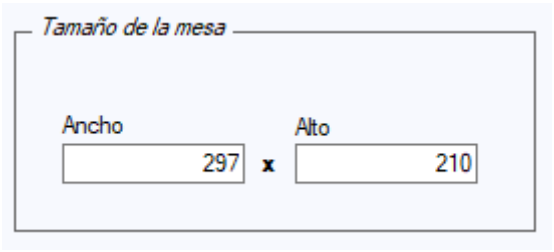
Como se puede ver en la imagen, este panel solamente permite la configuración de parámetros referentes a la máquina virtual, pero no contiene parámetros de configuración respecto al software de la aplicación. Eso es debido a que la configuración está planteada para que sea sencilla para ser usada por alumnos, siendo innecesario modificar elementos tales como el color de la mesa o del brazo que no les va a hacer más que molestar.

Para poder acceder a este panel es necesario que no haya ningún programa abierto en la máquina, en caso contrario se muestra el siguiente mensaje.



Cuando se acepta el panel con la nueva configuración se crea un archivo (SimuladorCNC.cfg) en la ruta de la aplicación que contiene todos los valores de configuración almacenados para que puedan ser recuperados en la siguiente ejecución de la aplicación.

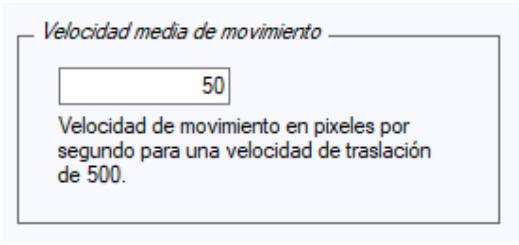
Los distintos parámetros que podemos configurar desde este panel son:



Tamaño de la mesa

En este elemento podemos definir el tamaño total que quiero que tenga la mesa de trabajo. La proporción la podemos contar tanto en pixeles como en milímetros ya que en este caso son equivalentes.

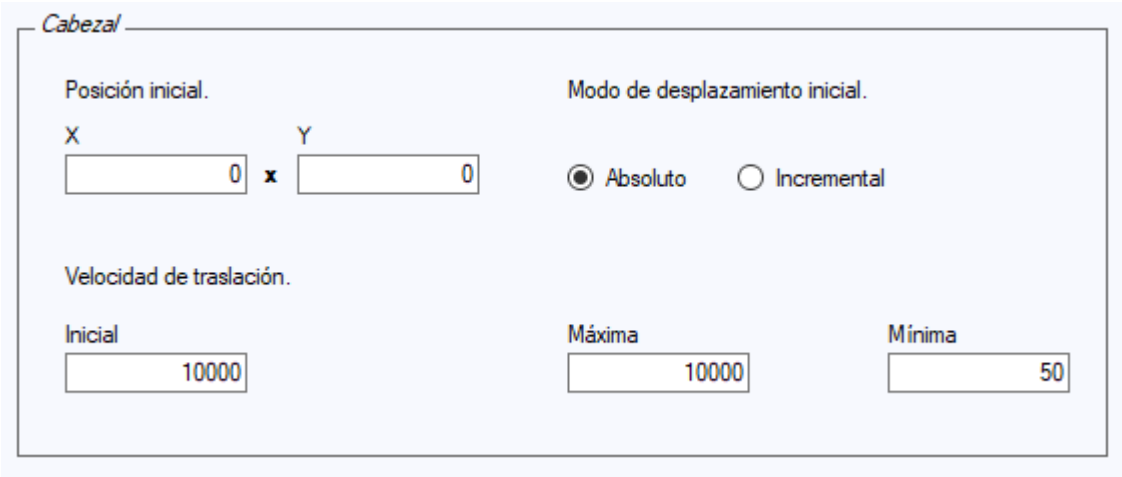
Los valores por defecto para el tamaño de la mesa son 297 mm de ancho por 210 mm de alto, los cuales corresponden al tamaño de una hoja DINA4, que es el tamaño de la mesa de dibujo que usa el prototipo de máquina real al que se simula por defecto**.**



Velocidad media de movimiento

De este elemento depende tanto la velocidad de simulación real como los cálculos del tiempo total invertido en la ejecución del programa. El valor por defecto es 50, que indica que para un movimiento de traslación de 500 la máquina dibujará 50 pixeles (o milímetros) por segundo. Para otros valores de traslación se calcula la proporción con respecto a este.

A4.4.2.- Configuración del cabezal



Esta zona agrupa todos los parámetros del cabezal (Brazo herramienta) que son configurables.

Posición inicial: Configura la posición en que se va a encontrar el Brazo herramienta cuando se inicie la aplicación o cuando se reinicie la máquina virtual. Por defecto es la posición base 0x0.

Modo de desplazamiento inicial: Configura la forma de interpretar las coordenadas de entrada al Brazo herramienta.

Las opciones son **Absoluto** que interpreta las coordenadas como coordenadas completas e **Incremental** que interpreta las coordenadas como coordenadas relativas que deben calcularse a partir de la coordenada anterior. En el código G-code existen varios códigos para cambiar esta configuración inicial del Brazo en tiempo de simulación.

Velocidad de traslación inicial: Configura dentro del Brazo la velocidad inicial a la que se va a mover, esta será válida mientras no se usen códigos G-code que modifiquen la velocidad de traslación.

Velocidad de traslación máxima: Define sobre la máquina virtual cual va a ser la velocidad máxima de traslación a la que va a poder desplazarse el Brazo herramienta.

Velocidad de traslación mínima: Define sobre la máquina virtual cual va a ser la velocidad mínima de traslación a la que va a poder desplazarse el Brazo herramienta.

A4.4.3.- Archivo de configuración (SimuladorCNC.cfg)

Este archivo de configuración **SimuladorCNC.cfg** se genera automáticamente al aceptar por primera vez el panel de configuración, destinado a recordar la última configuración guardada en la anterior sesión de la aplicación para volverla a recuperar en la siguiente. Si se quiere volver a la configuración por defecto no se tiene más que borrar este archivo.

En el caso de que el programa se encuentre en un soporte que imposibilite la creación o apertura de este archivo para escritura (por ejemplo CD-ROM), al aceptar el panel de configuración se mostrará un mensaje de advertencia indicando que no se ha podido guardar la configuración en disco y que la nueva configuración solamente permanecerá hasta que sea cerrada la aplicación.

Cada línea del archivo de configuración refleja un atributo de la aplicación, la configuración de atributos desde este fichero es más amplia que desde el panel de configuración de la propia aplicación:

Las claves de configuración del archivo son:

Ancho_Mesa 297
Define el tamaño del ancho de la mesa que se ha visto en el panel de configuración.

Alto_Mesa 210
Define el tamaño del alto de la mesa que se ha visto en el panel de configuración.

Inicio_Puntero_X 0
Define la posición X inicial del Brazo herramienta que se ha visto antes en el panel de configuración.

Inicio_Puntero_Y 0
Define la posición Y inicial del Brazo herramienta que se ha visto antes en el panel de configuración.

Puntero_Traslacion_Inicio 10000
Define la velocidad de movimiento inicial del Brazo herramienta que se ha visto en el panel de configuración.

Puntero_Traslacion_Maximo 10000
Define la velocidad de movimiento máxima a la que puede moverse el Brazo herramienta, la cual se ha visto en el panel de configuración.

Puntero_Traslacion_Minimo 50
Define la velocidad de movimiento mínima a la que puede moverse el Brazo herramienta, la cual se ha visto en el panel de configuración.

Inicio_Puntero_Absoluto True
Indica la configuración interna del Brazo herramienta para las coordenadas entrantes (true igual a **Absoluto**, false igual a **Incremental**), la cual se ha visto en el panel de configuración.

Pixeles_A_500_Traslacion_Por_Segundo 50
Define la velocidad de simulación real y el cálculo del tiempo en ejecución que se ha visto en el panel de configuración.

Simular True
Define el tipo de simulación por defecto que se aplica cuando se inicia el programa o se reinicia la máquina virtual. Si este es True la simulación es real, si es False la simulación es paso a paso. Por defecto este está a True.

Punta_Visible True
Indica si como configuración por defecto se va a representar el Brazo herramienta sobre la mesa o no. Esta configuración se aplica cuando se inicia el programa o se reinicia la máquina virtual. Si esta es True el Brazo herramienta será visible, si es False no se mostrará el Brazo herramienta. Por defecto esta está a True.

Altura_Cabezal 4
Define la altura máxima a la que puede elevarse el Brazo herramienta desde la mesa. Esta configuración solo permite números positivos pares. Por defecto el valor de elevación es 4.

Un ejemplo de esta configuración a distintos valores es:

Altura máxima 4

Altura máxima 8


Altura máxima 16

Está claro que cuanto más tenga que subir y bajar el Brazo herramienta más tiempo desperdiciará en ello.

Milisegundos_Por_Iteracion_Subir_Cabezal 34
Define la velocidad que va a tardar en subir o bajar el Brazo herramienta en cada iteración del procedimiento. El valor por defecto es de 34 milisegundos por iteración.

Tiempo_De_Paso_En_Estatico 1000
Define en milisegundos el tiempo que va a tardar de pasar desde la ejecución de un código al siguiente en simulación paso a paso. El valor por defecto es de 1000 milisegundos, o sea, 1 segundo.

Color_Paso_En_Tabla A: 255 R: 255 G: 255 B: 0
Indica en formato de color RGBA, el color que va a ser usado para marcar la posición actual del programa en ejecución en la tabla de códigos. El valor por defecto es R:255 G:255 B:0 A:255 (amarillo).

Un ejemplo de esta configuración a distintos valores es:

Nº Linea	Código	Parámetros
N10	G90	
N20	S-1	
N30	G1	F1000 X0 Y0
N40	G1	F1000 X200 Y150
N50	S1	
N60	G2	F500 X200 Y130 I200 J140
N70	G2	F500 X200 Y150 I200 J140
N80	S-1	
N90	G1	F1000 X200 Y130

Color R:255 G:255 B:0 A:255

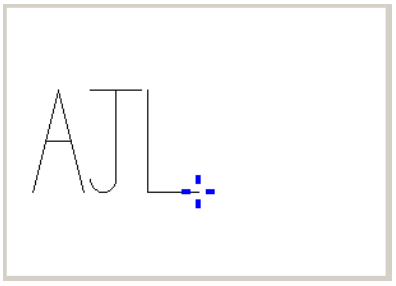
Nº Linea	Código	Parámetros
N10	G90	
N20	S-1	
N30	G1	F1000 X0 Y0
N40	G1	F1000 X200 Y150
N50	S1	
N60	G2	F500 X200 Y130 I200 J140
N70	G2	F500 X200 Y150 I200 J140
N80	S-1	
N90	G1	F1000 X200 Y130

Color R:255 G:0 B: 255 A:255

Color_Puntero A: 255 R: 255 G: 0 B: 0
Indica en formato de color RGBA, el color que va a ser usado para la representación del Brazo herramienta. El valor por defecto es R:255 G:0 B:0 A:255 (rojo).

Un ejemplo de esta configuración a distintos valores es:

Color R:255 G:0 B:0 A:255

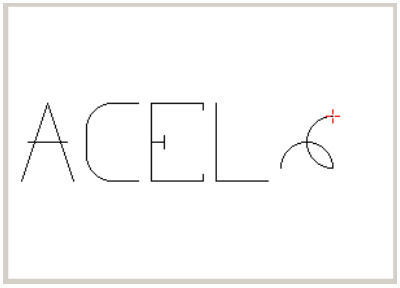


Color R:0 G:0 B: 255 A:255

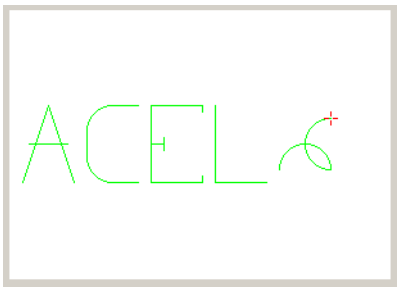
Color_Dibujo A: 255 R: 0 G: 0 B: 0

Indica en formato de color RGBA, el color que va a ser usado para la representación del dibujo sobre la mesa de trabajo. El valor por defecto es R:0 G:0 B:0 A:255 (negro).

Un ejemplo de esta configuración a distintos valores es:



Color R:0 G:0 B:0 A:255



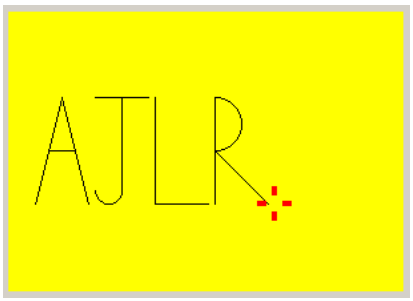
Color R:0 G:255 B:0 A:255

Color_Mesa A: 255 R: 255 G: 255 B: 255

Indica en formato de color RGBA, el color que va a ser usado para la representación del color base de la mesa de trabajo. El valor por defecto es R:255 G:255 B:255 A:255 (blanco).

Un ejemplo de esta configuración a distintos valores es:

Color R:255 G:255 B:255 A:255

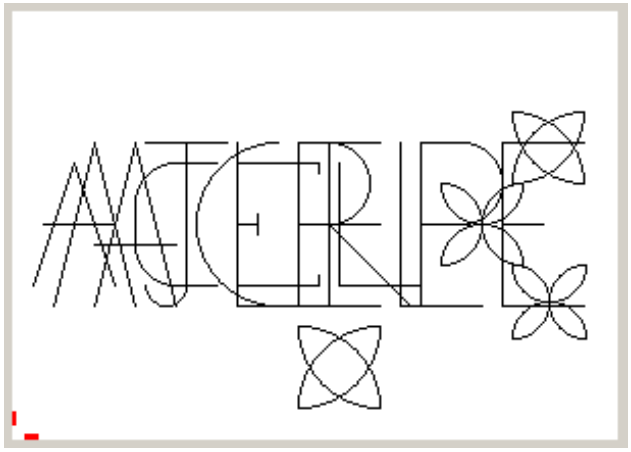


Color R:255 G:255 B:0 A:255

Limpiar_Mesa_Y_Cabezal True

Define si se va a limpiar la mesa de trabajo y reiniciar el Brazo herramienta cada vez que se carga un nuevo programa. Si la configuración es True, cada vez que se abre un programa se limpia la mesa y el Brazo vuelve a su configuración inicial, si es False se mantiene el estado de la mesa y la configuración actual del Brazo herramienta. El valor por defecto es True.

Ejemplo de la mesa con esta configuración a False después de ejecutar varios programas CNC seguidos:



Anexo 5.- Implementación del Simulador CNC.

El siguiente apartado contiene la implementación de la solución final desarrollada para este **Trabajo Final de Grado.**

Esta solución contiene las diversas clases desarrolladas en 6 grupos distintos, según la relación que hay entre clases

- **Control del programa.**
- **Mecanismos.**
- **Interfaz**
- **Menú**
- **Botones**
- **Gestión de errores**

A5.1.- Control De Programa.

El grupo **Control del programa** contiene 7 clases relacionadas con este grupo:

- **Control del programa.**
- **Configuración.**
- **Contador.**
- **Ejecución.**
- **Integridad.**
- **Tabla.**
- **Zoom.**

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using SimuladorCNC.Mecanismos;
using SimuladorCNC.Errores;
using SimuladorCNC.Menu;

namespace SimuladorCNC.ControlDePrograma
{
    class ControlDePrograma
    {
        String archivoPrograma = "";

        Principal front;
        tabla miTabla;
        Brazo miBrazo;
        Contador miContador;
        Boolean programaCargado;

        //comprobacion de codigo
        Boolean haceFaltaComprobar;

        integridad integrid;
        ejecucion ejecuc;

        public ControlDePrograma(Principal f)
        {
            front = f;
            miTabla=new tabla(front);
            miTabla.reiniciarTabla();
            miBrazo = f.getBrazo();
            miContador = new Contador(miBrazo, front);
            programaCargado = false;

            miTabla.setPaso(0, true);
            front.prog.Text = "Programa:  ";

            haceFaltaComprobar = false;
            ejecuc = new ejecucion(this, front, miTabla);
            integrid = new integridad(front, this, ejecuc, miTabla);
            miContador.reiniciarContador();
            sinPrograma();
        }

        public void sinPrograma()
        {
            programaCargado = false;
            front.getMarcha().desabilitarMarcha();
            archivoPrograma = "";
        }

        public void conPrograma()
        {
            programaCargado = true;
            front.getMarcha().habilitarMarcha();
        }

        public void reiniciarMaquina()
        {
            reiniciarMaquina(false);
        }

        public void reiniciarMaquina(Boolean desdeCargarNuevoPrograma)
        {
            miContador.reiniciarContador();
            front.getInsertarCodigo().ReiniciarInsertarCodigo();
            front.ControlSimulador.Enabled = false;
            front.TiempoPlay.Enabled = false;
            ejecuc.Reiniciado();
```

```
            miTabla.setPaso(0, true);
            miTabla.reiniciarTabla();
            front.prog.Text = "Programa:  ";
            if (!(!front.getConfig().getLimpiarMesaYBrazo() && desdeCargarNuevoPrograma))
            {
                front.getMesa().reiniciarMesa();
                miBrazo.reiniciarBrazo();
            }

            haceFaltaComprobar = false;
            sinPrograma();
        }

        public void HacerComprobar()
        {
            haceFaltaComprobar=true;
        }

        public bool Correcto()
        {
            if(haceFaltaComprobar)
            {
               return integrid.comprobarIntegridadCodigo();
            }
            return true;
        }

        public bool programaPreparado()
        {
            return programaCargado;
        }

        public integridad getIntegridad()
        {
            return integrid;
        }

        public ejecucion getEjecucion()
        {
            return ejecuc;
        }


        public Contador getContador()
        {
            return miContador;
        }

        public tabla getTabla()
        {
            return miTabla;
        }

        public bool getComprobar()
        {
            return haceFaltaComprobar;
        }
        public void setComprobar(bool C)
        {
            haceFaltaComprobar = C;
        }

        public void setArchivoPrograma(String a)
        {
            archivoPrograma = a;
        }

        public String getArchivoPrograma()
        {
            return archivoPrograma;
```

```
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Text;
using SimuladorCNC.Errores;

namespace SimuladorCNC.ControlDePrograma
{
    class Configuracion
    {
        String ruta;

        Principal front;

        //Valores por defecto
        int DefMesaX=297;
        int DefMesaY=210;

        int DefPunteroXIni=0;
        int DefPunteroYIni=0;
        int DefPunteroTraslacionIni=10000;
        int DefPunteroTraslacionMax = 10000;
        int DefPunteroTraslacionMin = 50;
        bool DefPunteroAbsolutoIni=true;

        int DefPixelesPorFrecuencia=50;

        bool DefSimular=true;
        bool DefBrazoVisible=true;

        int DefAlturaPuntero=4;
        int DefMilisPorIterSubidaPuntero=34;

        int DefTiempoPasoEnEstatico=1000;

        Color DefPasoDeTabla=Color.Yellow;

        Color DefColorPuntero=Color.Red;
        Color DefColorLinea=Color.Black;
        Color DefColorMesa=Color.White;

        Boolean DefLimpiarMesaYBrazo=true;

        //Valores
        int MesaX;
        int MesaY;

        int PunteroXIni;
        int PunteroYIni;
        int PunteroTraslacionIni;
        int PunteroTraslacionMax;
        int PunteroTraslacionMin;
        bool PunteroAbsolutoIni;

        int PixelesPorFrecuencia;

        //Valores avanzados
        bool Simular;
        bool BrazoVisible;

        int AlturaPuntero;
        int MilisPorIterSubidaPuntero;

        int TiempoPasoEnEstatico;

        Color PasoDeTabla;

        Color ColorPuntero;
        Color ColorLinea;
```

```
        Color ColorMesa;

        Boolean LimpiarMesaYBrazo; //En caso de falso el siguiente programa sigue por
            donde se quedo el otro anterior

        //valores finales
        const float VelocidadRefrescoPuntero=0.4F;
        //Para 0,2F se pueden poner todas las alturas pero va un poco lento
        //Para 0,4F va mas rapido pero solo se pueden poner alturas pares

        public Configuracion(Principal F)
        {
            ruta = Environment.CurrentDirectory;
            front = F;

            MesaX = DefMesaX;
            MesaY = DefMesaY;

            PunteroXIni = DefPunteroXIni;
            PunteroYIni = DefPunteroYIni;
            PunteroTraslacionIni = DefPunteroTraslacionIni;
            PunteroTraslacionMax = DefPunteroTraslacionMax;
            PunteroTraslacionMin = DefPunteroTraslacionMin;
            PunteroAbsolutoIni = DefPunteroAbsolutoIni;

            PixelesPorFrecuencia = DefPixelesPorFrecuencia;

            Simular = DefSimular;
            BrazoVisible = DefBrazoVisible;

            AlturaPuntero = DefAlturaPuntero;
            MilisPorIterSubidaPuntero = DefMilisPorIterSubidaPuntero;

            TiempoPasoEnEstatico = DefTiempoPasoEnEstatico;

            PasoDeTabla = DefPasoDeTabla;

            ColorPuntero = DefColorPuntero;
            ColorLinea = DefColorLinea;
            ColorMesa = DefColorMesa;

            LimpiarMesaYBrazo = DefLimpiarMesaYBrazo;

            cargarConfiguracion();
        }

        public int getMesaX(){return MesaX;}
        public void setMesaX(int M) { MesaX = M; }
        public int getMesaY(){return MesaY;}
        public void setMesaY(int M) { MesaY = M; }

        public int getPunteroXIni() { return PunteroXIni; }
        public void setPunteroXIni(int PI) { PunteroXIni = PI; }
        public int getPunteroYIni() { return PunteroYIni; }
        public void setPunteroYIni(int PI) { PunteroYIni = PI; }
        public int getPunteroTraslacionIni() { return PunteroTraslacionIni; }
        public void setPunteroTraslacionIni(int PT) { PunteroTraslacionIni = PT; }
        public int getPunteroTraslacionMax() { return PunteroTraslacionMax; }
        public void setPunteroTraslacionMax(int PT) { PunteroTraslacionMax = PT; }
        public int getPunteroTraslacionMin() { return PunteroTraslacionMin; }
        public void setPunteroTraslacionMin(int PT) { PunteroTraslacionMin = PT; }
        public bool getPunteroAbsolutoIni() { return PunteroAbsolutoIni; }
        public void setPunteroAbsolutoIni(bool A) { PunteroAbsolutoIni = A; }

        public bool getSimular() { return Simular; }
        public void setSimular(bool S) { Simular=S; }
        public bool getBrazoVisible() { return BrazoVisible; }
        public void setBrazoVisible(bool BV) { BrazoVisible = BV; }

        public int getAlturaPuntero() { return AlturaPuntero; }
```

```
        public void setAlturaPuntero(int AP) { AlturaPuntero=AP; }
        public int framesAlturaPuntero() { return (int) Math.Abs(AlturaPuntero /
            VelocidadRefrescoPuntero); }
        public int getMilisPorIterSubidaPuntero() { return MilisPorIterSubidaPuntero; }
        public void setMilisPorIterSubidaPuntero(int SM) { MilisPorIterSubidaPuntero=SM;
            }

        public int getTiempoPasoEnEstatico() { return TiempoPasoEnEstatico; }
        public void setTiempoPasoEnEstatico(int TPEE) { TiempoPasoEnEstatico=TPEE; }

        public Color getPasoDeTabla() { return PasoDeTabla; }
        public void setPasoDeTabla(Color PT) { PasoDeTabla=PT; }

        public Color getColorDelPuntero() { return ColorPuntero; }
        public void setColorDelPuntero(Color CT) { ColorPuntero=CT; }
        public Color getColorLinea() { return ColorLinea; }
        public void setColorLinea(Color CL) { ColorLinea=CL; }
        public Color getColorMesa() { return ColorMesa; }
        public void setColorMesa(Color CM) { ColorMesa=CM; }

        public Boolean getLimpiarMesaYBrazo() { return LimpiarMesaYBrazo; }
        public void setLimpiarMesaYBrazo(Boolean LMB) {LimpiarMesaYBrazo=LMB; }

        public int getPixelesPorFrecuencia() { return PixelesPorFrecuencia; }
        public void setPixelesPorFrecuencia(int PPF) { PixelesPorFrecuencia=PPF; }

        public float getVelocidadRefrescoPuntero() { return VelocidadRefrescoPuntero; }

        //obtener valores por defecto
        public int getDefMesaX() { return DefMesaX; }
        public int getDefMesaY() { return DefMesaY; }

        public int getDefPunteroXIni() { return DefPunteroXIni; }
        public int getDefPunteroYIni() { return DefPunteroYIni; }
        public int getDefPunteroTraslacionIni() { return DefPunteroTraslacionIni; }
        public int getDefPunteroTraslacionMax() { return DefPunteroTraslacionMax; }
        public int getDefPunteroTraslacionMin() { return DefPunteroTraslacionMin; }
        public bool getDefPunteroAbsolutoIni() { return DefPunteroAbsolutoIni; }

        public bool getDefSimular() { return DefSimular; }
        public bool getDefBrazoVisible() { return DefBrazoVisible; }

        public int getDefAlturaPuntero() { return DefAlturaPuntero; }
        public int getDefMilisPorIterSubidaPuntero() { return
            DefMilisPorIterSubidaPuntero; }

        public int getDefTiempoPasoEnEstatico() { return DefTiempoPasoEnEstatico; }

        public Color getDefPasoDeTabla() { return DefPasoDeTabla; }

        public Color getDefColorDelPuntero() { return DefColorPuntero; }
        public Color getDefColorLinea() { return DefColorLinea; }
        public Color getDefColorMesa() { return DefColorMesa; }

        public Boolean getDefLimpiarMesaYBrazo() { return DefLimpiarMesaYBrazo; }

        public int getDefPixelesPorFrecuencia() { return DefPixelesPorFrecuencia; }

        public int guardarConfiguracion()
        {
            try
            {
                String archivo = ruta+"\\SimuladorCNC.cfg";
                StreamWriter sw = new StreamWriter(archivo);

                sw.WriteLine("Ancho_Mesa " + MesaX.ToString());
                sw.WriteLine("Alto_Mesa " + MesaY.ToString());

                sw.WriteLine("Inicio_Puntero_X " + PunteroXIni.ToString());
                sw.WriteLine("Inicio_Puntero_Y " + PunteroYIni.ToString());
```

```
            sw.WriteLine("Puntero_Traslacion_Inicio " + PunteroTraslacionIni.ToString
                ());
            sw.WriteLine("Puntero_Traslacion_Maximo " + PunteroTraslacionMax.ToString
                ());
            sw.WriteLine("Puntero_Traslacion_Minimo " + PunteroTraslacionMin.ToString
                ());
            sw.WriteLine("Inicio_Puntero_Absoluto " + PunteroAbsolutoIni.ToString());

            sw.WriteLine("Pixeles_A_500_Traslacion_Por_Segundo " +
                PixelesPorFrecuencia.ToString());

            //Valores avanzados
            sw.WriteLine("Simular " + Simular.ToString());
            sw.WriteLine("Punta_Visible " + BrazoVisible.ToString());

            sw.WriteLine("Altura_Cabezal " + AlturaPuntero.ToString());
            sw.WriteLine("Milisegundos_Por_Iteracion_Subir_Cabezal " +
                MilisPorIterSubidaPuntero.ToString());
            sw.WriteLine("Tiempo_De_Paso_En_Estatico " + TiempoPasoEnEstatico.
                ToString());

            sw.WriteLine("Color_Paso_En_Tabla A: " + PasoDeTabla.A.ToString() + " R:
                " + PasoDeTabla.R.ToString() + " G: " + PasoDeTabla.G.ToString() + "
                B: " + PasoDeTabla.B.ToString());

            sw.WriteLine("Color_Puntero A: " + ColorPuntero.A.ToString() + " R: " +
                ColorPuntero.R.ToString() + " G: " + ColorPuntero.G.ToString() + " B:
                " + ColorPuntero.B.ToString());
            sw.WriteLine("Color_Dibujo A: " + ColorLinea.A.ToString() + " R: " +
                ColorLinea.R.ToString() + " G: " + ColorLinea.G.ToString() + " B: " +
                ColorLinea.B.ToString());
            sw.WriteLine("Color_Mesa A: " + ColorMesa.A.ToString() + " R: " +
                ColorMesa.R.ToString() + " G: " + ColorMesa.G.ToString() + " B: " +
                ColorMesa.B.ToString());

            sw.WriteLine("Limpiar_Mesa_Y_Cabezal " + LimpiarMesaYBrazo.ToString());
            sw.WriteLine("");

            sw.Flush();
            sw.Close();
        }
        catch
        {
            int error = 50;
            Error Er = new Error(error);
            Er.ShowDialog();
            return -error;
        }
        return 0;
    }

    public int cargarConfiguracion()
    {
        try
        {
            String archivo = ruta + "\\SimuladorCNC.cfg";
            String linea = "";
            String[] partes;
            StreamReader sr = new StreamReader(archivo);
            while ((linea = sr.ReadLine()) != null)
            {
                partes = linea.Split(' ');
                if (!((partes.Length == 1) && partes[0].Equals("")))
                {
                    switch (partes[0])
                    {
                        case "Ancho_Mesa":
                            MesaX = int.Parse(partes[1]);
                            break;
                        case "Alto_Mesa":
```

```
                            MesaY = int.Parse(partes[1]);
                            break;
                        case "Inicio_Puntero_X":
                            PunteroXIni = int.Parse(partes[1]);
                            break;
                        case "Inicio_Puntero_Y":
                            PunteroYIni = int.Parse(partes[1]);
                            break;
                        case "Puntero_Traslacion_Inicio":
                            PunteroTraslacionIni = int.Parse(partes[1]);
                            break;
                        case "Puntero_Traslacion_Maximo":
                            PunteroTraslacionMax = int.Parse(partes[1]);
                            break;
                        case "Puntero_Traslacion_Minimo":
                            PunteroTraslacionMin = int.Parse(partes[1]);
                            break;
                        case "Inicio_Puntero_Absoluto":
                            PunteroAbsolutoIni = bool.Parse(partes[1]);
                            break;
                        case "Pixeles_A_500_Traslacion_Por_Segundo":
                            PixelesPorFrecuencia = int.Parse(partes[1]);
                            break;
                        case "Simular":
                            Simular = bool.Parse(partes[1]);
                            break;
                        case "Punta_Visible":
                            BrazoVisible = bool.Parse(partes[1]);
                            break;
                        case "Altura_Cabezal":
                            AlturaPuntero = int.Parse(partes[1]);
                            break;
                        case "Milisegundos_Por_Iteracion_Subir_Cabezal":
                            MilisPorIterSubidaPuntero = int.Parse(partes[1]);
                            break;
                        case "Tiempo_De_Paso_En_Estatico":
                            TiempoPasoEnEstatico = int.Parse(partes[1]);
                            break;
                        case "Color_Paso_En_Tabla":
                            PasoDeTabla = Color.FromArgb(int.Parse(partes[2]), int.
                                Parse(partes[4]), int.Parse(partes[6]), int.Parse(
                                partes[8]));
                            break;
                        case "Color_Puntero":
                            ColorPuntero = Color.FromArgb(int.Parse(partes[2]), int.
                                Parse(partes[4]), int.Parse(partes[6]), int.Parse(
                                partes[8]));
                            break;
                        case "Color_Dibujo":
                            ColorLinea = Color.FromArgb(int.Parse(partes[2]), int.
                                Parse(partes[4]), int.Parse(partes[6]), int.Parse(
                                partes[8]));
                            break;
                        case "Color_Mesa":
                            ColorMesa = Color.FromArgb(int.Parse(partes[2]), int.
                                Parse(partes[4]), int.Parse(partes[6]), int.Parse(
                                partes[8]));
                            break;
                        case "Limpiar_Mesa_Y_Cabezal":
                            LimpiarMesaYBrazo = bool.Parse(partes[1]);
                            break;

                        default:
                            break;
                    }
                }
            }
            sr.Close();
        }
        catch { }
```

```
            return 0;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Text;
using SimuladorCNC.Mecanismos;

namespace SimuladorCNC.ControlDePrograma
{
    class Contador
    {
        Brazo miBrazo;
        Principal front;

        int distanciaRecorrida;
        int tiempoGastado;

        int distancia;
        int tiempo;

        int error;

        public Contador(Brazo B, Principal f)
        {
            miBrazo = B;
            front = f;
            distanciaRecorrida=0;
            tiempoGastado=0;
            distancia = 0;
            tiempo = 0;
            error = 0;
        }

        public void contadorG00(int X, int Y)
        {
            error = 0;
            if (!miBrazo.getBrazoSubido()) tiempo = contarSTiempo(miBrazo.getMovimiento()
                .getTiempoSBrazo(), front.getConfig().framesAlturaPuntero());
            else tiempo = 0;
            miBrazo.getPlanoOculto().Clear(Color.White);
            miBrazo.getPlanoOculto().DrawLine(new Pen(Color.Black), miBrazo.getX(),
                miBrazo.getY(), X, Y);
            actualizarDistanciaTiempoParaG00(front.getConfig().getPunteroTraslacionMax(),
                miBrazo.getX(), miBrazo.getY(), X, Y);
        }

        public void contadorG01(int F, int X, int Y)
        {
            error = 0;
            miBrazo.getPlanoOculto().Clear(Color.White);
            miBrazo.getPlanoOculto().DrawLine(new Pen(Color.Black), miBrazo.getX(),
                miBrazo.getY(), X, Y);
            actualizarDistanciaTiempo(F,miBrazo.getX(), miBrazo.getY(), X, Y);
        }

        public void contadorG02(int F, int X, int Y, int I, int J)
        {
            error = 0;
            miBrazo.getPlanoOculto().Clear(Color.White);
            miBrazo.getEstatico().crearCirculo90(F, X, Y, I, J, miBrazo.getX(), miBrazo.
                getY(), miBrazo.getPlanoOculto(), Color.Black);
            actualizarDistanciaTiempo(F, miBrazo.getX(), miBrazo.getY(), X, Y);
        }

        public void contadorG03(int F, int X, int Y, int I, int J)
        {
            error = 0;
            miBrazo.getPlanoOculto().Clear(Color.White);
            miBrazo.getEstatico().crearCirculo90(F, miBrazo.getX(), miBrazo.getY(), I, J,
                X, Y, miBrazo.getPlanoOculto(), Color.Black);
            actualizarDistanciaTiempo(F, miBrazo.getX(), miBrazo.getY(), X, Y);
```

```
    }

    public void contadorS1()
    {
        error = 0;
        distancia = 0;
        if (miBrazo.getBrazoSubido()) tiempo = contarSTiempo(miBrazo.getMovimiento().
            getTiempoSBrazo(), front.getConfig().framesAlturaPuntero());
        else tiempo = 0;
    }

    public void contadorS_1()
    {
        error = 0;
        distancia = 0;
        if (!miBrazo.getBrazoSubido()) tiempo = contarSTiempo(miBrazo.getMovimiento()
            .getTiempoSBrazo(), front.getConfig().framesAlturaPuntero());
        else tiempo = 0;
    }

    public void contadorVacio()
    {
        error = 0;
        distancia = 0;
        tiempo = 0;
    }


    private void actualizarDistanciaTiempo(int freq, int iniX, int iniY, int finX,
        int finY)
    {
        distancia = contarDistancia(iniX, iniY, finX, finY);
        tiempo = contarTiempo(freq, distancia);
    }

    private void actualizarDistanciaTiempoParaG00(int freq, int iniX, int iniY, int
        finX, int finY)
    {
        distancia = contarDistancia(iniX, iniY, finX, finY);
        tiempo += contarTiempo(freq, distancia);
    }

    public int contarDistancia(int iniX, int iniY, int finX, int finY)
    {
        int contador = 0;

        while (iniX != finX || iniY != finY)
        {
            error = miBrazo.getMovimiento().nuevaCoordenada(ref iniX, ref iniY);
            if (error == 1 || error == 2) return contador;
            contador++;
        }
        return contador;
    }

    public int contarTiempo(int freq, int pix)
    {
        int tiempo = 0;
        int tiempoPor1 = miBrazo.getMovimiento().tiempoTimer(freq);
        tiempo = tiempoPor1 * pix;
        return tiempo;
    }

    public int contarSTiempo(int tiempoPor1, int pix)
    {
        return tiempoPor1 * pix;
    }

    public void reiniciarContador()
    {
```

```
            distanciaRecorrida = 0;
            tiempoGastado = 0;
            miBrazo.getDetallesCabezal().setDistancia(distanciaRecorrida);
            miBrazo.getDetallesCabezal().setTiempo(tiempoGastado);
        }

        public void actualizarContador()
        {
            distanciaRecorrida += distancia;
            tiempoGastado += tiempo;
            miBrazo.getDetallesCabezal().setDistancia(distanciaRecorrida);
            miBrazo.getDetallesCabezal().setTiempo(tiempoGastado);
        }

        public void retrocederContador()
        {
            miBrazo.getDetallesCabezal().setDistancia(distanciaRecorrida);
            miBrazo.getDetallesCabezal().setTiempo(tiempoGastado);
        }

        public int getDistanciaRecorrida()
        {
            return distanciaRecorrida;
        }

        public int getDistancia()
        {
            return distancia;
        }

        public int getTiempoGastado()
        {
            return tiempoGastado;
        }

        public int getTiempo()
        {
            return tiempo;
        }

        public int getError()
        {
            return error;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using SimuladorCNC.Errores;
using SimuladorCNC.Menu;

namespace SimuladorCNC.ControlDePrograma
{
    class ejecucion
    {
        ControlDePrograma Control;
        Principal front;
        tabla miTabla;

        Boolean stop,play, NecesarioReiniciar;

        public ejecucion(ControlDePrograma C, Principal f,  tabla t)
        {
            Control = C;
            front = f;
            miTabla = t;
            stop = false;
            play = false;
            NecesarioReiniciar = false;
        }

        public bool extraerXY(String parametros, ref int x, ref int y, bool absoluto)
        {
            bool miX = false;
            bool miY = false;
            int asignados = 0;

            if (parametros != null)
            {
                String[] param = parametros.Trim().Split(' ');

                for (int i = 0; i < param.Length; i++)
                {
                    if (param[i] != "")
                    {
                        switch (param[i][0])
                        {
                            case 'X':
                                if (!int.TryParse(param[i].Substring(1), out x)) return
                                    false;
                                else if (!absoluto) x = front.getBrazo().absolutizarX(x);
                                miX = true;
                                asignados++;
                                break;
                            case 'Y':
                                if (!int.TryParse(param[i].Substring(1), out y)) return
                                    false;
                                else
                                {
                                    if (!absoluto) y = front.getBrazo().absolutizarY(y);
                                    y = front.getMesa().transformarY(y);
                                }
                                miY = true;
                                asignados++;
                                break;
                            default:
                                return false;
                        }
                    }
                }
            }
            if (miX && miY && asignados == 2) return true;
            else return false;
        }
```

```
    public bool extraerFXY(String parametros, ref int f, ref int x, ref int y, bool
        absoluto)
    {
        bool miF = false;
        bool miX = false;
        bool miY = false;
        int asignados = 0;

        if (parametros != null)
        {
            String[] param = parametros.Trim().Split(' ');

            for (int i = 0; i < param.Length; i++)
            {
                if (param[i] != "")
                {
                    switch (param[i][0])
                    {
                        case 'F':
                            if (!int.TryParse(param[i].Substring(1), out f)) return
                                false;
                            miF = true;
                            asignados++;
                            break;
                        case 'X':
                            if (!int.TryParse(param[i].Substring(1), out x)) return
                                false;
                            else if (!absoluto) x = front.getBrazo().absolutizarX(x);
                            miX = true;
                            asignados++;
                            break;
                        case 'Y':
                            if (!int.TryParse(param[i].Substring(1), out y)) return
                                false;
                            else
                            {
                                if (!absoluto) y = front.getBrazo().absolutizarY(y);
                                y = front.getMesa().transformarY(y);
                            }
                            miY = true;
                            asignados++;
                            break;
                        default:
                            return false;
                    }
                }
            }
        }
        if (miF && miX && miY && asignados == 3) return true;
        else return false;
    }

    public bool extraerFXYIJ(String parametros, ref int F, ref int X, ref int Y, ref
        int I, ref int J, bool absoluto)
    {
        bool miF = false;
        bool miX = false;
        bool miY = false;
        bool miI = false;
        bool miJ = false;
        int asignados = 0;

        if (parametros != null)
        {
            String[] param = parametros.Trim().Split(' ');

            for (int i = 0; i < param.Length; i++)
            {
                if (param[i] != "")
```

```
                {

                    switch (param[i][0])
                    {
                        case 'F':
                            if (!int.TryParse(param[i].Substring(1), out F)) return
                                false;
                            miF = true;
                            asignados++;
                            break;
                        case 'X':
                            if (!int.TryParse(param[i].Substring(1), out X)) return
                                false;
                            else if (!absoluto) X = front.getBrazo().absolutizarX(X);
                            miX = true;
                            asignados++;
                            break;
                        case 'Y':
                            if (!int.TryParse(param[i].Substring(1), out Y)) return
                                false;
                            else
                            {
                                if (!absoluto) Y = front.getBrazo().absolutizarY(Y);
                                Y = front.getMesa().transformarY(Y);
                            }
                            miY = true;
                            asignados++;
                            break;
                        case 'I':
                            if (!int.TryParse(param[i].Substring(1), out I)) return
                                false;
                            else if (!absoluto) I = front.getBrazo().absolutizarX(I);
                            miI = true;
                            asignados++;
                            break;
                        case 'J':
                            if (!int.TryParse(param[i].Substring(1), out J)) return
                                false;
                            else
                            {
                                if (!absoluto) J = front.getBrazo().absolutizarY(J);
                                J = front.getMesa().transformarY(J);
                            }
                            miJ = true;
                            asignados++;
                            break;
                        default:
                            return false;
                    }
                }
            }
        }
        if (miF && miX && miY && miI && miJ && asignados==5) return true;
        else return false;
    }

    public bool extraerFS(String parametros, ref char tipo, ref int valor)
    {
        if (parametros != null)
        {
            String[] param = parametros.Trim().Split(' ');

            for (int i = 0; i < param.Length; i++)
            {
                if (param[i] != "")
                {
                    if (param[i][0] == 'P')
                    {
                        if (int.TryParse(param[i].Substring(1), out valor))
                        {
```

```
                            tipo = param[i][0];
                            return true;
                        }


                    }
                    else return false;
                }
            }
        }
        return false;
    }

    // código de ejecución
    public int Stop()
    {
        front.getMarcha().getStop().StopActivo();
        stop = true;
        return 0;
    }

    public int Play()
    {
        bool posibleSalidaDeLaMesa=false;

        if (Control.programaPreparado() && Control.getComprobar())
            posibleSalidaDeLaMesa=true;

        if (Control.programaPreparado() && Control.Correcto())
        {
            if (posibleSalidaDeLaMesa)
            {
                if (MismoPaso() == -100) return -100;
            }

            miTabla.llamarAtencionFila(miTabla.getPaso()); //llamamos la atencion
                para que se muestre inicialmente el codigo por donde se va ejecutando

            front.getBrazo().setRefrescoPuntero(true);
            front.getMarcha().inicioDeLaMarcha();
            front.getBrazo().getMovimiento().setLlamada("Play");

            if (front.Simular.Checked)
            {
                String cod="";
                PasoMas(ref cod);
                return 0;
            }
            else
            {
                front.getBrazo().getMovimiento().setUsarEstatico(true);
                front.TiempoPlay.Interval = front.getConfig().getTiempoPasoEnEstatico
                    ();
                front.TiempoPlay.Enabled = true;

                stop = false;
                play = true;
                return 0;
            }
        }
        else return -6;
    }

    public void TiempoPlay()
    {
        if (NecesarioReiniciar && play)
        {
            NecesarioReiniciar = false;
            play = false;
            stop = false;
```

```
            return;
        }
        if (play == true)
        {
            if (stop == false)
            {
                String cod = "";
                PasoMas(ref cod);
                if(!cod.Equals("M2")) return;
            }
            acabarElPlayEstatico();
        }
    }

    public void acabarElPlayEstatico()
    {
        play = false;
        stop = false;
        front.getMarcha().finDeLaMarcha();
        front.TiempoPlay.Enabled = false;
        front.getBrazo().getMovimiento().setLlamada("");
    }

    public int Inicio()
    {
        if (Control.programaPreparado() && Control.Correcto())
        {
            Control.getContador().reiniciarContador();
            miTabla.setPaso(0, true);
            front.getMesa().limpiarMesa();
            front.getBrazo().reiniciarBrazo();
            return 0;
        }
        else return -6;
    }

    public int Final()
    {
        return pasoHasta(miTabla.numeroDeFilas());
    }

    public int PasoMas(ref String codigo)
    {
        bool posibleSalidaDeLaMesa = false;

        if (Control.programaPreparado() && Control.getComprobar())
            posibleSalidaDeLaMesa = true;

        if (Control.programaPreparado() && Control.Correcto())
        {
            if (posibleSalidaDeLaMesa)
            {
                if (MismoPaso() == -100) return -100;
            }

            miTabla.llamarAtencionFila(miTabla.getPaso());

            if (front.Simular.Checked)
            {
                front.getBrazo().getMovimiento().setUsarEstatico(false);
                evaluar_sentencia(miTabla.fila(miTabla.getPaso()));
                if (front.getBrazo().getMovimiento().getLLamada().Equals("")) front.
                    getMarcha().inicioDeLaMarcha();
                return 0;
            }
            else
            {
                front.getBrazo().getMovimiento().setUsarEstatico(true);
                front.getBrazo().setRefrescoPuntero(true);
                codigo = marchaHasta(miTabla.getPaso() + 1);
```

```
                return 0;
            }
        }
        else return -6;
    }

    public int PasoMenos()
    {
        int aux = miTabla.getPaso();
        if (aux > 0) aux --;
        return pasoHasta(aux);
    }

    public int MismoPaso()
    {
        return pasoHasta(miTabla.getPaso());
    }

    public int pasoHasta(int linea)
    {
        int error = 0;
        if (Control.programaPreparado() && Control.Correcto())
        {
            Control.getContador().reiniciarContador();
            front.getBrazo().getMovimiento().setUsarEstatico(true);
            front.getMesa().limpiarMesa();

            front.getBrazo().setRefrescoPuntero(false);
            front.getBrazo().reiniciarBrazoSinRefresco();

            int aux = miTabla.getPaso();

            miTabla.setPasoSinActualizarTabla(0);
            marchaHastaSinActualizarMesaNiTabla(linea);

            if (miTabla.getPaso() != linea)
            {
                error = -100;
                linea = miTabla.getPaso();
            }

            miTabla.cambiarSeleccion(aux, linea);

            front.getMesa().actualizarMesa();
            front.punteroImagen.Refresh();
            front.getBrazo().setRefrescoPuntero(true);
            return error;
        }
        else return -6;
    }

    public String marchaHasta(int final)
    {
        String codigo = "";
        while (miTabla.getPaso() < final && !codigo.Equals("M2") && !codigo.Equals("
            ERROR"))
        {
            codigo = evaluar_sentencia(miTabla.fila(miTabla.getPaso()));
            if (!codigo.Equals("M2") && !codigo.Equals("ERROR"))
            {
                miTabla.setPaso(miTabla.getPaso() + 1, true);
                Control.getContador().actualizarContador();
            }
            front.getMesa().actualizarMesa();

        }
        return codigo;
    }

    public String marchaHastaSinActualizarMesaNiTabla(int final)
```

```
        {
            String codigo = "";
            while (miTabla.getPaso() < final && !codigo.Equals("M2") && !codigo.Equals("
                ERROR"))
            {
                codigo = evaluar_sentencia(miTabla.fila(miTabla.getPaso()));
                if (!codigo.Equals("M2") && !codigo.Equals("ERROR"))
                {
                    Control.getContador().actualizarContador();
                    miTabla.setPasoSinActualizarTabla(miTabla.getPaso() + 1);
                }
                else front.getBrazo().setRefrescoPuntero(true);
            }
            return codigo;
        }

        private String evaluar_sentencia(DataGridViewRow linea)
        {
            String orden = (String)linea.Cells[1].Value;
            String param = (String)linea.Cells[2].Value;
            int F = 0, X = 0, Y = 0, I = 0, J = 0;

            if (orden == null) return front.getBrazo().FuncionVacia();

            if (orden.Trim().Equals("")) return front.getBrazo().FuncionVacia();

            if (orden.Equals("G0") || orden.Equals("G00"))
            {
                extraerXY(param, ref X, ref Y, front.getBrazo().getAbsoluto());
                return front.getBrazo().FuncionG00(X,Y);
            }

            if (orden.Equals("G1") || orden.Equals("G01"))
            {

                extraerFXY(param, ref F, ref X, ref Y, front.getBrazo().getAbsoluto());
                return front.getBrazo().FuncionG01(F, X, Y);
            }

            if (orden.Equals("G2") || orden.Equals("G02"))
            {
                extraerFXYIJ(param, ref F, ref X, ref Y, ref I, ref J, front.getBrazo().
                    getAbsoluto());
                return front.getBrazo().FuncionG02(F, X, Y, I, J);
            }

            if (orden.Equals("G3") || orden.Equals("G03"))
            {
                extraerFXYIJ(param, ref F, ref X, ref Y, ref I, ref J, front.getBrazo().
                    getAbsoluto());
                return front.getBrazo().FuncionG03(F, X, Y, I, J);
            }

            if (orden.Equals("G4") || orden.Equals("G04"))
            {
                char aux = '0';
                extraerFS(param, ref aux, ref F);
                return front.getBrazo().FuncionG04(aux, F);
            }

            if (orden.Equals("G90"))
            {
                return front.getBrazo().FuncionG90();
            }

            if (orden.Equals("G91"))
            {
                return front.getBrazo().FuncionG91();
            }
```

```
            if (!orden.Equals(""))
            {
                if (orden[0] == 'F')
                {
                    return front.getBrazo().FuncionFxxx(int.Parse(orden.Substring(1)));
                }
            }
            else return front.getBrazo().FuncionVacia();

            if (orden.Equals("S1"))
            {
                return front.getBrazo().FuncionS1();
            }

            if (orden.Equals("S-1"))
            {
                return front.getBrazo().FuncionS_1();
            }

            if (orden.Equals("M2"))
            {
                return front.getBrazo().FuncionM2();
            }

            return "";
        }

        public Boolean getPlay()
        {
            return play;
        }

        public Boolean getStop()
        {
            return stop;
        }

        public void setStop(Boolean ST)
        {
            stop=ST;
        }

        public void setPlay(Boolean SP)
        {
            play = SP;
        }

        public void Reiniciado()
        {
            if(play==true) NecesarioReiniciar = true;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using SimuladorCNC.Errores;

namespace SimuladorCNC.ControlDePrograma
{
    class integridad
    {
        Principal front;
        ControlDePrograma Control;
        tabla miTabla;
        ejecucion ejecuc;
        int error;

        bool absoluto;

        public integridad(Principal f, ControlDePrograma C, ejecucion ejec, tabla T)
        {
            front = f;
            Control = C;
            miTabla = T;
            ejecuc = ejec;
            error = 0;
            absoluto = front.getConfig().getPunteroAbsolutoIni();
        }

        public bool comprobarIntegridadCodigo()
        {
            error = 0;
            if (Control.getComprobar() == false)
            {
                return true;
            }

            String[] linea = new String[3];
            linea[0] = "";
            linea[1] = "";
            linea[2] = "";
            int valor = -1;
            bool bueno = true;
            absoluto = front.getConfig().getPunteroAbsolutoIni();
            //comprobamos primero
            if (miTabla.numeroDeFilas() > 0)
            {
                linea = miTabla.obtenerValoresFila(0);

                //Se comprueban valores
                if (!pruebaPrimerValor(linea[0])) bueno = false;
                else valor = int.Parse(linea[0].Substring(1));

                if (!pruebaSegundoValor(linea[1], linea[2], 0)) bueno = false;

                if (!pruebaTercerValor(linea[1], linea[2])) bueno = false;

                if (!bueno)
                {
                    Control.setComprobar(true);
                    Error Er;
                    switch (error)
                    {
                        case 2:
                        case 3:
                            break;
                        case 4:
                            Er = new Error(8, linea[0], linea[1], linea[2]);
                            Er.ShowDialog();
                            break;
                        case 6:
```

```
                        Er = new Error(9);
                        Er.ShowDialog();
                        break;
                    case 7:
                        Er = new Error(10);
                        Er.ShowDialog();
                        break;
                    case 8:
                        Er = new Error(14, linea[0], linea[1], linea[2]);
                        Er.ShowDialog();
                        break;
                    default:
                        Er = new Error(5, linea[0], linea[1], linea[2]);
                        Er.ShowDialog();
                        break;
                }
                miTabla.remarcarLinea(0);
                return bueno;
            }

        }

        for (int i = 1; i < miTabla.numeroDeFilas(); i++)
        {
            linea = miTabla.obtenerValoresFila(i);

            if (!pruebaPrimerValor(linea[0])) bueno = false;
            else
            {
                int aux = int.Parse(linea[0].Substring(1));
                if (aux <= valor)
                {
                    error = 1;
                    bueno = false;
                }
                else valor = aux;
            }

            if (!pruebaSegundoValor(linea[1], linea[2], i)) bueno = false;

            if (!pruebaTercerValor(linea[1], linea[2])) bueno = false;

            if (!bueno)
            {
                Control.setComprobar(true);
                Error Er;
                switch (error)
                {
                    case 2:
                    case 3:
                        break;
                    case 1:
                        String[] aux = new String[3];
                        aux[0] = "";
                        aux[1] = "";
                        aux[2] = "";
                        aux = miTabla.obtenerValoresFila(i - 1);
                        Er = new Error(5, aux[0], aux[1], aux[2]);
                        Er.ShowDialog();
                        break;
                    case 4:
                        Er = new Error(8, linea[0], linea[1], linea[2]);
                        Er.ShowDialog();
                        break;
                    case 6:
                        Er = new Error(9);
                        Er.ShowDialog();
                        break;
                    case 7:
                        Er = new Error(10);
```

```
                        Er.ShowDialog();
                        break;
                    case 8:
                        Er = new Error(14, linea[0], linea[1], linea[2]);
                        Er.ShowDialog();
                        break;
                    default:
                        Er = new Error(5, linea[0], linea[1], linea[2]);
                        Er.ShowDialog();
                        break;
                }
                miTabla.remarcarLinea(i);
                return bueno;
            }
        }

        if (bueno == true)
        {
            Control.setComprobar(false);
        }
        return bueno;
    }

    private bool pruebaPrimerValor(String cod)
    {
        if (cod == null)
        {
            error = 4;
            return false;
        }

        if (cod.Trim().Equals(""))
        {
            error = 4;
            return false;
        }


        bool bueno = true;

        if (cod[0] != 'N')
        {
            bueno = false;
        }
        int valor = 0;
        if (!int.TryParse(cod.Substring(1), out valor))
        {
            bueno = false;
        }
        return bueno;
    }

    private bool pruebaSegundoValor(String cod, string param, int posicion)
    {
        if (cod == null)
        {
            if (param == null) return true;
            if (param.Trim().Equals("")) return true;
            return false;
        }

        if (cod.Trim().Equals("M2"))
        {
            if (posicion == miTabla.numeroDeFilas() - 1) return true;
            else
            {
                error=6;
                return false;
            }
        }
```

```
            if (posicion == miTabla.numeroDeFilas() - 1)
            {
                if (cod.Trim().Equals("M2")) return true;
                else
                {
                    error = 7;
                    return false;
                }
            }

            if (cod.Trim().Equals("G0") || cod.Trim().Equals("G00") || cod.Trim().Equals(
                "G1") || cod.Trim().Equals("G01") || cod.Trim().Equals("G2") || cod.Trim(
                ).Equals("G02") || cod.Trim().Equals("G3") || cod.Trim().Equals("G03") ||
                cod.Trim().Equals("G4") || cod.Trim().Equals("G04") || cod.Trim().Equals(
                "G90") || cod.Trim().Equals("G91") || cod.Trim().Equals("S1") || cod.Trim
                ().Equals("S-1") || cod.Trim().Equals("M2") || cod.Trim().Equals(""))
            {
                return true;
            }
            else
            {
                if (cod[0] == 'F')
                {
                    int valor = 0;
                    if (int.TryParse(cod.Substring(1), out valor))
                    {
                        valor=int.Parse(cod.Substring(1));
                        if (comprobarTraslacion(valor)) return true;
                    }
                }
                return false;
            }
        }

        private bool pruebaTercerValor(String cod, String param)
        {
            int F = 0, X = 0, Y = 0, I = 0, J = 0;
            char tipo = '0';

            if (cod == null)
            {
                if (param == null) return true;
                if (param.Trim().Equals("")) return true;
                return false;
            }

            if (cod.Trim().Equals("S1") || cod.Trim().Equals("S-1") || cod.Trim().Equals(
                "M2") || cod.Trim().Equals(""))
            {
                if (param == null) return true;
                if (param.Trim().Equals(""))return true;
                else return false;
            }

            if (cod.Trim().Equals("G90"))
            {
                if (param == null)
                {
                    absoluto = true;
                    return true;
                }
                if (param.Trim().Equals(""))
                {
                    absoluto = true;
                    return true;
                }
                else return false;
            }
```

```
            if (cod.Trim().Equals("G91"))
            {
                if (param == null)
                {
                    absoluto = false;
                    return true;
                }
                if (param.Trim().Equals(""))
                {
                    absoluto = false;
                    return true;
                }
                else return false;
            }

            if (param == null) return false;

            if (cod.Trim().Equals("G0") || cod.Trim().Equals("G00"))
            {
                if (ejecuc.extraerXY(param, ref X, ref Y, absoluto))
                {
                    if(comprobarNumeroEnValor('X', X) && comprobarNumeroEnValor('Y', Y))
                        return true;
                }
            }
            if (cod.Trim().Equals("G1") || cod.Trim().Equals("G01"))
            {
                if (ejecuc.extraerFXY(param, ref F, ref X, ref Y, absoluto))
                {
                    if (comprobarTraslacion(F))
                    {
                        if (comprobarNumeroEnValor('X', X) && comprobarNumeroEnValor('Y',
                            Y)) return true;
                    }
                }
            }
            if (cod.Trim().Equals("G2") || cod.Trim().Equals("G02") || cod.Trim().Equals(
                "G3") || cod.Trim().Equals("G03"))
            {
                if (ejecuc.extraerFXYIJ(param, ref F, ref X, ref Y, ref I, ref J,
                    absoluto))
                {
                    if (comprobarTraslacion(F))
                    {
                        if (comprobarNumeroEnValor('X', X) && comprobarNumeroEnValor('Y',
                            Y)) return true;
                    }
                }
            }
            if (cod.Trim().Equals("G4") || cod.Trim().Equals("G04"))
            {
                if (ejecuc.extraerFS(param, ref tipo, ref F)) return true;
            }
            return false;
        }

        public bool comprobarNumeroEnValor(char coord, int num)
        {
            if(coord=='Y') num=front.getMesa().restaurarY(num);
            if (absoluto)
            {
                if (num < 0)
                {
                    error = 8;
                    return false;
                }
            } return true;
        }

        public bool comprobarTraslacion(int tras)
```

```
        {
            if (tras < front.getConfig().getPunteroTraslacionMin())
            {
                error = 2;
                Error E = new Error(20, tras.ToString(), front.getConfig().
                    getPunteroTraslacionMin().ToString());
                E.ShowDialog();
                return false;
            }

            if (tras > front.getConfig().getPunteroTraslacionMax())
            {
                error = 3;
                Error E = new Error(21, tras.ToString(), front.getConfig().
                    getPunteroTraslacionMax().ToString());
                E.ShowDialog();
                return false;
            }
            return true;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Drawing;
using System.Text;

namespace SimuladorCNC.ControlDePrograma
{
    class tabla
    {
        DataGridView miTabla;
        Principal front;
        Menu.MenuDerecho miMenuDerecho;
        int CeldaSeleccionadaFila;
        int CeldaSeleccionadaColumna;

        int paso;

        public tabla(Principal f)
        {
            front = f;
            paso = 0;
            miTabla = f.Tabla;
            miMenuDerecho = front.getMarcha().getMenuDerecho();
            CeldaSeleccionadaFila = 0;
            CeldaSeleccionadaColumna = 0;
        }

        public void programaCargado()
        {
            miTabla.Rows[0].Cells[0].Selected = false;
            seleccionarFila(0);
        }

        public void reiniciarTabla()
        {
            miTabla.Rows.Clear();
            paso = 0;
            CeldaSeleccionadaFila = 0;
            CeldaSeleccionadaColumna = 0;
            miMenuDerecho.reiniciarMenuDerecho();
        }

        public void setPaso(int paso, bool llamarAtencion)
        {
            if (this.paso != paso)
            {
                deseleccionarFila(this.paso);
                this.paso = paso;
            }
            if (paso >= 0 && paso < miTabla.RowCount)
            {
                if (llamarAtencion) seleccionarFila(paso);
                else seleccionarFilaSinLLamarAtencion(paso);
            }
        }

        public void setPasoSinActualizarTabla(int paso)
        {
            this.paso = paso;
        }

        public int getPaso()
        {
            return paso;
        }

        public void insertarEnTabla(String[] partes)
        {
            String[] linea = new String[3];
```

```
            linea[0] = partes[0];
            if (partes.Length == 1) linea[1] = "";
            else linea[1] = partes[1];
            linea[2] = "";

            for (int i = 2; i < partes.Length; i++)
            {
                linea[2] += partes[i];
                if (i < partes.Length) linea[2] += " ";
            }
            miTabla.Rows.Add(linea);
        }

        public void marcaDeLinea(int linea)
        {
            miTabla.Rows[linea].Selected=true;
        }

        public void remarcarLinea(int linea)
        {
            llamarAtencionFila(linea);
            marcaDeLinea(linea);
        }

        public void edicionDeCelda(int linea, int columna)
        {
            front.getControlDePrograma().HacerComprobar();
            miTabla.ReadOnly = false;
            miTabla.Rows[linea].Cells[columna].ReadOnly = false;
        }

        public void desedicionDeCelda()
        {
            miTabla.ReadOnly = true;
            miTabla.Rows[CeldaSeleccionadaFila].Cells[CeldaSeleccionadaColumna].ReadOnly
                = true;
        }

        public void cambiarSeleccion(int antiguo, int nuevo)
        {
            deseleccionarFila(antiguo);
            seleccionarFila(nuevo);
        }

        public void deseleccionarFila(int x)
        {
            if(x < miTabla.RowCount && x >= 0) miTabla.Rows[x].DefaultCellStyle = new
                DataGridViewCellStyle();
        }

        public void seleccionarFila(int x)
        {
            miTabla.Rows[x].DefaultCellStyle.BackColor = front.getConfig().getPasoDeTabla
                ();
            llamarAtencionFila(x);
        }

        public void seleccionarFilaSinLLamarAtencion(int x)
        {
            miTabla.Rows[x].DefaultCellStyle.BackColor = front.getConfig().getPasoDeTabla
                ();
        }

        public void llamarAtencionFila(int x)
        {
            int visibles = miTabla.DisplayedRowCount(false)/2;
            if (x > visibles) miTabla.FirstDisplayedScrollingRowIndex = x - visibles;
            else miTabla.FirstDisplayedScrollingRowIndex = 0;
```

```
            miTabla.Rows[x].Cells[0].Selected = true;
            miTabla.Rows[x].Cells[0].Selected = false;
        }

        public DataGridViewRow fila(int numFila)
        {
            if (miTabla.Rows[numFila].Cells[0].Value != null) miTabla.Rows[numFila].Cells
                [0].Value = ((String)miTabla.Rows[numFila].Cells[0].Value).Trim();
            if (miTabla.Rows[numFila].Cells[1].Value != null) miTabla.Rows[numFila].Cells
                [1].Value = ((String)miTabla.Rows[numFila].Cells[1].Value).Trim();
            if (miTabla.Rows[numFila].Cells[2].Value != null) miTabla.Rows[numFila].Cells
                [2].Value = ((String)miTabla.Rows[numFila].Cells[2].Value).Trim();
            return miTabla.Rows[numFila];
        }

        public int numeroDeFilas()
        {
            return miTabla.RowCount;
        }

        public String[] obtenerValoresFila(int x)
        {
            String[] fila = new String[3];

            fila[0] = (String) miTabla.Rows[x].Cells[0].Value;
            fila[1] = (String) miTabla.Rows[x].Cells[1].Value;
            fila[2] = (String) miTabla.Rows[x].Cells[2].Value;

            return fila;
        }

        public void insertarNuevaFilaEnFila(int x, String a, String b, String c)
        {
            front.getControlDePrograma().HacerComprobar();
            String[] fila = new String[3];

            fila[0] = a;
            fila[1] = b;
            fila[2] = c;

            if (x >= miTabla.RowCount-1)
            {
                if (miTabla.Rows[miTabla.RowCount - 1].Cells[1].Value != null)
                {
                    if (((String)miTabla.Rows[miTabla.RowCount - 1].Cells[1].Value).Trim(
                        ).Equals("M2"))
                    {
                        miTabla.Rows.Insert(x, fila);
                        if (x == paso) setPaso(paso + 1, false);
                        remarcarLinea(x);
                        return;
                    }
                }
            }
            miTabla.Rows.Insert(x+1, fila);
            if (x < paso) setPaso(paso + 1, false);
            remarcarLinea(x + 1);
        }

        public Menu.MenuDerecho getMenuDerecho()
        {
            return miMenuDerecho;
        }

        public void insertarFilaVacia(int fila, bool encima)
        {
            front.getControlDePrograma().HacerComprobar();
            String[] aux = new String[3];
            aux[0] = "";
            aux[1] = "";
```

```
            aux[2] = "";
            if (front.incrementar_filas.Checked) aux[0] = front.getInsertarCodigo().
                redimensionarFilas(fila, true);
            else aux[0] = front.getInsertarCodigo().averiguarFila(fila, true);

            miTabla.Rows.Insert(fila+1, aux);
            marcaDeLinea(fila + 1);
            if (encima)
            {
                if (fila <= paso) setPaso(paso + 1, false);
            }
            else
            {
                if (fila < paso) setPaso(paso + 1, false);
            }
        }

        public void eliminarFila(int fila)
        {
            front.getControlDePrograma().HacerComprobar();
            miTabla.Rows.RemoveAt(fila);
            if (fila <= paso && paso>0) setPaso(paso-1,false);
            if (fila == 0) setPaso(paso, false);
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SimuladorCNC.ControlDePrograma
{
    class Zoom
    {
        int zoom = 100;
        int alto;
        int ancho;
        Principal front;

        public Zoom(Principal f)
        {
            front = f;
            zoom = 100;
            front.Zoom.Text = zoom.ToString();
            setZoom(zoom);
        }

        public void setZoom(int z)
        {
            alto = (z * front.getConfig().getMesaY()) / 100;
            ancho = (z * front.getConfig().getMesaX()) / 100;
            zoom = z;
        }

        public int getZoom()
        {
            return zoom;
        }

        public int getY()
        {
            return alto;
        }

        public int getX()
        {
            return ancho;
        }

        public void actualizarMesaYBrazo()
        {
            setZoom(zoom);
            front.getMesa().actualizarZoomMesa();
            front.getBrazo().actualizarZoomBrazo();
        }
    }
}
```

A5.2.- Mecanismos.

El grupo **Mecanismos** contiene 4 clases relacionadas con este grupo:

- **Mesa.**
- **Brazo.**
- **Estático.**
- **Movimiento.**

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using System.Windows.Forms;

namespace SimuladorCNC.Mecanismos
{
    class Mesa
    {
        Principal front;

        Bitmap dibujo;
        Graphics plano;
        Panel panel;

        public Mesa(Principal f)
        {
            front = f;
            this.panel = front.panel;
            panel.Height = front.getZoom().getY();
            panel.Width = front.getZoom().getX();
            dibujo = new Bitmap(front.getConfig().getMesaX(),front.getConfig().getMesaY()
                , System.Drawing.Imaging.PixelFormat.Format24bppRgb);
            plano = Graphics.FromImage(dibujo);
            panel.BackgroundImage = dibujo;
            limpiarMesa();
        }

        public void reiniciarMesa()
        {
            panel.Height = front.getZoom().getY();
            panel.Width = front.getZoom().getX();
            dibujo = new Bitmap(front.getConfig().getMesaX(), front.getConfig().getMesaY(
                ), System.Drawing.Imaging.PixelFormat.Format24bppRgb);
            plano = Graphics.FromImage(dibujo);
            plano.Clear(front.getConfig().getColorMesa());
            limpiarMesa();
            panel.Refresh();
        }

        public void limpiarMesa()
        {
            plano.Clear(front.getConfig().getColorMesa());
            actualizarMesa();
        }

        public void actualizarMesa()
        {
            panel.BackgroundImage = dibujo;
            panel.Refresh();
        }

        public void actualizarZoomMesa()
        {
            panel.Height = front.getZoom().getY();
            panel.Width = front.getZoom().getX();
            panel.BackgroundImage = dibujo;
            panel.Refresh();
        }

        public void destruirMesa()
        {
            plano.Dispose();
        }

        public int transformarY(int posY)
        {
            return front.getConfig().getMesaY() - (posY + 1);
        }
```

```
        public int restaurarY(int posicion)
        {
            return - posicion + (front.getConfig().getMesaY() -1);
        }

        public Graphics getPlano()
        {
            return plano;
        }

        public int getAlto()
        {
            return front.getConfig().getMesaY();
        }

        public int getAncho()
        {
            return front.getConfig().getMesaX();
        }

        public Bitmap getDibujo()
        {
            return dibujo;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Text;
using SimuladorCNC.Menu;
using System.Windows.Forms;

namespace SimuladorCNC.Mecanismos
{
    class Brazo
    {
        Bitmap dibujoOculto, dibujoPuntero;
        Graphics planoOculto, planoPuntero;

        Boolean refrescoDelPuntero;

        Principal front;

        Mesa miMesa;
        bool brazoSubido;
        int X;
        int Y;
        int traslacion;
        bool absoluto;

        estatico est;
        movimiento mov;

        DetallesCabezal detallesCab;

        float iPen, lPen;

        public Brazo(Principal f)
        {
            //coge los valores de la clase configuracion
            front = f;
            front.punteroImagen.SizeMode = PictureBoxSizeMode.StretchImage; //configura
                el modo Stretch para el panel
            iPen = front.getConfig().getAlturaPuntero();
            lPen = front.getConfig().getAlturaPuntero()*2;

            traslacion = front.getConfig().getPunteroTraslacionIni();
            miMesa = front.getMesa();
            X = front.getConfig().getPunteroXIni();
            Y = miMesa.transformarY(front.getConfig().getPunteroYIni());
            absoluto = front.getConfig().getPunteroAbsolutoIni();
            brazoSubido = true;

            dibujoOculto = new Bitmap(miMesa.getAncho(), miMesa.getAlto(), System.Drawing
                .Imaging.PixelFormat.Format24bppRgb);
            dibujoPuntero = new Bitmap(miMesa.getAncho(), miMesa.getAlto(), System.
                Drawing.Imaging.PixelFormat.Format24bppRgb);
            dibujoPuntero.MakeTransparent(Color.White);
            planoOculto = Graphics.FromImage(dibujoOculto);
            planoPuntero = Graphics.FromImage(dibujoPuntero);
            planoPuntero.Clear(Color.Transparent);

            front.punteroImagen.Width = front.getZoom().getX();
            front.punteroImagen.Height = front.getZoom().getY();
            front.punteroImagen.Image = dibujoPuntero;

            refrescoDelPuntero = true;

            est=new estatico(this, front, miMesa);
            mov=new movimiento(this, front, miMesa);

            detallesCab = new DetallesCabezal(front, this);

            this.traslacion = front.getConfig().getPunteroTraslacionIni();
            puntero();
```

```
        }

        public void reiniciarBrazo()
        {
            dibujoOculto.Dispose();
            dibujoOculto = new Bitmap(miMesa.getAncho(), miMesa.getAlto(), System.Drawing
                .Imaging.PixelFormat.Format24bppRgb);
            planoOculto = Graphics.FromImage(dibujoOculto);
            planoOculto.Clear(Color.White);

            dibujoPuntero = new Bitmap(miMesa.getAncho(), miMesa.getAlto(), System.
                Drawing.Imaging.PixelFormat.Format24bppRgb);
            dibujoPuntero.MakeTransparent(Color.White);
            planoPuntero = Graphics.FromImage(dibujoPuntero);
            planoPuntero.Clear(Color.Transparent);

            front.punteroImagen.Width = front.getZoom().getX();
            front.punteroImagen.Height = front.getZoom().getY();
            front.punteroImagen.Image = dibujoPuntero;
            front.punteroImagen.Refresh();

            //coge los valores de la clase configuracion
             traslacion = front.getConfig().getPunteroTraslacionIni();
            X = front.getConfig().getPunteroXIni();
            Y = miMesa.transformarY(front.getConfig().getPunteroYIni());
            absoluto = front.getConfig().getPunteroAbsolutoIni();
            brazoSubido = true;
            refrescoDelPuntero = true;
            mov.setLlamada("");
            iPen = front.getConfig().getAlturaPuntero();
            lPen = front.getConfig().getAlturaPuntero() * 2;
            puntero();
            detallesCab.reiniciarDetallesCabezal();
        }

        public void reiniciarBrazoSinRefresco()
        {
            dibujoOculto.Dispose();
            dibujoOculto = new Bitmap(miMesa.getAncho(), miMesa.getAlto(), System.Drawing
                .Imaging.PixelFormat.Format24bppRgb);
            planoOculto = Graphics.FromImage(dibujoOculto);
            planoOculto.Clear(Color.White);

            dibujoPuntero = new Bitmap(miMesa.getAncho(), miMesa.getAlto(), System.
                Drawing.Imaging.PixelFormat.Format24bppRgb);
            dibujoPuntero.MakeTransparent(Color.White);
            planoPuntero = Graphics.FromImage(dibujoPuntero);
            planoPuntero.Clear(Color.Transparent);

            front.punteroImagen.Width = front.getZoom().getX();
            front.punteroImagen.Height = front.getZoom().getY();
            front.punteroImagen.Image = dibujoPuntero;

            //coge los valores de la clase configuracion
            traslacion = front.getConfig().getPunteroTraslacionIni();
            X = front.getConfig().getPunteroXIni();
            Y = miMesa.transformarY(front.getConfig().getPunteroYIni());
            absoluto = front.getConfig().getPunteroAbsolutoIni();
            brazoSubido = true;
            iPen = front.getConfig().getAlturaPuntero();
            lPen = front.getConfig().getAlturaPuntero() * 2;
            puntero();
            detallesCab.reiniciarDetallesCabezal();
        }

        public void destruirBrazo()
        {
            planoOculto.Dispose();
            planoPuntero.Dispose();
        }
```

```
        public String FuncionG00(int X, int Y)
        {
            if (!mov.getUsarEstatico())
            {
                return mov.FuncionG00(X, Y);
            }
            else return est.FuncionG00(X, Y);
        }

        public String FuncionG01(int F, int X, int Y)
        {
            if (!mov.getUsarEstatico())
            {

                return mov.FuncionG01(F, X, Y);
            }
            else return est.FuncionG01(F,X,Y);
        }

        public String FuncionG02(int F,int X,int Y,int I,int J)
        {
            if (!mov.getUsarEstatico())
            {
                return mov.FuncionG02(F,X,Y,I,J);
            }
            else return est.FuncionG02(F, X, Y, I, J);
        }

        public String FuncionG03(int F, int X, int Y, int I, int J)
        {
            if (!mov.getUsarEstatico())
            {
                return mov.FuncionG03(F, X, Y, I, J);
            }
            else return est.FuncionG03(F, X, Y, I, J);
        }

        public String FuncionG04(char aux, int F)
        {
            if (!mov.getUsarEstatico())
            {
                return mov.FuncionG04(aux, F);
            }
            else return est.FuncionG04(aux, F);
        }

        public String FuncionG90()
        {
            String resultado;
            if (!mov.getUsarEstatico())
            {
                resultado = mov.FuncionG90();
            }
            else resultado = est.FuncionG90();
            return resultado;
        }

        public String FuncionG91()
        {
            String resultado;
            if (!mov.getUsarEstatico())
            {
                resultado = mov.FuncionG91();
            }
            else resultado = est.FuncionG91();
            return resultado;
        }

        public String FuncionFxxx(int F)
```

```
        {
            traslacion = F;
            return "Fxxx";
        }

        public String FuncionS1()
        {
            String resultado;

            if (!mov.getUsarEstatico()) resultado = mov.FuncionS1();
            else resultado = est.FuncionS1();
            return resultado;
        }

        public String FuncionS_1()
        {
            String resultado;

            if (!mov.getUsarEstatico())resultado = mov.FuncionS_1();
            else resultado = est.FuncionS_1();
            return resultado;
        }

        public String FuncionVacia()
        {
            if (!mov.getUsarEstatico())
            {
                return mov.FuncionVacia();
            }
            else return est.FuncionVacia();
        }


        public String FuncionM2()
        {
            if (!mov.getUsarEstatico())
            {
                return mov.FuncionM2();
            }
            else return est.FuncionM2();
        }


        public Graphics getPlanoPuntero()
        {
            return planoPuntero;
        }


        public void puntero()
        {
            puntero(X, Y, lPen, iPen);
        }

        public void puntero(int X, int Y, float lPen, float iPen)
        {
            if (front.BrazoVisible.Checked)
            {
                Pen lapiz = new Pen(front.getConfig().getColorDelPuntero());
                lapiz.Width = iPen;
                planoPuntero.Clear(Color.Transparent);
                planoPuntero.DrawLine(lapiz, X, Y - 5 - lPen, X, Y - 2 - iPen);
                planoPuntero.DrawLine(lapiz, X, Y + 5 + lPen, X, Y + 2 + iPen);
                planoPuntero.DrawLine(lapiz, X - 5 - lPen, Y, X - 2 - iPen, Y);
                planoPuntero.DrawLine(lapiz, X + 5 + lPen, Y, X + 2 + iPen, Y);
                planoPuntero.Flush();
                if (refrescoDelPuntero == true) front.punteroImagen.Refresh();
            }
            else
            {
```

```
                planoPuntero.Clear(Color.Transparent);
                front.punteroImagen.Refresh();
            }
            detallesCab.Posicion(X, Y);
        }

        public int absolutizarX(int X)
        {
            return  X + this.X;
        }

        public int absolutizarY(int Y)
        {
            return front.getMesa().restaurarY(this.Y) + Y;
        }

        public void setRefrescoPuntero(Boolean re)
        {
            refrescoDelPuntero = re;
        }

        public Boolean getBrazoSubido()
        {
            return brazoSubido;
        }

        public void setBrazoSubido(Boolean BS)
        {
            brazoSubido = BS;
            detallesCab.Estado(brazoSubido);
        }

        public void setY(int Y)
        {
            this.Y = Y;
        }

        public int getY()
        {
            return Y;
        }

        public void setX(int X)
        {
            this.X = X;
        }

        public int getX()
        {
            return X;
        }

        public void setTraslacion(int T)
        {
            traslacion = T;
        }

        public int getTraslacion()
        {
            return traslacion;
        }

        public Boolean getAbsoluto()
        {
            return absoluto;
        }

        public void setAbsoluto(Boolean A)
        {
            absoluto = A;
```

```
            detallesCab.Absoluto(absoluto);
        }

        public void setLPen(float L)
        {
            lPen = L;
        }

        public float getLPen()
        {
            return lPen;
        }

        public void setIPen(float I)
        {
            iPen = I;
        }

        public float getIPen()
        {
            return iPen;
        }

        public Graphics getPlanoOculto()
        {
            return planoOculto;
        }

        public Bitmap getDibujoOculto()
        {
            return dibujoOculto;
        }

        public estatico getEstatico()
        {
            return est;
        }

        public movimiento getMovimiento()
        {
            return mov;
        }

        public DetallesCabezal getDetallesCabezal()
        {
            return detallesCab;
        }

        public void actualizarZoomBrazo()
        {
            front.punteroImagen.Width = front.getZoom().getX();
            front.punteroImagen.Height = front.getZoom().getY();
            front.punteroImagen.Refresh();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Text;
using SimuladorCNC.Errores;

namespace SimuladorCNC.Mecanismos
{
    class estatico
    {
        Brazo miBrazo;
        Principal front;
        Mesa miMesa;

        public estatico(Brazo B, Principal F, Mesa M)
        {
            miBrazo = B;
            front = F;
            miMesa=M;
        }

        public String FuncionG00(int X, int Y)
        {
            front.getControlDePrograma().getContador().contadorG00(X,Y);
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.setTraslacion(front.getConfig().getPunteroTraslacionMax());
                miBrazo.setX(X);
                miBrazo.setY(Y);
                miBrazo.setIPen(front.getConfig().getAlturaPuntero());
                miBrazo.setLPen(front.getConfig().getAlturaPuntero() * 2);
                miBrazo.puntero();
                miBrazo.setBrazoSubido(true);

                return "G00";
            }
        }

        public String FuncionG01(int F, int X, int Y)
        {
            if (miBrazo.getBrazoSubido() == false)
            {
                miMesa.getPlano().DrawLine(new Pen(front.getConfig().getColorLinea()),
                    miBrazo.getX(), miBrazo.getY(), X, Y);
            }

            front.getControlDePrograma().getContador().contadorG01(F, X, Y);
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.setX(X);
                miBrazo.setY(Y);
                miBrazo.setTraslacion(F);
                miBrazo.puntero();

                return "G01";
            }
        }

        public String FuncionG02(int F, int X, int Y, int I, int J)
        {
```

```
            if (miBrazo.getBrazoSubido() == false) crearCirculo90(F, X, Y, I, J, miBrazo.
                getX(), miBrazo.getY(), miMesa.getPlano(), front.getConfig().
                getColorLinea());

            front.getControlDePrograma().getContador().contadorG02(F, X, Y, I, J);
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.setX(X);
                miBrazo.setY(Y);
                miBrazo.setTraslacion(F);
                miBrazo.puntero();

                return "G02";
            }
        }

        public String FuncionG03(int F, int X, int Y, int I, int J)
        {
            if (miBrazo.getBrazoSubido() == false) crearCirculo90(F, miBrazo.getX(),
                miBrazo.getY(), I, J, X, Y, miMesa.getPlano(), front.getConfig().
                getColorLinea());

            front.getControlDePrograma().getContador().contadorG03(F, X, Y, I, J);
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.setX(X);
                miBrazo.setY(Y);
                miBrazo.setTraslacion(F);

                miBrazo.puntero();
                return "G03";
            }
        }

        public String FuncionG04(char aux, int F)
        {
            front.getControlDePrograma().getContador().contadorVacio();
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.puntero();
                return "G04";
            }
        }

        public String FuncionG90()
        {
            front.getControlDePrograma().getContador().contadorVacio();
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.setAbsoluto(true);
```

```
                miBrazo.puntero();
                return "G90";
            }
        }

        public String FuncionG91()
        {
            front.getControlDePrograma().getContador().contadorVacio();
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.setAbsoluto(false);
                miBrazo.puntero();
                return "G91";
            }
        }

        public String FuncionS1()
        {
            front.getControlDePrograma().getContador().contadorS1();
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.setIPen(0);
                miBrazo.setLPen(0);
                miBrazo.puntero();
                miBrazo.setBrazoSubido(false);
                return "S1";
            }
        }

        public String FuncionS_1()
        {
            front.getControlDePrograma().getContador().contadorS_1();
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                miBrazo.setIPen(front.getConfig().getAlturaPuntero());
                miBrazo.setLPen(front.getConfig().getAlturaPuntero() * 2);
                miBrazo.puntero();
                miBrazo.setBrazoSubido(true);
                return "S-1";
            }
        }

        public String FuncionM2()
        {
            front.getControlDePrograma().getContador().contadorVacio();
            if (front.getControlDePrograma().getContador().getError() != 0)
            {
                errorDelPuntero(front.getControlDePrograma().getContador().getError());
                return "ERROR";
            }
            else
            {
                //Finalizar
                miBrazo.getPlanoPuntero().Clear(Color.Transparent);
                return "M2";
```

```
        }
    }

    public String FuncionVacia()
    {
        front.getControlDePrograma().getContador().contadorVacio();
        if (front.getControlDePrograma().getContador().getError() != 0)
        {
            errorDelPuntero(front.getControlDePrograma().getContador().getError());
            return "ERROR";
        }
        else
        {
            return "";
        }
    }

    public int crearCirculo90(int F, int X, int Y, int I, int J, int posX, int posY,
        Graphics G, Color color)
    {
        int radio = Math.Abs((X - I) + (Y - J));

        int rx = 0;
        int ry = 0;
        int lado = 2 * radio;
        int grado1 = 0;
        int grado2 = 0;

        if (posX == I)
        {
            if (posY < J)
            {
                rx = posX - radio;
                ry = posY;
                grado1 = 270;
                if (X == I)
                    grado2 = 180;
                if ((X - radio) == I)
                    grado2 = 90;
                if ((X + radio) == I)
                    grado2 = 270;
            }

            if (posY > J)
            {
                rx = posX - radio;
                ry = posY - lado;
                grado1 = 90;
                if (X == I)
                    grado2 = 180;
                if ((X - radio) == I)
                    grado2 = 270;
                if ((X + radio) == I)
                    grado2 = 90;
            }
        }

        if (posY == J)
        {
            if (posX > I)
            {
                rx = posX - lado;
                ry = posY - radio;
                grado1 = 0;
                if (Y == J)
                    grado2 = 180;
                if ((Y - radio) == J)
                    grado2 = 90;
                if ((Y + radio) == J)
                    grado2 = 270;
```

```
                }

                if (posX < I)
                {
                    rx = posX;
                    ry = posY - radio;
                    grado1 = 180;
                    if (Y == J)
                        grado2 = 180;
                    if ((Y - radio) == J)
                        grado2 = 270;
                    if ((Y + radio) == J)
                        grado2 = 90;
                }
            }

            G.DrawArc(new Pen(color), rx, ry, lado, lado, grado1, grado2);
            return 0;
        }

        public void errorDelPuntero(int error)
        {
            miBrazo.getMovimiento().setLlamada("");
            front.getControlDePrograma().getEjecucion().setPlay(false);
            front.TiempoPlay.Enabled = false;
            front.ControlSimulador.Enabled = false;
            if (error == 1)
            {
                int err = 11;
                Error E = new Error(err);
                E.ShowDialog();
                front.getControlDePrograma().getTabla().remarcarLinea(front.
                    getControlDePrograma().getTabla().getPaso());
            }
            if (error == 2)
            {
                int err = 12;
                Error E = new Error(err);
                E.ShowDialog();
                front.getControlDePrograma().getTabla().remarcarLinea(front.
                    getControlDePrograma().getTabla().getPaso());
            }
            pasoParaAtras();
        }

        public void pasoParaAtras()
        {
            front.getControlDePrograma().getEjecucion().setPlay(false);
            front.TiempoPlay.Enabled = false;
            front.ControlSimulador.Enabled = false;
            front.getMarcha().finDeLaMarcha();
            front.getControlDePrograma().getEjecucion().pasoHasta(front.getTabla().
                getPaso());
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Text;
using SimuladorCNC.Errores;

namespace SimuladorCNC.Mecanismos
{
    class movimiento
    {
        Brazo miBrazo;
        Principal front;
        Mesa miMesa;
        int iniX, iniY, finX, finY;
        int Freq;
        int FreqMin, FreqMax;

        int tiempoSBrazo;
        int tiempo;

        Boolean UsarEstatico;
        String Instruccion;
        Boolean Movimiento;
        String llamada;

        public movimiento(Brazo B, Principal F, Mesa M)
        {
            front = F;

            FreqMin = front.getConfig().getPunteroTraslacionMin();
            FreqMax = front.getConfig().getPunteroTraslacionMax();
            miBrazo = B;
            miMesa = M;
            UsarEstatico = false;
            Movimiento = false;
            Freq = miBrazo.getTraslacion();
            llamada = "";
            Instruccion = "";
            tiempoSBrazo = front.getConfig().getMilisPorIterSubidaPuntero();
            tiempo = 0;
        }


        public String FuncionG00(int X, int Y)
        {
            front.getControlDePrograma().getContador().contadorG00(X,Y);

            miBrazo.getPlanoOculto().Clear(Color.White);
            miBrazo.getPlanoOculto().DrawLine(new Pen(Color.Black), miBrazo.getX(),
                miBrazo.getY(), X, Y);
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = X;
            finY = Y;
            Freq = miBrazo.getTraslacion();
            front.ControlSimulador.Interval = tiempoSBrazo;
            Instruccion = "G00";
            Movimiento = false;
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionG01(int F, int X, int Y)
        {
            front.getControlDePrograma().getContador().contadorG01(F, X, Y);

            miBrazo.getPlanoOculto().Clear(Color.White);
            miBrazo.getPlanoOculto().DrawLine(new Pen(Color.Black), miBrazo.getX(),
                miBrazo.getY(), X, Y);
            iniX = miBrazo.getX();
```

```
            iniY = miBrazo.getY();
            finX = X;
            finY = Y;
            Freq = F;
            front.ControlSimulador.Interval = tiempoTimer();
            Instruccion = "G01";
            Movimiento = true;
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionG02(int F, int X, int Y, int I, int J)
        {
            front.getControlDePrograma().getContador().contadorG02(F, X, Y, I, J);

            miBrazo.getPlanoOculto().Clear(Color.White);
            miBrazo.getEstatico().crearCirculo90(F, X, Y, I, J, miBrazo.getX(), miBrazo.
                getY(), miBrazo.getPlanoOculto(), Color.Black);
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = X;
            finY = Y;
            Freq = F;
            front.ControlSimulador.Interval = tiempoTimer();
            Instruccion = "G02";
            Movimiento = true;
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionG03(int F, int X, int Y, int I, int J)
        {
            front.getControlDePrograma().getContador().contadorG03(F, X, Y, I, J);

            miBrazo.getPlanoOculto().Clear(Color.White);
            miBrazo.getEstatico().crearCirculo90(F, miBrazo.getX(), miBrazo.getY(), I, J,
                X, Y, miBrazo.getPlanoOculto(), Color.Black);
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = X;
            finY = Y;
            Freq = F;
            front.ControlSimulador.Interval = tiempoTimer();
            Instruccion = "G03";
            Movimiento = true;
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionG04(char aux, int F)
        {
            front.getControlDePrograma().getContador().contadorVacio();
            front.ControlSimulador.Interval = 250;
            Instruccion = "G04";
            Movimiento = false;
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = miBrazo.getX();
            finY = miBrazo.getY();
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionG90()
        {
            front.getControlDePrograma().getContador().contadorVacio();
            miBrazo.setAbsoluto(true);
            front.ControlSimulador.Interval = 250;
            Instruccion = "G90";
            Movimiento = false;
```

```
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = miBrazo.getX();
            finY = miBrazo.getY();
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionG91()
        {
            front.getControlDePrograma().getContador().contadorVacio();
            miBrazo.setAbsoluto(false);
            front.ControlSimulador.Interval = 250;
            Instruccion = "G91";
            Movimiento = false;
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = miBrazo.getX();
            finY = miBrazo.getY();
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionS1()
        {
            front.getControlDePrograma().getContador().contadorS1();

            front.ControlSimulador.Interval = tiempoSBrazo;
            Instruccion = "S1";
            Movimiento = false;
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = miBrazo.getX();
            finY = miBrazo.getY();
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionS_1()
        {
            front.getControlDePrograma().getContador().contadorS_1();

            front.ControlSimulador.Interval = tiempoSBrazo;
            Instruccion = "S-1";
            Movimiento = false;
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = miBrazo.getX();
            finY = miBrazo.getY();
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionM2()
        {
            front.getControlDePrograma().getContador().contadorVacio();
            //Finalizar
            front.ControlSimulador.Interval = 10;
            Instruccion = "M2";
            Movimiento = false;
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = miBrazo.getX();
            finY = miBrazo.getY();
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public String FuncionVacia()
        {
```

```
            front.getControlDePrograma().getContador().contadorVacio();
            front.ControlSimulador.Interval = 250;
            Instruccion = "";
            Movimiento = false;
            iniX = miBrazo.getX();
            iniY = miBrazo.getY();
            finX = miBrazo.getX();
            finY = miBrazo.getY();
            front.ControlSimulador.Enabled = true;
            return Instruccion;
        }

        public int ControlSimuladorTimer()
        {
            if (Movimiento)
            {
                if (!miBrazo.getBrazoSubido()) miMesa.getDibujo().SetPixel(iniX, iniY,
                    front.getConfig().getColorLinea());
                miBrazo.puntero(iniX, iniY, miBrazo.getLPen(), miBrazo.getIPen());
                miMesa.actualizarMesa();
                if (iniX != finX || iniY != finY)
                {
                    int error;
                    error = nuevaCoordenada();
                    if (error != 0)
                    {
                        front.ControlSimulador.Enabled = false;
                        miBrazo.setX(finX);
                        miBrazo.setY(finY);
                        miBrazo.setTraslacion(Freq);
                    }
                    if (error != 0) miBrazo.getEstatico().errorDelPuntero(error);
                    else actualizarRapidoContador(true);
                    return error;
                }
            }
            else
            {
                if (Instruccion.Equals("G04") || Instruccion.Equals("G90") || Instruccion
                    .Equals("G91") || Instruccion.Equals(""))
                {
                    //waitActivo();
                    //no implementado
                }
                if (Instruccion.Equals("S1"))
                {
                    if (miBrazo.getBrazoSubido())
                    {
                        if (miBrazo.getIPen() > 0)
                        {
                            miBrazo.setIPen(miBrazo.getIPen() - front.getConfig().
                                getVelocidadRefrescoPuntero());
                            miBrazo.setLPen(miBrazo.getIPen() * 2);
                            miBrazo.puntero();
                            actualizarRapidoContador(false);
                            return 0;
                        }
                    }
                    miBrazo.setBrazoSubido(false);
                }
                if (Instruccion.Equals("S-1") || Instruccion.Equals("G00"))
                {
                    if (!miBrazo.getBrazoSubido())
                    {
                        if (miBrazo.getIPen() < front.getConfig().getAlturaPuntero())
                        {
                            miBrazo.setIPen(miBrazo.getIPen() + front.getConfig().
                                getVelocidadRefrescoPuntero());
                            miBrazo.setLPen(miBrazo.getIPen() * 2);
                            miBrazo.puntero();
```

```
                        actualizarRapidoContador(false);
                        return 0;
                    }
                }
                miBrazo.setBrazoSubido(true);
                if (Instruccion.Equals("G00"))
                {
                    front.ControlSimulador.Interval = tiempoTimer();
                    Movimiento = true;
                    return 0;
                }
            }
        }

        front.ControlSimulador.Enabled = false;

        front.getControlDePrograma().getContador().actualizarContador();

        miBrazo.setX(finX);
        miBrazo.setY(finY);
        miBrazo.setTraslacion(Freq);

        if (front.getControlDePrograma().getEjecucion().getStop() && llamada.Equals("
            Play"))
        {
            front.getControlDePrograma().getEjecucion().setStop(false);
            front.getControlDePrograma().getEjecucion().setPlay(false);
            llamada = "";
            front.getControlDePrograma().getTabla().setPaso(front.
                getControlDePrograma().getTabla().getPaso() + 1, true);
            front.getMarcha().finDeLaMarcha();
            return 0;
        }
        if (!Instruccion.Equals("M2")) front.getControlDePrograma().getTabla().
            setPaso(front.getControlDePrograma().getTabla().getPaso() + 1, true);
        else
        {
            miBrazo.getPlanoPuntero().Clear(Color.Transparent);
            front.punteroImagen.Refresh();
            llamada = "";
            front.getControlDePrograma().getEjecucion().setStop(false);
            front.getControlDePrograma().getEjecucion().setPlay(false);
            front.getMarcha().finDeLaMarcha();
        }
        if (llamada.Equals("")) front.getMarcha().finDeLaMarcha();
        else
        { //si la llamada se hace desde el play
            String cod = "";
            front.getControlDePrograma().getEjecucion().PasoMas(ref cod);
        }
        return 0;
    }

    private void waitActivo()
    {
        tiempo++;
        if (tiempo < 2) return;
        else tiempo = 0;
    }

    private int comprovarPosicion(int laX, int laY)
    {
        if (laX >= miMesa.getAncho() || laX < 0 || laY >= miMesa.getAlto() || laY < 0
            ) return 2;
        return 0;
    }

    public int nuevaCoordenada()
    {
        return nuevaCoordenada(ref this.iniX, ref this.iniY);
```

```
        }

        public int nuevaCoordenada(ref int iniX, ref int iniY)
        {
            int retorno = 0;
            miBrazo.getDibujoOculto().SetPixel(iniX, iniY, Color.White);

            if (comprovarPosicion(iniX, iniY - 1) == 2) retorno = 2;
            else
            {
                if (colorIgual(miBrazo.getDibujoOculto().GetPixel(iniX, iniY - 1), Color.
                    Black))
                {
                    iniY--;
                    return 0;
                }
            }
            if (comprovarPosicion(iniX - 1, iniY) == 2) retorno = 2;
            else
            {
                if (colorIgual(miBrazo.getDibujoOculto().GetPixel(iniX - 1, iniY), Color.
                    Black))
                {
                    iniX--;
                    return 0;
                }
            }
            if (comprovarPosicion(iniX, iniY + 1) == 2) retorno = 2;
            else
            {
                if (colorIgual(miBrazo.getDibujoOculto().GetPixel(iniX, iniY + 1), Color.
                    Black))
                {
                    iniY++;
                    return 0;
                }
            }
            if (comprovarPosicion(iniX + 1, iniY) == 2) retorno = 2;
            else
            {
                if (colorIgual(miBrazo.getDibujoOculto().GetPixel(iniX + 1, iniY), Color.
                    Black))
                {
                    iniX++;
                    return 0;
                }
            }
            if (comprovarPosicion(iniX + 1, iniY - 1) == 2) retorno = 2;
            else
            {
                if (colorIgual(miBrazo.getDibujoOculto().GetPixel(iniX + 1, iniY - 1),
                    Color.Black))
                {
                    iniY--; iniX++;
                    return 0;
                }
            }

            if (comprovarPosicion(iniX - 1, iniY - 1) == 2) retorno = 2;
            else
            {
                if (colorIgual(miBrazo.getDibujoOculto().GetPixel(iniX - 1, iniY - 1),
                    Color.Black))
                {
                    iniY--; iniX--;
                    return 0;
                }
            }
            if (comprovarPosicion(iniX - 1, iniY + 1) == 2) retorno = 2;
            else
```

```
            {
                if (colorIgual(miBrazo.getDibujoOculto().GetPixel(iniX - 1, iniY + 1),
                    Color.Black))
                {
                    iniY++; iniX--;
                    return 0;
                }
            }
            if (comprovarPosicion(iniX + 1, iniY + 1) == 2) retorno = 2;
            else
            {
                if (colorIgual(miBrazo.getDibujoOculto().GetPixel(iniX + 1, iniY + 1),
                    Color.Black))
                {
                    iniY++; iniX++;
                    return 0;
                }
            }
            if (retorno == 0) return 1;
            else return retorno;
        }

        private Boolean colorIgual(Color a, Color b)
        {
            if (a.A != b.A) return false;
            if (a.R != b.R) return false;
            if (a.G != b.G) return false;
            if (a.B != b.B) return false;
            return true;
        }

        private int tiempoTimer()
        {
            return tiempoTimer(this.Freq);
        }

        public int tiempoTimer(int Freq)
        {
            bool aux = false; ;
            return tiempoTimer(Freq, front.getConfig().getPunteroTraslacionMax(), front.
                getConfig().getPixelesPorFrecuencia(), ref aux);
        }

        public int tiempoTimer(int Freq, int FreqMax, int PixPorFreq, ref bool error)
        {
            error = false;
            int pixMax, pixAct, resultado;

            pixMax = (FreqMax * PixPorFreq) / 500; //500 es el porcentage medio en que
                calculo Pixeles y tiempo
            try
            {
                pixAct = (Freq * pixMax) / FreqMax;
            }
            catch (DivideByZeroException) { pixAct = (Freq * pixMax); }
            try
            {
                resultado = 1000 / pixAct;//1000 un segundo en milis
            }
            catch (DivideByZeroException) { resultado = 1000; }
            if (resultado == 0)
            {
                error = true;
                return 1;
            }
            else return resultado;
        }

        public Boolean getUsarEstatico()
        {
```

```
            return UsarEstatico;
        }

        public void setUsarEstatico(Boolean UE)
        {
            UsarEstatico = UE;
        }

        public void setLlamada(String ll)
        {
            llamada = ll;
        }

        public String getLLamada()
        {
            return llamada;
        }

        public void actualizarRapidoContador(Boolean posicion)
        {
            int auxTiempo = miBrazo.getDetallesCabezal().getTiempo() + front.
                ControlSimulador.Interval;
            int tiempoMaxBrazo = front.getControlDePrograma().getContador().
                getTiempoGastado() + front.getControlDePrograma().getContador().getTiempo
                ();

            if (posicion)
            {
                int auxDistancia = miBrazo.getDetallesCabezal().getDistancia() + 1;
                int distanciaMaxBrazo = front.getControlDePrograma().getContador().
                    getDistanciaRecorrida() + front.getControlDePrograma().getContador().
                    getDistancia();
                if (auxDistancia <= distanciaMaxBrazo) miBrazo.getDetallesCabezal().
                    setDistancia(auxDistancia);
                else miBrazo.getDetallesCabezal().setDistancia(distanciaMaxBrazo);
            }

            if (auxTiempo <= tiempoMaxBrazo) miBrazo.getDetallesCabezal().setTiempo(
                auxTiempo);
            else miBrazo.getDetallesCabezal().setTiempo(tiempoMaxBrazo);
        }

        public int getPixPorFreq()
        {
            return front.getConfig().getPixelesPorFrecuencia();
        }

        public int getFreqMin()
        {
            return FreqMin;
        }

        public int getTiempoSBrazo()
        {
            return tiempoSBrazo;
        }
    }
}
```

A5.3.- Interfaz.

El grupo **Interfaz** contiene 2 clases relacionadas con este grupo:

- **Principal.**
- **Principal (interfaz).**

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Collections;
using SimuladorCNC.Mecanismos;
using SimuladorCNC.Menu;
using SimuladorCNC.Errores;
using SimuladorCNC.ControlDePrograma;

namespace SimuladorCNC
{
    partial class Principal : Form
    {
        Mesa miMesa;
        Brazo miBrazo;
        ControlDePrograma.ControlDePrograma miPrograma;
        tabla miTabla;

        InsertarCodigo insertarCod;
        Marcha miMarcha;

        Configuracion miConfig;
        Zoom miZoom;

        public Principal()
        {
            InitializeComponent();

            miConfig = new Configuracion(this);
            miZoom = new Zoom(this);

            insertarCod = new InsertarCodigo(this);

            panel.Focus();
            miMesa = new Mesa(this);
            miBrazo = new Brazo(this);
            miMarcha = new Marcha(this);
            miPrograma = new ControlDePrograma.ControlDePrograma(this);
            miTabla = miPrograma.getTabla();

            Simular.Checked = miConfig.getSimular();
            BrazoVisible.Checked = miConfig.getBrazoVisible();
        }

        public Mesa getMesa()
        {
            return miMesa;
        }

        public Brazo getBrazo()
        {
            return miBrazo;
        }

        public ControlDePrograma.ControlDePrograma getControlDePrograma()
        {
            return miPrograma;
        }

        private void TiempoPlay_Tick(object sender, EventArgs e)
        {
            miPrograma.getEjecucion().TiempoPlay();
        }

        private void ControlSimulador_Tick(object sender, EventArgs e)
```

```
        {
            miBrazo.getMovimiento().ControlSimuladorTimer();
        }

        private void NUEVO_Click(object sender, EventArgs e)
        {
            Nuevo N = new Nuevo(this);
            N.ShowDialog();
        }
        private void GUARDAR_Click(object sender, EventArgs e)
        {
            Guardar g = new Guardar(this, GUARDAR);
            g.guarda(false);
        }

        private void GUARDARCOMO_Click(object sender, EventArgs e)
        {
            Guardar g = new Guardar(this, GUARDAR);
            g.guarda(true);
        }

        private void GUARDARImagen_Click(object sender, EventArgs e)
        {
            GuardarImagen g = new GuardarImagen(this, GUARDARImagen);
            g.guardaImagen();
        }

        private void ABRIR_Click(object sender, EventArgs e)
        {
            Abrir a = new Abrir(this, ABRIR);
            a.abre();
        }

        private void Salir_click(object sender, EventArgs e)
        {
            miMesa.destruirMesa();
            miBrazo.destruirBrazo();
            Application.Exit();
        }

        private void acercaDeProyectoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            acercade acerca = new acercade();
            acerca.Show();
        }

        private void Final_click(object sender, EventArgs e)
        {
            miPrograma.getEjecucion().Final();
        }

        private void Paso_click(object sender, EventArgs e)
        {
            String aux = "";
            miPrograma.getEjecucion().PasoMas(ref aux);
        }

        private void PasoMenos_click(object sender, EventArgs e)
        {
            miPrograma.getEjecucion().PasoMenos();
        }

        private void Inicio_click(object sender, EventArgs e)
        {
            miPrograma.getEjecucion().Inicio();
        }

        public void Stop_click(object sender, EventArgs e)
        {
            miPrograma.getEjecucion().Stop();
```

```
        }

        public void Play_click(object sender, EventArgs e)
        {
            miPrograma.getEjecucion().Play();
        }

        private void Reinicio_click(object sender, EventArgs e)
        {
            Simular.Checked = miConfig.getSimular();
            BrazoVisible.Checked = miConfig.getBrazoVisible();
            miPrograma.reiniciarMaquina();
        }

        private void BrazoVisible_CheckedChanged(object sender, EventArgs e)
        {
            if (BrazoVisible.Checked == false)
            {
                miBrazo.setRefrescoPuntero(false);
                miBrazo.getPlanoPuntero().Clear(Color.Transparent);
            }
            else
            {
                miBrazo.setRefrescoPuntero(true);
                miBrazo.puntero();
            }
            punteroImagen.Refresh();
        }

        private void Simular_CheckedChanged(object sender, EventArgs e)
        {
            if (miPrograma.programaPreparado())
            {
                miPrograma.getContador().retrocederContador();
                String aux = "";
                if (Simular.Checked && miBrazo.getMovimiento().getLLamada().Equals("Play"
                    ))
                {
                    TiempoPlay.Enabled = false;
                    ControlSimulador.Enabled = true;
                    miPrograma.getEjecucion().PasoMas(ref aux);
                }

                if (!Simular.Checked && miBrazo.getMovimiento().getLLamada().Equals("Play
                    "))
                {
                    miPrograma.getEjecucion().setPlay(true);
                    ControlSimulador.Enabled = false;
                    if (miPrograma.getEjecucion().getStop())
                    {
                        miPrograma.getEjecucion().acabarElPlayEstatico();
                        miPrograma.getEjecucion().PasoMas(ref aux);
                    }
                    else TiempoPlay.Enabled = true;
                }

                if (!Simular.Checked && miBrazo.getMovimiento().getLLamada().Equals("") &
                    & ControlSimulador.Enabled)
                {
                    miPrograma.getEjecucion().setPlay(false);
                    TiempoPlay.Enabled = false;
                    ControlSimulador.Enabled = false;
                    miMarcha.finDeLaMarcha();
                    miPrograma.getEjecucion().PasoMas(ref aux);
                }
            }
        }

        public Marcha getMarcha()
        {
```

```
        return miMarcha;
    }

    public Configuracion getConfig()
    {
        return miConfig;
    }

    public InsertarCodigo getInsertarCodigo()
    {
        return insertarCod;
    }

    public tabla getTabla()
    {
        return miTabla;
    }

    private void Configuración_Click(object sender, EventArgs e)
    {
        if (miPrograma.programaPreparado())
        {
            int error = 30;
            Error Er = new Error(error);
            Er.ShowDialog();
        }
        else
        {
            (new ConfBasica(this, miConfig)).ShowDialog();
        }
    }

    private void Tabla_MouseClick(object sender, MouseEventArgs e)
    {
        miMarcha.getMenuDerecho().setPosicionDelRaton(e.Location);
    }

     private void Tabla_CellClick(object sender, DataGridViewCellMouseEventArgs e)
    {
        if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
        {
            miTabla.marcaDeLinea(e.RowIndex);
            if (e.Button == MouseButtons.Right)
            {
                miMarcha.getMenuDerecho().MenuDerechoFila_Click(e);
            }
        }
    }

    private void Tabla_DoubleCellClick(object sender, DataGridViewCellMouseEventArgs
        e)
    {
        if (e.Button == MouseButtons.Left)
        {
            if (e.ColumnIndex >= 0 && e.RowIndex >= 0) miPrograma.getTabla().
                edicionDeCelda(e.RowIndex, e.ColumnIndex);
        }
    }

    private void Tabla_PierdeFocus(object sender, DataGridViewCellEventArgs e)
    {
        if (e.ColumnIndex >= 0 && e.RowIndex >= 0) miPrograma.getTabla().
            desedicionDeCelda();
    }

    private void Aceptar_Click(object sender, EventArgs e)
    {
        PanelCodigo.Visible = false;
    }
```

```
        private void RadioPosicionamiento_CheckedChanged(object sender, EventArgs e)
        {
            if (RadioPosicionamiento.Checked) insertarCod.RadioPosicionamientoChecked();
        }

        private void RadioSubirHerramienta_CheckedChanged(object sender, EventArgs e)
        {
            if (RadioSubirHerramienta.Checked) insertarCod.RadioSubirHerramientaChecked()
                ;
        }

        private void RadioBajarHerramienta_CheckedChanged(object sender, EventArgs e)
        {
            if (RadioBajarHerramienta.Checked) insertarCod.RadioBajarHerramientaChecked()
                ;
        }

        private void RadioPausa_CheckedChanged(object sender, EventArgs e)
        {
            if (RadioPausa.Checked) insertarCod.RadioPausaChecked();
        }

        private void RadioLineal_CheckedChanged(object sender, EventArgs e)
        {
            if (RadioLineal.Checked) insertarCod.RadioLinealChecked();
        }

        private void RadioCircularHoraria_CheckedChanged(object sender, EventArgs e)
        {
            if (RadioCircularHoraria.Checked) insertarCod.RadioCircularHorariaChecked();
        }

        private void RadioCircularAntiHoraria_CheckedChanged(object sender, EventArgs e)
        {
            if (RadioCircularAntiHoraria.Checked) insertarCod.
                RadioCircularAntiHorariaChecked();
        }

        private void AvanzarAEstaLinea_Click(object sender, EventArgs e)
        {
            miMarcha.getMenuDerecho().AvanzarAEstaLinea_Click();
        }

        private void NuevaFilaAnterior_Click(object sender, EventArgs e)
        {
            miMarcha.getMenuDerecho().NuevaFilaAnterior_Click();
        }

        private void NuevaFilaSiguiente_Click(object sender, EventArgs e)
        {
            miMarcha.getMenuDerecho().NuevaFilaSiguiente_Click();
        }

        private void BorrarFila_Click(object sender, EventArgs e)
        {
            miMarcha.getMenuDerecho().BorrarFila_Click();
        }

        private void Insertar_Click(object sender, EventArgs e)
        {
            insertarCod.Insertar_Click();
        }

        private void cabezalToolStripMenuItem_Click(object sender, EventArgs e)
        {
            miBrazo.getDetallesCabezal().Hide();
            miBrazo.getDetallesCabezal().Show();
        }

        private void BotonZoom_Click(object sender, EventArgs e)
```

```
        {
            int aux=0;
            int error = -1;
            if (Zoom.Text != null)
            {
                if (int.TryParse(Zoom.Text, out aux))
                {
                    if (aux > 0)
                    {
                        miZoom.setZoom(aux);
                        miZoom.actualizarMesaYBrazo();
                        return;
                    }
                }
            }
            error = 82;
            Error Er = new Error(error);
            Er.ShowDialog();
            this.Zoom.Text = miZoom.getZoom().ToString();
            return;
        }
        public Zoom getZoom() { return miZoom; }
    }
}
```

```
namespace SimuladorCNC
{
    partial class Principal
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new System.
                ComponentModel.ComponentResourceManager(typeof(Principal));
            this.archivoToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.abrirToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.salirToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.guardarToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.simulacionToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem
                ();
            this.Menu_Marcha_Item = new System.Windows.Forms.ToolStripMenuItem();
            this.ayudaToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.acercaDeProyectoToolStripMenuItem = new System.Windows.Forms.
                ToolStripMenuItem();
            this.ABRIR = new System.Windows.Forms.OpenFileDialog();
            this.GUARDARImagen = new System.Windows.Forms.SaveFileDialog();
            this.panel1 = new System.Windows.Forms.Panel();
            this.panel = new System.Windows.Forms.Panel();
            this.punteroImagen = new System.Windows.Forms.PictureBox();
            this.Tabla = new System.Windows.Forms.DataGridView();
            this.Numero = new System.Windows.Forms.DataGridViewTextBoxColumn();
            this.Codigo = new System.Windows.Forms.DataGridViewTextBoxColumn();
            this.Parametros = new System.Windows.Forms.DataGridViewTextBoxColumn();
            this.MenuDerechoTabla = new System.Windows.Forms.ContextMenuStrip(this.
                components);
            this.AvanzarAEstaLinea = new System.Windows.Forms.ToolStripMenuItem();
            this.toolStripSeparator5 = new System.Windows.Forms.ToolStripSeparator();
            this.NuevaFilaAnterior = new System.Windows.Forms.ToolStripMenuItem();
            this.NuevaFilaSiguiente = new System.Windows.Forms.ToolStripMenuItem();
            this.BorrarFila = new System.Windows.Forms.ToolStripMenuItem();
            this.prog = new System.Windows.Forms.Label();
            this.MenuMarchaItem = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuSuperior = new System.Windows.Forms.MenuStrip();
            this.MenuArchivo = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuNuevo = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuAbrir = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuGuardar = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuGuardarComo = new System.Windows.Forms.ToolStripMenuItem();
            this.toolStripSeparator4 = new System.Windows.Forms.ToolStripSeparator();
            this.MenuExportarImagenDeLaMesa = new System.Windows.Forms.ToolStripMenuItem(
                );
            this.toolStripSeparator1 = new System.Windows.Forms.ToolStripSeparator();
            this.MenuSalir = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuSimulacion = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuMarcha = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuParo = new System.Windows.Forms.ToolStripMenuItem();
            this.toolStripSeparator2 = new System.Windows.Forms.ToolStripSeparator();
            this.MenuSiguiente = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuAnterior = new System.Windows.Forms.ToolStripMenuItem();
            this.toolStripSeparator3 = new System.Windows.Forms.ToolStripSeparator();
            this.MenuInicio = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuFinal = new System.Windows.Forms.ToolStripMenuItem();
            this.reiniciarMáquina = new System.Windows.Forms.ToolStripMenuItem();
            this.MenuConfiguracion = new System.Windows.Forms.ToolStripMenuItem();
```

```
            this.MenuAyuda = new System.Windows.Forms.ToolStripMenuItem();
            this.manualToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.cabezalToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.TiempoPlay = new System.Windows.Forms.Timer(this.components);
            this.Simular = new System.Windows.Forms.CheckBox();
            this.BrazoVisible = new System.Windows.Forms.CheckBox();
            this.ControlSimulador = new System.Windows.Forms.Timer(this.components);
            this.GUARDAR = new System.Windows.Forms.SaveFileDialog();
            this.PanelCodigo = new System.Windows.Forms.Panel();
            this.incrementar_filas = new System.Windows.Forms.CheckBox();
            this.labelSegundos = new System.Windows.Forms.Label();
            this.LabelTiempo = new System.Windows.Forms.Label();
            this.TextoTiempo = new System.Windows.Forms.TextBox();
            this.RadioBajarHerramienta = new System.Windows.Forms.RadioButton();
            this.RadioSubirHerramienta = new System.Windows.Forms.RadioButton();
            this.RadioPausa = new System.Windows.Forms.RadioButton();
            this.RadioCircularAntiHoraria = new System.Windows.Forms.RadioButton();
            this.RadioCircularHoraria = new System.Windows.Forms.RadioButton();
            this.TextoJ = new System.Windows.Forms.TextBox();
            this.TextoI = new System.Windows.Forms.TextBox();
            this.TextoY = new System.Windows.Forms.TextBox();
            this.TextoX = new System.Windows.Forms.TextBox();
            this.TextoF = new System.Windows.Forms.TextBox();
            this.LabelJ = new System.Windows.Forms.Label();
            this.LabelI = new System.Windows.Forms.Label();
            this.LabelY = new System.Windows.Forms.Label();
            this.LabelX = new System.Windows.Forms.Label();
            this.LabelF = new System.Windows.Forms.Label();
            this.RadioLineal = new System.Windows.Forms.RadioButton();
            this.RadioPosicionamiento = new System.Windows.Forms.RadioButton();
            this.Insertar = new System.Windows.Forms.Button();
            this.BotonZoom = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.Zoom = new System.Windows.Forms.TextBox();
            this.Final = new System.Windows.Forms.Button();
            this.play = new System.Windows.Forms.Button();
            this.inicio = new System.Windows.Forms.Button();
            this.stop = new System.Windows.Forms.Button();
            this.pasoMenos = new System.Windows.Forms.Button();
            this.pasoMas = new System.Windows.Forms.Button();
            this.panel1.SuspendLayout();
            this.panel.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.punteroImagen)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.Tabla)).BeginInit();
            this.MenuDerechoTabla.SuspendLayout();
            this.MenuSuperior.SuspendLayout();
            this.PanelCodigo.SuspendLayout();
            this.SuspendLayout();
            //
            // archivoToolStripMenuItem
            //
            this.archivoToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms
                .ToolStripItem[] {
            this.abrirToolStripMenuItem,
            this.salirToolStripMenuItem,
            this.guardarToolStripMenuItem});
            resources.ApplyResources(this.archivoToolStripMenuItem, "
                archivoToolStripMenuItem");
            this.archivoToolStripMenuItem.Name = "archivoToolStripMenuItem";
            //
            // abrirToolStripMenuItem
            //
            resources.ApplyResources(this.abrirToolStripMenuItem, "abrirToolStripMenuItem
                ");
            this.abrirToolStripMenuItem.Name = "abrirToolStripMenuItem";
            //
            // salirToolStripMenuItem
            //
            resources.ApplyResources(this.salirToolStripMenuItem, "salirToolStripMenuItem
                ");
```

```
            this.salirToolStripMenuItem.Name = "salirToolStripMenuItem";
            //
            // guardarToolStripMenuItem
            //
            resources.ApplyResources(this.guardarToolStripMenuItem, "
                guardarToolStripMenuItem");
            this.guardarToolStripMenuItem.Name = "guardarToolStripMenuItem";
            this.guardarToolStripMenuItem.Click += new System.EventHandler(this.
                Salir_click);
            //
            // simulacionToolStripMenuItem
            //
            resources.ApplyResources(this.simulacionToolStripMenuItem, "
                simulacionToolStripMenuItem");
            this.simulacionToolStripMenuItem.Name = "simulacionToolStripMenuItem";
            //
            // Menu_Marcha_Item
            //
            this.Menu_Marcha_Item.Name = "Menu_Marcha_Item";
            resources.ApplyResources(this.Menu_Marcha_Item, "Menu_Marcha_Item");
            //
            // ayudaToolStripMenuItem
            //
            this.ayudaToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.
                ToolStripItem[] {
            this.acercaDeProyectoToolStripMenuItem});
            resources.ApplyResources(this.ayudaToolStripMenuItem, "ayudaToolStripMenuItem
                ");
            this.ayudaToolStripMenuItem.Name = "ayudaToolStripMenuItem";
            //
            // acercaDeProyectoToolStripMenuItem
            //
            resources.ApplyResources(this.acercaDeProyectoToolStripMenuItem, "
                acercaDeProyectoToolStripMenuItem");
            this.acercaDeProyectoToolStripMenuItem.Name = "
                acercaDeProyectoToolStripMenuItem";
            this.acercaDeProyectoToolStripMenuItem.Click += new System.EventHandler(this.
                acercaDeProyectoToolStripMenuItem_Click);
            //
            // ABRIR
            //
            this.ABRIR.DefaultExt = "pg";
            resources.ApplyResources(this.ABRIR, "ABRIR");
            //
            // GUARDARImagen
            //
            resources.ApplyResources(this.GUARDARImagen, "GUARDARImagen");
            //
            // panel1
            //
            resources.ApplyResources(this.panel1, "panel1");
            this.panel1.BackColor = System.Drawing.Color.Gray;
            this.panel1.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.panel1.Controls.Add(this.panel);
            this.panel1.Name = "panel1";
            //
            // panel
            //
            this.panel.BackColor = System.Drawing.SystemColors.ActiveCaptionText;
            resources.ApplyResources(this.panel, "panel");
            this.panel.Controls.Add(this.punteroImagen);
            this.panel.Name = "panel";
            //
            // punteroImagen
            //
            this.punteroImagen.BackColor = System.Drawing.Color.Transparent;
            resources.ApplyResources(this.punteroImagen, "punteroImagen");
            this.punteroImagen.Name = "punteroImagen";
            this.punteroImagen.TabStop = false;
            //
```

```
            // Tabla
            //
            this.Tabla.AllowUserToAddRows = false;
            this.Tabla.AllowUserToDeleteRows = false;
            this.Tabla.AllowUserToResizeColumns = false;
            this.Tabla.AllowUserToResizeRows = false;
            resources.ApplyResources(this.Tabla, "Tabla");
            this.Tabla.AutoSizeColumnsMode = System.Windows.Forms.
                DataGridViewAutoSizeColumnsMode.DisplayedCells;
            this.Tabla.BackgroundColor = System.Drawing.Color.FromArgb(((int)(((byte)(247
                )))), ((int)(((byte)(249))))), ((int)(((byte)(254)))));
            this.Tabla.BorderStyle = System.Windows.Forms.BorderStyle.None;
            this.Tabla.ColumnHeadersHeightSizeMode = System.Windows.Forms.
                DataGridViewColumnHeadersHeightSizeMode.DisableResizing;
            this.Tabla.Columns.AddRange(new System.Windows.Forms.DataGridViewColumn[] {
            this.Numero,
            this.Codigo,
            this.Parametros});
            this.Tabla.Cursor = System.Windows.Forms.Cursors.Hand;
            this.Tabla.EditMode = System.Windows.Forms.DataGridViewEditMode.EditOnEnter;
            this.Tabla.MultiSelect = false;
            this.Tabla.Name = "Tabla";
            this.Tabla.ReadOnly = true;
            this.Tabla.RowHeadersVisible = false;
            this.Tabla.RowHeadersWidthSizeMode = System.Windows.Forms.
                DataGridViewRowHeadersWidthSizeMode.AutoSizeToFirstHeader;
            this.Tabla.SelectionMode = System.Windows.Forms.DataGridViewSelectionMode.
                FullRowSelect;
            this.Tabla.CellMouseClick += new System.Windows.Forms.
                DataGridViewCellMouseEventHandler(this.Tabla_CellClick);
            this.Tabla.CellMouseDoubleClick += new System.Windows.Forms.
                DataGridViewCellMouseEventHandler(this.Tabla_DoubleCellClick);
            this.Tabla.RowLeave += new System.Windows.Forms.DataGridViewCellEventHandler(
                this.Tabla_PierdeFocus);
            this.Tabla.MouseClick += new System.Windows.Forms.MouseEventHandler(this.
                Tabla_MouseClick);
            //
            // Numero
            //
            this.Numero.AutoSizeMode = System.Windows.Forms.
                DataGridViewAutoSizeColumnMode.None;
            resources.ApplyResources(this.Numero, "Numero");
            this.Numero.Name = "Numero";
            this.Numero.ReadOnly = true;
            this.Numero.Resizable = System.Windows.Forms.DataGridViewTriState.False;
            this.Numero.SortMode = System.Windows.Forms.DataGridViewColumnSortMode.
                NotSortable;
            //
            // Codigo
            //
            this.Codigo.AutoSizeMode = System.Windows.Forms.
                DataGridViewAutoSizeColumnMode.None;
            resources.ApplyResources(this.Codigo, "Codigo");
            this.Codigo.Name = "Codigo";
            this.Codigo.ReadOnly = true;
            this.Codigo.Resizable = System.Windows.Forms.DataGridViewTriState.False;
            this.Codigo.SortMode = System.Windows.Forms.DataGridViewColumnSortMode.
                NotSortable;
            //
            // Parametros
            //
            this.Parametros.AutoSizeMode = System.Windows.Forms.
                DataGridViewAutoSizeColumnMode.Fill;
            resources.ApplyResources(this.Parametros, "Parametros");
            this.Parametros.Name = "Parametros";
            this.Parametros.ReadOnly = true;
            this.Parametros.SortMode = System.Windows.Forms.DataGridViewColumnSortMode.
                NotSortable;
            //
            // MenuDerechoTabla
```

```
            //
            this.MenuDerechoTabla.Items.AddRange(new System.Windows.Forms.ToolStripItem[]
                {
            this.AvanzarAEstaLinea,
            this.toolStripSeparator5,
            this.NuevaFilaAnterior,
            this.NuevaFilaSiguiente,
            this.BorrarFila});
            this.MenuDerechoTabla.Name = "MenuIzquierdoTabla";
            resources.ApplyResources(this.MenuDerechoTabla, "MenuDerechoTabla");
            //
            // AvanzarAEstaLinea
            //
            this.AvanzarAEstaLinea.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.AvanzarAEstaLinea.Name = "AvanzarAEstaLinea";
            resources.ApplyResources(this.AvanzarAEstaLinea, "AvanzarAEstaLinea");
            this.AvanzarAEstaLinea.Click += new System.EventHandler(this.
                AvanzarAEstaLinea_Click);
            //
            // toolStripSeparator5
            //
            this.toolStripSeparator5.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.toolStripSeparator5.Name = "toolStripSeparator5";
            resources.ApplyResources(this.toolStripSeparator5, "toolStripSeparator5");
            //
            // NuevaFilaAnterior
            //
            this.NuevaFilaAnterior.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.NuevaFilaAnterior.Name = "NuevaFilaAnterior";
            resources.ApplyResources(this.NuevaFilaAnterior, "NuevaFilaAnterior");
            this.NuevaFilaAnterior.Click += new System.EventHandler(this.
                NuevaFilaAnterior_Click);
            //
            // NuevaFilaSiguiente
            //
            this.NuevaFilaSiguiente.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.NuevaFilaSiguiente.Name = "NuevaFilaSiguiente";
            resources.ApplyResources(this.NuevaFilaSiguiente, "NuevaFilaSiguiente");
            this.NuevaFilaSiguiente.Click += new System.EventHandler(this.
                NuevaFilaSiguiente_Click);
            //
            // BorrarFila
            //
            this.BorrarFila.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247)
                ))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.BorrarFila.Name = "BorrarFila";
            resources.ApplyResources(this.BorrarFila, "BorrarFila");
            this.BorrarFila.Click += new System.EventHandler(this.BorrarFila_Click);
            //
            // prog
            //
            resources.ApplyResources(this.prog, "prog");
            this.prog.Name = "prog";
            //
            // MenuMarchaItem
            //
            resources.ApplyResources(this.MenuMarchaItem, "MenuMarchaItem");
            this.MenuMarchaItem.Name = "MenuMarchaItem";
            //
            // MenuSuperior
            //
            this.MenuSuperior.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(
                204)))), ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.MenuSuperior.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.MenuArchivo,
            this.MenuSimulacion,
```

```
            this.reiniciarMáquina,
            this.MenuConfiguracion,
            this.MenuAyuda,
            this.cabezalToolStripMenuItem});
            resources.ApplyResources(this.MenuSuperior, "MenuSuperior");
            this.MenuSuperior.Name = "MenuSuperior";
            //
            // MenuArchivo
            //
            this.MenuArchivo.DropDownItems.AddRange(new System.Windows.Forms.
                ToolStripItem[] {
            this.MenuNuevo,
            this.MenuAbrir,
            this.MenuGuardar,
            this.MenuGuardarComo,
            this.toolStripSeparator4,
            this.MenuExportarImagenDeLaMesa,
            this.toolStripSeparator1,
            this.MenuSalir});
            this.MenuArchivo.Name = "MenuArchivo";
            resources.ApplyResources(this.MenuArchivo, "MenuArchivo");
            //
            // MenuNuevo
            //
            this.MenuNuevo.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247))
                )), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuNuevo.Name = "MenuNuevo";
            resources.ApplyResources(this.MenuNuevo, "MenuNuevo");
            this.MenuNuevo.Click += new System.EventHandler(this.NUEVO_Click);
            //
            // MenuAbrir
            //
            this.MenuAbrir.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247))
                )), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuAbrir.Name = "MenuAbrir";
            resources.ApplyResources(this.MenuAbrir, "MenuAbrir");
            this.MenuAbrir.Click += new System.EventHandler(this.ABRIR_Click);
            //
            // MenuGuardar
            //
            this.MenuGuardar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247
                )))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuGuardar.Name = "MenuGuardar";
            resources.ApplyResources(this.MenuGuardar, "MenuGuardar");
            this.MenuGuardar.Click += new System.EventHandler(this.GUARDAR_Click);
            //
            // MenuGuardarComo
            //
            this.MenuGuardarComo.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
                (247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuGuardarComo.Name = "MenuGuardarComo";
            resources.ApplyResources(this.MenuGuardarComo, "MenuGuardarComo");
            this.MenuGuardarComo.Click += new System.EventHandler(this.GUARDARCOMO_Click)
                ;
            //
            // toolStripSeparator4
            //
            this.toolStripSeparator4.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.toolStripSeparator4.ForeColor = System.Drawing.Color.Transparent;
            this.toolStripSeparator4.Name = "toolStripSeparator4";
            resources.ApplyResources(this.toolStripSeparator4, "toolStripSeparator4");
            //
            // MenuExportarImagenDeLaMesa
            //
            this.MenuExportarImagenDeLaMesa.BackColor = System.Drawing.Color.FromArgb(((
                int)(((byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuExportarImagenDeLaMesa.Name = "MenuExportarImagenDeLaMesa";
            resources.ApplyResources(this.MenuExportarImagenDeLaMesa, "
                MenuExportarImagenDeLaMesa");
```

```
            this.MenuExportarImagenDeLaMesa.Click += new System.EventHandler(this.
                GUARDARImagen_Click);
            //
            // toolStripSeparator1
            //
            this.toolStripSeparator1.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.toolStripSeparator1.ForeColor = System.Drawing.Color.Transparent;
            this.toolStripSeparator1.Name = "toolStripSeparator1";
            resources.ApplyResources(this.toolStripSeparator1, "toolStripSeparator1");
            //
            // MenuSalir
            //
            this.MenuSalir.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247))
                )), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuSalir.Name = "MenuSalir";
            resources.ApplyResources(this.MenuSalir, "MenuSalir");
            this.MenuSalir.Click += new System.EventHandler(this.Salir_click);
            //
            // MenuSimulacion
            //
            this.MenuSimulacion.DropDownItems.AddRange(new System.Windows.Forms.
                ToolStripItem[] {
            this.MenuMarcha,
            this.MenuParo,
            this.toolStripSeparator2,
            this.MenuSiguiente,
            this.MenuAnterior,
            this.toolStripSeparator3,
            this.MenuInicio,
            this.MenuFinal});
            this.MenuSimulacion.Name = "MenuSimulacion";
            resources.ApplyResources(this.MenuSimulacion, "MenuSimulacion");
            //
            // MenuMarcha
            //
            this.MenuMarcha.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247)
                ))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuMarcha.Name = "MenuMarcha";
            resources.ApplyResources(this.MenuMarcha, "MenuMarcha");
            this.MenuMarcha.Click += new System.EventHandler(this.Play_click);
            //
            // MenuParo
            //
            this.MenuParo.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247)))
                ), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuParo.Name = "MenuParo";
            resources.ApplyResources(this.MenuParo, "MenuParo");
            this.MenuParo.Click += new System.EventHandler(this.Stop_click);
            //
            // toolStripSeparator2
            //
            this.toolStripSeparator2.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.toolStripSeparator2.Name = "toolStripSeparator2";
            resources.ApplyResources(this.toolStripSeparator2, "toolStripSeparator2");
            //
            // MenuSiguiente
            //
            this.MenuSiguiente.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(
                247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuSiguiente.Name = "MenuSiguiente";
            resources.ApplyResources(this.MenuSiguiente, "MenuSiguiente");
            this.MenuSiguiente.Click += new System.EventHandler(this.Paso_click);
            //
            // MenuAnterior
            //
            this.MenuAnterior.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(
                247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuAnterior.Name = "MenuAnterior";
```

```
            resources.ApplyResources(this.MenuAnterior, "MenuAnterior");
            this.MenuAnterior.Click += new System.EventHandler(this.PasoMenos_click);
            //
            // toolStripSeparator3
            //
            this.toolStripSeparator3.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.toolStripSeparator3.ForeColor = System.Drawing.SystemColors.Control;
            this.toolStripSeparator3.Name = "toolStripSeparator3";
            resources.ApplyResources(this.toolStripSeparator3, "toolStripSeparator3");
            //
            // MenuInicio
            //
            this.MenuInicio.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247)
                ))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.MenuInicio.Name = "MenuInicio";
            resources.ApplyResources(this.MenuInicio, "MenuInicio");
            this.MenuInicio.Click += new System.EventHandler(this.Inicio_click);
            //
            // MenuFinal
            //
            this.MenuFinal.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(253))
                )), ((int)(((byte)(253)))), ((int)(((byte)(253)))));
            this.MenuFinal.Name = "MenuFinal";
            resources.ApplyResources(this.MenuFinal, "MenuFinal");
            this.MenuFinal.Click += new System.EventHandler(this.Final_click);
            //
            // reiniciarMáquina
            //
            this.reiniciarMáquina.Alignment = System.Windows.Forms.ToolStripItemAlignment
                .Right;
            this.reiniciarMáquina.BackColor = System.Drawing.Color.FromArgb(((int)(((byte
                )(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            resources.ApplyResources(this.reiniciarMáquina, "reiniciarMáquina");
            this.reiniciarMáquina.ForeColor = System.Drawing.Color.DarkRed;
            this.reiniciarMáquina.Margin = new System.Windows.Forms.Padding(0, 0, 15, 0);
            this.reiniciarMáquina.Name = "reiniciarMáquina";
            this.reiniciarMáquina.Click += new System.EventHandler(this.Reinicio_click);
            //
            // MenuConfiguracion
            //
            this.MenuConfiguracion.Name = "MenuConfiguracion";
            resources.ApplyResources(this.MenuConfiguracion, "MenuConfiguracion");
            this.MenuConfiguracion.Click += new System.EventHandler(this.
                Configuración_Click);
            //
            // MenuAyuda
            //
            this.MenuAyuda.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[
                ] {
            this.manualToolStripMenuItem});
            this.MenuAyuda.Name = "MenuAyuda";
            resources.ApplyResources(this.MenuAyuda, "MenuAyuda");
            //
            // manualToolStripMenuItem
            //
            this.manualToolStripMenuItem.BackColor = System.Drawing.Color.FromArgb(((int)
                (((byte)(247)))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.manualToolStripMenuItem.Name = "manualToolStripMenuItem";
            resources.ApplyResources(this.manualToolStripMenuItem, "
                manualToolStripMenuItem");
            //
            // cabezalToolStripMenuItem
            //
            this.cabezalToolStripMenuItem.Alignment = System.Windows.Forms.
                ToolStripItemAlignment.Right;
            this.cabezalToolStripMenuItem.Margin = new System.Windows.Forms.Padding(0, 0,
                20, 0);
            this.cabezalToolStripMenuItem.Name = "cabezalToolStripMenuItem";
            resources.ApplyResources(this.cabezalToolStripMenuItem, "
```

```
            cabezalToolStripMenuItem");
        this.cabezalToolStripMenuItem.Click += new System.EventHandler(this.
            cabezalToolStripMenuItem_Click);
        //
        // TiempoPlay
        //
        this.TiempoPlay.Interval = 1000;
        this.TiempoPlay.Tick += new System.EventHandler(this.TiempoPlay_Tick);
        //
        // Simular
        //
        resources.ApplyResources(this.Simular, "Simular");
        this.Simular.Checked = true;
        this.Simular.CheckState = System.Windows.Forms.CheckState.Checked;
        this.Simular.Name = "Simular";
        this.Simular.UseVisualStyleBackColor = true;
        this.Simular.CheckedChanged += new System.EventHandler(this.
            Simular_CheckedChanged);
        //
        // BrazoVisible
        //
        resources.ApplyResources(this.BrazoVisible, "BrazoVisible");
        this.BrazoVisible.Checked = true;
        this.BrazoVisible.CheckState = System.Windows.Forms.CheckState.Checked;
        this.BrazoVisible.Name = "BrazoVisible";
        this.BrazoVisible.UseVisualStyleBackColor = true;
        this.BrazoVisible.CheckedChanged += new System.EventHandler(this.
            BrazoVisible_CheckedChanged);
        //
        // ControlSimulador
        //
        this.ControlSimulador.Tick += new System.EventHandler(this.
            ControlSimulador_Tick);
        //
        // GUARDAR
        //
        resources.ApplyResources(this.GUARDAR, "GUARDAR");
        //
        // PanelCodigo
        //
        resources.ApplyResources(this.PanelCodigo, "PanelCodigo");
        this.PanelCodigo.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247
            )))), ((int)(((byte)(249)))), ((int)(((byte)(254)))));
        this.PanelCodigo.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
        this.PanelCodigo.Controls.Add(this.incrementar_filas);
        this.PanelCodigo.Controls.Add(this.labelSegundos);
        this.PanelCodigo.Controls.Add(this.LabelTiempo);
        this.PanelCodigo.Controls.Add(this.TextoTiempo);
        this.PanelCodigo.Controls.Add(this.RadioBajarHerramienta);
        this.PanelCodigo.Controls.Add(this.RadioSubirHerramienta);
        this.PanelCodigo.Controls.Add(this.RadioPausa);
        this.PanelCodigo.Controls.Add(this.RadioCircularAntiHoraria);
        this.PanelCodigo.Controls.Add(this.RadioCircularHoraria);
        this.PanelCodigo.Controls.Add(this.TextoJ);
        this.PanelCodigo.Controls.Add(this.TextoI);
        this.PanelCodigo.Controls.Add(this.TextoY);
        this.PanelCodigo.Controls.Add(this.TextoX);
        this.PanelCodigo.Controls.Add(this.TextoF);
        this.PanelCodigo.Controls.Add(this.LabelJ);
        this.PanelCodigo.Controls.Add(this.LabelI);
        this.PanelCodigo.Controls.Add(this.LabelY);
        this.PanelCodigo.Controls.Add(this.LabelX);
        this.PanelCodigo.Controls.Add(this.LabelF);
        this.PanelCodigo.Controls.Add(this.RadioLineal);
        this.PanelCodigo.Controls.Add(this.RadioPosicionamiento);
        this.PanelCodigo.Controls.Add(this.Insertar);
        this.PanelCodigo.Name = "PanelCodigo";
        //
        // incrementar_filas
        //
```

```
            resources.ApplyResources(this.incrementar_filas, "incrementar_filas");
            this.incrementar_filas.Name = "incrementar_filas";
            this.incrementar_filas.UseVisualStyleBackColor = true;
            //
            // labelSegundos
            //
            resources.ApplyResources(this.labelSegundos, "labelSegundos");
            this.labelSegundos.Name = "labelSegundos";
            //
            // LabelTiempo
            //
            resources.ApplyResources(this.LabelTiempo, "LabelTiempo");
            this.LabelTiempo.Name = "LabelTiempo";
            //
            // TextoTiempo
            //
            resources.ApplyResources(this.TextoTiempo, "TextoTiempo");
            this.TextoTiempo.Name = "TextoTiempo";
            //
            // RadioBajarHerramienta
            //
            resources.ApplyResources(this.RadioBajarHerramienta, "RadioBajarHerramienta")
                ;
            this.RadioBajarHerramienta.Name = "RadioBajarHerramienta";
            this.RadioBajarHerramienta.UseVisualStyleBackColor = true;
            this.RadioBajarHerramienta.CheckedChanged += new System.EventHandler(this.
                RadioBajarHerramienta_CheckedChanged);
            //
            // RadioSubirHerramienta
            //
            resources.ApplyResources(this.RadioSubirHerramienta, "RadioSubirHerramienta")
                ;
            this.RadioSubirHerramienta.Name = "RadioSubirHerramienta";
            this.RadioSubirHerramienta.UseVisualStyleBackColor = true;
            this.RadioSubirHerramienta.CheckedChanged += new System.EventHandler(this.
                RadioSubirHerramienta_CheckedChanged);
            //
            // RadioPausa
            //
            resources.ApplyResources(this.RadioPausa, "RadioPausa");
            this.RadioPausa.Name = "RadioPausa";
            this.RadioPausa.UseVisualStyleBackColor = true;
            this.RadioPausa.CheckedChanged += new System.EventHandler(this.
                RadioPausa_CheckedChanged);
            //
            // RadioCircularAntiHoraria
            //
            resources.ApplyResources(this.RadioCircularAntiHoraria, "
                RadioCircularAntiHoraria");
            this.RadioCircularAntiHoraria.Name = "RadioCircularAntiHoraria";
            this.RadioCircularAntiHoraria.UseVisualStyleBackColor = true;
            this.RadioCircularAntiHoraria.CheckedChanged += new System.EventHandler(this.
                RadioCircularAntiHoraria_CheckedChanged);
            //
            // RadioCircularHoraria
            //
            resources.ApplyResources(this.RadioCircularHoraria, "RadioCircularHoraria");
            this.RadioCircularHoraria.Name = "RadioCircularHoraria";
            this.RadioCircularHoraria.UseVisualStyleBackColor = true;
            this.RadioCircularHoraria.CheckedChanged += new System.EventHandler(this.
                RadioCircularHoraria_CheckedChanged);
            //
            // TextoJ
            //
            resources.ApplyResources(this.TextoJ, "TextoJ");
            this.TextoJ.Name = "TextoJ";
            //
            // TextoI
            //
            resources.ApplyResources(this.TextoI, "TextoI");
```

```
            this.TextoI.Name = "TextoI";
            //
            // TextoY
            //
            resources.ApplyResources(this.TextoY, "TextoY");
            this.TextoY.Name = "TextoY";
            //
            // TextoX
            //
            resources.ApplyResources(this.TextoX, "TextoX");
            this.TextoX.Name = "TextoX";
            //
            // TextoF
            //
            resources.ApplyResources(this.TextoF, "TextoF");
            this.TextoF.Name = "TextoF";
            //
            // LabelJ
            //
            resources.ApplyResources(this.LabelJ, "LabelJ");
            this.LabelJ.Name = "LabelJ";
            //
            // LabelI
            //
            resources.ApplyResources(this.LabelI, "LabelI");
            this.LabelI.Name = "LabelI";
            //
            // LabelY
            //
            resources.ApplyResources(this.LabelY, "LabelY");
            this.LabelY.Name = "LabelY";
            //
            // LabelX
            //
            resources.ApplyResources(this.LabelX, "LabelX");
            this.LabelX.Name = "LabelX";
            //
            // LabelF
            //
            resources.ApplyResources(this.LabelF, "LabelF");
            this.LabelF.Name = "LabelF";
            //
            // RadioLineal
            //
            resources.ApplyResources(this.RadioLineal, "RadioLineal");
            this.RadioLineal.Name = "RadioLineal";
            this.RadioLineal.UseVisualStyleBackColor = true;
            this.RadioLineal.CheckedChanged += new System.EventHandler(this.
                RadioLineal_CheckedChanged);
            //
            // RadioPosicionamiento
            //
            resources.ApplyResources(this.RadioPosicionamiento, "RadioPosicionamiento");
            this.RadioPosicionamiento.Checked = true;
            this.RadioPosicionamiento.Name = "RadioPosicionamiento";
            this.RadioPosicionamiento.TabStop = true;
            this.RadioPosicionamiento.UseVisualStyleBackColor = true;
            this.RadioPosicionamiento.CheckedChanged += new System.EventHandler(this.
                RadioPosicionamiento_CheckedChanged);
            //
            // Insertar
            //
            resources.ApplyResources(this.Insertar, "Insertar");
            this.Insertar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204)))
                ), ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.Insertar.ForeColor = System.Drawing.Color.Black;
            this.Insertar.Name = "Insertar";
            this.Insertar.UseVisualStyleBackColor = false;
            this.Insertar.Click += new System.EventHandler(this.Insertar_Click);
            //
```

```
            // BotonZoom
            //
            resources.ApplyResources(this.BotonZoom, "BotonZoom");
            this.BotonZoom.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204))
                )), ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.BotonZoom.Name = "BotonZoom";
            this.BotonZoom.UseVisualStyleBackColor = false;
            this.BotonZoom.Click += new System.EventHandler(this.BotonZoom_Click);
            //
            // label1
            //
            resources.ApplyResources(this.label1, "label1");
            this.label1.Name = "label1";
            //
            // Zoom
            //
            resources.ApplyResources(this.Zoom, "Zoom");
            this.Zoom.Name = "Zoom";
            //
            // Final
            //
            resources.ApplyResources(this.Final, "Final");
            this.Final.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204)))),
                ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.Final.BackgroundImage = global::SimuladorCNC.Properties.Resources.final;
            this.Final.Name = "Final";
            this.Final.UseVisualStyleBackColor = false;
            this.Final.Click += new System.EventHandler(this.Final_click);
            //
            // play
            //
            resources.ApplyResources(this.play, "play");
            this.play.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204)))), (
                (int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.play.BackgroundImage = global::SimuladorCNC.Properties.Resources.play;
            this.play.Name = "play";
            this.play.UseVisualStyleBackColor = false;
            this.play.Click += new System.EventHandler(this.Play_click);
            //
            // inicio
            //
            resources.ApplyResources(this.inicio, "inicio");
            this.inicio.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204)))),
                ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.inicio.BackgroundImage = global::SimuladorCNC.Properties.Resources.
                inicio;
            this.inicio.Name = "inicio";
            this.inicio.UseVisualStyleBackColor = false;
            this.inicio.Click += new System.EventHandler(this.Inicio_click);
            //
            // stop
            //
            resources.ApplyResources(this.stop, "stop");
            this.stop.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204)))), (
                (int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.stop.BackgroundImage = global::SimuladorCNC.Properties.Resources.stop;
            this.stop.Name = "stop";
            this.stop.UseVisualStyleBackColor = false;
            this.stop.Click += new System.EventHandler(this.Stop_click);
            //
            // pasoMenos
            //
            resources.ApplyResources(this.pasoMenos, "pasoMenos");
            this.pasoMenos.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204))
                )), ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.pasoMenos.BackgroundImage = global::SimuladorCNC.Properties.Resources.
                pasoMenos;
            this.pasoMenos.Name = "pasoMenos";
            this.pasoMenos.UseVisualStyleBackColor = false;
            this.pasoMenos.Click += new System.EventHandler(this.PasoMenos_click);
```

```
            //
            // pasoMas
            //
            resources.ApplyResources(this.pasoMas, "pasoMas");
            this.pasoMas.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204))))
                , ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.pasoMas.BackgroundImage = global::SimuladorCNC.Properties.Resources.
                pasoMas;
            this.pasoMas.Name = "pasoMas";
            this.pasoMas.UseVisualStyleBackColor = false;
            this.pasoMas.Click += new System.EventHandler(this.Paso_click);
            //
            // Principal
            //
            resources.ApplyResources(this, "$this");
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(93)))), ((int)(
                ((byte)(107)))), ((int)(((byte)(153)))));
            this.Controls.Add(this.BotonZoom);
            this.Controls.Add(this.Zoom);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.PanelCodigo);
            this.Controls.Add(this.MenuSuperior);
            this.Controls.Add(this.panel1);
            this.Controls.Add(this.Final);
            this.Controls.Add(this.play);
            this.Controls.Add(this.Tabla);
            this.Controls.Add(this.BrazoVisible);
            this.Controls.Add(this.inicio);
            this.Controls.Add(this.stop);
            this.Controls.Add(this.pasoMenos);
            this.Controls.Add(this.Simular);
            this.Controls.Add(this.prog);
            this.Controls.Add(this.pasoMas);
            this.Name = "Principal";
            this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
            this.panel1.ResumeLayout(false);
            this.panel.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.punteroImagen)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.Tabla)).EndInit();
            this.MenuDerechoTabla.ResumeLayout(false);
            this.MenuSuperior.ResumeLayout(false);
            this.MenuSuperior.PerformLayout();
            this.PanelCodigo.ResumeLayout(false);
            this.PanelCodigo.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        private System.Windows.Forms.ToolStripMenuItem archivoToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem abrirToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem salirToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem guardarToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem simulacionToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem ayudaToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem acercaDeProyectoToolStripMenuItem;
        private System.Windows.Forms.OpenFileDialog ABRIR;
        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.DataGridViewTextBoxColumn Numero;
        private System.Windows.Forms.DataGridViewTextBoxColumn Codigo;
        private System.Windows.Forms.DataGridViewTextBoxColumn Parametros;
        public System.Windows.Forms.Panel panel;
        public System.Windows.Forms.DataGridView Tabla;
        public System.Windows.Forms.ToolStripMenuItem Menu_Marcha_Item;
        public System.Windows.Forms.ToolStripMenuItem MenuMarchaItem;
        public System.Windows.Forms.Label prog;
        private System.Windows.Forms.MenuStrip MenuSuperior;
        private System.Windows.Forms.ToolStripMenuItem MenuArchivo;
        private System.Windows.Forms.ToolStripMenuItem MenuSalir;
```

```
        private System.Windows.Forms.ToolStripMenuItem MenuAyuda;
        public System.Windows.Forms.ToolStripMenuItem MenuMarcha;
        private System.Windows.Forms.ToolStripSeparator toolStripSeparator1;
        public System.Windows.Forms.Button Final;
        public System.Windows.Forms.Button play;
        public System.Windows.Forms.Button stop;
        public System.Windows.Forms.Button pasoMenos;
        public System.Windows.Forms.Button pasoMas;
        public System.Windows.Forms.Button inicio;
        public System.Windows.Forms.ToolStripMenuItem MenuAbrir;
        public System.Windows.Forms.ToolStripMenuItem MenuSiguiente;
        public System.Windows.Forms.ToolStripMenuItem MenuAnterior;
        public System.Windows.Forms.ToolStripMenuItem MenuInicio;
        public System.Windows.Forms.ToolStripMenuItem MenuFinal;
        public System.Windows.Forms.Timer TiempoPlay;
        private System.Windows.Forms.ToolStripMenuItem reiniciarMáquina;
        public System.Windows.Forms.PictureBox punteroImagen;
        public System.Windows.Forms.CheckBox Simular;
        public System.Windows.Forms.CheckBox BrazoVisible;
        public System.Windows.Forms.Timer ControlSimulador;
        public System.Windows.Forms.ToolStripMenuItem MenuParo;
        public System.Windows.Forms.ToolStripMenuItem MenuConfiguracion;
        public System.Windows.Forms.ToolStripMenuItem MenuSimulacion;
        private System.Windows.Forms.ToolStripSeparator toolStripSeparator4;
        public System.Windows.Forms.ToolStripMenuItem MenuGuardarComo;
        private System.Windows.Forms.ToolStripSeparator toolStripSeparator2;
        private System.Windows.Forms.ToolStripSeparator toolStripSeparator3;
        private System.Windows.Forms.SaveFileDialog GUARDARImagen;
        public System.Windows.Forms.ToolStripMenuItem MenuExportarImagenDeLaMesa;
        private System.Windows.Forms.SaveFileDialog GUARDAR;
        public System.Windows.Forms.Panel PanelCodigo;
        public System.Windows.Forms.TextBox TextoF;
        public System.Windows.Forms.Button Insertar;
        public System.Windows.Forms.RadioButton RadioPosicionamiento;
        public System.Windows.Forms.TextBox TextoY;
        public System.Windows.Forms.TextBox TextoX;
        public System.Windows.Forms.Label LabelJ;
        public System.Windows.Forms.Label LabelI;
        public System.Windows.Forms.Label LabelY;
        public System.Windows.Forms.Label LabelX;
        public System.Windows.Forms.Label LabelF;
        public System.Windows.Forms.RadioButton RadioLineal;
        public System.Windows.Forms.RadioButton RadioCircularAntiHoraria;
        public System.Windows.Forms.RadioButton RadioCircularHoraria;
        public System.Windows.Forms.TextBox TextoJ;
        public System.Windows.Forms.TextBox TextoI;
        public System.Windows.Forms.RadioButton RadioPausa;
        public System.Windows.Forms.RadioButton RadioBajarHerramienta;
        public System.Windows.Forms.RadioButton RadioSubirHerramienta;
        public System.Windows.Forms.Label LabelTiempo;
        public System.Windows.Forms.TextBox TextoTiempo;
        private System.Windows.Forms.ToolStripSeparator toolStripSeparator5;
        public System.Windows.Forms.ContextMenuStrip MenuDerechoTabla;
        public System.Windows.Forms.ToolStripMenuItem AvanzarAEstaLinea;
        public System.Windows.Forms.ToolStripMenuItem NuevaFilaAnterior;
        public System.Windows.Forms.ToolStripMenuItem NuevaFilaSiguiente;
        public System.Windows.Forms.ToolStripMenuItem BorrarFila;
        public System.Windows.Forms.Label labelSegundos;
        private System.Windows.Forms.ToolStripMenuItem cabezalToolStripMenuItem;
        public System.Windows.Forms.CheckBox incrementar_filas;
        public System.Windows.Forms.ToolStripMenuItem MenuGuardar;
        public System.Windows.Forms.ToolStripMenuItem MenuNuevo;
        private System.Windows.Forms.Button BotonZoom;
        private System.Windows.Forms.Label label1;
        public System.Windows.Forms.TextBox Zoom;
        private System.Windows.Forms.ToolStripMenuItem manualToolStripMenuItem;
    }
}
```

A5.4.- Menú.

El grupo **Menú** contiene 12 clases relacionadas con este grupo:

- **Nuevo.**
- **Nuevo (interfaz).**
- **Abrir.**
- **Guardar**
- **Guardar imagen.**
- **Configuración básica.**
- **Configuración básica (interfaz).**
- **Detalles cabezal.**
- **Detalles cabezal (interfaz).**
- **Insertar código.**
- **Marcha.**
- **Menú derecho.**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SimuladorCNC.Errores;

namespace SimuladorCNC.Menu
{
    partial class Nuevo : Form
    {
        Principal front;
        ControlDePrograma.ControlDePrograma miPrograma;

        public Nuevo(Principal f)
        {
            InitializeComponent();
            front = f;
            miPrograma = front.getControlDePrograma();
            nombre.Text = "";
            lineas.Text = "10";
        }

        private void Aceptar_Click(object sender, EventArgs e)
        {
            if (correcto() > 0)
            {
                crearNuevoPrograma();
                this.Close();
            }
        }

        private void crearNuevoPrograma()
        {
            miPrograma.reiniciarMaquina(true);
            miPrograma.getTabla().setPaso(0, true);

            crearDatos(nombre.Text, int.Parse(lineas.Text));

            miPrograma.conPrograma();
        }

        private int correcto()
        {
            int aux, error = 1;

            if (nombre.Text == null)
            {
                error = 80;
                Error E = new Error(error);
                E.ShowDialog();
                return -error;
            }
            if (nombre.Text.Equals(""))
            {
                error = 80;
                Error E = new Error(error);
                E.ShowDialog();
                return -error;
            }
            if (!int.TryParse(lineas.Text, out aux))
            {
                error = 81;
                Error E = new Error(error);
                E.ShowDialog();
                return -error;
            }
            else
```

```
            {
                if (aux < 0)
                {
                    error = 81;
                    Error E = new Error(error);
                    E.ShowDialog();
                    return -error;
                }
            }
            return error;
        }

        private void Cancelar_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void crearDatos(String nombre, int filas)
        {
            miPrograma.HacerComprobar();

            front.prog.Text += nombre;
            String[] partes;
            int aux;
            for (int i = 1; i <= filas; i++)
            {
                partes = new String[1];
                aux = i * 10;
                partes[0] = "N" + aux.ToString();
                miPrograma.getTabla().insertarEnTabla(partes);
            }
            partes = new String[2];
            aux = (filas) * 100;
            partes[0] = "N" + aux.ToString();
            partes[1] = "M2";
            miPrograma.getTabla().insertarEnTabla(partes);
            if (miPrograma.getIntegridad().comprobarIntegridadCodigo())
            {
                miPrograma.setArchivoPrograma(filtrarNombre(nombre)+".pg");
                miPrograma.getTabla().programaCargado();
            }
        }

        private String filtrarNombre(String nom)
        {
            char b='_';
            String n =(String)nom.Clone();
            n = n.Replace('?', b);
            n = n.Replace('*', b);
            n = n.Replace('>', b);
            n = n.Replace('<', b);
            n = n.Replace('*', b);
            return n;
        }
    }
}
```

```
namespace SimuladorCNC.Menu
{
    partial class Nuevo
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.nombre = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.Cancelar = new System.Windows.Forms.Button();
            this.Aceptar = new System.Windows.Forms.Button();
            this.lineas = new System.Windows.Forms.TextBox();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)
                (0)));
            this.label1.Location = new System.Drawing.Point(33, 23);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(127, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "Nombre del programa";
            //
            // nombre
            //
            this.nombre.Location = new System.Drawing.Point(36, 45);
            this.nombre.Name = "nombre";
            this.nombre.Size = new System.Drawing.Size(242, 20);
            this.nombre.TabIndex = 1;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)
                (0)));
            this.label2.Location = new System.Drawing.Point(33, 90);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(142, 13);
            this.label2.TabIndex = 2;
            this.label2.Text = "Número inicial de lineas";
            //
            // Cancelar
            //
            this.Cancelar.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Cancelar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204)))
                ), ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.Cancelar.ForeColor = System.Drawing.Color.Black;
            this.Cancelar.Location = new System.Drawing.Point(100, 158);
            this.Cancelar.Name = "Cancelar";
            this.Cancelar.Size = new System.Drawing.Size(75, 23);
            this.Cancelar.TabIndex = 71;
            this.Cancelar.Text = "Cancelar";
            this.Cancelar.UseVisualStyleBackColor = false;
            this.Cancelar.Click += new System.EventHandler(this.Cancelar_Click);
```

```
            //
            // Aceptar
            //
            this.Aceptar.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Aceptar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204))))
                , ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.Aceptar.ForeColor = System.Drawing.Color.Black;
            this.Aceptar.Location = new System.Drawing.Point(203, 158);
            this.Aceptar.Name = "Aceptar";
            this.Aceptar.Size = new System.Drawing.Size(75, 23);
            this.Aceptar.TabIndex = 70;
            this.Aceptar.Text = "Aceptar";
            this.Aceptar.UseVisualStyleBackColor = false;
            this.Aceptar.Click += new System.EventHandler(this.Aceptar_Click);
            //
            // lineas
            //
            this.lineas.Location = new System.Drawing.Point(181, 87);
            this.lineas.Name = "lineas";
            this.lineas.Size = new System.Drawing.Size(97, 20);
            this.lineas.TabIndex = 72;
            this.lineas.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // Nuevo
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247)))), ((int)
                (((byte)(249)))), ((int)(((byte)(254)))));
            this.ClientSize = new System.Drawing.Size(315, 211);
            this.Controls.Add(this.lineas);
            this.Controls.Add(this.Cancelar);
            this.Controls.Add(this.Aceptar);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.nombre);
            this.Controls.Add(this.label1);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "Nuevo";
            this.ShowIcon = false;
            this.ShowInTaskbar = false;
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Nuevo programa";
            this.ResumeLayout(false);
            this.PerformLayout();
        }

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox nombre;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Button Cancelar;
        private System.Windows.Forms.Button Aceptar;
        private System.Windows.Forms.TextBox lineas;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.IO;
using SimuladorCNC.Errores;

namespace SimuladorCNC.Menu
{
    class Abrir
    {
        Principal front;
        OpenFileDialog ABRIR;
        ControlDePrograma.ControlDePrograma miPrograma;

        public Abrir(Principal f, OpenFileDialog A)
        {
            front = f;
            ABRIR = A;
            miPrograma = front.getControlDePrograma();
        }

        public int abre()
        {
            DialogResult resultado = ABRIR.ShowDialog();
            if (resultado == DialogResult.OK)
            {
                cargarNuevoPrograma(ABRIR);
                return 0;
            }

            if (resultado == DialogResult.No)
            {
                int error = 7;
                Error Er = new Error(error);
                Er.ShowDialog();
                return -error;
            }

            return -100;
        }

        public int cargarNuevoPrograma(FileDialog File)
        {
            miPrograma.reiniciarMaquina(true);
            miPrograma.getTabla().setPaso(0, true);
            StreamReader sr = new StreamReader(File.FileName);
            int resultado = obtenerDatos(sr, File.FileName);
            if (resultado != 0)
            {
                miPrograma.reiniciarMaquina();
            }
            else
            {
                miPrograma.conPrograma();
            }
            sr.Close();
            return resultado;
        }

        private int obtenerDatos(StreamReader sr, String filename)
        {
            int error = -1;

            int i1 = Convert.ToInt32("11", 16);
            int i2 = Convert.ToInt32("02", 16);
            string inicial = Char.ConvertFromUtf32(i1) + Char.ConvertFromUtf32(i2);

            int f1 = Convert.ToInt32("1C", 16);
            int f2 = Convert.ToInt32("03", 16);
```

```
            int f3 = Convert.ToInt32("1A", 16);
            string final = Char.ConvertFromUtf32(f1) + Char.ConvertFromUtf32(f2) + Char.
                ConvertFromUtf32(f3);

            string linea;
            int i = 0;
            bool nombre = false;

            miPrograma.HacerComprobar();

            while ((linea = sr.ReadLine()) != null)
            {
                string[] partes = linea.Trim().Split(' ');

                //Si es distinto a inicial o final
                if (!(inicial.Equals(partes[0]) || final.Equals(partes[0])))
                {
                    char valor;
                    if (partes[0].Length == 0) valor = ' ';
                    else valor = partes[0][0];

                    switch (valor)
                    {
                        case '%':
                            if (!nombre)
                            {
                                front.prog.Text += partes[0].Substring(1);
                                nombre = true;
                            }
                            else
                            {
                                error = 1;
                                Error E = new Error(error, i.ToString(), linea);
                                E.ShowDialog();
                                return -error;
                            }
                            break;

                        case 'N':
                            if (partes.Length < 2 || partes.Length > 7)
                            {
                                int aux = 0;
                                if (((partes.Length == 1 && partes[0][0] == 'N') || (
                                    partes.Length == 2 && partes[0][0] == 'N' && partes[1
                                    ].Equals(""))) && int.TryParse(partes[0].Substring(1)
                                    , out aux))
                                {
                                    miPrograma.getTabla().insertarEnTabla(partes);
                                }
                                else
                                {
                                    error = 2;
                                    Error E = new Error(error, i.ToString(), linea);
                                    E.ShowDialog();
                                    return -error;
                                }
                            }
                            else
                            {
                                if (partes[1].Length >= 7)
                                {
                                    if (partes[1].Substring(0, 6).Equals("(FILE="))
                                    {
                                        if (!nombre)
                                        {
                                            String prog = partes[1].Substring(7, (partes[
                                                1].Substring(7).Length - 1));
                                            front.prog.Text += partes[0].Substring(1);
                                            nombre = true;
                                        }
```

```
                                }
                            }
                            else
                            {

                                char v = partes[1][0];
                                if (v == 'G' || v == 'S' || v == 'F')
                                {
                                    miPrograma.getTabla().insertarEnTabla(partes);
                                }
                                else
                                {
                                    if (partes[1].Equals("M2"))
                                    {
                                        miPrograma.getTabla().insertarEnTabla(partes)
                                            ;

                                        if (miPrograma.getIntegridad().
                                            comprobarIntegridadCodigo())
                                        {
                                            //el programa esta bien para ejecutar
                                            if (miPrograma.getTabla().numeroDeFilas()
                                                > 0)
                                            {
                                                if (nombre == false)
                                                {
                                                    front.prog.Text += Path.
                                                        GetFileNameWithoutExtension(
                                                        filename);
                                                }
                                                miPrograma.setArchivoPrograma(Path.
                                                    GetFileName(filename));
                                                miPrograma.getTabla().programaCargado
                                                    ();
                                            }
                                            return 0;
                                        }
                                        else return -5;
                                    }
                                    else
                                    {
                                        error = 3;
                                        Error E = new Error(error, i.ToString(),
                                            linea);
                                        E.ShowDialog();
                                        return -error;
                                    }
                                }
                            }
                        }
                        break;

                    default:
                        if (valor != ' ')
                        {
                            error = 3;
                            Error E = new Error(error, i.ToString(), linea);
                            E.ShowDialog();
                            return -error;
                        }
                        break;
                }
            }
            i++;
        }
        error = 4;
        Error Er = new Error(error);
        Er.ShowDialog();
        return -error;
    }
```

```
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.IO;
using SimuladorCNC.Errores;

namespace SimuladorCNC.Menu
{
    class Guardar
    {
        Principal front;
        ControlDePrograma.ControlDePrograma miPrograma;
        SaveFileDialog GUARDAR;

        public Guardar(Principal f, SaveFileDialog G)
        {
            front = f;
            GUARDAR = G;
            miPrograma = front.getControlDePrograma();
        }

        public int guarda(bool Como)
        {
            int error = -1;
            String fichero = "";
            DialogResult resultado= DialogResult.No;

            if (Como == true)
            {
                resultado = GUARDAR.ShowDialog();
                fichero = GUARDAR.FileName;
            }
            else
            {
                fichero = miPrograma.getArchivoPrograma();
                if (fichero != "") resultado = DialogResult.OK;
            }
            if (resultado == DialogResult.OK)
            {
                if (fichero != "")
                {
                    try
                    {
                        FileStream fs = null;
                        if (Como) fs = (FileStream)GUARDAR.OpenFile();
                        else fs = new FileStream(fichero, FileMode.OpenOrCreate);
                        if (fs != null)
                        {
                            StreamWriter sw = new StreamWriter(fs);
                            guardarDatos(sw, fichero);
                            sw.Flush();
                            fs.Flush();
                            sw.Close();
                            fs.Close();
                            if (Como) miPrograma.setArchivoPrograma(Path.GetFileName(
                                fichero));
                            else miPrograma.setArchivoPrograma(fichero);
                        }
                        else
                        {
                            error = 61;
                            Error Er = new Error(error);
                            Er.ShowDialog();
                            return -error;
                        }
                    }
                    catch
                    {
                        error = 61;
```

```
                        Error Er = new Error(error);
                        Er.ShowDialog();
                        return -error;
                    }
                }
                else
                {
                    error = 60;
                    Error Er = new Error(error);
                    Er.ShowDialog();
                    return -error;
                }
            }
            return 0;
        }

        private int guardarDatos(StreamWriter sw, string filename)
        {
            int i1 = Convert.ToInt32("11", 16);
            int i2 = Convert.ToInt32("02", 16);
            string inicial = Char.ConvertFromUtf32(i1) + Char.ConvertFromUtf32(i2);

            int f1 = Convert.ToInt32("1C", 16);
            int f2 = Convert.ToInt32("03", 16);
            int f3 = Convert.ToInt32("1A", 16);
            string final = Char.ConvertFromUtf32(f1) + Char.ConvertFromUtf32(f2) + Char.
                ConvertFromUtf32(f3);

            String linea = "";
            String []aux=new String[3];

            //linea inicial
            sw.WriteLine(inicial);

            //nombre de programa
            if (front.prog.Text.Length > 11) sw.WriteLine("%" + front.prog.Text.Substring
                (11));
            else sw.WriteLine("%" + Path.GetFileNameWithoutExtension(filename));

            //lineas de programa
            for (int i = 0; i < miPrograma.getTabla().numeroDeFilas(); i++)
            {
                aux=miPrograma.getTabla().obtenerValoresFila(i);
                linea = aux[0] + " " + aux[1] + " " + aux[2];
                sw.WriteLine(linea.Trim());
            }

            //linea final
            sw.Write(final);

            return 0;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Imaging;
using System.IO;
using SimuladorCNC.Errores;

namespace SimuladorCNC.Menu
{
    class GuardarImagen
    {
        Principal front;
        ControlDePrograma.ControlDePrograma miPrograma;
        SaveFileDialog GUARDARImagen;

        public GuardarImagen(Principal f, SaveFileDialog G)
        {
            front = f;
            GUARDARImagen = G;
            miPrograma = front.getControlDePrograma();
        }

        public int guardaImagen()
        {
            DialogResult resultado = GUARDARImagen.ShowDialog();
            if (resultado == DialogResult.OK)
            {
                if (GUARDARImagen.FileName != "")
                {
                    try
                    {
                        FileStream fs;
                        if ((fs = (FileStream)GUARDARImagen.OpenFile()) != null)
                        {
                            switch (GUARDARImagen.FilterIndex)
                            {
                                case 1:
                                    front.getMesa().getDibujo().Save(fs, ImageFormat.Bmp)
                                        ;
                                    break;
                                case 2:
                                    front.getMesa().getDibujo().Save(fs, ImageFormat.Png)
                                        ;
                                    break;
                                case 3:
                                    front.getMesa().getDibujo().Save(fs, ImageFormat.Gif)
                                        ;
                                    break;
                                case 4:
                                    front.getMesa().getDibujo().Save(fs, ImageFormat.Jpeg
                                        );
                                    break;

                                default:
                                    int error = 60;
                                    Error Er = new Error(error);
                                    Er.ShowDialog();
                                    return -error;
                            }
                            fs.Flush();
                            fs.Close();
                        }
                        else
                        {
                            int error = 61;
                            Error Er = new Error(error);
                            Er.ShowDialog();
                            return -error;
                        }
```

```
                    }
                    catch
                    {
                        int error = 61;
                        Error Er = new Error(error);
                        Er.ShowDialog();
                        return -error;
                    }
                }
                else
                {
                    int error = 60;
                    Error Er = new Error(error);
                    Er.ShowDialog();
                    return -error;
                }

            }
            return 0;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SimuladorCNC.Errores;
using SimuladorCNC.ControlDePrograma;

namespace SimuladorCNC.Menu
{
    partial class ConfBasica : Form
    {
        Principal front;
        Configuracion conf;

        public ConfBasica(Principal F, Configuracion C)
        {
            InitializeComponent();
            front=F;
            conf=C;

            mostrarValores();
        }

        private void mostrarValores()
        {
            TextoAncho.Text = conf.getMesaX().ToString();
            TextoAlto.Text = conf.getMesaY().ToString();

            TextoVelocidad.Text = conf.getPixelesPorFrecuencia().ToString();

            TextoXPuntero.Text = conf.getPunteroXIni().ToString();
            TextoYPuntero.Text = conf.getPunteroYIni().ToString();

            if (conf.getPunteroAbsolutoIni()) RadioAbsoluto.Checked = true;
            else RadioIncremental.Checked = true;

            TextoVelocidadPunteroInicial.Text = conf.getPunteroTraslacionIni().ToString()
                ;
            TextoVelocidadPunteroMaxima.Text = conf.getPunteroTraslacionMax().ToString();
            TextoVelocidadPunteroMinima.Text = conf.getPunteroTraslacionMin().ToString();
        }

        private void Aceptar_Click(object sender, EventArgs e)
        {
            if (revisarValores())
            {
                actualizarValores();
                conf.guardarConfiguracion();
                front.getControlDePrograma().reiniciarMaquina();
                front.getZoom().actualizarMesaYBrazo();
                this.Close();
            }
        }

        private void Cancelar_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void ValoresPorDefecto_Click(object sender, EventArgs e)
        {
            mostrarValoresPorDefecto();
        }

        private void mostrarValoresPorDefecto()
        {
            TextoAncho.Text = conf.getDefMesaX().ToString();
```

```
            TextoAlto.Text = conf.getDefMesaY().ToString();

            TextoVelocidad.Text = conf.getDefPixelesPorFrecuencia().ToString();

            TextoXPuntero.Text = conf.getDefPunteroXIni().ToString();
            TextoYPuntero.Text = conf.getDefPunteroYIni().ToString();

            if (conf.getDefPunteroAbsolutoIni()) RadioAbsoluto.Checked = true;
            else RadioIncremental.Checked = true;

            TextoVelocidadPunteroInicial.Text = conf.getDefPunteroTraslacionIni().
                ToString();
            TextoVelocidadPunteroMaxima.Text = conf.getDefPunteroTraslacionMax().ToString
                ();
            TextoVelocidadPunteroMinima.Text = conf.getDefPunteroTraslacionMin().ToString
                ();
        }

        public bool revisarValores()
        {
            if (!revisarTamanoMesa()) return false;
            if (!revisarPosicionInicialPuntero()) return false;
            if (!revisarVelocidadTraslacion()) return false;
            if (!revisarVelocidadDeDibujo()) return false;
            return true;
        }

        public bool revisarTamanoMesa()
        {
            int error = -1;
            int ancho=0, alto=0;
            if (int.TryParse(TextoAncho.Text, out ancho))
            {
                if (ancho <= 0)
                {
                    error = 40;
                    Error Er = new Error(error, TextoAncho.Text);
                    Er.ShowDialog();
                    return false;
                }
            }
            else
            {
                error = 40;
                Error Er = new Error(error, TextoAncho.Text);
                Er.ShowDialog();
                return false;
            }

            if (int.TryParse(TextoAlto.Text, out alto))
            {
                if (alto <= 0)
                {
                    error = 41;
                    Error Er = new Error(error, TextoAlto.Text);
                    Er.ShowDialog();
                    return false;
                }
            }
            else
            {
                error = 41;
                Error Er = new Error(error, TextoAlto.Text);
                Er.ShowDialog();
                return false;
            }
            return true;
        }

        public bool revisarVelocidadDeDibujo()
```

```
        {
            int error = -1;
            int velocidad = 0;
            if (int.TryParse(TextoVelocidad.Text, out velocidad))
            {
                if (velocidad <= 0)
                {
                    error = 42;
                    Error Er = new Error(error, TextoVelocidad.Text);
                    Er.ShowDialog();
                    return false;
                }
                bool malo = false;
                front.getBrazo().getMovimiento().tiempoTimer(500, int.Parse(
                    TextoVelocidadPunteroMaxima.Text), velocidad, ref malo);
                if (malo)
                {
                    error = 49;
                    int VelMax = VelocidadMovimientoMax(500, 1);
                    Error Er = new Error(error, TextoVelocidad.Text, VelMax.ToString());
                    Er.ShowDialog();
                    return false;
                }
            }
            else
            {
                error = 42;
                Error Er = new Error(error, TextoVelocidad.Text);
                Er.ShowDialog();
                return false;
            }
            return true;
        }

        public bool revisarPosicionInicialPuntero()
        {
            int error = -1;
            int x = 0, y = 0;
            if (int.TryParse(TextoXPuntero.Text, out x))
            {
                if (x < 0)
                {
                    error = 43;
                    Error Er = new Error(error, TextoXPuntero.Text, TextoYPuntero.Text);
                    Er.ShowDialog();
                    return false;
                }
                if (x >= int.Parse(TextoAncho.Text))
                {
                    error = 44;
                    Error Er = new Error(error, TextoXPuntero.Text, TextoYPuntero.Text);
                    Er.ShowDialog();
                    return false;
                }
            }
            else
            {
                error = 43;
                Error Er = new Error(error, TextoXPuntero.Text, TextoYPuntero.Text);
                Er.ShowDialog();
                return false;
            }

            if (int.TryParse(TextoYPuntero.Text, out y))
            {
                if (y < 0)
                {
                    error = 43;
                    Error Er = new Error(error, TextoXPuntero.Text, TextoYPuntero.Text);
                    Er.ShowDialog();
```

```
                return false;
            }
            if (y >= int.Parse(TextoAlto.Text))
            {
                error = 44;
                Error Er = new Error(error, TextoXPuntero.Text, TextoYPuntero.Text);
                Er.ShowDialog();
                return false;
            }
        }
        else
        {
            error = 43;
            Error Er = new Error(error, TextoXPuntero.Text, TextoYPuntero.Text);
            Er.ShowDialog();
            return false;
        }
        return true;
    }

    public bool revisarVelocidadTraslacion()
    {
        int error = -1;
        int velocidad = 0;


        if (int.TryParse(TextoVelocidadPunteroMaxima.Text, out velocidad))
        {
            if (velocidad <= 0)
            {
                error = 46;
                Error Er = new Error(error, TextoVelocidadPunteroMaxima.Text);
                Er.ShowDialog();
                return false;
            }
        }
        else
        {
            error = 46;
            Error Er = new Error(error, TextoVelocidadPunteroMaxima.Text);
            Er.ShowDialog();
            return false;
        }

        if (int.TryParse(TextoVelocidadPunteroMinima.Text, out velocidad))
        {
            if (velocidad <= 0)
            {
                error = 47;
                Error Er = new Error(error, TextoVelocidadPunteroMinima.Text);
                Er.ShowDialog();
                return false;
            }
        }
        else
        {
            error = 47;
            Error Er = new Error(error, TextoVelocidadPunteroMinima.Text);
            Er.ShowDialog();
            return false;
        }
        if (int.TryParse(TextoVelocidadPunteroInicial.Text, out velocidad))
        {
            if (velocidad <= 0)
            {
                error = 45;
                Error Er = new Error(error, TextoVelocidadPunteroInicial.Text);
                Er.ShowDialog();
                return false;
            }
```

```
                if (velocidad < int.Parse(TextoVelocidadPunteroMinima.Text) || velocidad
                    > int.Parse(TextoVelocidadPunteroMaxima.Text))
                {
                    error = 48;
                    Error Er = new Error(error, TextoVelocidadPunteroInicial.Text);
                    Er.ShowDialog();
                    return false;
                }
            }
            else
            {
                error = 45;
                Error Er = new Error(error, TextoVelocidadPunteroInicial.Text);
                Er.ShowDialog();
                return false;
            }
            return true;
        }

        public void actualizarValores()
        {
            conf.setMesaY(int.Parse(TextoAlto.Text));
            conf.setMesaX(int.Parse(TextoAncho.Text));

            conf.setPixelesPorFrecuencia(int.Parse(TextoVelocidad.Text));

            conf.setPunteroXIni(int.Parse(TextoXPuntero.Text));
            conf.setPunteroYIni(int.Parse(TextoYPuntero.Text));

            if (RadioIncremental.Checked == true) conf.setPunteroAbsolutoIni(false);
            if (RadioAbsoluto.Checked == true) conf.setPunteroAbsolutoIni(true);

            conf.setPunteroTraslacionIni(int.Parse(TextoVelocidadPunteroInicial.Text));
            conf.setPunteroTraslacionMax(int.Parse(TextoVelocidadPunteroMaxima.Text));
            conf.setPunteroTraslacionMin(int.Parse(TextoVelocidadPunteroMinima.Text));
        }

        public int VelocidadMovimientoMax(int Freq, int MinimoTiempoMiliSeg)
        {
            return (1000 * 500) / (MinimoTiempoMiliSeg * Freq);
        }
    }
}
```

```
namespace SimuladorCNC.Menu
{
    partial class ConfBasica
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.CuadroConfTabla = new System.Windows.Forms.Label();
            this.TamanoDeLaMesa = new System.Windows.Forms.Label();
            this.ValoresPorDefecto = new System.Windows.Forms.Button();
            this.Aceptar = new System.Windows.Forms.Button();
            this.Cancelar = new System.Windows.Forms.Button();
            this.LabelAlto = new System.Windows.Forms.Label();
            this.LabelAncho = new System.Windows.Forms.Label();
            this.TextoAlto = new System.Windows.Forms.TextBox();
            this.TextoAncho = new System.Windows.Forms.TextBox();
            this.equis = new System.Windows.Forms.Label();
            this.TextoVelocidad = new System.Windows.Forms.TextBox();
            this.label4 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.LabelVelocidad = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.TextoYPuntero = new System.Windows.Forms.TextBox();
            this.TextoXPuntero = new System.Windows.Forms.TextBox();
            this.Te = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.LabelPuntero = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.label9 = new System.Windows.Forms.Label();
            this.RadioAbsoluto = new System.Windows.Forms.RadioButton();
            this.RadioIncremental = new System.Windows.Forms.RadioButton();
            this.label2 = new System.Windows.Forms.Label();
            this.TextoVelocidadPunteroInicial = new System.Windows.Forms.TextBox();
            this.label6 = new System.Windows.Forms.Label();
            this.TextoVelocidadPunteroMaxima = new System.Windows.Forms.TextBox();
            this.label10 = new System.Windows.Forms.Label();
            this.TextoVelocidadPunteroMinima = new System.Windows.Forms.TextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // CuadroConfTabla
            //
            this.CuadroConfTabla.BorderStyle = System.Windows.Forms.BorderStyle.
                FixedSingle;
            this.CuadroConfTabla.Font = new System.Drawing.Font("Microsoft Sans Serif",
                8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.
                Point, ((byte)(0)));
            this.CuadroConfTabla.Location = new System.Drawing.Point(24, 29);
            this.CuadroConfTabla.Name = "CuadroConfTabla";
            this.CuadroConfTabla.Size = new System.Drawing.Size(260, 100);
            this.CuadroConfTabla.TabIndex = 0;
            //
            // TamanoDeLaMesa
            //
            this.TamanoDeLaMesa.AutoSize = true;
            this.TamanoDeLaMesa.Font = new System.Drawing.Font("Microsoft Sans Serif",
                8.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point
                , ((byte)(0)));
            this.TamanoDeLaMesa.Location = new System.Drawing.Point(33, 20);
```

```
            this.TamanoDeLaMesa.Name = "TamanoDeLaMesa";
            this.TamanoDeLaMesa.Size = new System.Drawing.Size(100, 13);
            this.TamanoDeLaMesa.TabIndex = 1;
            this.TamanoDeLaMesa.Text = "Tamaño de la mesa";
            //
            // ValoresPorDefecto
            //
            this.ValoresPorDefecto.Anchor = ((System.Windows.Forms.AnchorStyles)((System.
                Windows.Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.
                Left)));
            this.ValoresPorDefecto.BackColor = System.Drawing.Color.FromArgb(((int)(((
                byte)(204)))), ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.ValoresPorDefecto.ForeColor = System.Drawing.Color.Black;
            this.ValoresPorDefecto.Location = new System.Drawing.Point(25, 397);
            this.ValoresPorDefecto.Name = "ValoresPorDefecto";
            this.ValoresPorDefecto.Size = new System.Drawing.Size(117, 23);
            this.ValoresPorDefecto.TabIndex = 2;
            this.ValoresPorDefecto.Text = "Valores por defecto";
            this.ValoresPorDefecto.UseVisualStyleBackColor = false;
            this.ValoresPorDefecto.Click += new System.EventHandler(this.
                ValoresPorDefecto_Click);
            //
            // Aceptar
            //
            this.Aceptar.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Aceptar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204))))
                , ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.Aceptar.ForeColor = System.Drawing.Color.Black;
            this.Aceptar.Location = new System.Drawing.Point(492, 397);
            this.Aceptar.Name = "Aceptar";
            this.Aceptar.Size = new System.Drawing.Size(75, 23);
            this.Aceptar.TabIndex = 68;
            this.Aceptar.Text = "Aceptar";
            this.Aceptar.UseVisualStyleBackColor = false;
            this.Aceptar.Click += new System.EventHandler(this.Aceptar_Click);
            //
            // Cancelar
            //
            this.Cancelar.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Cancelar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204)))
                ), ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.Cancelar.ForeColor = System.Drawing.Color.Black;
            this.Cancelar.Location = new System.Drawing.Point(392, 397);
            this.Cancelar.Name = "Cancelar";
            this.Cancelar.Size = new System.Drawing.Size(75, 23);
            this.Cancelar.TabIndex = 69;
            this.Cancelar.Text = "Cancelar";
            this.Cancelar.UseVisualStyleBackColor = false;
            this.Cancelar.Click += new System.EventHandler(this.Cancelar_Click);
            //
            // LabelAlto
            //
            this.LabelAlto.AutoSize = true;
            this.LabelAlto.Location = new System.Drawing.Point(161, 70);
            this.LabelAlto.Name = "LabelAlto";
            this.LabelAlto.Size = new System.Drawing.Size(25, 13);
            this.LabelAlto.TabIndex = 70;
            this.LabelAlto.Text = "Alto";
            //
            // LabelAncho
            //
            this.LabelAncho.AutoSize = true;
            this.LabelAncho.Location = new System.Drawing.Point(45, 69);
            this.LabelAncho.Name = "LabelAncho";
            this.LabelAncho.Size = new System.Drawing.Size(38, 13);
            this.LabelAncho.TabIndex = 71;
            this.LabelAncho.Text = "Ancho";
            //
```

```
            // TextoAlto
            //
            this.TextoAlto.Location = new System.Drawing.Point(164, 85);
            this.TextoAlto.Name = "TextoAlto";
            this.TextoAlto.Size = new System.Drawing.Size(91, 20);
            this.TextoAlto.TabIndex = 72;
            this.TextoAlto.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // TextoAncho
            //
            this.TextoAncho.Location = new System.Drawing.Point(48, 85);
            this.TextoAncho.Name = "TextoAncho";
            this.TextoAncho.Size = new System.Drawing.Size(91, 20);
            this.TextoAncho.TabIndex = 73;
            this.TextoAncho.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // equis
            //
            this.equis.AutoSize = true;
            this.equis.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)
                (0)));
            this.equis.Location = new System.Drawing.Point(145, 88);
            this.equis.Name = "equis";
            this.equis.Size = new System.Drawing.Size(13, 13);
            this.equis.TabIndex = 74;
            this.equis.Text = "x";
            //
            // TextoVelocidad
            //
            this.TextoVelocidad.Location = new System.Drawing.Point(335, 46);
            this.TextoVelocidad.Name = "TextoVelocidad";
            this.TextoVelocidad.Size = new System.Drawing.Size(91, 20);
            this.TextoVelocidad.TabIndex = 79;
            this.TextoVelocidad.TextAlign = System.Windows.Forms.HorizontalAlignment.
                Right;
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.label4.Location = new System.Drawing.Point(322, 20);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(156, 13);
            this.label4.TabIndex = 76;
            this.label4.Text = "Velocidad media de movimiento";
            //
            // label5
            //
            this.label5.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.label5.Location = new System.Drawing.Point(313, 29);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(256, 100);
            this.label5.TabIndex = 75;
            //
            // LabelVelocidad
            //
            this.LabelVelocidad.CausesValidation = false;
            this.LabelVelocidad.Location = new System.Drawing.Point(332, 70);
            this.LabelVelocidad.Name = "LabelVelocidad";
            this.LabelVelocidad.Size = new System.Drawing.Size(218, 44);
            this.LabelVelocidad.TabIndex = 80;
            this.LabelVelocidad.Text = "Velocidad de movimiento en pixeles por segundo
                para una velocidad de traslación d" +
    "e 500.";
```

```
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)
                (0)));
            this.label1.Location = new System.Drawing.Point(145, 224);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(13, 13);
            this.label1.TabIndex = 87;
            this.label1.Text = "x";
            //
            // TextoYPuntero
            //
            this.TextoYPuntero.Location = new System.Drawing.Point(164, 221);
            this.TextoYPuntero.Name = "TextoYPuntero";
            this.TextoYPuntero.Size = new System.Drawing.Size(91, 20);
            this.TextoYPuntero.TabIndex = 86;
            this.TextoYPuntero.TextAlign = System.Windows.Forms.HorizontalAlignment.Right
                ;
            //
            // TextoXPuntero
            //
            this.TextoXPuntero.Location = new System.Drawing.Point(48, 221);
            this.TextoXPuntero.Name = "TextoXPuntero";
            this.TextoXPuntero.Size = new System.Drawing.Size(91, 20);
            this.TextoXPuntero.TabIndex = 85;
            this.TextoXPuntero.TextAlign = System.Windows.Forms.HorizontalAlignment.Right
                ;
            //
            // Te
            //
            this.Te.AutoSize = true;
            this.Te.Location = new System.Drawing.Point(161, 205);
            this.Te.Name = "Te";
            this.Te.Size = new System.Drawing.Size(14, 13);
            this.Te.TabIndex = 84;
            this.Te.Text = "Y";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(45, 205);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(14, 13);
            this.label3.TabIndex = 83;
            this.label3.Text = "X";
            //
            // LabelPuntero
            //
            this.LabelPuntero.AutoSize = true;
            this.LabelPuntero.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25
                F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.LabelPuntero.Location = new System.Drawing.Point(33, 149);
            this.LabelPuntero.Name = "LabelPuntero";
            this.LabelPuntero.Size = new System.Drawing.Size(45, 13);
            this.LabelPuntero.TabIndex = 82;
            this.LabelPuntero.Text = "Cabezal";
            //
            // label7
            //
            this.label7.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.label7.Location = new System.Drawing.Point(24, 158);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(545, 214);
```

```
            this.label7.TabIndex = 81;
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(45, 182);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(79, 13);
            this.label8.TabIndex = 88;
            this.label8.Text = "Posición inicial.";
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(296, 182);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(157, 13);
            this.label9.TabIndex = 89;
            this.label9.Text = "Modo de desplazamiento inicial.";
            //
            // RadioAbsoluto
            //
            this.RadioAbsoluto.AutoSize = true;
            this.RadioAbsoluto.Location = new System.Drawing.Point(299, 224);
            this.RadioAbsoluto.Name = "RadioAbsoluto";
            this.RadioAbsoluto.Size = new System.Drawing.Size(66, 17);
            this.RadioAbsoluto.TabIndex = 90;
            this.RadioAbsoluto.TabStop = true;
            this.RadioAbsoluto.Text = "Absoluto";
            this.RadioAbsoluto.UseVisualStyleBackColor = true;
            //
            // RadioIncremental
            //
            this.RadioIncremental.AutoSize = true;
            this.RadioIncremental.Location = new System.Drawing.Point(384, 224);
            this.RadioIncremental.Name = "RadioIncremental";
            this.RadioIncremental.Size = new System.Drawing.Size(80, 17);
            this.RadioIncremental.TabIndex = 91;
            this.RadioIncremental.TabStop = true;
            this.RadioIncremental.Text = "Incremental";
            this.RadioIncremental.UseVisualStyleBackColor = true;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(45, 275);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(120, 13);
            this.label2.TabIndex = 92;
            this.label2.Text = "Velocidad de traslación.";
            //
            // TextoVelocidadPunteroInicial
            //
            this.TextoVelocidadPunteroInicial.Location = new System.Drawing.Point(48, 320
                );
            this.TextoVelocidadPunteroInicial.Name = "TextoVelocidadPunteroInicial";
            this.TextoVelocidadPunteroInicial.Size = new System.Drawing.Size(91, 20);
            this.TextoVelocidadPunteroInicial.TabIndex = 93;
            this.TextoVelocidadPunteroInicial.TextAlign = System.Windows.Forms.
                HorizontalAlignment.Right;
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(45, 304);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(34, 13);
            this.label6.TabIndex = 94;
            this.label6.Text = "Inicial";
            //
```

```
            // TextoVelocidadPunteroMaxima
            //
            this.TextoVelocidadPunteroMaxima.Location = new System.Drawing.Point(299, 320
                );
            this.TextoVelocidadPunteroMaxima.Name = "TextoVelocidadPunteroMaxima";
            this.TextoVelocidadPunteroMaxima.Size = new System.Drawing.Size(91, 20);
            this.TextoVelocidadPunteroMaxima.TabIndex = 95;
            this.TextoVelocidadPunteroMaxima.TextAlign = System.Windows.Forms.
                HorizontalAlignment.Right;
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.Location = new System.Drawing.Point(296, 304);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(43, 13);
            this.label10.TabIndex = 96;
            this.label10.Text = "Máxima";
            //
            // TextoVelocidadPunteroMinima
            //
            this.TextoVelocidadPunteroMinima.Location = new System.Drawing.Point(459, 320
                );
            this.TextoVelocidadPunteroMinima.Name = "TextoVelocidadPunteroMinima";
            this.TextoVelocidadPunteroMinima.Size = new System.Drawing.Size(91, 20);
            this.TextoVelocidadPunteroMinima.TabIndex = 97;
            this.TextoVelocidadPunteroMinima.TextAlign = System.Windows.Forms.
                HorizontalAlignment.Right;
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Location = new System.Drawing.Point(456, 304);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(42, 13);
            this.label11.TabIndex = 98;
            this.label11.Text = "Mínima";
            //
            // ConfBasica
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247)))), ((int)
                (((byte)(249)))), ((int)(((byte)(254)))));
            this.ClientSize = new System.Drawing.Size(596, 442);
            this.Controls.Add(this.label11);
            this.Controls.Add(this.TextoVelocidadPunteroMinima);
            this.Controls.Add(this.label10);
            this.Controls.Add(this.TextoVelocidadPunteroMaxima);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.TextoVelocidadPunteroInicial);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.RadioIncremental);
            this.Controls.Add(this.RadioAbsoluto);
            this.Controls.Add(this.label9);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.TextoYPuntero);
            this.Controls.Add(this.TextoXPuntero);
            this.Controls.Add(this.Te);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.LabelPuntero);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.LabelVelocidad);
            this.Controls.Add(this.TextoVelocidad);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.equis);
            this.Controls.Add(this.TextoAncho);
            this.Controls.Add(this.TextoAlto);
```

```
            this.Controls.Add(this.LabelAncho);
            this.Controls.Add(this.LabelAlto);
            this.Controls.Add(this.Cancelar);
            this.Controls.Add(this.Aceptar);
            this.Controls.Add(this.ValoresPorDefecto);
            this.Controls.Add(this.TamanoDeLaMesa);
            this.Controls.Add(this.CuadroConfTabla);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "ConfBasica";
            this.ShowInTaskbar = false;
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "Configuración";
            this.TopMost = true;
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        private System.Windows.Forms.Label CuadroConfTabla;
        private System.Windows.Forms.Label TamanoDeLaMesa;
        private System.Windows.Forms.Button ValoresPorDefecto;
        private System.Windows.Forms.Button Aceptar;
        private System.Windows.Forms.Button Cancelar;
        private System.Windows.Forms.Label LabelAlto;
        private System.Windows.Forms.Label LabelAncho;
        private System.Windows.Forms.TextBox TextoAlto;
        private System.Windows.Forms.TextBox TextoAncho;
        private System.Windows.Forms.Label equis;
        private System.Windows.Forms.TextBox TextoVelocidad;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label LabelVelocidad;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox TextoYPuntero;
        private System.Windows.Forms.TextBox TextoXPuntero;
        private System.Windows.Forms.Label Te;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label LabelPuntero;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.RadioButton RadioAbsoluto;
        private System.Windows.Forms.RadioButton RadioIncremental;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox TextoVelocidadPunteroInicial;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.TextBox TextoVelocidadPunteroMaxima;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.TextBox TextoVelocidadPunteroMinima;
        private System.Windows.Forms.Label label11;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SimuladorCNC.Mecanismos;

namespace SimuladorCNC.Menu
{
    partial class DetallesCabezal : Form
    {
        Principal front;
        Brazo miBrazo;

        public DetallesCabezal(Principal p, Brazo b)
        {
            front = p;
            miBrazo = b;
            InitializeComponent();
            reiniciarDetallesCabezal();
        }

        public void reiniciarDetallesCabezal()
        {
            posX.Text = front.getConfig().getPunteroXIni().ToString();
            posY.Text = front.getConfig().getPunteroYIni().ToString();
            Tiempo2.Text = "0";
            Distancia2.Text = "0";
            estado.Text = "Subido";
            if (front.getConfig().getPunteroAbsolutoIni()) posicionamiento.Text = "
                Absoluto";
            else posicionamiento.Text = "Relativo";
        }

        public void Actualizar()
        {
            posX.Text = miBrazo.getX().ToString();
            posY.Text = (front.getMesa().transformarY(miBrazo.getY())).ToString();
            if (miBrazo.getBrazoSubido()) estado.Text = "Subido";
            else estado.Text = "Bajado";
            if (miBrazo.getAbsoluto()) posicionamiento.Text = "Absoluto";
            else posicionamiento.Text = "Relativo";
        }

        public void Posicion(int x, int y)
        {
            posX.Text = x.ToString();
            posY.Text = (front.getMesa().transformarY(y)).ToString();
        }

        public void Estado(bool es)
        {
            if (es) estado.Text = "Subido";
            else estado.Text = "Bajado";
        }

        public void Absoluto(bool abs)
        {
            if (abs) posicionamiento.Text = "Absoluto";
            else posicionamiento.Text = "Relativo";
        }

        private void ocultar_Click(object sender, EventArgs e)
        {
            this.Hide();
        }

        public void setTiempo(int t)
```

```
        {
            Tiempo2.Text = t.ToString();
        }
        public void setDistancia(int d)
        {
            Distancia2.Text = d.ToString();
        }

        public int getTiempo()
        {
            return int.Parse(Tiempo2.Text);
        }

        public int getDistancia()
        {
            return int.Parse(Distancia2.Text);
        }
    }
}
```

```
namespace SimuladorCNC.Menu
{
    partial class DetallesCabezal
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.Te = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.posX = new System.Windows.Forms.Label();
            this.posY = new System.Windows.Forms.Label();
            this.posicionamiento = new System.Windows.Forms.Label();
            this.estado = new System.Windows.Forms.Label();
            this.button1 = new System.Windows.Forms.Button();
            this.Tiempo2 = new System.Windows.Forms.Label();
            this.Tiempo3 = new System.Windows.Forms.Label();
            this.Tiempo1 = new System.Windows.Forms.Label();
            this.Distancia1 = new System.Windows.Forms.Label();
            this.Distancia2 = new System.Windows.Forms.Label();
            this.Distancia3 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F, ((
                System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing
                .FontStyle.Italic))), System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(0, 9);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(206, 15);
            this.label1.TabIndex = 0;
            this.label1.Text = "Detalles del cabezal de dibujo.";
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.label8.Location = new System.Drawing.Point(12, 43);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(47, 13);
            this.label8.TabIndex = 89;
            this.label8.Text = "Posición";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(12, 71);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(17, 13);
            this.label3.TabIndex = 90;
            this.label3.Text = "X:";
            //
            // Te
```

```
            //
            this.Te.AutoSize = true;
            this.Te.Location = new System.Drawing.Point(107, 71);
            this.Te.Name = "Te";
            this.Te.Size = new System.Drawing.Size(17, 13);
            this.Te.TabIndex = 91;
            this.Te.Text = "Y:";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.label2.Location = new System.Drawing.Point(12, 102);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(46, 13);
            this.label2.TabIndex = 92;
            this.label2.Text = "Estado: ";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.label4.Location = new System.Drawing.Point(12, 133);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(90, 13);
            this.label4.TabIndex = 93;
            this.label4.Text = "Posicionamiento: ";
            //
            // posX
            //
            this.posX.Location = new System.Drawing.Point(26, 71);
            this.posX.Name = "posX";
            this.posX.Size = new System.Drawing.Size(76, 13);
            this.posX.TabIndex = 94;
            this.posX.TextAlign = System.Drawing.ContentAlignment.TopRight;
            //
            // posY
            //
            this.posY.Location = new System.Drawing.Point(119, 71);
            this.posY.Name = "posY";
            this.posY.Size = new System.Drawing.Size(76, 13);
            this.posY.TabIndex = 95;
            this.posY.TextAlign = System.Drawing.ContentAlignment.TopRight;
            //
            // posicionamiento
            //
            this.posicionamiento.Location = new System.Drawing.Point(108, 133);
            this.posicionamiento.Name = "posicionamiento";
            this.posicionamiento.Size = new System.Drawing.Size(76, 13);
            this.posicionamiento.TabIndex = 96;
            //
            // estado
            //
            this.estado.Location = new System.Drawing.Point(64, 102);
            this.estado.Name = "estado";
            this.estado.Size = new System.Drawing.Size(76, 13);
            this.estado.TabIndex = 97;
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204))))
                , ((int)(((byte)(213))))), ((int)(((byte)(240)))));
            this.button1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
                System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)
                (0)));
            this.button1.ForeColor = System.Drawing.Color.Black;
```

```
            this.button1.Location = new System.Drawing.Point(60, 297);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(80, 24);
            this.button1.TabIndex = 99;
            this.button1.Text = "OCULTAR";
            this.button1.UseVisualStyleBackColor = false;
            this.button1.Click += new System.EventHandler(this.ocultar_Click);
            //
            // Tiempo2
            //
            this.Tiempo2.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Tiempo2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
                System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.Tiempo2.Location = new System.Drawing.Point(45, 193);
            this.Tiempo2.Name = "Tiempo2";
            this.Tiempo2.Size = new System.Drawing.Size(92, 23);
            this.Tiempo2.TabIndex = 102;
            this.Tiempo2.Text = "0";
            this.Tiempo2.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            //
            // Tiempo3
            //
            this.Tiempo3.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Tiempo3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
                System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.Tiempo3.Location = new System.Drawing.Point(141, 194);
            this.Tiempo3.Name = "Tiempo3";
            this.Tiempo3.Size = new System.Drawing.Size(65, 23);
            this.Tiempo3.TabIndex = 101;
            this.Tiempo3.Text = "milisegudos";
            this.Tiempo3.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            //
            // Tiempo1
            //
            this.Tiempo1.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Tiempo1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F, ((
                System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing
                .FontStyle.Italic))), System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.Tiempo1.Location = new System.Drawing.Point(0, 170);
            this.Tiempo1.Name = "Tiempo1";
            this.Tiempo1.Size = new System.Drawing.Size(168, 23);
            this.Tiempo1.TabIndex = 100;
            this.Tiempo1.Text = "Tiempo total transcurrido";
            this.Tiempo1.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            //
            // Distancia1
            //
            this.Distancia1.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows
                .Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Distancia1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F, ((
                System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing
                .FontStyle.Italic))), System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.Distancia1.Location = new System.Drawing.Point(0, 227);
            this.Distancia1.Name = "Distancia1";
            this.Distancia1.Size = new System.Drawing.Size(129, 23);
            this.Distancia1.TabIndex = 103;
            this.Distancia1.Text = "Distancia recorrida";
            this.Distancia1.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            //
            // Distancia2
            //
            this.Distancia2.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows
                .Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
            this.Distancia2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
                System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((
```

```
            byte)(0)));
        this.Distancia2.Location = new System.Drawing.Point(51, 250);
        this.Distancia2.Name = "Distancia2";
        this.Distancia2.Size = new System.Drawing.Size(95, 23);
        this.Distancia2.TabIndex = 105;
        this.Distancia2.Text = "0";
        this.Distancia2.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
        //
        // Distancia3
        //
        this.Distancia3.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows
            .Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
        this.Distancia3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
            System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((
            byte)(0)));
        this.Distancia3.Location = new System.Drawing.Point(152, 250);
        this.Distancia3.Name = "Distancia3";
        this.Distancia3.Size = new System.Drawing.Size(54, 23);
        this.Distancia3.TabIndex = 104;
        this.Distancia3.Text = "milimetros";
        this.Distancia3.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
        //
        // DetallesCabezal
        //
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.None;
        this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247)))), ((int)
            (((byte)(249)))), ((int)(((byte)(254)))));
        this.CausesValidation = false;
        this.ClientSize = new System.Drawing.Size(207, 343);
        this.ControlBox = false;
        this.Controls.Add(this.Distancia1);
        this.Controls.Add(this.Distancia2);
        this.Controls.Add(this.Distancia3);
        this.Controls.Add(this.Tiempo2);
        this.Controls.Add(this.Tiempo3);
        this.Controls.Add(this.Tiempo1);
        this.Controls.Add(this.button1);
        this.Controls.Add(this.estado);
        this.Controls.Add(this.posicionamiento);
        this.Controls.Add(this.posY);
        this.Controls.Add(this.posX);
        this.Controls.Add(this.label4);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.Te);
        this.Controls.Add(this.label3);
        this.Controls.Add(this.label8);
        this.Controls.Add(this.label1);
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
        this.MaximizeBox = false;
        this.Name = "DetallesCabezal";
        this.ShowIcon = false;
        this.ShowInTaskbar = false;
        this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
        this.Text = "Cabezal";
        this.TopMost = true;
        this.ResumeLayout(false);
        this.PerformLayout();
    }

    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label8;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label Te;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label4;
    private System.Windows.Forms.Label posX;
    private System.Windows.Forms.Label posY;
    private System.Windows.Forms.Label posicionamiento;
    private System.Windows.Forms.Label estado;
    private System.Windows.Forms.Button button1;
```

```
        public System.Windows.Forms.Label Tiempo2;
        public System.Windows.Forms.Label Tiempo3;
        public System.Windows.Forms.Label Tiempo1;
        public System.Windows.Forms.Label Distancia1;
        public System.Windows.Forms.Label Distancia2;
        public System.Windows.Forms.Label Distancia3;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Drawing;
using System.Text;
using SimuladorCNC.Errores;

namespace SimuladorCNC.Menu
{
    class InsertarCodigo
    {
        Principal front;
        Panel PanelCodigo;

        public InsertarCodigo(Principal f)
        {
            front = f;
            PanelCodigo = front.PanelCodigo;
        }

        public void ReiniciarInsertarCodigo()
        {
            PanelCodigo.Visible = false;
            front.RadioPosicionamiento.Checked = true;
            front.incrementar_filas.Checked = false;
            RadioPosicionamientoChecked();
            renovarValores();
        }

        public void deshabilitarInsertarCodigo()
        {
            PanelCodigo.Visible = false;
        }

        public void habilitarInsertarCodigo()
        {
            PanelCodigo.Visible = true;
        }

        public void inicioMarcha()
        {
            front.Insertar.Enabled = false;
        }

        public void finMarcha()
        {
            front.Insertar.Enabled = true;
        }

        private void sinCuadros()
        {
            front.LabelTiempo.Visible = false;
            front.TextoTiempo.Visible = false;
            front.labelSegundos.Visible = false;

            front.LabelF.Visible = false;
            front.LabelX.Visible = false;
            front.LabelY.Visible = false;
            front.LabelI.Visible = false;
            front.LabelJ.Visible = false;
            front.TextoF.Visible = false;
            front.TextoX.Visible = false;
            front.TextoY.Visible = false;
            front.TextoI.Visible = false;
            front.TextoJ.Visible = false;
        }

        private void renovarValores()
        {
            front.TextoTiempo.Text = "0";
```

```
            front.TextoF.Text = "0";
            front.TextoX.Text = "0";
            front.TextoY.Text = "0";
            front.TextoI.Text = "0";
            front.TextoJ.Text = "0";
        }

        public void RadioSubirHerramientaChecked()
        {
            sinCuadros();
        }

        public void RadioBajarHerramientaChecked()
        {
            sinCuadros();
        }

        public void RadioPosicionamientoChecked()
        {
            front.LabelTiempo.Visible = false;
            front.TextoTiempo.Visible = false;
            front.labelSegundos.Visible = false;

            front.LabelF.Visible = false;
            front.LabelX.Visible = true;
            front.LabelY.Visible = true;
            front.LabelI.Visible = false;
            front.LabelJ.Visible = false;
            front.TextoF.Visible = false;
            front.TextoX.Visible = true;
            front.TextoY.Visible = true;
            front.TextoI.Visible = false;
            front.TextoJ.Visible = false;
        }

        public void RadioPausaChecked()
        {
            front.LabelTiempo.Visible = true;
            front.TextoTiempo.Visible = true;
            front.labelSegundos.Visible = true;

            front.LabelF.Visible = false;
            front.LabelX.Visible = false;
            front.LabelY.Visible = false;
            front.LabelI.Visible = false;
            front.LabelJ.Visible = false;
            front.TextoF.Visible = false;
            front.TextoX.Visible = false;
            front.TextoY.Visible = false;
            front.TextoI.Visible = false;
            front.TextoJ.Visible = false;
        }

        public void RadioLinealChecked()
        {
            front.LabelTiempo.Visible = false;
            front.TextoTiempo.Visible = false;
            front.labelSegundos.Visible = false;

            front.LabelF.Visible = true;
            front.LabelX.Visible = true;
            front.LabelY.Visible = true;
            front.LabelI.Visible = false;
            front.LabelJ.Visible = false;
            front.TextoF.Visible = true;
            front.TextoX.Visible = true;
            front.TextoY.Visible = true;
            front.TextoI.Visible = false;
            front.TextoJ.Visible = false;
```

```
        }

        public void RadioCircularHorariaChecked()
        {
            front.LabelTiempo.Visible = false;
            front.TextoTiempo.Visible = false;
            front.labelSegundos.Visible = false;

            front.LabelF.Visible = true;
            front.LabelX.Visible = true;
            front.LabelY.Visible = true;
            front.LabelI.Visible = true;
            front.LabelJ.Visible = true;
            front.TextoF.Visible = true;
            front.TextoX.Visible = true;
            front.TextoY.Visible = true;
            front.TextoI.Visible = true;
            front.TextoJ.Visible = true;
        }

        public void RadioCircularAntiHorariaChecked()
        {
            front.LabelTiempo.Visible = false;
            front.TextoTiempo.Visible = false;
            front.labelSegundos.Visible = false;

            front.LabelF.Visible = true;
            front.LabelX.Visible = true;
            front.LabelY.Visible = true;
            front.LabelI.Visible = true;
            front.LabelJ.Visible = true;
            front.TextoF.Visible = true;
            front.TextoX.Visible = true;
            front.TextoY.Visible = true;
            front.TextoI.Visible = true;
            front.TextoJ.Visible = true;
        }

        public void Insertar_Click()
        {
            bool correcto = true;
            int fila = buscarFilaSeleccionada();
            if (fila < 0) return;
            String linea = "";
            String codigo = "";
            String parametros = "";
            correcto = rellenarParametros(ref codigo, ref parametros);
            if (!correcto) return;
            if (front.incrementar_filas.Checked) linea = redimensionarFilas(fila, false);
            else linea = averiguarFila(fila, false);
            front.getTabla().insertarNuevaFilaEnFila(fila, linea, codigo, parametros);
        }

        //si no desplaza se utiliza este
        public String averiguarFila(int fila, bool menuDerecho)
        {
            int menor = 0;
            int mayor = 0;
            String valor;
            //esta al final e incrementa en 10 el valor del final
            if (fila >= front.Tabla.RowCount - 1)
            {
                if (front.Tabla.Rows[fila].Cells[1] != null && front.Tabla.Rows[fila].
                    Cells[1].Value != null)
                {
                    if (((String)front.Tabla.Rows[fila].Cells[1].Value).Equals("M2") &&
                        menuDerecho==false)
                    {
                        menor = -1;
                        for (int i = fila - 2; i > 0; i++)
```

```
                    {
                        menor = comprobarNumeroLinea(i);
                        if (menor != -1) break;
                    }
                    if (menor == -1) menor = 0;
                    mayor = comprobarNumeroLinea(fila);
                }
                else{
                    menor = comprobarNumeroLinea(fila);
                    mayor = 20 + menor;
                }
            }
            else{
                menor = comprobarNumeroLinea(fila);
                mayor = 20 + menor;
            }
        }
        else
        {
            //esta al principio
            if (fila == -1 && menuDerecho == true)
            {
                menor = 0;
                mayor = -1;
                for (int i = fila + 1; i < front.Tabla.RowCount; i++)
                {
                    mayor = comprobarNumeroLinea(i);
                    if (mayor != -1) break;
                }
                if (mayor == -1) mayor = 20 + menor;
            }
            else
            {
                if (fila == 0)
                {
                    menor = comprobarNumeroLinea(fila);
                    mayor = -1;
                    for (int i = fila + 1; i < front.Tabla.RowCount; i++)
                    {
                        mayor = comprobarNumeroLinea(i);
                        if (mayor != -1) break;
                    }
                    if (mayor == -1) mayor = 20 + menor;
                }
                //esta entre medias
                else
                {
                    menor = comprobarNumeroLinea(fila);
                    mayor = -1;
                    for (int i = fila + 1; i < front.Tabla.RowCount; i++)
                    {
                        mayor = comprobarNumeroLinea(i);
                        if (mayor != -1) break;
                    }
                    if (mayor == -1) mayor = 20 + menor;
                }
            }
        }
        int aux = (mayor - menor) / 2;
        if ((menor >= mayor) || ((aux + menor) == mayor) || ((aux + menor) == menor)
            || (aux < 0) || (menor < 0) || (mayor < 0))
        {
            int error = 74;
            Error Er = new Error(error);
            Er.ShowDialog();
            valor = "N???";
        }
        else valor = "N" + (aux+menor).ToString();
        return valor;
    }
```

```
        //si desplaza se utiliza este, el nuevo es averiguarFilas
        public String redimensionarFilas(int linea, bool menuDerecho)
        {
            int menor = 0;
            int mayor = 0;
            int incremento = 0;
            int valor = 0;
            if ((linea >= front.Tabla.RowCount - 1) && menuDerecho == false)
            {
                mayor = comprobarNumeroLinea(linea);
                menor = comprobarNumeroLinea(linea - 1);
            }
            else
            {
                mayor = comprobarNumeroLinea(linea + 1);
                menor = comprobarNumeroLinea(linea);
            }
            if (linea == -1 && menuDerecho == true) menor = 0;
            if ((mayor <= menor) || (mayor < 0) || (menor < 0) || (linea==-1 &&
                menuDerecho==true)) incremento = 10;
            else incremento = mayor - menor;

            if (menor < 0)
            {
                int error = 74;
                Error Er = new Error(error);
                Er.ShowDialog();
                return "N???";
            }
            else valor = menor;

            for (int i = 0; i < front.Tabla.RowCount; i++)
            {
                int aux = comprobarNumeroLinea(i);
                if (aux > valor)
                {
                    aux += incremento;
                    front.Tabla.Rows[i].Cells[0].Value = "N" + aux.ToString();
                }
            }
            valor += incremento;
            return "N" + valor.ToString();
        }

        public int comprobarNumeroLinea(int linea)
        {
            int error = -1;
            if (linea < 0 || linea >= front.Tabla.RowCount) return error;
            String numero = (String)front.Tabla.Rows[linea].Cells[0].Value;
            if (numero != null)
            {
                if (!numero.Trim().Equals(""))
                {
                    if (numero[0] == 'N')
                    {
                        int num = 0;
                        if (int.TryParse(numero.Substring(1), out num))
                        {
                            if (num >= 0) return num;
                        }
                    }
                }
            }
            return error;
        }

        public int buscarFilaSeleccionada()
        {
            int error = 0;
```

```
            if (front.Tabla.SelectedRows.Count <= 0)
            {
                error = 70;
                Error Er = new Error(error);
                Er.ShowDialog();
                return -error;
            }
            else
            {
                DataGridViewRow fila = front.Tabla.SelectedRows[0];
                return fila.Index;
            }
        }

        private bool rellenarParametros(ref String codigo, ref String parametros)
        {
            bool correcto = true;

            int F = 0, X = 0, Y = 0, I = 0, J = 0, tiempo = 0;

            if (front.RadioPosicionamiento.Checked)
            {
                codigo = "G0";
                correcto = comprobarPosicionX(front.TextoX.Text, ref X);
                if (!correcto) return false;
                correcto = comprobarPosicionY(front.TextoY.Text, ref Y);
                if (!correcto) return false;
                parametros = "X" + X.ToString() + " Y" + Y.ToString();
                return true;
            }

            if (front.RadioSubirHerramienta.Checked)
            {
                codigo = "S-1";
                parametros = "";
                return true;
            }

            if (front.RadioBajarHerramienta.Checked)
            {
                codigo = "S1";
                parametros = "";
                return true;
            }

            if (front.RadioPausa.Checked)
            {
                codigo = "G4";
                correcto = comprobarTiempoPausa(front.TextoTiempo.Text, ref tiempo);
                if (!correcto) return false;
                else parametros = "P" + tiempo.ToString();
                return true;
            }

            if (front.RadioLineal.Checked)
            {
                codigo = "G1";
                correcto = comprobarVelocidadTraslacion(front.TextoF.Text, ref F);
                if (!correcto) return false;
                correcto = comprobarPosicionX(front.TextoX.Text, ref X);
                if (!correcto) return false;
                correcto = comprobarPosicionY(front.TextoY.Text, ref Y);
                if (!correcto) return false;
                parametros = "F" + F.ToString() + " X" + X.ToString() + " Y" + Y.ToString
                    ();
                return true;
            }
            if (front.RadioCircularHoraria.Checked)
            {
```

```
                codigo = "G2";
                correcto = comprobarVelocidadTraslacion(front.TextoF.Text, ref F);
                if (!correcto) return false;
                correcto = comprobarPosicionX(front.TextoX.Text, ref X);
                if (!correcto) return false;
                correcto = comprobarPosicionY(front.TextoY.Text, ref Y);
                if (!correcto) return false;
                correcto = comprobarPosicionI(front.TextoI.Text, ref I);
                if (!correcto) return false;
                correcto = comprobarPosicionJ(front.TextoJ.Text, ref J);
                if (!correcto) return false;
                parametros = "F" + F.ToString() + " X" + X.ToString() + " Y" + Y.ToString
                    () + " I" + I.ToString() + " J" + J.ToString();
                return true;
            }
            if (front.RadioCircularAntiHoraria.Checked)
            {
                codigo = "G3";
                correcto = comprobarVelocidadTraslacion(front.TextoF.Text, ref F);
                if (!correcto) return false;
                correcto = comprobarPosicionX(front.TextoX.Text, ref X);
                if (!correcto) return false;
                correcto = comprobarPosicionY(front.TextoY.Text, ref Y);
                if (!correcto) return false;
                correcto = comprobarPosicionI(front.TextoI.Text, ref I);
                if (!correcto) return false;
                correcto = comprobarPosicionJ(front.TextoJ.Text, ref J);
                if (!correcto) return false;
                parametros = "F" + F.ToString() + " X" + X.ToString() + " Y" + Y.ToString
                    () + " I" + I.ToString() + " J" + J.ToString();
                return true;
            }
            return false;
        }

        private bool comprobarTiempoPausa(String x, ref int tiempo)
        {
            bool correcto = extraerNumeroValido(front.TextoTiempo.Text, ref tiempo);
            if (!correcto)
            {
                int error = 71;
                Error Er = new Error(error, "El tiempo");
                Er.ShowDialog();
                return false;
            }
            if (tiempo < 0)
            {
                int error = 76;
                Error Er = new Error(error, "El tiempo");
                Er.ShowDialog();
                return false;
            }
            return true;
        }


        private bool comprobarVelocidadTraslacion(String x, ref int F)
        {
            int error = 0;
            bool correcto = extraerNumeroValido(x, ref F);

            if ((!correcto) || (F <= 0))
            {
                error = 75;
                Error Er = new Error(error, "La velocidad de traslacion");
                Er.ShowDialog();
                return false;
            }
            if (F < front.getConfig().getPunteroTraslacionMin())
            {
```

```
                error = 20;
                Error Er = new Error(error, F.ToString(), front.getConfig().
                    getPunteroTraslacionMin().ToString());
                Er.ShowDialog();
                return false;
            }

            if (F > front.getConfig().getPunteroTraslacionMax())
            {
                error = 21;
                Error Er = new Error(error, F.ToString(), front.getConfig().
                    getPunteroTraslacionMax().ToString());
                Er.ShowDialog();
                return false;
            }

            return true;
        }

        private bool comprobarPosicionX(String x, ref int X)
        {
            int error = 0;
            bool correcto = extraerNumeroValido(x, ref X);
            if (!correcto)
            {
                error = 71;
                Error Er = new Error(error, "La posición X");
                Er.ShowDialog();
                return false;
            }
            if (X >= front.getConfig().getMesaX())
            {
                error = 72;
                Error Er = new Error(error, "La posición X");
                Er.ShowDialog();
                return false;
            }
            return true;
        }

        private bool comprobarPosicionY(String y, ref int Y)
        {
            int error = 0;
            bool correcto = extraerNumeroValido(y, ref Y);
            if (!correcto)
            {
                error = 71;
                Error Er = new Error(error, "La posición Y");
                Er.ShowDialog();
                return false;
            }
            if (Y >= front.getConfig().getMesaY())
            {
                error = 73;
                Error Er = new Error(error, "La posición Y");
                Er.ShowDialog();
                return false;
            }
            return true;
        }

        private bool comprobarPosicionI(String i, ref int I)
        {
            int error = 0;
            bool correcto = extraerNumeroValido(i, ref I);
            if (!correcto)
            {
                error = 71;
                Error Er = new Error(error, "La posición I");
                Er.ShowDialog();
```

```
                return false;
            }
            return true;
        }

        private bool comprobarPosicionJ(String j, ref int J)
        {
            int error = 0;
            bool correcto = extraerNumeroValido(j, ref J);
            if (!correcto)
            {
                error = 71;
                Error Er = new Error(error, "La posición J");
                Er.ShowDialog();
                return false;
            }
            return true;
        }

        public bool extraerNumeroValido(String x, ref int N)
        {
            int valor = 0;

            if (int.TryParse(x, out valor))
            {
                N = valor;
                return true;
            }
            else return false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using SimuladorCNC.Botones;

namespace SimuladorCNC.Menu
{
    class Marcha
    {
        Principal front;

        play miPlay;
        Stop miStop;
        pasoMas miPasoMas;
        pasoMenos miPasoMenos;
        final miFinal;
        inicio miInicio;
        InsertarCodigo miInsertarC;
        MenuDerecho miMenuDerecho;

        public Marcha(Principal f)
        {
            front = f;
            miPlay = new play(front);
            miStop = new Stop(front);
            miPasoMas = new pasoMas(front);
            miPasoMenos = new pasoMenos(front);
            miFinal = new final(front);
            miInicio = new inicio(front);
            miInsertarC = front.getInsertarCodigo();
            miMenuDerecho = new MenuDerecho(front);
        }

        public MenuDerecho getMenuDerecho()
        {
            return miMenuDerecho;
        }

        public void habilitarMarcha()
        {
            if (front.getControlDePrograma().programaPreparado() && front.
                getControlDePrograma().Correcto())
            {
                front.MenuNuevo.Enabled = true;
                front.MenuAbrir.Enabled = true;
                front.MenuGuardarComo.Enabled = true;
                front.MenuGuardar.Enabled = true;
                front.MenuConfiguracion.Enabled = true;
                front.MenuExportarImagenDeLaMesa.Enabled = true;
                front.MenuSimulacion.Enabled = true;

                miPlay.habilitarPlay();
                miPasoMas.habilitarPasoMas();
                miPasoMenos.habilitarPasoMenos();
                miFinal.habilitarFinal();
                miInicio.habilitarInicio();
                miStop.desabilitarStop();
                miInsertarC.habilitarInsertarCodigo();
            }
        }

        public void desabilitarMarcha()
        {
            front.MenuNuevo.Enabled = true;
            front.MenuAbrir.Enabled = true;
            front.MenuGuardarComo.Enabled = false;
            front.MenuGuardar.Enabled = false;
            front.MenuConfiguracion.Enabled = true;
            front.MenuExportarImagenDeLaMesa.Enabled = true;
            front.MenuSimulacion.Enabled = false;
```

```
            miPlay.desabilitarPlay();
            miPasoMas.desabilitarPasoMas();
            miPasoMenos.desabilitarPasoMenos();
            miFinal.desabilitarFinal();
            miInicio.desabilitarInicio();
            miStop.desabilitarStop();
            miInsertarC.deshabilitarInsertarCodigo();
        }

        public void inicioDeLaMarcha()
        {
            miPlay.inicioDeLaMarcha();
            miStop.inicioDeLaMarcha();
            miInsertarC.inicioMarcha();

            miPasoMas.desabilitarPasoMas();
            miPasoMenos.desabilitarPasoMenos();
            miFinal.desabilitarFinal();
            miInicio.desabilitarInicio();

            front.MenuNuevo.Enabled = false;
            front.MenuAbrir.Enabled = false;
            front.MenuGuardarComo.Enabled = true;
            front.MenuGuardar.Enabled = true;
            front.MenuConfiguracion.Enabled = false;
            front.MenuExportarImagenDeLaMesa.Enabled = true;
            front.MenuSimulacion.Enabled = true;

            front.Tabla.Enabled = false;
        }

        public void finDeLaMarcha()
        {
            miPlay.finDeLaMarcha();
            miStop.finDeLaMarcha();
            miInsertarC.finMarcha();

            miPasoMas.habilitarPasoMas();
            miPasoMenos.habilitarPasoMenos();
            miFinal.habilitarFinal();
            miInicio.habilitarInicio();

            front.MenuNuevo.Enabled = true;
            front.MenuAbrir.Enabled = true;
            front.MenuGuardarComo.Enabled = true;
            front.MenuGuardar.Enabled = true;
            front.MenuConfiguracion.Enabled = true;
            front.MenuSimulacion.Enabled = true;
            front.MenuExportarImagenDeLaMesa.Enabled = true;

            front.Tabla.Enabled = true;
        }

        public Stop getStop()
        {
            return miStop;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Text;
using System.Drawing;
using SimuladorCNC.ControlDePrograma;

namespace SimuladorCNC.Menu
{
    class MenuDerecho
    {
        int MenuDerechoFila;
        Principal front;
        Point posicionDelRaton;

        public MenuDerecho(Principal f)
        {
            front = f;
            MenuDerechoFila = 0;
            posicionDelRaton = new Point();
        }

        public void reiniciarMenuDerecho()
        {
            MenuDerechoFila = 0;
        }

        public void habilitarMenuDerecho()
        {
            //front.Tabla.ContextMenuStrip = front.MenuDerechoTabla;
        }

        public void desabilitarMenuDerecho()
        {
            reiniciarMenuDerecho();
        }

        public int getMenuDerechoFila()
        {
            return MenuDerechoFila;
        }
        public void setPosicionDelRaton(Point p)
        {
            posicionDelRaton = p;
        }

        public void MenuDerechoFila_Click(DataGridViewCellMouseEventArgs e)
        {

            MenuDerechoFila = e.RowIndex;
            front.MenuDerechoTabla.Show(front.Tabla, posicionDelRaton);
            front.getTabla().desedicionDeCelda();
        }

        public void AvanzarAEstaLinea_Click()
        {
            front.getControlDePrograma().getEjecucion().pasoHasta(MenuDerechoFila);
        }

        public void NuevaFilaAnterior_Click()
        {
            if (MenuDerechoFila >= 0) front.getTabla().insertarFilaVacia(MenuDerechoFila
                - 1, true);
        }

        public void NuevaFilaSiguiente_Click()
        {
            if (MenuDerechoFila >= 0) front.getTabla().insertarFilaVacia(MenuDerechoFila,
                false);
        }
```

```
        public void BorrarFila_Click()
        {
            front.getTabla().eliminarFila(MenuDerechoFila);
        }
    }
}
```

A5.5.- Botones.

El grupo **Botones** contiene 6 clases relacionadas con este grupo:

- **Inicio.**
- **Final.**
- **Paso más.**
- **Paso menos.**
- **Play.**
- **Stop.**

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SimuladorCNC.Botones
{
    class inicio
    {
        Principal front;

        public inicio(Principal f)
        {
            front = f;
        }

        public void habilitarInicio()
        {
            front.inicio.BackgroundImage = SimuladorCNC.Properties.Resources.inicio;
            front.inicio.Enabled = true;
            front.MenuInicio.Enabled = true;
        }

        public void desabilitarInicio()
        {
            front.inicio.BackgroundImage = SimuladorCNC.Properties.Resources.inicioB;
            front.inicio.Enabled = false;
            front.MenuInicio.Enabled = false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SimuladorCNC.Botones
{
    class final
    {
        Principal front;

        public final(Principal f)
        {
            front = f;
        }

        public void habilitarFinal()
        {
            front.Final.BackgroundImage = SimuladorCNC.Properties.Resources.final;
            front.Final.Enabled = true;
            front.MenuFinal.Enabled = true;
        }

        public void desabilitarFinal()
        {
            front.Final.BackgroundImage = SimuladorCNC.Properties.Resources.finalB;
            front.Final.Enabled = false;
            front.MenuFinal.Enabled = false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SimuladorCNC.Botones
{
    class pasoMas
    {
        Principal front;

        public pasoMas(Principal f)
        {
            front = f;
        }

        public void habilitarPasoMas()
        {
            front.pasoMas.BackgroundImage = SimuladorCNC.Properties.Resources.pasoMas;
            front.pasoMas.Enabled = true;
            front.MenuSiguiente.Enabled = true;
        }

        public void desabilitarPasoMas()
        {
            front.pasoMas.BackgroundImage = SimuladorCNC.Properties.Resources.pasoMasB;
            front.pasoMas.Enabled = false;
            front.MenuSiguiente.Enabled = false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SimuladorCNC.Botones
{
    class pasoMenos
    {
        Principal front;

        public pasoMenos(Principal f)
        {
            front = f;
        }

        public void habilitarPasoMenos()
        {
            front.pasoMenos.BackgroundImage = SimuladorCNC.Properties.Resources.pasoMenos
                ;
            front.pasoMenos.Enabled = true;
            front.MenuAnterior.Enabled = true;
        }

        public void desabilitarPasoMenos()
        {
            front.pasoMenos.BackgroundImage = SimuladorCNC.Properties.Resources.
                pasoMenosB;
            front.pasoMenos.Enabled = false;
            front.MenuAnterior.Enabled = false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SimuladorCNC.Botones
{
    class play
    {
        Principal front;

        public play(Principal f)
        {
            front = f;
        }

        public void habilitarPlay()
        {
            front.play.BackgroundImage = SimuladorCNC.Properties.Resources.play;
            front.play.Enabled = true;
            front.MenuMarcha.Enabled = true;
            front.MenuMarcha.Visible = true;
        }

        public void desabilitarPlay()
        {
            front.play.BackgroundImage = SimuladorCNC.Properties.Resources.playB;
            front.play.Enabled = false;
            front.MenuMarcha.Enabled = false;
            front.MenuMarcha.Visible = true;
        }

        public void inicioDeLaMarcha()
        {
            front.MenuMarcha.Enabled = false;
            front.MenuMarcha.Visible = false;
            front.play.BackgroundImage = SimuladorCNC.Properties.Resources.playB;
            front.play.Enabled = false;
        }

        public void finDeLaMarcha()
        {
            habilitarPlay();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Text;

namespace SimuladorCNC.Botones
{
    class Stop
    {

        Principal front;

        public Stop(Principal f)
        {
            front = f;
        }

        public void habilitarStop()
        {
            front.stop.BackgroundImage = SimuladorCNC.Properties.Resources.stop;
            front.MenuParo.ForeColor = Color.Black;
            front.stop.Enabled = true;
            front.MenuParo.Enabled = false;
            front.MenuParo.Visible = false;
        }

        public void StopActivo()
        {
            front.stop.BackgroundImage = SimuladorCNC.Properties.Resources.stopR;
            front.MenuParo.ForeColor=Color.Red;
        }

        public void desabilitarStop()
        {
            front.stop.BackgroundImage = SimuladorCNC.Properties.Resources.stopB;
            front.MenuParo.ForeColor=Color.Black;
            front.stop.Enabled = false;
            front.MenuParo.Enabled = false;
            front.MenuParo.Visible = false;
        }

        public void inicioDeLaMarcha()
        {
            front.MenuParo.ForeColor = Color.Black;
            front.MenuParo.Enabled = true;
            front.MenuParo.Visible = true;
            front.stop.BackgroundImage = SimuladorCNC.Properties.Resources.stop;
            front.stop.Enabled = true;
        }

        public void finDeLaMarcha()
        {
            desabilitarStop();
        }
    }
}
```

A5.6.- Gestión de errores.

El grupo **Gestión de errores** contiene 2 clases relacionadas con este grupo:

- **Error.**
- **Error (interfaz).**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SimuladorCNC.Errores
{
    public partial class Error : Form
    {
        public Error()
        {
            InitializeComponent();
            Aceptar.Focus();
            Texto.Text = error(-1, "", "", "");
        }

        public Error(int num, String altText1, String altText2, String altText3)
            : this()
        {
            Texto.Text = error(num, altText1, altText2, altText3);
        }

        public Error(int num, String altText1, String altText2)
            : this()
        {
            Texto.Text = error(num, altText1, altText2, "");
        }

        public Error(int num, String altText1)
            : this()
        {
            Texto.Text = error(num, altText1, "", "");
        }

        public Error(int num)
            : this()
        {
            Texto.Text = error(num, "", "", "");
        }

        public Error(String Error)
            : this()
        {
            Texto.Text = Error;
        }

        private void Aceptar_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private String error(int numErr, String txt1, String txt2, String txt3)
        {
            switch (numErr)
            {
                case 1:
                    return "Error al procesar la linea " + txt1 + ".\nLinea de programa
                        duplicada:\n\n" + txt2;

                case 2:
                    return "Error al procesar la linea " + txt1 + ".\nLinea no valida: \n
                        \n" + txt2;

                case 3:
                    return "Error al procesar la linea " + txt1 + ".\nLinea no reconocida
                        : \n\n" + txt2;
```

```
                case 4:
                    return "Error al procesar el fichero.\nNo existe código final \"M2\".
                        ";

                case 5:
                    return "Error de código.\n\n" + txt1 + " " + txt2 + " " + txt3;

                case 6:
                    return "Error al procesar el programa.\nEl programa no es válido.";

                case 7:
                    return "El archivo no se puede abrir.";

                case 8:
                    return "Error de código.\nLa linea no posee numeracion inicial.\n\n"
                        + txt1 + " " + txt2 + " " + txt3;

                case 9:
                    return "Terminación no esperada del código.";

                case 10:
                    return "Falta código de terminación del programa.";

                case 11:
                    return "Error de seguimiento.\nDestino no encontrado.";

                case 12:
                    return "Error de seguimiento.\nSe ha salido fuera de la mesa.";

                case 14:
                    return "Error de código.\nSe ha detectado por lo menos una posición
                        negativa dentro de una sección de código configurada en modo
                        absoluto.\n\n" + txt1 + " " + txt2 + " " + txt3;

                case 20:
                    return "Error al procesar la velocidad de traslacion.\nLa velocidad "
                        + txt1 + " es menor que la velocidad mínima (" + txt2 + ").";

                case 21:
                    return "Error al procesar la velocidad de traslacion.\nLa velocidad "
                        + txt1 + " es mayor que la velocidad máxima ("+txt2+").";

                case 30:
                    return "Para poder configurar los parametros de la máquina no puede
                        tener ningun programa cargado.";

                case 40:
                    return "Error en el ancho de la mesa: " + txt1 + "\nEl valor tiene
                        que ser un número entero positivo mayor que 0.";

                case 41:
                    return "Error en la altura de la mesa: " + txt1 + "\nEl valor tiene
                        que ser un número entero positivo mayor que 0.";

                case 42:
                    return "Error en la velocidad de movimiento: " + txt1 + "\nEl valor
                        tiene que ser un número entero positivo mayor que 0.";

                case 43:
                    return "Error en la posicion inicial del puntero: " + txt1 +" X "+
                        txt2 +" Y\nLos valores tienen que ser números enteros positivos."
                        ;

                case 44:
                    return "Error en la posicion inicial del puntero: " + txt1 + " X " +
                        txt2 + " Y\nLos valores tienen que ser menores al tamaño de la
                        mesa.";

                case 45:
                    return "Error en la velocidad inicial de traslación del puntero: " +
```

```
                txt1 + "\nEl valor tiene que ser un número entero positivo mayor
                que 0.";

            case 46:
                return "Error en la velocidad máxima de traslación del puntero: " +
                txt1 + "\nEl valor tiene que ser un número entero positivo mayor
                que 0.";

            case 47:
                return "Error en la velocidad mínima de traslación del puntero: " +
                txt1 + "\nEl valor tiene que ser un número entero positivo mayor
                que 0.";

            case 48:
                return "Error en la velocidad inicial de traslación del puntero: " +
                txt1 + "\nEl valor tiene que ser mayor o igual que la velocidad
                mínima de traslación y menor o igual que la velocidad máxima de
                traslación.";

            case 49:
                return "Error en la velocidad de movimiento: " + txt1 + "\nEl valor
                es demasiado grande, valor máximo "+txt2+".";

            case 50:
                return "Error al guardar el archivo de configuración.\nLos cambios en
                la configuración solo serán efectivos hasta que cierre el
                programa.";

            case 60:
                return "Error al guardar archivo.\nEl nombre del archivo no es válido
                .";

            case 61:
                return "Error al guardar archivo.\nEl archivo no se ha podido guardar
                .";

            case 70:
                return "Error al insertar el código.\nDebe seleccionar en la tabla la
                fila anterior a donde se insertará el nuevo código CNC.";

            case 71:
                return "Error al generar el código.\n"+txt1+" debe ser un número.";

            case 72:
                return "Error al generar el código.\n"+txt1+" debe ser menor que la
                anchura de la mesa.";

            case 73:
                return "Error al generar el código.\n" + txt1 + " debe ser menor que
                la altura de la mesa.";

            case 74:
                return "Error al generar el código.\nNo se puede determinar el número
                de linea consecutivo.";

            case 75:
                return "Error al generar el código.\n" + txt1 + " debe ser un número
                positivo mayor que 0.";

            case 76:
                return "Error al generar el código.\n" + txt1 + " debe ser un número
                positivo.";

            case 80:
                return "Debe darle un nombre al programa.";

            case 81:
                return "El número inicial de lineas debe ser un número positivo.";

            case 82:
```

```
                    return "El zoom debe ser un número positivo mayor que cero.";

                default:
                    return "Error desconocido.";
            }
        }
    }
}
```

```
namespace SimuladorCNC.Errores
{
    partial class Error
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.Texto = new System.Windows.Forms.RichTextBox();
            this.Aceptar = new System.Windows.Forms.Button();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.SuspendLayout();
            //
            // Texto
            //
            this.Texto.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(247)))),
                ((int)(((byte)(249)))), ((int)(((byte)(254)))));
            this.Texto.Cursor = System.Windows.Forms.Cursors.Default;
            this.Texto.Font = new System.Drawing.Font("Arial", 11.25F, System.Drawing.
                FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.Texto.ForeColor = System.Drawing.Color.Black;
            this.Texto.Location = new System.Drawing.Point(173, 28);
            this.Texto.Name = "Texto";
            this.Texto.ReadOnly = true;
            this.Texto.ShortcutsEnabled = false;
            this.Texto.Size = new System.Drawing.Size(396, 150);
            this.Texto.TabIndex = 64;
            this.Texto.Text = "";
            //
            // Aceptar
            //
            this.Aceptar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(204))))
                , ((int)(((byte)(213)))), ((int)(((byte)(240)))));
            this.Aceptar.ForeColor = System.Drawing.Color.Black;
            this.Aceptar.Location = new System.Drawing.Point(484, 199);
            this.Aceptar.Name = "Aceptar";
            this.Aceptar.Size = new System.Drawing.Size(75, 23);
            this.Aceptar.TabIndex = 67;
            this.Aceptar.Text = "Aceptar";
            this.Aceptar.UseVisualStyleBackColor = false;
            this.Aceptar.Click += new System.EventHandler(this.Aceptar_Click);
            //
            // pictureBox1
            //
            this.pictureBox1.BackgroundImageLayout = System.Windows.Forms.ImageLayout.
                None;
            this.pictureBox1.Image = global::SimuladorCNC.Properties.Resources.prohibido4
                ;
            this.pictureBox1.Location = new System.Drawing.Point(21, 28);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(124, 125);
            this.pictureBox1.TabIndex = 66;
            this.pictureBox1.TabStop = false;
            //
            // Error
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(93)))), ((int)(
                ((byte)(107)))), ((int)(((byte)(153)))));
```

```
            this.ClientSize = new System.Drawing.Size(591, 236);
            this.Controls.Add(this.Aceptar);
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.Texto);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "Error";
            this.ShowIcon = false;
            this.ShowInTaskbar = false;
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "Error";
            this.TopMost = true;
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.ResumeLayout(false);
        }

        private System.Windows.Forms.RichTextBox Texto;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Button Aceptar;
    }
}
```