



PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE  
ESCUELA DE INGENIERIA

## **CONTROL Y SUPERVISION DE FRESADO UTILIZANDO WEBCAMS**

**ANDRES CONSTANTE ANANIA IGLESIAS**

Tesis presentada a la Dirección de Pregrado  
como parte de los requisitos para optar al grado de  
Ingeniero Civil Electricista

Profesor Supervisor:  
**MIGUEL TORRES TORRITI**

Santiago de Chile, Diciembre 2012

© MMXII, ANDRES CONSTANTE ANANIA IGLESIAS



PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE  
ESCUELA DE INGENIERIA  
Departamento de Ingeniería Eléctrica

## **CONTROL Y SUPERVISION DE FRESADO UTILIZANDO WEBCAMS**

**ANDRES CONSTANTE ANANIA IGLESIAS**

Miembros del Comité:

MIGUEL TORRES TORRITI

MATIAS HUBE GINESTAR

DANI GUZMAN CARMINE

Memoria presentada a la Dirección de Pregrado  
como parte de los requisitos para optar al grado de  
Ingeniero Civil Electricista

Santiago de Chile, Diciembre 2012

© MMXII, ANDRES CONSTANTE ANANIA IGLESIAS

*A mi familia, polola y amigos*

## **AGRADECIMIENTOS**

Doy un especial agradecimiento a los profesores Miguel Torres y Dani Guzmán que siempre estuvieron dispuesto a resolver cualquier tipo de duda y entregarme la máxima ayuda posible. De ellos pude aprender una excelente metodología de trabajo que me servirá durante todo el desarrollo de mi carrera.

Agradezco a Carlos Alvarez y Jorge Reyes por su excelente disposición en cuanto se les solicitó ayuda, o algún material útil para el desarrollo de esta memoria.

## INDICE GENERAL

AGRADECIMIENTOS . . . . .	IV
INDICE DE FIGURAS . . . . .	IX
INDICE DE TABLAS . . . . .	XIII
RESUMEN . . . . .	XIV
ABSTRACT . . . . .	XV
Capítulo 1. INTRODUCCION . . . . .	1
1.1. Control numérico por computador . . . . .	1
1.2. Descripción del problema . . . . .	2
1.3. Motivación . . . . .	3
1.4. Técnicas existentes de control numérico . . . . .	4
1.4.1. Método de lazo abierto . . . . .	4
1.4.2. Métodos de medición de posición . . . . .	5
1.4.3. <i>Software</i> de control numérico por computador . . . . .	14
1.4.4. Motores . . . . .	16
1.4.5. <i>Software</i> de CNC . . . . .	18
1.5. Propuesta de desarrollo para el fresado a través del computador . . . . .	19
Capítulo 2. PRINCIPIO TEORICO Y MODELAMIENTO MATEMATICO DE LA MEDICION Y PROCESAMIENTO DE CONTROL NUMERICO POR COMPUTADOR . . . . .	21
2.1. Sensor para la obtención de imágenes . . . . .	21
2.1.1. Principio de funcionamiento del sensor . . . . .	21
2.1.2. Principales fuentes de error . . . . .	22
2.2. Estimación de centroide . . . . .	24
2.2.1. Método de centroide binario . . . . .	24

2.2.2. Método de centroide analógico . . . . .	25
2.2.3. Método de centroide ponderado . . . . .	25
2.3. Procesamiento de imágenes . . . . .	26
2.3.1. Transformación de espacio de colores . . . . .	27
2.3.2. Filtro Gaussiano . . . . .	27
2.3.3. Filtros de eliminación de <i>speckles</i> . . . . .	28
2.4. Distorsión y corrección del lente . . . . .	30
2.5. Transformación de ejes de coordenadas . . . . .	35
2.5.1. Proyección del plano real . . . . .	35
2.5.2. Corrección de otros ejes . . . . .	35
2.6. Análisis del error . . . . .	36
2.6.1. Error del lente . . . . .	36
2.6.2. Error de lectura del CCD y estimación de centroide . . . . .	37
2.6.3. Error de corrección . . . . .	38
2.6.4. Error mecánico . . . . .	39
 Capítulo 3. SIMULACIONES Y ANALISIS . . . . .	40
3.1. Simulación de CCD . . . . .	40
3.2. Simulación de filtro Gaussiano . . . . .	40
3.3. Simulación de cálculo de posición . . . . .	42
3.3.1. Simulación sin ruido . . . . .	43
3.3.2. Simulación con ruido . . . . .	43
3.3.3. Simulación con ruido más filtro Gaussiano . . . . .	44
3.3.4. Resumen de simulaciones . . . . .	44
 Capítulo 4. IMPLEMENTACION Y PRUEBAS . . . . .	45
4.1. Especificaciones Técnicas . . . . .	45
4.1.1. Dimensiones y especificaciones de la máquina . . . . .	45
4.1.2. Tornillos sin fin y reducciones . . . . .	46
4.1.3. <i>Hardware</i> de control de motores . . . . .	46

4.1.4. Especificaciones del computador . . . . .	46
4.1.5. Cámaras web . . . . .	47
4.1.6. Motores . . . . .	47
4.2. Implementación de <i>software</i> . . . . .	47
4.2.1. Pantalla <i>closed loop</i> en Mach3 . . . . .	48
4.2.2. <i>Software</i> Fress . . . . .	54
4.2.3. Comunicación entre programas . . . . .	62
4.3. Pruebas para selección de parámetros . . . . .	62
4.3.1. Pruebas de error por mal ajuste mecánico . . . . .	62
4.3.2. Pruebas de colores . . . . .	63
<b>Capítulo 5. RESULTADOS EXPERIMENTALES . . . . .</b>	<b>66</b>
5.1. Resultados de pruebas . . . . .	66
5.1.1. Resultados de eficiencia de <i>software</i> . . . . .	66
5.1.2. Errores de medición . . . . .	66
5.1.3. Resultados de prueba de repetibilidad . . . . .	66
5.1.4. Resultados de resolución de medición de movimiento . . . . .	69
5.2. Resultados finales . . . . .	69
5.2.1. Resolución del <i>software</i> Fress . . . . .	69
5.2.2. Resolución máxima . . . . .	72
5.2.3. Pruebas de creación de piezas . . . . .	72
<b>Capítulo 6. CONCLUSIONES Y TRABAJO FUTURO . . . . .</b>	<b>79</b>
6.1. Trabajo futuro . . . . .	80
<b>Referencias . . . . .</b>	<b>82</b>
<b>ANEXO A. Datasheet motores . . . . .</b>	<b>85</b>
<b>ANEXO B. <i>Datasheet hardware</i> de control de motores . . . . .</b>	<b>88</b>
<b>ANEXO C. Manual de <i>hardware</i> de control de motores . . . . .</b>	<b>120</b>

ANEXO D. Manual de uso . . . . .	129
ANEXO E. Software . . . . .	202
ANEXO F. Simulaciones . . . . .	203

## INDICE DE FIGURAS

1.1.	Diagrama de flujo de sistema en lazo abierto. . . . .	2
1.2.	Diagrama de flujo de sistema en lazo cerrado. . . . .	2
1.3.	Representación gráfica de <i>encoder</i> rotativo tipo absoluto. Imagen obtenida desde (Jiménez, 2009). . . . .	5
1.4.	Representación gráfica de <i>encoder</i> de tipo lineal obtenida desde (“Incremental Linear Encoders; Enclosed Models”, 2008). . . . .	6
1.5.	Representación gráfica de <i>encoder</i> de tipo rotativo obtenida desde (Jiménez, 2009). . . . .	7
1.6.	a) Representación gráfica del sensor de proximidad del tipo infrarrojo, b) Distancia versus voltaje de salida del sensor Sharp, modelo GP2Y0A02YK0F obtenida desde su hoja de datos (“Distance Measuring Sensor Unit Measuring distance: 20 to 150 cm Analog output type”, 2006). . . . .	8
1.7.	Representación gráfica del sensor Wheatstone HMC1501-1512 de la empresa Honeywell obtenida desde su hoja de datos. . . . .	10
1.8.	Representación gráfica de un condensador obtenida desde el libro de la referencia (Phillips, 2004). . . . .	11
1.9.	Representación de triangulación laser con, 1) Laser emisor, 2) Apertura, 3) Posición 1, 4) Posición 2, 5) Lector CCD o PSD. Imagen obtenida desde la referencia (Federico, 2011). . . . .	12
1.10.	Cuatro pasos de un motor de <i>stepper</i> simple, obtenida desde el manual (“Theory in AC motors”, 2001). . . . .	17
1.11.	Diagrama explicativo de motores de corriente continua obtenida desde los apuntes (Beauvais, 2003). . . . .	18

2.1.	Analogía del principio de funcionamiento de un CCD, imagen obtenida desde el libro de Janesick (Janesick, 2007). . . . .	22
2.2.	Error de quantización, figura obtenida desde (Guzmán, 2012). . . . .	24
2.3.	Simulación de imagen con error en el CCD. . . . .	26
2.4.	Cilindro de color HSV, imagen obtenida desde la referencia (Guerrero, 2011). . . . .	27
2.5.	Señal Gaussiana con ruido. . . . .	28
2.6.	Señal Gaussiana con ruido filtrada con filtro Gaussiano. . . . .	29
2.7.	Señal Gaussiana sin ruido. . . . .	29
2.8.	Ejemplo de imagen binarizada. . . . .	30
2.9.	Ejemplo de imagen binarizada con filtro de reducción aplicado. . . . .	31
2.10.	Cámara <i>pinhole</i> , imagen obtenida desde obtenida desde (Gary Bradski, 2008). . . . .	31
2.11.	Distorsión radial de las cámaras, imagen obtenida desde obtenida desde (Gary Bradski, 2008). . . . .	33
2.12.	Distorsión tangencial de las cámaras, imagen obtenida desde obtenida desde (Gary Bradski, 2008). . . . .	33
2.13.	Diferencias en los ejes de coordenadas de la cámara y la mesa. . . . .	36
3.1.	Comparación de error de la imagen sin filtrar e imagen filtrada. . . . .	42
3.2.	Pixelación de imagen real, a)Imagen pixelada, b) Imagen real. . . . .	43
4.1.	Pantalla <i>Closed Loop</i> del Mach3. . . . .	49
4.2.	Diagrama de bloques de función <i>Start</i> . . . . .	50
4.3.	Diagrama de bloques del algoritmo de decisión. . . . .	51
4.4.	Diagrama de bloques con función <i>Start from line</i> . . . . .	52
4.5.	Diagrama de bloques con función <i>Scale</i> . . . . .	52
4.6.	<i>Software Fress</i> . . . . .	55
4.7.	Diagrama de bloques de la función <i>Show</i> para seleccionar filtro de colores. . .	56

4.8.	Diagrama de bloques de la función <i>Start</i> para determinar posición y comunicarse con Mach3. . . . .	57
4.9.	Clasificación de colores. . . . .	59
4.10.	Calibración de posicionamiento. . . . .	60
4.11.	Parámetros iniciales de la segunda fase de calibración. . . . .	61
4.12.	Parámetros calibrados de la segunda fase de calibración. . . . .	62
4.13.	Comunicación entre Mach3 y el <i>software</i> Fress. . . . .	63
5.1.	Histograma del error medido en la estimación de centroide producido por el CCD. . . . .	67
5.2.	Error por repetibilidad y error mecánico. . . . .	68
5.3.	Distribución de resolución a 0.01 mm. . . . .	70
5.4.	Resolución con pasos de 0.01 mm. . . . .	71
5.5.	Error RMS de todo el campo de visión del <i>software</i> Fress. . . . .	72
5.6.	Pieza creada computacionalmente desde un <i>software</i> CAD. . . . .	73
5.7.	Pieza creada en lazo abierto sin error. . . . .	74
5.8.	Pieza 2 creada computacionalmente desde un <i>software</i> CAD. . . . .	75
5.9.	Pieza 2 creada con la máquina desde el <i>software</i> Mach3. . . . .	75
5.10.	Pieza creada en lazo abierto y con error en los pasos de los motores. . . . .	76
5.11.	Pieza con error y método 1 aplicado con lazo de control cerrado. . . . .	76
5.12.	Pieza con error y método 2 aplicado con lazo de control cerrado. . . . .	77
5.13.	Pieza sin error y método 2 aplicado con lazo de control cerrado. . . . .	78
F.1.	Simulaciones de estimación de centroide sin ruido para el eje X. . . . .	203
F.2.	Simulaciones de estimación de centroide sin ruido para el eje Y. . . . .	204
F.3.	Simulaciones de estimación de centroide con ruido para el eje Y. . . . .	204
F.4.	Simulaciones de estimación de centroide con ruido para el eje Y. . . . .	205

F.5.	Simulaciones de estimación de centroide con ruido y mitad de luz para el eje Y.	205
F.6.	Simulaciones de estimación de centroide con ruido y mitad de luz para el eje Y.	206
F.7.	Simulaciones de estimación de centroide con ruido y filtro Gaussiano para el eje X.	206
F.8.	Simulaciones de estimación de centroide con ruido y filtro Gaussiano para el eje Y.	207
F.9.	Simulaciones de estimación de centroide con ruido, filtro Gaussiano y mitad de luz para el eje X.	207
F.10.	Simulaciones de estimación de centroide con ruido, filtro Gaussiano y mitad de luz para el eje Y.	208

## **INDICE DE TABLAS**

3.1.	Resultados de simulación del filtro Gaussiano . . . . .	41
4.1.	Especificaciones de la máquina. . . . .	45
4.2.	Especificaciones de los motores. . . . .	48
4.3.	Error mecánico. . . . .	63
4.4.	Error de medición de centroide en pixeles. . . . .	64
4.5.	Error de centroide por tamaños. . . . .	65
5.1.	Error en estimación de centroide por ruido del CCD. . . . .	67

## **RESUMEN**

En esta investigación se aborda el problema de automatización de una máquina fresadora usando un computador digital, poniendo énfasis en la supervisión y optimización del proceso de manufactura de piezas y en la caracterización de los errores mecánicos y de estimación de posición de la fresa.

En el mercado existe una gran variedad de fresadoras automatizadas, pero su elevado costo ( US\$4000) es restrictivo para aplicaciones no industriales. Aquí se propone un método de control de bajo costo ( US\$700), de menor complejidad y con precisión similar a la de las alternativas comerciales.

Se trabajó con una fresadora Clarke MetalWork, tres motores *stepper*, un *hardware* de control de motores *stepper* y el *software* de control CNC Mach3. Adicionalmente, se desarrollo un software para determinar la posición de la fresa, la cual utiliza dos cámaras web para calcular la posición en los tres ejes cartesianos. La supervisión se consigue cerrando el lazo de control, comparando la posición calculada por el programa desarrollado con la posición de la fresa según el *software* Mach3.

El sistema de fresado creado presentó dos errores: error en la posición efectiva de la fresa respecto de la instrucción de movimiento y error en la medición de la posición. El primero es de tipo mecánico, éste se debe al desalineamientos en la máquina y pérdida de pasos en los motores. Se midieron errores aleatorios de  $\pm 0.15$  mm por desalineamiento y de hasta 1 mm por pérdida de pasos. El segundo es de la estimación de la posición que introduce el software desarrollado, obteniéndose una precisión de 0.3 mm. Esta precisión permite una correcta operación automática de la máquina y la posibilidad de detener o corregir los trabajos de fresado de acuerdo a la magnitud de los fallos mecánicos.

**Palabras Claves:** CNC, Fresadora, Centroide, Loop de control, Plate scale, HUE.

## ABSTRACT

This research focuses on the problem of the automation of a milling machine using a digital computer, emphasizing on the supervision and optimization of the manufacturing process of parts, and on the characterization of the mechanical errors and position estimation of the spindle.

The market offers a wide variety of automated milling machines, but their high cost ( U.S.\$4000) is restrictive for non-industrial applications. In this memory I propose a method of low cost ( U.S. \$700) controller, less complex and with a precision level that is similar to commercial alternatives.

The equipment used included a Clarke MetalWork milling machine, three stepper motors, a control hardware for the stepper motor and the Software Mach3 for the CNC control. Additionally, a software was developed to determine the position of the cutter, which uses two webcams to calculate the position in three dimensions. The supervision is achieved by closing the control loop, comparing the position calculated by the software developed, with the position of the cutter from the software Mach3.

There were two types of errors identified: error in the position of the cutter on the move instruction and error in the position measurement. The first error is mechanical and it is due to misalignments in the machine and loss of steps in the motors. Random errors were measured of  $\pm 0.15$  mm because of misalignment and up to 1 mm loss of steps. The second type of error occurs in the estimation of the position which is introduced by the developed software, yielding an accuracy of 0.3 mm. These times and accuracies allow a correct automatic operation of the machine and the chance to stop or correct milling operations according to the magnitude of mechanical failures.

**Keywords:** CNC, Mill, Centroid, Control loop, Plate scale, HUE.

## Capítulo 1. INTRODUCCION

### 1.1. Control numérico por computador

El control numérico por computador (CNC) es empleado para la automatización de la operación de máquinas de corte, tornos e impresoras 3D, entre otras aplicaciones. Los programas CNC describen los pasos de corte por medio de instrucciones de movimiento (comandadas desde un computador digital) referidas a un sistema de coordenadas solidario a la máquina. La manufactura de piezas se consigue por medio de la ejecución de una serie de instrucciones específicas para la geometría de la pieza en cuestión.

La programación de los trabajos de corte se puede realizar de dos maneras: de forma manual y directamente a través de líneas de comando en un lenguaje de programación llamado código-G; o bien de forma automática empleando programas CAM (*Computer-Aided Manufacturing*), los cuales generan instrucciones de movimiento a partir de diseños CAD (*Computer-Aided Design*) que describen la geometría de la pieza de interés (Escalona, 2002).

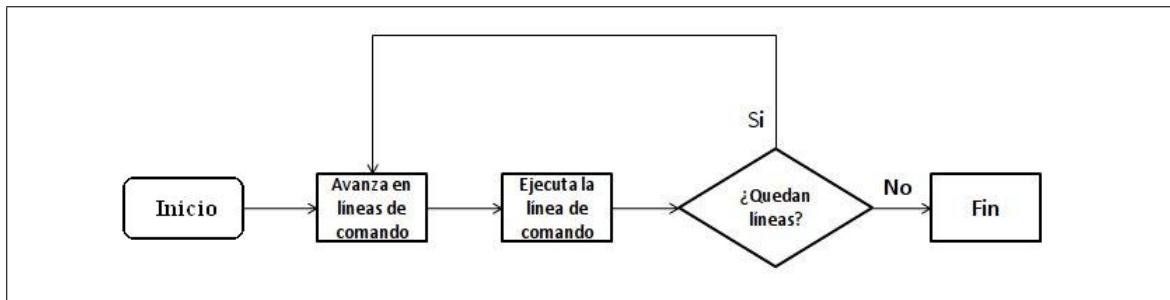
El control CNC puede ser de tipo punto a punto o de contorno. El control punto a punto permite únicamente trayectorias rectas y se utiliza principalmente para trabajos de perforación y fresado en una sola dirección, mientras que el control de contorno realiza interpolaciones, permitiendo así trayectorias curvas y mayor suavidad en los trabajos de corte y fresado.

Estas técnicas tienen la finalidad de acelerar los procesos de producción y facilitar la manufactura de piezas más complejas, lo cual se traduce en un aumento de la productividad, mejores acabados y abaratamiento de costos. Las aplicaciones son múltiples, incluyendo trabajos de carpintería, metalurgia, corte y modelación de plásticos, hasta la producción de instrumentación astronómica de alta precisión.

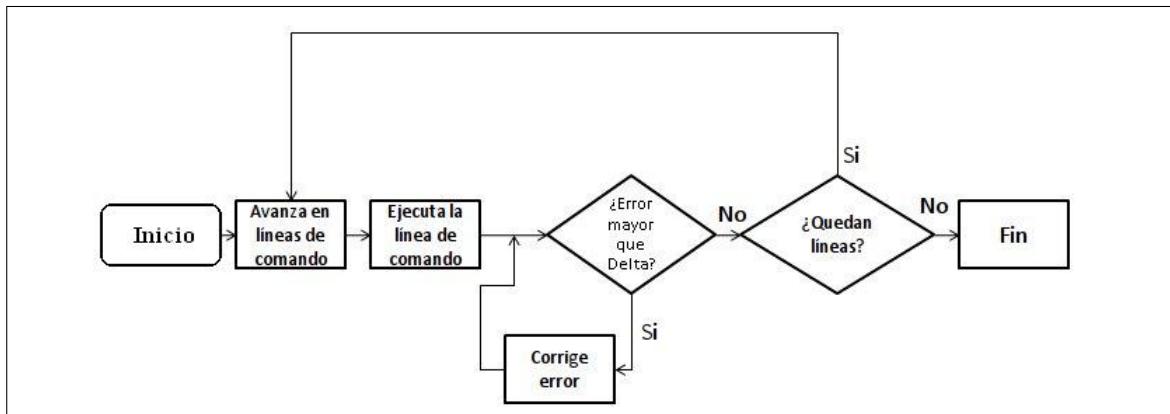
## 1.2. Descripción del problema

El problema consiste en automatizar una máquina fresadora marca Clarke, modelo Metalwork. La máquina posee tres motores *steppers*, dos *limit switches* por cada eje y un computador interno, el cual no cuenta con un sistema de control realimentado para el trabajo de fresado y su diseño es parte del problema de automatización. El problema puede ser dividido en tres etapas, las cuales se describen a continuación.

En primer lugar se consiguió la operación de la máquina en lazo abierto. Para ello se seleccionó el *software* necesario para el diseño de las piezas, tanto a nivel de modelamiento geométrico (programa CAD) como de instrucciones de corte (programa CAM). También se aseguró la correcta operación mecánica y la comunicación entre las partes de la máquina. Las figuras 1.1 y 1.2 describen los lazos de control abierto y cerrado.



**Figura 1.1:** Diagrama de flujo de sistema en lazo abierto.



**Figura 1.2:** Diagrama de flujo de sistema en lazo cerrado.

La segunda etapa (y la más importante) consiste en el desarrollo de un sistema de medición de posición para poder determinar el error en la posición de la fresa, lo

que posibilita cerrar el lazo de control y supervisar el trabajo de fresado. El error de desplazamiento de la fresa se mide respecto de la instrucción de movimiento y se produce principalmente por el salto de pasos en los motores *stepper*. La medición del error permite monitorear el fresado y decidir si un trabajo debe ser interrumpido o si puede ser corregido. La directriz principal en la búsqueda de una solución al problema consiste en el diseño de un sistema que minimice la razón precio/calidad, ya que actualmente el precio es la principal limitación para el acceso a sistemas precisos de control de fresas.

La última etapa consiste en incorporar el sistema de medición desarrollado a la operación de la fresa, ya sea en lazo abierto supervisado o en lazo de control cerrado. Esto implica la correcta comunicación y coordinación entre todas las partes del sistema de medición y la máquina fresadora.

### 1.3. Motivación

Existen múltiples metodologías de medición de posición para máquinas fresadoras, la mayoría de las cuales hace uso de sensores de alta precisión que encarecen la solución final, como por ejemplo *encoders* de resolución del orden de los  $\mu\text{m}$ . Estos sensores tienen un costo de aproximadamente \$70.000 pesos, y se necesitarían tres, uno por cada eje de la máquina. Por otra parte se encuentran los sensores de proximidad, que son menos costosos, pero su resolución bordea los 5 mm, como es el caso del sensor de proximidad infrarrojo Sharp, modelo GP2Y0A02YK0F, con un costo de \$11.870 pesos. (“Distance Measuring Sensor Unit Measuring distance: 20 to 150 cm Analog output type”, 2006).

El procesamiento de imágenes está cobrando cada vez mayor importancia en sistemas de medición y es posible encontrar cámaras de buena calidad a precios relativamente bajos. Por ejemplo las cámaras *Logitech C110*, con resolución máxima de 1024x768 pixeles, tienen un costo de \$9.990 pesos. Dos de estas cámaras serían necesarias para la implementación de un método de medición efectivo, con el cual sería posible llegar a resoluciones a nivel de subpixel. Por ejemplo, si se consigue una estimación de centroide

con resolución de 0.1 pixeles y con un campo de visión de 600 pixeles<sup>2</sup> se podría llegar a una medición de la posición de la fresadora con precisión de hasta 0.004 mm.

A raíz de esto se sugirió un método de medición a través de procesamiento de imágenes, en el cual se toma un objeto conocido de la imagen y se estima su posición, obteniéndose un sistema de bajo costo, fácil implementación y alta precisión.

## 1.4. Técnicas existentes de control numérico

### 1.4.1. Método de lazo abierto

Es posible operar una fresadora sin un método de medición de posición, usando un *loop* de control en lazo abierto similar al lazo que se muestra en la figura 1.1. En este caso se ejecuta un código-G secuencialmente, estimando la posición actual de la máquina como el desplazamiento acumulado de cada una de las instrucciones anteriores de movimiento que fueron enviadas desde un computador a los motores. Esta alternativa no permite la corrección del posible error en la posición de la fresa. Aun más, el error es de tipo acumulativo y se arrastra a lo largo de un trabajo de fresado, produciendo la manufactura de piezas defectuosas.

Pese a esto, es posible utilizar el sistema en lazo abierto debido a que los motores *steppers* se desplazan en magnitudes discretas. Esta magnitud es llamada paso. Cada paso del motor se relaciona con una cantidad de vueltas del tornillo sin fin del eje de la máquina, lo cual, a su vez, se traduce en un desplazamiento efectivo de la fresa. De esta forma se puede relacionar la cantidad de pasos con una magnitud de movimiento lineal.

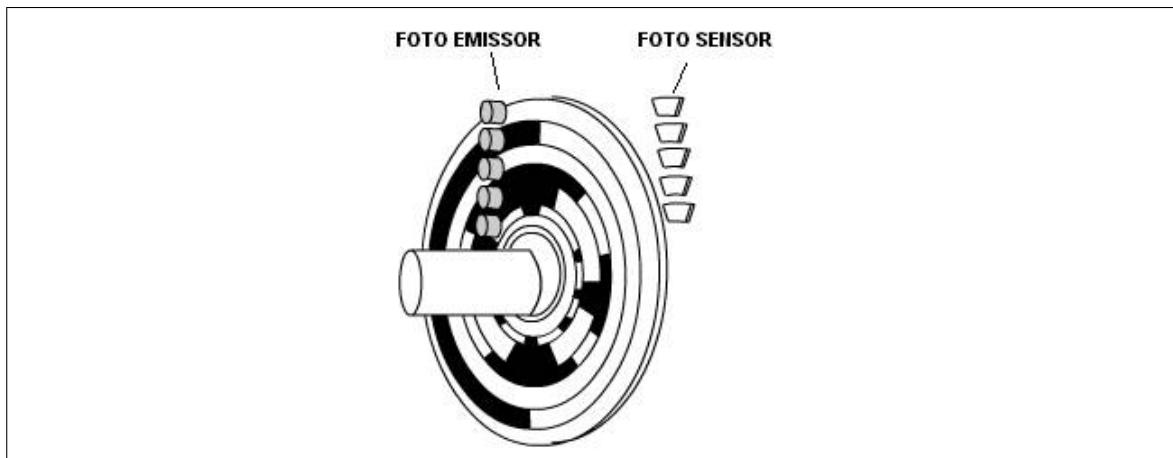
En una sistema CNC bien calibrado se esperaría que el error de posicionamiento en lazo abierto sea mínimo, haciendo posible manufacturar piezas correctamente. La precisión del sistema queda determinado por la resolución de los motores y la mecánica de la máquina fresadora. Sin embargo, en este tipo de sistemas CNC son frecuentes las pérdidas de pasos en los motores y otros errores de desplazamiento, por lo cual es poco aconsejable una operación en lazo abierto.

### 1.4.2. Métodos de medición de posición

En la siguiente sección se hace un breve análisis comparativo de los métodos de medición de posición y de su implementación sobre una máquina fresadora. De esta forma se evalúa la mejor alternativa dadas las restricciones del problema en particular que aquí se aborda. Se detallarán sensores del tipo encoders, de proximidad, triangulación laser y del tipo que utiliza cámaras web.

#### 1.4.2.1. *Encoders*

Los *encoders* son sensores que miden el desplazamiento, ya sea lineal o rotativo. Los de tipo absolutos poseen una marca única para cada distancia, en la figura 1.3 se muestra un ejemplo que por cada posición del disco existe un valor único con zonas transparentes y opacas que interrumpen un haz de luz captado por la lectura de los fotosensores. La resolución de este sensor está determinada por la cantidad de marcas existentes a lo largo del disco.



**Figura 1.3:** Representación gráfica de *encoder* rotativo tipo absoluto. Imagen obtenida desde (Jiménez, 2009).

Los *encoders* incrementales no poseen una posición conocida, al usarlo es necesario definir la posición inicial del sensor. La distancia se determina por la cantidad de movimiento que se produce en éste, cada movimiento de X (En su resolución mínima) produce un pulso positivo o negativo dependiendo del sentido de giro. La posición se caracteriza por la siguiente ecuación:

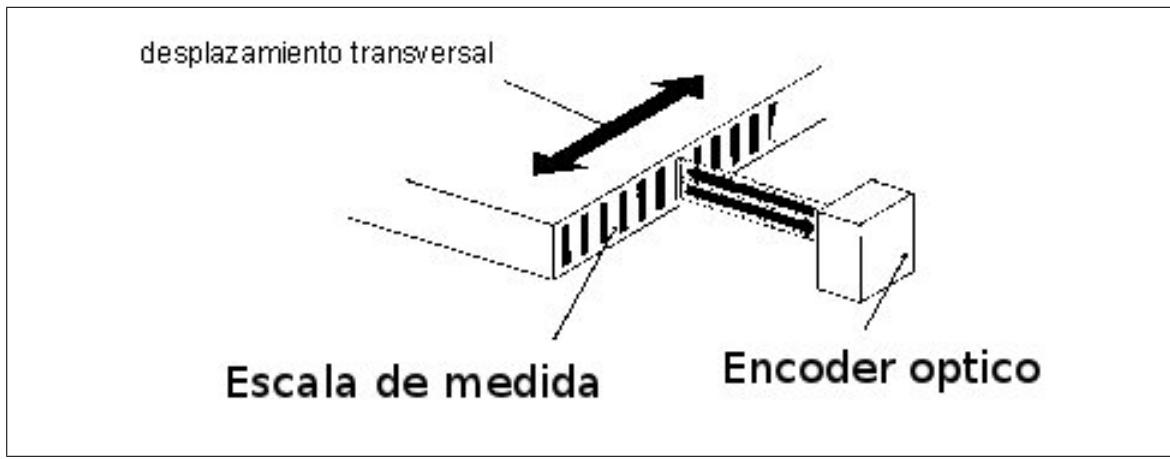
$$\text{Posición}_{actual} = \text{Posición}_{inicial} + \text{Pulsos}_k$$

$$\text{Pulsos}_k = \text{Pulsos}_{k-1} + \text{Pulso}_{producido}$$

Para cada tipo de *encoder* existen dos representaciones físicas, del tipo lineal y del tipo rotativo. A continuación se detallan estos tipos de modalidades:

### **Encoders lineales**

Los encoders lineales son sensores de distancias longitudinales, que para lograr la medición utilizan una guía ubicada en la mesa de la máquina, como se muestra en la figura 1.4. Estos sensores son precisos y pueden alcanzar resoluciones mejores que  $5 \mu\text{m}$ , como por ejemplo el *encoder* GAM-1640-5-A. El precio de estos sensores depende directamente de su resolución, encontrándose en torno a los \$150.000 pesos cada uno. Para la medición de la posición de la fresa son necesarios tres *encoders*, uno por cada eje, alcanzándose un costo total de \$450.000 pesos, por lo cual su uso queda restringido.

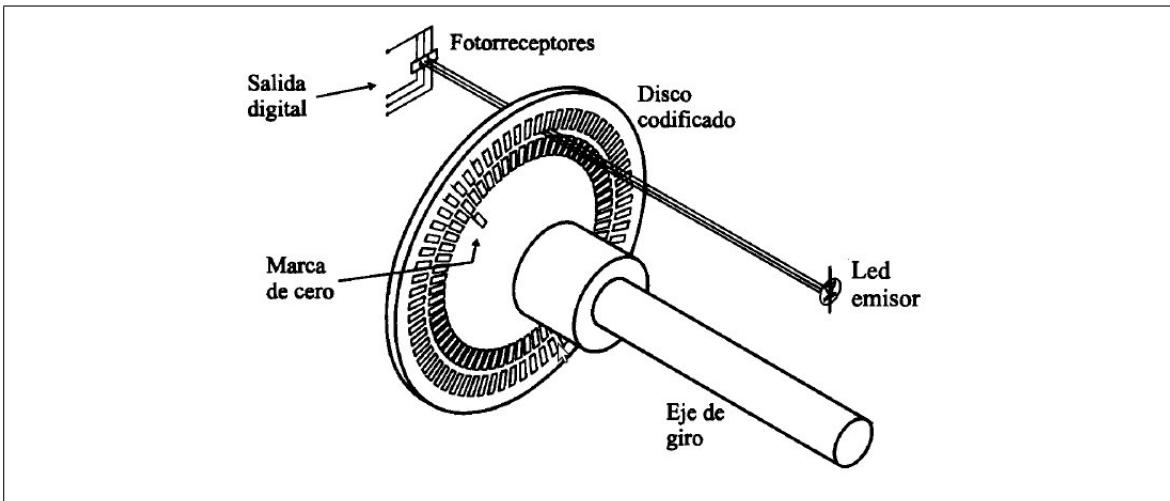


**Figura 1.4:** Representación gráfica de *encoder* de tipo lineal obtenida desde (“Incremental Linear Encoders; Enclosed Models”, 2008).

### **Encoder rotativos**

El *encoder* se conecta a través del eje del motor y mide su posición angular como se muestra en la figura 1.5. Puede llegar a una resolución superior a  $5 \mu\text{m}$ , pero a diferencia del caso lineal, a esta medición se le asocia un error de desalineamiento en el giro del eje, por lo que la resolución efectiva es dependiente de la calibración mecánica de la máquina. Su valor de mercado bordea los \$70.000 pesos por unidad. Tal como con los encoders

lineales, se requerirían tres unidades para determinar la posición de la fresa, y con un total de \$210.000 aproximadamente, el costo sigue siendo demasiado elevado.



**Figura 1.5:** Representación gráfica de *encoder* de tipo rotativo obtenida desde (Jiménez, 2009).

#### 1.4.2.2. Potenciómetros lineales

Método que utiliza el cambio en la resistencia de un potenciómetro rotativo para medir distancias. El potenciómetro se conecta al eje del motor el cual va acoplado mecánicamente a la mesa de la máquina, este tipo de medición resulta económico ya que sus valores suelen encontrarse bajo los \$10.000 y usualmente tienen una precisión del orden de 1 mm máximo (“Elap PLS/PL2S - Linear potentiometer; Potentiometer transducers”, s.f.). Todo esto hace que este tipo de sensores sean una opción viable para este tipo de aplicación.

Un defecto de esta medición es la necesidad de agregar circuitos extras de transducción y comunicación, además de toda la mecánica que habría que implementar la cual puede ser complicada e incómoda por el tamaño que ocuparía. Otra desventaja es que poseen una vida útil muy corta debido al desgaste que se produce por el roce al cambiar la intensidad de su resistencia.

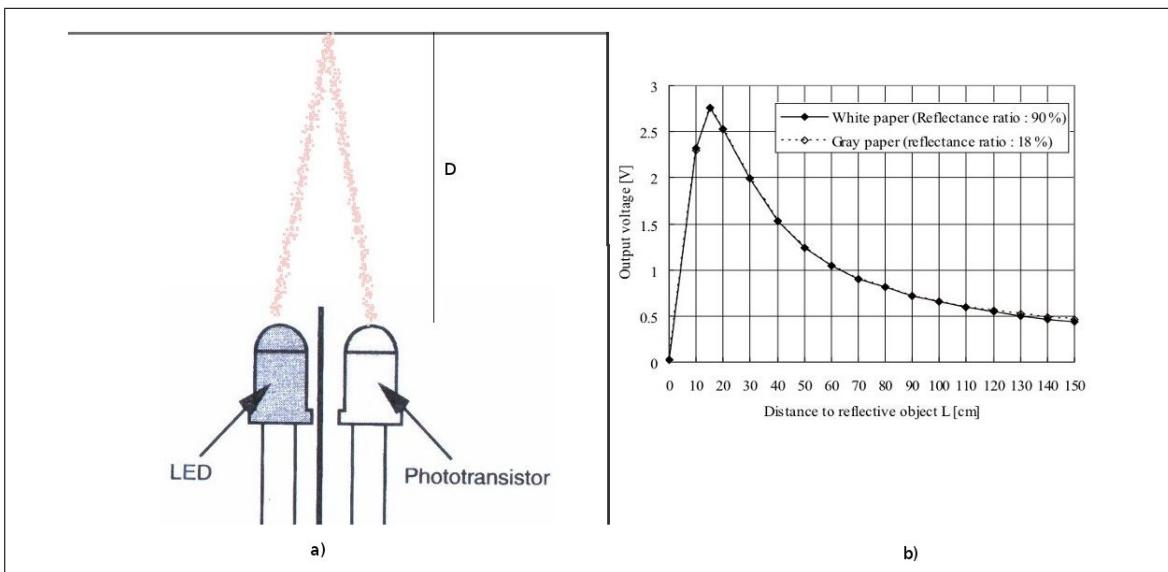
#### 1.4.2.3. Sensores de proximidad

En esta sección se detallan algunos sensores de proximidad que podrían ser útiles en la medición de distancias.

## Sensores ópticos infrarrojos

El sensor funciona a través de métodos de iluminación infrarroja (IR) y tiene una eficiencia que depende de la iluminación externa y la calidad del sensor. Además puede verse afectada por interferencias de otro tipo de señales infrarrojas que se presenten en el ambiente (Torres, 2006).

El principio de funcionamiento se basa en las propiedades de los diodos y los fototransistores, los cuales emiten y reciben luz respectivamente. La medición funciona mediante una configuración de diodo y fototransistor como se presenta en la figura 1.6 a). A través de este procedimiento se puede obtener una respuesta de voltaje que depende de la distancia a la que se encuentre el sensor y el objeto a medir. El gráfico de la figura 1.6 b) muestra la respuesta del sensor Sharp, modelo GP2Y0A02YK0F, con el cual es posible determinar la posición real del sensor *versus* el voltaje obtenido como respuesta a distintas distancias del objeto. (Paulo Malheiros y Costa, 2010).



**Figura 1.6:** a) Representación gráfica del sensor de proximidad del tipo infrarrojo, b) Distancia versus voltaje de salida del sensor Sharp, modelo GP2Y0A02YK0F obtenida desde su hoja de datos (“Distance Measuring Sensor Unit Measuring distance: 20 to 150 cm Analog output type”, 2006).

Estos sensores son de bajo costo, pero tienen la desventaja de que la resolución suele ser menor a la necesaria en aplicaciones del tipo CNC que son del orden de

los submilímetros. El error está en el rango entre [2.1, 3.5] mm (Mohammad, 2009) aproximadamente.

### **Sensores ultrasónicos**

Estos sensores se basan en principios físicos simples. Debido a que la velocidad del sonido se considera constante en el aire, se emite un pulso en una frecuencia mayor a la que el oído humano es capaz de percibir, para luego escuchar el eco y calcular el tiempo que éste demora en ir y volver utilizando la siguiente ecuación:

$$Distancia = \frac{V_{sonido}}{\left(\frac{t}{2}\right)} \quad (1.1)$$

Uno de los beneficios de este método de medición es su precisión. Pese a esto no alcanza la deseada para esta aplicación que debiese ser menor a 1 mm. Por ejemplo, con el sensor UB400-12GM-U-V1 que posee una buena resolución entre los rangos de [50, 400] mm, se llega a valores aproximados de error entre [1.8, 2.4] mm (Mohammad, 2009).

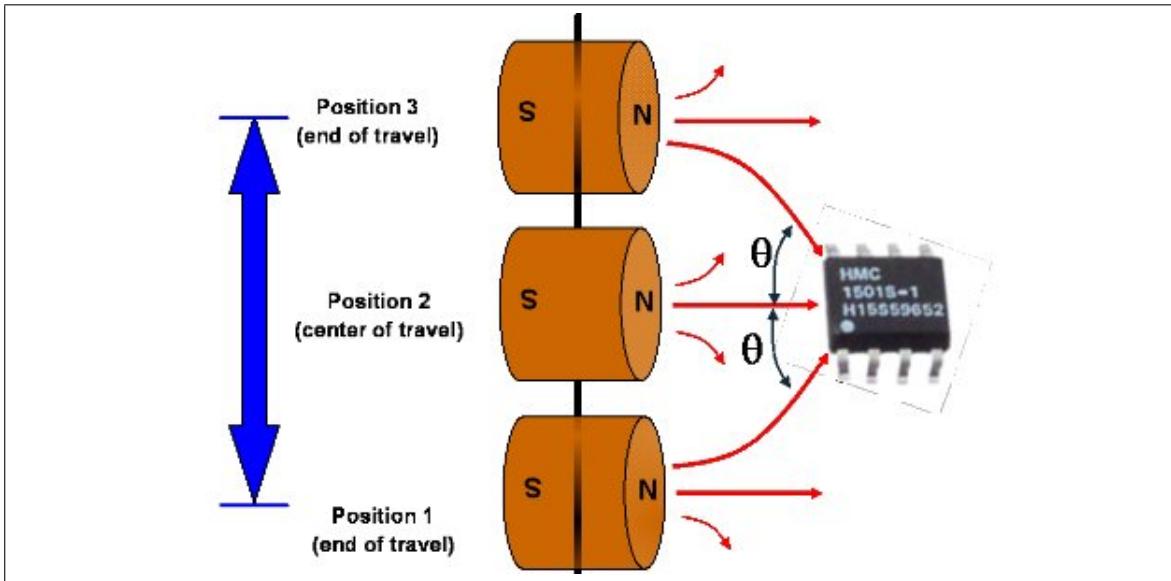
### **Sensores magneto-resistivos**

Esta medición se basa en un principio de magneto-resistencia, que se produce cuando cambia el campo magnético cercano al sensor. Utiliza el puente de Wheatstone (Hoffmann, s.f.) para medir la intensidad y dirección del campo magnético, (“Magnetic Displacement Sensors HMC1501/1512”, 2008).

Para medir una posición se necesita un objeto que produzca un campo magnético similar al de la figura 1.7. En aplicaciones que posean muchos metales cercanos, se puede producir ruido debido a los cambios no lineales que se generan en el cambio del flujo magnético que se produce en estos materiales. Además estos sensores poseen una saturación en los límites de medición.

Este sensor posee un rango de medición menor a 20 mm, lo que lo hace poco práctico para aplicaciones en CNC, en los que se desea medir cambios de posición mayores a 70 mm.

### **Sensores inductivos**



**Figura 1.7:** Representación gráfica del sensor Wheatstone HMC1501-1512 de la empresa Honeywell obtenida desde su hoja de datos.

La medición de posición de los sensores inductivos se basa en un principio similar al sensor magneto-resistivo, en el cual se utiliza una bobina con un flujo oscilante para formar un campo magnético variable cercano a la superficie metálica de detección. La inducción obtenida del campo magnético varía dependiendo de la distancia de la superficie a medir. Estas distancias suelen no sobrepasar los 30 mm para los sensores inductivos de largo alcance. Estos sensores suelen tener una precisión menor a 0, pero el rango de operación lo hace poco práctico en aplicaciones donde se desea medir distancias mayores a los 70 mm.

### Sensores de efecto capacitivo

Los capacitores consisten en dos placas metálicas paralelas separadas en una distancia  $d$  por algún material que es mal conductor de electricidad entre ellas, llamado dieléctrico. La figura 1.8 presenta una representación del capacitor, la siguiente ecuación es una caracterización de un condensador:

$$C(d) = \frac{\epsilon A}{d} = \frac{\epsilon_r \epsilon_0 A}{d} \quad (1.2)$$

con,

$\epsilon$  = Dieléctrico constante o permitividad.

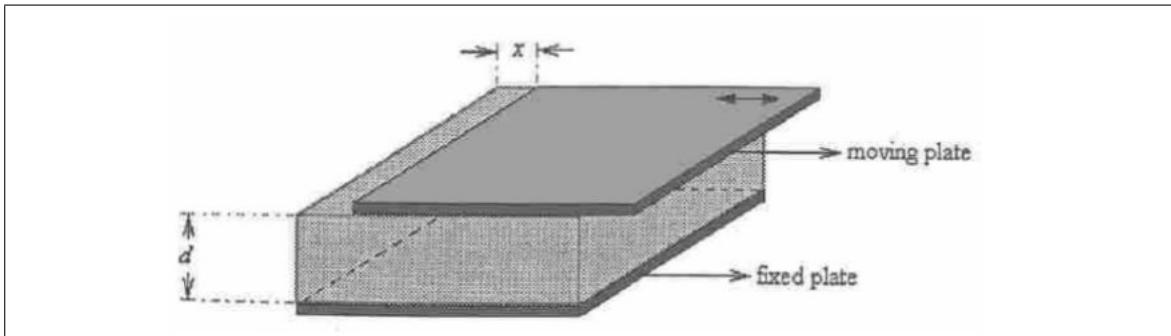
$\epsilon_r$  = Dieléctrico relativo a los materiales.

$\epsilon_0$  = Dieléctrico en el vacío.

$x$  = Distancia entre las placas.

$A$  = Área efectiva entre las placas.

Mediante un cálculo de los parámetros de éste y una estimación de la capacitancia es posible determinar con precisión la distancia  $d$  entre las placas.



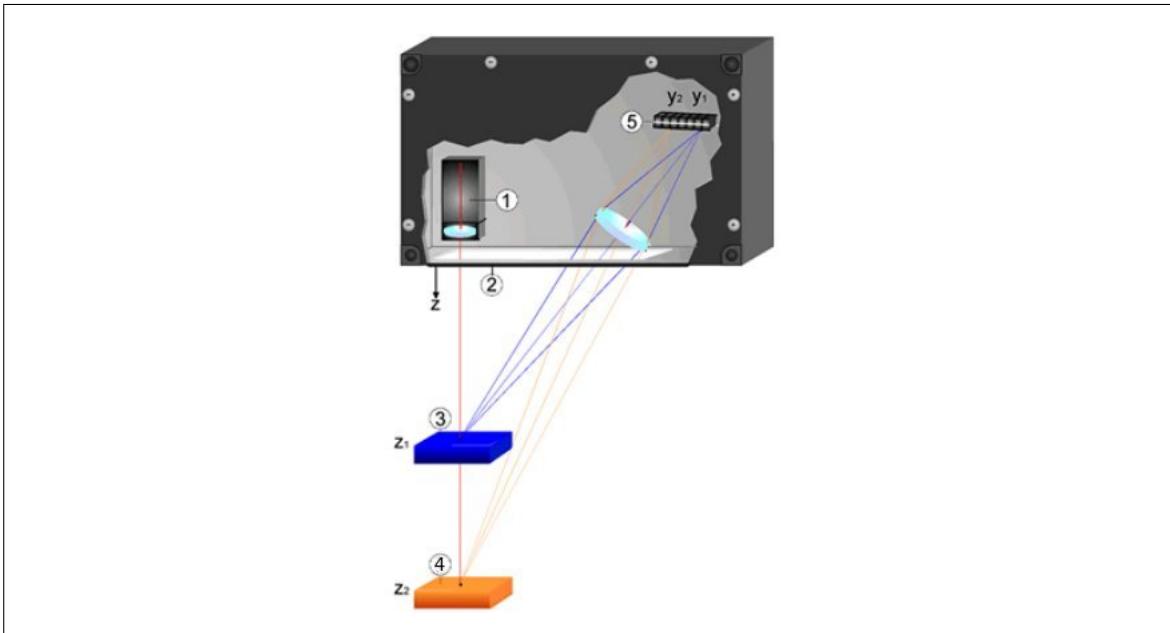
**Figura 1.8:** Representación gráfica de un condensador obtenida desde el libro de la referencia (Phillips, 2004).

Los sensores capacitivos miden distancias menores a 10 mm, tienen una resolución menor a 1  $\mu\text{m}$  y poseen un costo similar a los sensores inductivos. Esto hace que la utilización de este tipo de sistemas sea de un costo y beneficio similar a los sensores inductivos, lo que los hace poco prácticos para aplicaciones de CNC.

#### 1.4.2.4. Sensor de distancia mediante triangulación laser

El método de triangulación laser se basa en relaciones geométricas para determinar la altura de un objeto. Primero se crea un *spot* con un laser de un determinado color sobre la superficie que se desea medir. Luego la posición del reflejo de éste es medida a través de un detector llamado *Charge-Coupled Device* (CCD) que consiste en un sensor que obtiene una imagen similar a las cámaras fotográficas o por un sensor llamado *Position Sensitive Device* (PSD) que consiste en un sensor que detecta la posición de un determinado punto.

Con estos sensores se obtiene una posición relativa a la altura del objeto de forma similar a la figura 1.9. Para distintas alturas el *spot* se ve en posiciones distintas del detector.



**Figura 1.9:** Representación de triangulación láser con, 1) Laser emisor, 2) Apertura, 3) Posición 1, 4) Posición 2, 5) Lector CCD o PSD. Imagen obtenida desde la referencia (Federico, 2011).

La medición a través de este método produce buenos resultados, con estos sensores es posible obtener la posición de un objeto en diferentes rangos de distancia que puede ir entre pocos milímetros hasta varios metros y con una resolución menor a un 0,1 % (Federico, 2011). Los sensores más baratos tienen un costo por sobre los \$75.000 pesos y posee bajo rango de medición (hasta 100 mm). Los que poseen un amplio rango de medición (Hasta 100 m) tienen un costo superior a los \$500.000 pesos.

El sensor de distancia por triangulación posee una precisión alta pero un costo elevado y sobrepasa, en muchos casos, los rangos de medición deseados, por lo que no son óptimos para aplicaciones de CNC.

#### 1.4.2.5. Cámaras web

En esta memoria se plantearon dos métodos para la estimación de posición a través de cámaras *web*: uno consiste en medir la variación en grados del eje del motor (Método

indirecto) y otro en medir la posición de un punto con respecto a la cámara (Método directo).

Los métodos de estimación de posición tienen como ventaja la utilización de sólo dos cámaras *web* de bajo costo, valuadas en el mercado en aproximadamente \$10.000 pesos. Debido a la calidad del producto el precio es aceptable dentro de los estándares de economía que se desean para este proyecto. A continuación se explican ambos métodos y sus posibles ventajas y desventajas a la hora de aplicarlos.

### **Método directo**

Para el método directo se pone un punto de color situado en frente de la cámara, y mediante procesamiento de imágenes se hace una selección de color, saturación e intensidad para seleccionar los pixeles deseados y crear una imagen binaria. Luego, con esta imagen es posible calcular el centroide del punto de color y transformarlo a la posición real de la fresa. La resolución máxima de estimación de posición del método en milímetros se calcula a través de la siguiente ecuación:

$$\text{Resolución}_{\text{posición}} = \text{Resolución}_{\text{pixeles}} \frac{\text{Max}(\text{Distancia})}{\text{Max}(\text{pixeles})} \quad (1.3)$$

Se calcula la resolución máxima de este método con una máquina Clarke Metalworker modelo CMD10, un movimiento entre  $\pm 40$  mm y una cámara de 768 x 768 pixeles. Además se estima que con el cálculo de centroide es posible llegar a una resolución de 0.1 pixel, con la que se obtiene una resolución final teórica de 0.01 mm, la que es suficiente para obtener los resultados deseados. Este es un método absoluto, por lo que siempre es posible obtener la posición del punto. Una gran ventaja de esta medición es que la posición de la figura no está vinculada a errores por el tornillo sin fin ni a errores mecánicos.

### **Método indirecto**

En el método indirecto la cámara se posiciona frente a un punto que gira en el borde de un disco conectado al eje del motor que produce movimiento circular. Mediante el mismo método de procesamiento de imágenes y estimación de centroide, se determina la posición

del punto. De esta manera el movimiento queda vinculado al giro del motor y al movimiento que éste produce sobre la mesa de la máquina. A través de éste método se produce mayor movimiento del punto a medir, por una menor cantidad de movimiento sobre la mesa, es por esto que con un buen método de estimación de posición es posible obtener una mejor resolución que en el caso anterior, pero sujeta a errores de la mecánica de la fresadora y del tornillo sin fin.

Las desventajas de este método son que se necesita una cámara extra, un mayor procesamiento de la información, es necesario definir el cero de la máquina, y, además, se debe definir el tamaño de circunferencia del disco de giro con antelación. Aún así, una vez definidos estos parámetros la resolución de este método es mejor que la del método directo pero más compleja de calibrar.

#### **1.4.3. Software de control numérico por computador**

En la comunicación entre el prototipo diseñado y la máquina se utiliza un *software* de CNC. Existen una gran variedad de programas que pueden ser utilizados para distintos sistemas operativos y con distintas características. Entre los más utilizados está el Mach3 (pagado) para Windows, el LinuxCNC (gratuito) para Linux y TurboCNC (pagado) para Windows. A continuación se detallan las principales características de cada uno de ellos.

##### **1.4.3.1. Mach3**

El *software* Mach3 es uno de los programas más utilizados industrialmente porque posee buenas características de uso. Algunas de estas son:

- Controlar hasta 6 grados de libertad.
- Importar imágenes DXG, BMP, JPG directamente desde el LazyCam.
- Visualizar el código-G durante ejecución
- Generar código-G desde el LazyCam o el Wizards
- Interfaz editable, es decir, es posible agregar nuevas funcionalidades y pantallas a través de *Visual Basic*
- Control en lazo abierto.

- Interpretador de código-G.

El Mach3 sólo funciona en lazo abierto y con motores *steppers* o motores controlados de manera similar a los *steppers*. Generalmente con este tipo de motores no se generan lazos de control cerrado, ya que para poder hacerlo sería necesario crear una nueva interfaz y funcionalidad sobre el *software*, el cual sólo se puede corregir entre comandos de código-G. Posee buena documentación por parte de sus creadores.

#### **1.4.3.2. LinuxCNC**

El LinuxCNC, antes llamado EMC2, es un *software* bastante utilizado y funciona bajo el sistema operativo de Linux. Además es un *software* libre y *open source* que da libre acceso a modificar el *software* como se deseé. El EMC2 posee muchas características que lo hacen una opción viable para ser utilizado. Estas son:

- *Software* libre.
- Interfaz gráfica que simplifica su uso.
- Interpretador de código-G
- Operación de bajo nivel con el *hardware* de la máquina.
- Un *software* con lenguaje de tipo de controlador programable lógico (PLC).
- Es capaz de mover hasta 9 ejes.
- Puede operar tanto con servomotores como con motores *steppers*.

Este *software* posee la ventaja de ser capaz de manejar la máquina tanto en lazo abierto con motores *steppers*, como en lazo cerrado con motores continuos. No posee una documentación tan extendida como el Mach3, pero se puede encontrar ayuda en foros de la *web*.

#### **1.4.3.3. TurboCNC**

El *software* TurboCNC posee muy buenas capacidades de procesamiento, es pagado y posee muy buena documentación desde su sitio *web*. Sus características principales son:

- Trabaja con código-G obtenido del algún *software* CAM.
- Se comunica a través del puerto paralelo.

- Funciona en lazo abierto con motores de paso.
- Es capaz de mover hasta 8 ejes simultáneamente.

Su interfaz es compleja de usar, por lo que el Mach3 es una mejor opción.

#### **1.4.4. Motores**

En los capítulos siguientes se detallan los tipos de motores que podrían ser utilizados y sus características.

##### **1.4.4.1. Motores *Steppers***

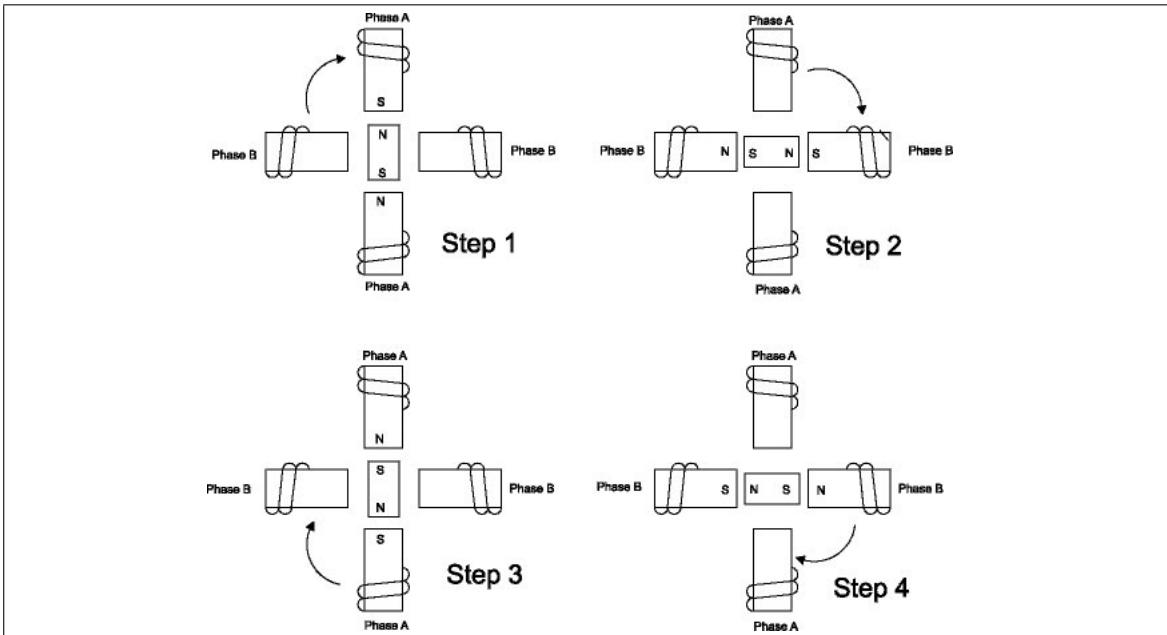
###### **Principio de funcionamiento**

El motor *stepper* funciona moviéndose en cantidades pequeñas de traslación llamadas pasos. En cada movimiento de pasos el eje gira una cantidad determinada de grados, producto de la conmutación del campo magnético en el estator. Por ejemplo, en la figura 1.10 se presenta el movimiento en cuatro pasos: el primer paso posee polaridad positiva en la fase A, en el segundo paso se produce un movimiento polarizando positivamente la fase B, el tercer paso polariza la fase A negativamente continuando el giro y en el cuarto paso se polariza la fase B negativamente para completar el giro. Para lograr el giro continuo se repiten todos los pasos iterativamente.

Estos motores poseen precisión angular fija y es posible alcanzar una cantidad pequeña de grados por paso. Por ejemplo hay motores de 200 pasos por vuelta con los cuales es posible producir un *micro-stepping* que puede dividir estos valores de pasos en 2, 4, 8, 16, (“TB6560AHQ, TB6560AHQ”, 2009) es decir, hasta 3200 pasos por vuelta, lo que se traduce en  $0.1125^\circ$  por cada paso, la desventaja de utilizar *micro-stepping*, es que los motores pueden excederse en torque y perder pasos en su giro.

###### ***Hardware* de control**

El *hardware* de control se encarga de producir la conmutación que genera el giro del motor. Es posible obtener mayor precisión con movimientos menores a un paso llamado *micro-stepping*.



**Figura 1.10:** Cuatro pasos de un motor de *stepper* simple, obtenida desde el manual (“Theory in AC motors”, 2001).

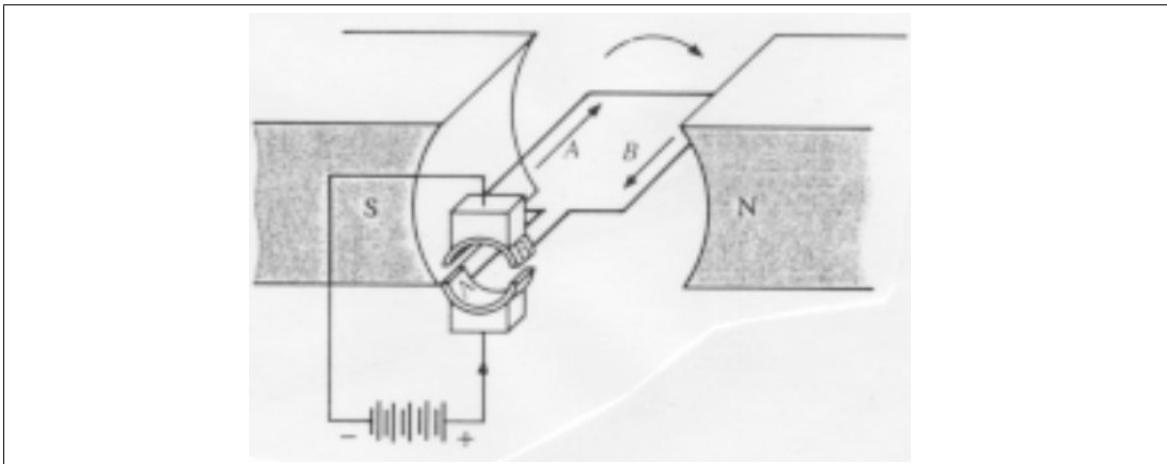
Este *hardware* controla la velocidad y el torque máximo de los motores con procesos más sencillos que para otro tipo de motores. Al poder producir una cantidad de pasos fijas es posible realizar métodos de lazo abierto para el control CNC.

#### 1.4.4.2. Motores continuos

##### Principio de funcionamiento

Los motores continuos funcionan por un campo magnético en el rotor generado por medio de corriente directa. El rotor se alimenta por escobillas conectadas a una fuente de alimentación que cambia su polarización a medida que gira el eje, la figura 1.11 representa esta configuración de escobillas. De esta manera el voltaje del rotor cambia constantemente para darle un giro al campo magnético y darle continuidad al giro. Estos motores poseen un cambio continuo y pueden tener una precisión mejor que en motores *steppers* que depende de los sensores de medición y del lazo de control (“Theory in DC motors”, 2001).

##### Hardware de control de motores de corriente continua



**Figura 1.11:** Diagrama explicativo de motores de corriente continua obtenida desde los apuntes (Beauvais, 2003).

Los motores de corriente continua se controlan mediante la intensidad de voltaje, por lo que no es necesario un *hardware* complejo. Mediante este método es difícil controlar la velocidad de los motores, la cual depende del torque necesario para moverse, el que se relaciona directamente con el material y el corte que se realiza.

#### 1.4.5. Software de CNC

La mayoría de los sistemas CNC en el mercado se basan en controladores que utilizan un lazo de control cerrado con *encoders* y motores continuos o lazos de control abierto con motores *steppers*. Los costos de estas fresadoras, que no son de marcas reconocidas, se sitúan por sobre los \$500.000 pesos para máquinas de sistemas en lazo abierto. Para máquinas de CNC que utilizan metodologías de lazo cerrado a través de encoders, los precios alcanzan los \$8.000.000 de pesos. Por ejemplo, las máquinas de la empresa Haas, con precisión de 0.01 mm, cuestan \$8.550.000 por EBay, sin considerar costo de envío. También se considera la máquina XQK9630S de Bolton *Hardware*, que cuenta una precisión de 0.02 mm para 4 grados de libertad y tiene un costo de \$5.000.000 de pesos.

## 1.5. Propuesta de desarrollo para el fresado a través del computador

Para el desarrollo de esta memoria se propone utilizar un método nuevo de medición que sea económico y pueda optimizar el proceso de fresado de la máquina CNC. Se selecciona el *software* óptimo y se desarrolla un programa nuevo.

Actualmente existe una máquina fresadora en el departamento de ingeniería eléctrica de la Pontificia Universidad Católica de Chile (PUC), pero se utiliza manualmente ya que carece de un sistema de control. El propósito de este desarrollo es automatizar la máquina de la manera más óptima posible, dada las condiciones que se poseen. Para mayores detalles revisar el capítulo 4, sección 4.1.

### 1.5.0.1. Motores y *hardware*

Para la automatización de la máquina se usaron motores *steppers* y un *hardware* de control. Las especificaciones de estos componentes se encuentran en la sección 4.1 y en los manuales al final de este documento.

### 1.5.0.2. Método de medición

El método de medición para estimar la posición es a través de cámaras *web*. Consiste en el método directo que se encuentra explicado en la sección 1.4.2.5.

#### *Software*

La etapa del *software* se separa en dos partes. La primera consiste en utilizar un programa comercial para hacer el maquinado, el cual se puede adaptar de la manera que uno desee. Además se utiliza un *software* para medir la posición de un objeto. A continuación se explica en más detalle.

#### **Software de control numérico por computador**

Como *softwares* CNC se utiliza el Mach3, cuyas características fueron descritas en la sección 1.4.3.1. El programa es editado para usar el método de medición a través de cámaras *web* y generar la corrección de error mediante un lazo cerrado.

#### **Creación del Software**

Para la medición de la posición mediante las cámaras *web* es necesario desarrollar un *software* capaz de trabajar con procesamiento de imágenes y estimaciones matemáticas de posición. Este programa debe ser capaz de comunicarse con el Mach3 para enviar las posiciones actuales que lee el *software*.

#### **1.5.0.3. Contribuciones**

En esta memoria se propone un método alternativo de medición, económico y con precisiones similares a los sistemas que se utilizan actualmente en la industria. En este desarrollo se pueden ver las posibilidades de medir posiciones mediante este método y analizar qué tan preciso puede llegar a ser. Se estudian los posibles errores y sus soluciones con la intención de optimizar la medición.

Debido a que las *webcams* son de fácil acceso y económicas, el método de estimación de posición a través de procesamiento de imágenes permite que el desarrollo de esta aplicación sea simple y utilizable en otras máquinas.

## Capítulo 2. PRINCIPIO TEORICO Y MODELAMIENTO MATEMATICO DE LA MEDICION DE POSICION Y PROCESAMIENTO CONTROL NUMERICO POR COMPUTADOR

En el capítulo presente se detallan todos los aspectos teóricos de las partes utilizadas en esta memoria, se detallará el funcionamiento de los métodos de adquisición de imágenes, modelos matemáticos para la estimación de posición y los filtros que se utilizan para disminuir los niveles de ruido de los sistemas.

### 2.1. Sensor para la obtención de imágenes

Los *Charge-Coupled Device* (CCD) son los sensores más utilizados para obtener imágenes de cualquier tipo de cámara digital.

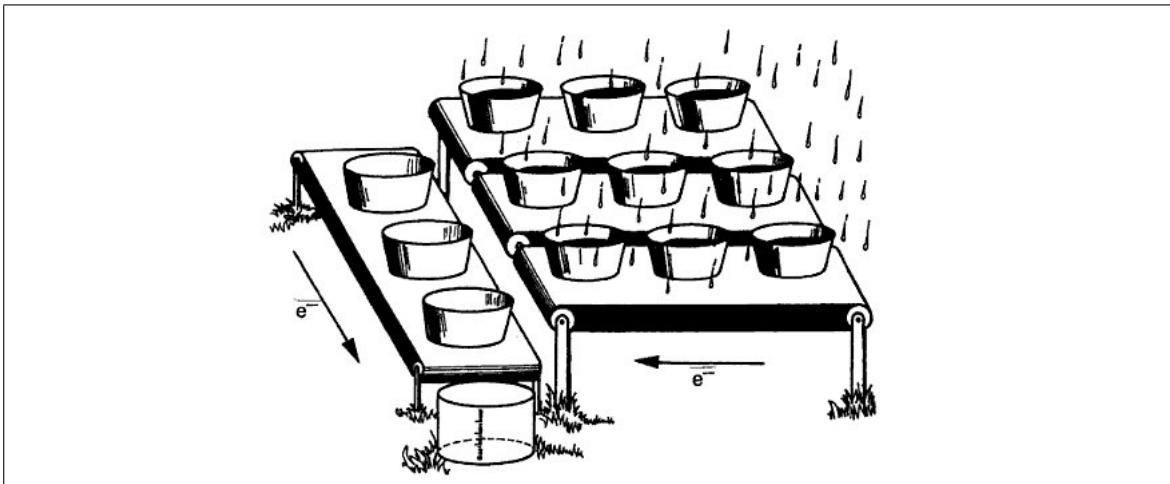
El CCD consiste en un sensor para la medición de la imagen. A continuación se presenta su método de funcionamiento y las características principales que se deben considerar para optimizar la obtención de la imagen.

#### 2.1.1. Principio de funcionamiento del sensor

El CCD consiste en un arreglo de un tamaño predeterminado, por ejemplo 800x600 pixeles, en que cada punto del arreglo (pixel), es un capacitor CMOS en un punto la imagen del mismo tamaño que el CCD que guarda la cantidad de fotones recibidos en forma de electrones. Se utiliza el principio fotoeléctrico como método de transformación de fotones a carga eléctrica (Guzmán, 2012), la que luego se envía a un computador y se crea la imagen final.

En la figura 2.1 se presenta una analogía obtenida desde el libro de Janesick (Janesick, 2007) que simplifica la comprensión del funcionamiento de estos sensores. Cada balde representa el pozo del CMOS donde se guardan los electrones. La lluvia representa a los fotones que caen sobre este sensor, y luego cuando se desea medir hasta dónde se han llenado estos baldes, se pasan fila por fila al nivel inferior y se comienza a medir el nivel de

agua (electrones) de cada balde. De esta manera se puede obtener el nivel de agua existente en cada recipiente del arreglo.



**Figura 2.1:** Analogía del principio de funcionamiento de un CCD, imagen obtenida desde el libro de Janesick (Janesick, 2007).

### 2.1.2. Principales fuentes de error

En esta sección se detallan las principales fuentes de error que afectan sobre la lectura del CCD al momento de obtener la imagen.

1 **Dark Current:** El *dark Current* es intrínseco en los semiconductores y ocurre debido a la temperatura a la que se encuentra el sensor, la que produce portadores minoritarios. La única forma de corregirlo es enfriando el sensor o disminuyendo el tiempo de exposición. La siguiente ecuación describe el ruido promedio de electrones en cada pixel por segundo.

$$D_R = CT^{1,5}e^{-\frac{E_g}{2kT}} \quad (2.1)$$

con,

$D_R$  = Dark Current promedio generada en  $\frac{e^-}{seg \cdot pixel}$ .

$C$  = Constante.

$T$  = Temperatura (K).

$E_g$  = Bandgap Silicio (eV).

$k$  = Constante de Boltzmann ( $8,62 \cdot 10^{-5} \frac{eV}{K}$ ).

De la fórmula anterior podemos ver que el error depende proporcionalmente de la temperatura, y el resultado del tiempo de exposición. El error se integra durante el tiempo de exposición del CCD, por lo que a mayor tiempo de exposición mayor es el error. Dado esto es posible disminuir el error bajando la temperatura al sensor o disminuyendo el tiempo de exposición.

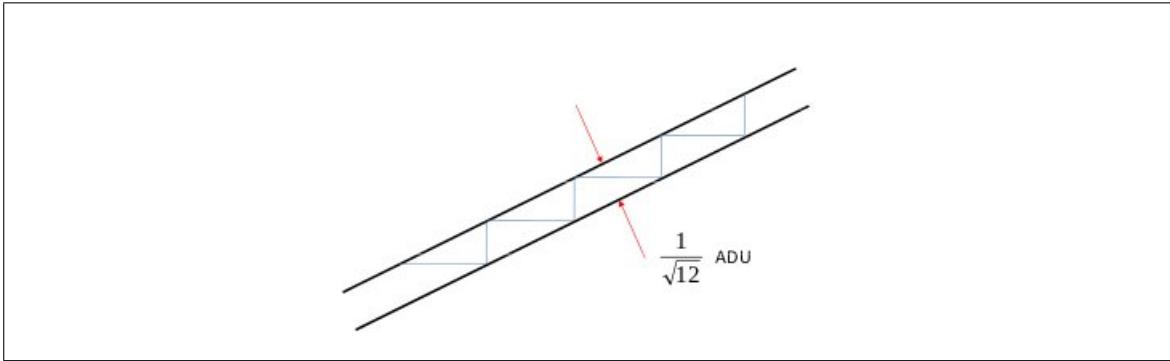
2 **Shot noise:** El *shot noise* se produce debido a fluctuaciones en los fotones detectados y se da bajo un proceso estadístico que sigue una distribución de Poisson. Este ruido es uno de los que afecta con mayor intensidad la lectura de la imagen mediante el CCD. La distribución de Poisson sigue la siguiente función de probabilidad:

$$f(k; \lambda) = Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

3 **Spurious charge:** El *Spurious Charge* se produce debido a la inversión del voltaje del *clock*, que hace que se recolecten huecos provenientes del *channel stop* en la parte inferior de la compuerta de cada pixel. Este error no es solucionable ya que se produce en la electrónica de lectura del CCD.

4 **Error de quantización:** Se produce por la transformación de la cantidad de carga análoga a un valor digital ADU. La figura 2.2 ejemplifica gráficamente este efecto.

5 **Otros:** Existen otros errores que podrían considerarse como, por ejemplo, los de interferencia y de imagen residual, entre otros. Sin embargo los más importantes son los definidos anteriormente.



**Figura 2.2:** Error de quantización, figura obtenida desde (Guzmán, 2012).

## 2.2. Estimación de centroide

La estimación de centroide es necesaria para calcular la posición de un objeto sobre una imagen. Mientras mejor sea el método más eficientes y mejores serán los resultados. Esta sección describe los algoritmos estudiados durante el desarrollo de esta memoria, se describen métodos binarios y métodos analógicos.

### 2.2.1. Método de centroide binario

Calcula el centro de masa de la imagen binarizada. El umbral de binarización debe ser ajustado para producir una segmentación correcta de la región de interés. Este método es óptimo cuando se hace muy difícil separar las intensidades precisas de cada pixel que se desea utilizar, o cuando éstas poseen variaciones de intensidad altas.

$$(x_c, y_c) = \frac{\sum_{i,j} (i, j) I_{ij} B_{ij}}{\sum_{i,j} I_{ij} B_{ij}} \quad (2.2)$$

con,

$(i, j)$  = Vector de posiciones  $(i, j)$  sobre el CCD.

$I_{ij}$  = Intensidad de luz para el pixel de la posición  $(i, j)$ .

$B_{ij}$  = Binario del filtro que es 1 si el pixel

corresponde y 0 si no para el pixel de la posición  $(i, j)$ .

### 2.2.2. Método de centroide analógico

Calcula el centro de masa de la imagen. Es el mejor de los tres, pero asume que la imagen solo posee el objeto al que se le desea calcular el centroide. Se puede llegar a posiciones menores a nivel de subpixeles en comparación a los otros dos métodos.

$$(x_c, y_c) = \frac{\sum_{i,j} (i, j) I_{ij}}{\sum_{i,j} I_{ij}} \quad (2.3)$$

con,

$(i, j)$  = Vector de posiciones  $(i, j)$  sobre el CCD.

$I_{ij}$  = Intensidad de luz para el pixel de la posición  $(i, j)$ .

### 2.2.3. Método de centroide ponderado

El método de centroide ponderado asume que se conoce la forma del objeto e intenta modelarlo para lograr medir los pixeles de mayor importancia y menor error con mayor ponderación. Tiene la complejidad de que debe conocer la forma de la figura y qué pixeles deben tener mayor ponderación (Akondi Vyas, 2009). En este caso se mide una posición de una circunferencia y el error es Gaussiano, es decir, es mayor en los bordes, por lo que se asume una distribución Gaussiana para su ponderación.

$$(x_c, y_c) = \frac{\sum_{i,j} (i, j) I_{ij} W_{ij}}{\sum_{i,j} I_{ij} W_{ij}} \quad (2.4)$$

$$W(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_c)^2}{2\sigma^2} - \frac{(y-y_c)^2}{2\sigma^2}} \quad (2.5)$$

con,

$(i, j)$  = Vector de posiciones  $(i, j)$  sobre el CCD.

$I_{ij}$  = Intensidad de luz para el pixel de la posición  $(i, j)$ .

$W_{ij}$  = Binario del filtro que es 1 si el pixel

corresponde y 0 si no para el pixel de la posición  $(i, j)$ .

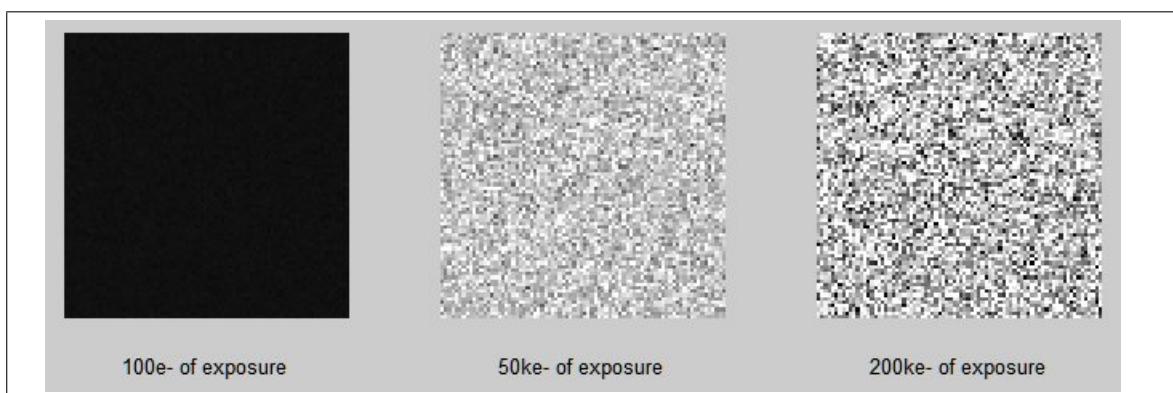
$x_c$  = Centroide anterior de x.

$y_c$  = Centroide anterior de y.

### 2.3. Procesamiento de imágenes

El procesamiento de imágenes ayuda a eliminar errores propios del CCD y aumenta la eficiencia de estimación de posición.

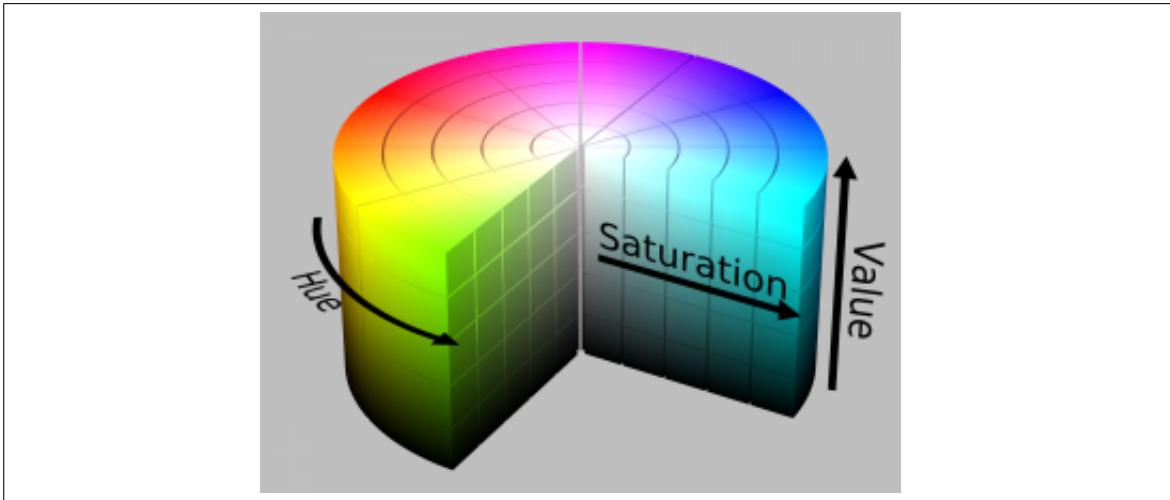
El error principal consiste en que cada pixel varía su lectura en el tiempo debido a errores internos del CCD. Por ejemplo, en la figura 2.3 se muestra una simulación de un CCD con *Shot Noise* para distintos tiempos de exposición, en la que se aprecia que cada pixel varía en intensidad por el error. A causa de este error, se producen problemas en la binarización debido a las variaciones de intensidad que se producen, esto ocurre principalmente en los pixeles del borde del objeto, ya que al no llegar luz al pixel completo podría caer fuera de los rangos de binarización.



**Figura 2.3:** Simulación de imagen con error en el CCD.

### 2.3.1. Transformación de espacio de colores

La transformación de color Hue Saturation Value (HSV) corresponde a un sistema de colores cilíndrico tal como se presenta en la figura 2.4, la cual que posee tres valores que definen el color final: *Hue* define el color, *Saturation* define la saturación y *Value* define la intensidad del color.



**Figura 2.4:** Cilindro de color HSV, imagen obtenida desde la referencia (Guerrero, 2011).

Esta transformación de colores es ideal para realizar filtros de color y separar los rangos deseados de la imagen, ya que simplifica la selección en los rangos de colores, intensidad y saturación en comparación con otras transformaciones de color.

### 2.3.2. Filtro Gaussiano

El filtro Gaussiano es conveniente para suavizar la imagen y eliminar grandes variaciones de intensidad entre píxeles. Para este caso se utiliza un filtro Gaussiano, que consiste en generar una pequeña señal Gaussiana en 2D y luego convolucionarla con la imagen que se desea suavizar. El nivel de suavidad depende del parámetro  $\sigma$  del filtro.

$$W(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2}{2\sigma^2} - \frac{y^2}{2\sigma^2}} \quad (2.6)$$

$$Img_{i+1} = Img_i * W(x, y) \quad (2.7)$$

con,

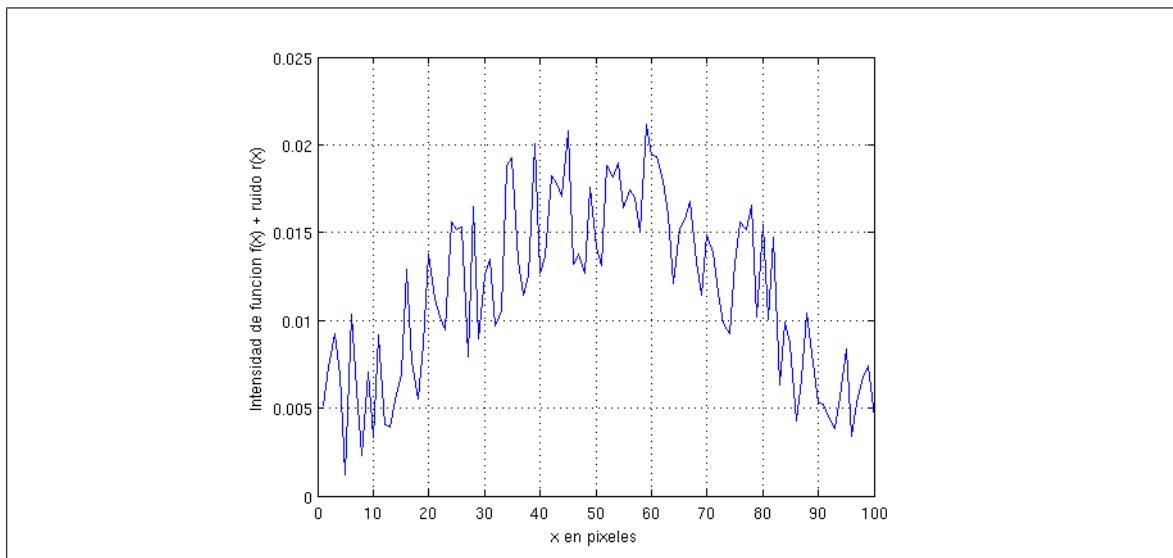
$\sigma$  = Es la desviación estándar de la señal Gaussiana.

$Img_i$  = Corresponde a la imagen original.

$Img_{i+1}$  = Corresponde a la imagen filtrada.

\* = Operador de convolución.

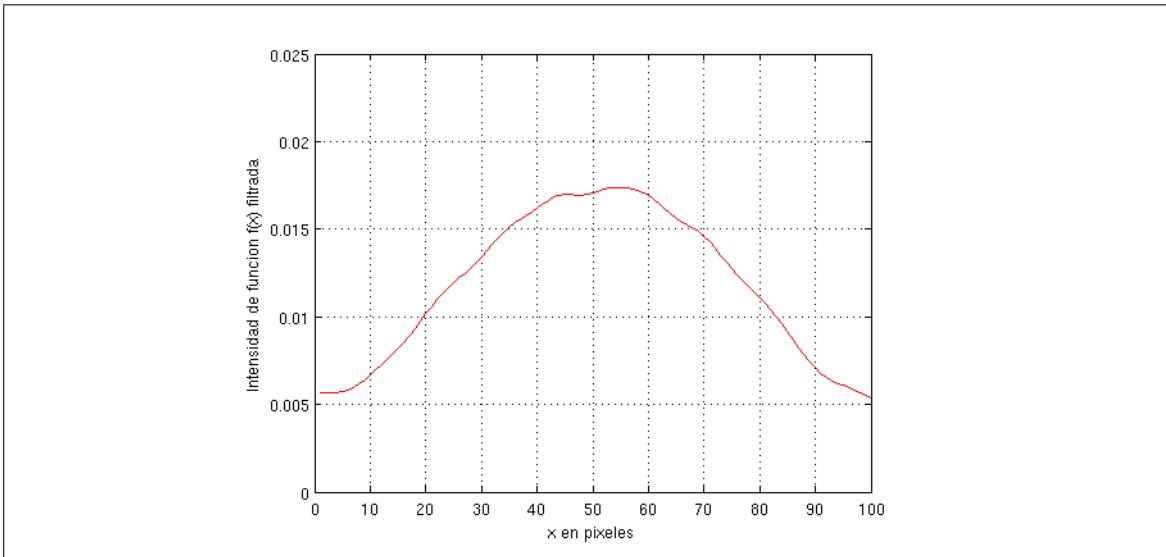
Las figuras 2.5, 2.6 y 2.7 grafican un ejemplo del filtro aplicado. Se puede apreciar que en la imagen de la figura 2.5 se muestra la señal con ruido y es suavizada al aplicar el filtro Gaussiano, de manera que se obtiene una función limpia y con mayor semejanza a la señal original. Esta semejanza se puede ver entre la señal original y la señal reconstruida en las figuras 2.7 y 2.6 respectivamente.



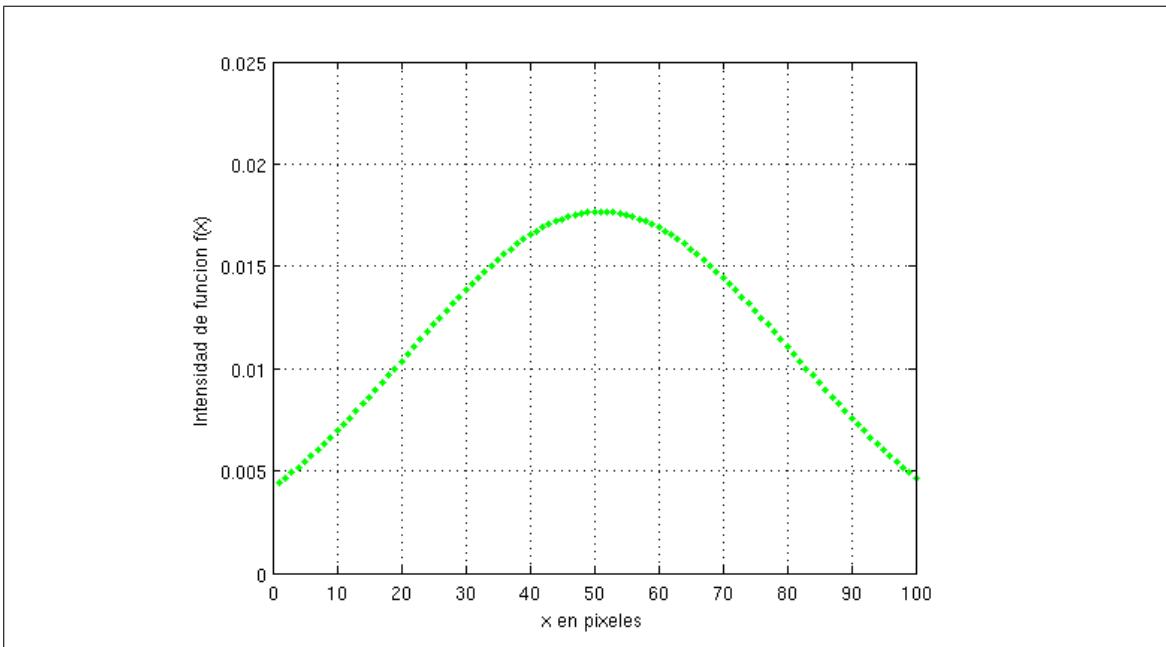
**Figura 2.5:** Señal Gaussiana con ruido.

### 2.3.3. Filtros de eliminación de *speckles*

Los *speckles* consisten en variaciones temporales sobre los pixeles de las imágenes de la *webcam*, estas variaciones se producen debido al ruido que afecta la binarización. En el caso de los filtros de colores, se producen en el borde del objeto de color filtrado y agregan error sobre la posición a medir.



**Figura 2.6:** Señal Gaussiana con ruido filtrada con filtro Gaussiano.



**Figura 2.7:** Señal Gaussiana sin ruido.

### 2.3.3.1. Filtro temporal

El filtro temporal sirve para eliminar las variaciones temporales que se producen sobre los pixeles. Consiste en ponderar un mismo pixel en el tiempo, lo que hace posible mantener los valores de los pixeles promediados.

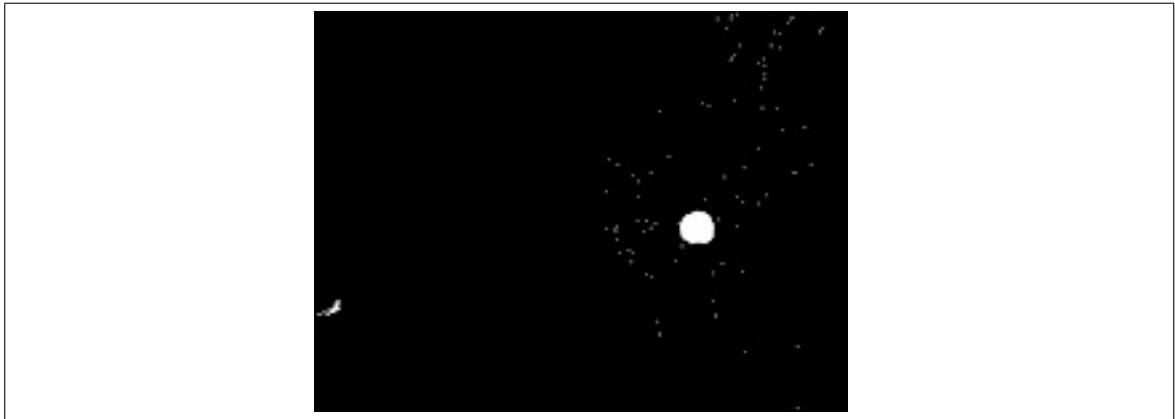
$$\begin{aligned}\overline{Pixel}_{ij} &= \frac{1}{N} \sum_{k=0}^N \lambda_k pixel_{ij,k} \\ \sum_{k=0}^N \lambda_k &= 1\end{aligned}\tag{2.8}$$

### 2.3.3.2. Filtro de reducción

El filtro de reducción consiste en quitar los píxeles del borde de un objeto de una imagen booleana. Se busca eliminar todos los píxeles sueltos y evitar que envíen la posición del centroide a los extremos de la imagen. Se compone por la función de la siguiente ecuación:

$$Img_{i,j}^{(2)} = \begin{cases} 0 & \text{si } Img_{i\pm 1,j} = 0 \text{ o } Img_{i,j\pm 1} = 0 \text{ o } Img_{i\pm 1,j\pm 1} = 0 \\ Img_{i,j} & \text{si no} \end{cases}\tag{2.9}$$

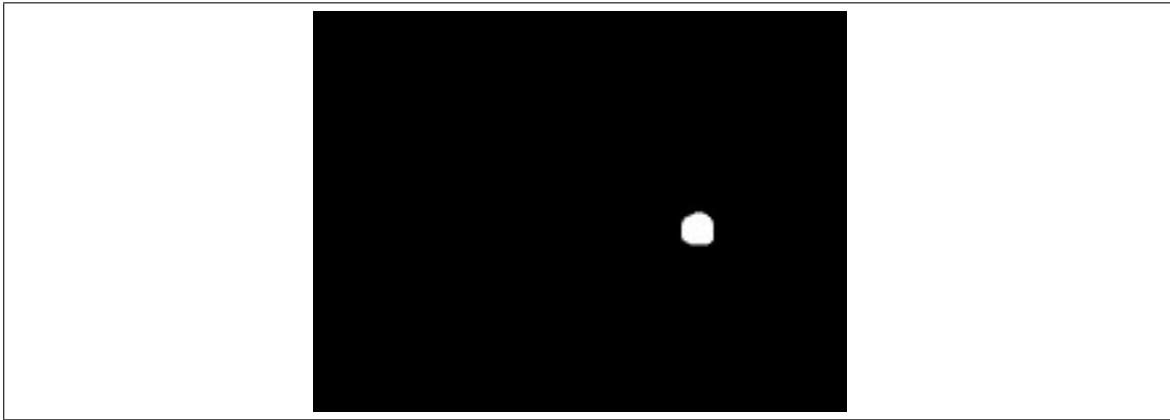
Esta ecuación se repite la cantidad de veces que se desean eliminar los píxeles en los bordes del objeto. Por ejemplo, en la figura 2.8 se muestra una imagen binarizada sin el filtro aplicado, y en la figura 2.9 con el filtro.



**Figura 2.8:** Ejemplo de imagen binarizada.

## 2.4. Distorsión y corrección del lente

La configuración geométrica de las *webcams* se puede representar a través de una cámara sin lente y con una pequeña apertura, este modelo se llama *Pinhole* y es similar

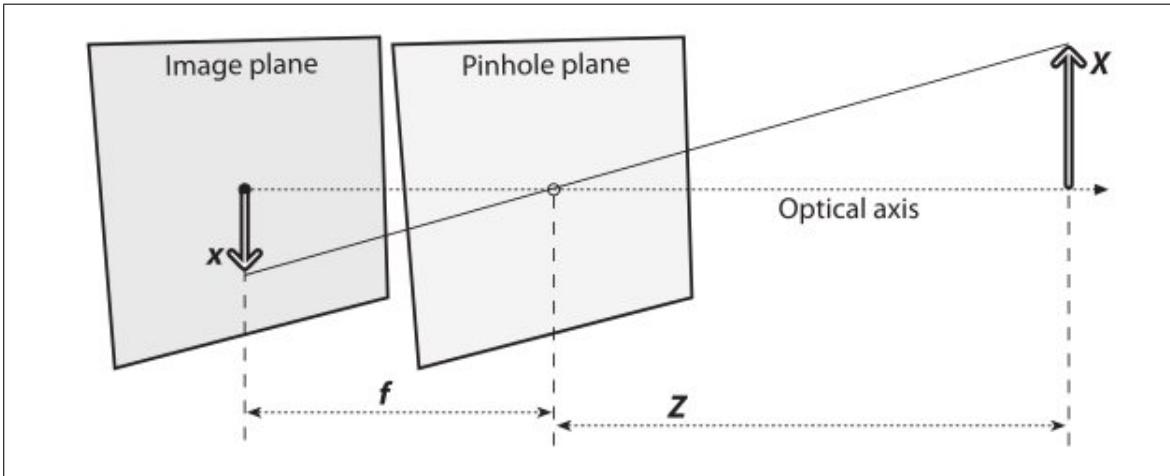


**Figura 2.9:** Ejemplo de imagen binarizada con filtro de reducción aplicado.

a la representación de la figura 2.10, en que los rayos provenientes del objeto se proyectan sobre un segundo plano correspondiente al CCD.

Se puede relacionar el objeto real con la proyección en el plano imagen del CCD (ver figura 2.10) mediante ecuaciones de triángulos semejantes centrados en el foco de la imagen.

$$-x = f \frac{X}{Z} \quad (2.10)$$



**Figura 2.10:** Cámara pinhole, imagen obtenida desde obtenida desde (Gary Bradski, 2008).

El CCD difícilmente calza con el eje de la imagen real, por lo que hay un desplazamiento en un cierto parámetro  $c_{eje}$ , por lo que la ecuación 2.10 queda de la siguiente forma:

$$x_{pantalla} = f_x \frac{X}{Z} + c_x$$

$$y_{pantalla} = f_y \frac{Y}{Z} + c_y \quad (2.11)$$

Con los parámetros de la cámara es posible realizar una representación matricial de las posiciones, de la misma manera que en la ecuación siguiente (Heikkila y Silven, 1997):

$$\begin{bmatrix} x \\ y \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2.12)$$

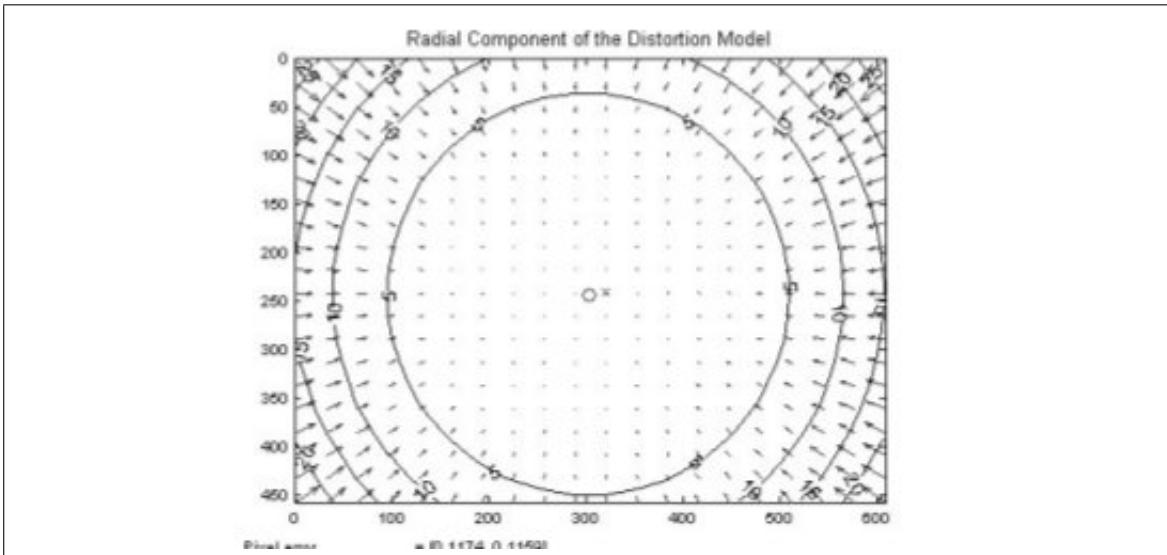
Los lentes de las cámaras producen distorsiones llamadas radial y tangencial sobre la imagen. A peor calidad de la cámara y lente, mayor son las distorsiones no deseadas. La distorsión radial consiste en una extensión de la imagen a medida que  $r$  aumenta, con  $r^2 = x^2 + y^2$ .

A mayor  $r$  la distancia entre pixeles se extiende. La figura 2.11 muestra este tipo de distorsión. La distorsión tangencial se da cuando el lente y el plano de proyección no se encuentran paralelos. La distorsión se ve cuando un objeto que está paralelo al plano de imagen se ve inclinado y se genera un punto de fuga artificial. Esto se muestra en la figura 2.12.

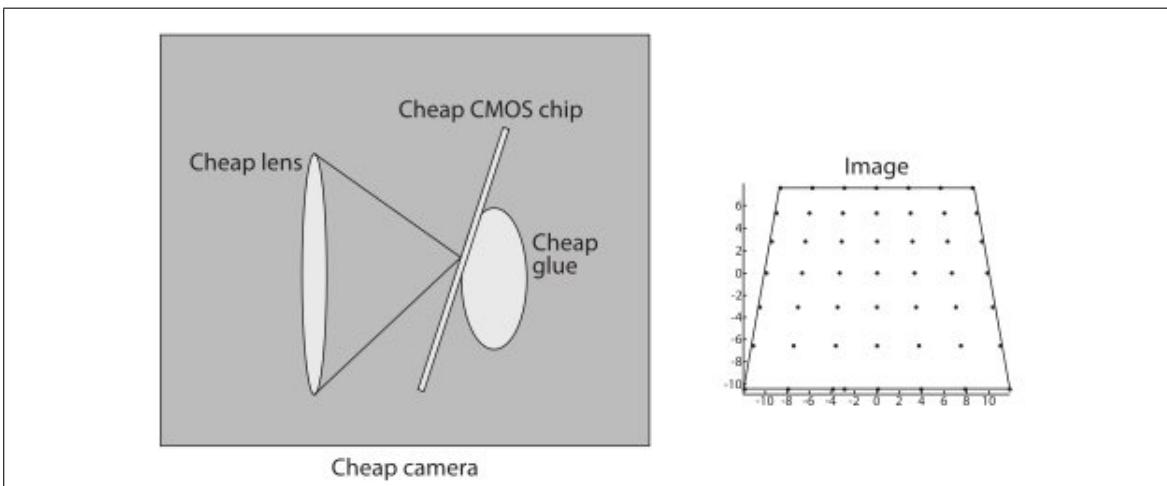
Para mejorar la imagen es necesario aplicar una corrección de ambas distorsiones. Las siguientes ecuaciones describen las expansiones de Taylor para la distorsión radial y tangencial. Los algoritmos utilizados usan los dos primeros órdenes de la expansión para la distorsión tangencial y los tres primeros para la distorsión radial.

$$x' = \frac{x}{z}$$

$$y' = \frac{y}{z}$$



**Figura 2.11:** Distorsión radial de las cámaras, imagen obtenida desde obtenida desde (Gary Bradski, 2008).



**Figura 2.12:** Distorsión tangencial de las cámaras, imagen obtenida desde obtenida desde (Gary Bradski, 2008).

$$x''_{radial} = x'(k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$y''_{radial} = y'(k_1 r^2 + k_2 r^4 + k_3 r^6)$ , Corrección radial.

$$x''_{tangencial} = 2p_1 x' y' + p_2(r^2 + 2x'^2)$$

$y''_{tangencial} = p_1(r^2 + 2y'^2) + 2p_2 x' y'$ , Corrección tangencial.

$$\begin{aligned} x'' &= x' + x''_{radial} + x''_{tangencial} \\ y'' &= y' + y''_{radial} + y''_{tangencial} \end{aligned} \quad (2.13)$$

donde  $r^2 = x'^2 + y'^2$

$$x_{imagen} = f_x \cdot x'' + c_x$$

$$y_{imagen} = f_y \cdot y'' + c_y$$

Para obtener los parámetros intrínsecos ( $f_x, f_y, c_x, c_y$ ) y extrínsecos ( $k_1, k_2, k_3, p_1, p_2$ ) de la cámara, se utilizan los algoritmos de estimación de Zhang (Zhang, 2000) en que se minimiza la función 2.14.

$$\sum_{i=1}^n \sum_{j=1}^m \|m_{ij} - \hat{m}(A, R_i, t_i, M_j)\|^2 \quad (2.14)$$

Los parámetros de la ecuación anterior se obtienen de imágenes de un objeto conocido, típicamente un tablero de ajedrez con cuadrados perfectos. Los parámetros se estiman a partir de la diferencia entre la imagen obtenida y la imagen que se debió obtener. Por ejemplo, un lente que presente distorsión radial curvará los bordes del tablero. Los parámetros  $R_i$  son de rotación,  $t_i$  de traslación y  $M_j$  de proyección de la imagen i. Las siguientes ecuaciones son una representación de la ecuación 2.12 y explican los parámetros de la ecuación 2.14.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

con,

$$m = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \text{ y } s \text{ un escalar de transformación.}$$

$r_{ij}$  y  $t_i$  corresponden a los parámetros de rotación y traslación.

## 2.5. Transformación de ejes de coordenadas

### 2.5.1. Proyección del plano real

Luego de la ecuación 2.10 se obtiene la relación entre ambos planos. El signo negativo se da debido a que ambos se encuentran en sentido opuesto del eje central.

Dada la dificultad de calcular  $f$  y  $Z$  es posible reordenar la ecuación 2.10 y obtener la siguiente relación:

$$-\frac{x}{X} = \frac{f}{Z} \quad (2.15)$$

De esta manera se puede estimar el parámetro  $\frac{f}{Z}$  de la forma  $\frac{f}{Z} = \lambda = \frac{\Delta x}{\Delta X}$ , donde  $\lambda$  corresponde al valor que define la relación entre milímetros y pixeles. Este parámetro es llamado *plate scale*.

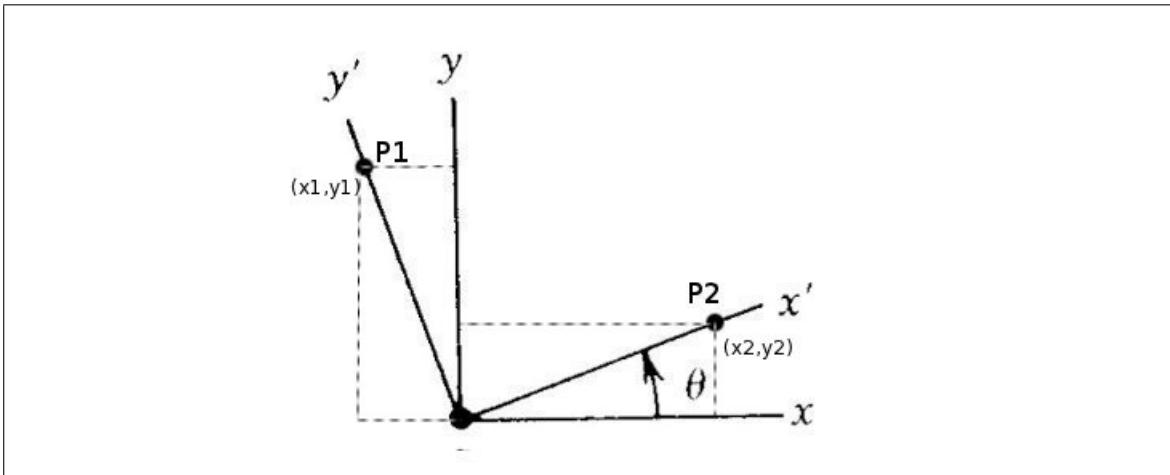
### 2.5.2. Corrección de otros ejes

Para la medición de posición se utilizan dos sistemas de coordenadas: el sistema de la máquina ( $x' - y'$ ) y el sistema propio de las cámaras ( $x - y$ ). Tal como se muestra en la figura 2.13, se pueden obtener directamente los resultados de posicionamiento de la cámara, pero es necesario realizar una transformación para calibrar los ejes de la cámara con los ejes de la mesa. Para esta transformación se utiliza una regla que consiste en que por cada movimiento en  $y'$  se genera un movimiento en  $x'$  equivalente a  $y' \sin(\theta)$ , y por cada  $x'$  se produce un movimiento en  $y'$  de  $-x' \sin(\theta)$ . Esta transformación se representa en las ecuaciones 2.16.

$$\begin{aligned} y &= y'k_1 - x'k_2 \\ x &= x'k_1 + y'k_2 \end{aligned} \quad (2.16)$$

Dado un ángulo constante se tiene que,

$$k_1 = \cos(\theta) \text{ y } k_2 = \sin(\theta)$$



**Figura 2.13:** Diferencias en los ejes de coordenadas de la cámara y la mesa.

## 2.6. Análisis del error

En esta sección se detallan los errores en la medición de la posición y en la máquina CNC. Una manera de estimar la desviación estándar del ruido de una distribución cualquiera, es la medida del cálculo de la varianza muestral con la siguiente ecuación:

$$\sigma^2 = \frac{1}{n_{\text{población}}} \sum_{i:\text{población}} (x_{\text{medido}}^i - \bar{x})^2 \quad (2.17)$$

Los errores que se describen en esta sección corresponden a los errores que se presentan en el sistema CNC desarrollado en esta memoria.

### 2.6.1. Error del lente

Dado que el lente posee dos tipos de errores (radial y tangencial), la medición de posición tiene un error desconocido dependiente de  $r^2$  y de  $(x, y)$ , el cual está dado por las siguientes ecuaciones:

$$\begin{aligned}
x_d &= x_u + \underbrace{(x_u - x_c) \left( \sum_{i=1}^n (k_i r^{2i}) \right)}_{\text{Error radial}} \\
&\quad + \underbrace{\left( P_1(r^2 + 2(x_u - x_c)^2) + P_2(x_u - x_c)(y_u - y_c) \right) (1 + \sum_{i=3}^n P_i r^i)}_{\text{Error tangencial}} \\
y_d &= y_u + \underbrace{(y_u - y_c) \left( \sum_{i=1}^n (k_i r^{2i}) \right)}_{\text{Error radial}} \\
&\quad + \underbrace{\left( P_1(r^2 + 2(y_u - y_c)^2) + P_2(x_u - x_c)(y_u - y_c) \right) (1 + \sum_{i=3}^n P_i r^i)}_{\text{Error tangencial}}
\end{aligned} \tag{2.18}$$

con  $r^2 = (x_u - x_c)^2 + (y_u - y_c)^2$

$(x_d, y_d)$  = Puntos de la imagen distorsionada

$(x_u, y_u)$  = Puntos de la imagen sin distorsión

$(x_c, y_c)$  = Puntos centrales de la imagen distorsionada

$K_m$  = Coeficiente de distorsión radial

$P_n$  = Coeficiente de distorsión tangencial

### 2.6.2. Error de lectura del CCD y estimación de centroide

Debido al *Dark Current* del CCD se obtiene un error de lectura proporcional a la ecuación 2.1. Dado el error se producen cambios en la intensidad de los colores y saturación, por lo que se produce una mala selección de píxeles en la binarización. El error se describe de la siguiente manera:

$$\begin{aligned}
(\hat{x}_c, \hat{y}_c) &= \frac{\sum_{i,j} X_{ij} I_{ij}}{\sum_{i,j} I_{ij}} - \overbrace{\frac{\sum_{i,j: \text{No encontrados}} X_{ij} I_{ij}}{\sum_{i,j} I_{ij}}}^{\text{Error con varianza } \sigma_{\text{medición}}^2} \\
(\hat{x}_c, \hat{y}_c) &= \frac{\sum_{i,j} X_{ij} I_{ij}}{\sum_{i,j} I_{ij}} - V(0, \sigma_{\text{medición}}^2)
\end{aligned}$$

Con  $V(0, \sigma_{\text{medición}}^2)$  El error dependiente de  $\sigma_{\text{medición}}^2(D_R)$ .

La estimación de estos errores se hace de manera empírica utilizando la ecuación 2.17.

### 2.6.3. Error de corrección

El cálculo del ángulo  $\theta$  de rotación de los ejes de la máquina (Sección 2.5.2), es difícil de estimar de manera exacta, por lo que se genera un error  $\Delta$  de rotación en la estimación de los ángulos. Este error se produce de manera que la ecuación 2.16 queda de la forma de la ecuación 2.19, y dado un ángulo constante con error en su estimación, es posible obtener una ecuación del error mediante las fórmulas 2.20.

$$\begin{aligned}
\cos(\theta + \Delta) &= \cos(\theta) \cos(\Delta) + \sin(\theta) \sin(\Delta) \\
\sin(\theta + \Delta) &= \sin(\theta) \cos(\Delta) - \sin(\Delta) \cos(\theta)
\end{aligned}$$

Con  $\Delta \rightarrow 0$

$$\begin{aligned}
&\overbrace{\cos(\theta) \cos(\Delta)}^{k_1} + \overbrace{\sin(\theta) \sin(\Delta)}^{\Delta} = k_1 + \Delta_1 \\
&\lim_{\Delta \rightarrow 0} \left( \cos(\theta) \underbrace{\cos(\Delta)}_1 + \sin(\theta) \underbrace{\sin(\Delta)}_\Delta \right) = k_1 + \Delta_1 \\
&\overbrace{\sin(\theta) \cos(\Delta)}^{k_2} - \overbrace{\sin(\Delta) \cos(\theta)}^{-\Delta_2} = k_2 + \Delta_2
\end{aligned} \tag{2.19}$$

Luego

$$y = y'(k_1 + \Delta_1) - x'(k_2 + \Delta_2)$$

$$x = x'(k_1 + \Delta_1) + y'(k_2 + \Delta_2)$$

Finalmente se obtienen los errores ( $e_x(x, y)$ ,  $e_y(x, y)$ ) dependientes de la posición (x, y)

$$\begin{aligned} y &= y'(k_1) - x'(k_2) + \overbrace{(y'\Delta_1 - x'\Delta_2)}^{e_y(x,y)} \\ x &= x'(k_1) + y'(k_2) + \overbrace{(x'\Delta_1 + y'\Delta_2)}^{e_x(x,y)} \end{aligned} \quad (2.20)$$

#### 2.6.4. Error mecánico

El error mecánico es propio de la máquina, debido a malos calces entre los tornillos y los ejes de la máquina, entre otros. Este error se produce dentro de un rango  $[0, Max_{error}]$  con las vibraciones de la máquina.

## Capítulo 3. SIMULACIONES Y ANALISIS

Se simulan los métodos de medición de centroide para el método directo de la sección 1.4.2.5. Se realizan distintas pruebas: con y sin ruido de sensor, método de estimación de centroide binario y analógico, pruebas con y sin filtros de imágenes. Bajo estas pruebas es estima cual es el mejor método para el cálculo de posición de la fresa.

### 3.1. Simulación de CCD

Para realizar una simulación que se asemeje a la realidad, es necesario considerar el ruido que se produce en el CCD, estas consideraciones se presentan a continuación:

- 1** *Shot noise*: se considera *shot noise* con una distribución de Poisson (Guzmán, 2012) (sección 2.1.2).
- 2** *Quantum Efficiency* (QE): eficiencia cuántica de transformación de fotones en electrones de 0.9
- 3** Tamaño: tamaño de CCD fijado por el usuario.
- 4** *Output Responsivity* (OR): transformación de carga eléctrica en el pixel en voltaje, su valor es de  $4.5 \cdot 10^{-6} \frac{\mu V}{electron}$ .
- 5** Un ADC de 16 bits (Sección 2.1.2).
- 6** Una cantidad de fotones variable siguiendo la forma de la figura a representar en el CCD.
- 7** Características similares a las del CCD CCD39-01.

Los resultados obtenidos varían principalmente por el tiempo de exposición y la cantidad de fotones que llegan al CCD.

### 3.2. Simulación de filtro Gaussiano

El filtro Gaussiano permite limpiar la imagen de la manera que se describe en la sección 2.3. Se prueba mediante una simulación que el filtro es capaz de mejorar el resultado de posicionamiento del sistema. Se realizó un cálculo de centroide binario para un círculo de

diámetro de un 10 % del tamaño total y los resultados obtenidos se presentan en la siguiente tabla, con los campos Real, Señal con ruido y Señal filtrada, correspondientes a la posición real, la posición estimada de la señal con ruido y la posición estimada de la señal filtrada, respectivamente.

Las figuras 2.7, 2.5 y 2.6 muestran resultados con una Gaussiana sin ruido, con ruido y con ruido mas filtro aplicado respectivamente, y los campos  $\sigma_{ruido}$ ,  $\sigma_{Gaussiana}$  corresponden a los valores del ruido y del tamaño de la Gaussiana del filtro.

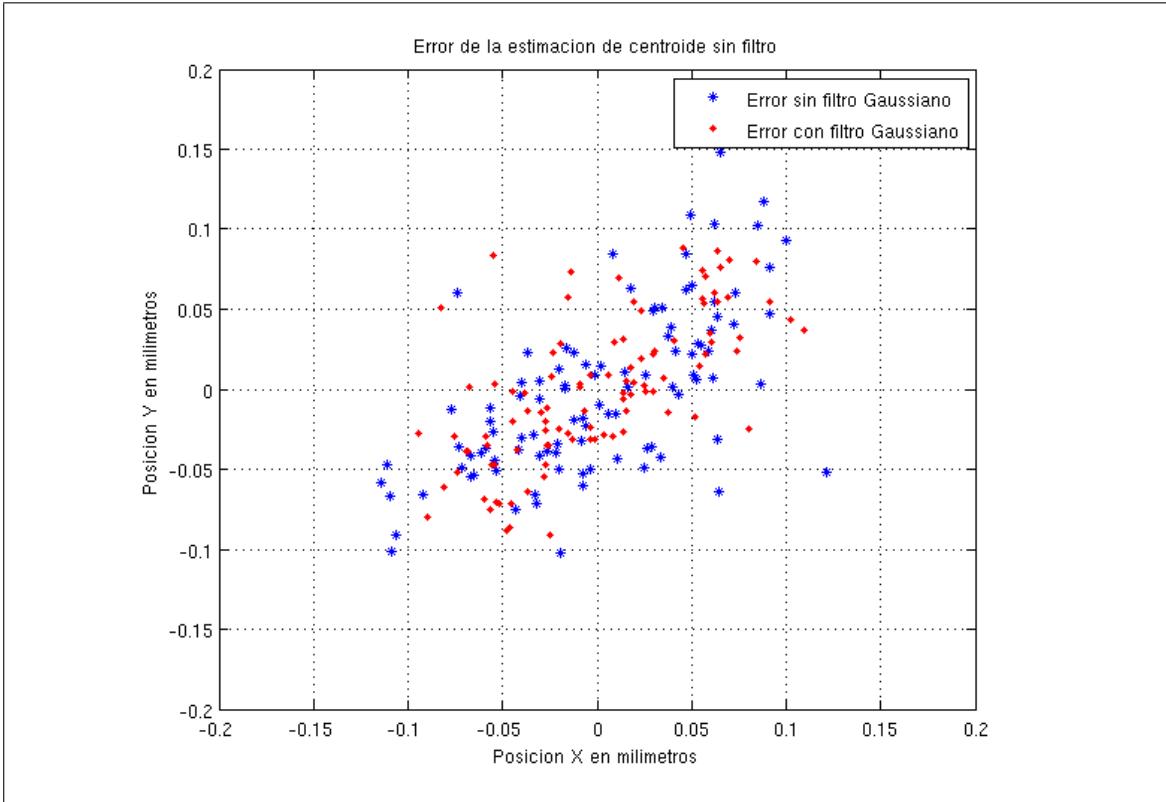
$\sigma_{ruido}$	$\sigma_{Gaussiana}$	Posición Centroide		
		Real (mm)	Señal con ruido (mm)	Señal filtrada (mm)
3	5	51	50.318	51.09
2	5	51	50.7	50.97
2	3	51	51.6	51.08
3	3	51	52.184	51.245

TABLA 3.1. Resultados de simulación del filtro Gaussiano

Los valores  $\sigma_{ruido}$  y  $\sigma_{Gaussiano}$  están definidos en pixeles y las posiciones del centroide están en pixeles relativas a un sensor de tamaño 101x101 pixeles. Cabe señalar que en la tabla 3.1 se obtienen mejores resultados de posición del centroide para el caso de la señal filtrada con el filtro Gaussiano, y que los valores del  $\sigma_{Gaussiano}$  afectan notablemente en el resultado final. De esta manera hay que colocar especial atención en la variación del  $\sigma_{Gaussiano}$  para obtener resultados óptimos. En la figura 3.1 se presenta una comparación de la posición obtenida en la simulación sin filtro Gaussiano y con el filtro aplicado. A simple vista no es notable la mejora, pero al calcular la suma del error según la siguiente ecuación:

$$\sum_{i=0}^N |Error_{X_i}| + |Error_{Y_i}|$$

Se obtiene que el error para la estimación de centroide con la imagen filtrada es más precisa que la sin filtrar: 7.8219 mm contra 8.7368 mm, respectivamente.

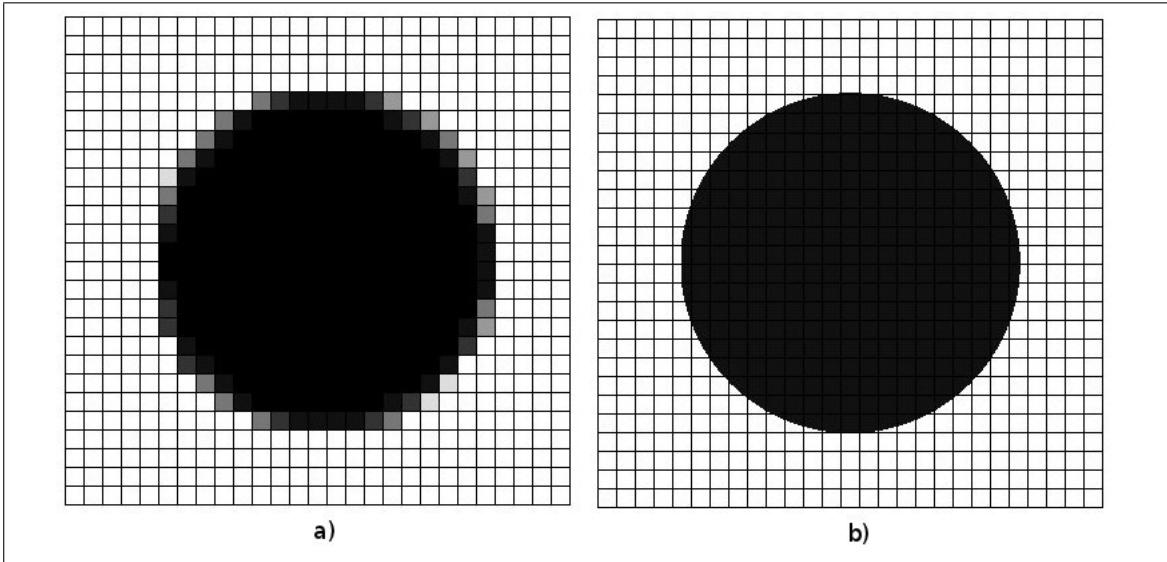


**Figura 3.1:** Comparación de error de la imagen sin filtrar e imagen filtrada.

### 3.3. Simulación de cálculo de posición

Para la simulación del cálculo de posición se ponen a prueba los algoritmos de estimación de centroide bajo varios supuestos, tales como: sistema sin ruido, sistema con ruido y sistema con ruido y con filtros de imágenes aplicados. De esta manera se puede comparar el funcionamiento de cada método de medición.

Con el fin de probar a qué niveles de exactitud es posible llegar con la medición del centroide a nivel de subpixel, se hace una simulación de la siguiente manera: primero se toma una imagen que esté levemente pixelada ( $N \times M$ , con  $N$  y  $M$  muy grandes), figura 3.2 b) y se promedian todos los cuadros que entran dentro de un pixel, lo que queda similar a la figura 3.2 a). Luego se agrega el ruido de lectura del CCD y los filtros respectivos para simular de la mejor manera posible la ubicación de las cámaras *web* del laboratorio. Finalmente se utilizan los métodos de centroide y se estima la posición de la imagen real y la imagen sobre el CCD, con la finalidad de realizar la comparación entre ambas.



**Figura 3.2:** Pixelación de imagen real, a)Imagen pixelada, b) Imagen real.

En el anexo F se presentan los resultados de ruido de estas simulaciones en mayor detalle.

### 3.3.1. Simulación sin ruido

Para la simulación sin error se realiza una prueba sin considerar el ruido del CCD ni la aplicación del filtro Gaussiano. Con esta simulación y utilizando el método binario se logra un error en el cálculo de posición de  $\pm 0.04$  pixeles en el peor caso. Utilizando el método análogo se obtiene un error de  $\pm 0.0004$  pixeles en el peor caso, pero no es un resultado aplicable ya que existen objetos externos al objeto medido de la imagen que agregan ruido en la medición de posición.

### 3.3.2. Simulación con ruido

La simulación con ruido se realizó de manera similar al caso anterior, pero se agregó ruido con distribución de *Poisson* en la lectura del CCD. Es una simulación más real que el caso anterior, lo que permite obtener resultados más representativos que en el caso anterior. Utilizando el método se obtiene un error de  $\pm 0.1$  pixeles, por lo que se puede concluir que el resultado empeora al comenzar a agregar el ruido de lectura.

### **3.3.3. Simulación con ruido más filtro Gaussiano**

Para esta simulación se mantuvieron las mismas características que en la anterior, pero se agrega un filtro Gaussiano para mejorar la precisión. A través de este método se logra una precisión de alrededor de un 10 % menor que en el caso anterior, a partir de lo cual se puede concluir que este filtro sí sirve para mejorar los resultados en la estimación, tal como se presenta en la sección 2.3.2.

### **3.3.4. Resumen de simulaciones**

Mediante las simulaciones se logra estimar que el resultado óptimo bajo condiciones similares a las reales, se obtuvo un nivel de precisión de 0.1 pixeles. Se pudo determinar que es posible llegar a una estimación 0.01 mm con las condiciones dispuestas en la sección 1.4.2.5.

Estos resultados fueron obtenidos bajo los algoritmos de estimación de centroide analógico y binario. Se concluye que algoritmo binario y con filtro gaussiano es óptimo ya que posee una estimación de posicionamiento de más del doble de velocidad de cálculo y con niveles de precisión similares en ambos métodos de estimación (binario y analógico). En el caso del método analógico puro, tal como se muestra en el 4º cuadro de las imágenes en los resultados del Anexo F, los resultados son excelentes, pero su obtención es compleja porque es necesario aislar completamente el objeto de la imagen.

## Capítulo 4. IMPLEMENTACION Y METODOLOGIA DE PRUEBAS

En este capítulo se describe, la implementación del sistema de medición de posición a través de cámaras *web* y la metodología que se utiliza para evaluar el método CNC aquí propuesto. Se presentan los componentes utilizados y sus especificaciones técnicas.

### 4.1. Especificaciones Técnicas

En esta sección se presentan las especificaciones básicas de la máquina y los componentes utilizados. Se entregan detalles específicos respecto de todos los componentes como *datasheets* de *hardware* y manual de uso en los anexos de esta memoria.

#### 4.1.1. Dimensiones y especificaciones de la máquina

La máquina corresponde a una fresadora Clarke Metalworker modelo CMD10 que posee las especificaciones que se presentan en la tabla 4.1.

Máquina Clarke MetalWorker	
Model	CMD10
Part N°	7610850
Potencia	150 Watts
Corriente	5 A
Alto sin carga	100-2000 RPM
Bajo sin carga	100-1000 RPM
Movimiento de la mesa	90 mm
Peso	32 Kg
Tamaño de la mesa	240x145 mm

TABLA 4.1. Especificaciones de la máquina.

Las corrientes, cargas y potencias son respecto al motor del taladro de la máquina y el resto de las especificaciones se refieren al dimensionamiento de ésta. Además, la

máquina cuenta con seis *limit switches*, uno por cada eje, con el fin de marcar los límites de movimiento y que no se sobrepasen ni dañen las piezas.

La máquina tiene un límite de movilidad de 13 cm en el eje X, 12 cm en el eje Y y 2 cm para el eje Z. Todos estos son valores máximos. Además el eje Z posee otro grado de libertad, se controla manualmente y genera una movilidad mayor a 10cm lo que permite escoger rangos de operación en el eje Z.

#### **4.1.2. Tornillos sin fin y reducciones**

Los tornillos poseen un movimiento de  $80 \frac{\text{ticks}}{\text{vuelta}}$ , y se mueve una distancia de  $0,025 \text{ mm}$  por cada tick en los ejes X e Y. Teniendo en consideración que el motor posee 200 pasos, se tiene una resolución máxima de  $\frac{0,025 \cdot 80}{200} \text{ mm}$ , es decir,  $0,01 \text{ mm}$ . El eje Z posee  $40 \frac{\text{ticks}}{\text{vuelta}}$  y se mueve  $0,05 \text{ mm}$  por tick, por lo que se obtiene  $\frac{0,05 \cdot 40}{200} \text{ mm}$ , es decir, una resolución máxima de  $0,01 \text{ mm}$ .

#### **4.1.3. Hardware de control de motores**

Para la alimentación y control de los motores *steppers* se utiliza un *hardware* basado en el integrado TB6560, que es capaz de controlar hasta cuatro motores por separado.

El *hardware* permite controlar los motores de paso a través del computador utilizando el puerto paralelo del ordenador. Además, permite que se puedan manejar los motores mediante el *software* CNC Mach3, el cual comanda la cantidad de pasos, velocidad y sentido de giro de cada motor.

Las hojas de especificaciones del *hardware* y del integrado que utiliza se encuentran en los anexos B y C.

#### **4.1.4. Especificaciones del computador**

Los requerimientos mínimos *hardware* y *software* son los siguientes:

- *Windows XP* (actualizado con *Service Pack 3*) con *DirectX 9.0c*, y *Visual Studio*.
- Procesador 2.6 GHz *Pentium IV* o equivalente AMD Athlon.

- 2 GB de espacio libre en el disco duro.
- 1 GB RAM en XP,
- El sistema operativo puede variar pero será necesario volver a compilar el código del *software* Fress.
- Dos cámaras *web* con la posibilidad de calibrar sus parámetros internos de exposición y *White balance*.

#### **4.1.5. Cámaras web**

El sistema cuenta con dos cámaras *Logitech c110* que sirven para medir las posiciones. Si se desean cambiar las cámaras, se debe tener en consideración que el requisito es que posean una resolución mayor o igual a  $1024 \times 768$  pixeles y que el *driver* permita controlar los parámetros de *White Balance* y *Exposition*. Las cámaras no deben estar conectadas mecánicamente de ninguna manera con la máquina, ya que las vibraciones de esta producen cambios que desvían las cámaras, descalibrando el sistema.

#### **4.1.6. Motores**

La máquina cuenta con tres motores *steppers Hybridstepping motor Model 160-010-00200*. En el cuadro 4.2 se presentan las especificaciones de estos motores. Cada motor va conectado a cada eje de la máquina para mover las posiciones X, Y, Z. En el caso que el torque supere el *holding torque*, los motores perderán pasos y a su vez agregarán error al movimiento.

Se anexan las hojas de datos de los motores en el anexo A.

### **4.2. Implementación de *software***

Para lograr el objetivo de utilizar un *feedback* entre la ejecución del código-G y el sensor de posicionamiento, es necesario crear una nueva pantalla del Mach3 (figura 4.1) y un nuevo *software* que determine la posición. Por último tiene que haber una comunicación entre ambos programas para poder hacer la relación de movimiento de los motores en lazo abierto con el sensor de posicionamiento.

Motores	
Angulo por paso	1,8°
Part N°	4,5 V
Voltaje	2,5 $\frac{A}{Fase}$
Resistencia	2 $\frac{\Omega}{Fase}$
Inductancia	3,6 $\frac{mH}{Fase}$
Holding torque	180 N · cm
Largo de motor	76 mm
Inercia de motor	440 g · cm <sup>2</sup>
Peso de motor	1,1 kg
Insulation class	B

TABLA 4.2. Especificaciones de los motores.

El código implementado en Visual Basic sobre el Mach3 para la interfaz gráfica, está incluido en el CD anexado, ya que son más de 50 páginas.

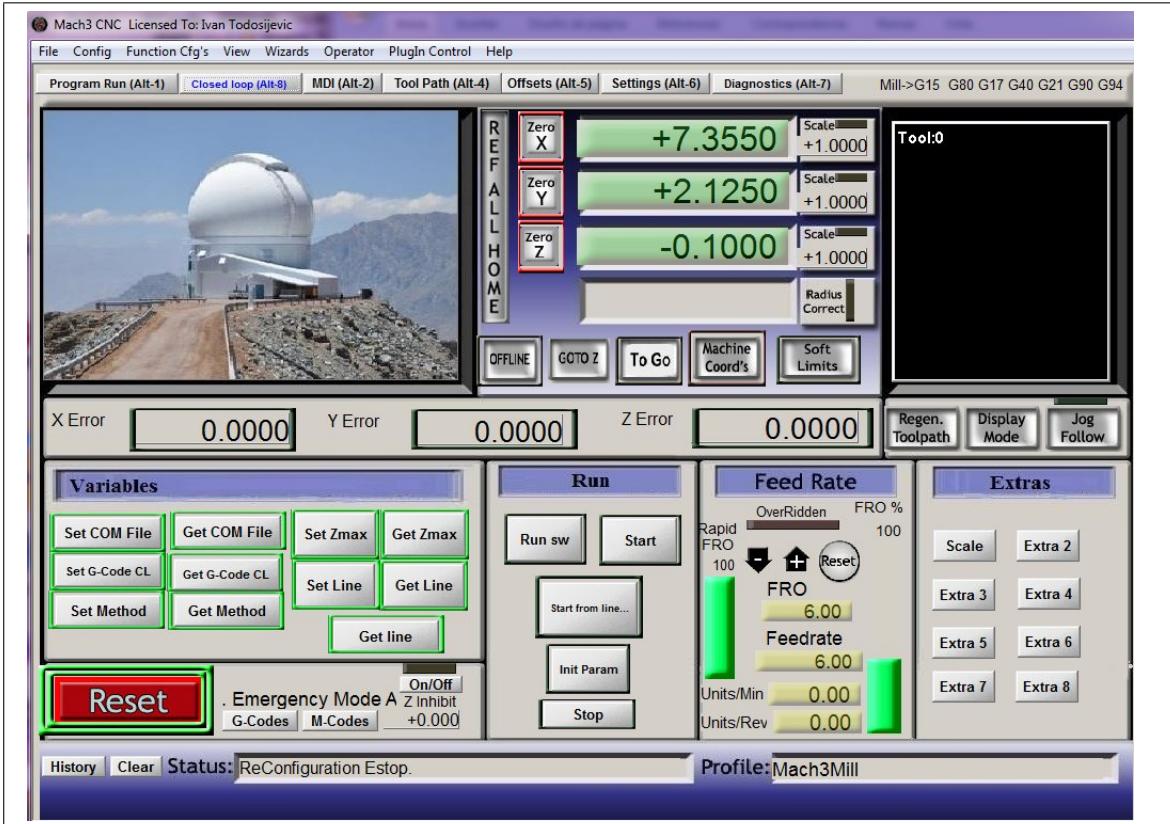
#### 4.2.1. Pantalla *closed loop* en Mach3

##### 4.2.1.1. Extensión del software Mach3

La posibilidad de editar el Mach3 permitió crear una nueva pantalla con nuevas funcionalidades, la que se llamó *closed loop*.

En esta sección se entregan detalles sobre las funciones realizadas en *Visual Basic*. Primero se generalizan con respecto a las funciones *Set* y *Get*, ya que simplemente guardan y obtienen parámetros. Las funciones *Set* tienen una estructura en la que NN es el nombre de la variable que se quiere asignar y las funciones *Get* sirven para obtener los parámetros de NN.

```
functionSetNN(Line) {
    setNN to Line
}
```



**Figura 4.1:** Pantalla *Closed Loop* del Mach3.

```
Var functionGetNN(Line) {
    return NN
}
```

La función *Run sw* queda de la siguiente manera:

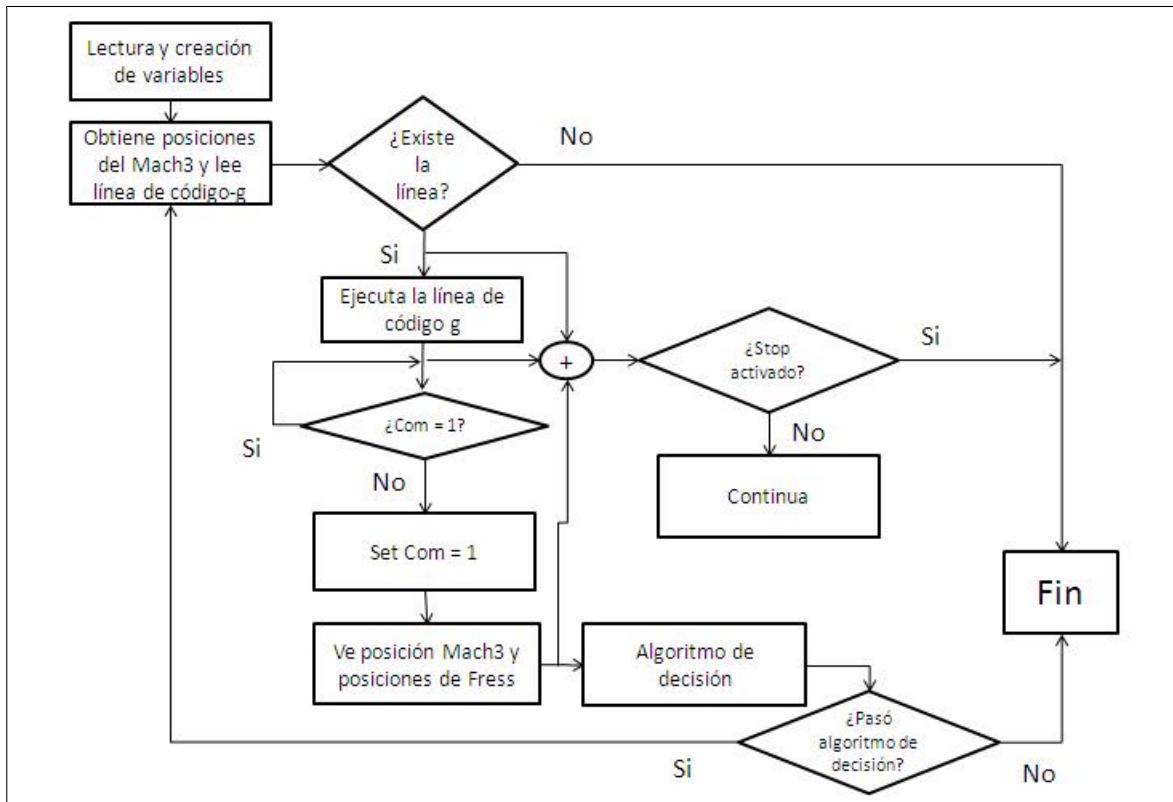
```
Void function{
    Execute Fress
}
```

Por simplicidad las funciones, *Start*, *Start from line...* y, *Scale*, se detallan a través de diagramas de bloques.

El diagrama de la figura 4.2 representa la función que posee el botón *Start*. Comienza en el bloque Lectura y creación de variable donde se leen y crean todas las variables necesarias para la ejecución del *software*. Luego continua el orden de los siguientes

bloques, en que comienza a leer el código-G y a ejecutarlo línea por línea comunicándose con el *software* Fress. De esta manera es posible hacer una comparación de las posiciones leídas por la cámara y las posiciones que posee el Mach3 para estimar el error. Después entra al bloque de algoritmo de decisión del diagrama de la figura 4.3, que decide qué algoritmo usar en caso de que el error sea mayor que un cierto valor de delta inicializado en el código.

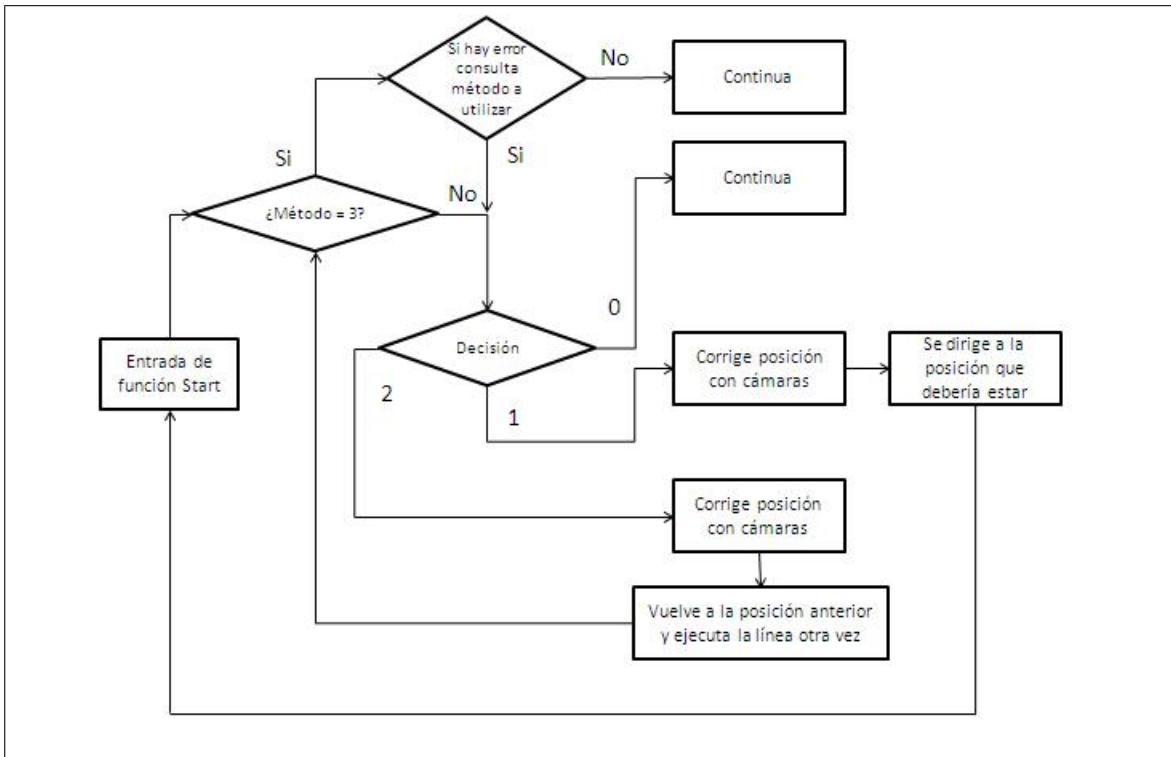
Finalmente se detiene si sucede alguna de las siguientes situaciones: se acaban las líneas del código-G, se presiona *Stop* o no pasa el algoritmo de decisión. El algoritmo de decisión está explicado más abajo.



**Figura 4.2:** Diagrama de bloques de función *Start*.

El diagrama de la figura 4.3 detalla el algoritmo de decisión. Parte desde el bloque que consulta si el método utilizado es 3 o no. De ser así se le consulta al usuario qué método desea utilizar (0, 1 o 2). Si el método es 0 el sistema continua sin corregir errores y cuando el error sobrepase un cierto  $\Delta$  de error permitido pregunta si se desea continuar. Si el método

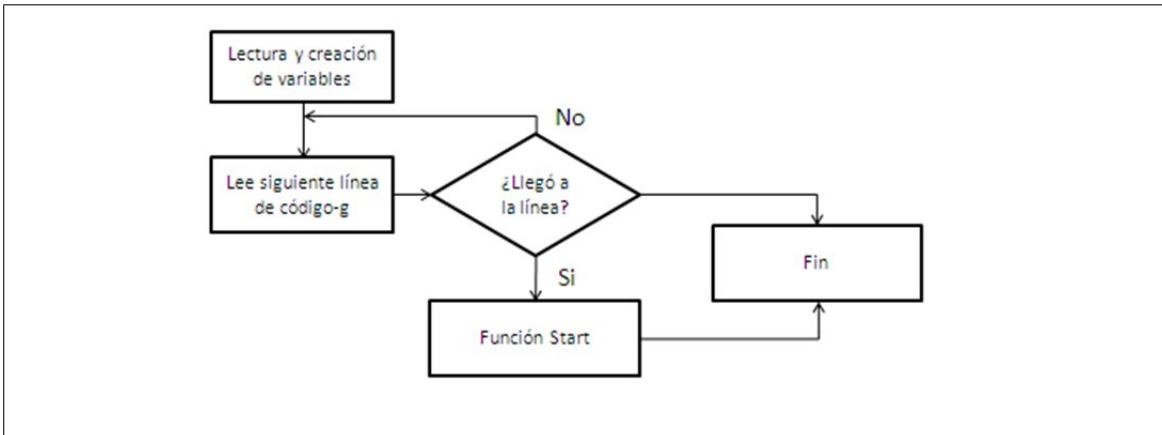
es 1, el sistema se autocorrege de acuerdo a las posiciones de lectura del *software Fress* y si el método es 2, el sistema vuelve a la posición anterior y ejecuta la línea nuevamente de acuerdo a las posiciones medidas por Fress.



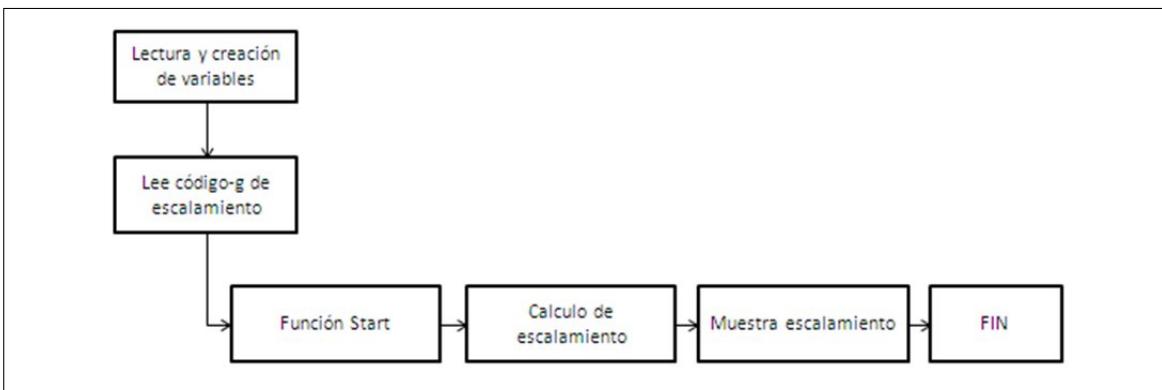
**Figura 4.3:** Diagrama de bloques del algoritmo de decisión.

El diagrama de la figura 4.4 muestra la función del botón *Start from line*. Este método comienza leyendo el código-G e itera hasta llegar a la posición de la línea deseada para luego comenzar a ejecutar la misma función que posee el botón *Start*.

Por último, el diagrama de la figura 4.5 presenta el algoritmo del botón *Scale*. Este algoritmo consiste en ejecutar la función *Start* utilizando un código-G para estimar ciertas posiciones desde el Mach3 y el *software Fress*. De esta manera se puede calcular un escalamiento entre la medición de ambas posiciones, es decir,  $\text{Posición}_{\text{Mach3}} = \beta \cdot \text{Posición}_{\text{Fress}}$  con  $\beta$  como el escalamiento. Si este valor es correcto debe ser semejante a 1. En caso contrario significa que hay problemas en la configuración del Mach3 o en los parámetros de calibración.



**Figura 4.4:** Diagrama de bloques con función *Start from line*.



**Figura 4.5:** Diagrama de bloques con función *Scale*.

#### 4.2.1.2. Explicación de funcionalidades de Mach3

Las nuevas funcionalidades del Mach3 están diseñadas especialmente para funcionar en conjunto con el *software* Fress. Si el *software* Fress no se encuentra en funcionamiento, este componente de Mach3 no se ejecutará debido a la falta de comunicación entre ambos programas. Esta aplicación se muestra en la figura 4.1, aquí se presentan todos los componentes disponibles.

Todas las mediciones y correcciones del error de posicionamiento se hacen al final de la ejecución de cada línea de código-G, ya que no es posible editar el Mach3 mientras el movimiento se esté ejecutando. Todos los botones *Get* son para obtener el valor deseado y los *Set* son para dar un valor a los parámetros.

**COM File:** es el archivo de comunicación que se usa entre Mach3 y el *software* Fress.

**G-Code CL:** es la ruta del código-G a ejecutar.

**Method:** es el método de control que se utiliza.

- 0- No utiliza ningún método de control, pero está constantemente revisando si el error es mayor a un  $\Delta$ . Si el error es mayor a  $\Delta$ , el *software* le pregunta al usuario si desea seguir con el fresado.
- 1- Utiliza un método de control directo, es decir, si posee un error mayor a un  $\Delta$  la fresadora se re-direcciona automáticamente a la posición final de la línea del código.
- 2- Utiliza un método de control indirecto que consiste en que si la posición final posee un error mayor a  $\Delta$ , la fresa vuelve a la posición anterior a la detección del error y comienza a ejecutar los comandos de código-G nuevamente. Esto se realiza direccionaldo el eje Z a un máximo de altura definido en el Mach3 y moviendo las ubicaciones X, Y a las posiciones anteriores. Luego baja el eje Z a la posición anterior y se comienzan a ejecutar los comandos de código-G desde la línea anterior.
- 3- Si existe un error mayor a un  $\Delta$ , el *software* le consulta al usuario que método utilizar.

**Zmax:** la altura máxima para volver a la posición anterior mediante el método 3.

**Line:** la línea se ejecuta desde *Start from line*.

**Run SW:** ejecuta el *software* Fress.

**Start:** comienza a ejecutar el código-G.

**Start from line:** comienza a ejecutar el código-G desde la línea ingresada en *Set line*.

**Init Param:** inicializa parámetros estándar.

**Stop:** detiene la máquina después de haber ejecutado la línea de código. Si se desea detener instantáneamente es necesario presionar el botón *reset* o el botón de emergencia que se encuentra en la máquina misma.

**Scale:** es una prueba de calibración y configuración de la máquina. Se revisa la relación de mm de la máquina con los mm del sensor de las cámaras *web*. Es decir, si el *software* Fress dice 1 mm, se verifica que efectivamente sea 1 mm en la máquina. Los resultados se entregan como la relación  $\frac{\text{mm Mach3}}{\text{mm Fress}}$ .

**X Error, Y Error, Z Error:** entrega el error obtenido entre el Mach3 y el *software* Fress después de la ejecución de cada línea de código-G.

#### 4.2.2. *Software* Fress

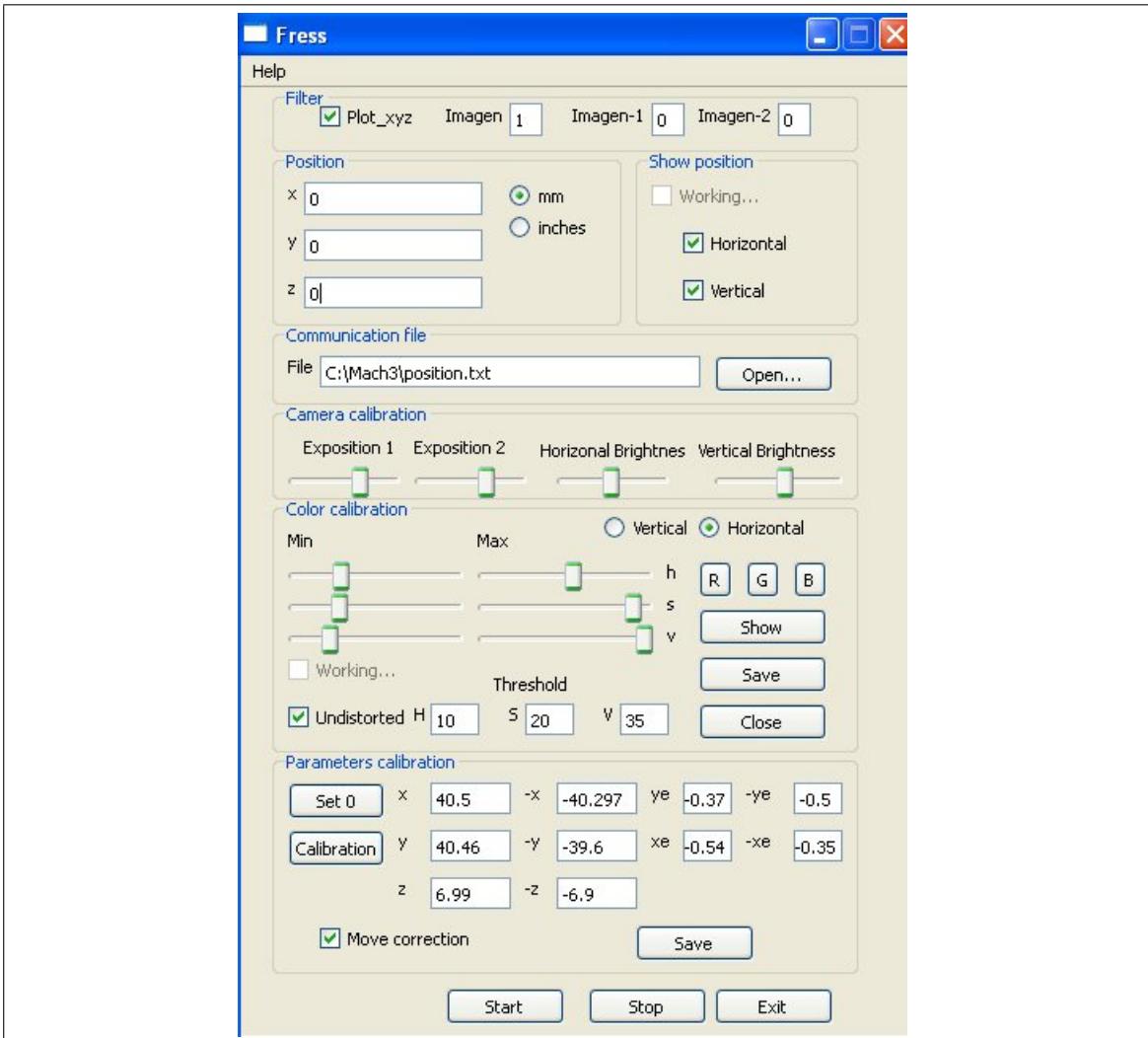
##### 4.2.2.1. Desarrollo de Fress

El *software* Fress consiste en un sensor que mide la posición (X, Y, Z) de la fresadora y la envía al Mach3 para que estime los posibles errores de la máquina. Este desarrollo esta implementado en C/C++ a través de *Visual Studio* en *Windows XP*. Como librerías de procesamiento de imágenes se utiliza *OpenCV*; como Graphical User Interface (GUI) se utilizan las librerías QT4, que poseen un buen comportamiento en el procesamiento paralelo; y para controlar las cámaras *web* se utilizan las librerías DirectX9 de Windows. El código de Fress se encuentra disponible en el CD anexado en esta memoria.

Para medir las distancias se utiliza un método que discrimina por colores. Para un rango de colores HUE se genera una imagen binaria a la que finalmente se le calcula el centroide para obtener la posición en pixeles del objeto de color. Una vez realizado este proceso, es necesario pasar a la etapa de transformación de medidas de pixeles a milímetros, para finalmente obtener la posición absoluta de la máquina en un determinado sistema referencial y en una unidad conocida (milímetros).

La figura 4.6 muestra una pantalla del *software* Fress con todos sus botones y funcionalidades.

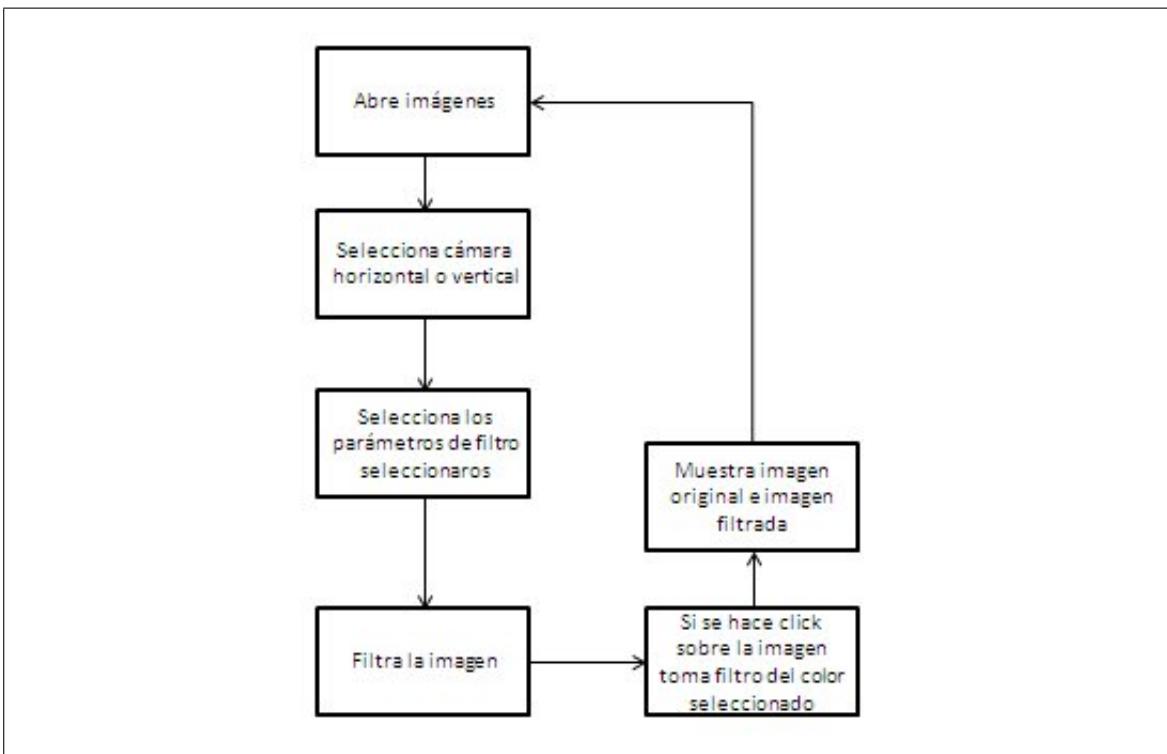
El diagrama de bloques de las principales funciones del *software* Fress se presentan en las figuras 4.7 y 4.8, éstas son: la función *Show* permite la calibración del filtro de colores y la función *Start* realiza la estimación de las posiciones y se comunica con el Mach3. El botón *Show* comienza por el bloque "Abre imágenes" que presenta las figuras de las



**Figura 4.6:** Software Fress.

cámaras *web* en la pantalla del computador. Luego se seleccionan los parámetros de filtro de colores, ya sea por valores fijados manualmente o haciendo *click* sobre el color deseado. Finalmente se filtran los colores no deseados de la figura y se muestran las nuevas imágenes con estos filtros aplicados. El botón *Start* comienza por el bloque "Abre imágenes actuales de las *webcams*". Luego se presentan las figuras de las cámaras *web* en la pantalla y se corrigen las distorsiones radiales y tangenciales del lente sobre la imagen. Se filtran los colores no deseados de la imagen con los parámetros estimados a través del botón *Show*, se aplican filtros de eliminación de ruido sobre la imagen, se estima la posición del centroide

y se realiza la comunicación con el Mach3. Finalmente se devuelve al bloque "Obtiene imagen de las cámaras" para seguir repitiendo estos bloques hasta que se presione *Stop*.

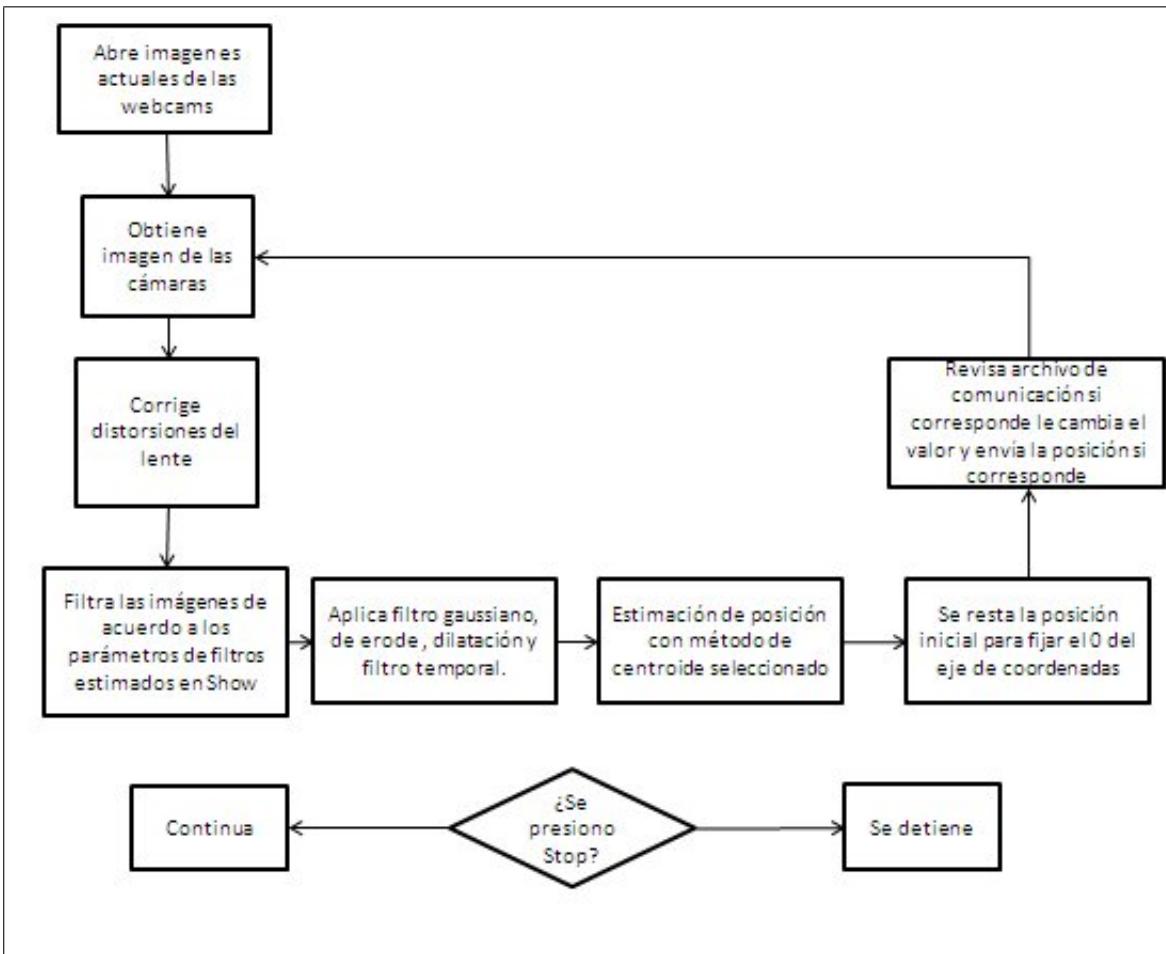


**Figura 4.7:** Diagrama de bloques de la función *Show* para seleccionar filtro de colores.

#### 4.2.2.2. Componentes del *software* Fress

##### Calibración de configuración de exposición y brillo

Esta etapa consiste en la calibración de la cámara. Es un procedimiento esencial ya que con una mala calibración de cámaras se obtienen imágenes binarias muy ruidosas, que hace variar la posición sensada con las cámaras *web* de manera aleatoria. Para la calibración se necesitan cámaras que permitan el cambio de configuración en los parámetros de *exposition* y *white balance*. Generalmente en las cámaras viene por *default* la calibración automática, lo que hace que se produzcan cambios constantes de color para corregir los colores y la iluminación. En este caso no se requieren correcciones automáticas en colores e intensidad. Esta calibración se hace cada vez que se ejecute el *software* a través de las librerías DirectShow de Windows.



**Figura 4.8:** Diagrama de bloques de la función *Start* para determinar posición y comunicarse con Mach3.

El *software* Fress posee una sección *Camera calibration* destinada a la calibración de los parámetros de exposición e iluminación. Para una buena medición desde las cámaras *web*, se requiere que la zona que se desea medir se encuentre bien iluminada (a mayor iluminación menor es el error). Esto es para minimizar el tiempo de exposición de las cámaras, ya que el error es acumulativo y a mayor tiempo de exposición mayor es la integración del error. Con el fin de mantener una buena velocidad de *software*, la calibración de exposición se hace únicamente al comienzo de cada aplicación del *software*. Para hacer efectiva la variación en la exposición se debe detener la aplicación que se esté ejecutando y volver a ejecutarla (ya sea *Show* o *Start*). Después de cada calibración se debe presionar el

botón *Save* que se encuentra en *color calibration*, el cual sirve para guardar la configuración de calibración.

El brillo es independiente y mientras se mantenga una buena captura de imágenes en las cámaras *web* el brillo puede variar sin perjudicar los resultados.

### **Calibración de distorsión**

Existe otro parámetro de calibración y viene dado por la distorsión propia del lente de la cámara. Esta corrección sólo se hace para la cámara horizontal, ya que para la medición de las posiciones de esta cámara se abarca el campo de visión casi completo, a diferencia del eje vertical. Para más detalles sobre estas distorsiones revisar la sección 2.4.

Para realizar la calibración de distorsiones se creó un *software* que obtiene los parámetros intrínsecos y extrínsecos de las cámaras y que son guardados en archivos independientes que usa el *software* Fress. Este *software* obtiene de bordes de un tablero de ajedrez para la estimación de los parámetros de distorsión.

Es necesario tener impreso un tablero de ajedrez de 9x6 bordes. En el anexo A del manual se adjunta la imagen que se utiliza para esta calibración.

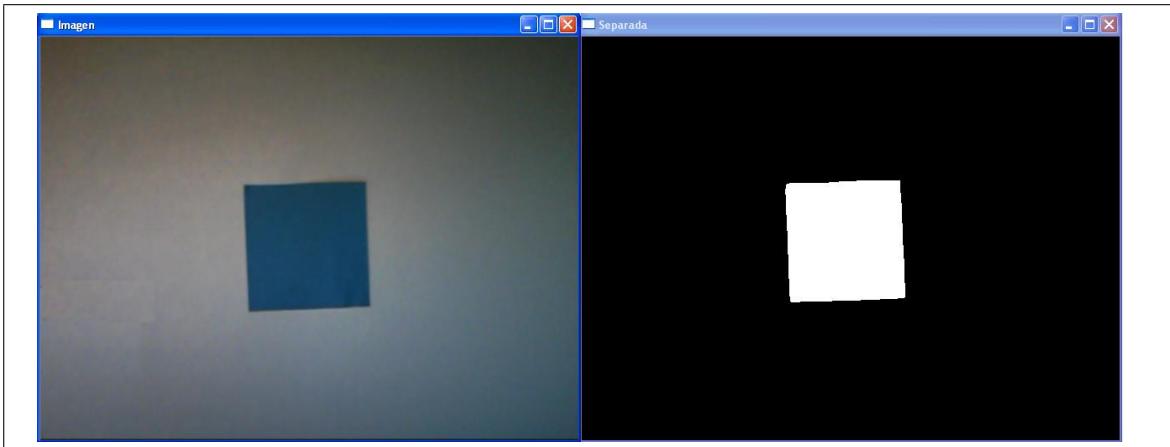
### **Calibración de lectura de colores**

La calibración de colores se hace desde la sección *Color calibration* desde el *software* Fress tal como se muestra en la figura 4.6. La clasificación se hace sobre una transformación de colores HSV. Luego, de acuerdo a los parámetros fijados en los intervalos en los rangos de H, S, V, se hace una clasificación de todos los pixeles que pertenezcan a este rango.

Los parámetros del *threshold* se refieren al  $\Delta$  del intervalo que se utiliza para la selección de colores. Actualmente se encuentra en los valores que se presentan en la figura 4.6, dado que fueron los mejores resultados obtenidos después de varias pruebas.

Para seleccionar el rango de colores, saturación e intensidad, primero se debe ejecutar el *software* mediante el botón *Show*. Luego se debe seleccionar sobre qué eje se desea calibrar, ya sea vertical u horizontal y hacer *click* sobre el punto que posee el color que se desea clasificar. De esta manera automáticamente se seleccionan los datos del pixel

con valor  $\pm threshold$  de clasificación. En la figura 4.9 se presenta un ejemplo de esto. Además se movieron los rangos de saturación y valor aumentándolos para obtener mejores resultados. Una vez finalizada la calibración hay que hacer *click* sobre el botón *save* para que los parámetros queden guardados. Mientras mejor sea esta calibración, más eficiente es la precisión obtenida por el sensor a nivel de subpixeles.



**Figura 4.9:** Clasificación de colores.

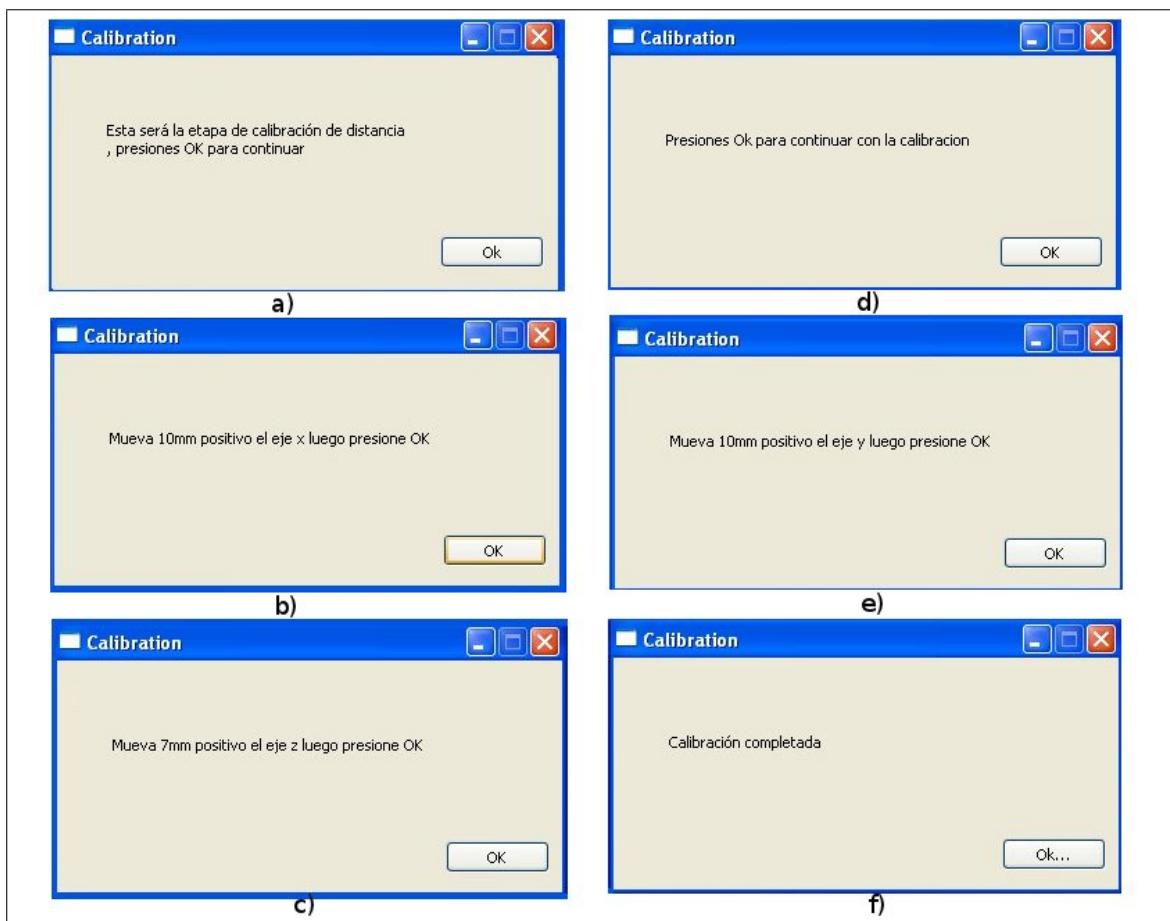
### Calibración de distancias

Esta etapa consiste en relacionar las distancias en pixeles con el movimiento real de la máquina. Es necesario ingresar varios parámetros de calibración y mover la máquina en las distancias pedidas para poder hacer el cálculo de  $X$  de la siguiente fórmula.

$$\Delta Distancia_{mm} = X \cdot \Delta Distancia_{pixeles} \quad (4.1)$$

El parámetro X es calculado para cada eje. Es necesario tener el *software* funcionando a través del botón *Start* y posicionar el círculo lo más al centro posible de la imagen, ya que es la posición inicial de calibración y corresponde a la posición referencial del sistema. Luego se debe presionar "Set 0" para marcarlo como cero de referencia y presionar *Calibration* para comenzar la calibración. Al realizar este proceso, aparece en la pantalla la imagen de la figura 4.10. Para las pantallas A y B, el punto deberá estar en el centro, ya que ese se fija como el cero de la máquina y luego se posiciona respecto a ese punto. En el caso de C, se debe mover el eje X de la máquina en 10 mm, positiva o negativamente. En D se

hace lo mismo para el eje Y desde la última posición fijada y para la figura E, lo mismo para el eje Z pero con 7 mm. Con los pasos anteriores el *software* ya es capaz de realizar la transformación de pixeles a mm, con la relación de las ecuaciones que se encuentran en la sección 2.5.1. Los valores de la calibración se guardan en un archivo que se lee cada vez que se ejecuta el *software*.



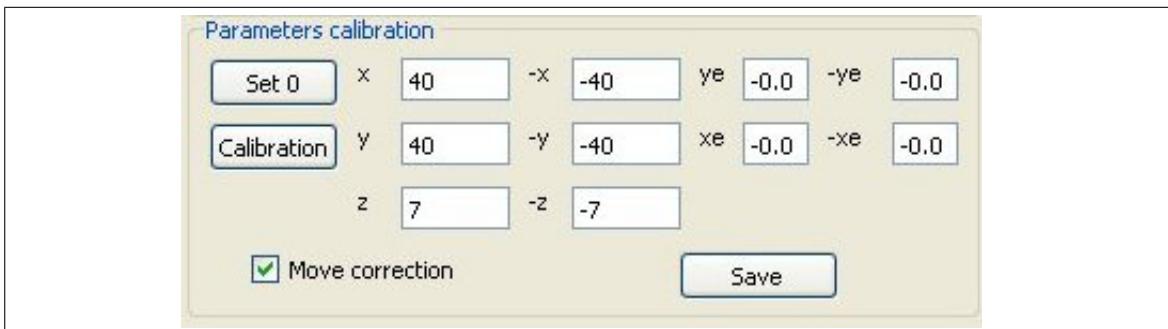
**Figura 4.10:** Calibración de posicionamiento.

Dado que no es posible corregir el total de las distorsiones del lente, aun queda un error a grandes distancias, por lo que es necesario agregar una nueva calibración para corregirlo. Primero se fijan los parámetros tal cual como se muestra en la figura 4.11 y luego se corrigen de la siguiente manera:

Primero se debe mover el eje X a la posición 40 mm y al origen en los otros ejes. La posición obtenida en X se guarda en el bloque "x", la posición obtenida como error en

Y se guarda en el bloque "ye" del *software* y luego se presiona *Save*. Este error "ye" se debe a que el eje cartesiano X-Y no calza perfectamente con el eje de la cámara, por lo tanto cada movimiento en X implica un movimiento en Y, también su recíproco. Luego de presionar *Save* se puede ver directamente que el error disminuye prácticamente a cero, de lo contrario se debe corregir el valor ingresado. Con esta corrección se elimina el error a grandes distancias. Luego se deben repetir los pasos anteriores para las posiciones -40 mm en cada eje y completar los campos "-x", "-ye", "y", "xe", "-y", "-xe" y finalmente para "z" y "-z". El eje "z" no posee corrección de otros ejes ya que no es dependiente ni del eje y ni del eje X. Finalmente se presiona *Save*, guardando los datos para futuras ejecuciones del *software*. La figura 4.12 muestra un ejemplo de una calibración final.

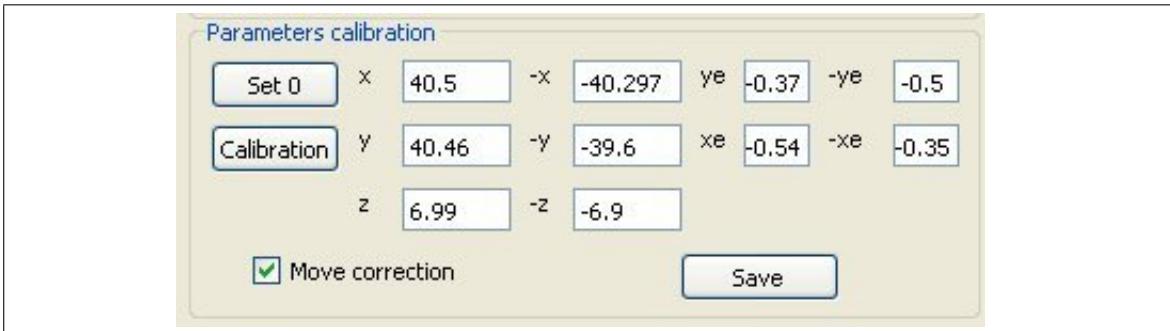
Lo que se hace con esta calibración es una estimación del parámetro  $\lambda$  que se presenta en la sección 2.5.1. Además se hace una re-estimación de los parámetros de transformación de pixeles a mm para distancias mayores de 20 mm, con la que se usa la ecuación  $\frac{40}{pos_{40}} \cdot pos_{actual}$  como corrección a grandes distancias, con  $pos_{40}$  la posición obtenida al mover la máquina a la posición 40 con el Mach3.



**Figura 4.11:** Parámetros iniciales de la segunda fase de calibración.

## Filtros

La sección *filter* posee un *checkbox* que corresponde a si se muestra o no la imagen que presenta el seguimiento de la trayectoria de la fresa (Ploteo). Esta imagen corresponde al seguimiento completo de la trayectoria X-Y con el trazado de las líneas que unen dos puntos del centroide encontrado en las imágenes  $Imagen_{t-1}$  y  $Imagen_t$ . Los otros tres bloques Imagen, Imagen-1 e Imagen-2, corresponden a un filtro temporal para eliminar los



**Figura 4.12:** Parámetros calibrados de la segunda fase de calibración.

*speckles* que se producen. Estos valores corresponden a un filtro temporal y los tres valores deben sumar 1.

#### 4.2.3. Comunicación entre programas

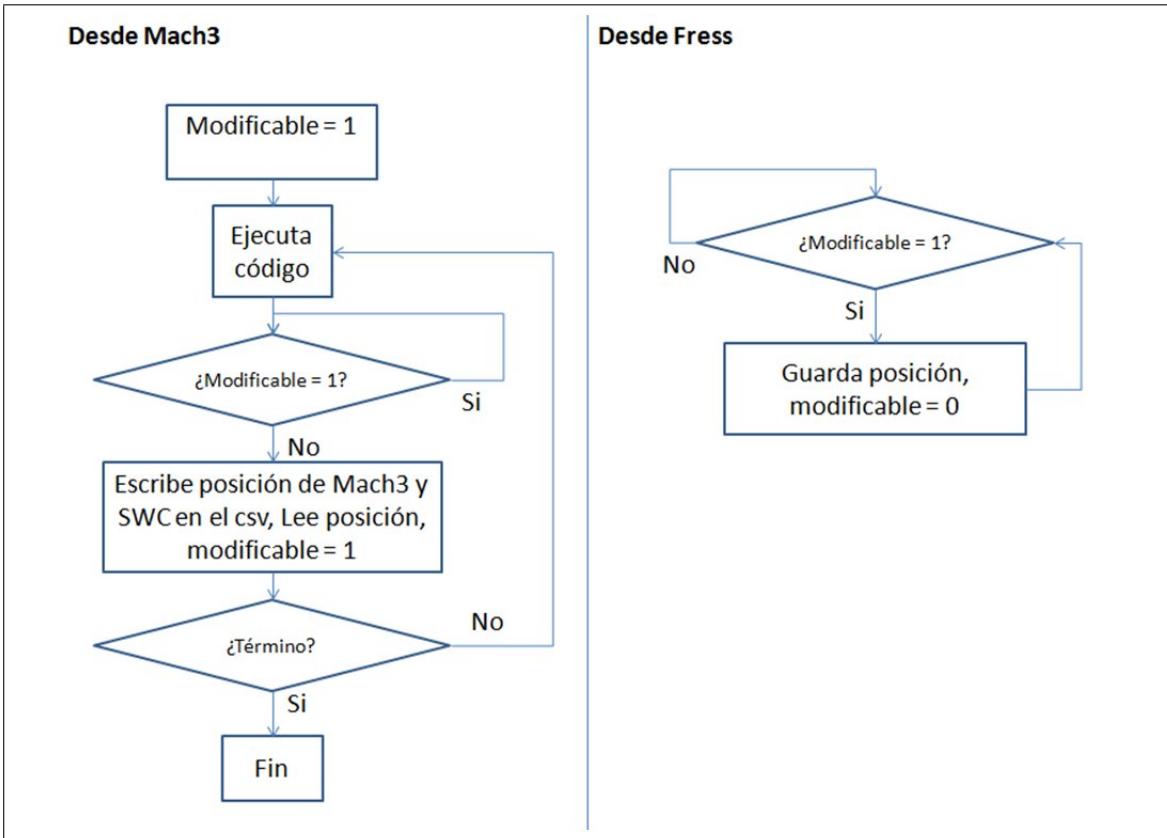
La comunicación entre ambos *software* se realiza a través de archivos, para lo que se crearon dos archivos, uno principal que envía las posiciones leídas desde la cámaras *web* (un archivo que se cambia de valor desde ambos *software*) y otro verificador que permite revisar a qué *software* le corresponde acceder al archivo de comunicación.

Se realiza un protocolo de comunicación, el cual se muestra en un diagrama explicativo en la figura 4.13. Aquí existen dos archivos, al que se desea acceder y posee las posiciones de las cámaras y el que posee un parámetro llamado Modifiable, el cual representa a un parámetro que dice a qué *software* le toca acceder al archivo de posición. Este parámetro Modifiable posee un valor de 1 si le toca acceder al *software* Fress o cero si le toca acceder al *software* Mach3. Una vez que se accede al archivo el parámetro Modifiable cambia de valor para darle paso al otro *software* a acceder.

### 4.3. Pruebas para selección de parámetros

#### 4.3.1. Pruebas de error por mal ajuste mecánico

Se hace una prueba sencilla para comprobar el desajuste mecánico que existe en la máquina. Se fija una posición de la fresa con la máquina calibrada, luego se fuerza manualmente para probar cuánto movimiento por descalce posee. De esta manera es posible



**Figura 4.13:** Comunicación entre Mach3 y el *software* Fress.

estimar el error que se produce por mal calce mecánico del tornillo y estructura de la máquina.

Los errores mecánicos obtenidos para cada eje son los que se muestran en el siguiente cuadro:

$Error_x$	$Error_y$	$Error_z$
$\pm 0,09$	$\pm 0,18$	$\pm 0,115$

TABLA 4.3. Error mecánico.

#### 4.3.2. Pruebas de colores

Para la prueba de colores se utilizan varios colores con un mismo fondo, y se mire el centroide de un objeto estático de manera de poder determinar las variaciones que se producen debido al ruido del CCD y ver qué color es más fácil de segmentar y con mejor

precisión, dado que a menor variación en el centroide mejor es la detección completa del color. Los cuadros 4.4 y 4.5 muestran los mejores resultados obtenidos para figuras de distintos colores y tamaños. Los mejores resultados fueron obtenidos para el color celeste de tamaño 7.8x7.8 cm<sup>2</sup> a una distancia aproximada de 15 cm de la cámara.

<b>Color</b>	<b>Varianza X</b>	<b>Varianza Y</b>
Amarillo	0.003212477	0.001192735
Azul	0.003531178	0.004623181
<b>Celeste</b>	<b>6.56325E-05</b>	<b>5.2289E-05</b>
Naranjo	0.001328903	0.002044791
Naranjo claro	7.48177E-05	6.64936E-05
Rojo	0.000252828	0.000154864
Rosado	6.96101E-05	0.000111713
Verde	8.04667E-05	9.04728E-05
Verde Limón	0.000194723	0.000118751

TABLA 4.4. Error de medición de centroide en pixeles.

Tamaño	$Varianza_x$	$Varianza_y$
0.2x0.2	0.032830222	0.003299242
0.3x0.4	0.001688811	0.000153368
0.5x0.6	0.000334636	0.000259935
1x0.9	5.95601E-05	9.92656E-05
1.4x1.3	6.46452E-05	6.07811E-05
1.9x1.8	5.48418E-05	3.49981E-05
2.6x2.6	1.36541E-05	1.01175E-05
2.9x2.9	1.92733E-05	2.37279E-05
3.5x3.1	2.44792E-05	1.3492E-05
4.3x4.5	1.0262E-05	8.29452E-06
5.5x5.1	1.3046E-05	7.47244E-06
6.5x6.3	1.27059E-05	7.07E-06
<b>7.8x7.8</b>	<b>4.45258E-06</b>	<b>5.51E-06</b>
8.9x8.9	6.39368E-06	4.40344E-05
10x10	1.20281E-05	3.38541E-05

TABLA 4.5. Error de centroide por tamaños.

## Capítulo 5. RESULTADOS EXPERIMENTALES

En este capítulo se presentan los resultados experimentales obtenidos de la implementación realizada. Se presentan los resultados del *software* Fress para la medición de posición y los de fresado a través de la combinación de Fress y la aplicación creada sobre el Mach3.

### 5.1. Resultados de pruebas

#### 5.1.1. Resultados de eficiencia de *software*

Las pruebas de eficiencia se realizan sobre un computador Pentium 4 de 2.4 GHz y 512 Mb de RAM y una tarjeta de video SVGA II de 128 MB de memoria RAM.

Bajo estas condiciones, al ejecutar el botón *Start* el computador pasa de un uso de procesador de 0-5 % a 97-100 %, y 100 MB de RAM. Cada iteración tarda entre 0.2 y 1.2 segundos. La eficiencia de *software* define la cantidad de imágenes y estimación de posición que se procesan por segundo, por lo que se establece un tiempo mínimo de ejecución de 1.2 segundos entre ejecuciones de comandos de código-G.

#### 5.1.2. Errores de medición

Los errores de medición se evalúan mediante una prueba similar a la de la sección 4.3.2. Se mide la variación de posición en mm que se obtiene en el centroide debido al ruido medido desde el sensor.

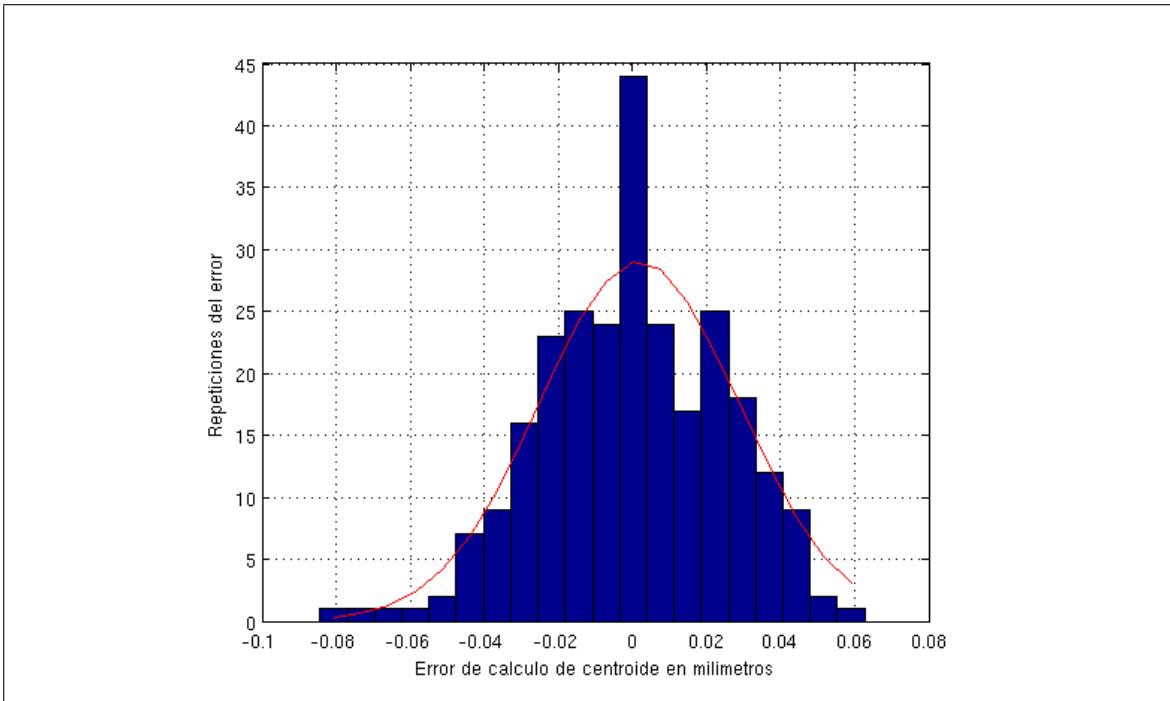
En la tabla 5.1 se puso el valor de 3 desviación estándar ( $3\sigma$ ), por cada eje y el promedio  $\bar{X}$ . Con estos valores y asumiendo que el gráfico del error de x de la figura 5.1 posee una distribución normal, se obtiene un 99.73 % de posibilidades de que el error esté entre el intervalo  $\bar{X} \pm 3\sigma$ . Es decir, para los ejes X-Y se obtienen errores de medición menores a  $0.00249 \pm 0.0057$  mm y 0.01 para el eje Z con un 99.73 % de confianza.

#### 5.1.3. Resultados de prueba de repetibilidad

La prueba de repetibilidad consiste en llevar a cabo un experimento reiteradas veces y comprobar qué tanta semejanza posee uno con el otro, ya que existe un nivel de variabilidad

	<b>X</b>	<b>Y</b>	<b>Z</b>
$3\sigma_{\text{pixeles}}$	0.015127334	0.018095102	0.085676919
$3\sigma_{\text{milímetros}}$	0.005718132	0.006839949	0.0112385875
$\bar{X}_{\text{pixeles}}$	0.019770992	0.021984733	0.118496183
$\bar{X}_{\text{mm}}$	0.002491145	0.002770076	0.014930519

TABLA 5.1. Error en estimación de centroide por ruido del CCD.

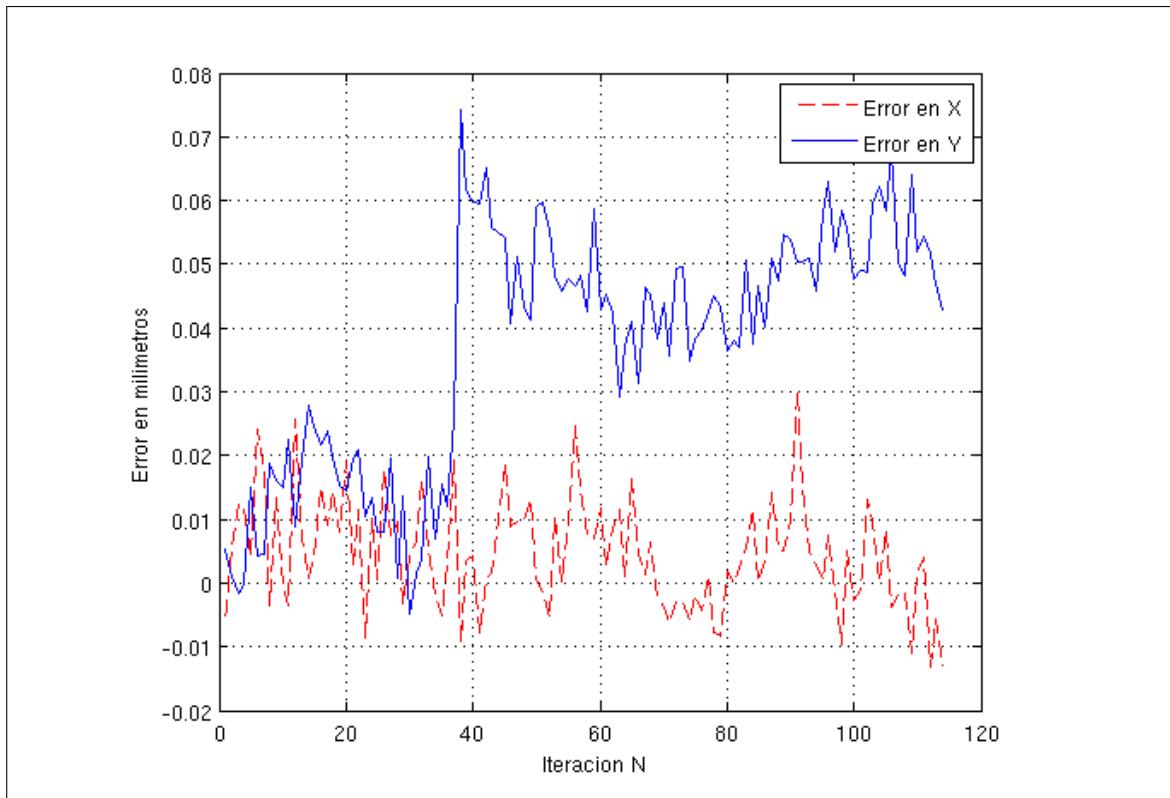
**Figura 5.1:** Histograma del error medido en la estimación de centroide producido por el CCD.

entre cada experimento que se realiza. De esta manera se puede comprobar las diferencias inevitables que se producen al momento de crear una pieza. Para esta prueba se hace un movimiento repetitivo hacia una posición fija de ida y vuelta N veces y se comprueban los errores.

En esta prueba se repite 15 veces, de la misma manera, un movimiento. La máquina se mueve desde 0 a una posición cualquiera para luego volver a 0; acto seguido se registra la posición que se lee en 0. De la prueba se logra extraer el error por repetibilidad y el error mecánico producido.

El error mecánico se produce en cualquiera de los ejes X-Y bajo cualquier movimiento X-Y. Por ejemplo en la figura 5.2 se produce sobre el eje Y bajo el movimiento del eje X. El error producido por el descalce mecánico se describe en detalle en la sección 4.3.1 y se produce de manera aleatoria en cualquiera de los ejes.

El error producido por repetibilidad y sin la consideración de estos errores mecánicos es de 7.6 % del error mecánico máximo. Por lo tanto este error se considera despreciable bajo el supuesto de que el error mecánico producido está en los rangos definidos en la sección 4.3.1



**Figura 5.2:** Error por repetibilidad y error mecánico.

Sólo se consideran los errores de lectura del sensor, repetibilidad y error mecánico. Los dos primeros tienden a 0 y se representan por la siguiente ecuación:

$$\begin{aligned}
 Error_{repetibilidadtotal} &= \overbrace{Error(\mu_{sensor}, \sigma_{sensor})}^{\epsilon} \\
 &+ \overbrace{Error(\mu_{repetibilidad}, \sigma_{repetibilidad})}^{\epsilon} + Error_{mecánico} \\
 \text{con } \epsilon &\rightarrow 0
 \end{aligned} \tag{5.1}$$

#### 5.1.4. Resultados de resolución de medición de movimiento

Los resultados de resolución de medición se prueban una vez calibrada la máquina tras varios movimientos en su cantidad mínima. Es decir, 0.01 mm desde una posición 0 hasta una distancia de 0.5 mm, lo que permite comprobar cuál es la resolución mínima de medición de la máquina y estimar el movimiento mínimo perceptible.

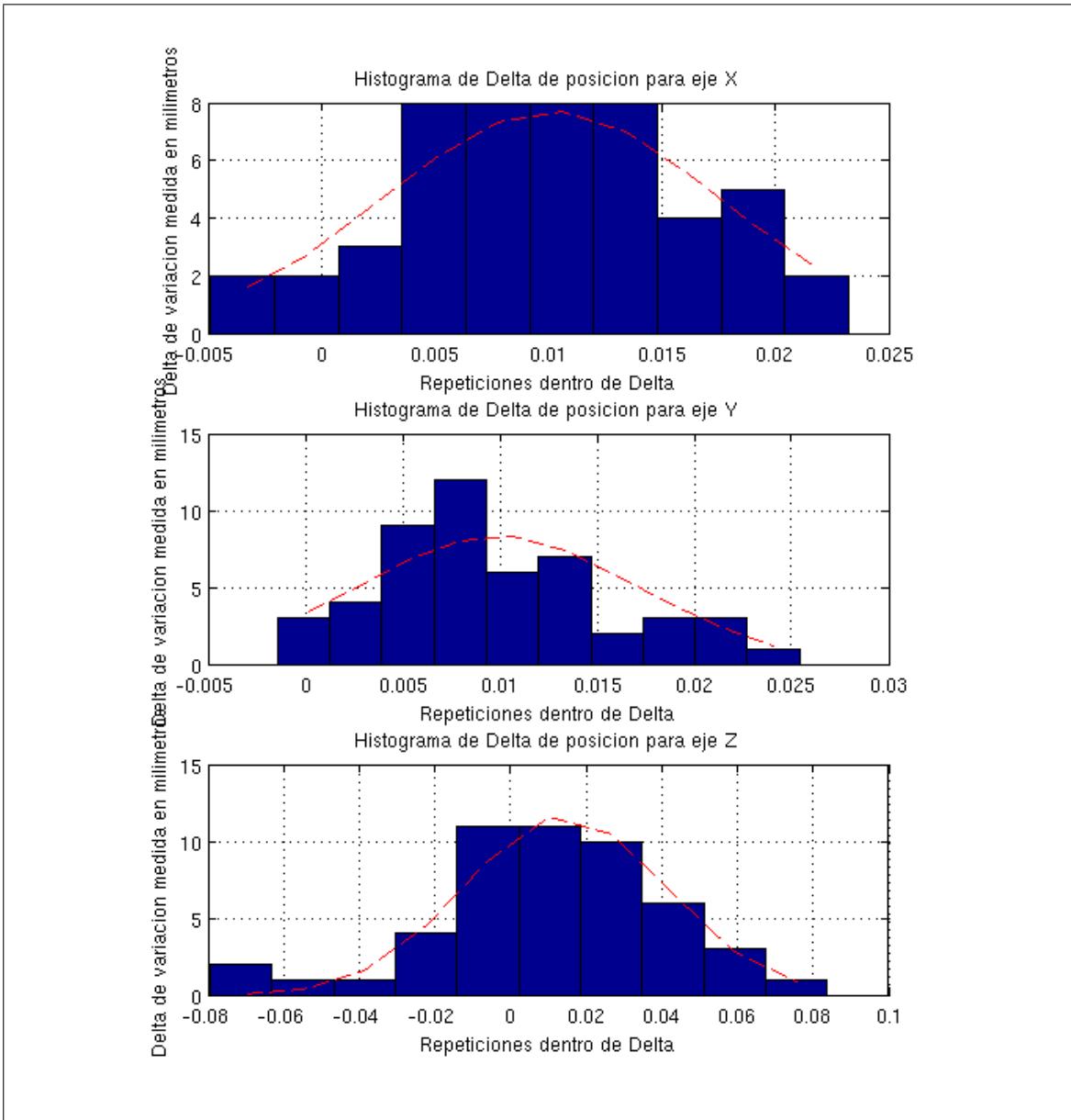
Dado que el movimiento mínimo se produce a 0.01 mm aproximadamente, la resolución mínima se estima en base a esta cantidad de movimiento. Para este cálculo se asume una distribución normal. En la figura 5.3 se puede ver que el error de medición del paso de 0.01 mm puede seguir una distribución normal y en la figura 5.4 se puede ver la medición con error de media 0.01 mm para el eje Y.

Usando el criterio de test de hipótesis con  $3\sigma$  se puede estimar una resolución de  $0,01 \pm 0,0201 \text{ mm}$  de movimiento en el eje Y con una confianza de 99.73 %. Se realizó la misma prueba para distintas cantidades de pasos, (0.02 mm, 0.03 mm y 0.04 mm) y para todos se obtuvo un valor de  $3\sigma \cong 0,0201 \text{ mm}$ . Se puede concluir que es posible obtener una resolución máxima de medición hasta 0.01 mm, pero con un error de 0.018 mm. Utilizando el mismo criterio de  $3\sigma$ , se obtuvo una precisión de  $0,01 \pm 0,0224 \text{ mm}$  para el eje X y  $0,01 \pm 0,0724 \text{ mm}$  para el eje Z.

### 5.2. Resultados finales

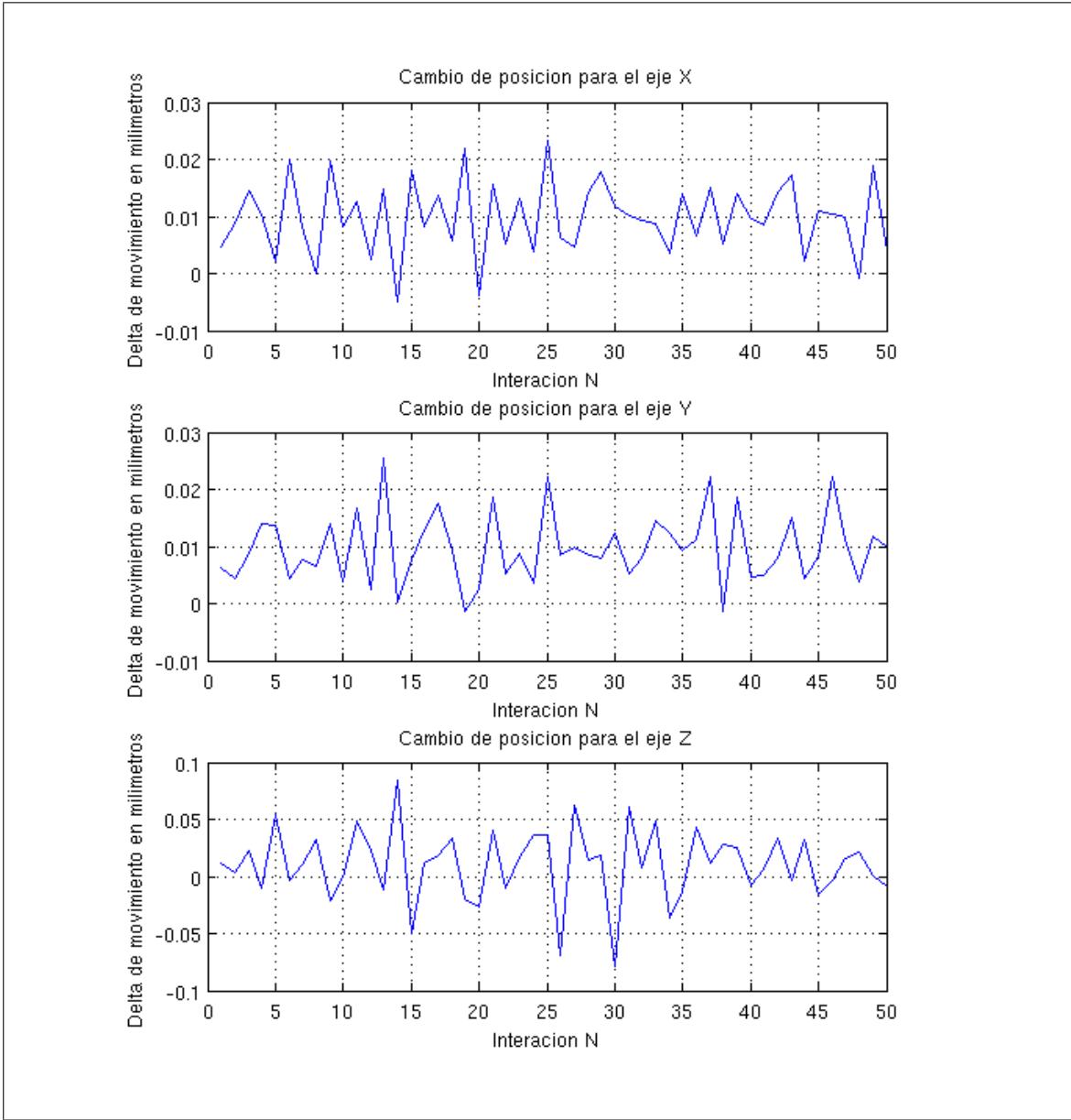
#### 5.2.1. Resolución del software Fress

Dado que el error es directamente dependiente de la posición, se hace una prueba en todo el espacio de visión de la cámara y se determina el error en la posición estimada respecto a la posición real. Para revisar el campo completo se mueve la máquina de



**Figura 5.3:** Distribución de resolución a 0.01 mm.

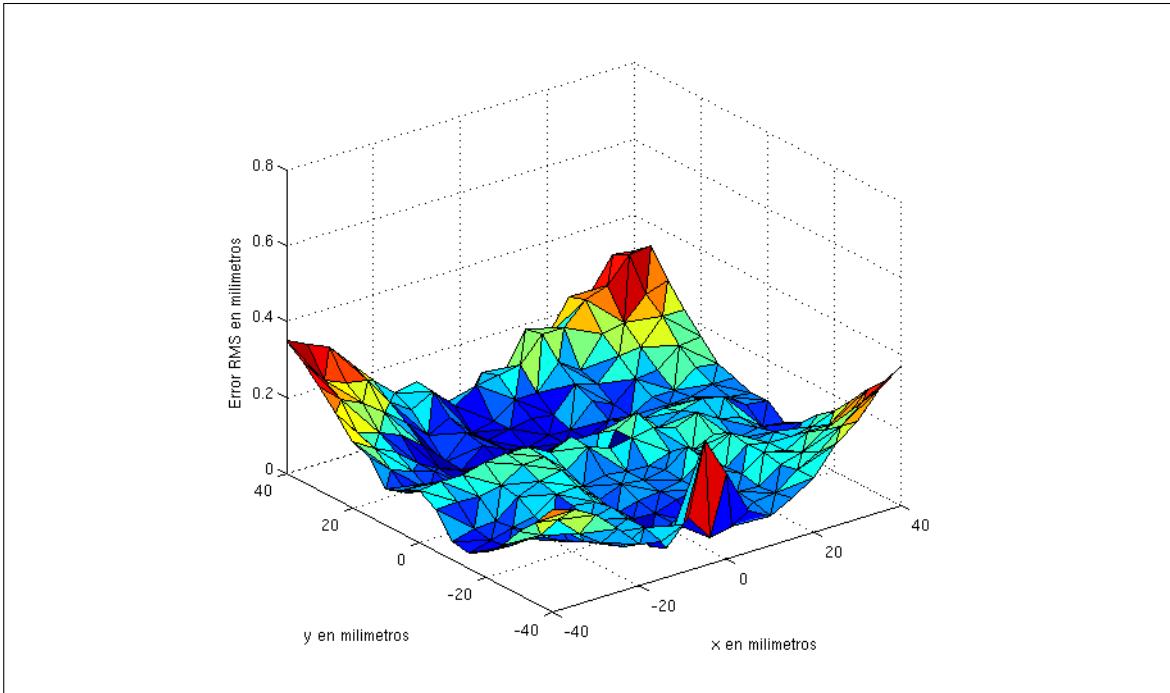
modo de medir una grilla completa por cada 5 mm. Para verificar que no existe error de posición se utiliza un punto de referencia, el cual debe tener el mismo valor cada vez que se mueve la fresa a esta posición, luego cada cinco líneas de comando se envía la fresa a la posición de verificación. De esta manera se puede crear una función del error  $Error(5 \text{ mm} \cdot N_1, 5 \text{ mm} \cdot N_2)$ , con  $N_1, N_2 = -8, -7, \dots, 7, 8$ .



**Figura 5.4:** Resolución con pasos de 0.01 mm.

Los resultados obtenidos del error en todo el campo de visión se muestran en la figura 5.5, en la cual se presenta el error  $Error_{rms}(x, y) = \sqrt{Error_x(x, y)^2 + Error_y(x, y)^2}$  con,  $Error_{\vec{x}} = |\vec{X}_{real} - \vec{X}_{sensor}|$ . Para una observación en un espacio de 40x40 mm se obtuvo un error máximo de 0.386 mm en la estimación de posición para las zonas más externas de la imagen. Este error sucede mayoritariamente por la distorsión, incluso tras la calibración de la cámara. La corrección se ve limitada por los órdenes en la estimación

de las distorsiones tangenciales y radiales. Se hicieron varias pruebas y en todas se obtuvo resultados similares, en las que no se supera un error de 0.4 mm.



**Figura 5.5:** Error RMS de todo el campo de visión del software Fress.

### 5.2.2. Resolución máxima

La resolución máxima obtenible queda determinada por el error que se produce en la mecánica de la máquina, es decir,  $\pm Error_{eje_i}$ , con  $i$  el eje X, Y o Z (los valores se encuentran en la sección 4.3.1). La mayor dificultad para alcanzar estas resoluciones está relacionado con los errores que se producen aleatoriamente en los motores, que generan pérdidas en los pasos de éstos por exceder los torques permitidos por los motores. En algunos casos son posibles de corregir con la aplicación *closed loop* y el sensor de las cámaras *web*, mientras que para otros es posible detenerlo antes que el error se siga acumulando.

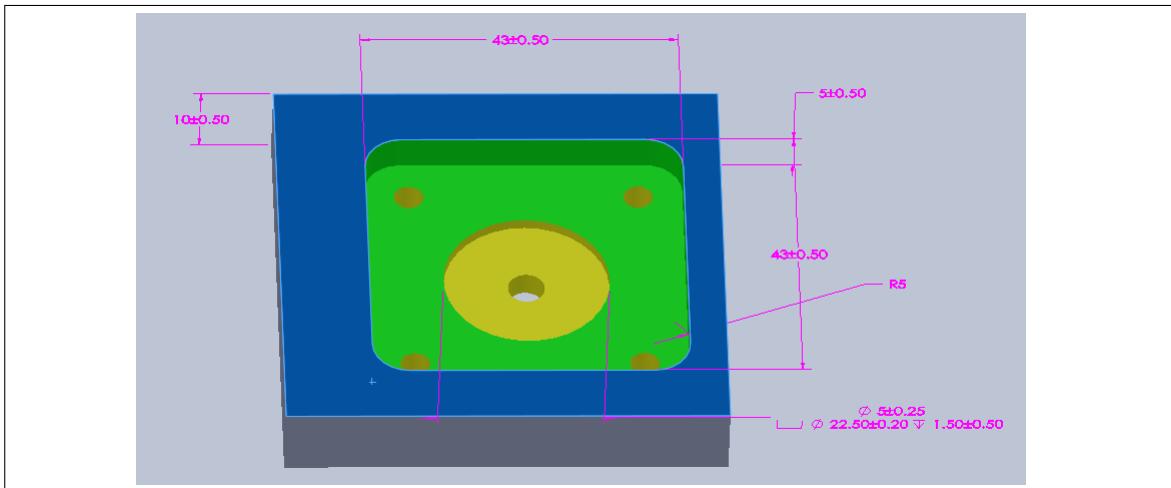
### 5.2.3. Pruebas de creación de piezas

En esta sección se presentan algunas pruebas simples de fresado en distintos materiales y situaciones, para ver su comportamiento bajo diferentes métodos que se puedan utilizar.

### 5.2.3.1. Lazo abierto

#### Caso sin error

En el fresado de lazo abierto la pieza desbastada no presentó error aparente. Las dimensiones fueron las mismas a las de la pieza diseñada desde el *software CAD*. El diseño CAD y la pieza final se presentan en las figuras 5.6 y 5.7 respectivamente. Para comprobar las dimensiones se midió el largo de la pieza, el diámetro de la circunferencia del centro y las distancias entre los agujeros. En las figuras 5.8 y 5.9 se presenta otro ejemplo de una pieza CAD y una creada. Aquí se desbastó el canal de la parte superior de la figura 5.8. Esta pieza corresponde a una cámara para un sensor CCD que se enfriá con nitrógeno líquido y se creó sin error aparente.

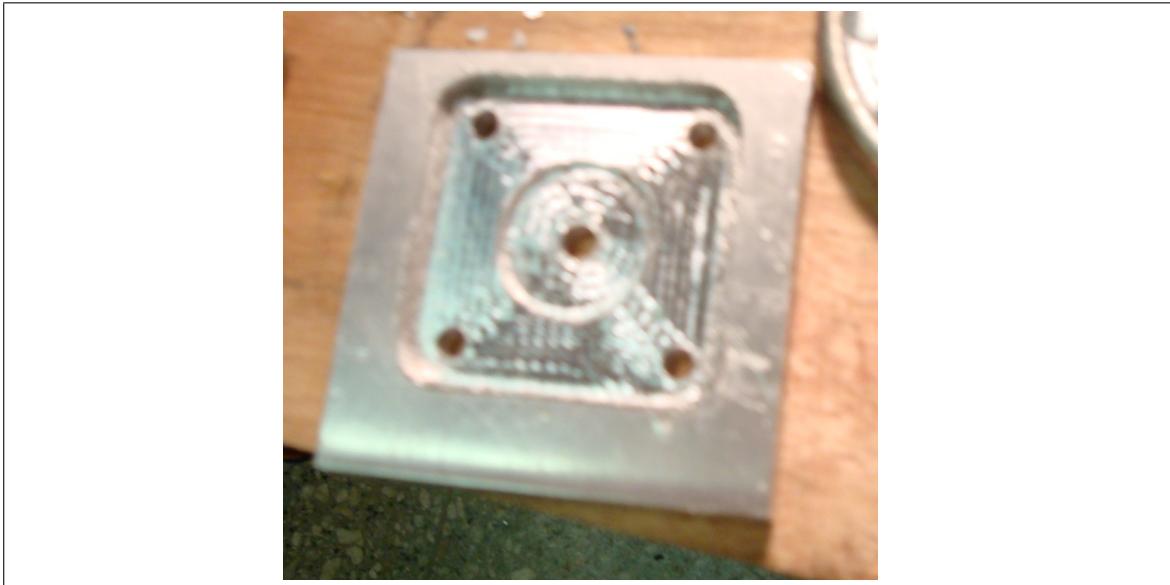


**Figura 5.6:** Pieza creada computacionalmente desde un *software CAD*.

#### Caso con error

Por otro lado se utilizó una pieza de prueba para observar los posibles efectos que se pudieran producir en el fresado de una pieza. La figura 5.8 muestra una pieza en la que se fresó únicamente el canal superior de ésta. Luego se comprobaron las dimensiones con los distintos métodos de fresado, agregando pérdida de pasos en los motores para simular errores de fresado.

Para el caso de la figura 5.10 se agregó un error en el movimiento del eje X. Dado que se encuentra en lazo abierto, aunque se presente un error, el fresado continúa y no vuelve a



**Figura 5.7:** Pieza creada en lazo abierto sin error.

la posición deseada. Este error se puede comprobar fácilmente por la falta de exactitud en las curvaturas del fresado de la figura.

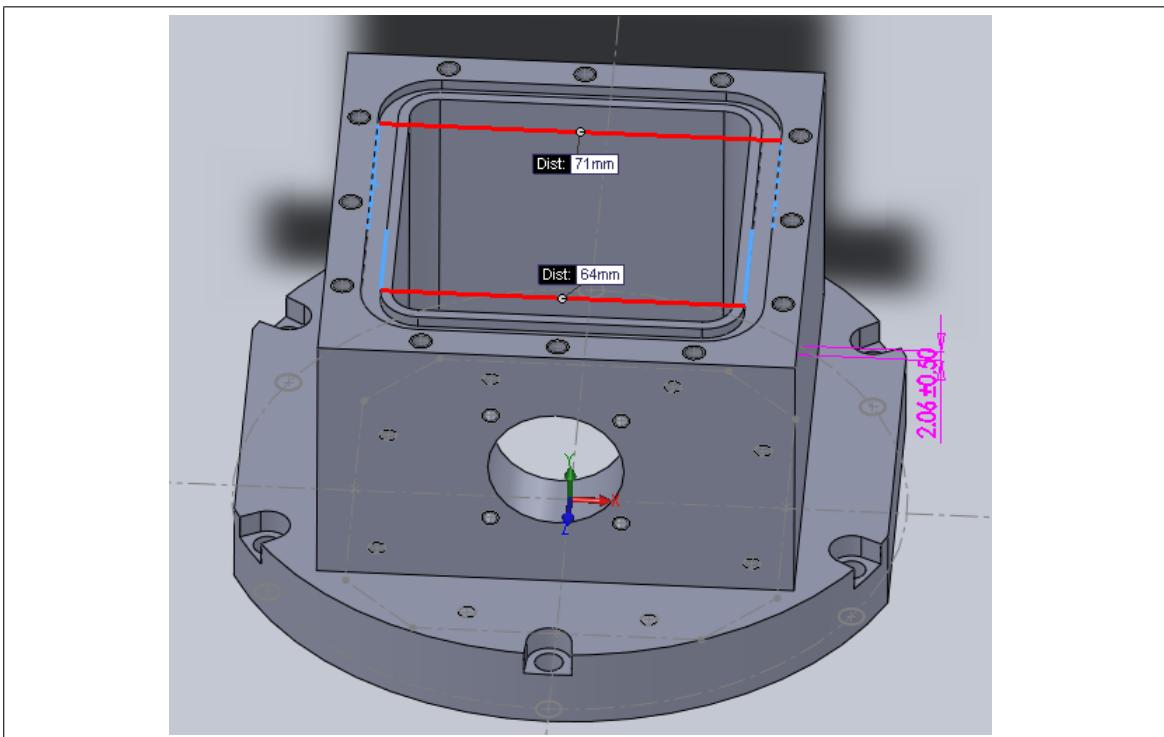
#### 5.2.3.2. Lazo cerrado

##### Método 1

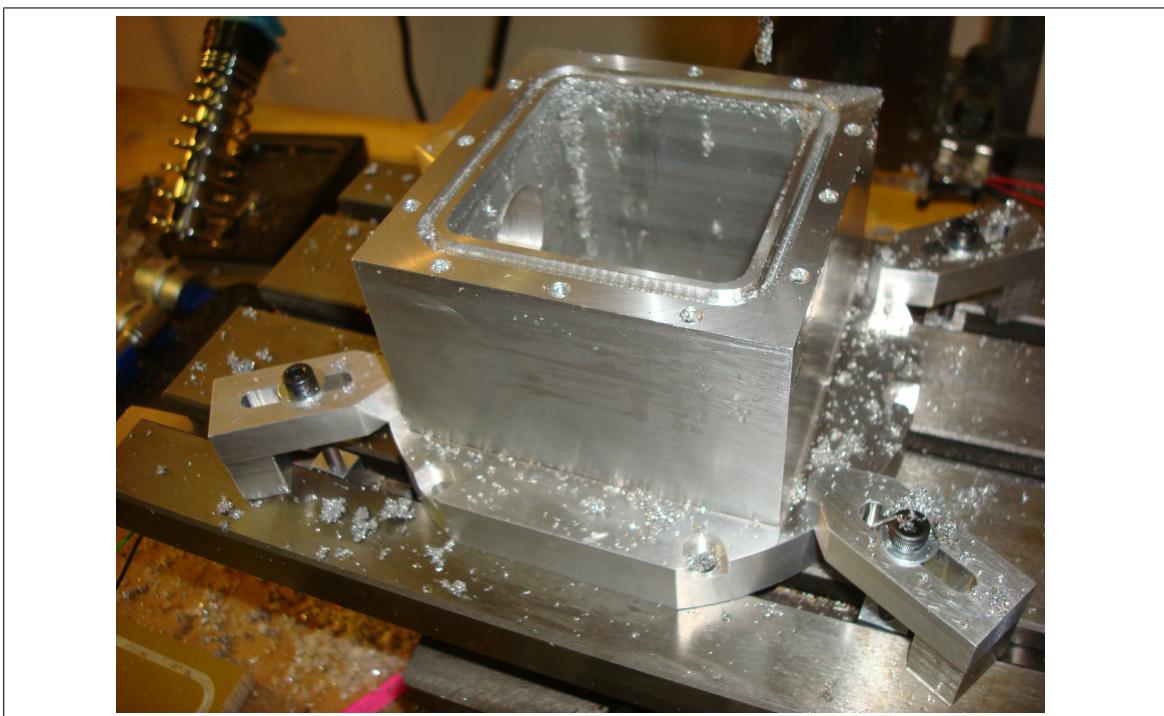
Dado el mismo error que en el caso de lazo abierto, se cierra el lazo y se aplica una corrección con el método 1. Se puede comprobar una mejor precisión y que en ocasiones la pieza comienza a perder la posición del desbastado. Luego la fresadora vuelve a la posición anterior del error y ejecuta el comando de código-G nuevamente de manera de poder corregir el error, comenzando desde la posición anterior. De esta manera se logra seguir la trayectoria perfecta.

##### Método 2

En este caso se aplica el método 2 de lazo de control cerrado. Aquí la pieza pierde posición pero se recupera automáticamente siguiendo una línea recta hasta la posición en la que debería estar la máquina. De esta manera se elimina el error acumulativo y se sigue la trayectoria deseada. En caso que el error se produzca en trayectorias circulares se genera un error en la figura fresada, dejando trazos lineales donde deberían ser circulares. En el



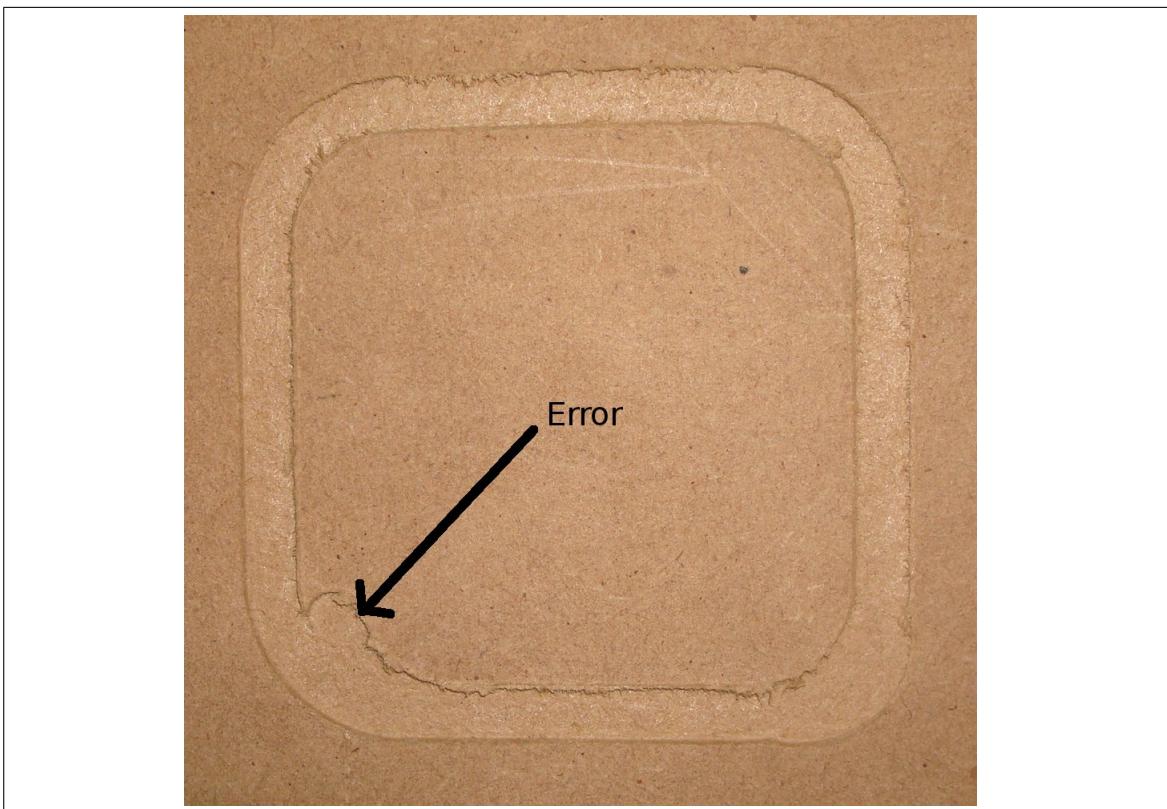
**Figura 5.8:** Pieza 2 creada computacionalmente desde un *software CAD*.



**Figura 5.9:** Pieza 2 creada con la máquina desde el *software Mach3*.



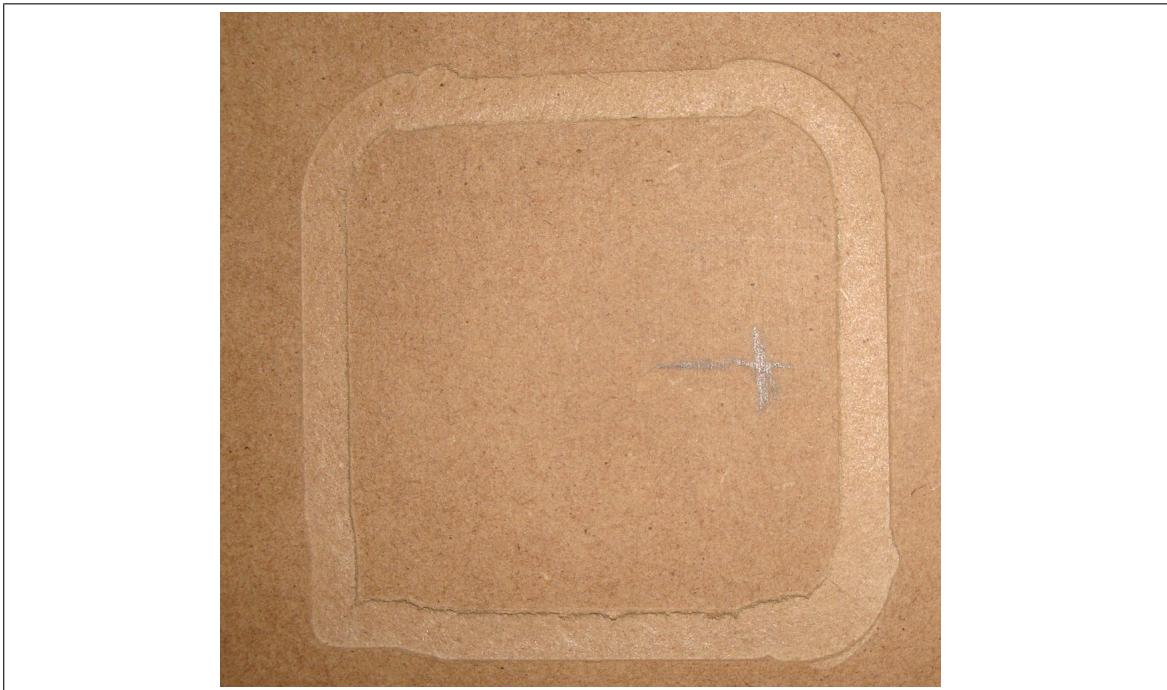
**Figura 5.10:** Pieza creada en lazo abierto y con error en los pasos de los motores.



**Figura 5.11:** Pieza con error y método 1 aplicado con lazo de control cerrado.

caso de ser trazos lineales la pieza queda perfecta y la precisión depende de la medición de posición del *software* Fress.

Para verificar las dimensiones de la pieza fresada se empleó un pie de metro, no encontrándose diferencias respecto del diseño CAD.



**Figura 5.12:** Pieza con error y método 2 aplicado con lazo de control cerrado.



**Figura 5.13:** Pieza sin error y método 2 aplicado con lazo de control cerrado.

## Capítulo 6. CONCLUSIONES Y TRABAJO FUTURO

Agregando motores *stepper* a una fresadora manual, se logró automatizar su operación para la manufactura de piezas metálicas con una precisión del orden de 0.2 mm cuando no se presenta error por pérdida de pasos. Con la aplicación es posible corregir un error mayor de 0.4 mm y con una precisión de este mismo valor. La manufactura se realiza en base a un modelo CAD que alimentan el sistema, el cual es supervisado por medio del uso de *webcams* para la estimación de la posición de la fresa, permitiendo la interrupción o corrección de los trabajos de fresado en caso de fallos o desajustes mecánicos. El sistema desarrollado tiene un costo del orden de 6 veces menor que otros sistemas con la misma precisión disponibles en el mercado.

El método propuesto de detección de posición usando cámaras *web* tiene la característica de ser completamente modular, por lo que podría ser fácilmente implementado en otras máquinas fresadoras o aplicaciones similares que requieran de un monitoreo de posición de alta resolución.

La precisión lograda en la estimación de la posición es de  $\pm 0.4$  mm en el peor de los casos. Este se da en los extremos del campo visual debido a la distorsión radial introducida por el lente, disminuyendo el error hacia el centro de la imagen capturada. Otro factor determinante es la mala iluminación, por lo que esta debe ser ajustada apropiadamente para asegurar una buena detección del centroide en las imágenes. El error pudo ser reducido hasta 0.3 mm, pero con un incremento de 10 veces en el tiempo de cómputo. Para favorecer la usabilidad del sistema se renunció a la precisión en virtud de la velocidad de manufactura. Finalmente, cabe señalar que el campo visual cubierto por el sistema de medición es de  $\pm 40$  mm para los ejes X e Y, y de  $\pm 5$  mm para el eje Z, lo cual fue ajustado de acuerdo al rango dinámico de la máquina, siendo posible la extensión del campo en caso de ser necesario.

Al margen del error de estimación de posición, se encontró que los principales factores de imprecisión en el trabajo de fresado corresponden a errores de tipo mecánico. Estos se

deben a desalineamientos en la máquina y pérdida de pasos en los motores. Se midieron errores aleatorios de  $\pm 0.15$  mm por desalineamiento y de hasta 1 mm por pérdida de pasos.

El sistema desarrollado permite varios métodos de evaluación de la supervisión del trabajo de fresado, siendo aconsejable escoger aquel más apropiado para la manufactura de cada pieza en particular. La manufactura puede ser interrumpida o corregida de acuerdo a distintos niveles de tolerancia de error. La corrección, además, puede ser de dos tipos: direccionando la fresa a la posición en la que debería estar o volver a la posición anterior y volver a ejecutar el comando de código-G.

## 6.1. Trabajo futuro

En términos de *hardware* hay dos elementos principales que incidirían fuertemente sobre el desempeño del sistema. En primer lugar la calidad de las cámaras: dispositivos de mayor resolución y menor distorsión (tanto radial como tangencial) permitirían una importante reducción del error de estimación de posición de la fresa, limitando el error de manufactura casi exclusivamente a los desajustes mecánicos inherentes a la mecánica de la fresa. Por otro lado, sería posible acelerar el procesamiento de imágenes para la estimación del centroide si se empleara un computador más potente. Actualmente, usando un computador de escritorio, Pentium IV de 2.6 GHz, 512MB de RAM, se registran tiempos de procesamiento del orden de  $0.2 \pm 0.3$  segundos. Una mayor velocidad en el procesamiento incidiría directamente sobre el tiempo total del trabajo de fresado, optimizando el recurso tiempo en el empleo de la máquina.

En términos de software, una línea importante de trabajo sería la implementación de una evaluación en tiempo real del fresado. La estimación de la trayectoria de la fresa posibilitaría la corrección de una instrucción mientras esta se está ejecutando, y no solo al término de la misma, como ocurre con el diseño actual. Esta mejora en el sistema está limitada por la velocidad de procesamiento y no es factible bajo las condiciones actuales de hardware.

Las tres líneas de trabajo antes descritas presuponen un aumento en el costo del sistema final, lo cual debe ser correctamente evaluado en virtud del objetivo inicial de esta investigación, consistente en el desarrollo de un sistema de control de bajo costo respecto de las alternativas comerciales.

## Referencias

- Akondi Vyas, B. R. P., M B Roopashree. (2009). Optimization of existing centroiding algorithms for shack hartmann sensor. En *Innovative computational intelligence and security systems* (p. 400-405).
- Beauvais, K. (2003). *The simple dc motor: A teacher's guide* (Inf. Téc.). Massachusetts Institute of Technology.
- Distance measuring sensor unit measuring distance: 20 to 150 cm analog output type [Manual de software informático]. (2006).
- Elap pls/pl2s - linear potentiometer; potentiometer transducers [Manual de software informático]. (s.f.).
- Escalona, I. (2002). *Diseño y manufactura asistidos por computadora, introducción al cnc* (Inf. Téc.). Instituto Politécnico Nacional.
- Federico, B. (2011). *Medición con laser por triangulación* (Inf. Téc.). Universidad Tecnológica Nacional, Facultad Regional San Nicolás.
- Gary Bradski, A. K. (2008). *Learning opencv; computer vision with the opencv library* (1.<sup>a</sup> ed.). O'REILLY.
- Guerrero, M. (2011). *Overview of the hsv color space* (Inf. Téc.). Bright HUB. (<http://www.brighthub.com/multimedia/publishing/articles/122040.aspx>)

Guzmán, D. (2012). *Apuntes del curso de detectores para astronomía* (Inf. Téc.). Pontificia Universidad Católica de Chile.

Heikkila, y Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. En *Proceedings of the 1997 conference on computer vision and pattern recognition* (p. 1106).

Hoffmann, K. (s.f.). *Applying the wheatstone bridge circuit* (Inf. Téc.).

Incremental linear encoders; enclosed models [Manual de software informático]. (2008). (Datasheet)

Janesick, J. R. (2007). *Photon transfer* (1.<sup>a</sup> ed.). SPIE Press  $DN \rightarrow \lambda$ .

Jiménez, J. C. (2009). *Apuntes del curso de robótica* (Inf. Téc.). Universidad de Alicante. (<http://www.dccia.ua.es/dccia/inf/asignaturas/ROB/optativos/Sensores/mas.html>)

Magnetic displacement sensors hmc1501/1512 [Manual de software informático]. (2008).

Mohammad, T. (2009). Using ultrasonic and infrared sensors for distance measurement. *World Academy of Science, Engineering and Technology*(51), 293-298.

Paulo Malheiros, J. G., y Costa, P. (2010). *Towards a more accurate infrared distance sensor model* (Inf. Téc.). University of Porto.

Phillips, I. S. G. . W. R. (2004). *Electromagnetism* (2.<sup>a</sup> ed.). Wiley.

Tb6560ahq, tb6560ahq [Manual de software informático]. (2009). (Datasheet)

Theory in ac motors [Manual de software informático]. (2001). (Company Manual)

Theory in dc motors [Manual de software informático]. (2001). (Company Manual)

Torres, M. (2006, Mayo). *Apuntes del curso de sensores y actuadoras para robótica: Sensores - proximidad/ contacto* (Inf. Téc.). Pontificia Universidad Católica de Chile.

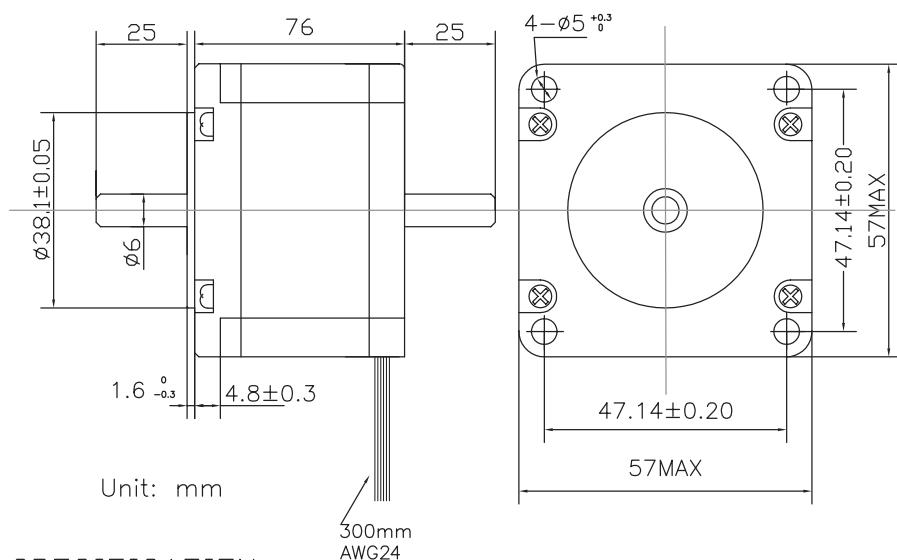
Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*(22(11)), 1330-1334.

**ANEXO A. DATASHEET MOTORES**

En este apéndice se anexan las especificaciones de los motores.

## HYBRID STEPPING MOTOR

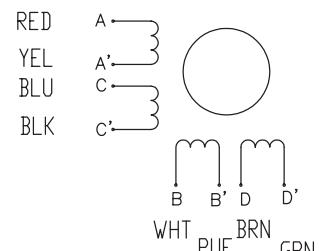
### MODEL 160-010-00200



#### SPECIFICATION

STEP ANGLE	1.8Deg
VOLTAGE	4.5 V
CURRENT	2.5A/PHASE
RESISTANCE	2 OHM/PHASE
INDUCTANCE	3.6 mH/PHASE
HOLDING TORQUE	180N.cm
MOTOR LENGTH	76mm
ROTOR INERTIA	440g.cm^2
MOTOR WEIGHT	1.1KG
INSULATION CLASS	B

#### WIRING DIAGRAM



DESIGN		
CHECK		

**Arc Euro Trade**

### **Motor connections for Arc Euro Trade Stepper Motors**

<b>160-010-00100:</b>	36Ncm, 1A/Phase, 6mm Shaft
<b>160-010-00200:</b>	180Ncm, 2.5A/Phase, 6mm Shaft
<b>160-010-00300:</b>	180Ncm, 2.5A/Phase, ¼" Shaft
<b>160-010-00400:</b>	220Ncm, 2.5A/Phase, 10mm Shaft
<b>160-010-00430:</b>	300Ncm, 4.2A/Phase, 8mm Shaft

#### **For BIPOLE SERIES:**

Join YELLOW A to BLUE C and insulate connection  
 Join PURPLE B to BROWN D and insulate connection  
 Winding One then equals RED and BLACK  
 Winding Two then equals WHITE and GREEN

#### **For BIPOLE PARALLEL:**

Join RED to BLUE  
 Join YELLOW to BLACK. This is then Winding One.  
 Join WHITE to BROWN  
 Join PURPLE to GREEN. This is then Winding Two.

#### **For UNIPOLAR FOUR PHASE:**

Use RED, BLACK, WHITE and GREEN as the PHASE wires.  
 Join YELLOW to BLUE, and PURPLE to BROWN; these then become the POWER connections.  
 The phase sequence is RED, GREEN, BLACK WHITE (or WHITE, BLACK, GREEN, RED for reverse).

The Arc Euro Trade **160-020-00101** 4.2A controller will drive our **160-010-00100**, **160-010-00200**, **160-010-00300**, **160-010-00400** and **160-010-00430** motors at their rated power if set up correctly and connected to a **smoothed DC power supply rated at 2x the current setting used on the controller**.

We regret we are unable to advise you on the suitability of these motors for any application.

### **Additional information for the Arc Euro Trade 4" and 6" Rotary Tables fitted with our 160-010-00300 Stepper Motor**

The motor is wired in Bi-Polar Parallel mode to a 4 pin XLR plug

Pin 1: Red and Blue  
 Pin 2: Yellow and Black  
 Pin 3: Purple and Green  
 Pin 4: White and Brown

We also supply a suitable XLR chassis socket, Code 160-030-00225

**ANEXO B. DATASHEET HARDWARE DE CONTROL DE MOTORES**

En este apéndice se anexa el *datasheet* del integrado TB6560 que usa el *hardware* de control de los motores.

**TOSHIBA****TB6560HQ/FG****Preliminary**

TOSHIBA BiCD Integrated Circuit Silicon Monolithic

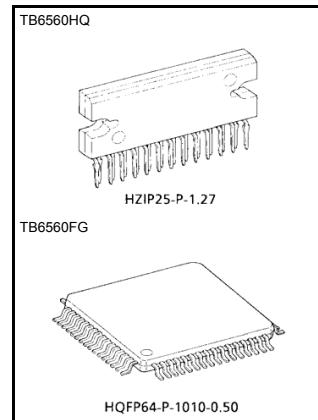
**TB6560HQ, TB6560FG**

**PWM Chopper-Type bipolar  
Stepping Motor Driver IC**

The TB6560HQ/FG is a PWM chopper-type sinusoidal micro-step bipolar stepping motor driver IC. It supports both 2-phase/1-2-phase/W1-2-phase/2W1-2-phase excitation mode and forward/reverse mode and is capable of low-vibration, high-performance drive of 2-phase bipolar type stepping motors using only a clock signal.

**Features**

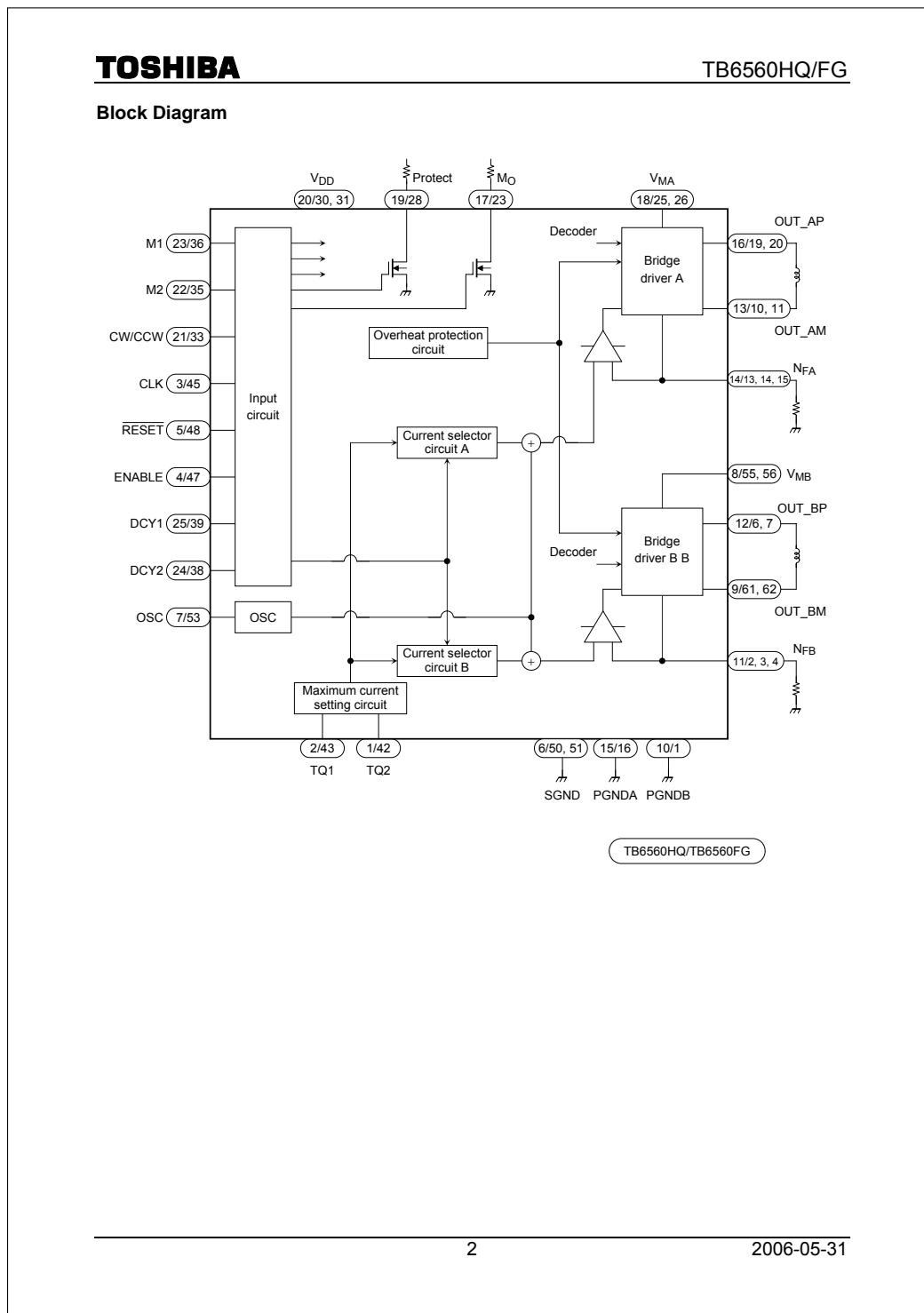
- Single-chip bipolar sinusoidal micro-step stepping motor driver
- Uses high withstand voltage BiCD process:  
 $R_{on}$  (upper: lower) = 0.6  $\Omega$  (typ.)
- Forward and reverse rotation control available
- Selectable phase drive (2, 1-2, W1-2, and 2W1-2)
- High output withstand voltage:  $V_{CEO} = 40$  V
- High output current:  $I_{OUT} =$  HQ: 3.5 A (peak)  
FG: 2.5 A (peak)
- Packages: HZIP25-P-1.27/HQFP64-P-1010-0.50
- Built-in input pull-down resistor: 100 k $\Omega$  (typ.)
- Output monitor pin equipped: MO current ( $I_{MO}$  (max)) = 1 mA
- Equipped with reset and enable pins
- Built-in overheat protection circuit



Weight:  
HZIP25-P-1.27: 9.86 g (typ.)  
HQFP64-P-1010-0.50: 0.26 g (typ.)

The TB6560HQ/FG is a Pb-free product.  
The following conditions apply to solderability:  
\*Solderability  
1. Use of Sn-63Pb solder bath  
\*solder bath temperature = 230°C  
\*dipping time = 5 seconds  
\*number of times = once  
\*use of R-type flux  
2. Use of Sn-3.0Ag-0.5Cu solder bath  
\*solder bath temperature = 245°C  
\*dipping time = 5 seconds  
\*the number of times = once  
\*use of R-type flux

\*: Since this product has a MOS structure, it is sensitive to electrostatic discharge. These ICs are highly sensitive to electrostatic discharge. When handling them, please be careful of electrostatic discharge, temperature and humidity conditions.



**TOSHIBA****TB6560HQ/FG****Pin Functions**

Pin No.		I/O	Symbol	Functional Description
HQ	FG			
1	42	Input	TQ2	Torque setting input (current setting) (built-in pull-down resistor)
2	43	Input	TQ1	Torque setting input (current setting) (built-in pull-down resistor)
3	45	Input	CLK	Step transition, clock input (built-in pull-down resistor)
4	47	Input	ENABLE	H: Enable; L: All output OFF (built-in pull-down resistor)
5	48	Input	RESET	L: Reset (output is reset to its initial state) (built-in pull-down resistor)
6	50/51	—	SGND	Signal ground (control side) (Note 1)
7	53	—	OSC	Connects to and oscillates CR. Output chopping.
8	55/56	Input	VMB	Motor side power pin (B phase side) (Note 1)
9	61/62	Output	OUT_BM	OUT_B output (Note 1)
10	1	—	PGNDB	Power ground
11	2/3/4	—	NFB	B channel output current detection pin (resistor connection). Short the two pins for FG. (Note 1)
12	6/7	Output	OUT_BP	OUT_B output (Note 1)
13	10/11	Output	OUT_AM	OUT_A output (Note 1)
14	13/14/15	—	NFA	A channel output current detection pin (resistor connection). Short the two pins for FG. (Note 1)
15	16	—	PGNDA	Power ground
16	19/20	Output	OUT_AP	OUT_A output (Note 1)
17	23	Output	M0	Initial state detection output. ON when in initial state (open drain).
18	25/26	Input	VMA	Motor side power pin (A phase side) (Note 1)
19	28	Output	Protect	When TSD, ON (open drain). Normal Z.
20	30/31	Input	VDD	Control side power pin. (Note 1)
21	33	Input	CW/CCW	Forward/Reverse toggle pin. L: Forward; H: Reverse (built-in pull-down resistor)
22	35	Input	M2	Excitation mode setting input (built-in pull-down resistor)
23	36	Input	M1	Excitation mode setting input (built-in pull-down resistor)
24	38	Input	DCY2	Current Decay mode setting input (built-in pull-down resistor)
25	39	Input	DCY1	Current Decay mode setting input (built-in pull-down resistor)

HQ: No Non-connection (NC)

FG: Other than the above pins, all are NC

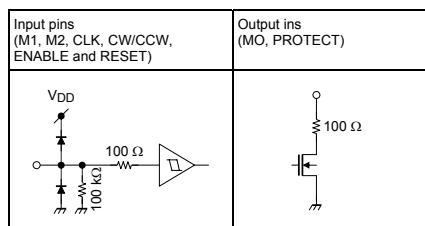
(Since NC pins are not connected to the internal circuit, a potential can be applied to those pins.)

All control input pins: Pull-down resistor 100 kΩ (typ.)

Note 1: If the FG pin number column indicates more than one pin, the indicated pins should be tied to each other at a position as close to the pins as possible.

(The electrical characteristics of the relevant pins in this document refer to those when they are handled in that way.)

&lt;Terminal circuits&gt;



**TOSHIBA**

TB6560HQ/FG

**Absolute Maximum Ratings (Ta = 25°C)**

Characteristic			Symbol	Rating	Unit	
Power supply voltage			V <sub>DD</sub>	6	V	
			V <sub>MA/B</sub>	40		
Output current	Peak	HQ	I <sub>O</sub> (PEAK)	3.5	A/phase	
		FG		2.5		
MO drain current			I <sub>(MO)</sub>	1	mA	
Input voltage			V <sub>IN</sub>	5.5	V	
Power dissipation	P <sub>D</sub>	HQ		5 (Note 1)	W	
		FG		43 (Note 2)		
				1.7 (Note 3)		
				4.2 (Note 4)		
Operating temperature			T <sub>opr</sub>	-30 to 85	°C	
Storage temperature			T <sub>stg</sub>	-55 to 150	°C	

Note 1: Ta = 25°C, No heat sink.

Note 2: Ta = 25°C, with infinite heat sink (HZIP25).

Note 3: Ta = 25°C, with soldered leads.

Note 4: Ta = 25°C, when mounted on the board (4-layer board).  
Susceptible to the board layout and the mounting conditions.**Operating Range (Ta = -20 to 85°C)**

Characteristic			Symbol	Test Condition	Min	Typ.	Max	Unit	
Power supply voltage			V <sub>DD</sub>	—	4.5	5.0	5.5	V	
			V <sub>MA/B</sub>	V <sub>MA/B</sub> ≥ V <sub>DD</sub>	4.5	—	26.4	V	
Output current		HQ	I <sub>OUT</sub>	—	—	—	3	A	
		FG		—	—	—	1.5		
Input voltage			V <sub>IN</sub>	—	0	—	5.5	V	
Clock frequency			f <sub>CLK</sub>	—	—	—	15	kHz	
OSC frequency			f <sub>OSC</sub>	—	—	—	600	kHz	

**TOSHIBA****TB6560HQ/FG****Electrical Characteristics (Ta = 25°C, V<sub>DD</sub> = 5 V, V<sub>M</sub> = 24 V)**

Characteristic	Symbol	Test Circuit	Test Condition	Min	Typ.	Max	Unit
Input voltage	High	1	M1, M2, CW/CCW, CLK, <u>RESET</u> , ENABLE, DECAY, TQ1, TQ2, ISD V <sub>IN</sub> = 5.0 V Built-in pull-down resistor	2.0	—	V <sub>DD</sub>	V
	Low			-0.2	—	0.8	
Input hysteresis voltage	V <sub>H</sub>	1	—	400	—	—	mV
Input current	I <sub>IN</sub> (H)	1	M1, M2, CW/CCW, CLK, <u>RESET</u> , ENABLE, DECAY, TQ1, TQ2, ISD V <sub>IN</sub> = 5.0 V Built-in pull-down resistor	30	55	80	μA
	I <sub>IN</sub> (L)			V <sub>IN</sub> = 0 V	—	—	
Consumption current V <sub>DD</sub> pin	IDD1	1	Output open, <u>RESET</u> : H, ENABLE: H (2, 1-2 phase excitation)	—	3	5	mA
	IDD2		Output open, <u>RESET</u> : H, ENABLE: H (W1-2, 2W1-2 phase excitation)	—	3	5	
	IDD3		RESET : L, ENABLE: L	—	2	5	
	IDD4		RESET : H, ENABLE: L	—	2	5	
Consumption current V <sub>M</sub> pin	I <sub>M1</sub>	1	RESET : H/L, ENABLE: L	—	0.5	1	mA
	I <sub>M2</sub>		RESET : H/L, ENABLE: H	—	0.7	2	
Output channel margin of error	ΔV <sub>O</sub>	—	B/A, C <sub>OOSC</sub> = 0.0033 μF	-5	—	5	%
VNF level Level differential	V <sub>NFH</sub>	—	TQ1 = H, TQ2 = H	10	20	30	%
	V <sub>NFL</sub>		TQ1 = L, TQ2 = H	47	50	55	
	V <sub>NFL</sub>		TQ1 = H, TQ2 = L	70	75	80	
	V <sub>NFL</sub>		TQ1 = L, TQ2 = L	—	—	100	
Minimum clock pulse width	t <sub>W</sub> (CLK)	—	—	—	100	—	ns
MO output residual voltage	V <sub>OL</sub> MO	—	I <sub>OL</sub> = 1 mA	—	—	0.5	V
TSD	TSD	—	(Design target value)	—	170	—	°C
TSD hysteresis	TSDhys	—	(Design target value)	—	20	—	°C
Oscillating frequency	fosc	—	C = 330 pF	60	130	200	kHz

**TOSHIBA**

TB6560HQ/FG

**Electrical Characteristics (Ta = 25°C, V<sub>DD</sub> = 5 V, V<sub>M</sub> = 24 V)****Output Block**

Characteristic			Symbol	Test Circuit	Test Condition		Min	Typ.	Max	Unit
Output ON resistor			Ron U1H	4	I <sub>OUT</sub> = 1.5 A		—	0.3	0.4	Ω
			Ron L1H				—	0.3	0.4	
			Ron U1F		I <sub>OUT</sub> = 1.5 A		—	0.35	0.5	
			Ron L1F				—	0.35	0.5	
2W1-2-phase excitation	W1-2-phase excitation	1-2-phase excitation	Vector	θ = 0	TQ1 = L, TQ2 = L	—	100	—	%	
2W1-2-phase excitation	—	—		θ = 1/8		93	98	100		
2W1-2-phase excitation	W1-2-phase excitation	—		θ = 2/8		87	92	97		
2W1-2-phase excitation	—	—		θ = 3/8		78	83	88		
A-B chopping current (Note)	2W1-2-phase excitation	W1-2-phase excitation	1-2-phase excitation			θ = 4/8	66	71	76	
	2W1-2-phase excitation	—	—			θ = 5/8	51	56	61	
	2W1-2-phase excitation	W1-2-phase excitation	—			θ = 6/8	33	38	43	
	2W1-2-phase excitation	—	—			θ = 7/8	15	20	25	
	2-phase excitation					—	—	100	—	
	Reference voltage			V <sub>NF</sub>	—	TQ1, TQ2 = L (100%) OSC = 100 kHz	450	500	550	mV
	Output transistor switching characteristics			t <sub>r</sub>	7	R <sub>L</sub> = 2 Ω, V <sub>NF</sub> = 0 V, C <sub>L</sub> = 15 pF	—	0.1	—	μs
	t <sub>f</sub>	—	0.1	—						
	Delay time			t <sub>pLH</sub>		RESET to output	—	0.1	—	
	t <sub>pLH</sub>	—	0.3	—						
	t <sub>pHL</sub>	ENABLE to output	—	0.2		—				
	Output leakage current		I <sub>LH</sub>	—		—	1	—	μA	
	Upper side		I <sub>LL</sub>	6	VM = 40 V		—	—	1	—
	Lower side									

Note: Maximum current ( $\theta = 0$ ): 100%

**TOSHIBA****TB6560HQ/FG****Description of Functions****1. Excitation Settings**

You can use the M1 and M2 pin settings to configure four different excitation settings. (The default is 2-phase excitation using the internal pull-down.)

Input		Mode (Excitation)
M2	M1	
L	L	2-phase
L	H	1-2-phase
H	L	W1-2-phase
H	H	2W1-2-phase

**2. Function**

When the ENABLE signal goes Low level, it sets an OFF on the output. The output changes to the Initial mode shown in the table below when the RESET signal goes Low level. In this mode, the status of the CLK and CW/CCW pins are irrelevant.

Input				Output Mode
CLK	CW/CCW	RESET	ENABLE	
	L	H	H	CW
	H	H	H	CCW
X	X	L	H	Initial mode
X	X	X	L	Z

X: Don't care

**3. Initial Mode**

When RESET is used, the phase currents are as follows. In this instance, the MO pin is L (connected to open drain).

Excitation Mode	A Phase Current	B Phase Current
2-phase	100%	-100%
1-2-phase	100%	0%
W1-2-phase	100%	0%
2W1-2-phase	100%	0%

**4. Current Decay Settings**

Output is generated by four PWM blasts; 25% decay is created by inducing decay during the last blast in Fast mode; 50% decay is created by inducing decay during the last two blasts in Fast mode; and 100% decay is created by inducing all four blasts in Fast mode.

If there is no input with the pull-down resistor connection then the setting is Normal.

Dcy2	Dcy1	Current Decay Setting
L	L	Normal 0%
L	H	25% Decay
H	L	50% Decay
H	H	100% Decay

**TOSHIBA****TB6560HQ/FG****5. Torque Settings (Current Value)**

The current ratio used in actual operations is determined in regard to the current setting due to resistance. Configure this for extremely low torque scenarios such as when Weak Excitation mode is stopped. If there is no input with the pull-down resistor connection then the setting is 100% torque.

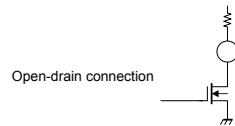
TQ2	TQ1	Current Ratio
L	L	100%
L	H	75%
H	L	50%
H	H	20% (weak excitation)

**6. Protect and MO (Output Pins)**

You can configure settings from the receiving side by using an open-drain connection for the output pins and making the pull-up voltage variable.

When a given pin is in its designated state it will go ON and output at Low level.

Pin State	Protect	MO
Low	Overheat protection operation	Initial state
Z	Normal operation	Other than initial state

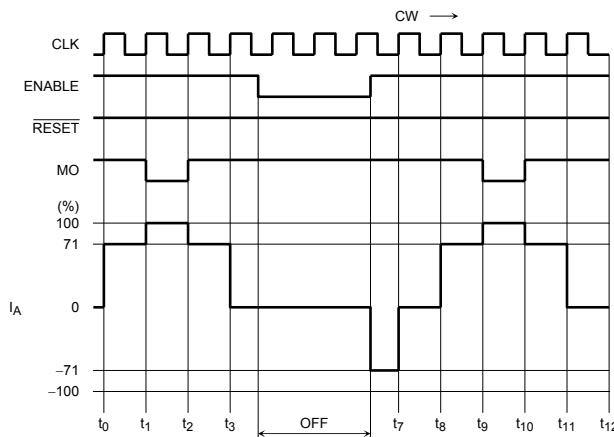
**7. OSC**

Output chopping waves are generated by connecting the condenser and having the CR oscillate. The values are as shown below (roughly:  $\pm 30\%$  margin of error).

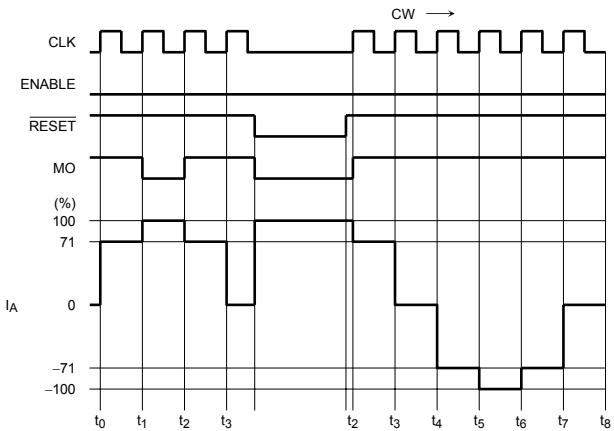
Condenser	Oscillating Frequency
1000 pF	44 kHz
330 pF	130 kHz
100 pF	400 kHz

**TOSHIBA**

TB6560HQ/FG

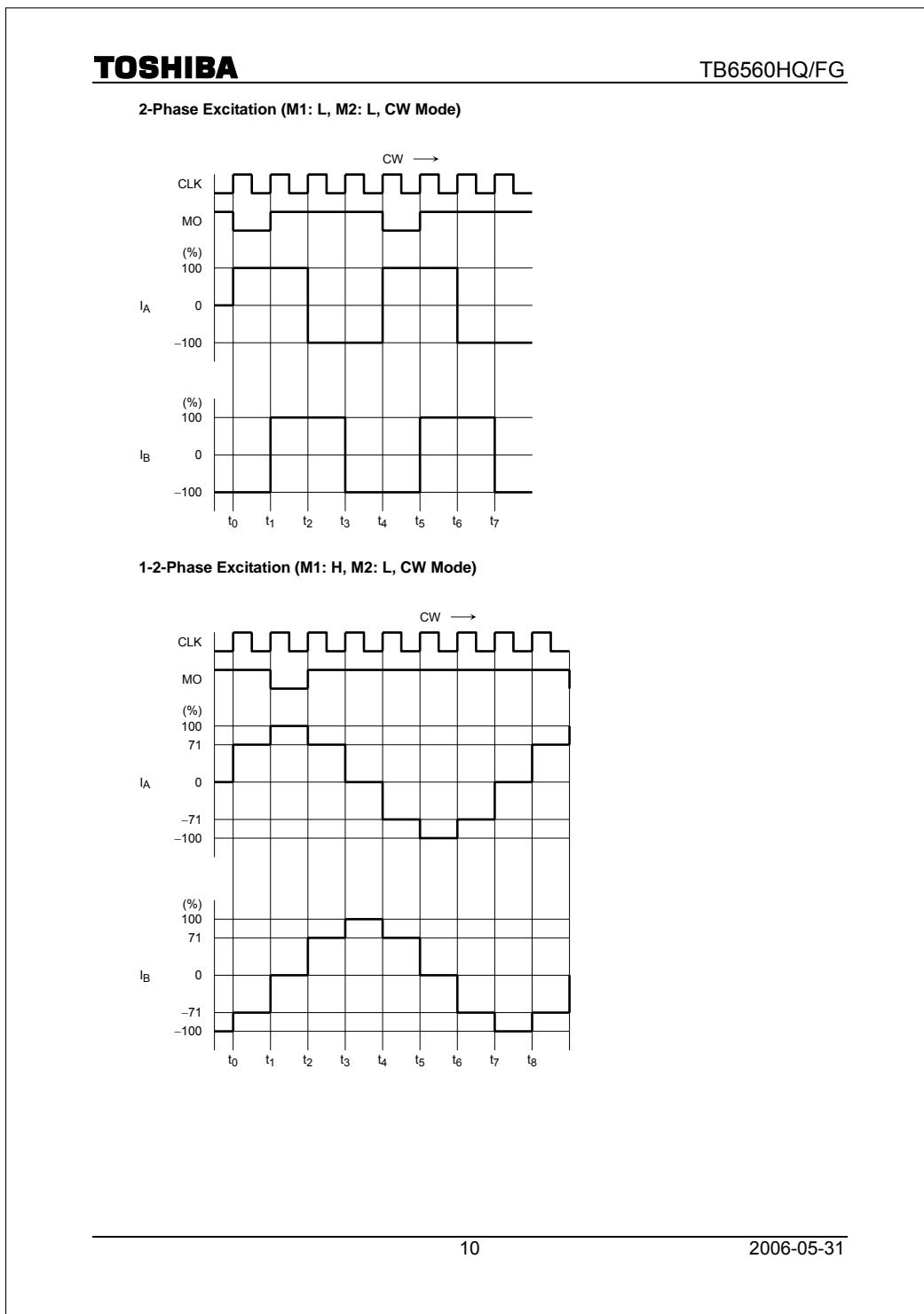
Relationship between Enable, RESET and Output (OUT and MO)**Ex-1: ENABLE 1-2-Phase Excitation (M1: H, M2: L)**

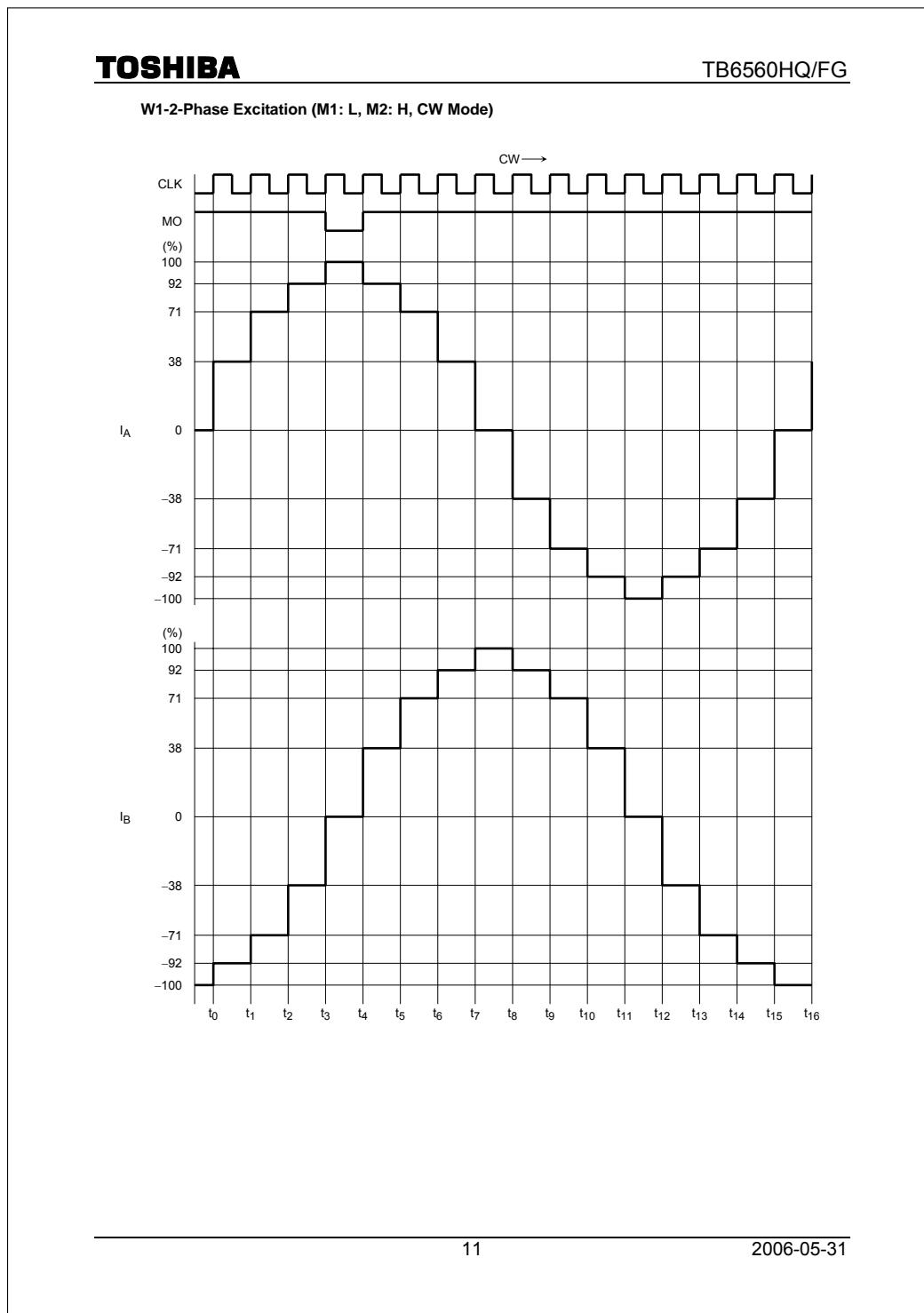
The ENABLE signal at Low level disables only the output signals. Internal logic functions proceed in accordance with input clock signals and without regard to the ENABLE signal. Therefore output current is initiated by the timing of the internal logic circuit after release of disable mode.

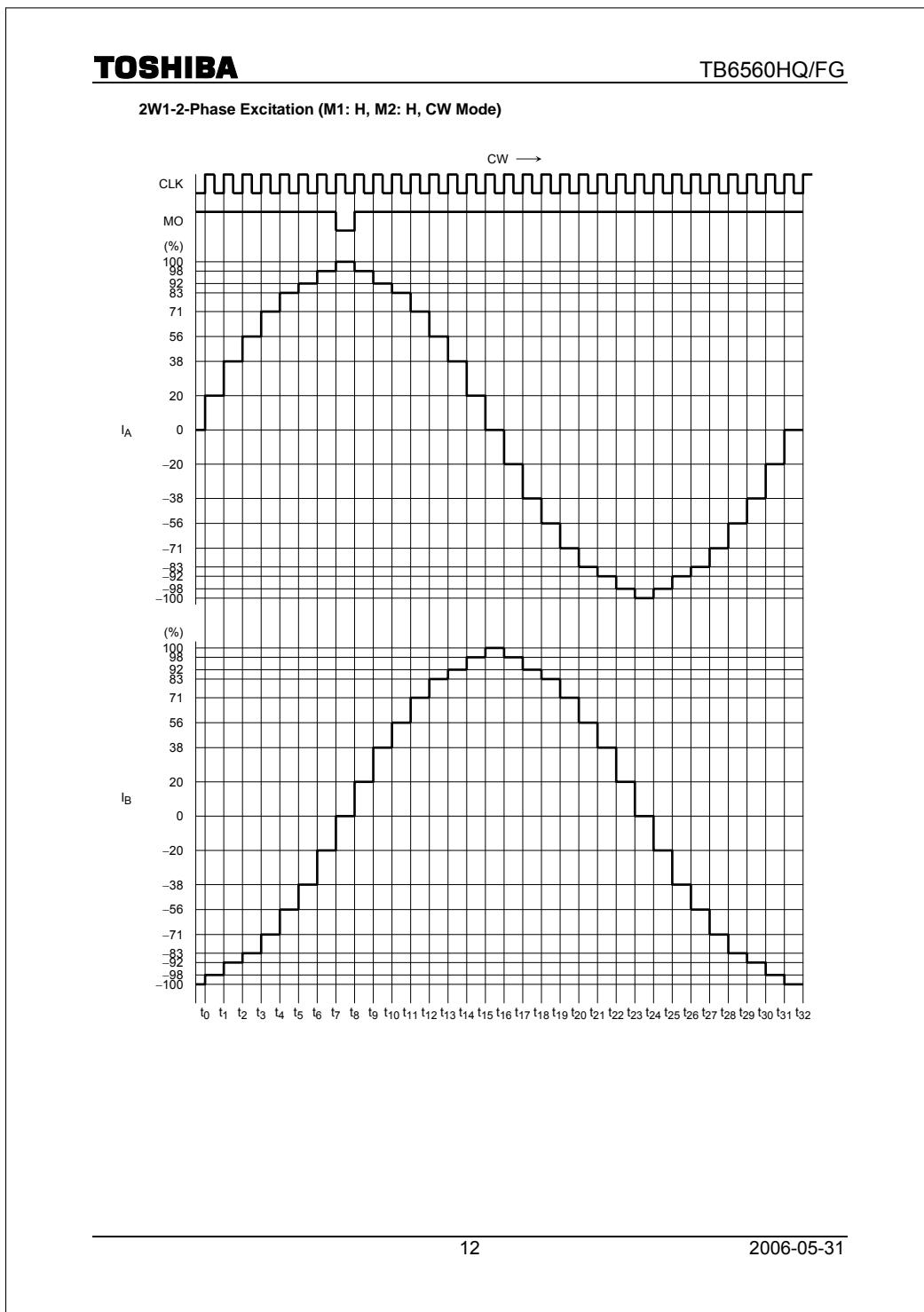
**Ex-2: RESET 1-2-Phase Excitation (M1: H, M2: L)**

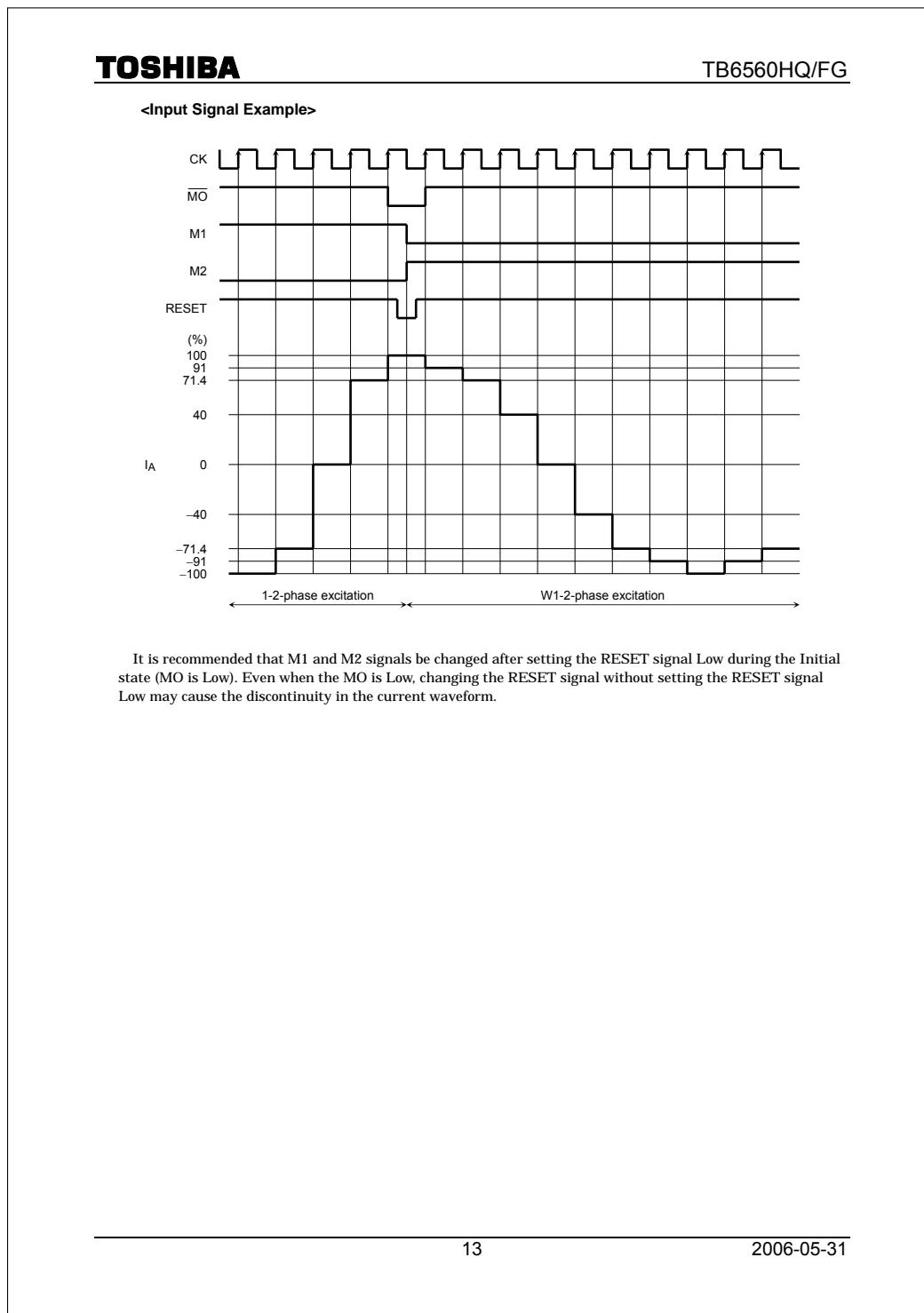
When the RESET signal goes Low level, output goes Initial state and the MO output goes Low level (Initial state: A Channel output current is 100%).

Once the RESET signal returns to High level, output continues from the next state after Initial from the next raise in the Clock signal.









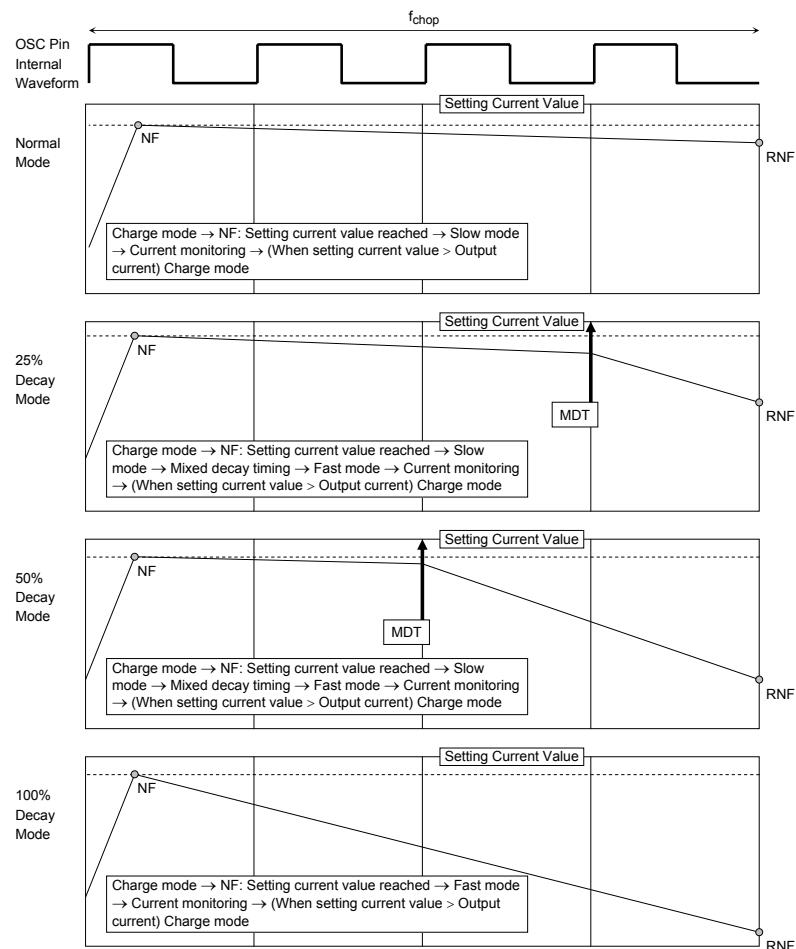
**TOSHIBA****TB6560HQ/FG**

### 1. Current Waveform and Settings of Mixed Decay Mode

You can configure the points of the current's shaped width (current's pulsating flow) using 1-bit input in Decay mode for constant-current control.

"NF" refers to the point at which the output current reaches its setting current value and "RNF" refers to the monitoring timing of the setting current.

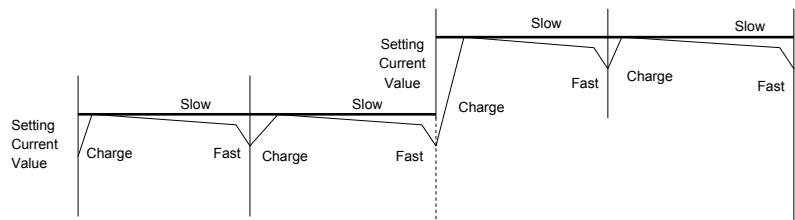
The smaller the MDT value, the smaller the current ripple (current wave peak), and the current's decay capability will fall.



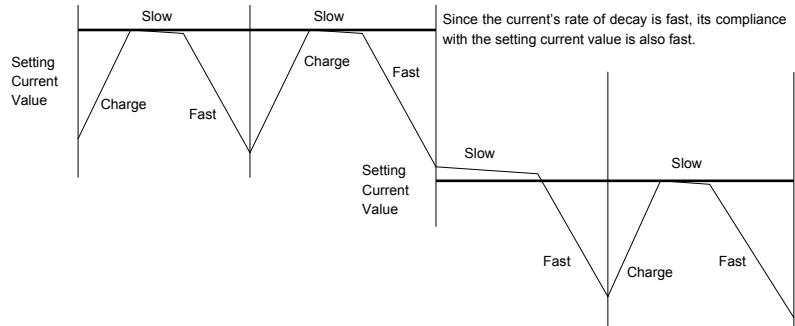
**TOSHIBA****TB6560HQ/FG**

## 2. Current Control Modes (Decay Mode effect)

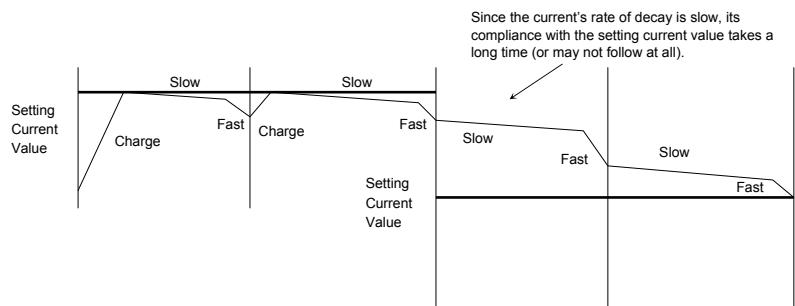
- Direction in which current value increases (sine wave)



- Direction in which sine wave decreases  
(when a high decay ratio (MDT%) is used in Mixed Decay mode)



- Direction in which sine wave decreases  
(when a low decay ratio (MDT%) is used in Mixed Decay mode)

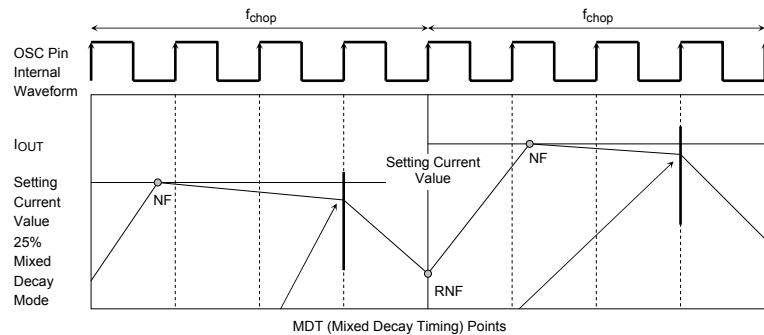


During Mixed Decay mode and Fast Decay mode, if the setting current value < output current at RNF: current monitoring point, the Charge mode at the next chopping cycle will disappear and the pattern will change to Slow, Fast Mode (Slow → Fast occurs at MDT). (In reality, a charge is applied momentarily to confirm the current.)

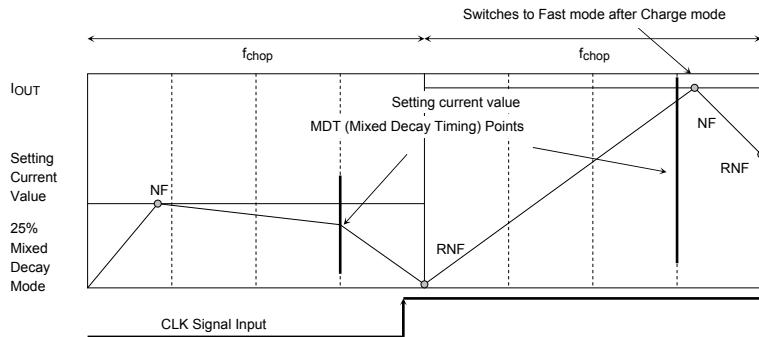
Note: These figures are intended for illustrative purposes only. If designed more realistically, they would show transient response curves.

**TOSHIBA****TB6560HQ/FG**

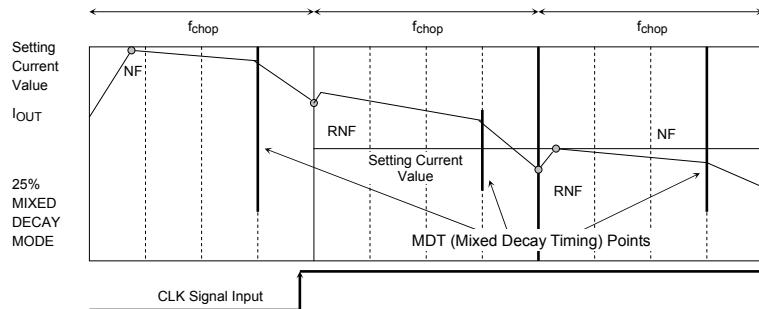
### 3. Mixed Decay Mode Waveform (Current Waveform)



- When the NF points come after mixed decay timing



- When the output current value > Setting current value in mixed decay mode



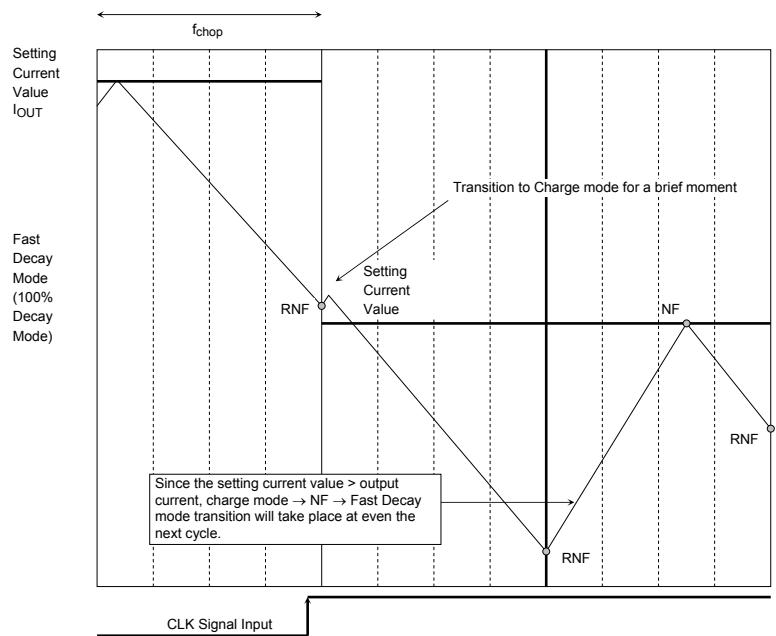
\*: Even if the output current rises above the setting current at the RNF point, a charge is applied momentarily to confirm the current.

**TOSHIBA**

TB6560HQ/FG

#### 4. Fast Decay Mode Waveform

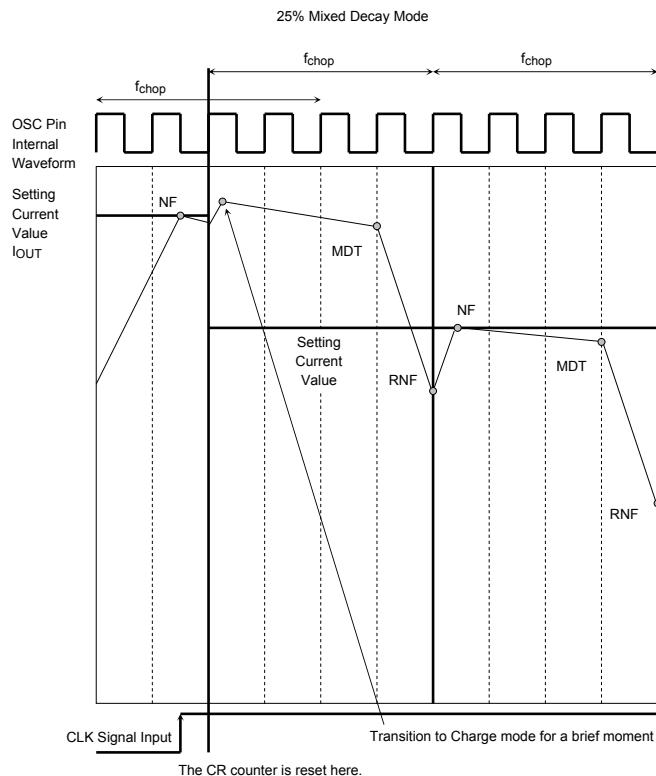
After the current value set by RNF, torque or other means is attained, the output current to load will make the transition to full regenerative mode.



**TOSHIBA**

TB6560HQ/FG

**5. CLK Signal and Internal CR CK Output Current Waveform  
(when the CLK signal is input in the middle of Slow mode)**



When the CLK signal is input, the Chopping Counter (OSC Counter) is forcibly reset at the timing of the OSC.

As a result, the response to input data is fast in comparison to methods that don't reset the counter.

The delay time is one OSC cycle: 10  $\mu$ s @100 kHz Chopping using the Logic Block logic value.

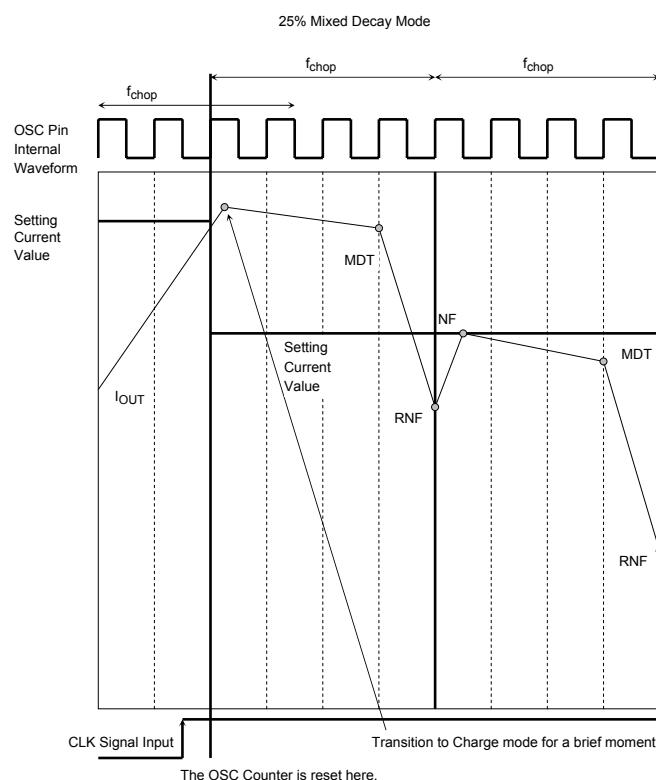
After the OSC Counter is reset by CLK signal input, the transition is invariably made to Charge mode for a brief moment to compare the current.

Note: Even in Fast Decay Mode, the transition is invariably made to Charge mode for a brief moment to compare the current.

**TOSHIBA**

TB6560HQ/FG

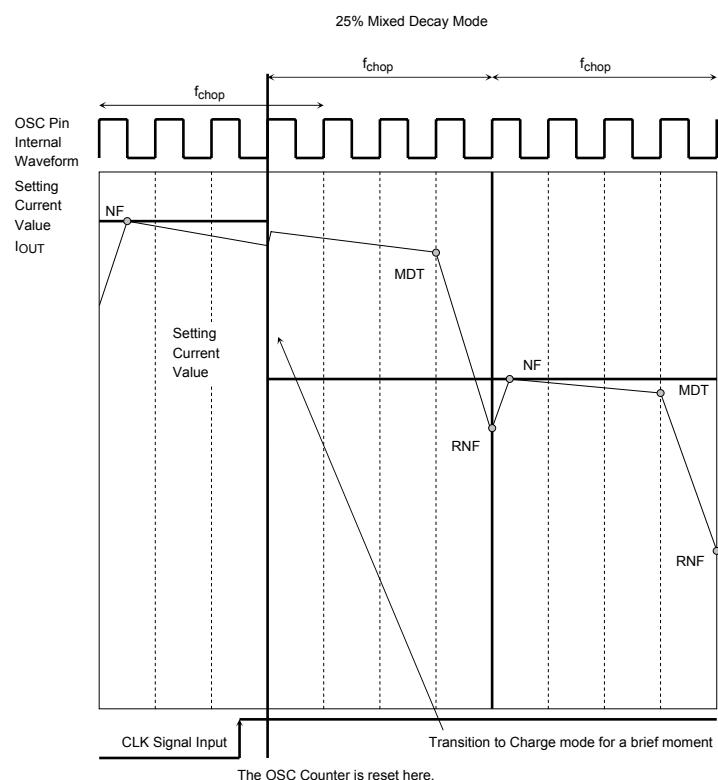
**6. CLK Signal and Internal OSC Output Current Waveform  
(when the CLK signal is input in the middle of Charge mode)**



**TOSHIBA**

TB6560HQ/FG

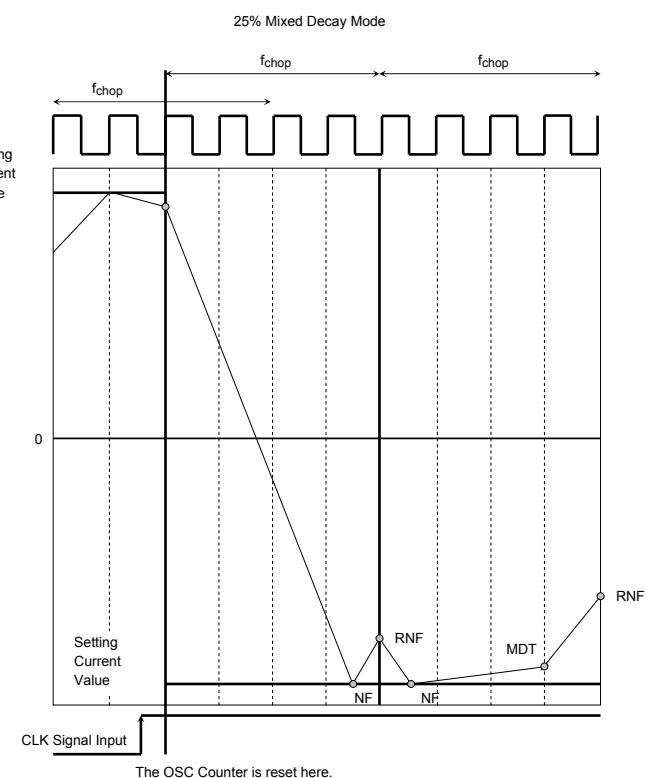
**7. CLK Signal AND Internal OSC Output Current Waveform  
(when the CLK signal is input in the middle of Fast mode)**



**TOSHIBA**

TB6560HQ/FG

**8. Internal OSC Output Current Waveform when Setting Current is Reverse  
(when the CLK signal is input using 2-phase excitation)**



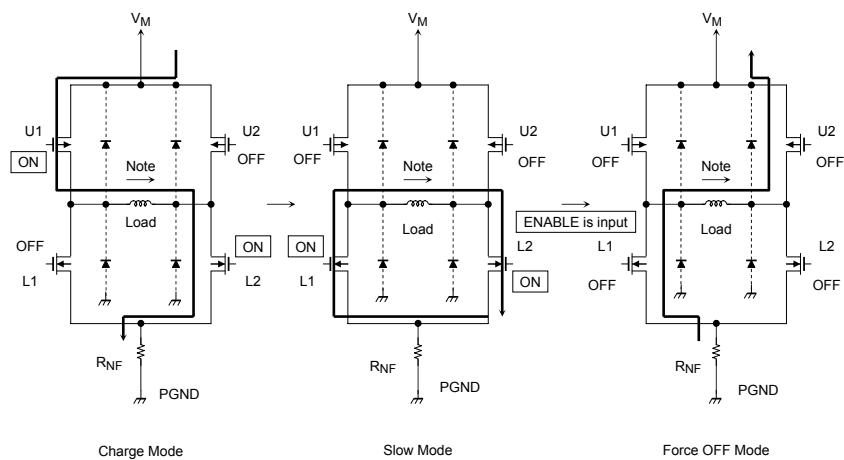
**TOSHIBA**

TB6560HQ/FG

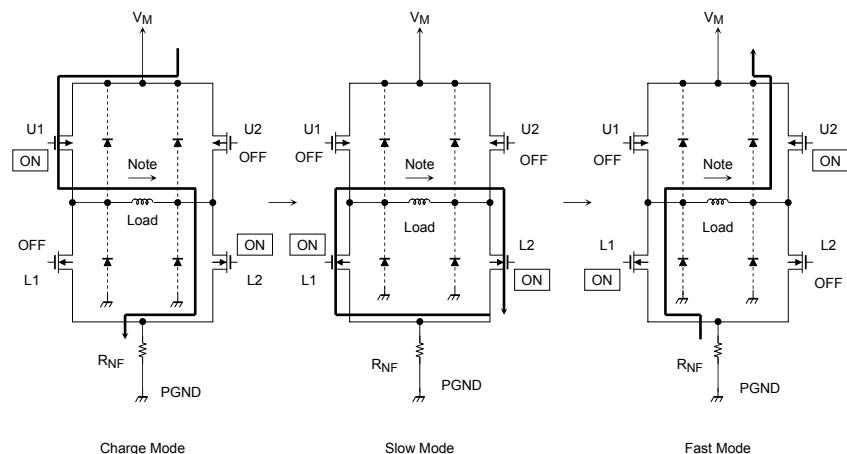
**Current Draw-out Path when ENABLE is Input in Mid Operation**

When all the output transistors are forced OFF during Slow mode, the coil energy is drawn out in the following modes:

Note: Parasitic diodes are indicated on the designed lines. However, these are not normally used in Mixed Decay mode.



As shown in the figure above, an output transistor has parasitic diodes.  
Normally, when the energy of the coil is drawn out, each transistor is turned ON and the power flows in the opposite-to-normal direction; as a result, the parasitic diode is not used. However, when all the output transistors are forced OFF, the coil energy is drawn out via the parasitic diode.

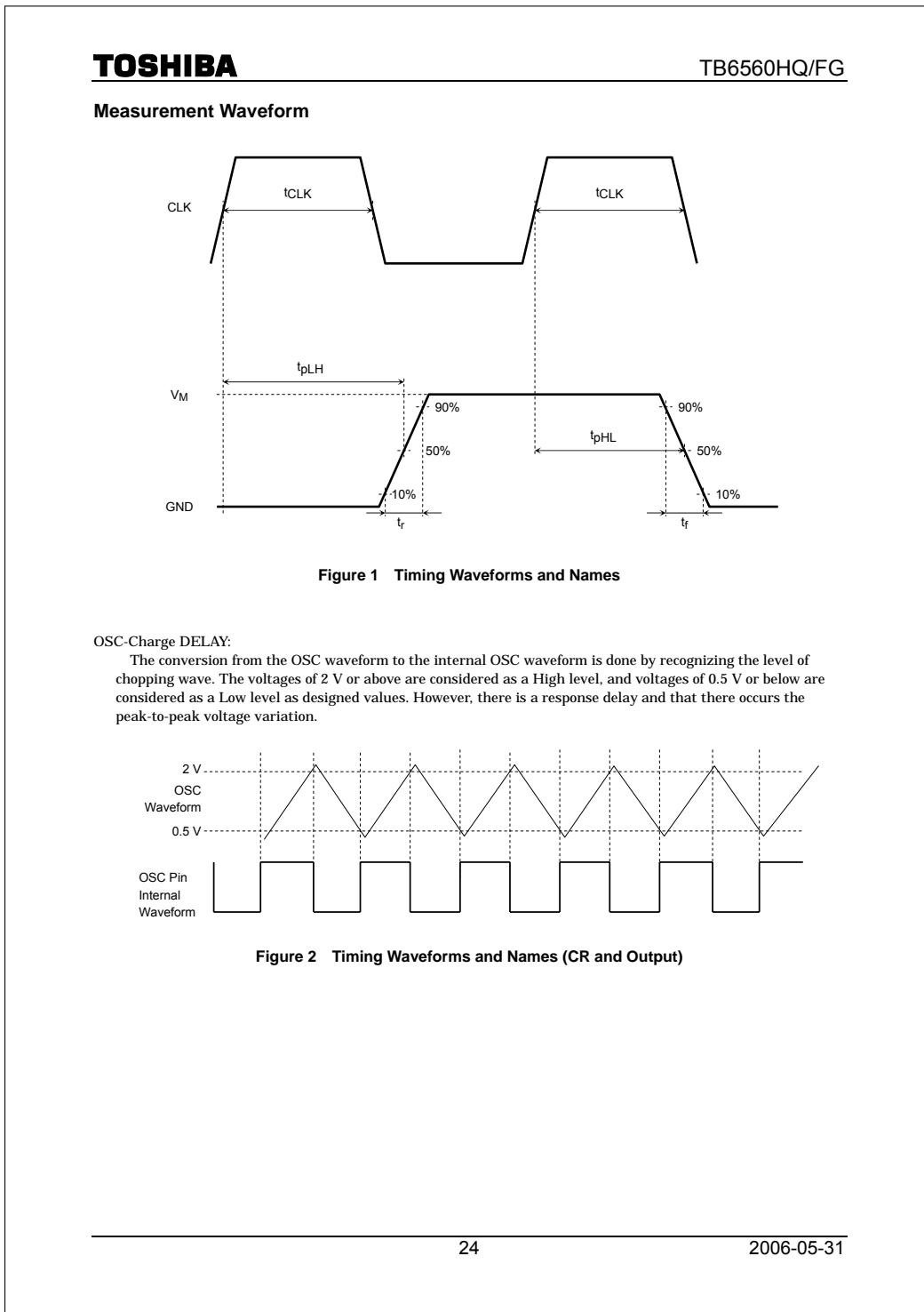
**TOSHIBA****TB6560HQ/FG****Output Stage Transistor Operation Mode****Output Stage Transistor Operation Functions**

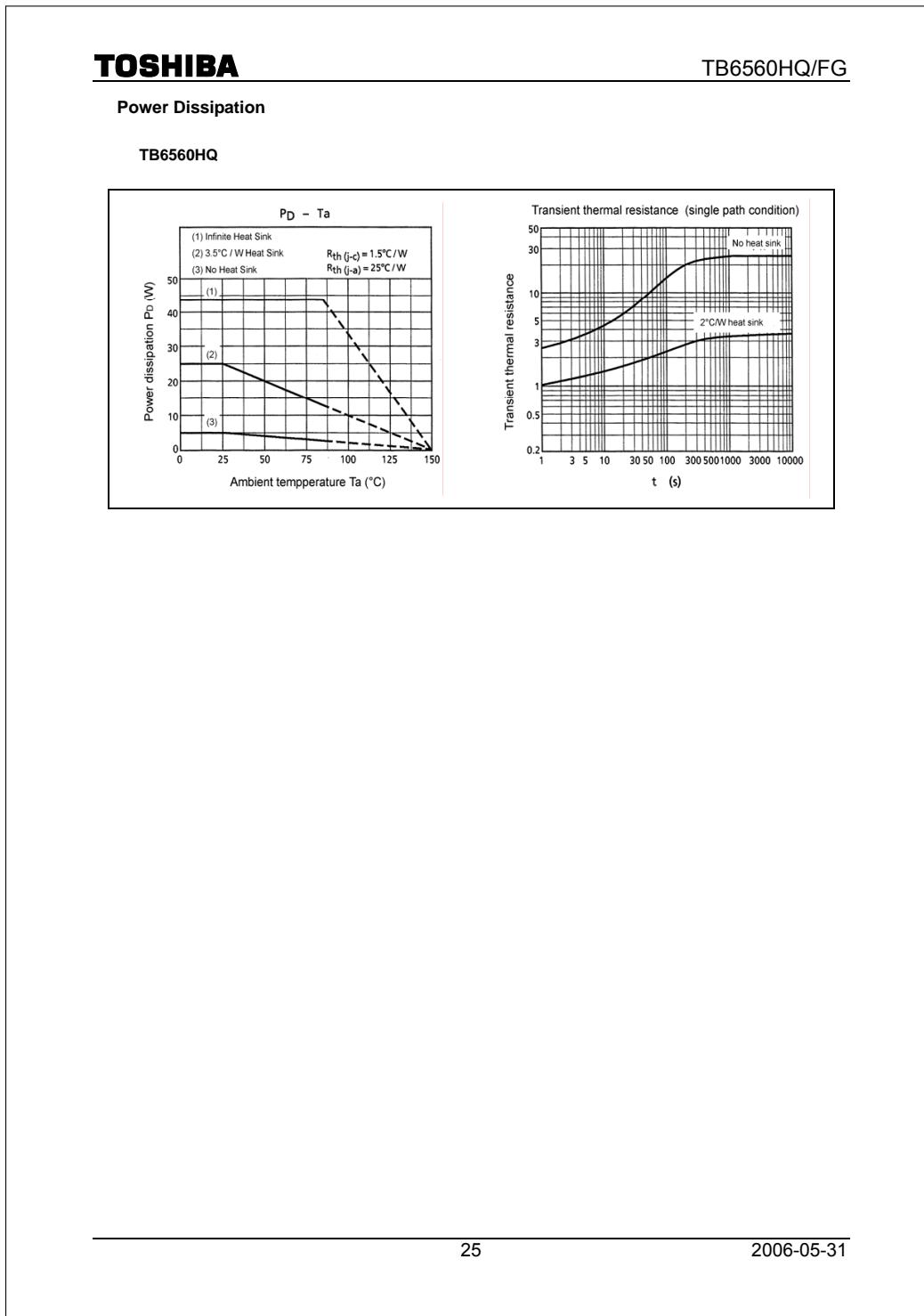
CLK	U1	U2	L1	L2
CHARGE	ON	OFF	OFF	ON
SLOW	OFF	OFF	ON	ON
FAST	OFF	ON	ON	OFF

Note: The above chart shows an example of when the current flows as indicated by the arrows in the above figures.  
If the current flows in the opposite direction, refer to the following chart:

CLK	U1	U2	L1	L2
CHARGE	OFF	ON	ON	OFF
SLOW	OFF	OFF	ON	ON
FAST	ON	OFF	OFF	ON

Upon transitions of above-mentioned functions, a dead time of about 300 ns is inserted respectively.





**TOSHIBA****TB6560HQ/FG****1. How to Turn on the Power**

Turn on V<sub>DD</sub>. When the voltage has stabilized, turn on V<sub>MA/B</sub>.

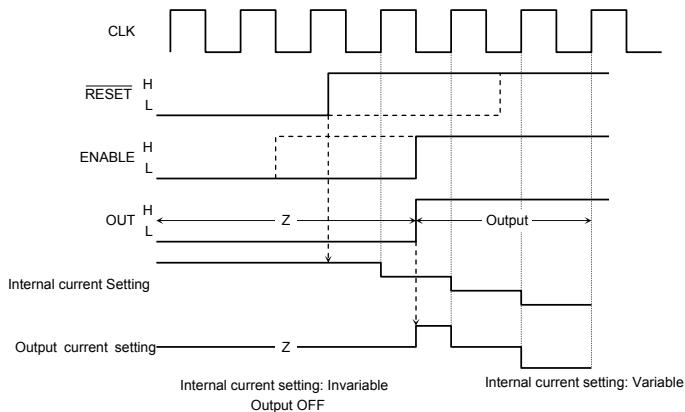
In addition, set the Control Input pins to Low when inputting the power.

(All the Control Input pins are pulled down internally.)

Once the power is on, the CLK signal is received and excitation advances when RESET goes high and excitation is output when ENABLE goes high. If only RESET goes high, excitation won't be output and only the internal counter will advance. Likewise, if only ENABLE goes high, excitation won't advance even if the CLK signal is input and it will remain in the initial state.

The following is an example:

<Recommended Control Input Sequence>

**2. Calculating the Setting Current**

To perform constant-current operations, it is necessary to configure the base current using an external resistor. If the voltage on the NFA (B) pin is 0.5 V (with a torque of 100%) or greater, it will not charge.

Ex.: If the maximum current value is 1 A, the external resistance will be 0.5 W.

**3. PWM Oscillator Frequency (External Condenser Setting)**

An external condenser connected to the OSC pin is used to internally generate a saw tooth waveform. PWM is controlled using this frequency. Toshiba recommends 100 to 3300 pF for the capacitance, taking variations between ICs into consideration.

Approximation:  $f_{osc} = 1/(C_{osc} \times 1.5 \times (10/C_{osc} + 1)/66) \times 1000 \text{ kHz}$

**4. Power Dissipation**

The IC power dissipation is determined by the following equation:

$$P = V_{DD} \times I_{DD} + I_{OUT} \times R_{on} \times 2 \text{ drivers}$$

The higher the ambient temperature, the smaller the power dissipation.

Check the PD-Ta curve, and be sure to design the heat dissipation with a sufficient margin.

**5. Heat Sink Fin Processing**

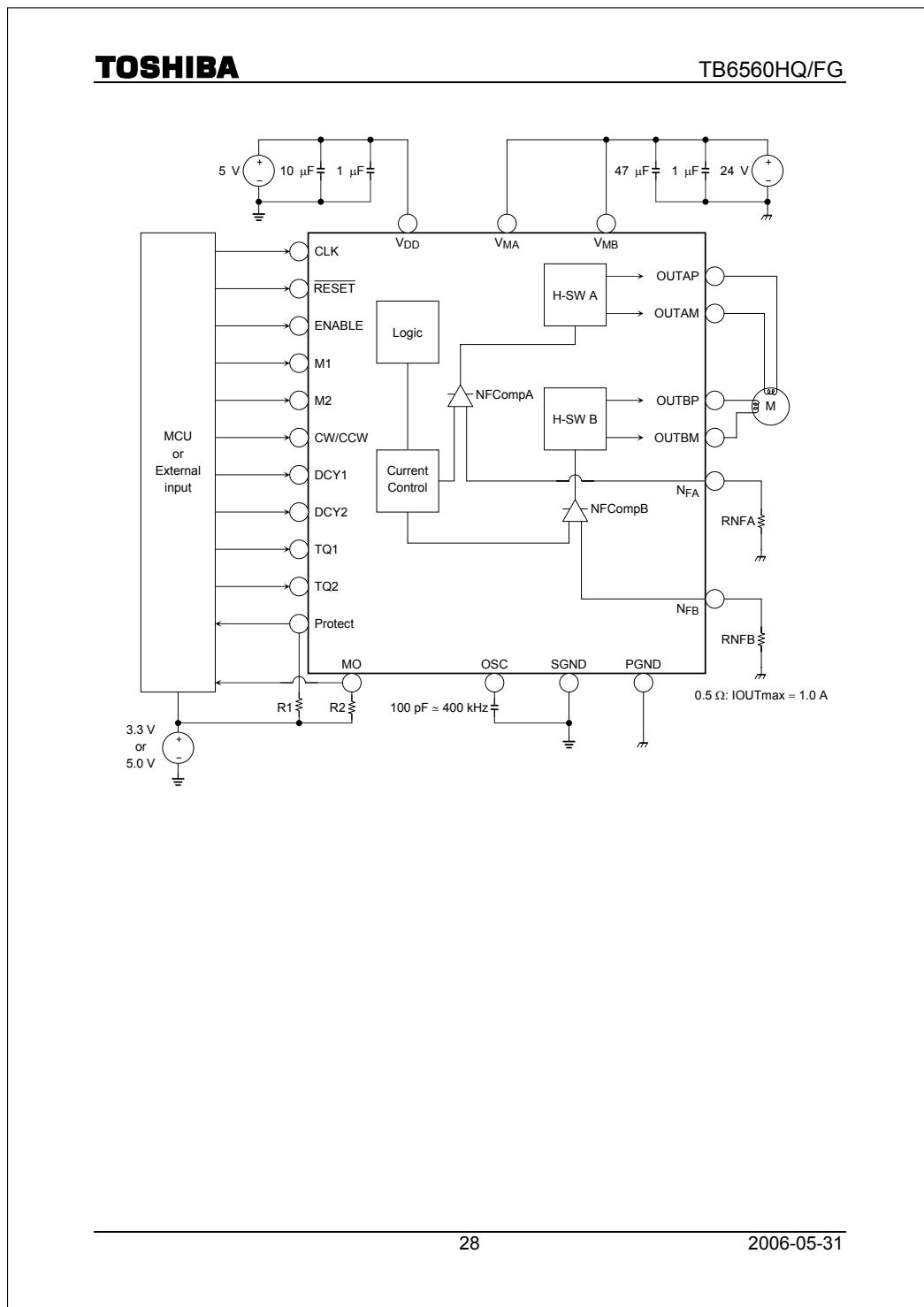
The IC fin (rear) is electrically connected to the rear of the chip. If current flows to the fin, the IC will malfunction. If there is any possibility of a voltage being generated between the IC GND and the fin, either ground the fin or insulate it.

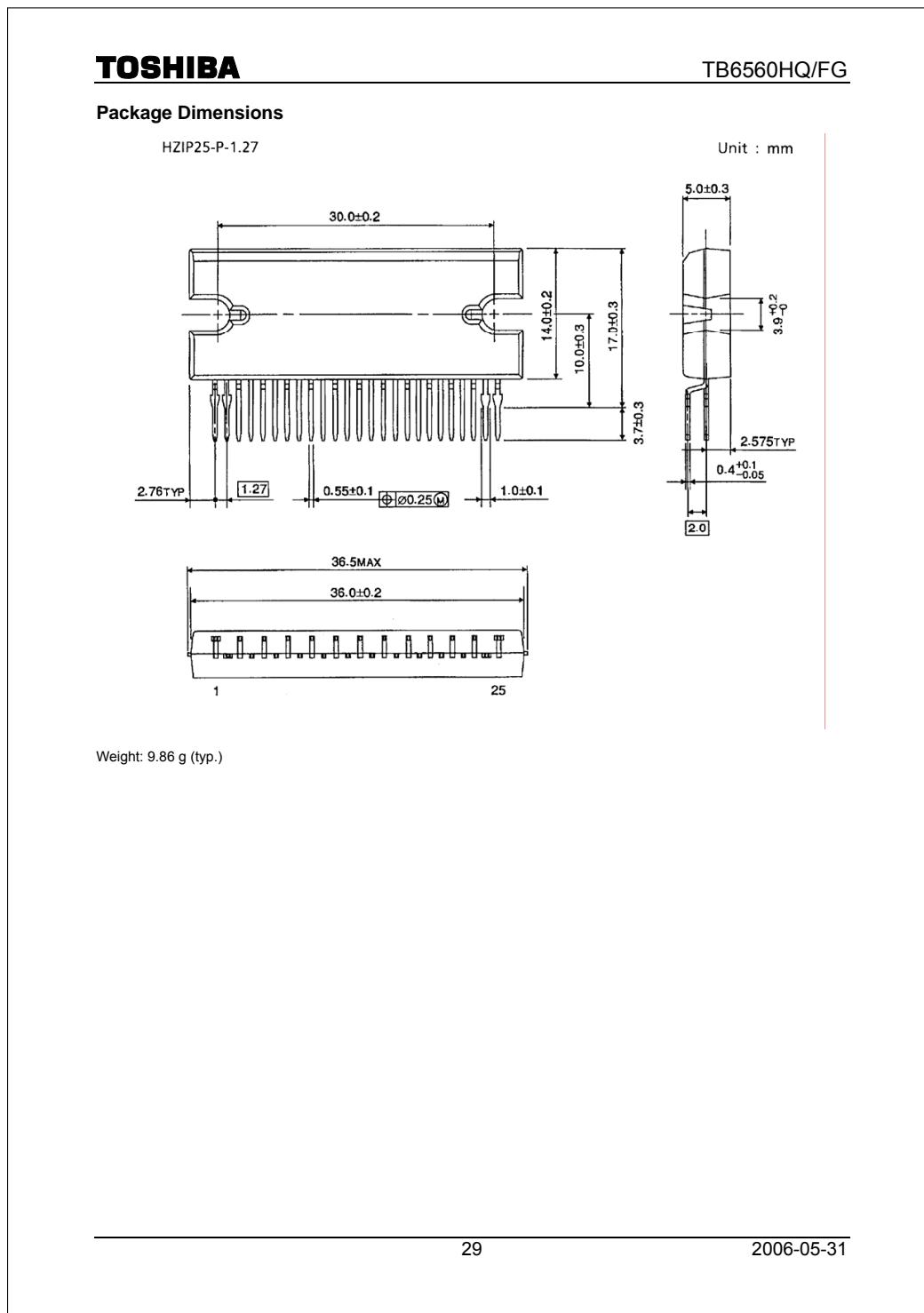
**TOSHIBA**

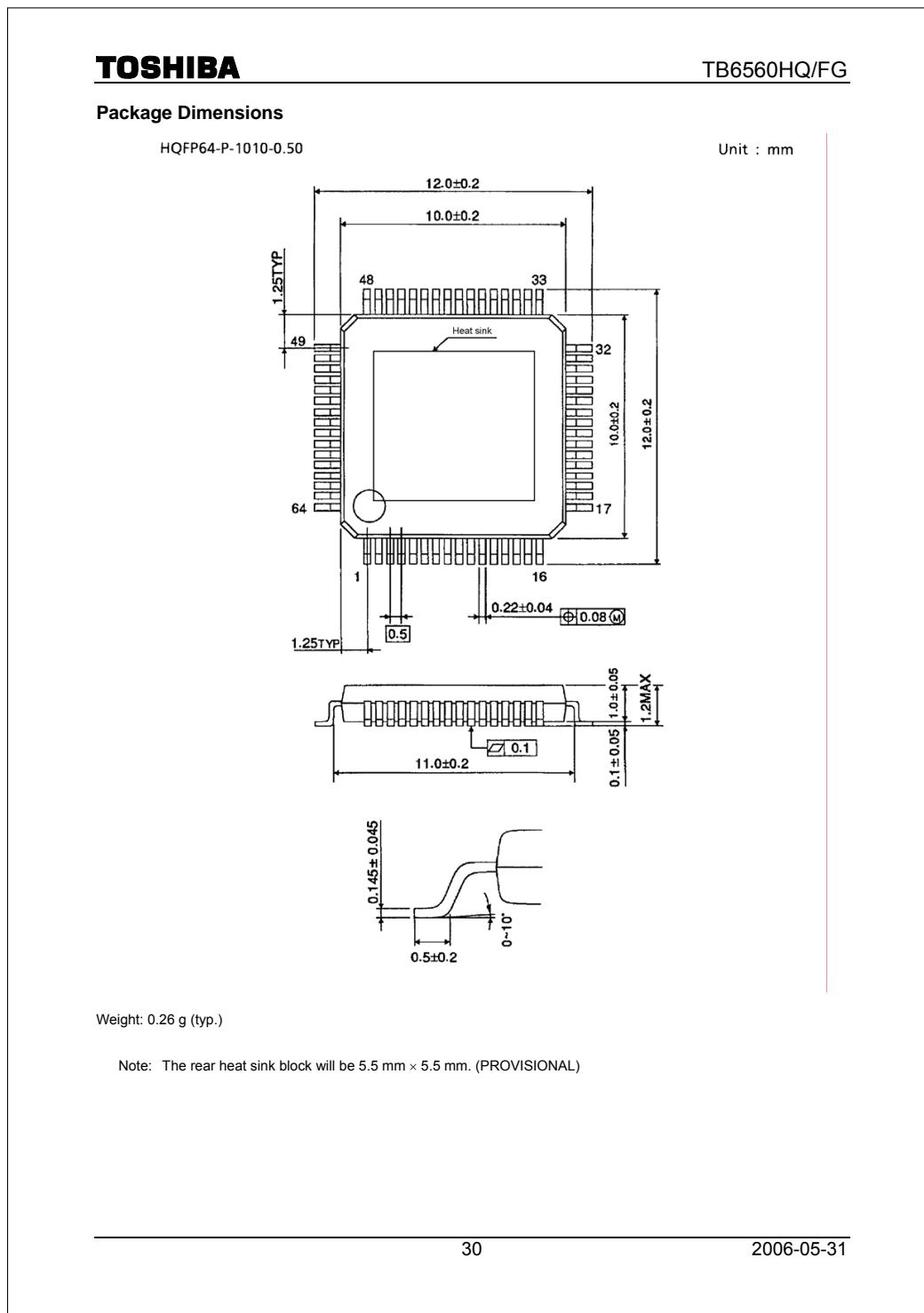
TB6560HQ/FG

**6. Thermal Protection**

When the temperature reaches 170°C (as standard value), the thermal protection circuit is activated switching the output to off. There is a variation of plus or minus about 20°C in the temperature that triggers the circuit operation.







**TOSHIBA**

TB6560HQ/FG

**RESTRICTIONS ON PRODUCT USE**

060116EBA

- The information contained herein is subject to change without notice. 021023\_D
- TOSHIBA is continually working to improve the quality and reliability of its products. Nevertheless, semiconductor devices in general can malfunction or fail due to their inherent electrical sensitivity and vulnerability to physical stress. It is the responsibility of the buyer, when utilizing TOSHIBA products, to comply with the standards of safety in making a safe design for the entire system, and to avoid situations in which a malfunction or failure of such TOSHIBA products could cause loss of human life, bodily injury or damage to property.  
In developing your designs, please ensure that TOSHIBA products are used within specified operating ranges as set forth in the most recent TOSHIBA products specifications. Also, please keep in mind the precautions and conditions set forth in the "Handling Guide for Semiconductor Devices," or "TOSHIBA Semiconductor Reliability Handbook" etc. 021023\_A
- The TOSHIBA products listed in this document are intended for usage in general electronics applications (computer, personal equipment, office equipment, measuring equipment, industrial robotics, domestic appliances, etc.). These TOSHIBA products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury ("Unintended Usage"). Unintended Usage include atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, medical instruments, all types of safety devices, etc. Unintended Usage of TOSHIBA products listed in this document shall be made at the customer's own risk. 021023\_B
- The products described in this document shall not be used or embedded to any downstream products of which manufacture, use and/or sale are prohibited under any applicable laws and regulations. 060106\_Q
- The information contained herein is presented only as a guide for the applications of our products. No responsibility is assumed by TOSHIBA for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of TOSHIBA or others. 021023\_C
- The products described in this document are subject to the foreign exchange and foreign trade laws. 021023\_E

**ANEXO C. MANUAL DE *HARDWARE* DE CONTROL DE MOTORES**

En este apéndice se anexa el manual de *ebay* del *hardware* de control de los motores.

4 Axis TB6560 CNC Stepper Motor Driver Controller Board | eBay <http://www.ebay.com/itm/ws/eBayISAPI.dll?ViewItem&rt=nc&item=2...>

Business & Industrial > Electrical & Test Equipment > Industrial Automation, Control > Drives & Motion Control > Motor Controls > Stepper Controls & Drives

Item: 250922956222



4 Axis TB6560 CNC Stepper Motor Driver Controller Board

marcmart.usa ( 5143 ⭐ ) 

Item condition: New

Time left: 9h 36m 16s (Nov 04, 2011 17:05:51 PDT)

Bid history: 0 bids

Starting bid: US \$9.90

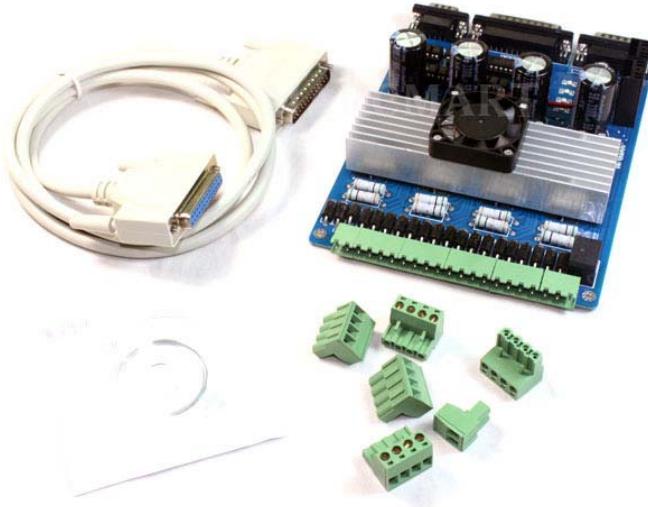
Sell one like this

[Learn more](#)

---

Seller's description

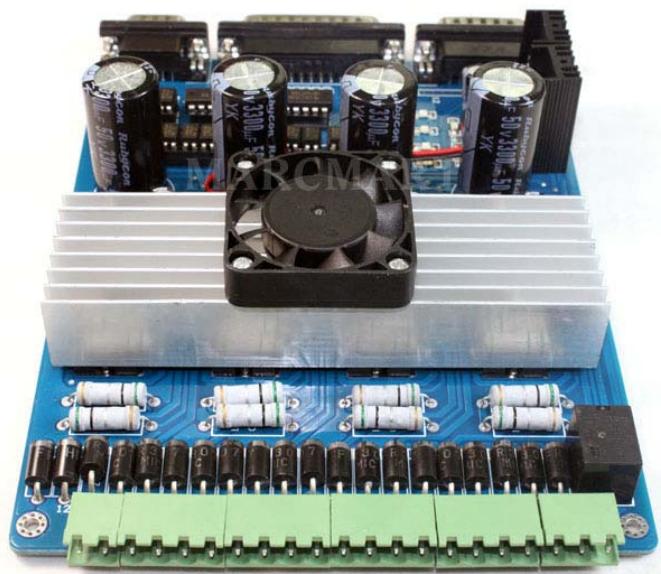
4 Axis TB6560 CNC Stepper Motor Driver Controller Board (UC046)



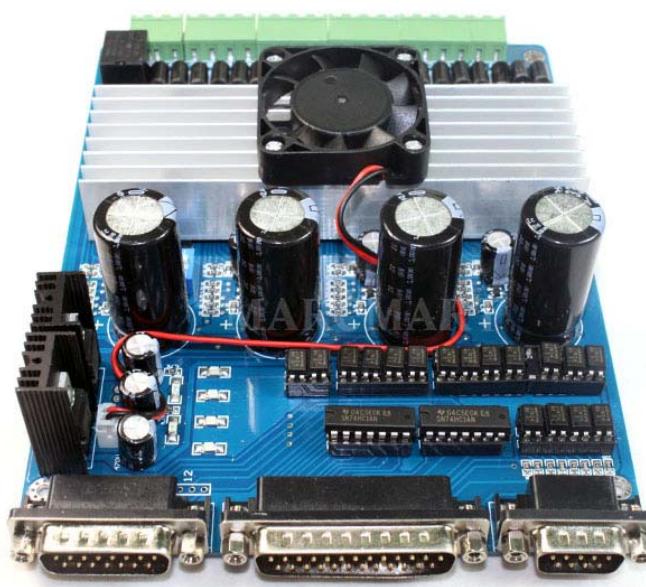
1 de 8 04-11-2011 11:30



4 Axis TB6560 CNC Stepper Motor Driver Controller Board | eBay <http://www.ebay.com/ws/eBayISAPI.dll?ViewItem&rt=nc&item=2...>



4 Axis TB6560 CNC Stepper Motor Driver Controller Board | eBay <http://www.ebay.com/itm/ws/eBayISAPI.dll?ViewItem&rt=nc&item=2...>



#### Description

##### Specification:

Electrical properties (ambient temperature T <sub>j</sub> = 25 °C pm) :	
Input Power	12 - 36V DC power supply
Stepper motor drive current	1.5A - 3A/phase
Drive type	Double-pole constant flow PWM actuation output
Compatible Stepper motors	2 or 4 phase, 4.6 or 8 lead stepper motors, 3A max.
Dimensions	18 * 11 * 4 cm (L*W*H)

##### Product Features:

- Toshiba TB6560AHQ chip - High power, maximum 3.5A drive current chipset !  
[For TB6560AHQ Datasheet, Please click here.](#)
- 1-1/16 microstep setting - Higher accuracy and smoother operation than standard 1, 1/2 step!
- Adjustable 1.5A-3A drive current settings for each axis - 25%, 50%, 75%, 100% of full current can be set for different stepper motors  
\*Attention: This driver is the 1.5A-3A version, it's specially designed for stepper motors with rated current from 1.5A to 3A, it's NOT compatible with stepper motors with rated current below 1.5A or above 3A!  
If the rated current your stepper motors is below 1.5A, you can choose the 0.5A-2.5A version!  
Please let us know which version (1.5A-3A version is default) you need immediately after placing a bid!
- Overload, over-current and over-temperature safety - Full protection for your computer and peripheral equipment !
- On board current switching - Power output can be set according to specific user requirement !
- Full closed-type optical isolation to protect the user's computer and equipment
- Relay spindle interface - Outputs Max. 36V 7.5A for spindle motors or coolant pump (only one device can be powered by this output!)
- 4 channel inputs interface- Can be used for XYZ limit and emergency stop !
- Professional design - Two stage signal processing with super anti-jamming !
- Bipolar constant current chopper drive with non-resonant region - Controls motors smoothly through range without creep effect !
- Four control inputs (divided into pairs of knives) - Allows setting of limit and emergency stop !

4 Axis TB6560 CNC Stepper Motor Driver Controller Board | eBay <http://www.ebay.com/itm/ws/eBayISAPI.dll?ViewItem&rt=nc&item=2...>

- Universal architecture - Supports most parallel software MACH3,KCAM4 etc!
- \*For compatibility with other softwares, please feel free to contact us!

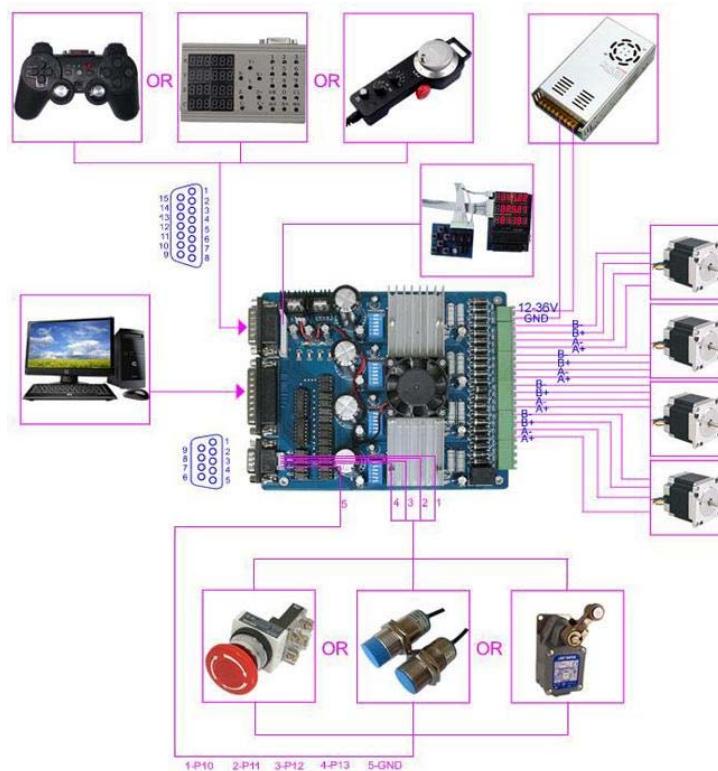
**Dip settings:**

Current Setting	1	2	Decay Mode Settings	3	4	MicroStep Settings	5	6
100%	ON	ON	FAST	ON	ON	1	ON	ON
75%	ON	OFF	25%	ON	OFF	1/2	ON	OFF
50%	OFF	ON	50%	OFF	ON	1/8	OFF	ON
25%	OFF	OFF	SLOW	OFF	OFF	1/16	OFF	OFF

**Further Details:**

- Power supply DC 12-36V (not included, please feel free to contact us if you need !)  
\*This driver get its power from the same unit as the steppers, it doesn't require a separate power source.
- \***Voltage Selection:**  
12-16V DC power supply for Nema 17 stepper motors  
16-24V DC power supply for Nema 23 stepper motors  
24-36V DC power supply for Nema 34 stepper motors  
(High voltage will burn up the chips or stepper motors!!!)
- \***Ampertage Selection:**  
Output current of the power supply can be calculated by the following expressions:  
Output current = Rated current of your stepper motors \* quantity + 2A  
(For example, if you want to drive 3 \* 3A Nema 23 stepper motors, theoretically 24V 11A DC power supply is recommended, but higher power such as 24V 15A also will be good.  
If you are not sure about the selection of power supply, please feel free to contact us for help)
- The power output of 12V shall be applied to the radiator fan of 12V
- Driver output compatible with 2 or 4 phase, 4,6 or 8 lead stepper motors, 3A max.
- Suitable for unipolar or bipolar stepper motors.
- Voltage regulated spindle speed controlled by parallel interface as function of supply voltage.

4 Axis TB6560 CNC Stepper Motor Driver Controller Board | eBay <http://www.ebay.com/itm/ws/eBayISAPI.dll?ViewItem&rt=nc&item=2...>



**Package include:**

- 1 x TB6560 Stepper Motor Driver 1.5A-3A
- 6 x Wire connectors
- 1 x DB25 parallel cable
- 1 x CD software(English Manual & Mach3 demo version)

4 Axis TB6560 CNC Stepper Motor Driver Controller Board | eBay <http://www.ebay.com/itm/ws/eBayISAPI.dll?ViewItem&rt=nc&item=2...>



#### Important Note:

- We aim to achieve a 5 stars rating for each and every transaction.
- If you are not 100% satisfied,**PLEASE** don't leave a neutral or negative but contact us **First** so that we can help you.
- Also please note that due to the recent Air Cargo Security Alerts, there may be delays totally outside our control.

#### Shipping

- Item will be shipped within 3 business days of cleared payment via Post Airmail.
- Only shipped to buyer's PayPal address, please confirm your address on PayPal before you bid.
- Usually takes 7-20 business days to most Countries: US, Canada, UK, Australia and Europe.
- Optionally, Express Shipments with tracking service via EMS/DHL/UPS with a quoted delivery time of 5 to 7 business days is also available .

#### Refund

- It is buyer's responsibility to contact us within 30 calendar days of receiving a defective product. Otherwise, no replacement or refund will be made. All items MUST be returned in their original condition.
- Buyer is responsible for shipping and proof of delivery on all returns.

#### Payment

- We only accept PayPal.
- Payment must be made within 7 calendar days of the auction end time. Non-Paying bidders will be reported.
- We reserve the right to cancel your order if the payment is not made within 8 calendar days of checkout.

powered by  
**PUSHAUCTION**

Your partner for managing your online business

4 Axis TB6560 CNC Stepper Motor Driver Controller Board | eBay <http://www.ebay.com/itm/ws/eBayISAPI.dll?ViewItem&rt=nc&item=2...>

**00000041**

**Questions and answers about this item**

No questions or answers have been posted about this item.

This page is formatted for printing and does not include all the information contained in the listing. You must select all options to print all of the information in the listing including the listing summary, seller's description, and images.

Copyright © 1995-2011 eBay Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this Web site constitutes acceptance of the eBay User Agreement and Privacy Policy.

8 de 8 04-11-2011 11:30

**ANEXO D. MANUAL DE USO**

Aquí se anexa el manual de uso de la máquina CNC completa.



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE  
ESCUELA DE INGENIERÍA  
DEPARTAMENTO DE INGENIERIA ELECTRICA

## **Manual de uso**

### **Fresadora**

Andrés Ananía Iglesias  
12/08/2012

## Índice

1.	Especificaciones técnicas .....	7
1.	Motores: .....	7
2.	Hardware de motores <i>steppers</i> .....	8
a.	Especificaciones .....	8
b.	Conexiones .....	10
3.	Cámaras .....	11
4.	Requisitos de hardware y software .....	11
5.	Limits Switches .....	12
6.	Máquina .....	12
a.	Capacidad de movimientos .....	13
b.	Tornillos y reducciones .....	13
2.	Software .....	14
1.	Software de CNC .....	14
1.1.	<i>Mach3</i> .....	15
2.	Software de diseño 3D .....	37
a.	<i>Solidworks</i> .....	38
3.	Software <i>CAM</i> , ( <i>MasterCam 3D</i> ) .....	43
a.	<i>MasterCam</i> .....	43
4.	Software <i>Fress</i> .....	53
5.	Referencias .....	71
6.	Anexo A: <i>Chessboard</i> de calibración .....	72

**Índice de diagramas**

Diagrama 1: Diagrama de bloques con función “ <i>Start</i> ”.....	32
Diagrama 2: Diagrama de bloques de algoritmo de decisión .....	33
Diagrama 3: Diagrama de bloques con función “ <i>Start from line...</i> ” .....	34
Diagrama 4: Diagrama de bloques con función “ <i>Scale</i> ” .....	34
Diagrama 5: Diagrama de bloques de la función Show para seleccionar filtro de colores .....	63
Diagrama 6: Diagrama de bloques de la función Start para determinar posición y comunicarse con <i>Mach3</i> .....	64

## Índice de figuras

Figura 1: Diagrama de conexión del motor .....	8
Figura 2: Hardware de control de los motores <i>steppers</i> .....	9
Figura 3: Imagen de la máquina .....	13
Figura 4: Primera pantalla <i>Mach3</i> .....	15
Figura 5: Primera pantalla <i>Mach3Mill</i> .....	17
Figura 6: Configuración del puerto paralelo.....	18
Figura 7: Configuración de salida.....	19
Figura 8: Señales de salida .....	19
Figura 9: <i>EStop</i> de entrada .....	20
Figura 10: Señales de entrada .....	20
Figura 11: Configuración de motor.....	21
Figura 12: Configuración de motor en el eje x .....	22
Figura 13: Configuración de motor en el eje y .....	23
Figura 14: Configuración de motor en el eje z .....	23
Figura 15: Comandos de <i>Código G</i> .....	24
Figura 16: Editor de pantallas <i>Mach3</i> , <i>MachScreen</i> .....	26
Figura 17: Editor de pantallas <i>Mach3</i> , <i>MachScreen</i> .....	27
Figura 18: Pantalla creada “ <i>Closed loop (Alt-8)</i> ” .....	28
Figura 19: <i>Edit Button Script</i> .....	29
Figura 20: Resultados del botón <i>Scale</i> .....	36
Figura 21: Comunicación entre <i>Mach3</i> y el software <i>Fress</i> .....	37
Figura 22: Software <i>SolidWorks</i> con un documento <i>solidworks</i> creado.....	39
Figura 23: Normalización del plano.....	40
Figura 24: Plano normalizado y creación de <i>sketch</i> .....	40
Figura 25: Creación de cuadrado 2D. .....	41
Figura 26: Generación de nueva dimensión. .....	41
Figura 27: Creación de nuevo <i>Sketch</i> sobre bloque creado.....	42
Figura 28: Quitar de nuevo material.....	42
Figura 29: Guardado en formato <i>parasolid</i> .....	43

Figura 30: <i>MasterCam</i> con pieza abierta .....	46
Figura 31: Asignación fresa.....	46
Figura 32: Método de creación de maquinado .....	47
Figura 33: Devastado rápido.....	47
Figura 34: Configuración de la fresa y velocidades.....	48
Figura 35: Configuración de corte. ....	49
Figura 36: Configuración de maquinado.....	50
Figura 37: Configuración de planos y posición inicial. ....	51
Figura 38: Simulación de maquinado y creación de <i>Código G</i> .....	52
Figura 39: Prueba de maquinado y figura final. ....	53
Figura 40: <i>Software fress</i> .....	54
Figura 41: Etapa de calibración de colores (Botón “ <i>Show</i> ”) .....	58
Figura 42: Etapa de calibración de distancias (Botón “ <i>Calibration</i> ”) .....	59
Figura 43: Etapa de calibración con respecto a distancias grandes .....	60
Figura 44: Etapa medición de distancia y comunicación constante (Botón “ <i>Start</i> ”)...	61
Figura 45: Generación de trayectorias desde el sensor .....	62

## Introducción

Una fresadora es una máquina que desgasta una pieza metálica en 3 dimensiones para lograr crear una pieza con la forma deseada y de manera totalmente automática. Para esto se le envían los datos de movimiento a través de un computador que posee toda la información de la forma que se desea imitar. Este informe está destinado a documentar acerca del control que se realiza sobre una fresadora.

Este manual está diseñado para enseñar a utilizar y modificar el software de la máquina fresadora que se encuentra actualmente en el departamento de ingeniería eléctrica de la Pontificia Universidad Católica de Chile. Esta máquina será utilizada principalmente por el profesor Dani Guzmán para el diseño de componentes para el área de instrumentación astronómica.

El control de la fresadora se realiza mediante una retroalimentación implementada con cámaras web que indicarán la posición actual de la fresa para así poder corregir los errores mecánicos generados por la máquina. Para lograr esto, se utilizará el software *Mach3* el cual se comunicará con otro *software* realizado para esta memoria que actualizará las posiciones absolutas de la máquina. Este último *software* utilizará el sistema de cámaras para determinar la posición actual de la máquina y mediante un cálculo interno estimará la posición absoluta de la fresadora y se lo enviará al *Mach3* que producirá el cambio de posición.

Este manual está creado de manera que se comenzará con las especificaciones técnicas de la máquina, en que se detallarán las especificaciones de hardware, entradas y salidas, y la mecánica misma de ésta. Siguiendo con los softwares utilizados y creados, especificando las configuraciones necesarias, el modo de empleo de éstos y cómo editarlos para posibles mejoras. Estos programas están categorizados en cuatro partes, primero el software de CNC, segundo el software de diseño, tercero el software de transformación a Código G y por último el software de comunicación que tendrá las lecturas de las cámaras.

## 1. Especificaciones técnicas

### 1. Motores:

La máquina cuenta con tres motores “*Hybrid stepping motor*” Model 160-010-00200.

Sus especificaciones son:

Especificaciones	
Ángulo por paso	1.8 Deg
Voltaje	4.5 V
Corriente	2.5 A/Fase
Resistencia	2 Ohm/Fase
Inductancia	3.6 mH/Fase
Holding torque	180 N.cm
Largo de motor	76 mm
Inercia de rotor	440 g.cm^2
Peso del motor	1.1 Kg
Insulation class	B

Tabla 1: Especificaciones técnicas

La figura 1 muestra el diagrama de conexión de los motores.

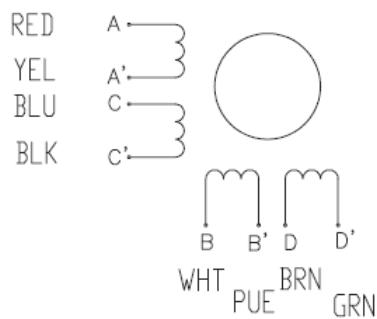


Figura 1: Diagrama de conexión del motor

## 2. Hardware de motores *steppers*

### a. Especificaciones

Para la alimentación y control de los motores *steppers* se utiliza un *hardware* basado en el integrado TB6560, que es capaz de controlar hasta cuatro motores por separado.

El hardware permite controlar los motores de paso a través del computador utilizando el puerto paralelo del ordenador. Además permite que se puedan manejar los motores mediante el software CNC (*Computer Numerical Control*) *Mach3* que le envía la cantidad de pasos y sentido en que se desea que gire cada motor.

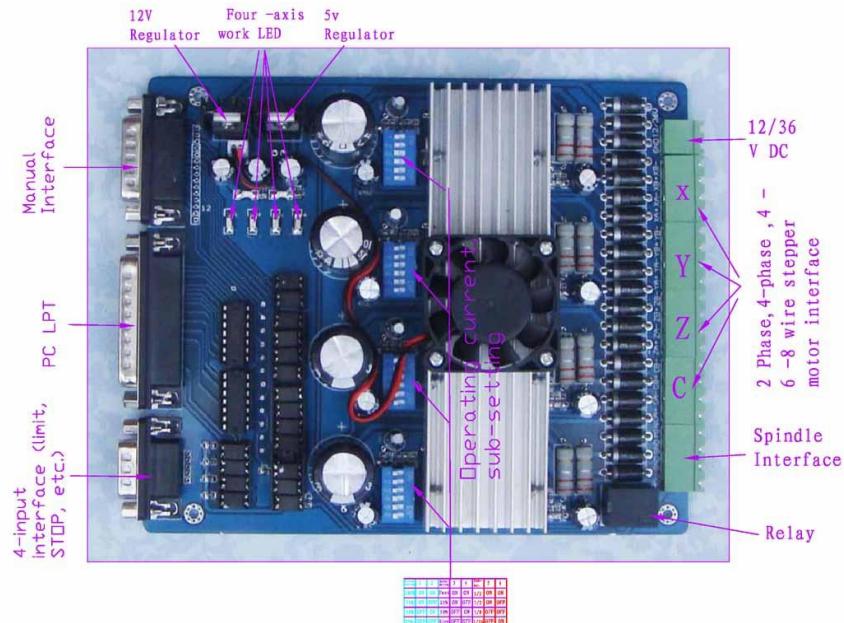


Figura 2: Hardware de control de los motores steppers.

En la figura 2 se muestra una imagen del hardware de control, al lado izquierdo se muestran los inputs, donde la entrada superior corresponde al control manual de los motores (*Manual interface*). El segundo de arriba hacia abajo, es para conectar el puerto paralelo y por último el tercero es para los *Emergency stops* y *Limit switches*.

En el centro están los *Dip switch* para la configuración de cada motor, en que la configuración corresponde a los que se encuentran en la tabla 2.

<b>Current Setting</b>	<b>1</b>	<b>2</b>	<b>Decay Mode Settings</b>	<b>3</b>	<b>4</b>	<b>MicroStep Settings</b>	<b>5</b>	<b>6</b>
100%	ON	ON	FAST	ON	ON	1	ON	ON
75%	ON	OFF	25%	ON	OFF	1/2	ON	OFF
50%	OFF	ON	50%	OFF	ON	1/8	OFF	OFF
25%	OFF	OFF	SLOW	OFF	OFF	1/16	OFF	ON

Tabla 2: Configuración de los motores.

Actualmente esta configuración está en un 75% de *Current Setting*, ya que para valores mayores a este, los motores se comienzan a sobrecalentar. Un *microstep* de 1 ya que para valores menores a este los motores pierden pasos muy fácilmente por falta de torque, y el *Decay Mode Settings* se dejó como estaba, con esta configuración se logró un buen comportamiento de la máquina pero puede variar dependiendo del mantenimiento de la máquina y de la aplicación que se desee diseñar.

Finalmente a la derecha del hardware se encuentran los outputs de la tarjeta que van hacia los motores. De arriba hacia abajo, primero está la alimentación de energía que puede ser entre 12 y 36 V. Las cuatro salidas siguientes corresponden a cuatro ejes (en este caso sólo tres) y por último una salida para el *Spindle*, el cual no será controlado a través del *Mach3* debido a que la máquina no posee una entrada para controlar este parámetro.

#### b. Conexiones

Por último las conexiones se realizan de la siguiente manera especificando los pines.

Definición del Puerto paralelo, entradas del 1 al 25.

PIN2	PIN4	PIN1	PIN16	PIN17	PIN7	PIN14	PIN5	PIN6	PIN3	PIN5	PIN8
Spindle motor	X Enable	X Dir	X Step	Y Enable	Y Dir	Y Step	Z Enable	Z Dir	Z Step	C Enable	C Dir

Tabla 3: Pines del puerto paralelo

Definición del *Manual Interface* en los pines del 1 al 15

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
Z/C Enable	C Step	Z Step	X Dir	X Enable	Y Enable	Y Dir	Z Dir	C Dir	Spindl e motor	Y Step	X Step	STOP	GND	5V/vdd

Tabla 4: Pines de la salida de interface manual

Definición del DB9 para los botones de emergencia

P1	P2	P3	P4	P5	P6	P7	P8	P9
X Limit	Y Limit	Z Limit	STOP	Empty	GND	GND	GND	GND
P10	P11	P12	P13					

Tabla 5: Entrada de los pines de los *limits switches* y el *emergency stop*

Input 1	Input 2	Input 3	Input 4
Corresponding P10	Corresponding P11	Corresponding P12	Corresponding P13

Tabla 6: Auxiliares de pines

### 3. Cámaras

El sistema cuenta con dos cámaras Logitech c110 que servirán para medir las posiciones. Para que el sistema funcione será necesario que ambas se encuentren conectadas. Si se desea cambiar las cámaras, será necesario tener en cuenta que el requisito es que posean una resolución mayor o igual a 1024 x 768 y que el driver permita controlar los parámetros de *White Balance* y *Exposition*.

### 4. Requisitos de hardware y software

Los requerimientos de hardware y software mínimos serán los siguientes.

- Windows XP (actualizado con el último *Service Pack*) con DirectX 9.0c, y Visual Studio instalado.
- Procesador 2.6 GHz Pentium IV o equivalente AMD Athlon.
- 2 GB de espacio libre en el disco duro.
- 1 GB RAM en XP, 1.5 GB

El sistema operativo puede variar pero será necesario volver a compilar el código del software *Fress*.

- Dos cámaras web con la posibilidad de calibrar sus parámetros internos de exposición y White balance.

## 5. Limit Switches

La máquina actualmente cuenta con seis *limit switches* (dos por cada eje, uno a la izquierda y otro a la derecha) de fin de carrera para marcar los límites hasta dónde puede llegar la máquina y detenerse automáticamente si se llega a los topes.

Además cuenta con un *emergency stop* que al ser presionado detiene la máquina instantáneamente para cualquier emergencia, es un botón rojo pequeño que se encuentra al lado de la máquina.

## 6. Máquina

La máquina consiste en una fresadora Clarke Metalworker modelo CMD10 con las especificaciones que se presentan en la tabla 7.

Clarke MetalWorker	
Model	CMD10
Part Nº	7610850
Potencia	150 Watts
Corriente	5 A
Alto sin carga	100-2000 RPM
Bajo sin carga	100-1000 RPM

Tabla 7: Especificaciones de la máquina

Las corrientes, cargas y potencias son respecto al taladro de la máquina. La figura 3 muestra una imagen de esta máquina.



Figura 3: Imagen de la máquina

**a. Capacidad de movimientos**

Debido a las capacidades de la máquina se posee un límite utilizable de movilidad de 13 cm máximo para el eje X, 12 cm máximo para el eje Y, mientras que en el eje z es de 2 cm máximo. El eje Z posee otro grado de libertad que es controlado manualmente y genera movilidad mayor a 40 cm.

**b. Tornillos y reducciones**

Los tornillos poseen un movimiento de 80 ticks/vuelta, y por cada tick se mueve una distancia de 0.025 mm para los ejes X e Y. Teniendo en consideración que el motor posee 200 pasos se tiene una resolución máxima de 0.025.80/200mm, es decir, 0.01mm. El eje Z posee 40 ticks y 0.05 mm por tick, es decir, los mismos 0.01 mm de resolución máxima.

## 2. Software

Para el maquinado de una pieza, desde la creación computacional hasta el fresado de ésta, se realiza un proceso en el que se utilizarán cuatro softwares: *SolidWorks* para la creación de las piezas 3D; *MasterCam* para la transformación de una forma 3D en *Código G*; *Fress* para la lectura de la posición a través de las cámaras web, y *Mach3* para el procesamiento del Código G. A continuación se detallará el uso y configuración de cada software para lograr realizar un maquinado adecuado. Además se agregan algunas secciones para personalización del *Mach3*. Para mayores detalles es recomendable consultar las páginas que se encuentran en las referencias al final de este documento.

### 1. Software de CNC

Para la comunicación entre el *Código G* y la máquina, se utiliza un *software CNC* (*Computer Numerical Control*) para el que existe un sinfín de *software* utilizables con este propósito y cada uno de estos *software* posee su propio sistema operativo, dentro los más utilizados son el *Mach3* (pagado) para Windows y el *EMC2* (gratuito) para Linux. Ambos programas son para control en lazo abierto. En este caso nos concentraremos únicamente en el *Mach3* ya que el computador que se utiliza cuenta con Windows y además posee este software. El *Mach3* también cuenta con muchas características que lo hacen ser una buena alternativa, por ejemplo, es posible generar *Scripts* en VB con el cual se puede realizar la comunicación con otros *softwares* y que también permitirá crear el lazo de control cerrado.

### 1.1. Mach3

La implementación será desarrollada sobre la opción *Mill* del *Mach3* por lo que se deberá abrir esta opción del programa o seleccionar la opción *Mach3Mill* al iniciar *Mach3loader.exe*. Para profundizar en detalle revise el manual del *mach3* [1] *Mach3 CNC Controller Software Installation and Configuration* y [2] *Using Mach3* que se encuentran en la web de la página oficial del *Mach3* en la referencia [1].

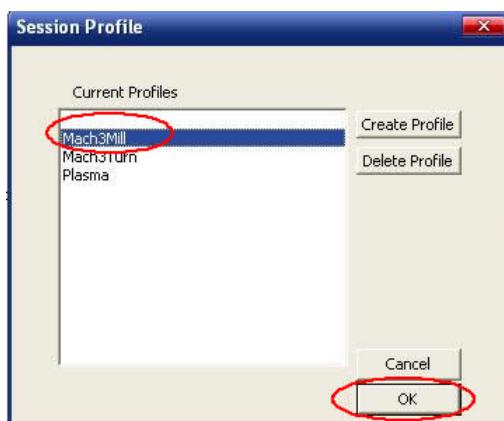


Figura 4: Primera pantalla *Mach3*

#### a. Instalación

Junto a este manual se encuentra anexado un CD que posee el software del *Mach3* en la carpeta “/Mach3\_3.042”. Dentro de esta carpeta se encuentra el archivo “Mach3Version3.042.020.exe”, que debe ser ejecutado e instalado siguiendo las instrucciones de instalación dentro de la raíz del disco en la carpeta *Mach3* (Ejemplo. C:\Mach3\). Finalmente para poder utilizar el software libremente será necesario agregar el archivo de licencia “Mach1Lic.dat” dentro de la carpeta del *Mach3* (C:\Mach3\), con esto debiese quedar listo para ser utilizado en su versión original.

Luego será necesario adaptarlo para que pueda trabajar con el sistema de lazo cerrado. Para esta adaptación será necesario pegar los archivos que se encuentran dentro de la carpeta “/Mach3\_3.042/Mach3/” en la carpeta raíz del *Mach3* (Ej. [C:/Mach3/](#)) y remplazar los existentes. Estos serán los archivos de comunicación, datos y calibración de las cámaras. Junto con el traspaso anterior se anexaron las configuraciones actuales de la máquina y algunos archivos necesarios para el software *Fress*.

#### **b. Configuración**

##### **i. Configuración de entrada**

La configuración se debería encontrar lista ya que es guardada en el archivo “Mach3Mill.xml” y se encuentra en el disco anexado en la memoria en “/Mach3\_3.042/Mach3/” y debió haber sido copiado en la sección 2.1.a.

Para configurar todos los pines de entradas y salidas desde la configuración de los pins y puertos desde “*Config -> Ports and Pins*” en el *Mach3* tal como se muestra en la figura 5, se deberán seguir los pasos que se encuentran en las imágenes siguientes.

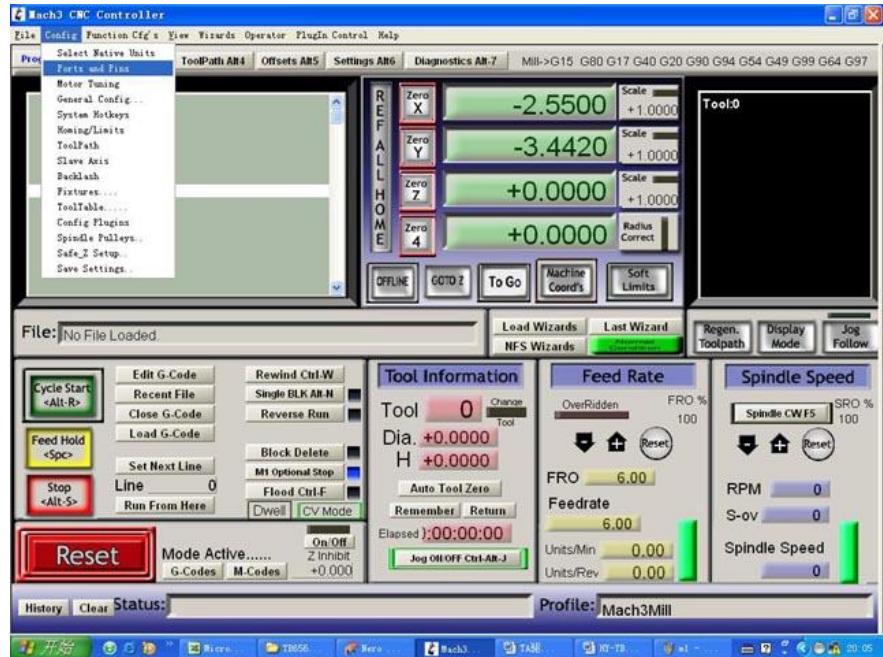


Figura 5: Primera pantalla Mach3Mill

En la primera pantalla se puede configurar el puerto paralelo, para esto es necesario configurarlo de la manera que indica la figura 6, es decir, 35000 Hz, la dirección del puerto paralelo (se puede ver desde *Hardware* de sistema usualmente es 0x378).

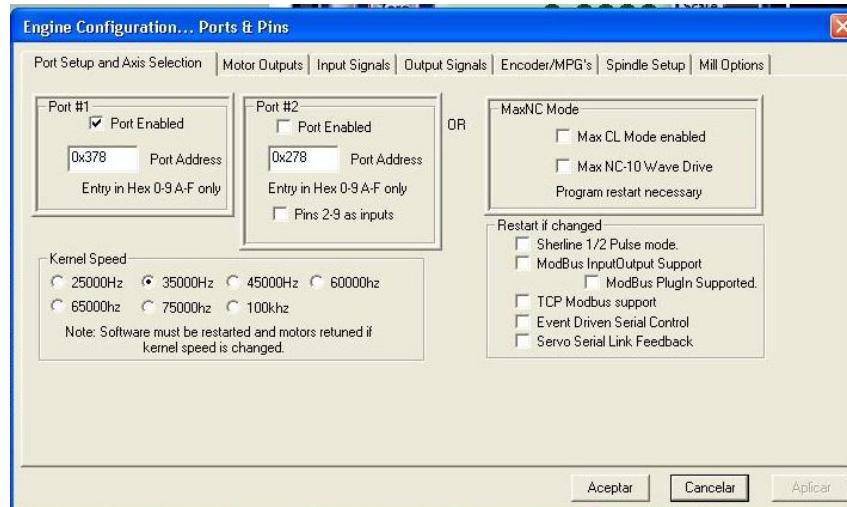


Figura 6: Configuración del puerto paralelo

La figura 7 muestra la configuración necesaria de los pines de salida. Estos pines corresponden a los que van hacia el motor y con este orden van directamente a cada eje de la máquina. Además las figuras 8, 9, 10 muestran las configuraciones que serán necesarias para configurar los pines de salida, el *EStop* de emergencia y las señales de entrada.

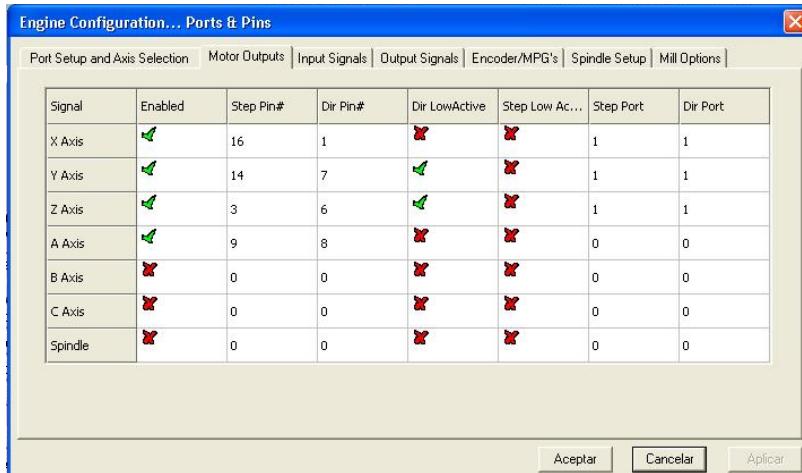


Figura 7: Configuración de salida

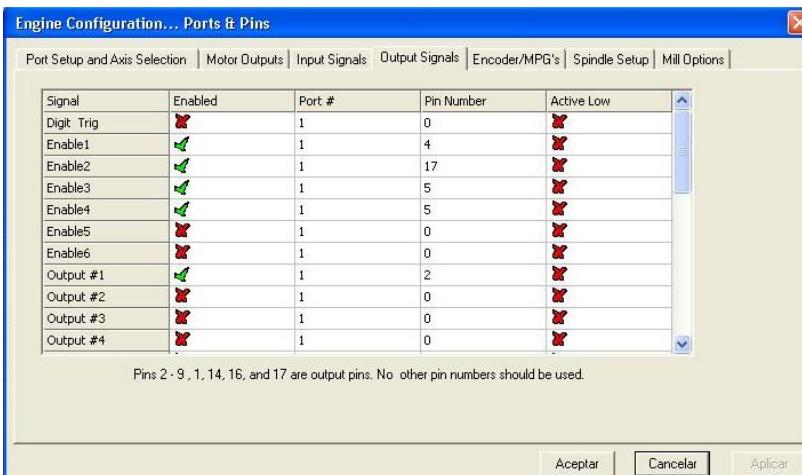


Figura 8: Señales de salida

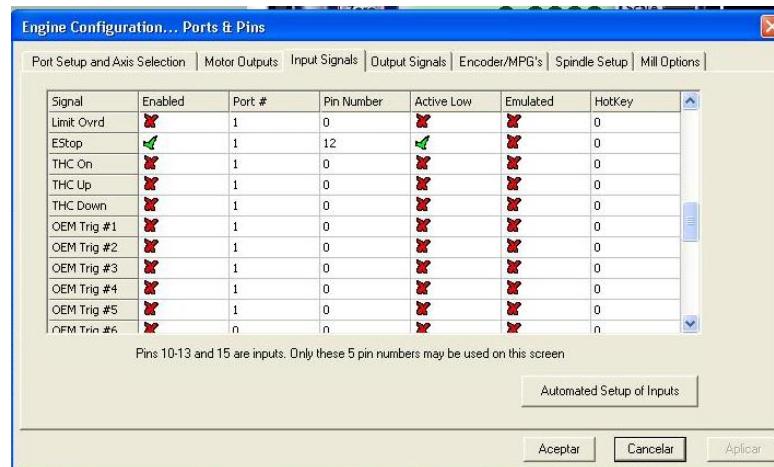


Figura 9: EStop de entrada

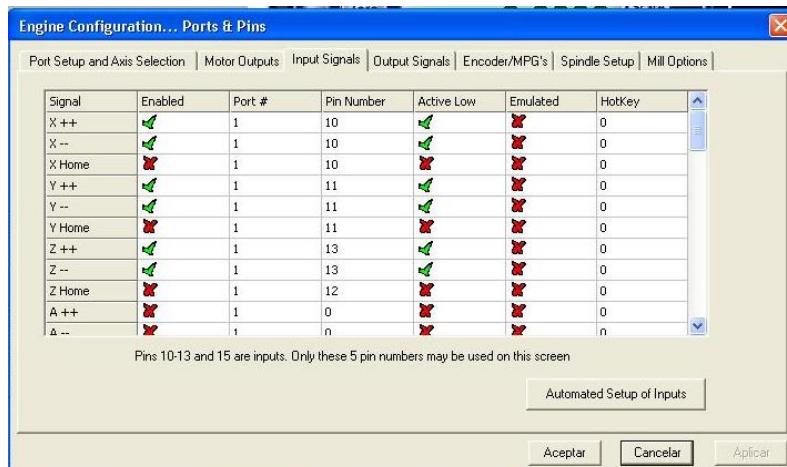


Figura 10: Señales de entrada

## ii. Configuración del motor

Para un correcto funcionamiento y una buena calibración del motor, también es necesario configurar los pasos que da cada uno de ellos, la velocidad máxima y la aceleración de los motores *steppers*. Con una buena configuración de estos parámetros es posible evitar la pérdida de pasos y asegurar el fresado final con un correcto dimensionamiento. Esta configuración se realiza desde el *Mach3* en *Config->Motor Tuning* tal como se muestra en la figura 11.

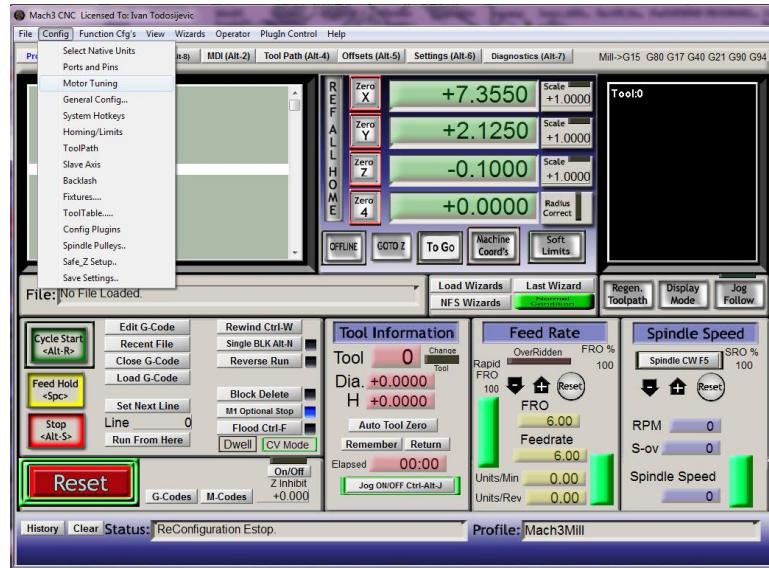


Figura 11: Configuración de motor

Desde ahí se configurarán los pasos de cada motor por cada 1 mm, en este caso serán 100 pasos por cada 1 mm. Estas configuraciones se muestran en las figuras 12, 13 y 14 para los ejes X, Y, Z respectivamente. Estos parámetros son 100 steps per, 120[mm per min] y una aceleración 1.48 mm/sec/sec para los ejes X e Y y aceleración de 0.5 mm/sec/sec para el eje Z. Finalmente hay que presionar el botón *SAVE AXIS SETTINGS* de modo que si no se presiona no se guardará la configuración. Estas velocidades

pueden variar pero al aumentar mucho las velocidades de los motores comienzan a perder pasos y agregan error en el fresado de una pieza.

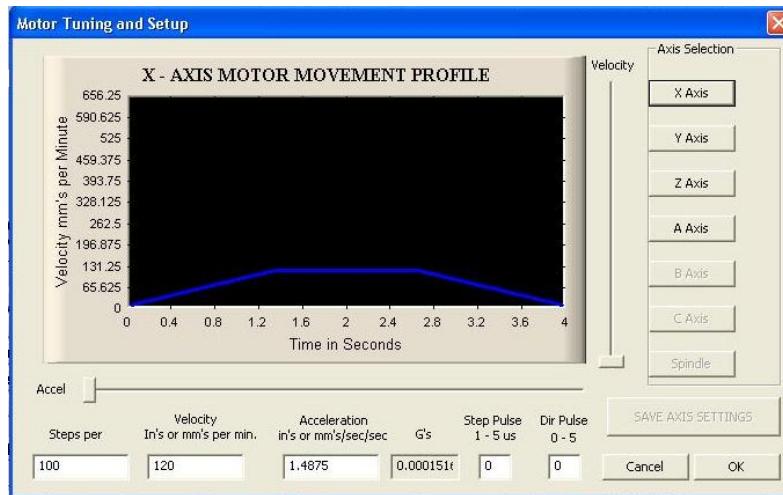


Figura 12: Configuración de motor en el eje x

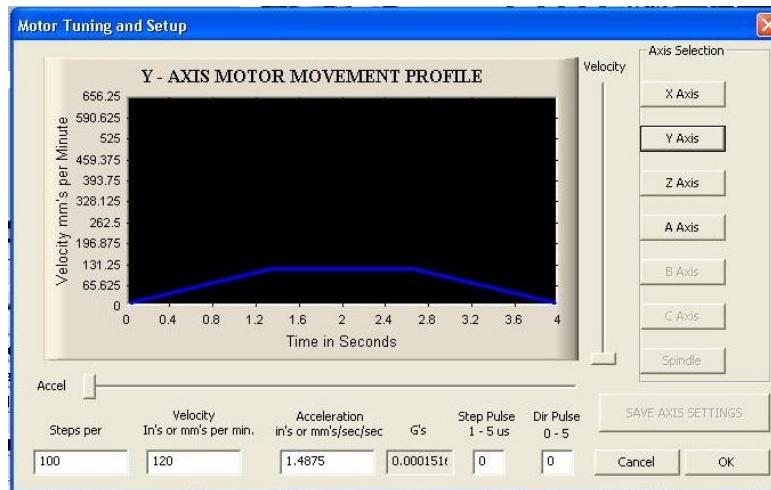


Figura 13: Configuración de motor en el eje y

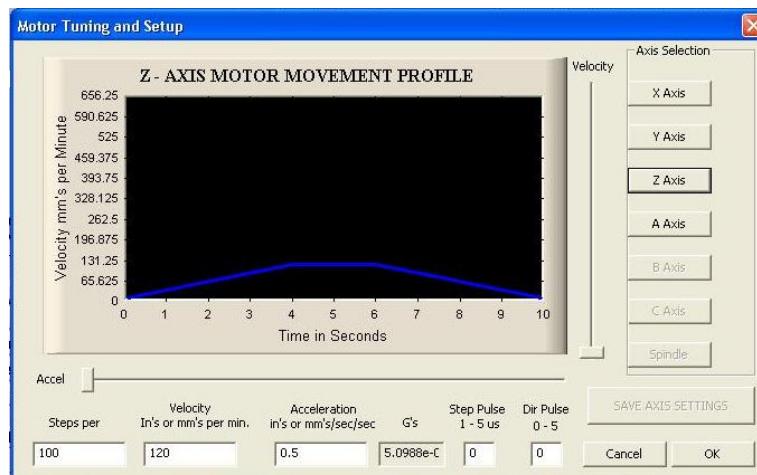


Figura 14: Configuración de motor en el eje z

### c. Código G

El Código G es un tipo de codificación ampliamente utilizado para el control numérico CNC (*Computer Numerical Control*). Este lenguaje es utilizado para comandar la fresadora sobre cómo y hacia dónde moverse, controlando la velocidad, la posición siguiente y la trayectoria a seguir de la fresa para el devastado de material.

Los comandos básicos utilizados se muestran generalizados en la figura 15.

<b>The modal Groups for G codes are</b>
<ul style="list-style-type: none"> <li>• group 1 = {G00, G01, G02, G03, G38.2, G80, G81, G82, G84, G85, G86, G87, G88, G89} motion</li> <li>• group 2 = {G17, G18, G19} plane selection</li> <li>• group 3 = {G90, G91} distance mode</li> <li>• group 5 = {G93, G94} feed rate mode</li> <li>• group 6 = {G20, G21} units</li> <li>• group 7 = {G40, G41, G42} cutter radius compensation</li> <li>• group 8 = {G43, G49} tool length offset</li> <li>• group 10 = {G98, G99} return mode in canned cycles</li> <li>• group 12 = {G54, G55, G56, G57, G58, G59, G59.xxx} coordinate system selection</li> <li>• group 13 = {G61, G61.1, G64} path control mode</li> </ul>
<b>The modal groups for M codes are:</b>
<ul style="list-style-type: none"> <li>◆ group 4 = {M0, M1, M2, M30} stopping</li> <li>◆ group 6 = {M6} tool change</li> <li>◆ group 7 = {M3, M4, M5} spindle turning</li> <li>◆ group 8 = {M7, M8, M9} coolant (special case: M7 and M8 may be active at the same time)</li> <li>◆ group 9 = {M48, M49} enable/disable feed and speed override controls</li> </ul>
<b>In addition to the above modal groups, there is a group for non-modal G codes:</b>
<ul style="list-style-type: none"> <li>◆ group 0 = {G4, G10, G28, G30, G53, G92, G92.1, G92.2, G92.3}</li> </ul>

Figura 15: Comandos de Código G.

Ejemplo de un código básico.

```

G90 G80 G49 'Configuración de parámetros de la máquina
G0 Z1.0000   'Se mueve Z en su máxima velocidad a la posición 1
S333          'Configuración de la velocidad de la fresa (Spindle)
G0 Z-0.1      'Se mueve Z en su máxima velocidad a la posición -1
G0 X0.0845 Y0.0341 'Se mueven los ejes X, y en su máxima velocidad a la posición (0.0845, 0.0341)
F5000         'Se configura la velocidad de los motores steppers
G1 X0.0936 Y-0.0037 'Se mueven los ejes X, y a velocidad configurada a la posición (0.0845, 0.0341)

```

Para mayor profundización sobre Código G por favor revisar el manual de *Mach3* de la referencia [2] *Using Mach3*.

#### d. Edición de pantallas del *Mach3*

Para editar las pantallas del *Mach3* será necesario instalar el software desde el CD anexado en el manual y que se encuentra en la ruta “/Mach3\_3.042/Mach3Screen/MachScreenV1.55.exe”. Este software se utiliza para agregar, editar y/o borrar las distintas pantallas del *Mach3*. En el caso de esta memoria se utilizó para crear la pantalla *Closed loop* y se podrá seguir editando a través de este software.

Luego de abrir el software será necesario cargar el archivo de las pantallas del *mach3* 1024.set que se encuentra en “C:\Mach3\1024.set”. Simplemente editando este archivo se editarán las pantallas automáticamente y quedará guardado cualquier cambio que se haga con este software.

Una vez abierto el software y cargado el archivo del *Mach3* aparecerán las pantallas de las figuras 16 y 17. Este editor puede modificar todas las pantallas contenidas en el *Mach3* y agregar algunas extras si fuese necesario. La pantalla de la figura 17 (*MachScreen properties*) permite seleccionar cuál de las pantallas (ejemplo: *Program Run* (Alt-1), *Closed loop* (Alt-8) u otros) se desea editar y los tamaños de ésta. Desde *Mach3Screen properties* (Figura 17), se pueden agregar diferentes componentes, estos son:

DRO: Componentes que poseen el valor de la posición del eje X, Y, Z, A, B, C u otro posible eje.

Image: Una imagen cualquiera.

Button: Un botón, aquí se puede agregar cualquier tipo de funcionalidad tal como se muestra en la figura 17 en el lado derecho, en el caso de dejarlo como *Execute Basic-Script* queda la posibilidad de agregar funcionalidades del modo de *scripts* en *Visual Basic*, los *scripts* se agregarán directamente desde el *Mach3* y en la sección siguiente del manual se comentará en más detalle.

Jogball: Una bola que representa el control desde un teclado u otro tipo de control externo.

LED: Leds para representar las señales o errores.

Label: Labels para marcar sectores.

Image button: Botón de imagen.

MDI: Bloque para escribir manualmente Código G.

Código G: Muestra el Código G y su avance.

Toolpath: Muestra el gráfico de posicionamiento del Código G.

Para crear el sistema con lazo cerrado se creó la pantalla *Closed loop* (*Alt-8*)

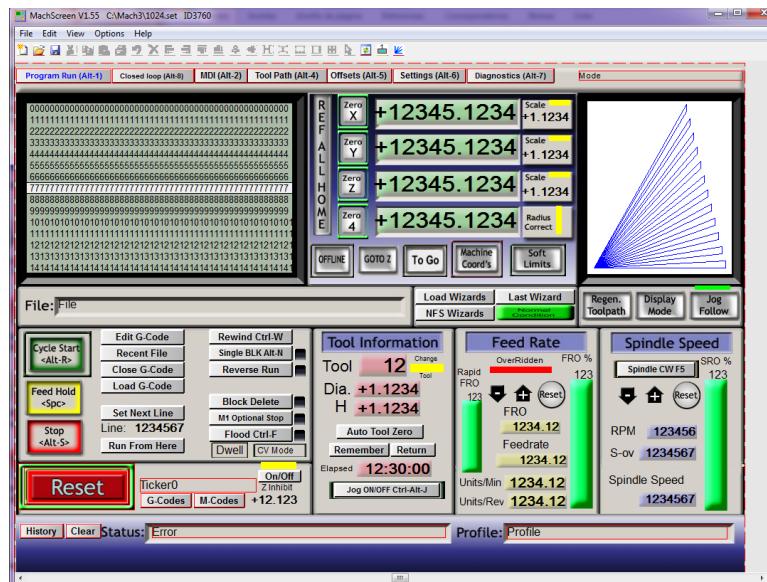
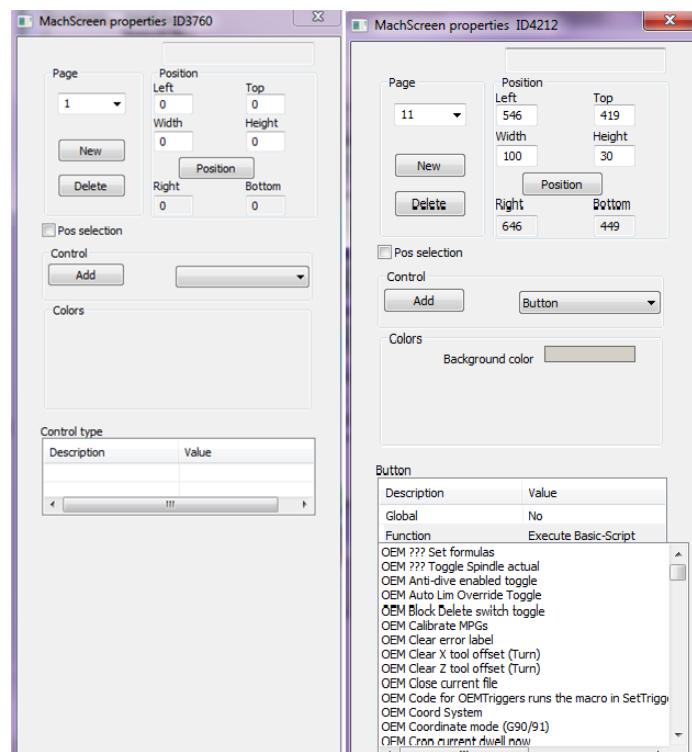


Figura 16: Editor de pantallas *Mach3*, *MachScreen*

Figura 17: Editor de pantallas *Mach3, MachScreen*

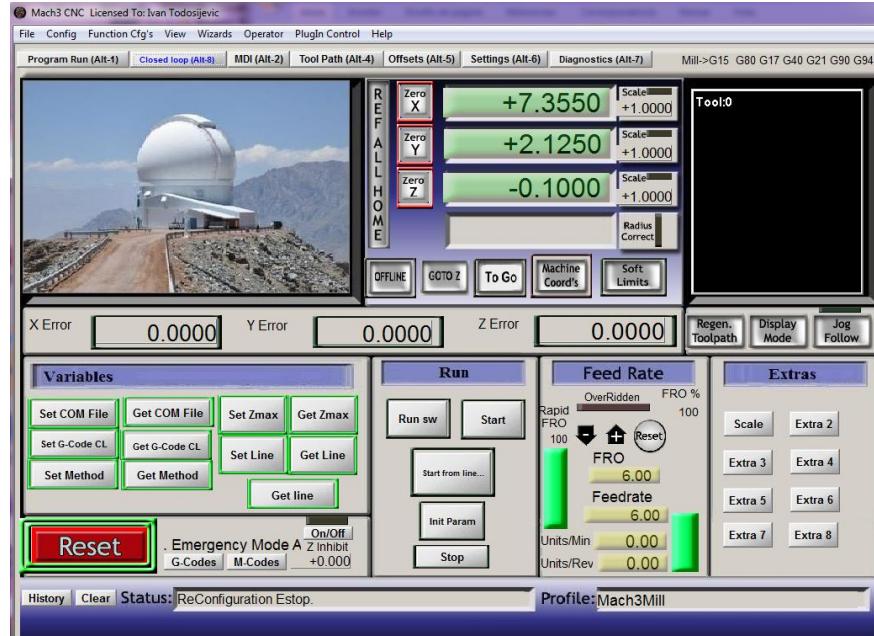


Figura 18: Pantalla creada “Closed loop (Alt-8)”

#### e. Diseño de *Scripts* a través de VB

##### i. Edición de *Scripts* de botones

El propósito de este capítulo es describir cómo se pueden generar scripts para realizar funciones diferentes a las que trae el *Mach3* y detallar sobre las funciones básicas que posee el *Mach3* para poder ejecutar código y guardar variables entre scripts. Para agregar funcionalidades en los botones es necesario que en *Function* se los deje como *Execute Basic Script* tal como se muestra en la figura 17 a la derecha y así quedan libres para agregar cualquier funcionalidad en VB. Luego para agregar las funciones que se deseen realizar se deberán generar directamente desde al *Mach3*, en *Operator->Edit Button Script* y luego presionar el botón al que se le desea agregar scripts. En la

referencia [3] se encuentra un manual completo de Visual Basic (VB) para *Mach3* con detalles sobre la programación en VB.

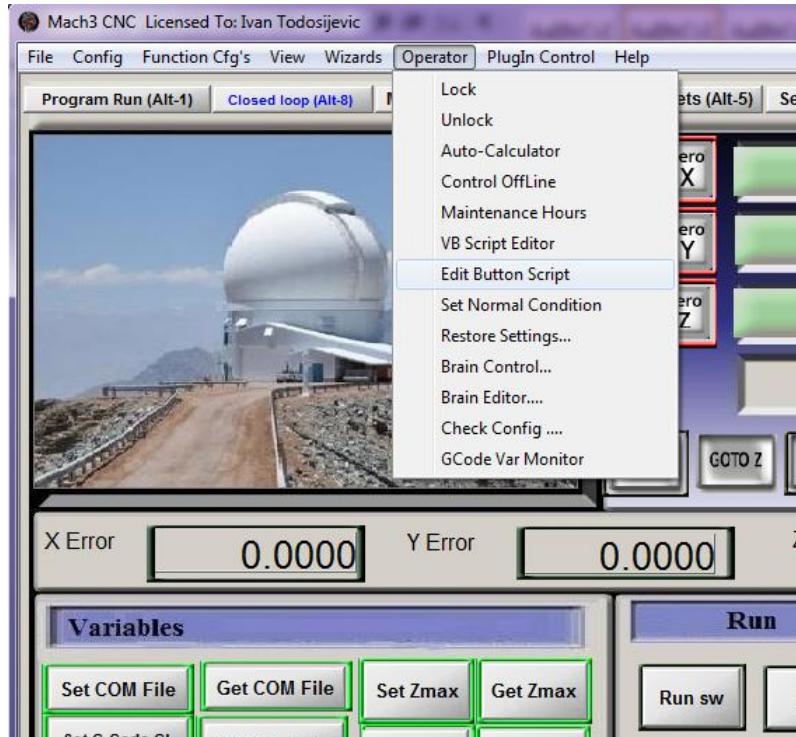


Figura 19: *Edit Button Script*

La pantalla *Closed loop* posee varios botones *Extra N* para agregar funciones en caso de que sea necesario. Las funciones básicas en VB del *Mach3* son:

*GetDRO(var)*: Función que retorna el valor de la posición de cada eje del *Mach3*, var = 0 para X, var = 1 para el eje Y y var = 2 para el eje Z.

*SetDRO(eje,valor)*: Función que configura el eje del *Mach3* en un determinado valor.

InputBox(""): Función que permite ingresar valores al *Mach3* desde una pantalla externa.

SetUserLabel(constante,valor): Función que permite guardar un valor en una determinada variable. Este valor es guardado en el software en una memoria interna del *Mach3* por lo que es accesible desde cualquier script.

GetUserLabel(constante): Función que retorna el valor de una determinada constante.

Al ingresar al *Mach3* en la pantalla *Closed loop* se pueden ver ejemplos de las distintas funciones en los diferentes botones creados.

## ii. Detalles de funciones de *Closed loop*

En esta sección se entregarán detalles sobre las funciones realizadas en Visual Basic para posibles mejoras en el código. Primero se va a generalizar con respecto a las funciones Set y Get ya que simplemente guardan y obtienen parámetros.

Las funciones Set tendrán una estructura en la que NN será el nombre de la variable que se quiere asignar.

```
void function SetNN(Line){
    set NN to Line
}
var function GetNN(Line){
    return NN
}
```

La función *Run sw* queda de la siguiente manera: Si la función no ejecuta el software es probable que sea debido a que está mal direccionado el *path* y será necesario corregirlo.

```
void function Run sw{
```

```
    Execute Fress  
}
```

Para mayor simpleza el resto de las funciones, *Start*, *Start from line...* y, *Scale*, se detallarán a través de diagramas de bloques.

El diagrama 1 representa la función que posee el botón *Start*. Comienza en el bloque Lectura y creación de variable, luego sigue el orden del diagrama, en que comienza a leer el Código G y a ejecutarlo línea por línea comunicándose con el software fress para poder hacer una comparación de las posiciones leídas por la cámara y las posiciones que posee el *mach3* y así estimar el error. Luego entra al bloque de algoritmo de decisión del diagrama 2, que decide qué algoritmo usar en caso de que el error sea mayor que un cierto delta de los parámetros inicializados en el código.

Finalmente si se acaban las líneas, se presiona *Stop* o no pasa el algoritmo de decisión, finaliza el ciclo y deja de ejecutar el Código G. El algoritmo de decisión se encuentra explicado más abajo.

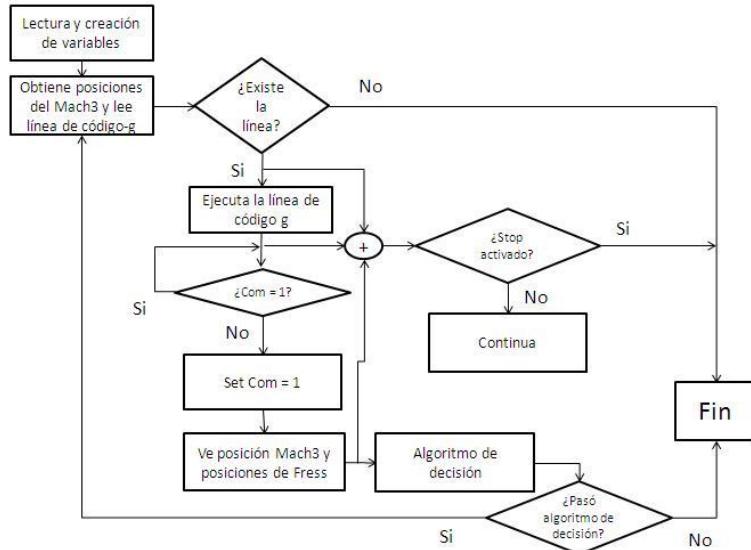


Diagrama 1: Diagrama de bloques con función "Start"

El diagrama 2 explica el algoritmo de decisión. Parte desde el bloque que consulta si el método utilizado es 3 o no (el método 3 consulta qué otro método utilizar), si es así le consultará al usuario qué método desea utilizar (0, 1 o 2). Si el método es 0 el sistema continuará sin corregir errores y cuando el error sobrepase un delta preguntará si se desea continuar. Si el método es 1, el sistema se autocorregirá de acuerdo a las posiciones de lectura del software *Fress* y si el método es 2, el sistema volverá a la posición anterior y ejecutará la línea nuevamente de acuerdo a las posiciones medidas por *Fress*.

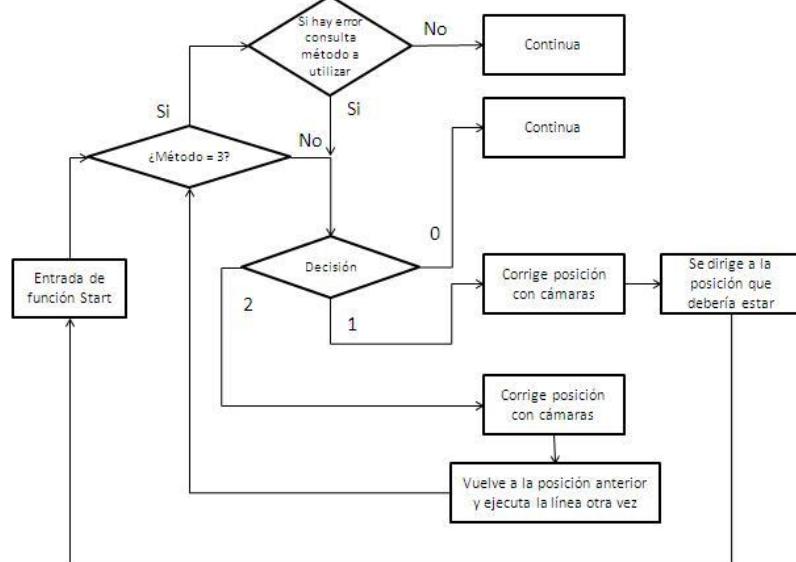
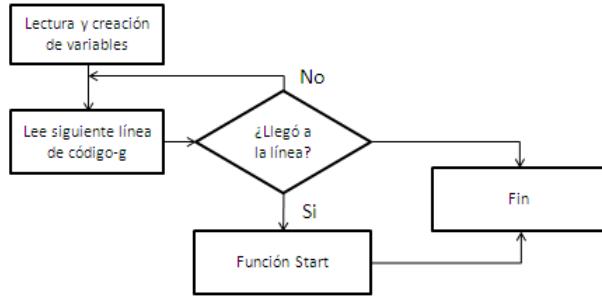


Diagrama 2: Diagrama de bloques de algoritmo de decisión

El diagrama 3 muestra la función del botón *Start from line...*, este método comienza leyendo el Código G e itera hasta llegar a la posición de la línea deseada para luego comenzar a ejecutar la misma función que posee el botón *Start*. Si se desea modificar el botón *Start*, también deberá hacerse desde esta parte.



Por último el diagrama 4 presenta el algoritmo del botón *Scale*. Este algoritmo consiste en ejecutar la función *Start* utilizando un Código G especial para estimar ciertas posiciones desde el *Mach3* y *Fress*. De esta manera se puede estimar un escalamiento entre la medición de ambas posiciones, es decir,  $Posicion_{Mach3} = \beta * Posicion_{Fress}$  con beta como el escalamiento. Si este valor se encuentra correcto deberá ser semejante a 1. En caso contrario hay significa que hay problemas o en la configuración del *Mach3* o en los parámetros de calibración.

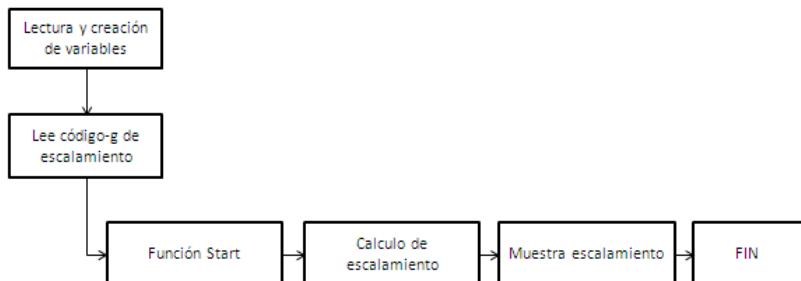


Diagrama 4: Diagrama de bloques con función “Scale”

#### f. Uso de *Closed loop*

Este componente del *Mach3* está diseñado especialmente para funcionar en conjunto con el *software Fress*. Si el *software Fress* no se encuentra en funcionamiento, este componente de *Mach3* no correrá debido a la falta de comunicación entre ambos programas. Esta aplicación se muestra en la figura 18 y ahí se pueden ver todos los componentes disponibles.

Todas las mediciones y correcciones del error de posicionamiento se harán al final de la ejecución de cada línea de Código G, ya que no es posible editar el *mach3* mientras el movimiento se esté ejecutando.

Todos los botones *Get* son para obtener el valor deseado y los *Set* son para dar un valor a los parámetros.

**COM File:** Es el archivo de comunicación que se usa entre *Mach3* y el *software Fress*.

**Código G CL:** Es la ruta del Código G a generar.

**Method:** Es el método de control que se utiliza.

- 0- No utiliza ningún método de control, pero está constantemente revisando si el error es mayor a un delta. Si el error es mayor al delta el software le preguntará al usuario si desea seguir con el fresado.
- 1- Utiliza un método de control directo, es decir, si posee un error mayor a un delta la fresa se re-direcciará automáticamente a la posición final de la línea del código.
- 2- Utiliza un método de control indirecto, es decir, si la posición final posee un error mayor a *delta*, la fresa volverá al punto anterior y ejecutará nuevamente esa línea de código.
- 3- Si existe un error mayor a un *delta*, el *software* consulta qué método utilizar.

**Zmax:** La altura máxima para volver a la posición anterior mediante el método 2.

**Line:** La línea a ejecutar desde *Start from line...*

**Run SW:** Ejecuta el *software Fress*.

**Start:** Comienza a ejecutar el Código G.

**Start from line:** Comienza a ejecutar el Código G desde la línea ingresada en *Set line*.

**Init Param**: Inicializa parámetros estándar.

**Stop**: Detiene la máquina después de haber ejecutado la línea de código. Si se desea detener instantáneamente es necesario presionar el botón *reset* o el botón de emergencia que se encuentra en la máquina misma.

**Scale**: Es una prueba de calibración y configuración de la máquina. Se revisa la relación de mm de la máquina con los mm del sensor de las cámaras web. Es decir, si el *software fress* dice 1 mm, de verifica que efectivamente sea 1 mm en la máquina. Los resultados se entregarán como la relación (mm *Mach3*)/(mm *Fress*) y se presentarán como se muestran en la figura 20.

**X Error, Y Error, Z Error**: Entrega el error obtenido entre el *Mach3* y el *software Fress* después de la ejecución de cada línea de Código G.

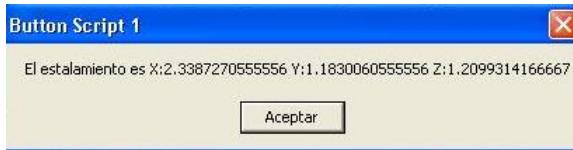


Figura 20: Resultados del botón *Scale*

#### g. Comunicación entre *Mach3* y *software fress*

La comunicación entre ambos *software* se realiza a través de archivos, para lo que se crearon dos archivos, uno principal que envía las posiciones leídas desde la cámaras *web* (archivo que se *setea* desde ambos *software*) y otro archivo verificador que permite revisar a qué *software* le corresponde acceder al archivo de comunicación. Ambos archivos se encuentran en la ruta “C:\Mach3\”.

Se realiza un protocolo de comunicación, el cual se muestra en un diagrama explicativo en la figura 21. Aquí existen dos archivos, al que se desea acceder y posee las posiciones de las cámaras y el que posee un parámetro llamado “Modifiable”, el cual representa a un parámetro que dice a qué *software* le toca acceder al archivo de posición. Este parámetro modifiable posee un valor de 1 si le toca acceder al *software*

*Fress* o 0 si le toca acceder al software *Mach3*. Una vez que se accede al archivo el parámetro modifiable cambia de valor para darle paso al otro software a acceder.

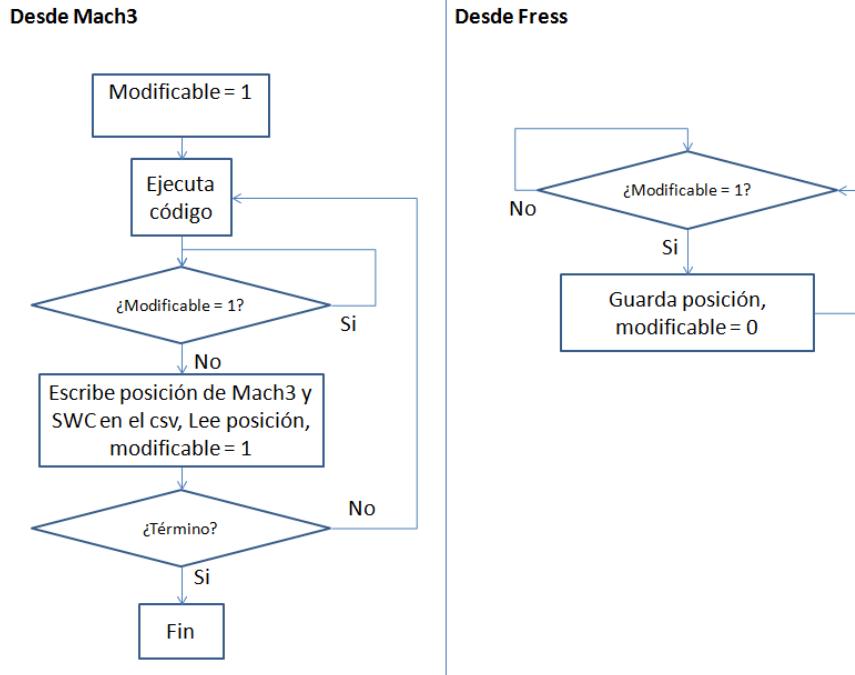


Figura 21: Comunicación entre *Mach3* y el software *Fress*.

## 2. Software de diseño 3D

Existe una gran variedad de *software* para diseños 3D. Estos programas son llamados CAD (*Computer Aided Design*) y se refiere al uso de computadores para la ayuda en la creación de diseños técnicos 3D a través de un software. Mediante estos programas se puede crear una infinidad de formas y se pueden replicar objetos con bastante precisión de manera de poder simular cualquier tipo de pieza mecánica. Existen muchos programas que sirven para realizar prototipos 3D, ya sea “*Autodesk Inventor*”,

“CATIA”, “SolidWorks”, “AutoCad”, entre otros. Estos *softwares* sirven para crear las piezas que finalmente se utilizarán para replicar a través de la fresadora y el CNC.

Todos estos programas ofrecen un gran potencial de desarrollo y con cualquiera de ellos se puede llegar a formar piezas bastante complejas y con las mediciones deseadas. Se optó por escoger el software *SolidWorks* porque los futuros usuarios de la fresadora ya poseen experiencia en el diseño a través de este *software*.

En el siguiente apartado se detallará brevemente cómo realizar piezas simples 3D.

#### *a. Solidworks*

Este software es bastante sencillo de usar. A continuación se explicará paso a paso y con ejemplos, cómo se debe utilizar para crear una pieza sencilla.

Primero, crear un nuevo documento *solidworks* y seleccionar la sección “*a 3D representation of a single design component*”. Aparecerá la misma pantalla que aparece en la figura 22.

Segundo, normalizar la imagen sobre el *front plane* que luego simplificará el uso del siguiente software. La normalización se muestra en la figura 23 y debe quedar como se muestra en la figura 24.

Tercero, crear el bloque en *2D* sobre el cual se desea trabajar. En la figura 25 se muestra cómo se crea una línea y a la izquierda del *software* se fija el dimensionamiento en milímetros que se le desea dar a la línea. De esta manera se crea un cuadrado con las dimensiones deseadas y luego se presiona sobre el botón *Sketch* para salir.

Cuarto, agregar una tercera dimensión a este cuadrado dándole altura. Para realizar esto se debe presionar sobre *features* como se muestra en la figura 26 y luego *Extruded Boss/Base* donde se fijan las dimensiones de altura. Luego si se desea crear un nuevo bloque sobre la superficie de este bloque, se repiten los pasos anteriores generando un *sketch* sobre la base del bloque.

Quinto, en el caso de querer generar una cara bajo el bloque quitando material, se debe generar la figura deseada en un nuevo *sketch* sobre la cara del bloque y luego en *features* presionar sobre *Extruded Cut* y escoger cuándo se desea que se elimine del bloque principal tal como se muestra en la figura 28.

Finalmente con la figura ya lista se debe guardar en formato “*Parasolid .X\_T*” para luego poder trabajarla con el software CAM y transformarla en Código G.

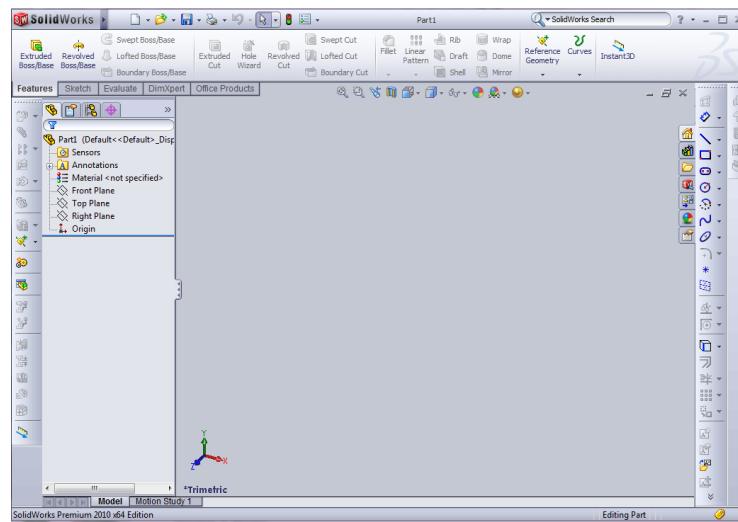


Figura 22: Software SolidWorks con un documento solidworks creado.

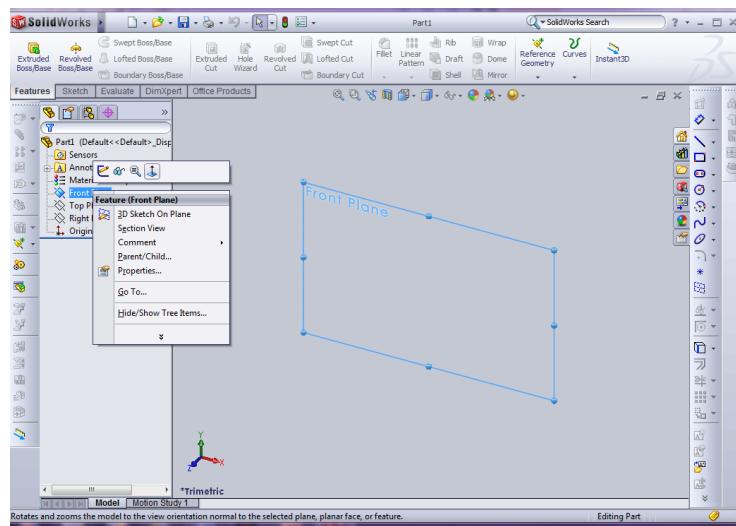


Figura 23: Normalización del plano.

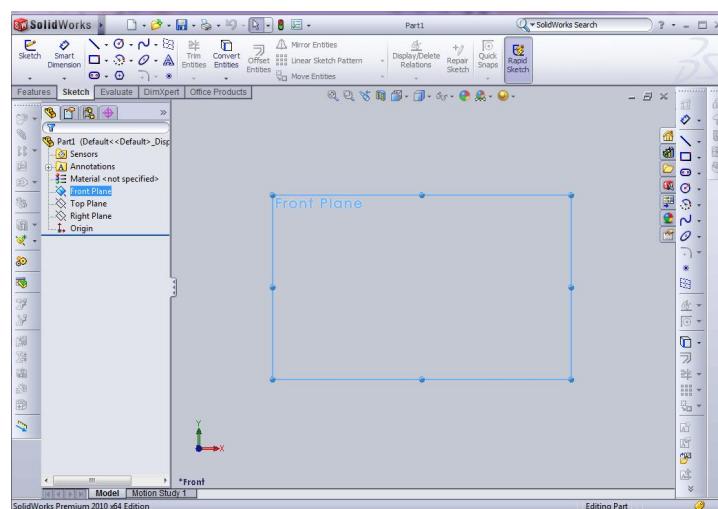


Figura 24: Plano normalizado y creación de sketch

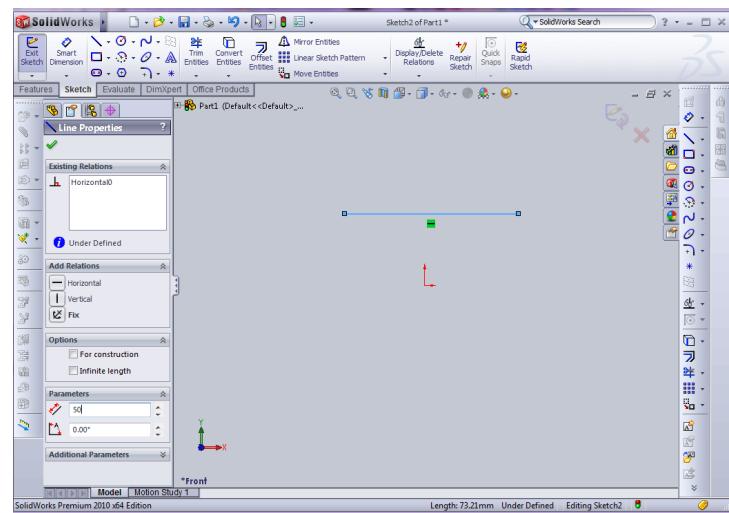


Figura 25: Creación de cuadrado 2D.

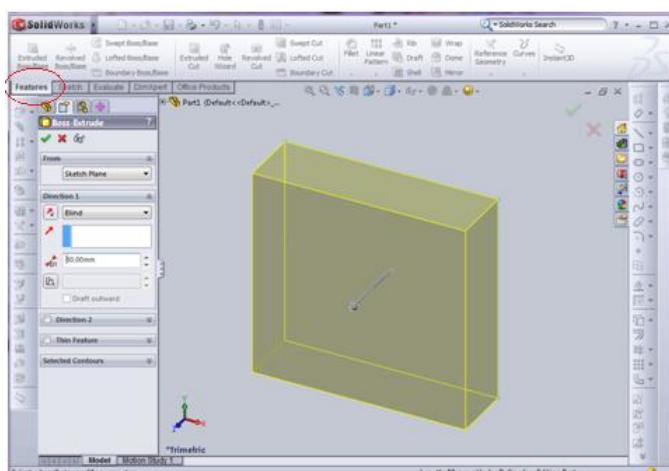


Figura 26: Generación de nueva dimensión.

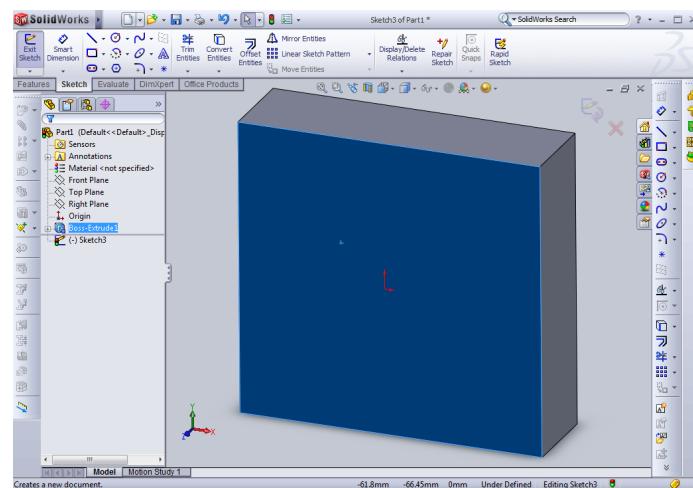


Figura 27: Creación de nuevo Sketch sobre bloque creado

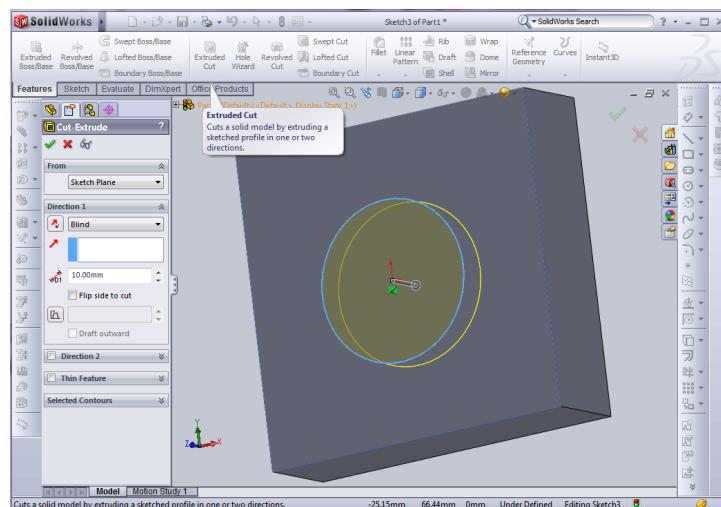
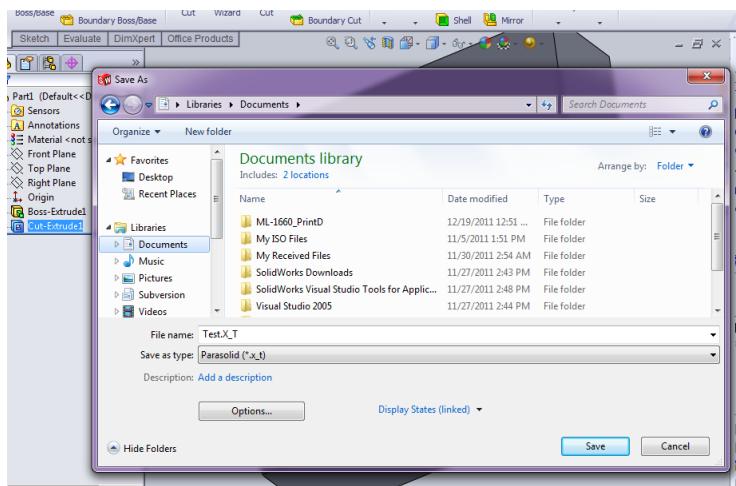


Figura 28: Quitar de nuevo material.

Figura 29: Guardado en formato *parasolid*.

### 3. Software CAM, (*MasterCam 3D*)

*CAM* se refiere a *Computer Aided Manufacturing* y corresponde al software que controla la máquina dedicada a la fabricación de piezas. El software básicamente toma una pieza 3D y la transforma a *Código G* que luego se deriva al *software CNC*. Se estudiaron dos *software*: SolidCam y MasterCam, debido a que son los más utilizados por su cantidad de funciones que simplifican la transformación a Código G. Se optó por utilizar MasterCam por ser fácil de usar y por tener un buen desempeño tras algunas pruebas.

#### a. *MasterCam*

Este software posee varias opciones para el devastado de piezas, desde el devaste rápido para comenzar la pieza, hasta el devaste fino para afinar los detalles de ésta. Posee una función que es capaz de realizar un devaste rápido desde el principio y es bastante sencillo de utilizar. Con una buena configuración se puede hacer cada vez más compleja hasta llegar a un devaste fino. A continuación se explicará cómo lograr utilizar

esta función para transformar las piezas computacionales a Código G, con el fin de generar piezas con bastante precisión.

Primero se deberá abrir el archivo de la pieza guardada anteriormente en formato *parasolid*. Luego se definirá el tipo de máquina que se utilizará. En este caso bastará con definir la máquina que viene por default y esto se hará desde el software MasterCam *Machine Type->Mill->Default* (la máquina por *Default*). Las figuras 30 y 31 muestran estas dos primeras etapas. Luego para cambiar la forma en que se ve la figura a sólido, será necesario hacer *click* sobre el círculo azul que se encuentra marcado en rojo en la figura 30.

A continuación se comenzará a generar el Código G para el devastado, para esto es necesario seleccionar desde *toolpath->Surface High Speed* y seleccionar la pieza. El paso siguiente es configurar el devastado, el eje de coordenadas, la posición inicial, la velocidad y otros parámetros internos de fresado. Para esto se deben seguir los siguientes pasos: primero se seleccionará el tipo de devastado y se utilizará el más simple denominado *Area clearance*, que consiste en el devastado rápido y que bastará para muchas piezas que se realicen. En la figura 33 se muestran algunos tipos devastados.

Luego desde *Tool* se configurará el tipo de broca que se utilizará y las velocidades de movimiento. Y las velocidades de corte son *Feed rate*, *retract* y *plunge rate* y bastará un valor de 200 en todos para un buen funcionamiento. En la figura 34 se muestra esta pantalla de configuración y los parámetros que se deben configurar.

La siguiente configuración será sobre los parámetros de corte *Cut Parameters*. Los parámetros importantes en esta sección son: *Stepdown*, que consiste en cuando baja la fresa por cada desgaste en el eje Z; *Stock to leave on walls* y *stock to leave on floors*, que consisten en la cantidad de espacio que hay que dejar en los bordes. Para que la pieza sea exactamente de los mismos tamaños propuestos se debe fijar la posición 0 de la pieza sobre la misma posición 0 que se fijó en el software *CAM*, esta configuración se puede apreciar en la figura 35.

En *linking parameters* están los parámetros de movimiento de la máquina, es decir, en las distancias máximas en que la fresa se puede mover libremente y sin tope con la pieza. Aquí en *retract* hay que tener ojo ya que la máquina no posee mucho

movimiento en el eje Z, por ejemplo, con valores mayores a 10 la máquina podría sobrepasar sus límites.

Ahora los planos de corte se configurarán desde *Planes (WCS)* como se muestra en la figura 37. Si la pieza se creó siguiendo los pasos de diseño con *Solidworks* de la sección anterior, entonces el plano debería corresponder al de la pieza, de no ser así, será necesario comprobar que calcen y de lo contrario configurarlo para que la sección superior de la pieza calce con los ejes de la fresadora. Para cambiar los planos se utilizan los botones marcados con un círculo rojo de la figura 37. Ahí se debe seleccionar sobre qué plano se desea trabajar. Luego se definirán las posiciones iniciales con el botón que se encuentra marcado con un círculo azul en la figura 37. En esta instancia se seleccionará la posición en que se desea que sea el 0 de la máquina y así se podrá referenciar la máquina y la posición de la pieza a una misma posición.

Una vez realizado todo lo anterior, el software está listo para crear el Código G, pero es recomendable probar antes los resultados desde el botón marcado con círculo rojo de la figura 38 izquierda. Desde ahí aparecerá una simulación del fresado tal como se ve en la figura 39 y sólo será necesario poner el botón *Play* para comprobar que el fresado se esté comportando adecuadamente antes de generar el Código G. Una vez probado el fresado se creará el Código G a través del botón marcado en rojo de la figura 39 a la derecha y se guardará en una ruta conocida para luego abrirla desde el *Mach3*.

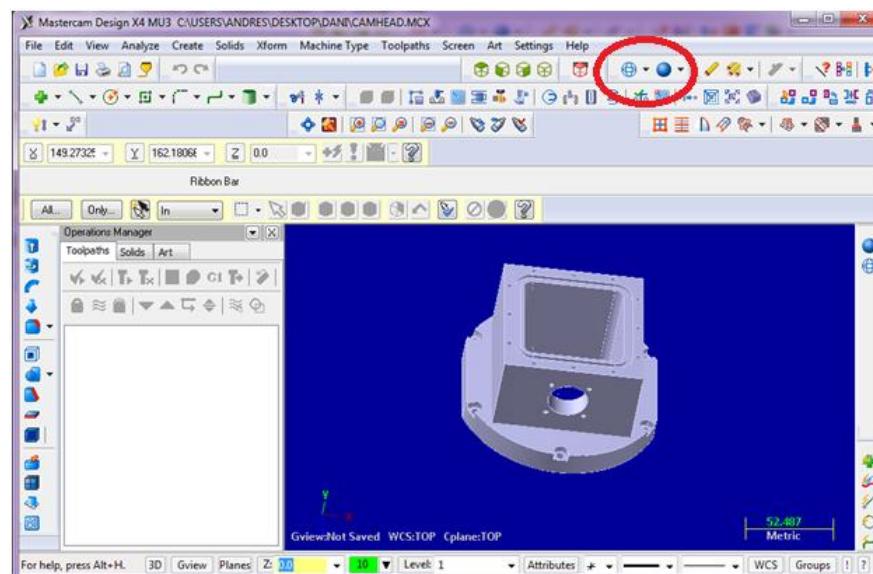


Figura 30: MasterCam con pieza abierta.

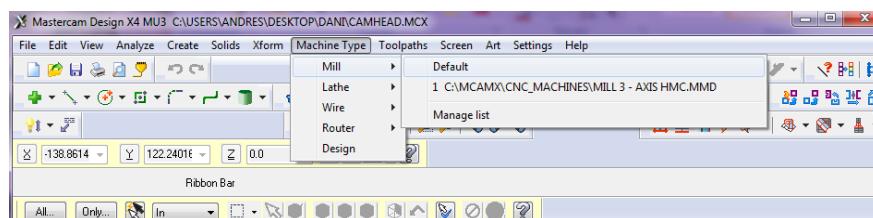


Figura 31: Asignación fresa.

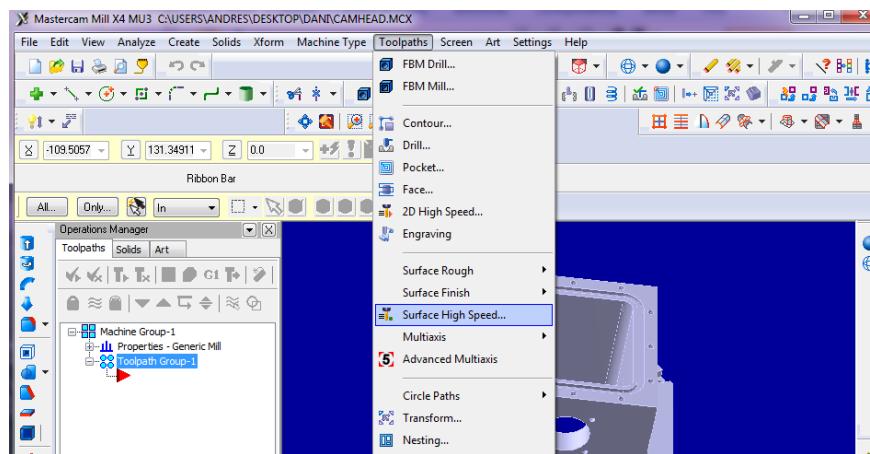


Figura 32: Método de creación de maquinado.

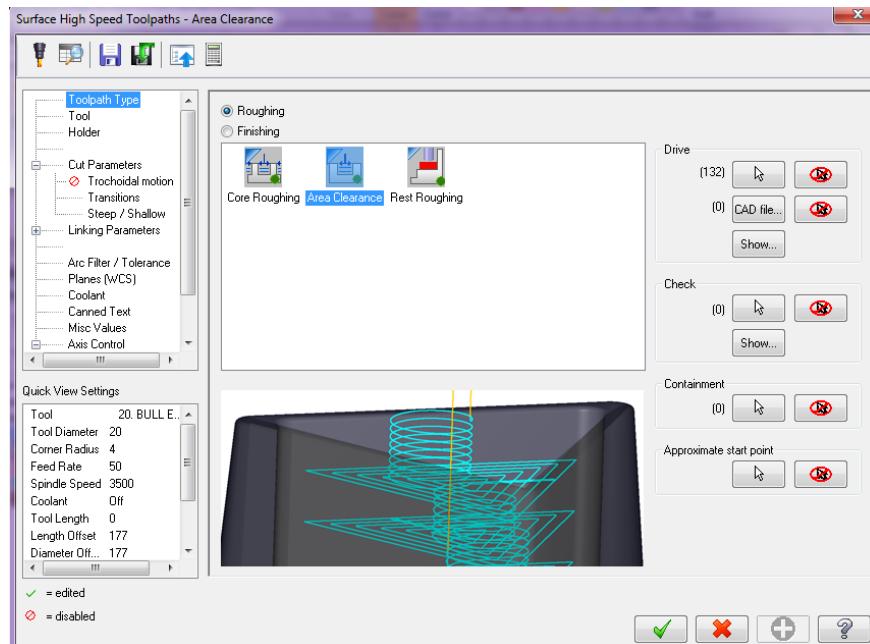


Figura 33: Devastado rápido

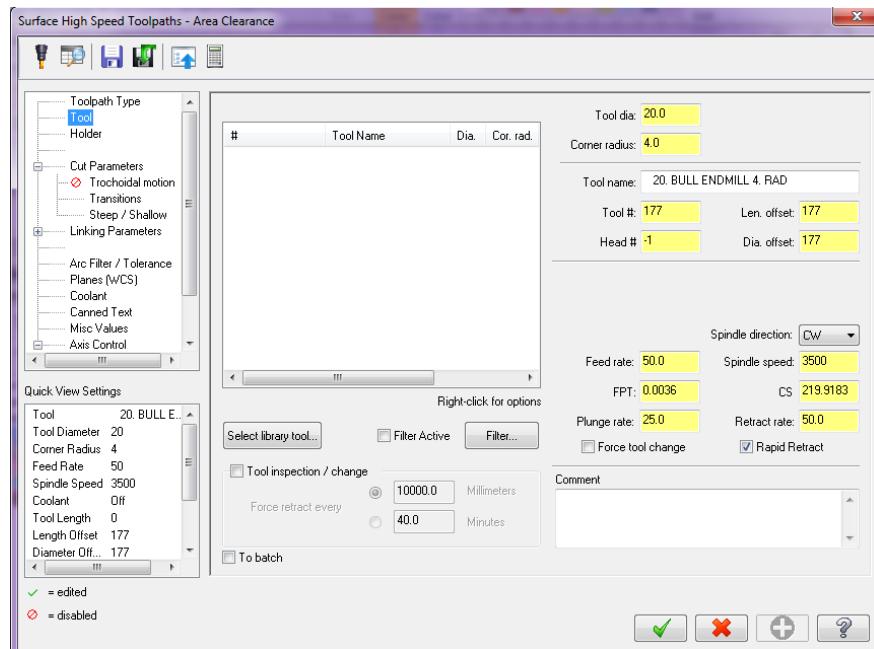


Figura 34: Configuración de la fresa y velocidades.

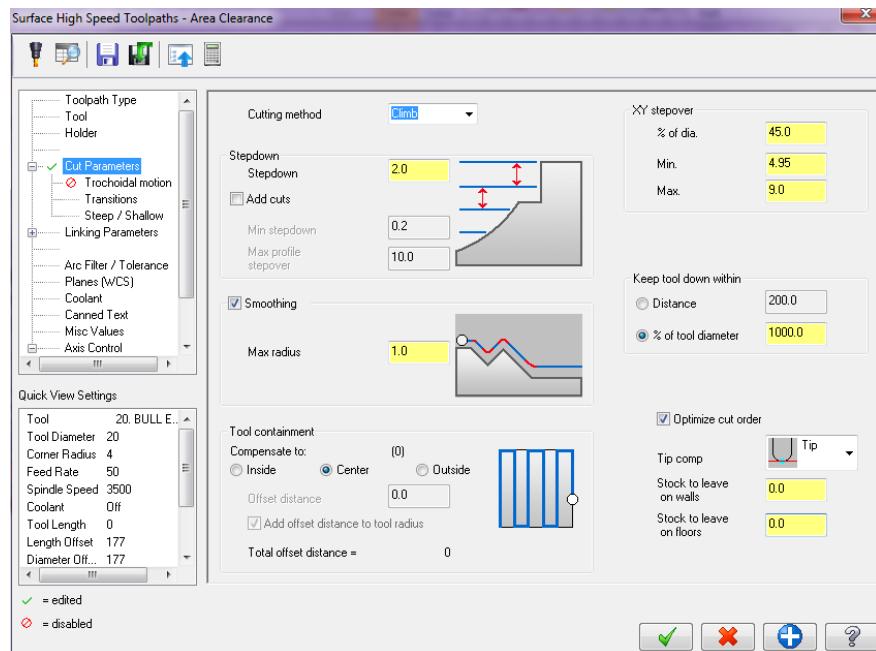


Figura 35: Configuración de corte.

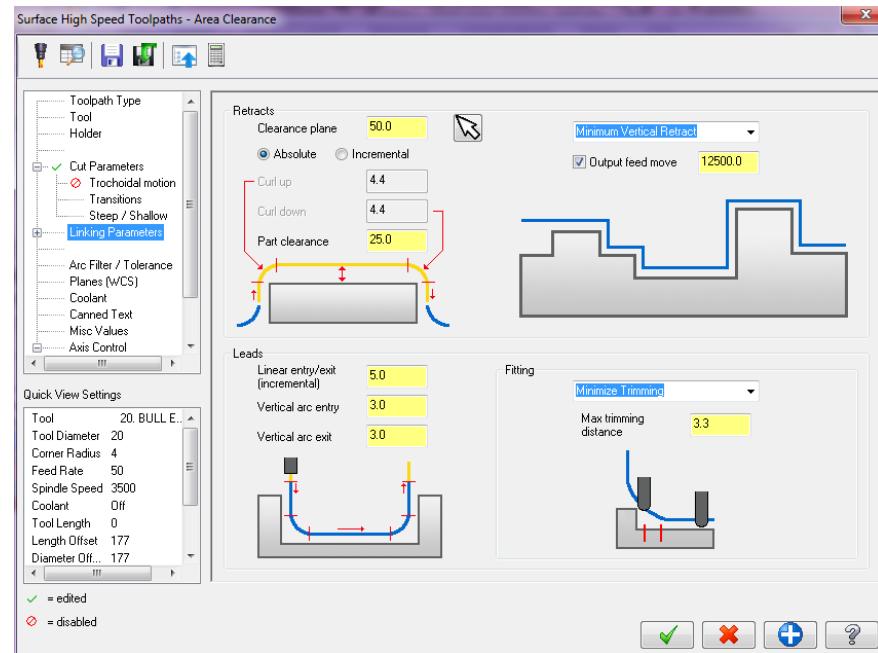


Figura 36: Configuración de maquinado.

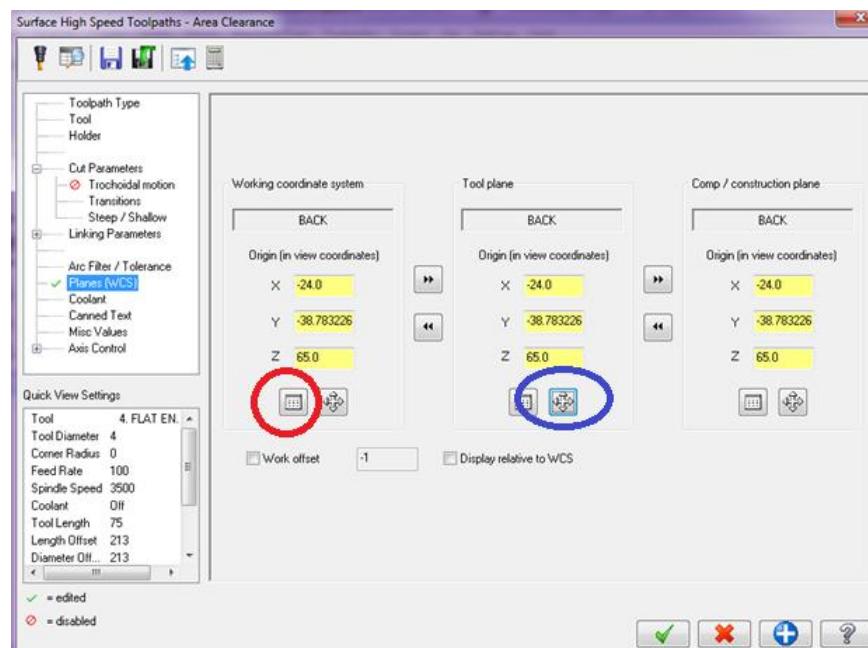


Figura 37: Configuración de planos y posición inicial.

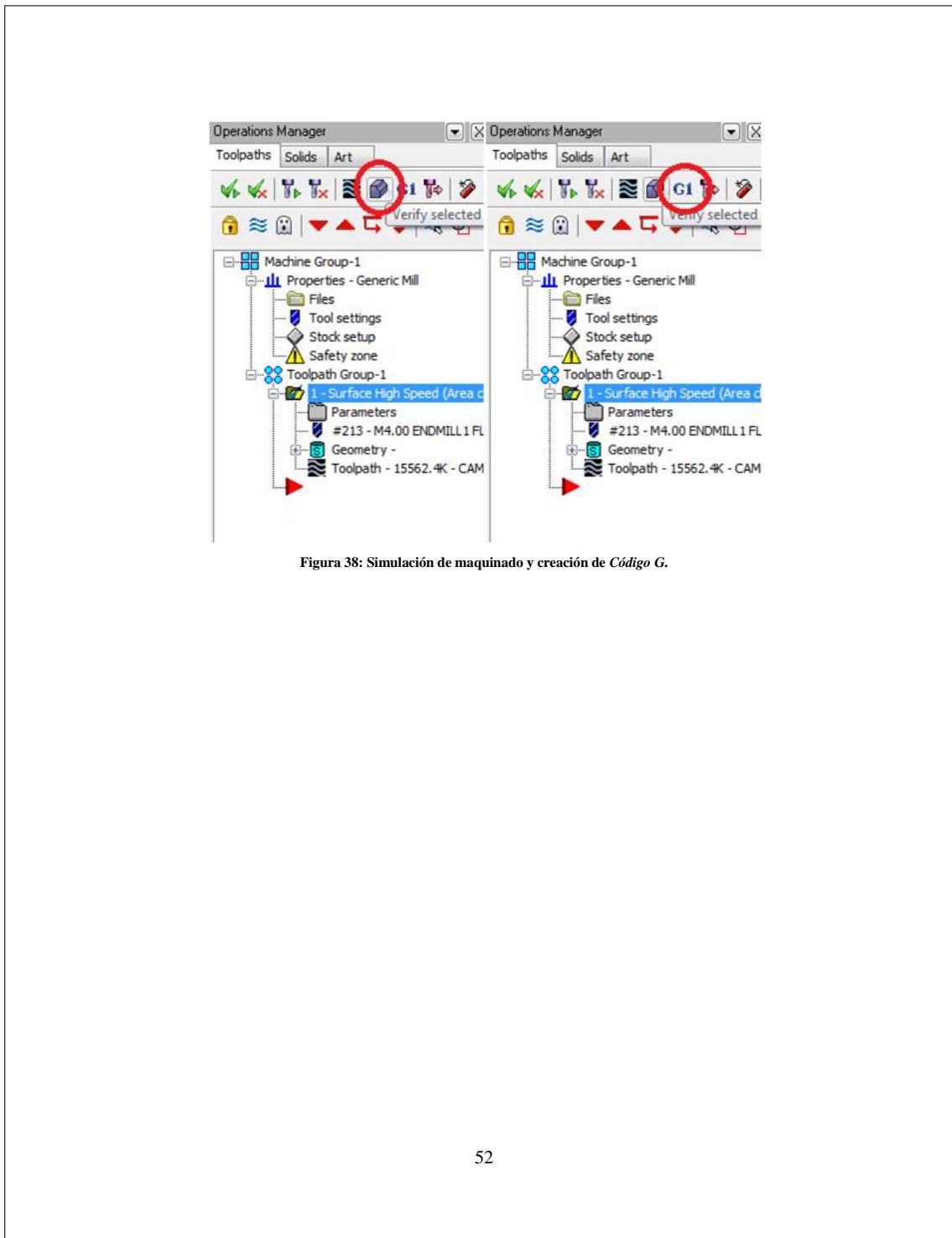


Figura 38: Simulación de maquinado y creación de Código G.

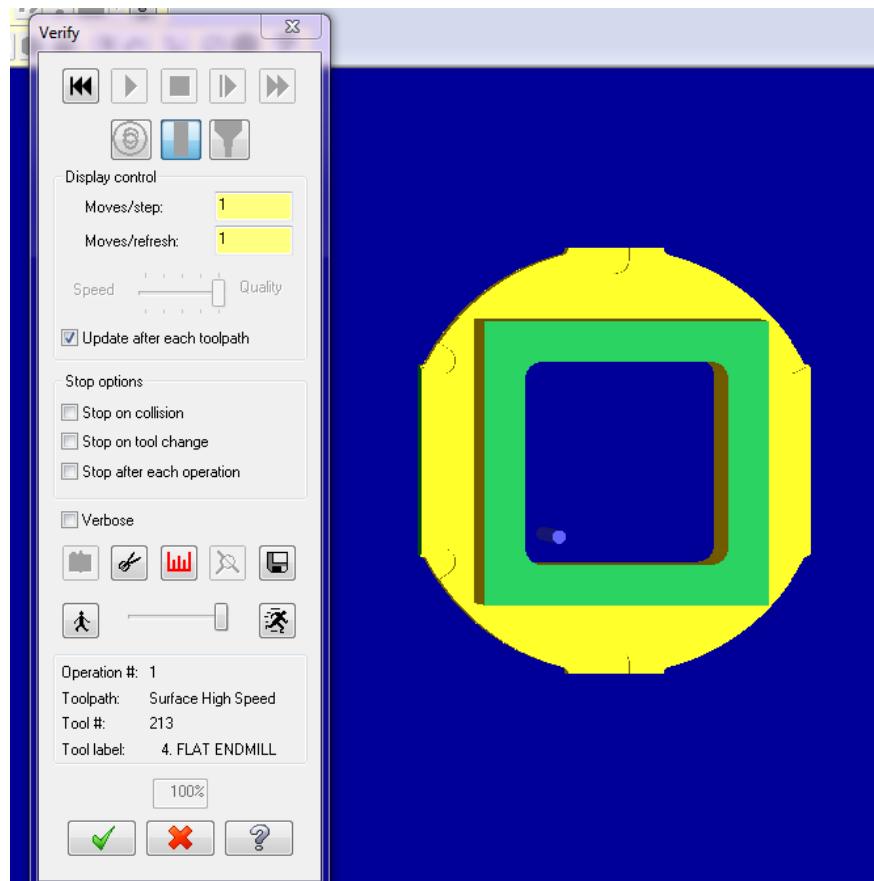


Figura 39: Prueba de maquinado y figura final.

#### 4. Software *Fress*

El software *Fress* consiste básicamente en un sensor que mide las posiciones (X, Y, Z) de la fresa y se las envía al *Mach3* para que estime los posibles errores de la máquina. Para medir las distancias se utiliza un método que discrimina por colores. Para un rango de colores HUE se genera una imagen binaria a la que finalmente se le calcula

el centroide para obtener la posición en pixeles del objeto de color. Una vez realizado este proceso, será necesario pasar a la etapa de transformación de medidas de pixeles a milímetros, para poder finalmente obtener la posición absoluta de la máquina para un determinado sistema referencial y en una unidad conocida (milímetros).

La figura 40 muestra una pantalla del software *fress* con todos sus botones y funcionalidades.

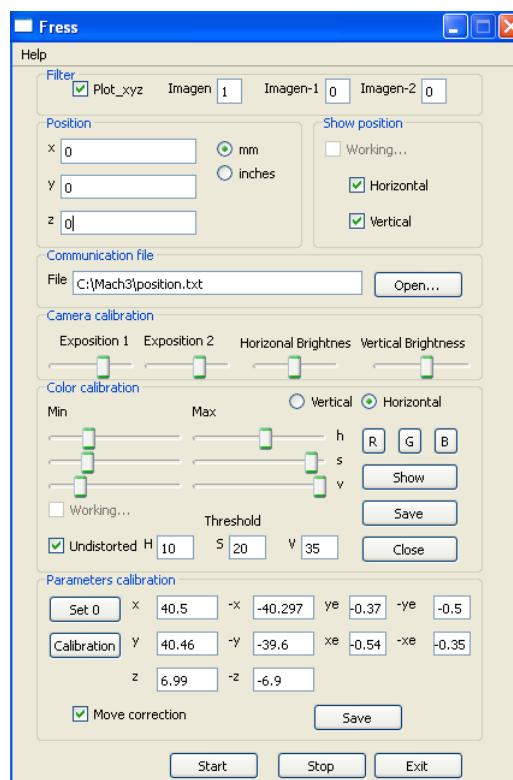


Figura 40: Software *fress*.

#### a. Modo de uso

Las siguientes secciones estarán destinadas a explicar el uso de este *software*, desde la calibración hasta el uso final.

### i. Calibración de cámara

#### 1. Calibración de configuración de exposición y brillo.

Esta etapa consiste en la calibración de la cámara. Es un procedimiento esencial ya que con una mala calibración de cámaras no se podrán obtener resultados constantes y se tendrá un ruido variable y no medido, que hará variar la posición sensada con las cámaras web de manera aleatoria. Para la calibración se necesitarán cámaras que permitan el cambio de configuración en los tiempos de exposición y en el *white balance*. Generalmente en las cámaras viene por default la calibración automática, lo que hará que se produzcan cambios constantes de color para corregir los colores y la iluminación. En este caso no se requieren correcciones automáticas, ya que se desea una imagen constante en colores e intensidad.

El *software fress* posee una sección *Camera calibration* destinada a la calibración de los parámetros de exposición e iluminación. Para una buena medición desde las cámaras web, se requerirá que la zona que se desee medir se encuentre bien iluminada (a mayor iluminación menor será el error). Esto es para intentar minimizar el valor de la exposición de las cámaras, ya que el error será acumulativo y a mayor exposición mayor será la integración del error. Con el fin de mantener una buena velocidad de *software*, la calibración de exposición se hace únicamente al comienzo de cada aplicación del *software*. Para hacer efectiva la variación en la exposición se deberá detener la aplicación que se esté ejecutando y volver a ejecutarla (ya sea *Show o Start*). Después de cada calibración es necesario presionar el botón *Save* que se encuentra en *color calibration*, el cual sirve para guardar la configuración de calibración.

El brillo será independiente y mientras se mantenga una buena medición de las cámaras estos valores pueden variar sin perjudicarla.

#### 2. Calibración de distorsión.

Existirá otro parámetro de calibración y viene dado por la distorsión propia de la cámara. Esta corrección sólo se hará para la cámara horizontal, ya que para la medición de las posiciones cd esta cámara abarca el campo de visión casi completo, a diferencia del eje vertical.

Actualmente los parámetros de calibración de la cámara Logitech c110 se encuentran guardados en el disco anexado en la carpeta “/Mach3\_3.042/Mach3/” como Intrinsic.xml y Distortion.xml y que están guardados en la ruta del *Mach3* “C:\Mach3\”.

Si los parámetros de calibración no se encuentran bien o se utiliza una cámara distinta a las logitech c110, se deberán reestimar estos parámetros generando nuevamente los archivos “Intrinsic.xml” y “Distortion.xml” a través de otra aplicación que se encuentra en el disco anexado bajo el nombre de “\calibration\_params\”. Para esta aplicación será necesario tener impreso un tablero de ajedrez de 9x6 bordes, en el anexo A se adjunta una imagen que se puede utilizar, la cual se deberá encontrar sobre una superficie plana para una correcta calibración.

Además de la imagen, en el anexo se agregó el archivo de imagen del tableo sobre la misma ruta en la que se encuentra el programa de calibración y que también se puede utilizar para la calibración.

Una vez con el tablero impreso se deberá abrir el software de calibración y asignar a primeros los parámetros “Enter the numbers of spanspots =” y “Enter the numbers of frames to skip = “ un valor de 1 en ambos. Luego se posiciona el tablero frente a la cámara hasta que se encuentre la cantidad de bordes que tiene la imagen (deben ser 9x6, en caso contrario no funcionará). De no encontrar nada después de varios intentos, el software intentará encontrar de nuevo los bordes de la imagen pero preguntará la cantidad de éstos que posee la imagen y será necesario agregar la cantidad exacta, si no nuevamente no encontrará nada. Una vez que el tablero es encontrado aparecerá con marcas de colores y el *software* se cerrará creando los archivos Intrinsic.xml y Distortion.xml sobre la misma carpeta. Luego será necesario enviar estos archivos a la carpeta del *Mach3*, ya que el *software fress* obtiene esta configuración desde esa carpeta. La mejor calibración se encontró utilizando una sola imagen así que será mejor no perder el tiempo intentando asignar con distintas combinaciones de imágenes, esto se concluyó

después de varios intentos con 5, 10, 15 y 20 imágenes con el tablero en distintas posiciones.

Para más detalles de la calibración del lente se recomienda revisar la memoria y la referencia [5] desde la web.

### **ii. Calibración de lectura de colores**

La calibración de colores se hará desde la sección *Color calibration* desde el software *fress* tal como se muestra en la figura 40.

La clasificación se hará sobre una transformación de colores HUE (para mayores detalles de esta transformación revisar la memoria). Luego, de acuerdo a los parámetros fijados en los intervalos, se hará una clasificación de todos los pixeles que logren pertenecer a ese rango.

Los parámetros del *threshold* se refieren al delta del intervalo que se utilizará para la selección de colores. Actualmente se encuentra en los valores que se presentan en la figura 40, dado que fueron los mejores resultados obtenidos después de varias pruebas.

Para seleccionar el rango de colores, saturación e intensidad, primero se deberá ejecutar el software mediante el botón *Show*. Luego se debe seleccionar sobre qué eje se desea calibrar, ya sea *Vertical* u *Horizontal* y hacer *click* sobre el punto que posee el color que se desea clasificar. De esta manera automáticamente se seleccionarán los datos del pixel con *valor  $\pm threshold$*  de clasificación. En la figura 41 se presenta un ejemplo de esto. Además se movieron los rangos de saturación y valor aumentándolos para obtener mejor intensidad y saturación. Una vez finalizada la calibración hay que hacer click sobre el botón *save* para que los parámetros queden guardados. Mientras mejor sea esta calibración, más eficiente será la precisión obtenida por el sensor a nivel de subpixeles (actualmente se llega a una precisión aproximada de 0.2 pixeles).

De las pruebas hechas, los mejores resultados se obtienen acotando al máximo el rango de colores *h* y aumentando los rangos de saturación *s* e intensidad *v*.

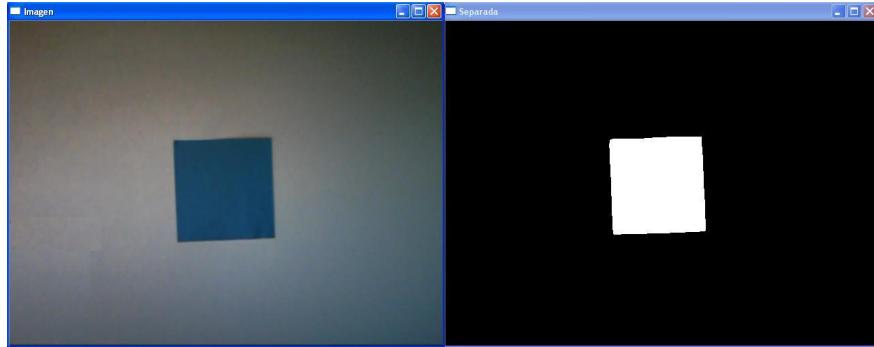


Figura 41: Etapa de calibración de colores (Botón “Show”)

### iii. Calibración de distancias

Esta etapa consiste en relacionar las distancias en pixeles con el movimiento real de la máquina, para esto, será necesario ingresar varios parámetros de calibración y mover la máquina en las distancias pedidas para poder hacer el cálculo de  $n$  de la siguiente fórmula

$$\Delta\text{Distancia en mm} = X * \Delta\text{Distancia en pixeles}$$

El parámetro  $X$  será calculado para cada eje. Será necesario tener el software funcionando a través del botón *Start* y posicionar el círculo lo más al centro posible de la imagen (no es requisito pero si preferencia). Luego se deberá presionar *Set 0* para marcarlo como 0 de referencia y presionar *Calibration* para comenzar la calibración. Al realizar este proceso, aparecerá en la pantalla la imagen de la figura 42. Para las pantallas A y B, el punto deberá estar en el centro, ya que ese se fijará como el 0 de la máquina y luego se posicionará relativamente a ese punto. En el caso de la figura C, se deberá mover el eje X de la máquina en 10 mm, positiva o negativamente. En la figura D se hará lo mismo con el eje Y desde la última posición fijada y para la figura E lo mismo para el eje Z pero con 7 mm. Con eso la máquina ya debería quedar calibrada con la transformación de pixeles a mm. Estos valores quedarán guardados en el archivo “C:\Mach3\pixel2unit.txt”

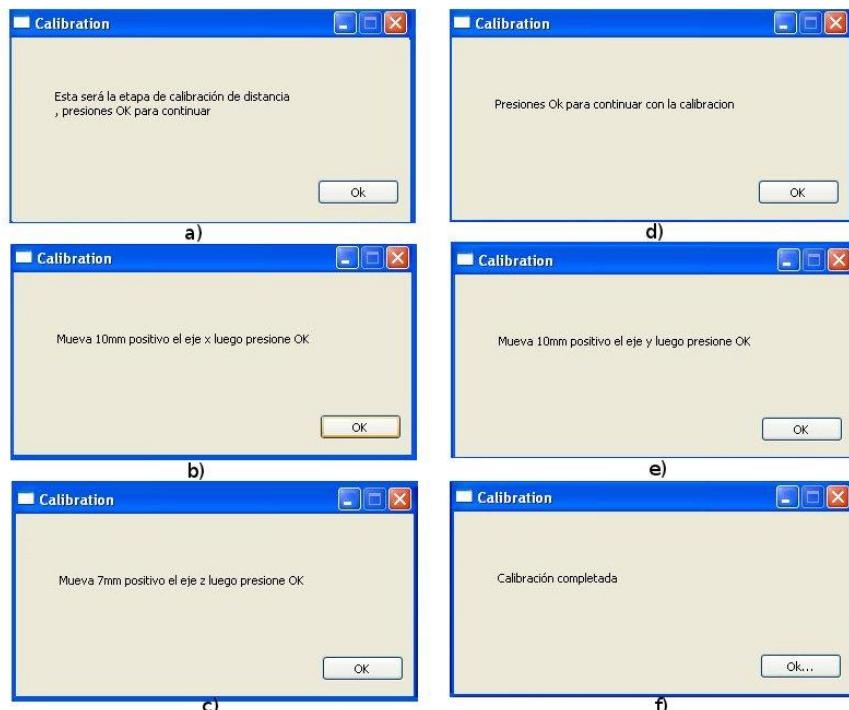


Figura 42: Etapa de calibración de distancias (Botón “Calibration”)

Dado que no es posible corregir el total de las distorsiones del lente, aun queda un error a grandes distancias, por lo que es necesario agregar una nueva calibración para corregirlo al máximo. Primero se fijarán los parámetros tal cual como se muestra en la figura 43 abajo y luego se corregirán de la siguiente manera:

Primero debe moverse a la posición 40 mm en el eje X y 0 en los otros ejes. La posición obtenida en X se guardará en X y la posición obtenida como error en Y se guardará en *ye* y luego se presionará *Save*. Este error *ye* se debe a que el eje cartesiano X-Y no calza perfectamente con el eje de la cámara, por lo tanto cada movimiento en X implicará un movimiento en Y, también su recíproco. Luego de presionar *Save* se podrá ver directamente que el error disminuye prácticamente a 0, de lo contrario se deberá corregir el valor puesto. Con esta corrección se eliminará el error a grandes distancias.

Luego se debe hacer lo mismo para las posiciones  $-x$ ,  $-ye$ , luego  $y$ ,  $x$ , luego  $-y$ ,  $-xe$  y finalmente para  $z$  y  $-z$ . El eje  $z$  no posee corrección de otros ejes ya que no es dependiente ni del eje  $y$  ni del eje  $x$ . Finalmente se pondrá *Save*, guardando los datos para futuras ejecuciones del software. La figura 43 arriba muestra un ejemplo de una calibración final.

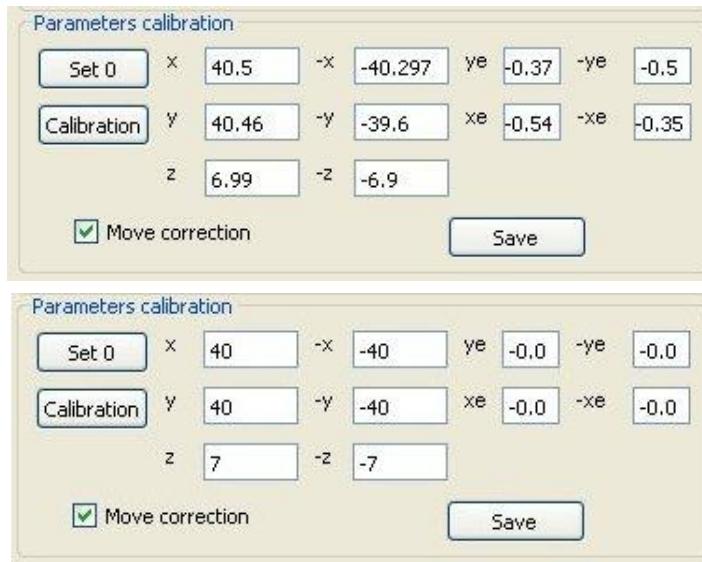


Figura 43: Etapa de calibración con respecto a distancias grandes

#### iv. Filtros

La sección *filter* posee un *checkbox* que corresponde a si se muestra o no la imagen *Ploteo*. Esta imagen corresponde al seguimiento completo de la trayectoria  $x$ - $y$ .

Los otros tres bloques *Imagen*, *Imagen-1* e *Imagen-2*, corresponden a un filtro temporal de la imagen para eliminar los *speckles* temporales que se producen en las imágenes. Estos valores corresponden a un filtro temporal y los 3 valores deben sumar 1 cerrado, es decir:

$$\text{Ponderacion}(\text{Imagen}) + \text{Ponderacion}(\text{Imagen}_{-1}) + \text{Ponderacion}(\text{Imagen}_{-2}) = 1$$

Hasta el momento sólo se utiliza la imagen actual sacrificando error debido a que los tiempos de cálculo de posición se hacen muy largos, lo que hace que sean poco aplicables en la realidad dado que la mayoría de los códigos de Código G posee más de 500 líneas y esto hace que se transforme en un proceso excesivamente lento. Por ejemplo, si el delta en usar las 3 imágenes fuese de 3 segundos, habría que esperar en más de  $500*3 = 1500$  segundos (25 minutos) sólo de retardo de lectura. Estos parámetros quedan libres de cambiar dependiendo de qué es lo que se busque, ya sea precisión o tiempo de ejecución. Los valores razonables serían:

$$\text{Ponderacion Imagen} = 0.4$$

$$\text{Ponderacion Imagen}_{-1} = 0.3$$

$$\text{Ponderacion(Imagen}_{-2}) = 0.3$$

#### v. Pantallas de funcionamiento

En esta sección se explicará y mostrará un ejemplo de los resultados obtenidos por el software en funcionamiento a través del botón *Start*. La figura 44 muestra un ejemplo de lo que ven las cámaras y se dibuja un punto sobre el punto encontrado. Si ese punto varía mucho, se debe a que las cámaras no se encuentran bien calibradas en el filtro de colores.

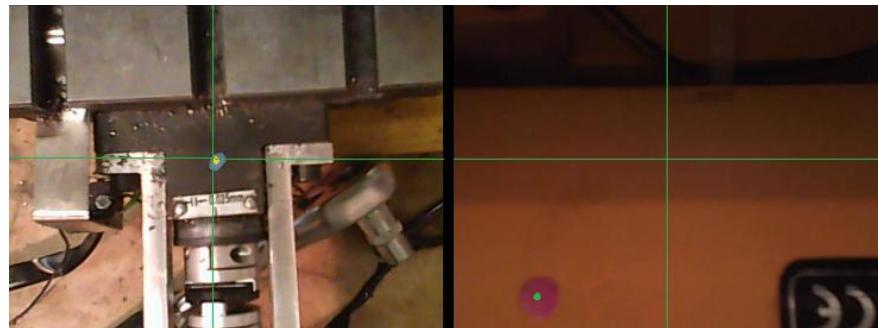


Figura 44: Etapa medición de distancia y comunicación constante (Botón *Start*)

La figura 45 muestra un ejemplo de la trayectoria que siguen las cámaras. Aquí se muestra una trayectoria cuadrada que se obtiene desde el sensor. El movimiento en X-Y representa la posición en el eje X-Y, Y el eje Z se representará por el nivel de intensidad de color azul. Para eliminar los datos pasados de la imagen *Ploteo* será necesario *checkear* el cuadro *Plot\_xyz* dos veces y se reiniciará el gráfico.

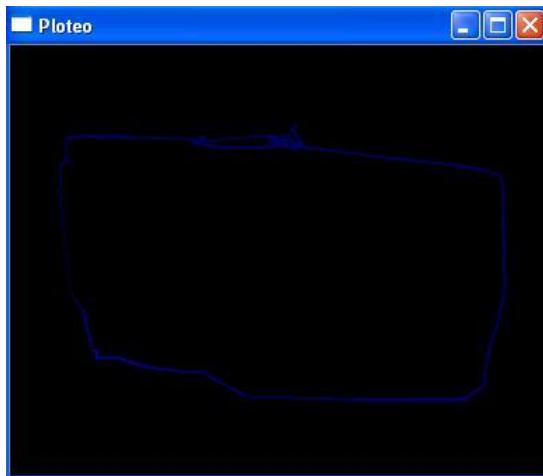


Figura 45: Generación de trayectorias desde el sensor

#### b. Detalles de código y compilación

Para poder compilar este código será necesario tener las librerías de OpenCV con una versión mayor o igual a la V2.1. Además se debe tener instalado el programa Qt by Nokia v4.8.0 (VS2010 OpenSource) y la versión V7.1 de Microsoft Windows SDK. Todos estas librerías y programas podrán encontrarse anexados en el CD dentro de la carpeta /Librerías/. En caso de no poseer este disco se pueden descargar gratuitamente desde internet.

Este código posee dos clases importantes, clase *Fress*: realiza todo el procesamiento interno de imágenes y estimación de las posiciones del objeto de color filtrado. La otra clase es *Calibration*: realiza la calibración de transformación de pixeles a

milímetros de los ejes X, Y, Z. A continuación se explicará cada clase por separado. Se harán diagramas para detallar las funciones principales del software, que se presentarán más abajo.

### 1. *Fress*

A continuación en los diagramas 5 y 6 se presentan diagramas de bloques de las funciones principales del software *Fress*: la función *Show* que es para la calibración del filtro de colores y la función del botón *Start* que es el que realiza la estimación de las posiciones y se comunica con el *Mach3*.

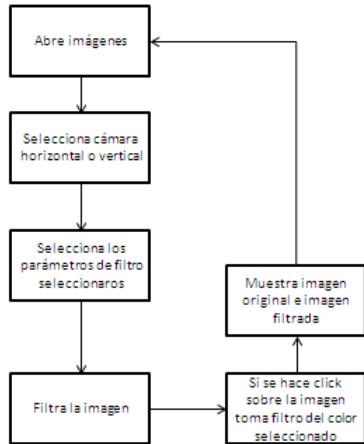


Diagrama 5: Diagrama de bloques de la función *Show* para seleccionar filtro de colores

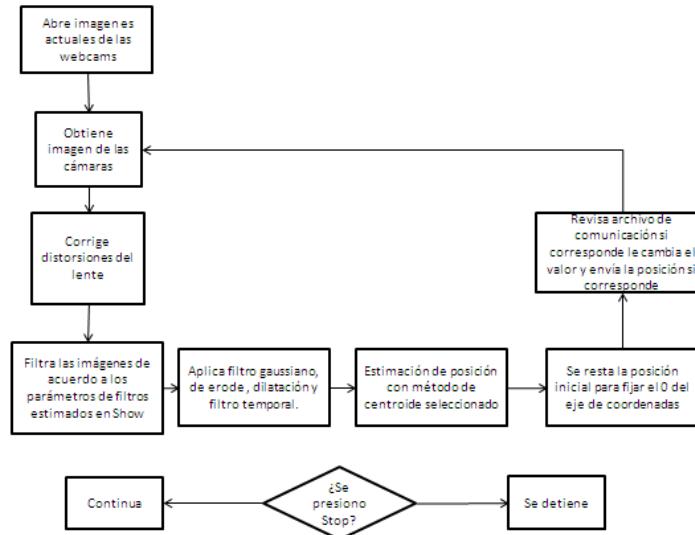


Diagrama 6: Diagrama de bloques de la función Start para determinar posición y comunicarse con Mach3

### Parámetros globales

```

Form w; //Parámetro para mostrar un "About" del software
Calibration z; //Parámetro de calibración
int key;
IplImage* frame_horizontal; //Imagen horizontal
IplImage* frame_vertical; //Imagen vertical
CvCapture* capture_horizontal; //Asignación de la cámara web horizontal
CvCapture* capture_vertical; //Asignación de la cámara web horizontal

//Parámetros de rangos para filtros de colores para hacer una imagen binaria
int h11, h12, h21, h22, s11, s12, s21, s22, v11, v12, v21, v22;

IplImage* imgHSV; //Imagen donde se guardará la imagen en HSV
IplImage* imgFilteredh; //Imagen donde se guardará la imagen filtrada
IplImage* imgFilteredh_bef; //Imagen donde se guardará la imagen filtrada anterior
IplImage* imgFilteredv; //Imagen donde se guardará la imagen filtrada
IplImage* imgFilteredv_bef; //Imagen donde se guardará la imagen filtrada anterior

//Posiciones de x, y, z de la cámara, auxiliares de posiciones y transformación de
//pixeles a mm + corrección de error de que los ejes no son perfectamente
//correspondientes con la posición de la cámara
double xyz[4], xyzaux[3], pixel2mm[5];
ofstream position; //Archivo con la posición x, y, z obtenidos desde la cámara

//Referencias para setear a 0 las posiciones x, y, z
  
```

```
double refx, refy, refz;
int modifiable;

Funciones


/*
 * Constructor que inicializa las variables
 *
 * QWidget *parent, Qt::WFlags flags: Parámetros internos de QT
 */
Fress::Fress(QWidget *parent, Qt::WFlags flags)

/*
 * Funcion que activa el boton "Refrescar"
 */
void Fress::on_posiciones_2_clicked()

/*
 * Funcion para cerrar el programa
 */
void Fress::on_exit_clicked()

/*
 * Funcion que muestra la pantalla about
 */
void Fress::about()

/*
 * Funcion que activa el boton Start y que abre las cámaras, estima la posición del punto
 * y genera la posición final relativa en mm
 */
void Fress::on_Start_clicked()

/*
 * Funcion que detiene el programa al presionar el botón Stop
 */
void Fress::on_Stop_clicked()

/*
 * Funcion que visualiza las camaras de calibracion y permite seleccionar
 * un pixel para filtrar el color al presionar el botón Show
 */
void Fress::on_Show_clicked()

/*
 * Cierra la pantalla de calibracion al presionar el botón Close
 */
void Fress::on_Close_clicked()

```

```

/**
 * Manda a Rojo la calibración de los parametros al presionar el botón R
 */
void Fress::on_R_clicked()

/**
 * Manda a Verde la calibración de los parametros al presionar el botón G
 */
void Fress::on_G_clicked()

/**
 * Manda a Azul la calibración de los parametros al presionar el botón B
 */
void Fress::on_B_clicked()

/**
 * Muestra los parámetros guardados para la imagen vertical
 */
void Fress::rvertical()

/**
 * Muestra los parámetros guardados para la imagen horizontal
 */
void Fress::rhorizontal()

/**
 * Guarda los parametros de las barras al presionar el botón save
 */
void Fress::on_save_clicked()

/**
 * Retorna la posición xy en el arreglo de 3 dimensiones
 * Method = 1, usa una imagen binaria para determinar la posición del centroide
 * Method = 2, usa la intensidad de los pixeles para determinar la posición del centroide
 * Method = 3, usa la intensidad de los pixeles para determinar la posición del centroide
 * + una gaussiana para dar prioridad al centro
 * Method = valores anteriores, usa el método de OpenCV para determinar la posición
 * (Más lento)
 *
 * IplImage* imagexy: Imagen horozntal
 * IplImage* imagez: Imagen vertical
 * IplImage* imageHSVh: Imagen horizontal en hsv
 * IplImage* imageHSVv: Imagen vertical en hsvs
 * double *xyz: Posiciones x, y, z de los puntos de la cámara
 * int method: Método de estimación de centroide, ver memoria para más detalles
 */
void Fress::posxyz(IplImage* imagexy, IplImage* imagez,IplImage* imageHSVh, IplImage*
imageHSVv, double *xyz, int method)

/**
 * Imprime valores en un archivo de texto "String"
 */

```

```

void Fress::write(IplImage* image, string file)

< /**
 * Permite seleccionar un archivo de un directorio y guardarlo en la pantalla de Fress
 * al presionar el botón Open
 */
void Fress::on_Open_clicked()

< /**
 * Función que setea la posición actual como el 0 de fress al presionar el botón set 0
 */
void Fress::on_set_0_clicked()

< /**
 * Lee archivo de configuración y utiliza estos parámetros de configuración
 * del filtro de colores
 *
 * string path: Dirección del archivo de comunicación
 */
void Fress::readfile(string file)

< /**
 * Guarda en archivo los parámetros configuración de los filtros de colores
 *
 * string path: Dirección del archivo de comunicación
 */
void Fress::writefile(string file)

< /**
 * Lee el archivo de comunicación y lo guarda en modifiable
 *
 * string path: Dirección del archivo de comunicación
 */
void Fress::readmodifiable(string path)

< /**
 * Lee el entero de un archivo "path" y lo retorna
 *
 * string path: Dirección del archivo de comunicación
 */
double * Fress::readint(string path)

< /**
 * Escribe 0 en el archivo de comunicación
 *
 * string path: Dirección del archivo de comunicación
 */
void Fress::writemodifiable(string path)

< /**
 * Función para seleccionar colores con clicks
 */

```

```

/*
* int event: Parámetro de eventos
* int x: Posición x en pixeles de selección
* int y: Posición y en pixeles de selección
* int flags: Marcación
* void* param: Memoria de parámetros
*/
void mouseHandler(int event, int x, int y, int flags, void* param)

/**
* Función que llama a la calibración al presionar el botón Calibration
*/
void Fress::on_calibration_clicked()

/**
* Lee los parámetros de transformación desde un archivo
*/
void Fress::read_pixel2mm()

/**
* Apaga white balance si las cámaras lo permiten
*/
void Fress::camera_calibration()

/**
* Configura el parámetro de exposición si la cámara lo permite
*/
void Fress::camera_calibration2()

/**
* Devuelve la media del arreglo
*
* int arrelo[20]: Arreglo a promedia
* int cant_prom: Cantidad de datos a promediar
*/
double mean_all(double arrelo[20], int cant_prom)

/**
* Lee los parámetros de calibración para grandes números
* desde el archivo
*/
void Fress::read_calibparam()

/**
* Guarda los parámetros de calibración en un archivo para
* que la configuración que registrar al reiniciar el software
*/
void Fress::on_savecalib40_clicked()

/**
* Función que lee parámetros de un csv

```

```
 */
void lastposition()
```

## 2. Calibration

### Parámetros globales

```
double* pixel2mm; //Parámetros de transformación de pixeles de mm
double xyz_ini[4]; //Posiciones iniciales de calibración
double pos; //Variables auxiliares
```

### Funciones

```
/*
* Constructor vacío
*/
Calibration::~Calibration()

/**
* Función que despliega la pantalla y función de calibración
*/
void Calibration::on_next_clicked()

/**
* Setea la misma memoria para la posición xyz que la de la clase Fress
*
* double* xyz_s: Puntero a la memoria donde se encuentran las posiciones de * x, y, z
*/
void Calibration::setXYZ(double* xyz_s)

/**
* Setea la misma memoria para la posición pixel2mm que la de la clase Fress
*
* double* pixel2mm_ui: Puntero a la memoria donde se encuentran las
* transformaciones de pixeles a mm de x, y, z
*/
void Calibration::setPixel2mm(double* pixel2mm_ui)

/**
* Escribe los parámetros de transformacion pixel2mm
*
* string path: String con el archivo donde se guardarán los datos
*/
void Calibration::writepixel2mm(string path)

/**
```

```
* Función que estima el parámetro de transformación de pixeles a milímetros  
* y lo guarda dentro de la memoria interna  
**/  
void Calibration::calibration1()
```

## 5. Referencias.

[1] Mach3 Developers. Mach3 CNC Controller Software Installation and Configuration.

[http://www.machsupport.com/docs/Mach3Mill\\_Install\\_Config.pdf](http://www.machsupport.com/docs/Mach3Mill_Install_Config.pdf)

[2] Art Fenerty, John Prentice. Using Mach3Mill.

[http://www.machsupport.com/docs/Mach3Mill\\_1.84.pdf](http://www.machsupport.com/docs/Mach3Mill_1.84.pdf)

[3] Mach3 Developers. Mach Script Language Reference.

[www.machsupport.com/docs/VBScript\\_Commands.pdf](http://www.machsupport.com/docs/VBScript_Commands.pdf)

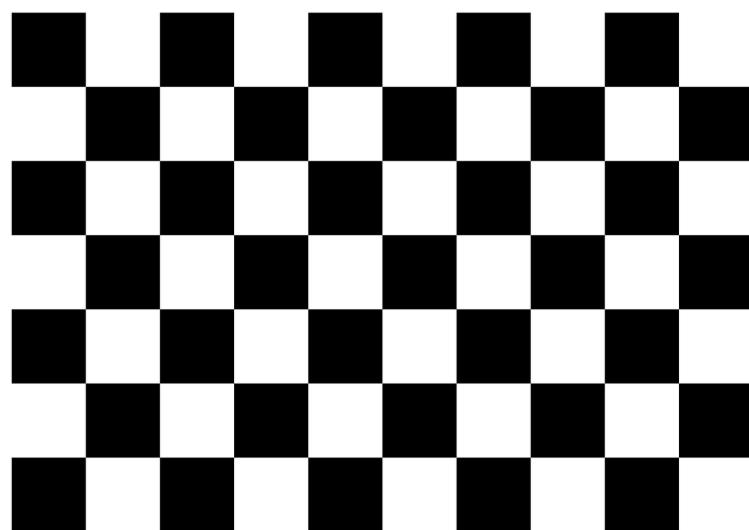
[4] MasterCam Developers , Basic 2D Machining.

[http://www.mastercam.com/Blog/2d\\_Tutorial\\_Sample.pdf](http://www.mastercam.com/Blog/2d_Tutorial_Sample.pdf)

[5] DsynFLO: Camera Calibration using OpenCV.

<http://dsynflo.blogspot.com/2010/03/camera-calibration-using-opencv.html>

**6. Anexo A: *Chessboard* de calibración.**

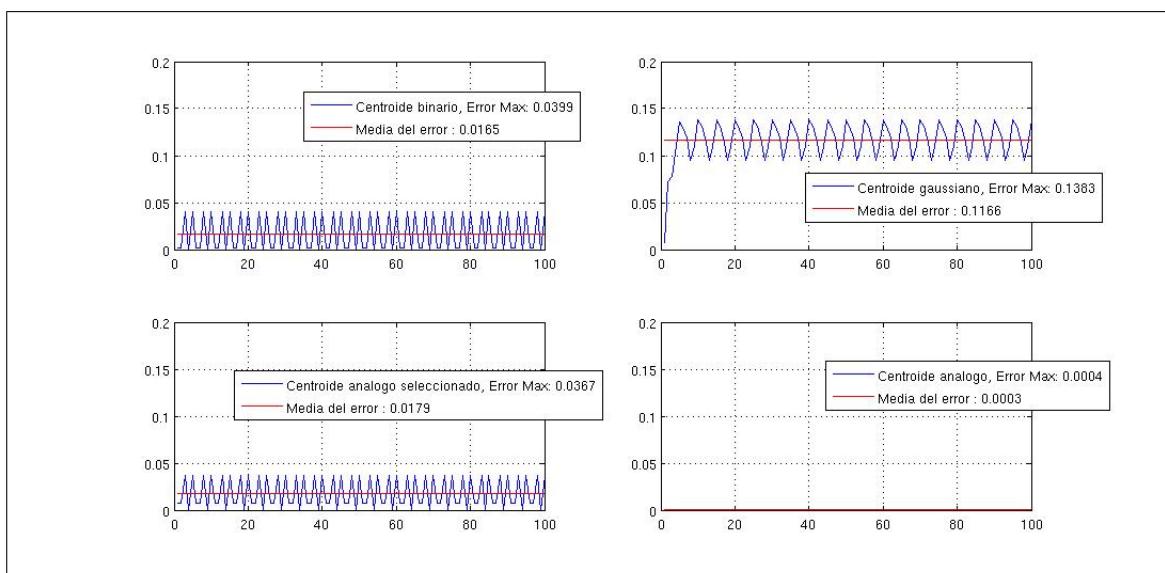


**ANEXO E. SOFTWARE**

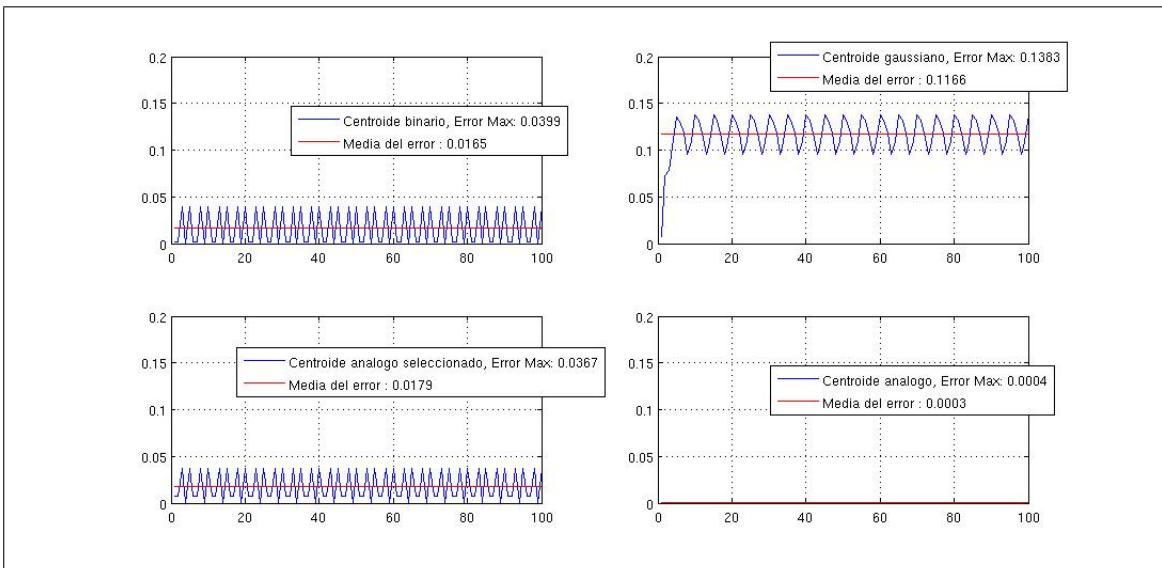
Al final de esta memoria se anexa el CD con todo el contenido de *software*, manual para el desarrollo y uso de la máquina CNC, para más detalle de su contenido revisar el manual de uso.

## ANEXO F. SIMULACIONES

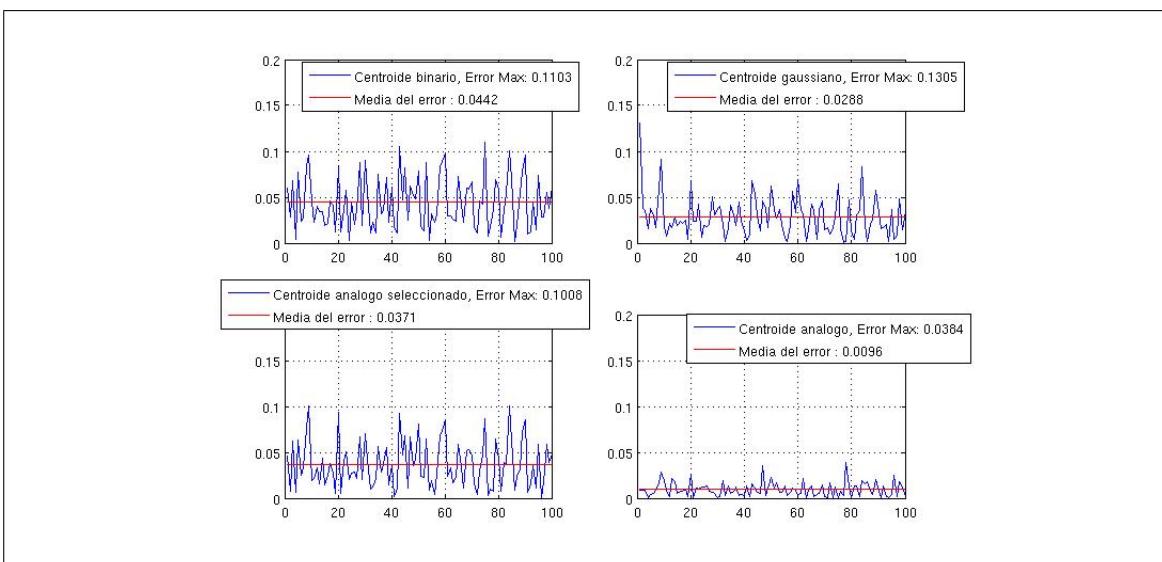
Cada imagen los resultados de las simulaciones representan cuatro situaciones y en cada cuadro se muestra el máximo valor obtenido y un promedio del total de las mediciones del error obtenido en la diferencia de posiciones de centroide real vs posición de centroide medido en el CCD a través de distintos métodos, el primer cuadro muestra los resultados para la medición de centroide con el método de centroide binario, el segundo de arriba utiliza el método de centroide ponderado, el tercero utiliza el método analógico con selección binaria y el último utiliza el método analógico.



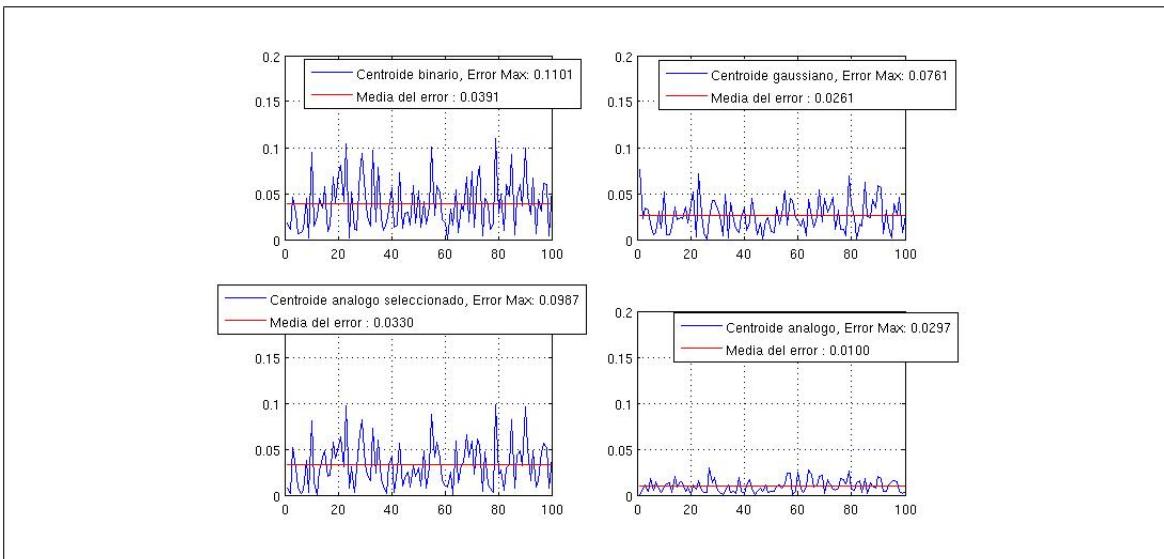
**Figura F.1:** Simulaciones de estimación de centroide sin ruido para el eje X.



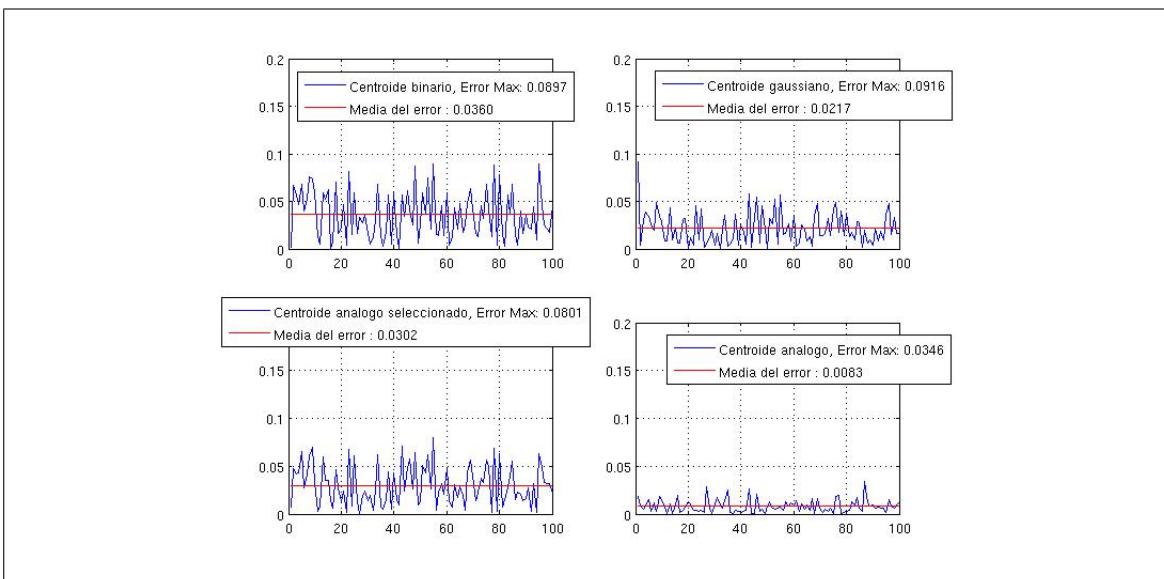
**Figura F.2:** Simulaciones de estimación de centroide sin ruido para el eje Y.



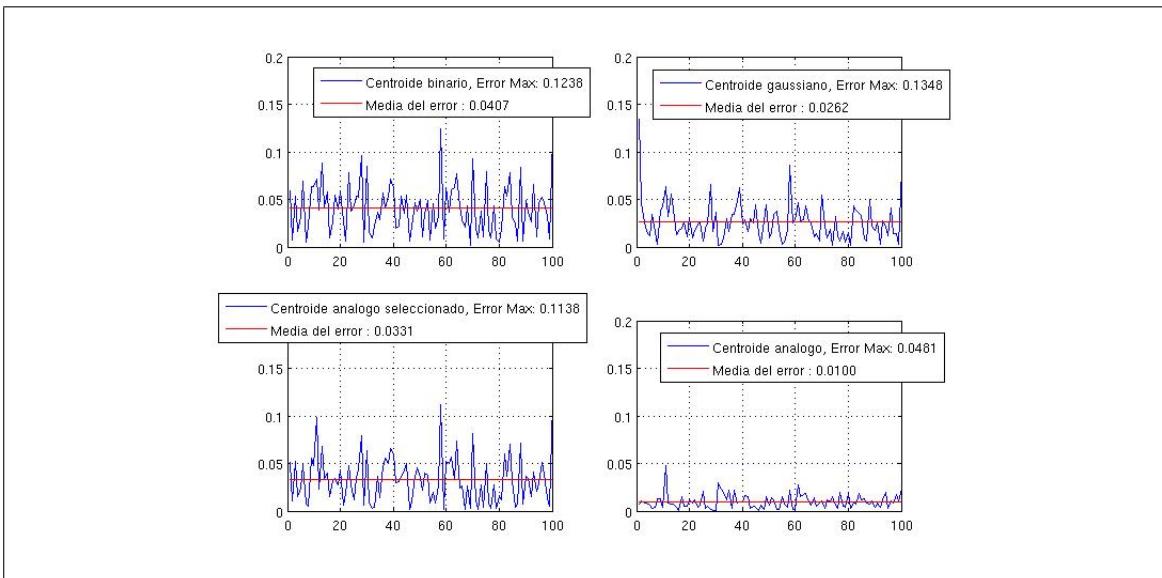
**Figura F.3:** Simulaciones de estimación de centroide con ruido para el eje Y.



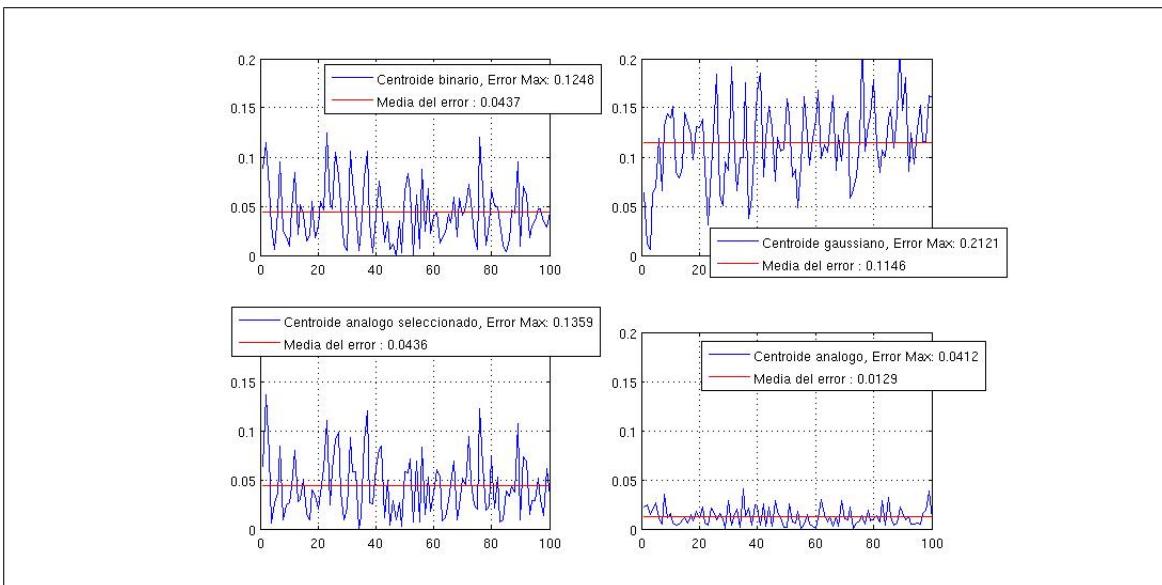
**Figura F.4:** Simulaciones de estimación de centroide con ruido para el eje Y.



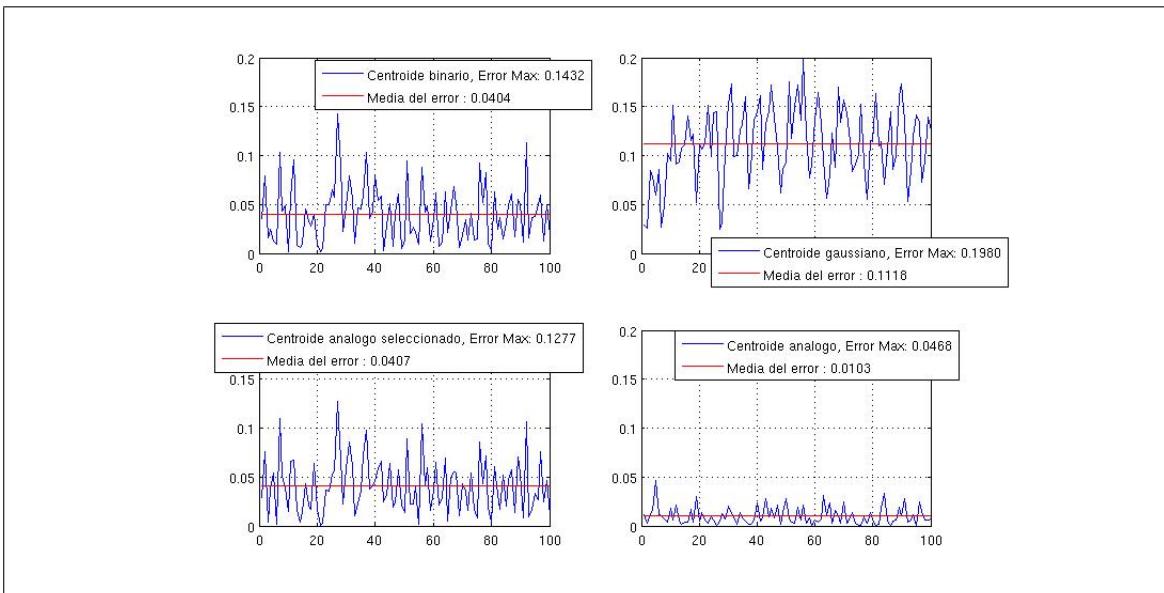
**Figura F.5:** Simulaciones de estimación de centroide con ruido y mitad de luz para el eje Y.



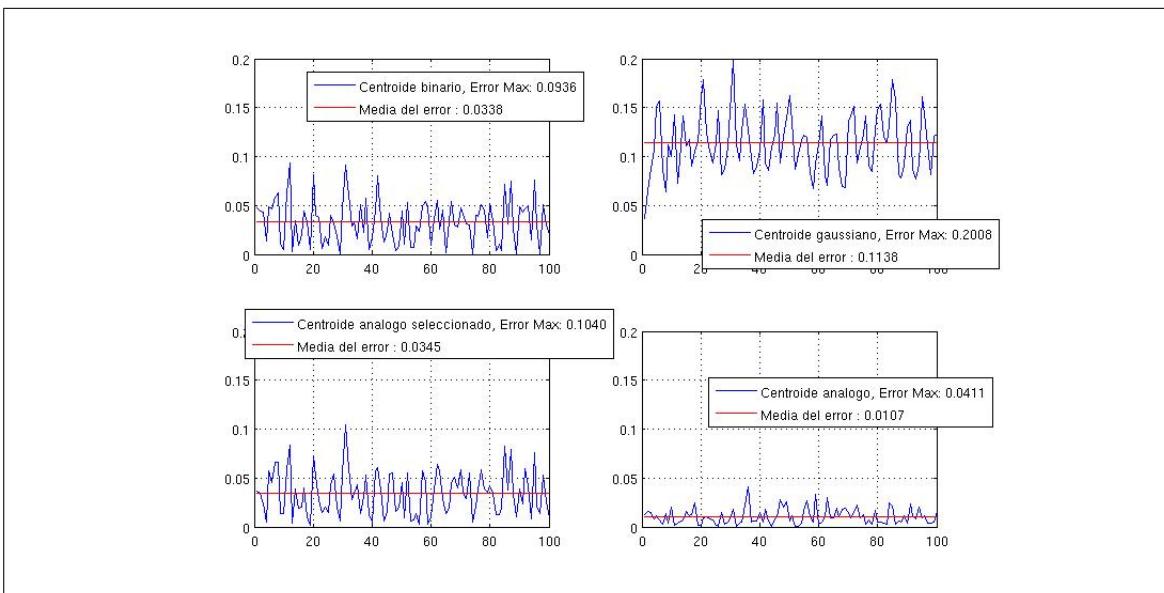
**Figura F.6:** Simulaciones de estimación de centroide con ruido y mitad de luz para el eje Y.



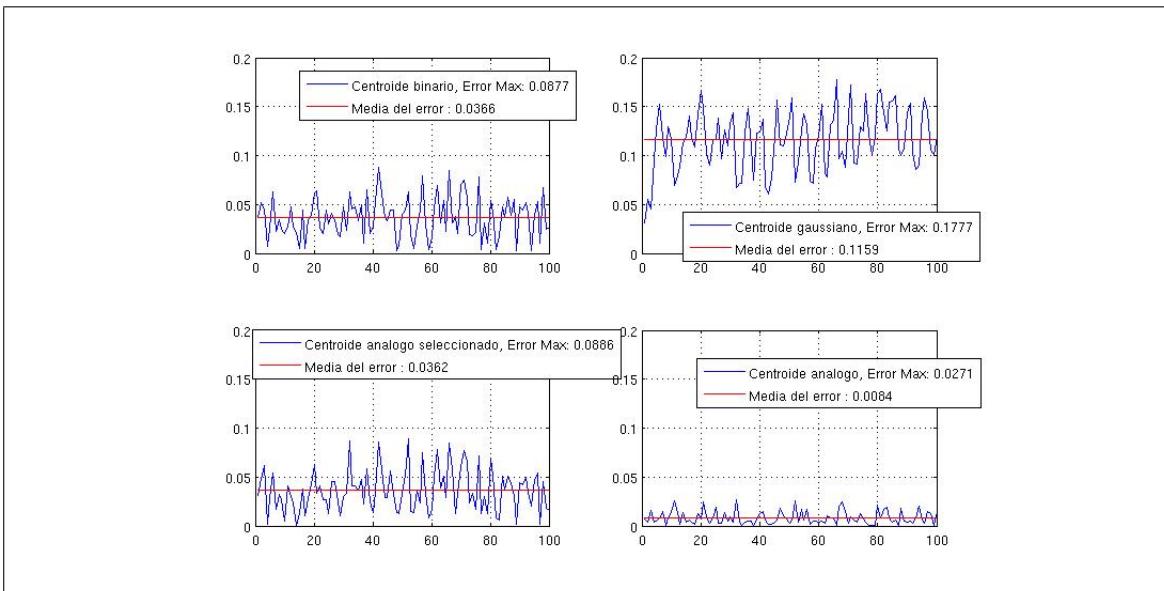
**Figura F.7:** Simulaciones de estimación de centroide con ruido y filtro Gaussiano para el eje X.



**Figura F.8:** Simulaciones de estimación de centroide con ruido y filtro Gaussiano para el eje Y.



**Figura F.9:** Simulaciones de estimación de centroide con ruido, filtro Gaussiano y mitad de luz para el eje X.



**Figura F.10:** Simulaciones de estimación de centroide con ruido, filtro Gaussiano y mitad de luz para el eje Y.