INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

TRABAJO PROFESIONAL

COMO REQUISITO PARA OBTENER EL TÍTULO DE:

INGENIERO ELECTRÓNICO

QUE PRESENTA:

JOSÉ YAVEGNY QUINTERO GARCÍA

CON EL TEMA:

ROBOT PARALELO TIPO DELTA

ASESOR:

M.C. RAÚL MORENO RINCÓN

MEDIANTE:

OPCIÓN I
(TESIS PROFESIONAL)

TUXTLA GUTIÉRREZ CHIAPAS **SEPTIEMBRE DEL 2013**

TRABAJO PROFESIONAL

COMO REQUISITO PARA OBTENER EL TÍTULO DE:

INGENIERO ELECTRÓNICO

QUE PRESENTA:

JOSÉ YAVEGNY QUINTERO GARCÍA

CON EL TEMA:

ROBOT PARALELO TIPO DELTA

ASESOR:

M.C. RAÚL MORENO RINCÓN

MEDIANTE:

OPCIÓN I
(TESIS PROFESIONAL)

TUXTLA GUTIÉRREZ CHIAPAS **SEPTIEMBRE DEL 2013**

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Subsecretaría de Educación Superio
Dirección General de Educación Superior Tecnológic
Instituto Tecnológico de Tuxtla Gutiérre

"2013, Año de la Lealtad Institucional y Centenario del Ejército Mexicano"

DIRECCIÓN
SUBDIRECCIÓN ACADÉMICA
DIVISIÓN DE ESTUDIOS PROFESIONALES
Tuxtla Gutiérrez, Chiapas 23 de mayo 2013

OFICIO NUM. DEP-CT-108-2013

C. **JOSÉ YAVEGNY QUINTERO GARCÍA**
PASANTE DE LA CARRERA DE INGENIERÍA ELECTRÓNICA
EGRESADO DEL INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ.
P R E S E N T E.

Habiendo recibido la comunicación de su trabajo profesional por parte de los CC. M.C. RAUL MORENO RINCON M.C. ANGEL SEIN PÉREZ RODRIGUEZ Y DR. ALEJANDRO MEDINA SANTIAGO. en el sentido que se encuentra satisfactorio el contenido del mismo como prueba escrita, **AUTORIZO** a Usted a que se proceda a la impresión del mencionado Trabajo denominado:

" ROBOT PARALELO TIPO DELTA"

Registrado mediante la opción:
I (TESIS PROFESIONAL)

A T E N T A M E N T E
"CIENCIA Y TECNOLOGÍA CON SENTIDO HUMANO"

Vo. Bo.

M.I. APOLINAR PÉREZ LÓPEZ
ENCARGADO DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

M. en C. JOSÉ LUIS MÉNDEZ NAVARRO
DIRECTOR

C.c.p.- Departamento de Servicios Escolares
C.c.p.- Expediente
I'JLMN/M'APL/l'eeam

Secretaría de Educ. Pública
Instituto Tecnológico
de Tuxtla Gutiérrez
Div. de Est. Profesionales

Carretera Panamericana Km. 1080, C.P. 29050, Apartado Postal 599
Tuxtla Gutiérrez, Chiapas; Tels. (961) 61 54285, 61 50461
www.ittg.edu.mx

AGRADECIMIENTOS:

A DIOS, Por brindarme la vida, salud y guía para llegar a ésta etapa tan importante. ¡Alabado seas por siempre Señor! ...

A MIS ASESORES Y REVISORES, Por sus aportes, guía y atenciones brindadas para llegar al final de este trabajo profesional.

AL CICATA, En especial al Dr Castillo y al M.T.A. Álbert B., quienes me permitieron trabajar con el Robot en las instalaciones para llevar a cabo ésta Tesis.

A MIS PADRES, Rubisel Q. y María Soledad G., por brindarme los cimientos y el apoyo constante desde siempre, sin escatimar esfuerzo, para lograr esta meta en mi vida…

A MIS ABUELITOS, Por sus recomendaciones, apoyo, ánimos y comprensión durante todo el proceso, A Papá Marcos Q.E.P.D. a mamá Chonita C., a papá Julio G. y mamá Celia M. y todos los familiares que me dieron atenciones y han brindado cariño y apoyos.

A MIS PADRINOS, Juanita, Sinar y primas, Quienes con su esfuerzo, confianza y guía me ayudaron a escalar estos últimos peldaños abriéndome sin restricciones su hogar.

A MIS AMIGOS: Margarita, Hugo, Darinel, Toledo, por su amistad, constante compañía y apoyo más que amigos son hermanos que a diario nos vimos y compartimos muchas experiencias en nuestra preparación. Y finalmente con broche de oro a la Mtra. Adriana L. por su cariño, apoyo incondicional y compañía en cada paso dado juntos.

¡A TODOS GRACIAS!

ROBOT PARALELO TIPO DELTA

RESUMEN

El Centro De Investigación En Ciencia Aplicada Y Tecnología Avanzada (CICATA) es un centro de investigación científico y tecnológico del Instituto Politécnico Nacional (IPN), ubicado Cerro Blanco No. 141. Col. Colinas del Cimatario, Santiago de Querétaro, Querétaro que desarrolla diversos dispositivos capaces de resolver problemas y realizar actividades que al ser humano se le dificulte hacer, permitiendo así mismo mejorar, agilizar y aumentar procesos.

En el laboratorio de mecatrónica del CICATA se tiene la línea de investigación en mecatrónica en la cual los investigadores trabajan principalmente en temas relacionados con el análisis y simulación, el diseño de mecanismos, la robótica, los procesos industriales, la instrumentación, prototipos y la opto-mecatrónica.

Uno de los dispositivos desarrollados como parte de una línea de investigación, es un robot paralelo tipo delta, de tres grados de libertad, el cual a diferencia de un robot serial, cuenta con grandes ventajas que permite su escalamiento a nivel industrial para la automatización de procesos y así aumentar la productividad en este sector.

Se presenta el desarrollo de un robot paralelo tipo delta, desde el diseño hasta la construcción, integración de los controladores y la instrumentación con equipo industrial, así como la programación de los controladores mediante una interface de usuario con el seguimiento de trayectorias, simuladas y ejecutadas por el robot, dentro de su espacio de trabajo para visualizar los índices de desempeño del mecanismo.

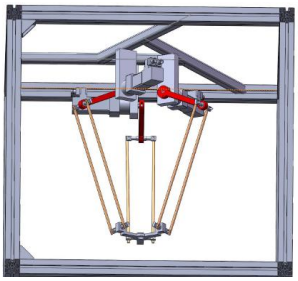


Ilustración 1.- Robot paralelo

Contenido

ÍNDICE DE ILUSTRACIONES

CAPÍTULO I. INTRODUCCIÓN

1.1.-PROBLEMAS A RESOLVER

El problema persistente es causado a que este tipo de robots, cuentan con un espacio de trabajo muy limitado, sin embargo, el rediseño de esta estructura permite alcanzar un espacio de trabajo mayor al logrado en trabajos anteriores e implementándole servomotores que den mejoras en la velocidad, precisión y repetitividad como índices de desempeño.

Así mismo se requiere de la Generación de trayectorias en función de la cinemática del robot, con la implementación de servomotores de la marca Baldor y una interface en un lenguaje de programación, en las que se controle el robot y pueda formar rutinas dentro del espacio de trabajo para realizar movimientos controlados.

Como parte del objetivo general del proyecto, se necesita armar el robot, ya que el diseño es documental y solo se cuenta físicamente con los brazos y antebrazos como únicos materiales. Así como mandar a fabricar las piezas restantes como la estructura de la base fija. Conectar y hacer un solo módulo para el control, con los drivers y el controlador principal del sistema.

Programar una interface gráfica que permita realizar los algoritmos del modelo inverso del robot, y visualizar de manera clara el estado de los motores, como posición y despliegue de errores en caso de tenerlo. Hacer un algoritmo de referencia para posicionar los motores respecto al 0 o posición inicial del robot, conocido como Homing, que permita hacerlo de manera automática.

Así mismo, se requiere de la Generación de trayectorias en función de la cinemática del robot, con la implementación de servomotores de la marca Baldor. En las que el robot pueda formar figuras o bien alguna rutina dentro del espacio de trabajo para realizar movimientos controlados.

Realizar algoritmos para dar seguimiento de las trayectorias definidas, es decir hacer una simulación para cada trayectoria, usando Matlab®, en donde se visualice las rutas o los puntos a seguir durante el desarrollo de la trayectoria.

La integración del control al sistema robótico consiste en agregarle los elementos que necesita el robot para que se desempeñe en sus tareas asignadas o requeridas, así como instrumentarlo para complementar su correcto funcionamiento, es decir adicionar los sensores, actuadores y controladores necesarios en todo robot para su operación.

1.2.-HIPOTESIS

Con la construcción del robot se ampliará el espacio de trabajo con respecto a otros modelos existentes en el mercado así mismo se trabajará con velocidades mayores que permitan desplazar al robot más rápidamente en el espacio de trabajo prescrito. La interface permitirá tener un mejor control del robot con los dispositivos de uso industrial.

1.3.-JUSTIFICACION

Con la implementación del nuevo sistema de control usando Servomotores Baldor, sus respectivos controladores, y la interface gráfica, favorecen al robot en índices de desempeño, como velocidad, aceleración y precisión, y así lograr su escalamiento a otros sectores como el industrial.

Esto mas el modelo rediseñado del manipulador paralelo, como parte de una línea de investigación de maestría y doctorado, el cual presenta un espacio de trabajo mayor a otros prototipos, lo convierte en un candidato para realizar tareas exigentes gracias a sus ventajas para operaciones de manufactura.

Además la mayoría de los laboratorios que han desarrollado investigación en este campo, se han limitado en un marco exclusivamente teórico, siendo muy pocos los construidos para aplicaciones industriales, dándole otro punto a favor al proyecto.

1.4.-OBJETIVOS

1.4.1.-OBJETIVO GENERAL

Instrumentar y controlar un robot paralelo de tres grados de libertad para tareas de posicionamiento de alta velocidad y precisión

1.4.1.-OBJETIVOS ESPECÍFICOS

- Generación de trayectorias de motores DC, en función de la cinemática del robot. Obtener y simular el posicionamiento del sistema mediante algoritmos para ver posible reacciones del dispositivo, con el uso de actuadores del prototipo.
- Seguimiento de las trayectorias definidas. Simulación y revisión de las trayectorias, análisis de posicionamiento y problemas a fines.
- Integración del control al sistema robótico. Construir e instrumentar el robot, implementando el control y verificar la respuesta real del sistema
- Validación experimental. Verificar el funcionamiento del sistema, sintonizar los controladores avances con pruebas de ejecución de trayectorias.

1.5.-ALCANCES Y LIMITACIONES

Con el desarrollo de esta parte del proyecto se obtuvieron distintos productos, entre ellos destaca la interface de control de movimiento para el robot, pudiendo así mismo manipular otros motores para el controlador, como ejemplo la implementación que se le hará al robot llamado “Cicabot”, otro desarrollo vigente en el laboratorio de mecatrónica.

Se diseñó una interface gráfica para controlar los motores con las restricciones del robot, tomando los parámetros que definen el comportamiento del robot como velocidad, aceleración y desaceleración, además considerando a que existen singularidades que provoquen coaliciones entre la estructura por movimientos incorrectos, así como problemas provocados por fricción y la inercia del mecanismo.

También se logró implementar un sistema multi-eje para el control de motores de la marca Baldor, métodos de sintonización con control PID de estos drivers y un manual de usuario para el manejo de los motores ya sea con la interface o con el programa que proporciona el fabricante y así usar n drivers que se necesiten.

Sin embargo durante el desarrollo del proyecto se encontraron distintas complicaciones que detenían el avance del proyecto, como lo son problemas con el programa proporcionado con el fabricante de los motores, al no aceptar algoritmos complejos como el cálculo del modelo inverso del robot, y no poder acceder a funciones trigonométricas con matrices.

Problemas por licencias para el uso de otro compilador como el caso del programa LabView, por lo que se corrigió usando el programa Visual Basic ® para realizar la aplicación de control, ya que el CICATA cuenta con las licencias correspondientes para este programa, que no solo se logra acceder al controlador mediante los componentes y métodos de Active X, si no que permite resolver el modelo inverso, pudiendo complementar con la interface gráfica para el usuario.

Fallas en la fabricación de las piezas para armar el robot fue otra de las complicaciones, así como la demora para la entrega o para el acceso, fueron otras de las problemáticas que no permitían avanzar oportunamente con el ensamblaje del prototipo, causando las piezas con errores un mal desempeño en la ejecución de movimientos.

Aunado a esto, el sobrepeso en la estructura un problema fuerte, ya que los motores tienden a oscilar por la carga excesiva, se intentó compensar el error mediante el uso de contrapesos, pero al aumentar el número de coaliciones con la estructura, se cambió por resortes como alternativa y así evitar sobrecargar los motores, y recurriendo a hacer sintonizaciones distintas en cada punto, ya el resorte es una ayuda tentativa en lo que se diseñan y maquilan los contrapesos requeridos para mover el robot correctamente.

Los problemas mecánicos por manufactura son comunes ya que no se tiene el control total de la correcta fabricación de las piezas, siendo un problema fuerte que en repetidas ocasiones detiene el avance proyecto, sin embargo se puede hacer una validación experimental para verificar el funcionamiento del sistema, de ser necesario sintonizar los controladores continuamente para poder realizar sus tareas como la ejecución de trayectorias.

Finalmente se observó que uno de los motores (Axis 4), cuenta con el motor-reductor dañado, ya que al momento de agregarle una perturbación. El lazo cerrado no lo detecta, siendo un otra problemática que se resolverá al remplazarle tal pieza, al no tiene la capacidad de retener la posición física que se le asigna, ya que el encoder marca la cuenta correspondiente, afirmando que el puro motor en si está en buenas condiciones para su uso.

Al sumar todas las limitantes podrían darse demoras que afecten la parte de la integración del control al robot o la validación, por lo cual, con tener los productos anteriormente planteados, como la interface de control y los manuales, resulta favorable para darle seguimiento al proyecto.

CAPÍTULO II. MARCO TEÓRICO

2.1.-ROBOT PARALELO

La mecatrónica es un área multidisciplinaria que relaciona la ingeniería mecánica, electrónica, eléctrica, matemáticas y sistemas computacionales enfocados a la industria, la investigación, la exploración entre otros campos de aplicación que resuelvan problemas de automatización y mejora de procesos.

Es por ello que en el Centro De Investigación En Ciencia Aplicada Y Tecnología Avanzada (CICATA) se está trabajando en el desarrollo de diversos dispositivos capaces de resolver problemas y realizar actividades que al ser humano se le dificulte hacer, ya sean procesos repetitivos o bajo condiciones y medios hostiles, permitiendo así mismo mejorar, agilizar y aumentar tales procedimientos.

Uno de los dispositivos desarrollados como parte de una línea de investigación en el CICATA, es un robot paralelo tipo delta, de tres grados de libertad, el cual a diferencia de un robot serial, cuenta con grandes ventajas que permite su escalamiento a nivel industrial para la automatización de procesos y así aumentar la productividad en este sector.

Las investigaciones previas con esta estructura, han dejado grandes aportaciones para el rediseño que se abordará en este trabajo, por ello a continuación se describen los más importantes.

El robot Parallix LKF 2040, manipulador con estructura de mecanismo paralelo, fue creado con fines didácticos, para satisfacer las necesidades de competencia educativa, facilitando así una herramienta para interactuar en las disciplinas de la mecatrónica, y una mejor alternativa en cuanto a costos, para la adquisición de unidades robóticas .

El primer prototipo contó con una electrónica diseñada y construida para el control de servomotores SANYO de 3,000 rpm con enconder de 1000 ppr con reductor Harmonic Drive 100:1, controlando tres motores simultáneamente. Para realizar tareas de posicionamiento se recurrió a rediseñar el controlador con la tarjeta PMAC de Delta Tau Data Systems.

Se obtuvo el modelo geométrico inverso del robot, calculando el volumen del espacio de trabajo, haciendo un barrido de coordenadas, para los puntos alcanzables, por la plataforma móvil.

Las modificaciones más recientes de este robot, fue agregarle un control con tarjetas PIC-Servo, con un sistema de ventilación ya que presentaba sobrecalentamiento, y un efector final usando una ventosa de la marca Festo. El control se hizo conectado las tarjetas pic-servo, a una computadora, agregándole el modelo geométrico inverso para definir sus trayectorias.

2.1.1 Definición

Un robot paralelo es un mecanismo de cadena cinemáticas en lazo cerrado, cuyo efector final es unido a la base por varias cadenas cinemáticas independientes). Una cadena cinemática es un ensamble de segmentos o eslabón unidos por una articulación. (Ceccarelli,2006)

Los robots paralelos son poco conocidos, debido a que la mayor parte de la robótica se enfoca a los tipos seriales, los cuales son mecanismos de cadena abierta, con eslabones que actúan generalmente en serie, uno a continuación de otro, con amplio espacio de trabajo, buena maniobrabilidad pero poca rigidez así como mayor inercia, características que los robots paralelos se ve disminuida.

Con esta arquitectura, se tiene en cada cadena una articulación activa, esto es con un solo actuador, siendo las demás articulaciones pasivas, permitiendo posicionar los actuadores en la estructura fija o base del robot, haciendo que los eslabones del manipulador sean más ligeros ya que no soportan el peso del actuador, y así la realización de operaciones a alta velocidad.(Márquez,2002)

Algunas definiciones principales que se usarán a lo largo del documento se detallan a continuación para mejor compresión de los temas, tales conceptos son:

Grado de libertad: Cada uno de los movimientos básicos que definen la movilidad de un determinado robot. Puede indicar un movimiento longitudinal o de rotación.

Cinemática del robot: Estudia el movimiento del mismo con respecto a un sistema de referencia. Así, la cinemática se interesa en la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

Modelo geométrico directo (Cinemática Directa): Determinar la posición y orientación del extremo del robot, con respecto a un sistema de coordenadas de referencia, conocidos los valores numéricos de las variables y los parámetros geométricos de los elementos del robot.

Modelo geométrico inverso (Cinemática Inversa): Determinar la configuración que debe adoptar el robot para alcanzar una posición y orientación conocidas. Consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot. Para que su extremo se posicione y oriente según una determinada localización espacial. Pueden existir soluciones múltiples para estos valores.

Espacio de trabajo: El espacio de trabajo se refiere al volumen dentro del cual puede desplazarse el actuador. Las coordenadas de los puntos en el espacio a las cuales puede llegar el actuador mediante algún movimiento posible de las cadenas cinemáticas.

La invención de la estructura del robot, data de 1949, cuando Eric McGough inventó una máquina para probar llantas. Posteriormente Alan Stewart, utilizó dicha estructura como simulador de vuelos, divulgándolo, por lo que a él se atribuye su origen.

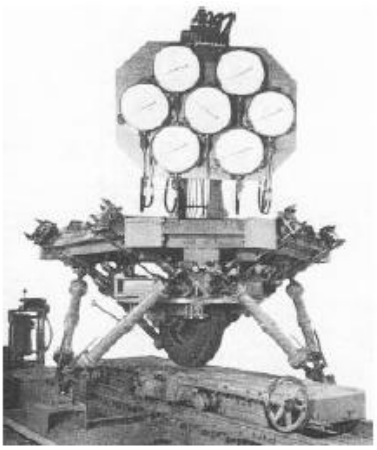


Ilustración 2. Primer manipulador paralelo de Eric Gough

En 1990, R. Clavel realizó una modificación al manipulador Stewart que presentaba desventajas respecto con su difícil análisis en la cinemática, un elevado costo de fabricación por las uniones esféricas, llegando a ser el manipulador Delta de tres y cuatro grados de libertad. Su objetivo fue realizar tareas de alta velocidad, tareas de carga y descarga. (Balmaceda, 2011)

En los estudios de estas estructuras se enfocan en encontrar nuevas configuraciones mecánicas con diferentes actuadores, para incrementar las ventajas, como facilitar el modelo, resolver problemas con su estudio cinemático, propiciar una construcción sencilla y permitir altas velocidades y aceleración.

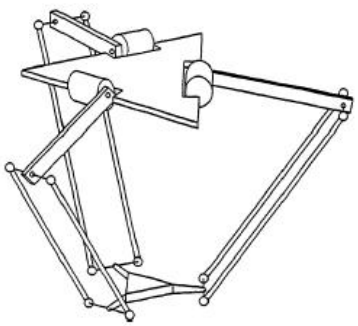


Ilustración 2. Robot paralelo delta **AdeptTecnology Inc.**

En el mercado internacional se ofertan robots paralelos para la industria, como lo es el Delta AdeptQuattro S65H, de la compañía AdeptTecnology. El cual tiene gran alcance y desempeño. Otro robot comercializado es el IRB 360 de Flex Picker, también cuenta con buen espacio de trabajo.

2.1.2Ventajas y desventajas del manipulador

Las ventajas son por mucho mayores al tener una relación de carga- potencia alta, gracias a que los accionamientos de potencia conectan directamente la base del motor al efector final, permitiéndole manipular cargas más grandes que los robots seriales.

Presentan alta rigidez, lo que implica mayor precisión que los seriales, ya que la base fija y la base móvil forma una cadena cinemática cerrada.

Arquitectura de alta velocidad y precisión en tareas de posicionamiento. La carga está distribuida en los actuadores, usando el torque de cada motor para acelerar la carga. Presenta masa móvil reducida, al colocar actuadores en la base fija, evitando ser parte de la carga, pudiendo así seleccionar motores con reductores sin restricciones de peso. (Castañeda, 2003).

Sin embargo como se ha marcado, el espacio de trabajo de estos manipuladores es muy reducido, por lo que un buen diseño puede garantizar alto desempeño en un espacio de trabajo determinado en el caso de aplicaciones industriales, donde se requiere un espacio de trabajo competente.

La definición de las posiciones singulares y la construcción del modelo cinemático son más complejas que para el caso de los robots seriales, aunado a que tiene poca difusión sobre todo en el país.

2.2.-ANALISIS DEL MODELO

2.2.1 Configuración del robot

El robot con el que se trabajó, es un sistema mecánico cuyo órgano terminal, conocido como plataforma móvil, está conectado a su base a través de tres cadenas cinemáticas independientes, teniendo dos eslabones articulados. La plataforma móvil será desplazada en su espacio de trabajo.

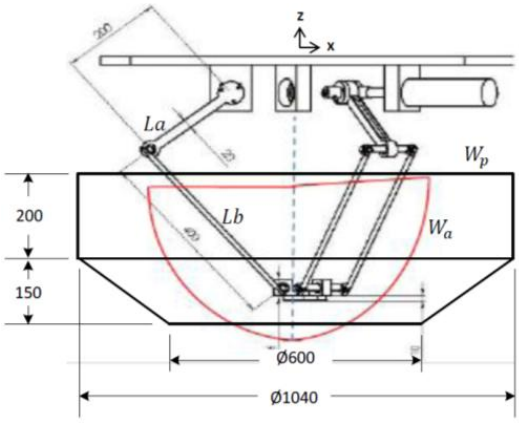


Ilustración 3. Rediseño del Robot Paralelo

El robot tiene 3GDL trasnacionales y consiste en una plataforma fija en forma de triangulo equilátero, donde van instalados los motores que transmiten el movimiento a la base móvil. Cada brazo está formado por 2 eslabones, siendo estos el brazo y antebrazo, una articulación rotacional y dos esféricas. (Balmaceda, 2011).

El primer brazo es de longitud La, fabricada en una sola pieza, anexando un acople al motor, el segundo eslabón es de longitud Lb, formado por barras paralelas. Los ángulos de giro de los brazos se definen como $a_{1i}a_{2i}a_{3i}$;

Ilustración 4. Eslabones de una cadena

2.2.2 Modelo geométrico inverso

La cinemática directa, se ocupa de la descripción del movimiento sin tener en cuenta sus causas, a través de sus procedimientos matemáticos y leyes físicas. Sin embargo el objetivo de la cinemática inversa consiste en encontrar el gesto que deben adoptar las diferentes articulaciones para que el final del sistema articulado llegue a una posición concreta.

Consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $\mathrm{q} = [\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_\mathrm{n}]$para que su extremo se posicione y oriente según una determinada localización espacial.

Al contrario que el problema de cinemática directa, el cálculo de la cinemática inversa no es sencillo ya que consiste en la resolución de una serie de ecuaciones fuertemente dependiente de la configuración del robot, además de existir diferentes $\mathrm{n} - \mathrm{nq} = [\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_\mathrm{n}]$que resuelve el problema.

La acción individual de cada una de las articulaciones debe ser controlada a fin de ejecutar la rutina de movimiento deseada. Específicamente, la ubicación espacial del efector final puede ser determinada mediante el conocimiento del desplazamiento traslacional o rotacional de cada articulación con respecto a una posición de referencia que determina la posición cero o de origen del robot.

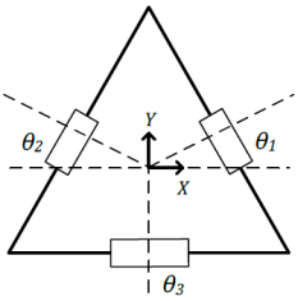


Ilustración 5. Base superior del robot

El modelo matemático del robot se definió en (Velázquez 2003), donde los ángulos de los giros de lo brazos se define con a_1,a_2 y a_3, asi como el ángulo entre motores en la base fija es $\theta_i(i = 1, \dots 3)$ siendo $\theta_1 = 30°, \theta_2 = 150°, \theta_3 = 270°$, según la ilustración 7.

Sea el punto P las coordenadas$[X_p, Y_p, Z_p]$ el centro de la base móvil del manipulador,

$$x_p = cos\theta_i(r + Lacos\alpha_{1i} + Lbcos\alpha_{3i}\cos(\alpha_{1i} + \alpha_{2i}) - rpm) - Lbsen\theta_i sen\alpha_{3i}$$

$$y_p = sen\theta_i(r + Lacos\alpha_{1i} + Lbcos\alpha_{3i}\cos(\alpha_{1i} + \alpha_{2i}) - rpm) - Lbcos\theta_i sen\alpha_{3i}$$

$$z_p = Lacos\alpha_{1i} + Lbcos\alpha_{3i}\cos(\alpha_{1i} + \alpha_{2i})$$

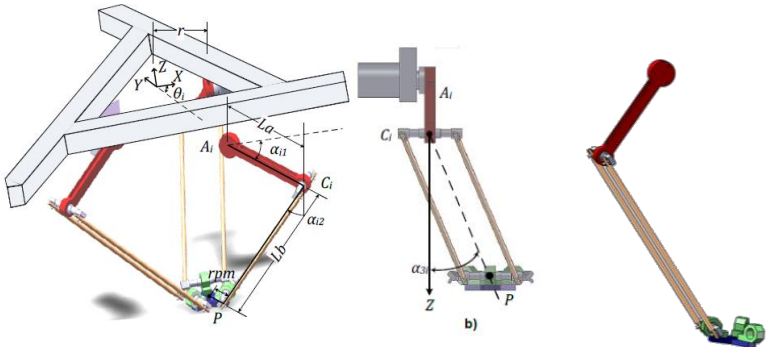


Ilustración 6.Cadena cinemática del robot

Elevando al cuadrado las anteriores, se tiene:

$$\left[(R + Lacos\alpha_i)cos\theta_i - x_p\right]^2 + [(R + Lacosa_i\, sen\theta_i - yp]^2 + \left[-Lasena_i - z_p\right]^2 - Lb^2 = 0$$

Formación del ángulo α_{3i}en la cadena cinemática. Las coordenadas de Ci según la figura del robot son:

$$C_i = \begin{bmatrix} R + Lacos\alpha_i \\ 0 \\ -Lasen\alpha_i \end{bmatrix}$$

Donde $R = r - rpm$; sustituyendo con X,Y, Z queda:

$$C_i = \begin{bmatrix} X \\ 0 \\ Z \end{bmatrix}$$

Matiz de rotación en base a θ_i

$$Rot(Z, \theta_i) = \begin{bmatrix} cos\theta_i & -sen\theta_i & 0 \\ sen\theta_i & cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplicando las matrices anteriores queda:

$$C_i = \begin{bmatrix} cos\theta_i & -sen\theta_i & 0 \\ sen\theta_i & cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ 0 \\ Z \end{bmatrix} = \begin{bmatrix} Xcos\theta_i \\ Xsen\theta_i \\ Z \end{bmatrix}$$

Debido a que Ci pertenece a una esfera por tal ecuación entonces queda:

$$\left(Xcos\theta_i - X_p\right)^2 + \left(Xsen\theta_i - Y_p\right)^2 + \left(Z - Z_p\right)^2 = Lb^2$$

Factorizando y sustituyendo por $X = r + Lacos\alpha_i$, $Z = Lasen\alpha_i$ y simplificando:

$$2RLacos\alpha_i - Lacos\alpha_i\left(2x_p cos\theta_i + 2y_p cos\theta_i\right) + 2Z_p Lasen\alpha_i =$$

$$Lb^2 - La^2 - R^2 - X_p^2 - Y_p^2 - Z_p^2 + R(2X_p\, cos\theta_i + 2Y_p cos\theta_i)$$

Dividiendo entre La y sustituyendo

$q_i = 2X_p\, cos\theta_i + 2Y_p cos\theta_i$ y $s = \frac{1}{La}(Lb^2 - La^2 - R^2 - X_p^2 - Y_p^2 - Z_p^2)$

Queda entonces

$$(2R - q_i)cos\alpha_i + 2Z_p sen\alpha_i = s + \frac{q_i}{La}R$$

Ahora bien aplicando identidades trigonométricas y resolviendo para $\tan\left(\alpha\frac{1}{2}\right)$:

$$tan\frac{\alpha_i}{2} = \frac{\left(-2Z_p \pm \sqrt{4Z_p^2 + 4R^2 - s^2 - q_i\left(4R + \frac{2sR}{La}\right) + q_i^2\left(1 - \frac{R^2}{La^2}\right)}\right)}{-2R - s - q_i(\frac{R}{La} - 1)}$$

Por lo tanto, con la ecuación se resuelve la cinemática inversa del mecanismo encontrando los ángulos de los puntos alcanzables en su espacio de trabajo.

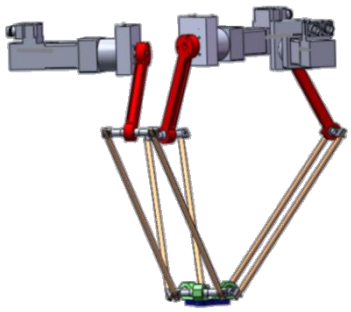


Ilustración 7. Cadenas conectadas a los motores

2.2.3 Espacio de trabajo del robot

El espacio de trabajo de un robot paralelo es la región de puntos que puede ser alcanzada por un punto de referencia en la extremidad de un manipulador. El volumen de trabajo del robot Delta es aproximadamente como la intersección de un prisma hexagonal con tres cuerpos de revolución.(Ceccarelli, 2006)

Los algoritmos de discretización son usualmente usados para determinar el espacio de trabajo del manipulador, pero requieren una gran capacidad de almacenamiento de datos en disco y por ende una gran capacidad de cómputo. Consiste en discretizar el espacio en 3 dimensiones, resolviendo la cinemática inversa por cada punto y verificando las restricciones que limitan el espacio de trabajo. (Ottaviano, 2002).

Para ello se realizó un barrido lineal de coordenadas en rangos pertenecientes a las dimensiones, esto es el alcance máximo de las cadenas cinemáticas y el límite máximo y mínimo de los ángulos de giro de brazos. El cálculo del espacio de trabajo se considera la longitud de las cadenas cinemáticas, el rango de giro máximo y mínimo de los α_1 y el alcance máximo de la base móvil en el eje Z. (Velázquez, 2003).

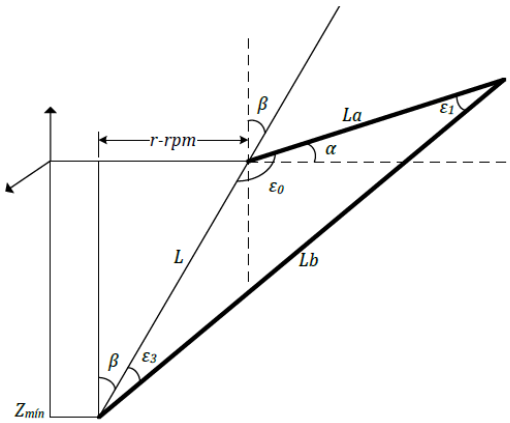


Ilustración 8. Modelo geométrico del robot

En el mecanismo hay posiciones de la base móvil que no deben ocurrir por riesgo de colisiones y bloqueos, por lo que se debe considerar al momento del análisis geométrico. Considerando las dimensiones que tendrá el robot:

La =240 Lb =790 r =100 rpm =90

Si, $\lim_{\varepsilon 1 \to 0} L = \sqrt{(Lb^2) + La^2 - (2LaLb)}$

Y

$$\lim_{\varepsilon 1 \to 0} \varepsilon_0 = 180°$$

Entonces queda:

$$\alpha + \beta = 90°$$

Además:

$$\beta = arcsen(\frac{r - rpm}{L})$$

Por lo tanto:

$$\alpha_{min} = \frac{\pi}{2} - \beta$$

Si $ai < amin$ entonces se tendrá una colisión inminente.

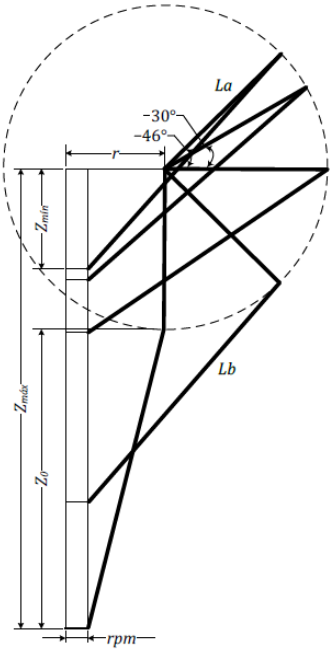


Ilustración 9. Desplazamientos de una cadena cinemática

En la figura se ven los ángulos de apertura mínimo y máximo, que determinan el limite mecánico para el espacio de trabajo, de aquí se toma la condición para evitar colisión entre los eslabones La y Lb, siendo Z min, el menor desplazamiento sobre z.

$$Z_0 = \sqrt{(Lb)^2 - (r - rpm)^2} \quad ; \qquad Zmax = \sqrt{(La + Z_0)}$$

2.2.4 Posicionamiento de los motores punto a punto.

Un tipo de trayectoria es el conocido como trayectoria punto a punto, donde cada articulación del robot evoluciona desde su posición inicial a la final sin realizar consideración alguna sobre el estado o evolución de las demás articulaciones. Cada actuador trata de llevar a su articulación al punto de destino, según la coordenada en el espacio de trabajo, en el menor tiempo posible, pudiéndose distinguir ya sea movimiento eje a eje y movimiento simultáneo de ejes.

El movimiento eje a eje solo se mueve uno a la vez, dependiendo la secuencia que se siga, comienza la primera articulación, una vez alcanzado el punto final, lo hará la segunda y así sucesivamente. Este tipo de trayectoria tiene un menor consumo de potencia instantánea por parte de los actuadores.

La desventaja de la trayectoria punto a punto, modo eje a eje, es que lleva mucho tiempo ejecutarla. Además al tratarse de un robot paralelo, el cual la característica es tener una cadena cinemática cerrada, el movimiento de un actuador de manera independiente puede estar restringido por las cadenas cinemáticas de los actuadores restantes.

Por otra parte el movimiento simultáneo de ejes, todos los actuadores comienzan simultáneamente a mover las articulaciones del robot a una velocidad específica para cada una de ellas. Los actuadores se sincronizan de manera que todos se coordinan comenzando y acabando su movimiento a la vez adaptando los demás ejes al más lento.

Para definir este movimiento únicamente se toma el punto de partida p_1 y el punto de llegada p_2, quedando

$$\Gamma = [\, p_1 \,, p_2]$$

Para una trayectoria con n puntos de pasos está definida por una secuencia de puntos de tal forma que

$$\Gamma = [\, p_1 \,, p_2, p_3, \dots, p_n]$$

Donde cada punto está descrito por:

$$p_n(x, y, z) = [p_{nx} \,, p_{ny} \,, p_{nz}]$$

Para unir tales puntos se hace que el robot se detenga en cada uno de ellos, teniendo velocidad nula, haciendo subtrayectorias para cada coordenada de par de puntos dado por:

$$\vec{r}(t) = \vec{p}_k + g(t)(\vec{p}_{k-1} - \vec{p}_k)$$

Donde k=0, 1, 2,…n y g(t) permite el proceso de interpolación, sus valores están dados por las siguientes relaciones:

$$g(t) = \begin{cases} 0 \; si \; t = t_0 \\ 1 \; si \; t = t_f \end{cases} ;$$

Donde t_0 es tiempo inicial y t_f es el tiempo final de cada subtrayectoria, asegurando que la velocidad inicial y final sea igual a cero. Además la función de interpolación define una trayectoria continua en aceleración.

$$g(t) = 10\left(\frac{t}{t_f}\right)^3 - 15\left(\frac{t}{t_f}\right)^4 + 6\left(\frac{t}{t_f}\right)^5$$

2.3.-SISTEMA DE CONTROL BALDOR

Baldor es un proveedor de equipamiento industrial, como son motores de corriente alterna, motores de corriente directa, servomotores, controladores, generadores, en fin una amplia gama de productos para solución de problemas sobre automatización. También cuentan con soporte técnico y reparación de equipo desde 1982, abasteciendo a múltiples empresas con equipo confiable y de alto rendimiento.

Ilustración 10. Marca Baldor

2.3.1 El MicroFlexe-100

El MicroFlex e-100 de Baldor es una avanzada unidad servo AC, que integra las capacidades en tiempo real Ethernet Power Link en los controles para un rendimiento superior, la integración de la red y ahorro de costes. MicroFlex e100 es compatible con el controlador de movimiento NextMove e100 para proporcionar una solución completamente integrada. (Baldor motion ,2012)

El e-100 MicroFlex es un driver compacto del servomotor Baldor, disponible en una sola fase de 110-230VCA o bien 230VCAa 3 fases, encontrándose con capacidad de corriente de 3, 6 y 9A. Proporciona un control de alto rendimiento para servo motores sin escobillas tanto rotativos y lineales, con su potente DSP (Digital SignalProcessor). Permite el control de Posición, Aceleración, Velocidad y corriente mediante un algoritmo de control PID.

Ilustración 11. Controlador MicroFlex e-100

El sistema cuenta con una conectividad muy versátil; USB, RS232, CAN y Ethernet. Se puede conectar a través del puerto USB de la PC o bien por medio de CANopen y Ethernet. Cuenta con un modulo de entrada para habilitación por hardware en forma remota, una salida indicadora de error para uso remoto, una entrada y una salida digital y dos terminales de entradas de alta velocidad como se aprecia en la siguiente figura.

La terminal X1 es para alimentar el controlador con red eléctrica de preferencia usando un filtro. Así mismo, cuenta con las terminales para alimentar los servomotores mediante el cable naranja de 8 pines, para esta aplicación se requiere usar los conectores U, V, W con los cables marcados con 1, 2, 3 respectivamente de acuerdo a la hoja de datos del fabricante. Los demás se omiten debido a que el modelo usado no cuenta con resistencia de frenado.

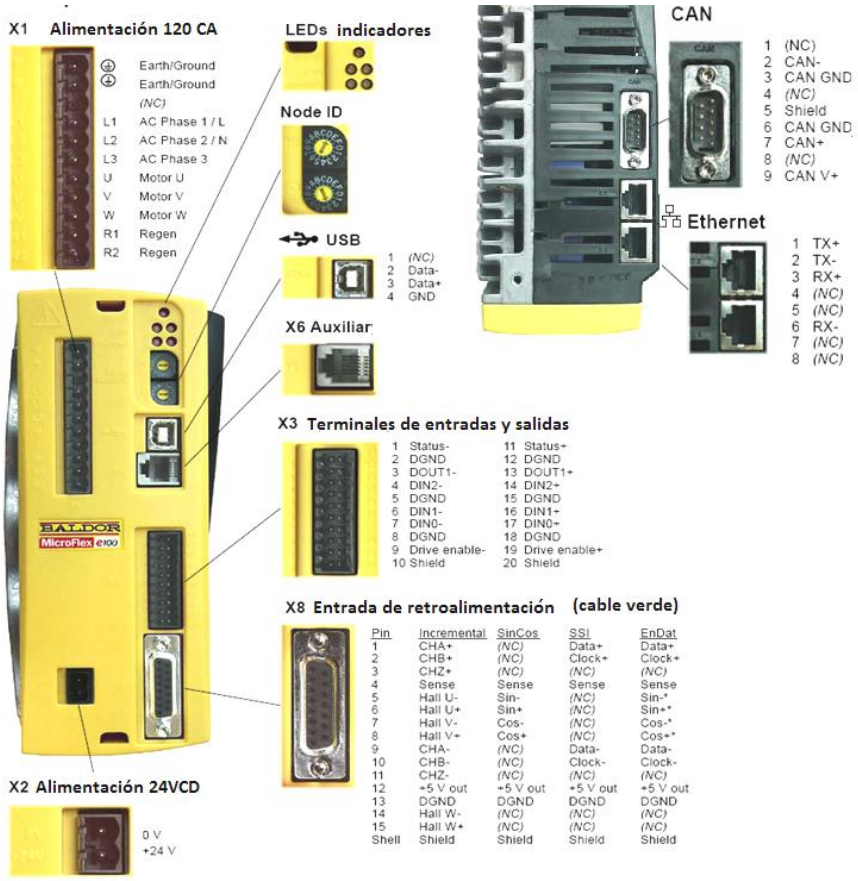


Ilustración 12. Partes y terminales del módulo MicroFlex e100

X2 es usada para alimentar la electrónica del módulo, por lo que se requiere 24VCD, según la hoja de datos del dispositivo. X3 cuenta con los módulos de entradas y salidas, así como la habilitación (pin 9 a 0V y 19 a +24V), que es necesaria para poder accionar los motores. Posteriormente esta terminal se usa como paro de emergencia ya que es una interrupción instantánea ante cualquier conflicto, sin afectar los datos procesados.

El puerto USB es el conector para la PC, los puertos EPL para hacer un tendido con otros controladores y con el Controlador inteligente para hacer un sistema multieje. Nodo ID es usado para dar una posición dentro de la red Ethernet en modo EPL, se numera en hexadecimal, respetando las siguientes direcciones:

- 00H nodo reservado
- 001H – EFH nodos esclavos
- F0H nodo maestro
- F1H – FFH reservados para propósitos especiales
  En el sistema se tiene el 03, 04 y 05 para el cada controlador

X8 es la retroalimentación del servomotor con el controlador, permite leer el encoder y demás sensores que conforman el servomotor, solo se requiere introducir cuidadosamente los conectores ya que es un par conectores en cada extremo con 16 pines uno para el controlador y en el otro con el servomotor.

El control está definido por el siguiente diagrama, donde se observa el control de posición, control de velocidad y controlador de torque. Esta configuración comprende 3 tipos de lazos, un lazo por corriente, un lazo con la velocidad, y un lazo por posición.

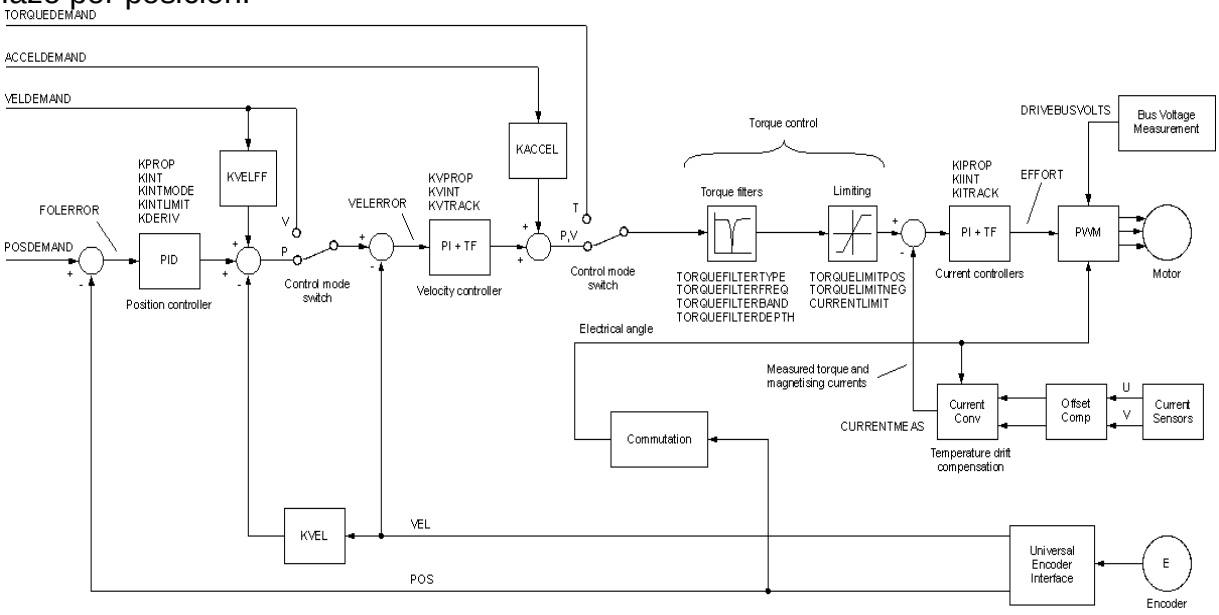


Ilustración 13. Esquema interno del controlador Micro Flex

La interface universal del encoder lee la posición del rotor y calcula la velocidad. El conmutador usa la posición para calcular el ángulo del rotor. El sensor de corriente, U y V, mide las fases de la corriente, alimentando un block de conversión de corriente que convierte en cantidades que representan el torque producido y las corrientes magnéticas.

En realimentación con la corriente, la demanda de corriente y la corriente final medida, forman las entradas para un sistema de control PI (Proporcional e integral). Esto determina un voltaje que alimenta el PWM. El PWM usa el método vector-espacio para convertir estos voltajes a una secuencia de U, V, W interruptores de señales de fase, que son aplicadas al puente de salida del driver.
El PWM usa el voltaje medido del puerto en Corriente Directa, para compensar las variaciones de la fuente de poder.

El control por Torque, convierte la demanda del torque en demanda de corriente y compensa para varias cargas no lineales. Un 2-Satage notch o un filtro pasa bajas, permite reducir el efecto de la carga. Para evitar daños al motor, un límite de corriente definida por el usuario, es también aplicado, así como un límite de torque positivo y negativo.

En el control por velocidad, una demanda de velocidad es medida, como entrada de un sistema de control PI, la salida del control es una demanda de torque, que cuando el controlador está operando como un controlador de velocidad, forma la entrada para el lazo de control de corriente.

Finalmente, en el control de posición, una demanda de posición es medida como entrada de un controlador PID, (Proporcional, integral y derivativo) que incorpora la retroalimentación de velocidad, velocidad de avance, y retroalimentación de aceleración. La salida del control de posición es una demanda de velocidad, cuando el driver está operando como un controlador de posición, forma la entrada en lazo de control de velocidad.

2.3.2 Servomotores de Baldor

La implementación que se le hará al nuevo prototipo consiste en servomotores de CA sin escobillas de la serie BSM N-, que ofrecen aplicaciones industriales de control de movimiento con baja inercia para alcanzar la mayor capacidad de aceleración, que le permite colocar rápidamente un punto. El BSM de la serie N se utiliza en máquinas rápidas y en aplicaciones exigentes.

Ilustración 14. Servomotor con cable de realimentación

Los motores sin escobillas de la serie servo N ofrecen un diseño robusto y duradero con alta energía de neodimio-hierro-boro magnetismo. Esta serie ofrece capacidad de torque continuo que va desde 0,45 Nm (3,9 libras por pulgada) a 354 lb-in (40 Nm). Capacidad de par máximo es nominalmente 4 veces continua. Esta serie tiene la menor inercia para proporcionar un par máximo.

BSM 50N-275AF			**MICROFLEX e100**	
Corriente	1.42A(RMS)	4.87Ap	6 A RMS	3A PROM
Voltaje		64.35Vp	320v BUS VOLT	
Inductancia	33.2mH			
Resistencia	16.06 Ohms		ENCODER	
Velocidad Max	10,000rpm		25000 X 4=100000	
Polos/par	4=2polos/par		1REV=100000	
Velocidad Prom	2000		Torque ConstStall 0.91NM	
Potencia	0.18Kw		reductor 10:1	

Ilustración 15. Tabla comparativa de característica de motores

2.3.3 Controlador NextMove e100

Este dispositivo proporciona soluciones en la automatización de la gestión en tiempo real, se programa con el compilador Mint WorkBench, o en diferentes lenguajes de programación, y cuenta con conexione sencillas, simplificando el diseño del sistema y la instalación, y simultáneamente amplía las capacidades de control de otros dispositivos.

Ilustración 16 . Controlador inteligente NextMove e 100

El NextMove E100 puede realizar interconexión de 8, 12 o 16 ejes, ya sea como un grupo de coordenadas único o como múltiples sistemas de coordenadas de funcionamiento independiente, conectándolos en la red. Su programación es por medio de MintMT - multitarea básico, integrado a 'C' o los componentes ActiveX suministrados. NextMove e100 está disponible con un puerto USB puerto de programación y el usuario puede seleccionar el puerto serie RS232/485.

El NextMove*e*100 es compatible con el protocolo ETHERNET Power Link (EPL). Este protocolo provee comunicación muy precisa y predecible en tiempo real, a través de una conexión de 100 Mbit/s (100Base-T) de Fast Ethernet (IEEE 802.3u). Esto lo hace adecuado para la transmisión de señales de control y realimentación entre el NextMove*e*100 y otros controladores habilitados para EPL, como el MicroFlex*e*100.

2.4.-CONTROL POR COMPUTADORA

El proveedor proporciona una herramienta de control por computadora para estos dispositivos, esta aplicación se llama Mint WorkBench, de descarga libre en la página Baldor. Con esta herramienta de Mint, se pueden realizar ajustes en lo motores como sintonización, calibraciones, revisar el estado de los motores, y programar rutinas rápidas y sencillas con una ventana de comandos para realizar movimientos con los motores.

2.4.1. Movimientos básicos

Uno de los movimientos básicos es el posicionamiento Absoluto, el cual sitúa el eje del motor a una posición respecto al cero formado al encender el dispositivo, moviéndose a partir de esa posición cero. Este tipo de movimientos se analizará posteriormente ya que con otras herramientas como el Homing se tiene una posición absoluta del robot.

Otro movimiento básico es el posicionamiento Relativo, el cual sirve para mover las n cuentas que se le indique, sumando o restando a las cuentas actuales según sean movimientos positivos o negativos del eje.

Como el robot tiene la capacidad para desplazarse en el espacio, es necesario tener una posición inicial para ser nuestra posición de control, para ello el encoder no es el único sensor que se usara ya que a pesar de dar movimientos absolutos, no tiene un punto constante en el cual inicialice al arrancar.

Para ello se requiere usar la herramienta Homing, la cual usamos los encoders y sensores de fin de carrera para que al girar y llegar a tal posición se inicialice una marca y así tomar asignar un valor conocido o referente a esa posición.

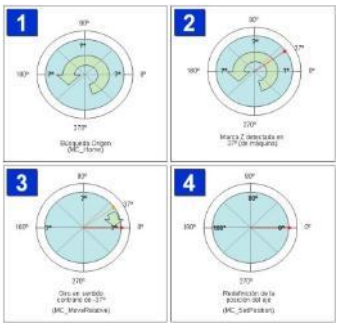


Ilustración 17. Homing

2.4.2 Movimientos usando Splines

Los Splines sirven para posicionar los motores en determinados ángulos por medio de la velocidad especificada en términos de tiempo. Un movimiento de este tipo es creado dividiendo el movimiento en n segmentos, cada segmento define que tan lejos llegara el motor en un periodo de tiempo dado y opcionalmente la velocidad del eje al final de tal segmento.

Una vez definido tales segmentos, el controlador interpolará entre tales datos o llenar los faltantes, produciendo un movimiento suave. El usuario controla el tamaño y la precisión de tales movimientos, el movimiento crítico y segmentos deben ser definidos. Los segmentos son definidos por arreglos de posición y opcionalmente por arreglos de velocidad y segmentos de duración.

La posición puede ser interpretada por las siguientes formas:

- El valor representa movimientos relativos. (Cada valor es relativo a la distancia que el eje debe mover hasta llegar a tal segmento).
- El valor representa una posición absoluta dentro del ciclo de Splines. (Cada repetición es conocida como ciclo).
- El valor representa una posición absoluta verdadera. El valor es absoluto referente a la posición actual. La posición cero es cuando el eje estaba en reposo.

2.4.3 Programación con otros lenguajes

Baldor permite realizar interfaces de usuario para controlar los motores de Baldor, a través de un script llamado Mint controller, esta herramienta desarrollada bajo Mint WorkBech, permite hacer modificaciones con la misma estructura, pudiendo controlar motores pero con mayores ventajas.

ActiveX (se suministra de forma gratuita con cada controlador) proporciona una interfaz completa al controlador desde cualquier entorno de programación compatible (National Instruments LabView, Visual Studio y los productos de Microsoft Office).

El control ActiveX ofrece todas las capacidades para diagnóstico, captura de datos manipulación y movimiento completo y la secuencia de E / S. Las características incluyen: Common API (Application Programming Interface) con Mint.Acceso a todas las E/S y funciones de movimiento del controlador/driver.

Capacidad de alarmas sobre los eventos Mint - operacionales sobre el bus PCI y USB. Crea interfaces de forma rápida y fácil entre los productos de Baldor. Compatible con todos los controladores de movimiento de Baldor y servo accionamientos.

2.4.3.1 Programación en Visual Studio

Microsoft Visual Basic está diseñado para generar de manera productiva aplicaciones con seguridad de tipos y orientadas a objetos. Visual Basic permite a los desarrolladores centrar el diseño en Windows, el Web y dispositivos móviles.

Como con todos los lenguajes que tienen por objetivo Microsoft .NET Framework, los programas escritos en Visual Basic se benefician de la seguridad y la interoperabilidad de lenguajes. Esta generación de Visual Basic continúa la tradición de ofrecer una manera rápida y fácil de crear aplicaciones basadas en .NET Framework.

Visual Basic vuelve a incluir la compatibilidad para Editar y continuar e incluye nuevas características para el desarrollo rápido de aplicaciones. Una de estas características, proporciona acceso rápido a las tareas frecuentes de .NET Framework, así como información e instancias de objeto predeterminadas que estén relacionadas con la aplicación y su entorno en tiempo de ejecución.

El modelo de objetos basado en componentes (COM), se introdujo a mediados de los años 90 como una vía para conseguir un mayor aprovechamiento del código, al situarlo en componentes reutilizables por más de una aplicación. Lo que lo hace un lenguaje de programación de alto desempeño para realizar aplicaciones de control.

Ilustración 18. Visual Basic de Microsoft Visual Studio

2.4.4 Diseño por computadora

Solid Works es un programa de diseño asistido por computadora para modelado mecánico desarrollado en la actualidad por Solid Works Corp. Es un modelador de sólidos para métrico. Permite diseñar piezas en 2D y 3D, tiene el parecido al programa Inventor, solo que se optó por este programa ya que se cuenta con licencia institucional. (Gómez, 2008)

El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos como otro tipo de información necesaria para la producción. Es un programa que funciona con base en las nuevas técnicas de modelado con sistemas CAD. El proceso consiste en trasladar la idea mental del diseñador al sistema CAD, "construyendo virtualmente" la pieza o conjunto. Posteriormente todas las extracciones (planos y ficheros de intercambio) se realizan de manera bastante automatizada.

2.4.5Entorno de Solid Works

El programa cuenta con una interface accesible para poder diseñar una pieza desde cero, además cuenta con las herramientas para hacer un modelado y análisis estructural mecánico para verificar la optimización de las piezas. Así mismo cuenta con animaciones y simulaciones para realizar videos sobre el prototipo.

Ilustración 19. Solid Works

Además de crear piezas específicas permite hacer composiciones llamados ensambles de un conjunto de piezas para armar un equipo total, agregando y quitando propiedades y parámetros.

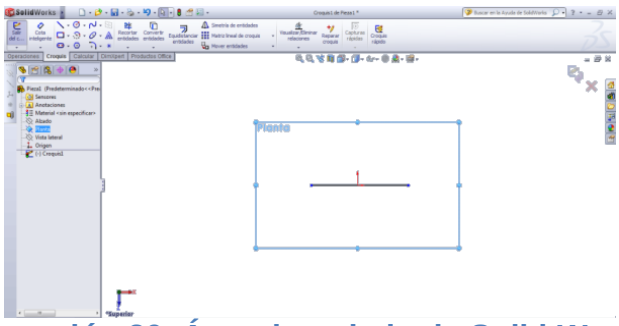


Ilustración 20. Área de trabajo de Solid Works

CAPÍTULO III.- DESARROLLO DEL PROYECTO

3.1.- INVESTIGACIÓN DOCUMENTAL

Primeramente se realizó una documentación general del equipo existente, así como fundamentos teóricos del sistema en el que se desarrollaron las demás actividades de la residencia profesional, reconocimiento del equipo a usar.

Se requiere de un sistema de control de movimiento que permita manipular el robot, para ello se implementó un equipo de servomotores de la marca Baldor, los cuales son dispositivos de alta velocidad con drivers para cada motor y un control maestro para lograr tareas de posicionamiento con precisión.

La una parte de la investigación radica en usar estos motores, ya que como el modelo esta rediseñado para escalarse a nivel industrial con un espacio de trabajo mayor a los robots convencionales, se probará el comportamiento con esta instrumentación. Se documentó sobre sus formas de conexión y pruebas básicas de funcionamiento.

Una vez conectado se procede a calibrar mediante el programa, el cual es proporcionado gratuitamente desde la página http://www.baldor.com/products/motioncontrol/mintmt.asp.

Ilustración 21. Mint WorkBench de Baldor

Con el programa ya instalado en la PC, se puede realizar auto calibración e iniciar a mover el servomotor con rutinas específicas. Para ello encendiendo el driver, y se conecta a la PC, para que al abrir el programa detecte el controlador. Aparece una primera ventana, donde se puede crear, abrir o editar un proyecto, al ser la primera vez que se sintonizara el driver, se abrirá un nuevo proyecto.

Ilustración 22. Ventana de selección de proyecto

El asistente indica seleccionar un controlador, en este paso, debe estar el dispositivo listo, seleccionando MicroFlex e100, si no aparece dar clic en Scan, para buscar los controladores conectados al equipo, si no aparece asegúrese de que esté correctamente conectado al equipo. A seleccionar el controlador, igualmente indicamos el Nodo ID al que pertenece, indicando el ID el dispositivo, este paso lo hace automáticamente al conectar el Next-Move.

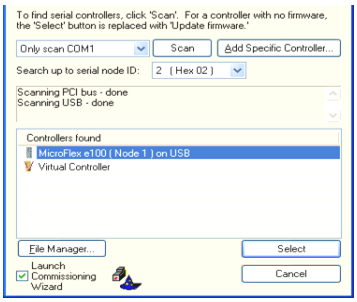


Ilustración 23. Seleccionar controlador MicroFlex e100

Verificar que esté marcado “iniciar asistente” en la parte inferior izquierda de la ventana y dar click en seleccionar. Aparece la ventana siguiente, seleccionando Metric en sistema de medición a usar, dar click en siguiente.

Ilustración 24.Asistente de Mint WB.

Especificamos el puerto de comunicación que se usara en la conexión, seleccionar Ethernet ya que más adelante se hará un tendido de red Ethernet con los demás drivers y el Controlador inteligente Next-Move.

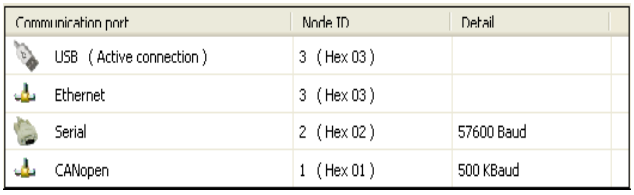


Ilustración 25. Conexión

Seleccionar el modelo del motor, en este caso es el servomotor BSM 50N 275 AF.

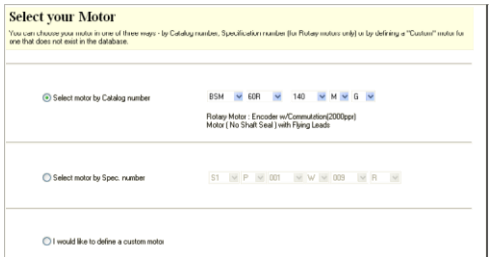


Ilustración 26. Modelo del motor

La siguiente ventana indica especificaciones del motor y del encoder, se le da siguiente. Es importante verificar la hoja de datos para cualquier aclaración si no corresponde, puede que no se haya seleccionado correctamente el motor, de ser así regresar a la ventana anterior.

De lo contrario dar clic en siguiente y aparece la ventana para seleccionar el modo operativo y recursos del control, para ello seleccionar nuevamente EPL, ya que posteriormente se hará el tendido de red. Y dejar el modo operativo en Posición a menos que se requiera un control de velocidad, torque o ninguno de los anteriores.

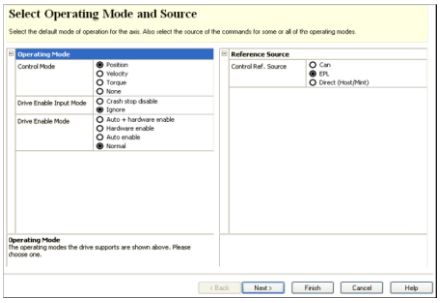


Ilustración 27. Pantalla de modo de control.

Posteriormente el asistente permite visualizar la configuración del sistema confirmar dando click en siguiente, hasta finalizar. Hay que poner atención en la información que proporciona el asistente por cualquier duda que se tenga de los servomotores.

Posteriormente se procedió a conectar los tres servomotores que se implementarán al robot, para hacer la calibración inicial y así tenerlos listos al armar la estructura o base fija del robot. Se hicieron pruebas de desempeño como velocidad, precisión, torque y conexión segura al equipo. Ya que se presentaban problemas de desconexión repentina al finalizar la sintonización. Para hacer un sistema multieje, es necesario sintonizar todos los drivers de los servomotores existentes con el auto tuning, de manera individual, y así posteriormente realizar la conexión entre ellos con el Controlador multieje Nextmove e100.

Para conectar el NextMove*e*100 a otros dispositivos con EPL, se usan cables CAT5e tipo Ethernet, S/UTP (cables trenzados apantallados o recubiertos en aluminio sin blindado) o preferentemente S/FTP (cables trenzados apantallados o recubiertos en aluminio con blindado completo). El fabricante proporciona un juego de cable color amarillo con el cual se comunicó el sistema. El NextMove*e*100 incorpora un nodo de red repetidor integrado, que provee dos puertos para la conexión de otro equipo. Esto permite que los nodos se conecten como una red con bus de encadenamiento de hasta 10 nodos

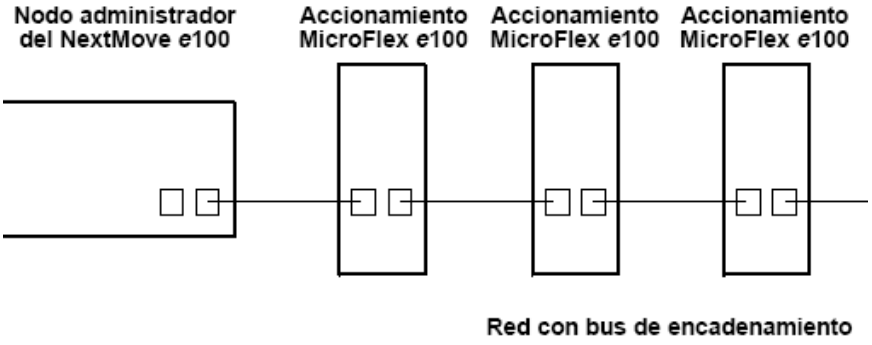


Ilustración 28. Múltiples controladores en Red

El NextMove*e*100 se puede conectar al PC utilizando tanto USB, TCP/IP, RS232 o RS485/422. En este caso se optó por usar el puerto USB ya que se cuenta con el cable de conexión, una vez concluido los pasos anteriores de calibración y hechas las conexiones lo que prosigue es configurar el controlador Nextmove e100, esto se hace a través del mismo software, Mint Workbench.

Nuevamente iniciamos el programa abriendo un nuevo proyecto, y seleccionando el Controlador Nextmove e100, nos aparece la ventana de configuración, sele da clic en system configuration, agregamos los dispositivos existentes en la red, notando que aparecen con un Nodo y un ID, el cual identifica a cada controlador conectado al sistema, se recomienda poner el mismo número para ambos casos, hacer clic en **Aceptar**. Se visualizará la ventana de Asignación de recursos.

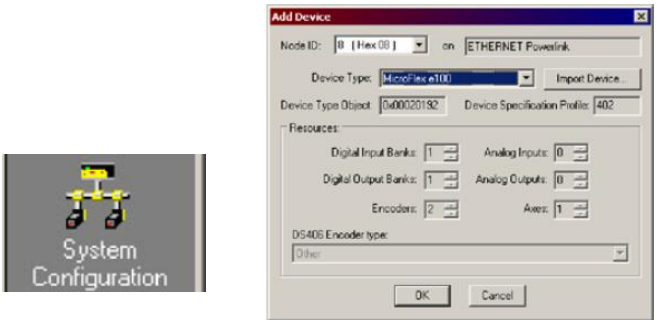


Ilustración 29. Agregar dispositivos al controlador durante la configuración.

Al finalizar de agregar los dispositivos aparecerán marcados en la lista como se ilustra en la figura. Dar clic en propiedades de cada uno de los controladores, si están correctamente instalados, aparecerán sus elementos marcados con un acierto color verde, si no han sido cargados correctamente aparecerán con una cruz y deberá indicar la configuración de forma manual, editando cada uno de los elementos actualizando el numero de nodo.

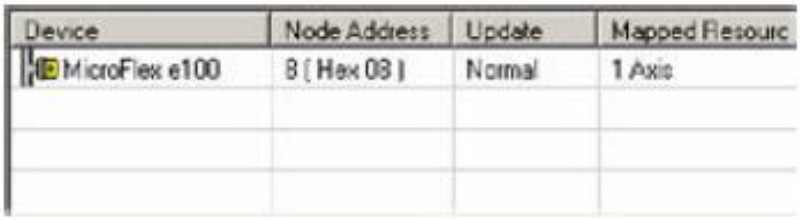


Ilustración 30. Un dispositivo agregado

Se familiarizó interactuando con el programa Mint WorkBench, realizando códigos de programación y tenerlos como rutinas listas para posteriormente analizar si su uso era conveniente con los objetivos del proyecto, de crear el control de los motores con las restricciones del robot paralelo tipo delta.

Sin embargo al no poder acceder a algoritmos usando matrices y la solución de ecuaciones del modelo geométrico inverso desde Mint WorkBench se documentó sobre algún otro lenguaje de programación que permitiera controlar los motores Baldor usando estos algoritmos fundamentales para realizar el control de los motores sin llegar a posiciones prohibidas del robot.

Por ello se optó por usar un lenguaje de programación más robusto que permitiera la solución de estas ecuaciones y así generar una interface gráfica para el usuario, se optó por Labview, pero por un problema de licencias, se cambió a Visual Basic, ya que la institución donde se realizó la residencia cuenta con las licencias de este programa y se pueden tomar los componentes Active X para el control de los motores.

Mint tiene muchas limitantes en cuanto a una interface visual y el procesamiento de los algoritmos complejos como los meta heurísticos. Procesamiento clave para realizar el análisis de trayectorias, posicionamientos para el control de robots, y con el objetivo de controlar un robot paralelo con estos motores aumenta la necesidad de realizar una programación más exigente en cuanto a procesamiento.

Sin embargo Baldor proporciona librerías y métodos como los ActiveX, para desarrollar aplicaciones en otros lenguajes, tales como LabView de National Instruments. LabView® al igual que Visual Studio, que son software de diseño de aplicaciones, construido específicamente para tareas realizadas con alto rendimiento y cubrir necesidades de control y automatización.

El complemento de ActiveX permite realizar interfaces de usuario para controlar los motores de Baldor, a través de un script llamado Mint controller, esta herramienta desarrollada bajo Mint WorkBech, permite hacer modificaciones con la misma estructura, pudiendo controlar motores pero con mayores ventajas.

ActiveX (se suministra de forma gratuita con cada controlador) proporciona una interfaz completa al controlador desde cualquier entorno de programación compatible (National Instruments LabView,Visual Studio y los productos de Microsoft Office). El control ActiveX ofrece todas las capacidades para diagnóstico, captura de datos manipulación y movimiento completo y la secuencia de E / S.

Las características incluyen:

- Common API (Application Programming Interface) con Mint.
- Acceso a todas las E/S y funciones de movimiento del controlador/driver.
- Capacidad de alarmas sobre los eventos Mint - operacionales sobre el bus PCI y USB.
- Crea interfaces de forma rápida y fácil entre los productos de Baldor.
- Compatible con todos los controladores de movimiento de Baldor y servo accionamientos.

Se desarrollaron algoritmos de pruebas para los servomotores con lo que se describe en la documentación, y así generar el código necesario para realizar las operaciones más complejas con Visual Basic, como aplicarle el modelo inverso, el uso de las herramientas que Baldor proporciona y acceso a la información en tiempo real de los motores y controladores.

Por otra parte se documentó sobre software de diseño para realizar simulaciones y otras pruebas que complementan las actividades de la residencia, ya que es un área multidisciplinaria, se requiere de diseño de piezas, uno de estos programas fue el uso de Solid Works, un programa CAD que permite realizar el modelo 3D, y fue así que se desarrolló parte del robot con esta plataforma de diseño mecánico.

Ilustración 31. Módulo de control conectado

3.2.- GENERACIÓN DE TRAYECTORIAS DEL ROBOT

Con el objetivo de hacer trayectorias, es importante tener el programa de control de motores Baldor que tenga las restricciones y la solución de la cinemática inversa para poder llegar a las posiciones permitidas o válidas del robot.

Para ello la importancia de tener presente siempre las restricciones del robot y así realizar trayectorias punto a punto, con movimientos correctos en el robot. Cabe mencionar que hay movimientos que el robot no puede hacer por conflictos de choques con la estructura.

Los tres motores han sido calculados con el modelo geométrico inverso de acuerdo al punto donde se desea llegar, se complementa la ejecución de los motores gracias al encoder que poseen, esto con el fin de asegurar que se llego a la posición deseada, con esto se logra un control de lazo cerrado. Para ello se sugieren algunas trayectorias diseñadas para poder asignar los puntos de intersección y así llevarlos a cabo.

Se generó un cuadrado que permita posicionar en cuatro coordenadas en un plano, y así generar una trayectoria básica. Los puntos para X, Y, Z que se usaron fueron para generar un cuadrado de 60 cm = 600 mm como distancia entre punto y punto. Como es un plano, el eje Z permanece constante.

Una de las tareas comunes de un robot paralelo es posicionar una serie de objetos en una caja en un orden preestablecido para ello se generó una trayectoria tipo pick and place donde simulamos tomar una serie de objetos y colocarlos en 8 diversas pociones.

Se seleccionaron 8 puntos ordenados y un punto de acopio, donde en la simulación se puede visualizar las interacciones de la plataforma móvil. Al igual que la generación del cuadrado, nos permite obtener los ángulos que hacen llegar al robot a tal posición y así mediante un algoritmo llegar a tales lugares.

Para la generación de un círculo se recurrió a una función la cual genera una serie de puntos para inscribir un círculo en el plano X, Y, donde Z permanece constante, con el programa en Matlab se asigna la coordenada X, Y, para el centro del círculo, el radio, en el caso de la simulación se recurrió a un circulo con radio 30 cm = 300mm, con un total de 100 puntos.

Estos puntos logran afinar el movimiento del efector, pero entre más puntos se inscriban, mayor será el tiempo de ejecución. El programa genera los puntos en X, en Y, para luego procesarlas con la cinemática inversa del robot paralelo y simula la generación de la nueva trayectoria. Como el robot tiene la capacidad para desplazarse en el espacio, se elaboraron trayectorias para visualizar su comportamiento desplazándose variando X, Y, Z simultáneamente. Para ello se proponen figuras conocidas como una espiral y un cono que a continuación se detallan.

Primeramente se generó tales figuras a manera que puedan manipularse fácilmente. Con la función usada, nos da una serie de puntos que son necesarios para poder describir tales figuras. Teniendo esos puntos en X, Y, Z, se analizan con la cinemática inversa para ver si no hay conflictos y una vez analizado nos arroja los ángulos necesarios para describir tales movimientos.

Con los mismos pasos, se procedió a realizar un cono y una espiral, el cual va aumentando su radio conforme baja, con la misma relación anterior, haciendo que inicie en el punto 0,0,-550, y al finalizar regresar a la posición inicial.

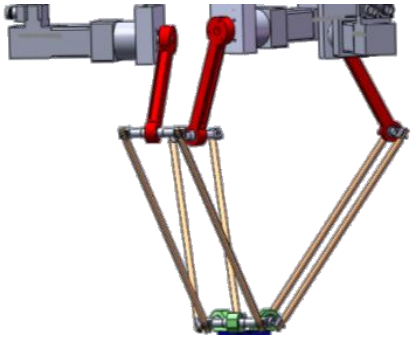


Ilustración 32. Prototipo en Solid Works

3.3.- SEGUIMIENTO DE LAS TRAYECTORIAS

Las trayectorias del robot paralelo necesitan estar simuladas para poder ver el resultado en el prototipo, por ello se elaboró un algoritmo en Matlab que permita realizar, verificar y corregir los puntos del robot paralelo. Por otra parte, la simulación y el diseño por computadora obligan a analizar otras posibles trayectorias para dejar definidas en el programa de control que posteriormente se implementarán.

Para asegurar el posicionamiento se hace una simulación en Matlab, mediante el cual se posicionan los eslabones notándose el alcance en el espacio de trabajo, solucionando la cinemática inversa, se adquieren los ángulos para cada motor, se proyectan los eslabones del robot y se visualizan los flexiones y extensiones del robot . Esto permite realizar, verificar y corregir los puntos del robot paralelo.

Se simulan para ver el comportamiento y así poderlas aplicar a los motores para posteriormente verlos en el robot. Las trayectorias analizadas permiten visualizar e implementarlas en el prototipo para evitar problemas como colisiones, singularidades o problemas mecánicos, así mismo da una idea del comportamiento del sistema con las dimensiones del rediseño.

En el programa desarrollado también se puede introducir una coordenada X, Y, Z para ver el alcance del robot en la simulación, o bien un conjunto de coordenadas visualizadas durante la simulación. Continuando con las trayectorias anteriores, fue desarrollar el código en visual Studio, y así poder visualizar los movimientos en los motores, que no se contó con el dispositivo completo. Sin embargo, se hicieron estimaciones viendo el comportamiento de los ejes.

Con los ángulos que el algoritmo de Matlab, fue posible pasar directamente las matrices, ya que los puntos fueron analizados con la cinemática inversa y así ahorrar tiempo de cómputo, durante la ejecución del programa, cosa que puede alentar la respuesta o ejecución de los algoritmos.Con los puntos generados en la simulación se ponen en un archivo .txt y pasarlos al control en Visual Basic, como alternativa para introducir las coordenadas previstas.

3.4.- INTEGRACIÓN DE CONTROL

Teniendo listas las trayectorias ahora queda codificarlas en el programa de control de robot, y así poder mover los motores en las coordenadas propuestas, Fue posible generar los movimientos ya que se puede convertir los ángulos en cuentas del motor, además la herramienta de los Splines de Baldor, se introducen los ángulos en una matriz y automáticamente el controlador hace el cálculo de la velocidad en función del tiempo, con movimientos, en este caso absolutos.

Primeramente se genera el vector de n datos, donde el primer valor se indica el total de movimientos a hacer, posteriormente se ponen todos los n movimientos. Se llena la tabla de Splines, donde se indica el numero de motor, el vector de movimientos, Velocidad (en este caso 0 para que haga el cálculo en función del tiempo, duración (igual 0 por default).

Posteriormente se habilita el motor, se configura el tiempo para cada movimiento, el tipo de Spline, en este caso absoluto total (1 or 2) y finalmente el arranque. Se procede con los otros dos motores, con la misma configuración para hacer movimientos sincronizados.

Con las piezas que el tornero proporcionó, se pudo armar y fijar la estructura con el montaje de los motores Baldor, del cableado para alimentación y retroalimentación, así como tener listo el modulo de control. Con esto se hizo los diagramas de instalación eléctrica correspondiente. El circuito eléctrico cuenta con interruptores y un indicador luminoso para visualizar la conexión del sistema, para ello se hizo el siguiente diagrama donde se detalla las conexiones realizadas para su armado.

Para implementar el encendido del sistema se uso un modulo industrial el cual tiene las siguientes características, además se agregó un botón de paro de emergencia por cualquier movimiento o comportamiento en falso. Con la construcción parcial y la calibración de los motores se implementó la herramienta Homing, y así poder mover los brazos del robot de manera continua, con las trayectorias con poca carga.

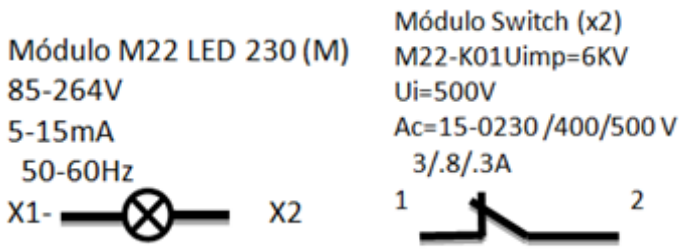


Ilustración 33. Módulos de encendido y apagado con botón luminoso

Se implementaron sensores de fin de carrera para hacer el proceso de Homing en el robot y determinar la posición de inicio como autoajuste en una posición del robot conocida, en este caso la posición horizontal, que es igual a la posición 40,40,40 cuentas de motores; 0,0,0 grados en los motores, o bien X=0, Y=0, y Z=750.

Para realizar el montaje del sistema de movimiento del robot, con servomotores Baldor, se usaron los siguientes componentes:

- NextMove e100 (Controlador Inteligente multieje o multiservos)
- MicroFlex e100 (Módulo de control y potencia para un servomotor)
- Servomotor Baldor
- PC para el software de control
- Cable de potencia para servomotor -Driver(Naranja)
- Cable de retroalimentación para servomotor-Driver (Verde)
- Cableado para alimentación CD, y CA.
- Fuente de poder con salida de 24VCD.
- Filtro para CA.
- Cable USB
- Cable EPL (Ethernet Power Link)

La mayoría de estos componentes son adquiridos por el proveedor de productos Baldor, los cables de alimentación de CD y CA, se adquirieron por separado. Se recomienda tener una PC con Sistema Operativo actualizado, con Windows ® XP o superior. El cable USB es usado para conectar el Controlador Next-Move e100 con la PC, los cables EPL para interconectar los módulos de potencia con el controlador multieje.

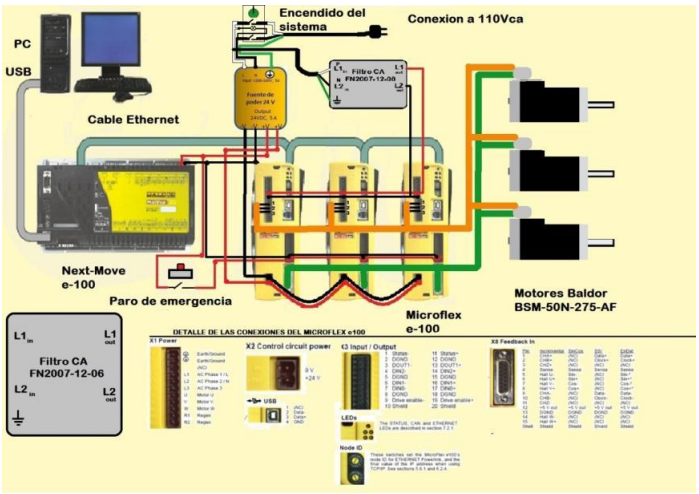


Ilustración 34. Diagrama eléctrico y de interconexión para el sistema multieje

3.5.- VALIDACIÓN EXPERIMENTAL

Como consecuencia de la instalación de los motores, se procedió a calibrar los motores para poder visualizar el efecto de moverlos, así como la calibración con carga, ya que el fabricante proporciona una herramienta para calcular la carga y ajustar los motores con el PID correspondiente.La calibración con carga en los motores Baldor, permite realizar movimientos de los brazos a distintos ángulos y así con la simulación comprar las posiciones que realiza al variarlas continuamente.

Para ello es necesario hacer una rutina que permita colocar al robot en una posición conocida de referencia como punto de partida para poder hacer las demás ejecuciones. Ya que solo se cuenta con los encoders de los motores los cuales son incrementales y no cuentan con un cero absoluto, desde el punto de vista del robot.

El controlador y los drivers de los motores Baldors, poseen terminales de entras y salidas digitales, las cuales servirán para implementarle al robot una serie de interruptores de fin de carrera, (limit switches), los cuales permitirán saber la posición limite o de referencia para los brazos.

Con las lecturas de las entradas se procedió a elaborar un algoritmo en el cual los brazos se extendieron paulatinamente de manera que presione correctamente el sensor posicionado sobre la horizontal de los brazos, ya que esta es la posición 0°, 0°,0° que adopta el robot.

Al llegar a esta posición los brazos activan la señal y cada uno es desacelerado para poder cambiar el valor de los encoders. Para ellos se usaron una serie de funciones que Baldor integra en el lenguaje de programación para poder realizar este proceso. Para instrumentar este proceso, se usaron 3 limit switches normalmente abiertos, adquiridos por separado, así como cables y soldadura para hacer el circuito correspondiente.

Tomando el código en Mint WorkBench, bajo la misma lógica de programación se procedió a realizar el código en Visual Basic, ahora con los métodos del active X, primero se procuró hacer las lecturas del controlador, bajo el mismo esquema de leer 1's y 0's que se le introdujo mediante los interruptores de fin de carrera. En el programa se introdujo un botón el cual se llama Posición inicial, para que el usuario oprima el botón y se inicie la secuencia de posicionamiento como se ve en la figura.

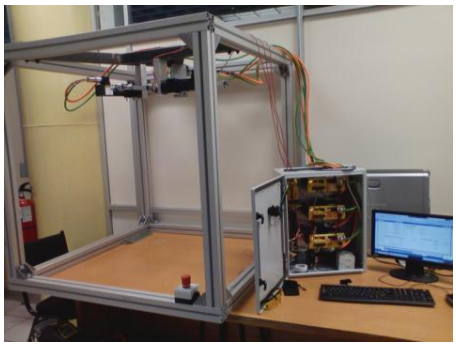


Ilustración 35. Elevamiento de los brazos a la posición 0°,0°,0°, sobre la horizontal

Con la rutina anterior es posible verificar las trayectorias, sin embargo debido a problemas con la plataforma móvil aún no es posible ensamblarla, lo que dificulta hacer validación de los movimientos realizados.

Por ejemplo se notó que habría que invertir los signos de las posiciones ya definidas por el ajuste mecánico que se hizo con la estructura, por una inconsistencia con los barrenos, sin embargo por programa es fácil implementar este cambio, negando todos los valores dados por la cinemática inversa.

Con estos resultados y con las trayectorias listas, ya se pueden ejecutar los movimientos de los brazos libremente por el espacio de trabajo, lo que permite tener como producto una interface de control óptima para el desarrollo del robot.

La implementación de las trayectorias, con solo los brazos conectados, son correctas, por lo que resta realizar pruebas con otras trayectorias y demás pruebas de la cinemática inversa. Se continuó pero ahora con los brazos añadidos a los ejes de los motores y con la calibración con carga. Además se agregó los antebrazos con el fin de visualizar la respuesta del robot.

Una forma de visualizar el comportamiento del robot fue con la ayuda de los brazos ya que con ellos se puede movilizar a las posiciones asignadas y verificar su ejecución. Para ello se calibraron los motores con carga, en este caso los brazos, para ver si había alguna dificultad.

Uno de los motores presentó problemas ya que no registra la posición al realizarse un esfuerzo con el brazo, perdiendo rigidez y no pudiendo leer el encoder el desplazamiento realizado. Complicación que resulta ser un problema mecánico con el motor-reductor del equipo. Se intentó corregir modificando las ganancias del PID, pero sin conseguir resultados favorables, ya que el encoder no se ve alterado, al igual que corriente del motor, descartando así un problema del control electrónico.

Sin embargo, se puede llegar a las posiciones que se le asignen, probando las trayectorias con los brazos, a una velocidad de 10,000 cuentas/seg, observándose movimientos rápidos y sin problemas en la ejecución.

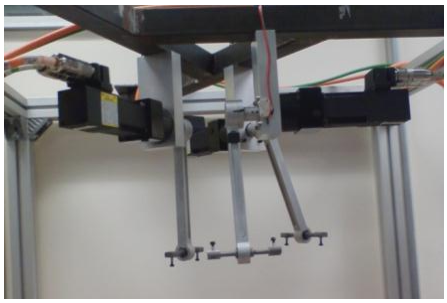


Ilustración 36. Ejecución de trayectorias con brazos instalados

Con la implementación de movimientos con la cadena cerrada se consiguió una pieza que permitiera incorporar los antebrazos, ya que aún no se cuenta con las piezas requeridas, pudiendo construir el robot en su totalidad. Como era de esperarse la pieza no cumplió ampliamente con su función, ya que el robot presentó un problema mecánico que generaba inestabilidad con los PID sintonizados.

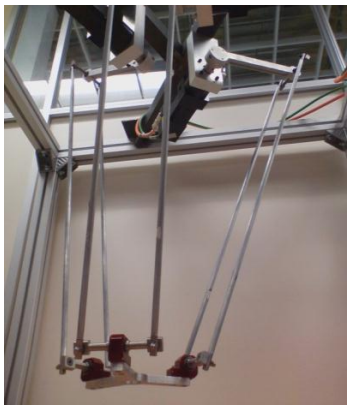


Ilustración 37. Eslabones físicos del robot

Por ello se volvió a sintonizar los controladores ajustando las ganancias hasta lograr estabilidad con el robot. Pero no se logró ya que los motores permanecieron oscilantes hasta calibrarlos con la ayuda de unos resortes.

Al detectarse este problema se le implementarán contrapesos al robot con tal de reducir el esfuerzo que se realizan en lo ejes, sin embargo eso implica llevarlos al tornero y un tiempo de espera mayor. Por ahora se sobrepuso un sistema de contra pesos de 800 gr cada uno, cuando por cálculo debería ser de 1.22 KgF ya que el peso del robot está en 2,300g entonces:

$$Fa = Rb$$

$$R = Fa/b$$

$$R = \frac{\left(\frac{2.3KgF}{3}\right)(24cm)}{15} = 1.22KgF$$

$$R = 1.22KgF$$

Donde F= 2.3KgF/3, a=240mm, b=150mm, R=Contrapeso

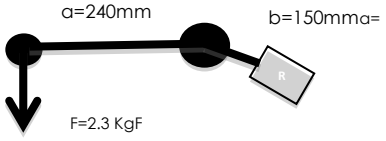


Ilustración 38. Contrapesos

Como parte final del proyecto, se realizaron diversas pruebas con el mecanismo, así como la redacción del informe final, para llegar a la conclusión de esta etapa del proyecto, así como la validación experimental del las trayectorias analizadas con la simulación.

Se implementaron las trayectorias en el dispositivo con un sistema adicional de compensación al peso de la estructura, usando resortes de manera provisional, ya que los contrapesos tardarán aun más en poder fabricarse, por lo que se recurrió a corregir el problema de sobrepeso.

Así mismo se hizo otro proceso de sintonización de los motores con el control PID, lográndose mover el robot correctamente en posiciones determinadas ya que la acción de los resortes no es continua y hay posiciones en las que no las oscilaciones regresan al robot.

Es por ello que se hicieron modificaciones a las trayectorias delimitándolas, para lograr realizarlas, debido a este problema mecánico persistente. Con el fin de verificar el funcionamiento del robot, del controlador y la interface, se realizaron las trayectorias basicas en el espacio de trabajo con un radio de 150mm (15cm) interpolando 32 puntos para hacer la trayectoria continua, con el mismo procedimiento a las trayectorias anteriormente generadas.

Se finalizó la interface de control generando el archivo ejecutable para su instalación en cualquier computadora con Sistema Operativo Windows Xp, en base a las licencias en CICATA IPN, que servirá de control en tiempo real, para ello se recurrió a agregar un nuevo proyecto, ejecutable (set-up).

CAPÍTULO IV.- RESULTADOS

4.1.-RESULTADOS, PLANOS, GRÁFICAS, PROTOTIPOS Y PROGRAMAS

EL robot quedó conformado con las medidas1120mm de ancho, 1050mm de largo y 1000mm de alto de la estructura de Base. El rediseño del prototipo Parallix LKF2040, se hizo en Solid Works, este programa ya que cuenta con un cambio en la estructura fija, usando perfiles de la empresa Aluminio Industrial de Querétaro, los cuales cuentan con buena resistencia, durabilidad alta y una interface de acople muy conveniente usando ángulos, tornillos y tuercas provistas por el fabricante.

Ilustración 39. Estructura en el modelo 3D

Esto permite ampliar el área de trabajo del robot, teniendo mayor desplazamiento y mayor libertad de movimiento. Ayudando así a manipular con mayor libertad el prototipo, aunado a la velocidad y capacidad de los motores Baldor.

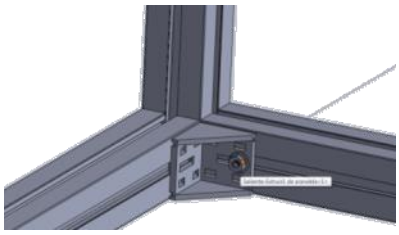


Ilustración 40. Vista del ángulo con un tornillo puesto,

Se necesita que los motores estén centrados lo mas exacto posible para poder realizar la cinemática correctamente.

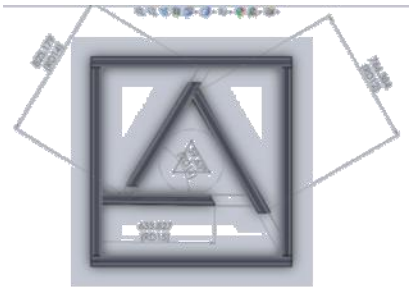


Ilustración 41. Formación del triangulo interno

Por ello se optó a rediseñar la cara superior de la plataforma, realizando un circulo en el diseño para poder acceder al centro exacto del esboce. Para ello se requiere de un triangulo ya que en cada lado inscrito del triangulo se posicionarán los motores consecutivamente según los cálculos realizados.

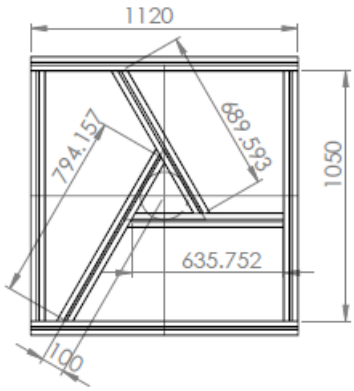


Ilustración 42. Medidas del robot

Con 100 mm del centro al extremo del triangulo donde irán los brazos del robot. Como esta estructura del robot no se podía hacer con Aluminio Boch, por los ángulos requeridos y el acoplamiento con las otras piezas se recurrió a hacer un perfil en PTR, el cual se soldó con trabajo de herrería, quedando de la siguiente manera.

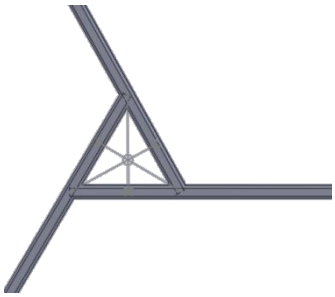


Ilustración 43. Base superior con perfil PTR

Los motores necesitan posicionarse correctamente en la estructura, pero debido a las fallas del trabajo de herrería, se tiene que diseñar una plantilla que indique la posición exacta de los soportes. Además de tomar las consideración mecánicas y geométricas necesarias.

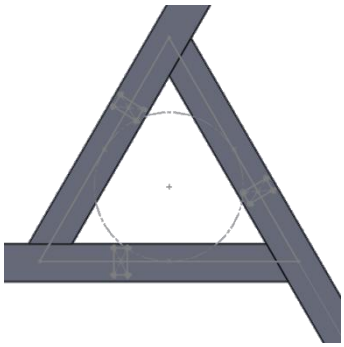


Ilustración 44. Asignación de las posiciones para soportes de los motores.

Además de crear las piezas con los valores resultantes, es decir con los ángulos y distancias necesarias para encajar perfectamente en la estructura realizada se procede a hacer el análisis del nuevo prototipo.

Para hacer una estimación total del volumen del prototipo es necesario hacer la representación de los elementos que conforman, para ello se requiere hacer el modelado de motores, controladores, fuentes de poder, caja contenedora y demás dispositivos que conformaran el robot. Se procedió a hacer una representación tridimensional de los dispositivos que conforman la instrumentación del robot.

4.1.1 Modelo tridimensional del motor

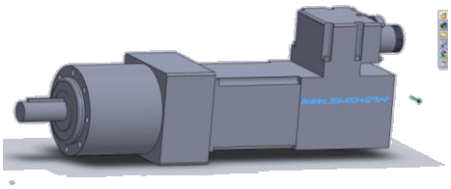


Ilustración 45. Motor Baldor BSM 50N 275 AF

El motor cuenta con cinco piezas fundamentales, la caja de control, donde está el encoder y conexiones necesarias para alimentar el motor, en el centro se encuentra el motor, posteriormente la caja de engranes de baja inercia y la flecha o eje que ejecuta el movimiento.

4.1.2Modelotridimensional del micro Flex e100

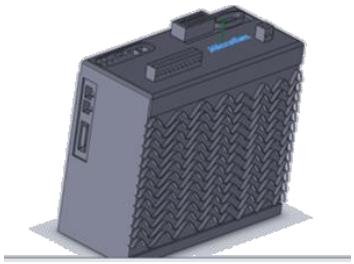


Ilustración 46. Microflex e100

Este dispositivo se encarga de suministrar los parámetros necesarios para mover los motores, se puede conectar vía Ethernet o bien serial con un cable USB a la computadora. También se conecta a la alimentación. Cuenta con un disipador térmico y luces de estado.

4.1.3 Modelo tridimensional del NextMove e100

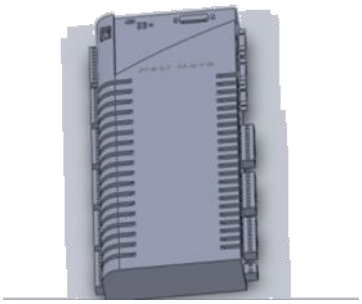


Ilustración 47. Controlador multieje

Cuenta con conexión USB y Ethernet para comunicarse con la computadora y con los micros Flex, además una serie de puertos para otras aplicaciones, la que se usara es la de Ethernet Power Link, para conectar en cascada los demás drivers. También cuenta con bornera de alimentación, luz indicadora y otros protocolos de comunicación.

- Modelo tridimensional del Gabinete, Filtro y Fuente

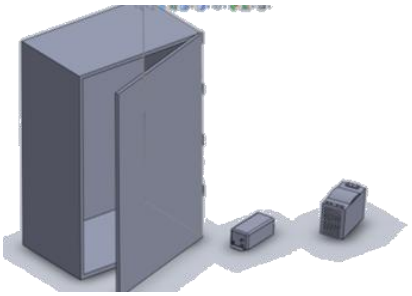


Ilustración 48. Otros dispositivos que se incluirán.

El modelado tridimensional permite visualizar y proyectar mejoras al dispositivo además permite seleccionar tácticamente una buena caja para poder acomodar y organizar los dispositivos. Por otra parte enriquece el proyecto teniendo una imagen previa del prototipo con todos sus componentes.

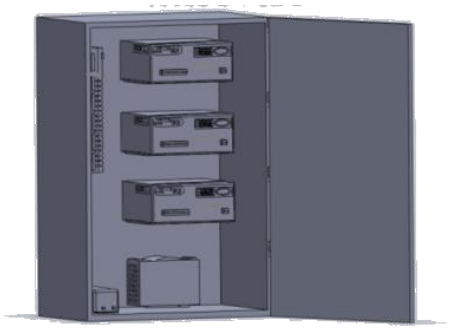


Ilustración 49. Modulo de potencia completo

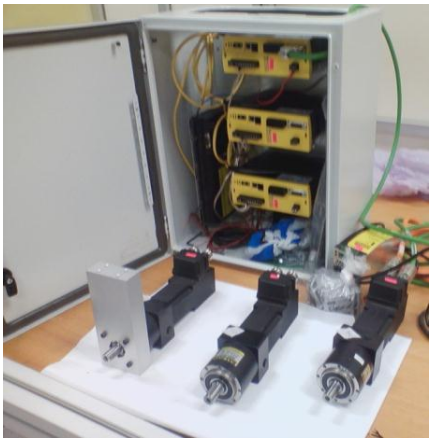


Ilustración 50. Gabinete y servomotores físicamente

El prototipo cuenta con tres cadenas cinemáticas que se cierran con la plataforma móvil, el cual se posiciona en la coordenada X, Y, Z que se desea situar, de acuerdo al estudio de la cinemática inversa.

Para desarrollar el prototipo se recurrieron a usar un par de eslabones para cada brazo los cuales unirán el eje del motor con la base del órgano efector. El primer eslabón se conecta al motor con el acople, seguido por el eslabón medio unidos por articulaciones esféricas y una pieza de aluminio que conectara con la plataforma móvil.

El segundo eslabón consta de dos varillas conectadas con las articulaciones, debido a un juego necesario que evite el sobre esfuerzo en la estructura, ya que muchas fuerzas son sometidas en el mecanismo, el cual debe ser correctamente acoplado.

Diseño de un brazo del robot paralelo acoplado del motor a la plataforma móvil.

la =240mm	lb =790mm	r =100mm	rpm =90mm

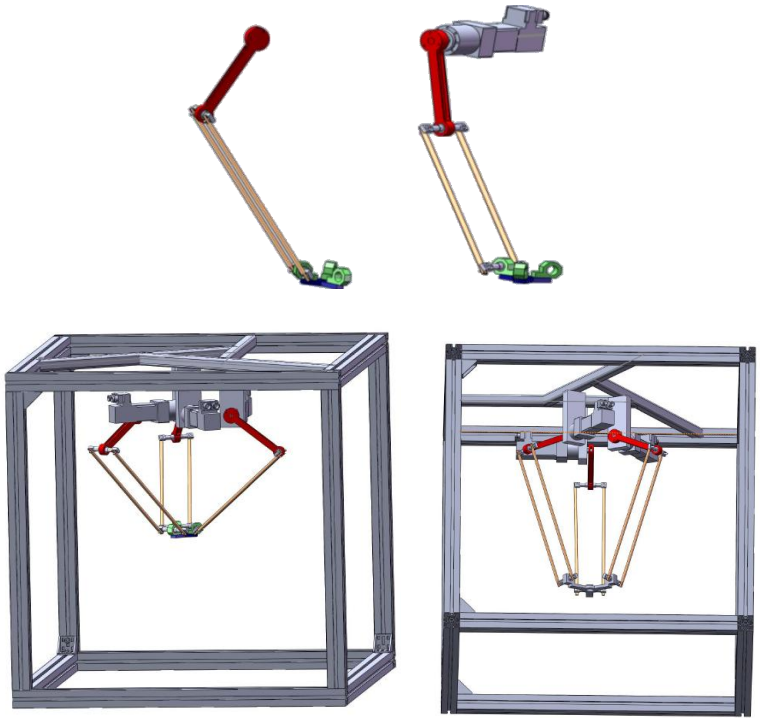


Ilustración 51. Modelo del robot en Solid Works

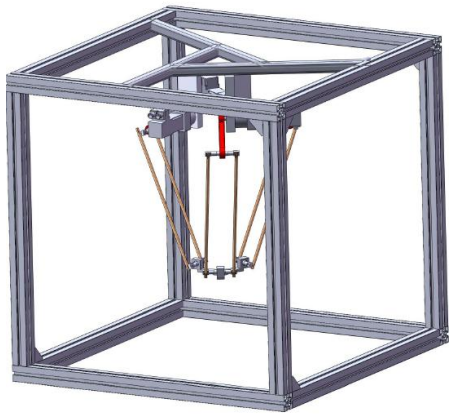


Ilustración 52. Modelo del robot en Solid Works

Con todas estas medidas y siguiendo la simulación, se sacaron los siguientes resultados haciendo un cuadro comparativo con otros modelos como el Adept Quattro S50H, IRB 360-3 Flex Picker y el Parallix LKF-2040.

Ilustración 53. Cuadro comparativo del robot con otros en el mercado y su nueva versión

Especificaciones	Adept Quattro s650H	IRB 360-3/1130 FlexPicker	Parallix LKF-2040	Parallix nueva versión (prescrito)
Alcance en z (altura)	500 mm	300 mm	250 mm	350 mm
Alcance en *x* e *y*	1,300 mm	1,130 mm	600 mm	1,040 mm
Número de ejes	4	4	3	3
Compañía / Institución	Adept Technology, Inc.	ABB (Asea Brown Boveri) Robotics	IPN - CICATA	IPN - CICATA
Nacionalidad	Estados Unidos	Suecia	México	México

De acuerdo a las especificaciones del fabricante se presenta una tabla con los datos sobre el funcionamiento con características eléctricas y mecánicas de los servomotores usados. Donde se rescata un motor reductor de 10:1, cuenta con sus cables especiales y conexiones al controlador.

Ilustración 54. Tabla de datos

ESPECIFICACIONES DEL BSM 50N-275AF			MICROFLEX e100	
Corriente	1.42A(RMS)	4.87Ap	6 A RMS	3A PROM
Voltaje		64.35Vp	320v BUS VOLT	
Inductancia	33.2mH			
Resistencia	16.06 Ohms		ENCODER	
Velocidad Max	10,000rpm		2500 X 4=10,000 Reductor 10:1	
Polos/par	4=2polos/par		1REV=100000	
Velocidad Prom	2000		Torque ConstStall 0.91NM	
Potencia	0.18Kw			

4.2.-ANÁLISIS CON MATLAB

Se usó Matlab para el análisis del espacio de trabajo del robot, de acuerdo con las dimensiones del nuevo prototipo, para ello se realizó el algoritmo mostrado en anexos para saber el alcance del robot dentro del espacio del trabajo prescrito. La región alcanzable por las dimensiones del robot se representa en la gráfica, donde se observa la geometría del espacio de trabajo:

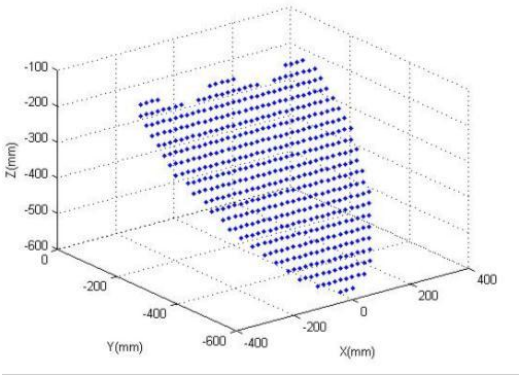


Ilustración 55. Geometría del espacio de trabajo, con puntos de prueba.

Posteriormente se analizó el algoritmo de programación en Matlab para resolver el barrido de puntos que se necesitan para realizar posteriormente el análisis de posibles trayectorias, sin embargo a continuación se determinan ciertos puntos en los ejes x, y,z que permiten desplazarse el motor en el espacio de trabajo declarado según las dimensiones del robot.

Ilustración 56. Captura de X, Y, Z puntos alcanzables por el robot con Pasos=20, muestras=52

X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
160	-220	-340	60	-200	-340	-80	-180	-340	180	-180	-340
-180	-200	-340	80	-200	-340	-60	-180	-340	200	-180	-340
-160	-200	-340	100	-200	-340	-40	-180	-340	-220	-160	-340
-140	-200	-340	120	-200	-340	-20	-180	-340	-200	-160	-340
-120	-200	-340	140	-200	-340	0	-180	-340	-180	-160	-340
-100	-200	-340	160	-200	-340	20	-180	-340	-160	-160	-340
-80	-200	-340	180	-200	-340	40	-180	-340	-140	-160	-340
-60	-200	-340	-200	-180	-340	60	-180	-340	-120	-160	-340
-40	-200	-340	-180	-180	-340	80	-180	-340	-100	-160	-340
-20	-200	-340	-160	-180	-340	100	-180	-340	-80	-160	-340
0	-200	-340	-140	-180	-340	120	-180	-340	-60	-160	-340
20	-200	-340	-120	-180	-340	140	-180	-340	-40	-160	-340
40	-200	-340	-100	-180	-340	160	-180	-340	-20	-160	-340

4.2.1Generación de un cuadrado

Para visualizar el comportamiento del robot, se recurrió al análisis de la cinemática resolviendo las ecuaciones en el programa Matlab, graficando las posiciones que cumple en robot en tal coordenada, se hizo la representación de los eslabones y sus cadenas con la plataforma móvil.

Se hicieron simulaciones con esto para ver el alcance que tiene el robot de una manera práctica y dinámica, ya que se pudo graficar una serie de puntos que permitían simular el comportamiento del robot.

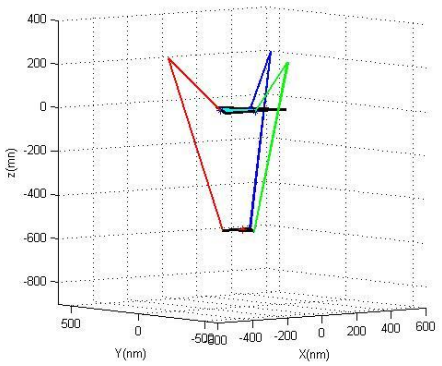


Ilustración 57. Máxima elevación de la plataforma móvil del robot en X= 0,Y=0,Z=-550 , Valor Real 560

Con el modelo anterior es posible verificar las trayectorias, sin embargo se notó que habría que invertir los signos de las posiciones ya definidas por el ajuste mecánico que se hizo con la estructura, por una inconsistencia con los barrenos, sin embargo por programa es fácil implementar este cambio, negando todos los valores dados por la cinemática inversa.

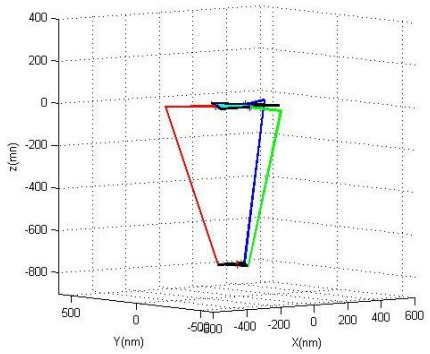


Ilustración 58. Brazos sobre la horizontal en 0,0,750 (posición inicial físicamente)

Valor real 0,0,750

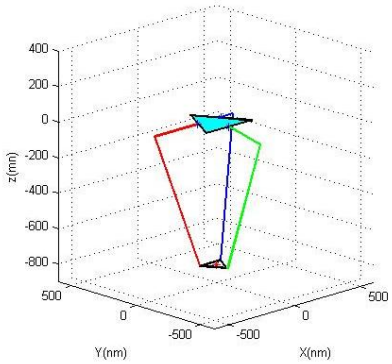


Ilustración 59. Posición mínima del robot Simulada en Z =1029

Real Z=1020

Para iniciar la simulación y generar el código de simulación se procedió a realizar un cuadrado que permita posicionar en cuatro coordenadas en un plano, y así generar una trayectoria básica.

Los puntos para X, Y, Z que se usaron fueron para generar un cuadrado de 60 cm = 600 mm como distancia entre punto y punto. Como es un plano, el eje Z permanece constante. Como resultado de la generación de la trayectoria se obtuvieron los siguientes datos:

Ilustración 60-Descripcion de un Cuadrado

Descripción de un Cuadrado						
Trayectoria	**Inicio**	**Punto 1**	**Punto 2**	**Punto 3**	**Punto 4**	**Punto 5**
X	0	300	300	-300	-300	300
y	0	-300	300	300	-300	-300
Z	-550	-650	-650	-650	-650	-650
Ángulo 1	-87.0684	-2.5637	-27.2263	17.3766	39.7063	-2.5637
Ángulo 2	-87.0684	39.7063	17.3766	-27.2263	-2.5637	39.7063
Ángulo 3	-87.0684	-18.9897	32.4932	32.4932	-18.9897	-18.9897

Como se puede ver en la tabla, se obtuvieron los puntos X,Y,Z, para luego aplicarle la cinemática inversa, y determinar el ángulo de cada uno de los motores para así posicionar el efector final en el punto asignado. Por otra parte el algoritmo diseñado, permite obtener la conversión a cuentas del encoder, para poderlo aplicar al motor.

Teniendo los ángulos del motor, estos son los ángulos absolutos para poder generar la trayectoria, es decir, el ángulo que necesita tener el equivalente en X, Y, Z para lograr esa posición. Se observa que en la simulación aparecen los puntos del cuadrado, todos ellos alcanzables por el robot.

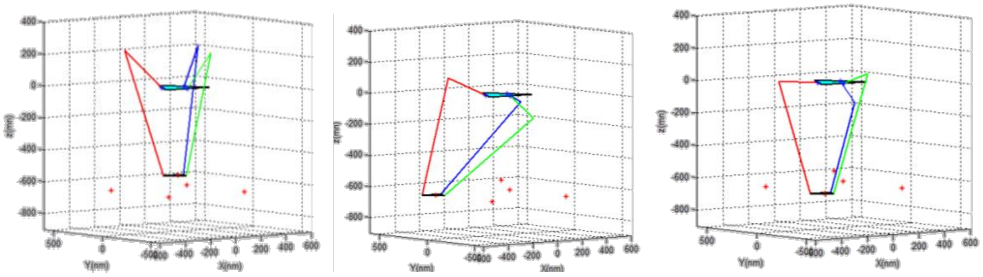


Ilustración 61. Puntos del cuadrado

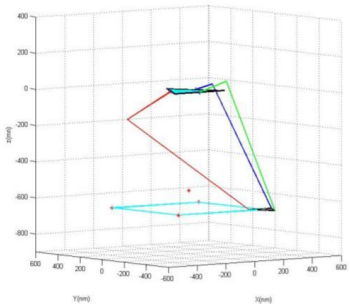


Ilustración 62. Trayectoria finalizada del cuadrado con el robot.

4.2.1Generación de una trayectoria tipo “Pick and Place” con el robot

Una de las tareas comunes de un robot paralelo es posicionar una serie de objetos en una caja en un orden preestablecido para ello se generó una trayectoria tipo pick and place donde simulamos tomar una serie de objetos y colocarlos en 8 diversas pociones.

Para ello se seleccionaron 8 puntos ordenados y un punto de acopio, donde en la simulación se puede visualizar las interacciones de la plataforma móvil. Al igual que la generación del cuadrado, nos permite obtener los ángulos que hacen llegar al robot a tal posición y así mediante un algoritmo llegar a tales lugares.

Ilustración 63. Descripción Del pick and place

Descripción de "Pick and Place"

	Inicio	Pieza 1			pieza 2	…	
Punto		**Punto 1**	**Punto 2**	**Punto 3**	**Punto 4**	**Punto 5**	**Punto 6**
X	0	-400	-400	350	-400	-400	450
y	0	400	-300	200	400	-300	200
Z	-550	-750	-650	-600	-750	-650	-600
Áng. 1	-87.0684	71.9523	54.9812	-41.9122	71.9523	54.9812	-34.552
Áng. 2	-87.0684	8.0176	2.3457	12.0644	8.0176	2.3457	34.0203
Áng. 3	-87.0684	80.8445	-7.0069	11.7763	80.8445	-7.0069	26.8417

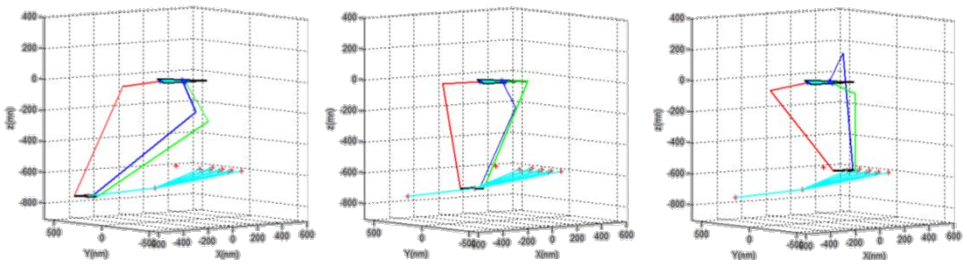


Ilustración 64. Simulación de posicionamiento del robot tipo pick and place.

4.2.2Generación de un círculo

Para la generación de un círculo se recurrió a una función la cual genera una serie de puntos para inscribir un círculo en el plano X, Y, donde Z permanece constante, con el programa en Matlab se asigna la coordenada X, Y, para el centro del círculo, el radio, en el caso de la simulación se recurrió a un circulo con radio 30 cm = 300mm, con un total de 100 puntos.

Estos puntos logran afinar el movimiento del efector, pero entre mas puntos se inscriban, mayor será el tiempo de ejecución. El programa genera los puntos en X, en Y, para luego procesarlas con la cinemática inversa del robot paralelo y simula la generación de la nueva trayectoria.

Programa para la generación del círculo.

```
%% Generar los puntos del círculo

xcentro= input('X centro');

ycentro= input('Y centro');

rcir= input('Radio');

pts= input('No. Puntos');

LS=linspace(0,2*pi,pts)

xd=xcentro+rcir*cos(LS)

yd=ycentro+rcir*sin(LS)

plot(xd,yd),axis('equal'),title(Trayectoria de un circulo')

grid

%%puntos para z constante

zd(1)=-550;   %0

for(pt=2:1:pts)

zd(pt)=-650;%1
```

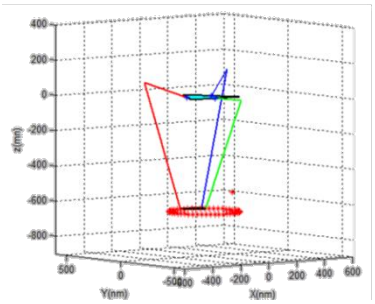


Ilustración 65. Formación de un círculo con el robot

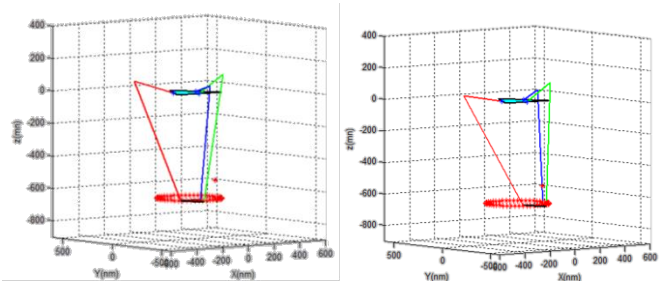


Ilustración 66. Simulación para la generación del círculo.

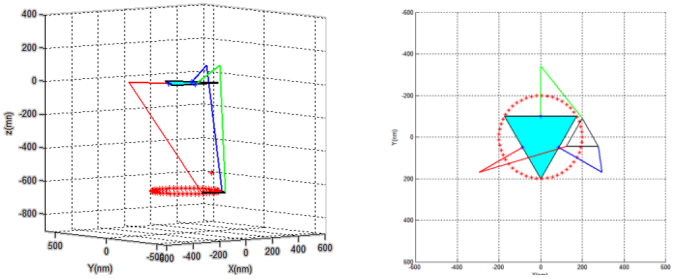


Ilustración 67. Generación de un círculo

4.2.3Generación de una trayectoria tipo Espiral

Como el robot tiene la capacidad para desplazarse en el espacio, se elaboraron trayectorias para visualizar su comportamiento desplazándose variando X, Y, Z simultáneamente. Para ello se proponen figuras conocidas como una espiral y un cono que a continuación se detallan.

Primeramente se generó tales figuras a manera que puedan manipularse fácilmente. Con la función usada, nos da una serie de puntos que son necesarios para poder describir tales figuras. Teniendo esos puntos en X, Y, Z, se analizan con la cinemática inversa para ver si no hay conflictos y una vez analizado nos arroja los ángulos necesarios para describir tales movimientos. Al igual que las anteriores, se simulan para ver el comportamiento y así poderlas aplicar a los motores para después verlos en el robot.

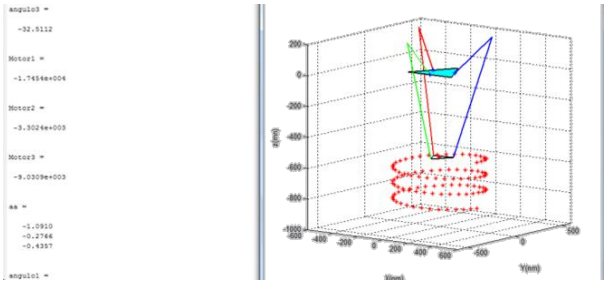


Ilustración 68. Simulación y obtención de la cinemática inversa de la espiral.

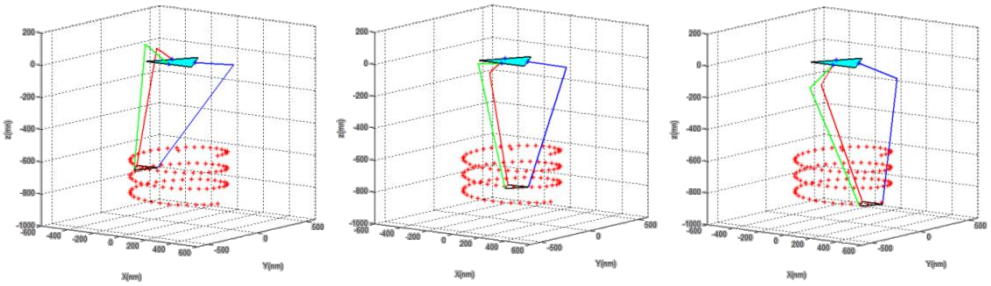


Ilustración 69.Simulación para movimiento de la trayectoria en forma de espiral.

```
%%Codigo para generar la Espiral
xcentro=0;
ycentro=0;
rcircu= 250
pts= 30
piso= 3
paso= 100
fact=piso*2;
pasesp=paso/(pts/piso);
%%puntos
zd(1)=-550;
t=linspace(0,fact*pi,pts+1);
rcir=rcircu;
for(pt=2:1:pts+1)
zd(pt)=zd(pt-1)-pasesp;
end
xd=xcentro+rcir*cos(t)
yd=ycentro+rcir*sin(t)
zd(1)=-550;   %pto inicial
xd(1)=0;
yd(1)=0;
zd(pts+2)=-550;   %ptofinal
xd(pts+2)=0;
yd(pts+2)=0;
```

42.4Generación de trayectoria tipo cono

Con los mismos pasos, se procedió a realizar un cono, el cual va aumentando su radio conforme baja, con la misma relación anterior, haciendo que inicie en el punto 0,0,-550, y al finalizar regresar a la posición inicial.

```
%% figura cono

xcentro=0;
ycentro=0;
pts= 30
piso= 3
paso= 100
fact=piso*2;
pasesp=paso/(pts/piso);
zd(1)=-550;   %pto inicial
xd(1)=0;
yd(1)=0;
radioo(1)=0;
t=linspace(0,fact*pi,pts+1);

for(pt=2:1:pts+1)
zd(pt)=zd(pt-1)-pasesp
radioo(pt)=radioo(pt-1)+pasesp;

xd(pt)=radioo(pt)*cos(t(pt))
yd(pt)=radioo(pt)*sin(t(pt))
end
grid
zd(pts+2)=-550;   %Puntofinal
xd(pts+2)=0;
yd(pts+2)=0;
```

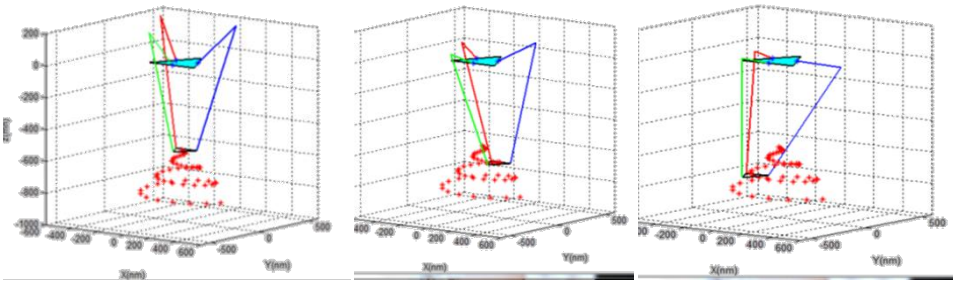


Ilustración 70.Simulación del cono.

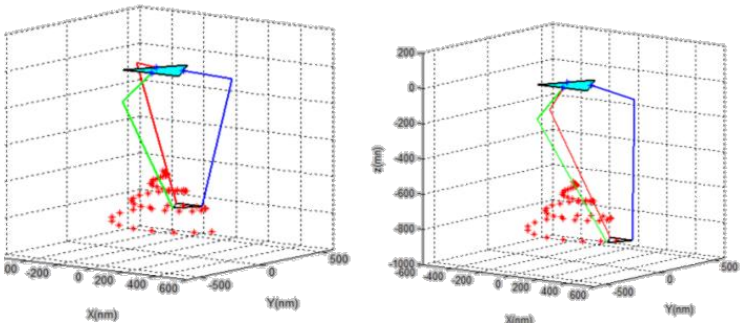


Ilustración 71.Fin de la Simulación del cono

4.3.-PRUEBAS CON MINT WORKBENCH

En este programa se escriben los comandos con la siguiente estructura y tener presente que el encoder genera 10,000 cuentas en cada revolución, pero en el sistema montado se tiene un motor reductor 10:1, por lo que cada vuelta requiere 100,000 cuentas.

```
ACCEL (3)=5000          'aceleración para el eje 3 de 5000cuentas/seg
DECEL (3)=5000          'desaceleración para el eje 3 de 5000 cuentas/seg
SPEED (3)=50000             'velocidad para el eje 3 de 50,000 cuentas/seg
DRIVEENABLE (3)=1           'habilita el driver 3
DRIVEENABLE (3)=0           'deshabilita el driver 3
MOVER (3)=100,000           realiza movimiento relativo de 1rev  en el eje 3
GO (3,1)                    inicia el movimiento en los ejes 3 y 1
Wait (1000)                 retardo de 1000 ms
Loop                        inicia un ciclo
Endl                        termina un ciclo
End                         termina programa
```

Haciendo una regla de tres por cada 90° corresponden 25000 cuentas del encoder, quedando la siguiente relación para movimientos angulares.

Ilustración 72. Correspodencia de angulos y cuentas del motor

$\begin{bmatrix} 45 & y = 12500 \\ 90 & y = 25000 \\ 135 & y = 37500 \\ 180 & y = 50000 \\ 225 & y = 62500 \\ 270 & y = 75000 \\ 315 & y = 87500 \\ 360 & y = 100000 \end{bmatrix}$	o bien	$\begin{bmatrix} \frac{\pi}{4} & y = 12500 \\ \frac{\pi}{2} & y = 25000 \\ \frac{3\cdot\pi}{4} & y = 37500 \\ \pi & y = 50000 \\ \frac{5\cdot\pi}{4} & y = 62500 \\ \frac{3\cdot\pi}{2} & y = 75000 \\ \frac{7\cdot\pi}{4} & y = 87500 \\ 2\cdot\pi & y = 100000 \end{bmatrix}$

Por ejemplo se tiene el siguiente programa para realizar movimientos de tres motores en un ciclo.

```
Dim ang1 As Integer
Dim ang2 As Integer
Dim ang3 As Integer
Loop
  'INICIALIZACION
ACCEL(3)=100000
```

```
ACCEL(4)=100000
ACCEL(5)=100000
DECEL(3)=100000
DECEL(4)=100000
DECEL(5)=100000
SPEED(3)=1000000
SPEED(4)=1000000
SPEED(5)=1000000

DRIVEENABLE(3)=1
DRIVEENABLE(4)=1
DRIVEENABLE(5)=1
'MOVIMIENTOS a 45 GRADOS
  ang1 = Acos(0)'90
  ang2 = Acos(0)'90
ang3 =((Acos(.7071))*100000)/360
MOVER(3)=((ang1)*100000)/360
MOVER(4)=-((ang2)*100000)/360
MOVER(5)=ang3
   Print ang3
GO(3,4,5)
Wait(2000)
  'REGRESAR A POSICION INICIAL
MOVER(3)=-(ang1*100000)/360
MOVER(4)=((ang2)*100000)/360
MOVER(5)=-ang3
GO(3,4,5)
Wait(2000)
MOVER([ 3, 4, 5]) = 10000, 20000, -10000
GO(3, 4,5)
Wait(2000)
MOVER([ 3, 4, 5]) = -10000, -20000, 10000
GO(3, 4,5)
Wait(2000)
Endl' FIN DEL BUCLE
End
```

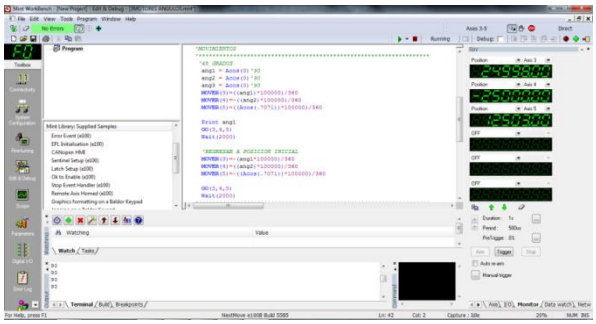


Ilustración 73.Ejecución del algoritmo en Mint W.B.

Se hicieron diferentes pruebas con tal de tener una programación directamente en Mint WorkBench, pero al realizar un bucle, se produjeron errores de precisión al revolucionar, se reajustaron valores de calibración para analizar estas anomalías.

Se ajustó la herramienta llamada KFint, el cual ajusta la ganancia integral del controlador de flujo para el control de motores. Recordando que internamente tienen controladores PID, también se le puede cambiar la ganancia proporcional, según el ajuste del error.

Se notó un movimiento deficiente, sin embargo se implementaron otras herramientas para lograr la precisión de todos los movimientos. Pero resultaron movimientos con bajo desempeño en velocidad y precisión.

Se resolvió esta problemática con rutinas y segmentos en LabView y posteriormente en Visual Basic. Con esto se logró introducir en varios ciclos donde el usuario podría seleccionar entre varias opciones como posicionar los motores, usando movimientos absolutos, relativos y Splines.

Ilustración 74.Tabla de resultados en diferentes ciclos asignados y realizados por el motor.

# De Ciclos	Ángulo por ciclo	Desaceleración	Parámetros (Aceleración,Velocidad)	Cuentas totalasignadas	Cuentas realizadas
12	30	100000cuentas/seg	100000 cuentas/seg	100000 (1rev)	83730
12	30	100000 cuentas/seg	100000 cuentas/seg	100000 (1rev)	83968
24	30	100000 cuentas/seg	100000 cuentas/seg	200000 (2rev)	167590
12	30	1,000,000cuentas/seg	100000 cuentas/seg	100,000	92292
12	30	2,000,000cuentas/seg	100000 cuentas/seg	100,000	92114
1	360	1,000,000cuentas/seg	100000 cuentas/seg	100,000	100534

4.4.- PROGRAMA EN VISUAL BASIC

Se procedió a manipular los motores Baldor con la nueva interface en Visual Studio. Para ello se requieren los siguientes pasos:

Abrir Visual Studio hacer nuevo proyecto dando clic en FILE, NEW PROJECT. En la ventana emergente, dar clic en VISUAL BASIC, y seleccione Windows FormApplication, asigne el nombre del nuevo archivo, ruta de acceso y dele clic en OK.

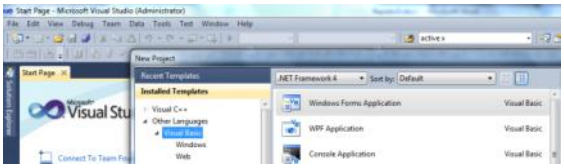


Ilustración 75. Nuevo proyecto de Visual Basic

Aparece la ventana con el formulario donde podrá introducir botones, etiquetas, controles en fin todas las herramientas que tiene este programa. El entorno de Visual Basic es muy parecido a otros programas, ya que es una aplicación en C, C++. Agregando los controles y herramientas podemos acceder a la ventana de programación.

Ilustración 76.Hacer una aplicacion rápida

Como es la primera vez que se usaran los controles de Mint WorkBech es necesario agregarlos a los componentes del ToolBox. Para ello podemos ir al menú Project, add reference, seleccionar COM.Ahí podemos visualizar nuestros complementos para controlar los motores.

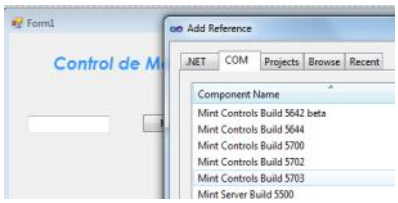


Ilustración 77.Seleccionar el controlador

Desde aquí podemos habilitarlos, sin embargo para agregarlos al Toolbox, podemos introducirlos con lo siguiente:

Ir al menú Tools, ChooseToolbox ítems.

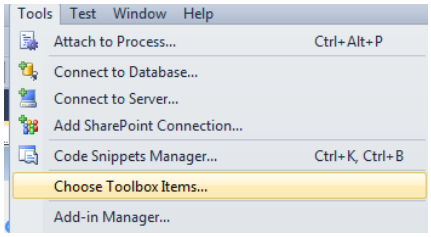


Ilustración 78.Menú Herramientas

Aparece la ventana que nos muestran los controles que podemos anexar. Seleccionamos COM Components, y habilitamos las casillas necesarias. En este caso damos clic en los Mint ControllerBuild 5302, según la versión del Controlador que se tenga. Posteriormente se da clic en aceptar.

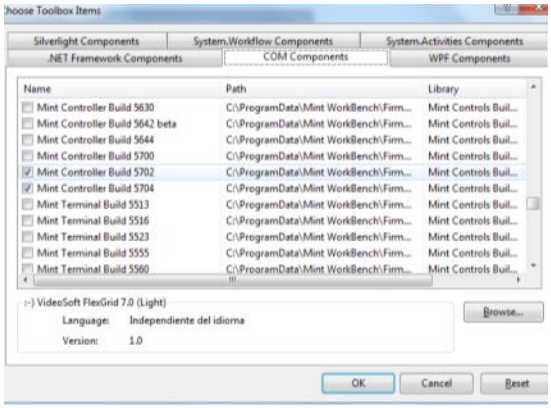


Ilustración 79.Agregar componentes de Activex

En el Toolbox aparece el ícono Mint ControllerBuild 5702, para nuestro caso, este número es importante ya que se requiere al momento de usar las herramientas requeridas en los algoritmos. Agregamos el control al formulario actual para empezar a programar los motores.

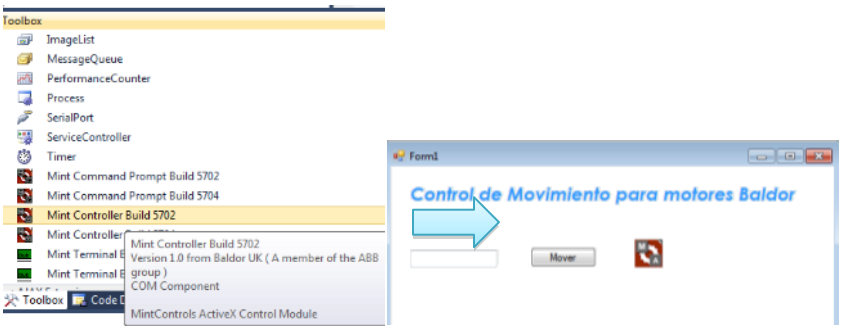


Ilustración 80.Aplicación de prueba

La declaración que pondremos a continuación es para lograr la comunicación entre la PC con los drivers del motor. Se inicializó usando esta variable para no tener que repetir toda la ruta, resumiéndola como sigue, en este caso una ‘ a ‘ arbitrariamente solo para probar la conexión.

```
PublicClassForm1
DimaAsNew MintControls5702Lib.MintController
DimpossAs Int32

PrivateSub Form1_Load(ByVal sender AsSystem.Object, ByVal e AsSystem.EventArgs)
HandlesMyBase.Load

a.SetUSBControllerLink(1)    'Puerto de conexión actual
EndSub

PrivateSub Button1_Click(ByVal sender AsSystem.Object, ByVal e AsSystem.EventArgs) Handles
Button1.Click

TextBox1.Text = poss'la posición que deseemos mover
a.Accel(3) = 10000'la configuración de los parámetros
a.Decel(3) = 10000
a.Speed(3) = 10000
a.DriveEnable(3) = 1
a.MoveR(3) = poss
a.DoGo1(3)
EndSub
```

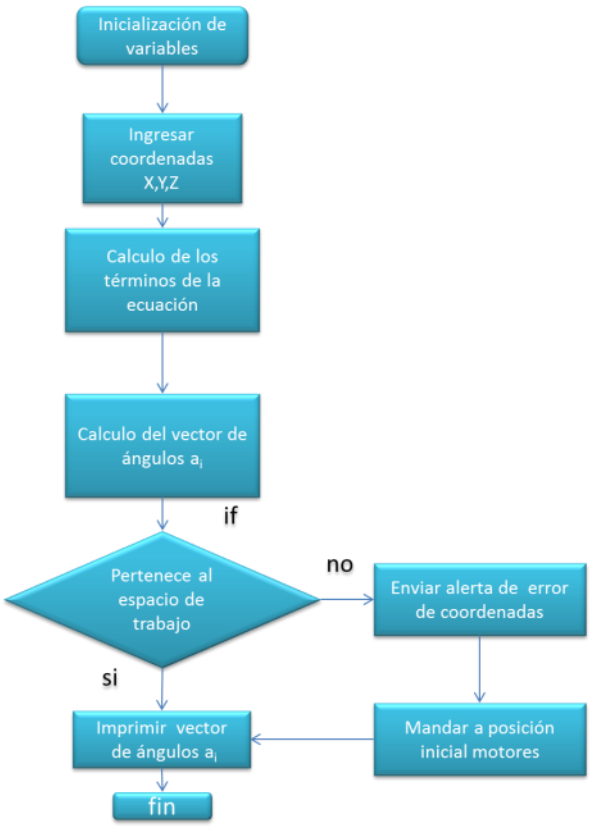
Una de las tareas que realizará el robot consiste en situarse en su espacio de trabajo, siendo necesario restringir posiciones ajenas a estas condiciones. Para ello es importante tener en el software de control, un algoritmo que calcule estas restricciones, llamado modelo inverso, o cinemática inversa, el cual calcula los valores de los ángulos en función de la posición X, Y, Z que se le asigne definiendo los movimientos del robot.

Una vez lograda la manipulación de motores Baldor, se procedió a pasar el código de la cinemática inversa anteriormente desarrollada, con algoritmos en C++, para pasarlos procesarlos con Visual Studio.

Para ello se resuelve la ecuación del robot quedando:

$$tan\frac{\alpha_i}{2} = \frac{-2z_p \pm \sqrt{4z_p^2+4R^2-s^2-q_i\left(4R+\frac{2sR}{La}\right)+{q_i}^2\left(1-\frac{R^2}{La^2}\right)}}{-2R-s-q_i\left(\frac{R}{La}-1\right)}$$

El diagrama a bloques de programación resulta:



El algoritmo determina la configuración que debe adoptar el robot para alcanzar una posición y orientación deseada. Esta parte de la interface de manera generalizada, permite al usuario introducir unas coordenadas X, Y, Z y el algoritmo regresa, si se trata de una coordenada correcta y admisible, los tres ángulos que deben girar cada uno de los motores para poder efectuar tal movimiento.

El programa resultante quedó como un módulo para poderlo usar cada que la aplicación lo requiera, codificándolo de la siguiente manera:

```
Moduletrayectorias
Function cinematica(ByVal Xc AsDouble, ByVal Yc AsDouble, ByVal Zc AsDouble, ByRef matrizc()
AsDouble) AsBoolean


Dim g(0 To 2) AsDouble
Dim la AsDouble = 240
Dim lb AsDouble = 790
Dim pi AsDouble = 3.1415926535897931
Dim L, j, beta, x0, y0, z0, X(0 To 2), Y(0 To 2), Z(0 To 2), r2, lb2, la2, st(0 To 2), ct(0 To 2), alfamax,
alfamin, alfat(0 To 2) AsDouble
```

```
Dim q(0 To 2), s(0 To 2), r_m, b1(0 To 2), b2, b3(0 To 2), b7(0 To 2), b4(0 To 2), b5(0 To 2), b6(0 To
2), aa(0 To 2) AsDouble
Dim rad AsDouble = 100
Dim rpm AsDouble = 90
Dim teta() AsDouble = {pi / 6, 5 * pi / 6, -pi / 2}
Dim X1, Y1, Z1 AsDouble
X1 = Xc
    Y1 = Yc
    Z1 = Zc
'**********
    r_m = rad - rpm
    L = Math.Sqrt(lb ^ 2 + la ^ 2 - 2 * la * lb)
beta = Math.Asin(r_m / L)
    alfamin = -((pi / 2) - beta)
    alfamax = pi / 2
Dim ff AsShort
For ff = 0 To 2
        ct(ff) = Math.Cos(teta(ff))
        st(ff) = Math.Sin(teta(ff))
Next

    lb2 = lb * lb
    la2 = la * la
r2 = rad * rad
    b2 = 4 * r2
    j = 1
    x0 = X1
    y0 = Y1
z0 = Z1

For ff = 0 To 2
        X(ff) = x0 + rpm * ct(ff)

Next

For ff = 0 To 2
        Y(ff) = y0 + rpm * st(ff)
Next
For ff = 0 To 2
        Z(ff) = z0
Next
Dim f(0 To 2) AsDouble
Dim aux1, aux2 AsDouble
For ff = 0 To 2
        aux1 = 0
        aux2 = 0
        aux1 = 2 * X(ff) * ct(ff)
        aux2 = 2 * Y(ff) * st(ff)
```

```
        q(ff) = aux1 + aux2
Next
For ff = 0 To 2
s(ff) = (1 / la) * (-1 * X(ff) * X(ff) - 1 * Y(ff) * Y(ff) - 1 * Z(ff) * Z(ff) + lb2 - la2 - r2)
Next
For ff = 0 To 2
        f(ff) = (-1 * X(ff) * X(ff) - 1 * Y(ff) * Y(ff)) - 1 * Z(ff) * Z(ff) + lb2 - la2 - r2
Next
For ff = 0 To 2
        b1(ff) = 4 * Z(ff) * Z(ff)
b3(ff) = q(ff) * q(ff) * (1 - (r2 / la2))
b4(ff) = q(ff) * ((-2 * rad * s(ff) / la) - 4 * rad)
b5(ff) = q(ff) * (rad / la - 1)
b6(ff) = (-2 * rad - b5(ff) - s(ff))
        b7(ff) = b1(ff) + b2 - s(ff) * s(ff) + b3(ff) + b4(ff)
Next
If ((Not (b6(0) = 0)) And (Not (b6(1) = 0)) And (Not (b6(2) = 0)) And b7(0) >= 0 And b7(1) >= 0 And
b7(2) >= 0) Then
For ff = 0 To 2
            alfat(ff) = (-2 * Z(ff) - Math.Sqrt(b7(ff))) / b6(ff)
            aa(ff) = 2 * Math.Atan(alfat(ff))
Next
Else
For ff = 0 To 2
aa(ff) = -0.4639 ' En caso de ser indeterminada Poner en  0,0,-650,
Next
        MsgBox("Error, Coordenadas indeterminadas", Title:="Error en Modelo inverso")
ReturnFalse
Exit Function
EndIf
If ((aa(0) > alfamin And aa(1) > alfamin And aa(2) > alfamin) And (aa(0) < alfamax And aa(1) <
alfamax And aa(2) < alfamax)) Then
For ff = 0 To 2
            matrizc(ff) = aa(ff)
Next
Else
        MsgBox("La coordenada no está en el espacio de trabajo")
For ff = 0 To 2
            aa(ff) = -0.4639 ' Colisión, situar  en  0,0,-650,
Next
EndIf
ReturnTrue' carga aa
EndFunction
Function angulos(ByVal Xc AsDouble, ByVal Yc AsDouble, ByVal Zc AsDouble, ByRef matrizc()
AsDouble) AsBoolean
Dim a1, a2, a3 AsDouble
Dim matriza(0 To 2) AsDouble
Dim check AsBoolean
```

```
    check = cinematica(Xc, Yc, Zc, matriza)
If (Not (check = True)) Then
MsgBox("Valor no permitido...", Title:="Modelo inverso")
For ff = 0 To 2
        matrizc(ff) = 0 ' posicion inicial
Next
ReturnFalse
Exit Function
EndIf
a1 = matriza(0) * 180 / 3.1415926535897931
a2 = matriza(1) * 180 / 3.1415926535897931
    a3 = matriza(2) * 180 / 3.1415926535897931
    matrizc(0) = (100000.0 * a1) / 360.0
    matrizc(1) = (100000.0 * a2) / 360.0
    matrizc(2) = (100000.0 * a3) / 360.0
ReturnTrue
EndFunction
EndModule
```

Como se alcanza a apreciar, se calculan ángulos y la conversión a cuentas del motor para no realizar más operaciones en la interface, si no con el modulo se minimiza la demanda de computo, acelerando el proceso de todo el algoritmo.

Un ejemplo de uso de esa función se describe a continuación donde se leen las coordenadas X, Y Z de archivos Txt para luego ser procesados con la cinemática anterior y luego mandar a mover los motores Baldor. En esta aplicación se guardan así mismo las posiciones resultantes para cada coordenada y así pasarlos a los Splines.

```
Dim gg AsInteger = 0
Using MyReader AsNew _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox6.Text)
MyReader.TextFieldType = FileIO.FieldType.Delimited
      MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
        currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow

          gg = gg + 1

Next
```

```
EndWhile
EndUsing
Dim puntosx(0 To gg) AsSingle
Dim puntosy(0 To gg) AsSingle
Dim puntosz(0 To gg) AsSingle
Dim conter1 AsInteger = 1
Dim conter2 AsInteger = 1
Dim conter3 AsInteger = 1
Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox6.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                puntosx(conter1) = currentField
                conter1 = conter1 + 1
Next
EndWhile
EndUsing

Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox7.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                puntosy(conter2) = currentField
                conter2 = conter2 + 1
Next
EndWhile
EndUsing

Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox25.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
```

```
            puntosz(conter3) = currentField
            conter3 = conter3 + 1
Next
EndWhile
EndUsing

Dim puntos3(0 To gg) AsSingle
Dim puntos4(0 To gg) AsSingle
Dim puntos5(0 To gg) AsSingle
Dim ss AsInteger
Dim textito1, textito2, textito3 AsString
For ss = 1 To gg
angulos(CDbl(puntosx(ss)), CDbl(puntosy(ss)), CDbl(puntosz(ss)), matrizCu)

puntos3(ss) = CInt(matrizCu(0)) * -1 'Cuentas del Motor
        puntos4(ss) = CInt(matrizCu(1)) * -1
        puntos5(ss) = CInt(matrizCu(2)) * -1
        textito1 = CStr(puntos3(ss))
        textito2 = CStr(puntos4(ss))
textito3 = CStr(puntos5(ss))

My.Computer.FileSystem.WriteAllText("File1.txt", _
ss &".- "&"X= "& puntosx(ss) &", Y= "& puntosy(ss) &", Z= "& puntosz(ss) & vbCrLf, True) 'con
retorno de carrro vbCrLf

My.Computer.FileSystem.WriteAllText("ABC.txt", _
ss &".- "&"M3= "& textito1 &", M4="& textito2 &", M5="& textito3 & vbCrLf, True) 'guarda las
cuentas con separador
My.Computer.FileSystem.WriteAllText("3.txt", _
textito1 & vbCrLf, True)  'guarda las cuentas sin formato
My.Computer.FileSystem.WriteAllText("4.txt", _
textito2 & vbCrLf, True)  'guarda las cuentas sin formato
My.Computer.FileSystem.WriteAllText("5.txt", _
textito3 & vbCrLf, True)  'guarda las cuentas sin formato
Next
    a.DriveEnable(3) = 1
    a.DriveEnable(4) = 1
    a.DriveEnable(5) = 1
If (a.ErrData(1) > 0) Then
        MsgBox(a.ErrString, Title:="Error")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")
        MsgBox("¿Limpiar Errores?", MsgBoxStyle.YesNo, Title:="Error en Axis")
If (MsgBoxResult.Yes) Then
            a.DoErrorClear(3, -1)

Else
            MsgBox("No se podra ejecutar movimientos"& a.ErrString, Title:="Error")
```

```
EndIf
Else
        puntos3(0) = CInt(gg) ' total de datos
        puntos4(0) = CInt(gg)
        puntos5(0) = CInt(gg)
        a.SplineTable(3, puntos3, 0, 0)
        a.SplineTime(3) = 500
        a.Spline(3) = 1 Or 2
        a.SplineTable(4, puntos4, 0, 0)
        a.SplineTime(4) = 500
        a.Spline(4) = 1 Or 2
        a.SplineTable(5, puntos5, 0, 0)
        a.SplineTime(5) = 500
        a.Spline(5) = 1 Or 2

a.DoGo3(3, 4, 5)
EndIf
     total = 2

EndSub
```

4.5.- PROGRAMACIÓN DE LA SECUENCIA HOMING DESDE MINT WORKBENCH

El controlador y los drivers de los motores Baldors, poseen terminales de entras y salidas digitales, las cuales servirán para implementarle al robot una serie de interruptores de fin de carrera, (limits witches), los cuales permitirán saber la posición limite o de referencia para los brazos. Se puede implementar en los drivers (Microflex) o bien en el controlador (NextMove).

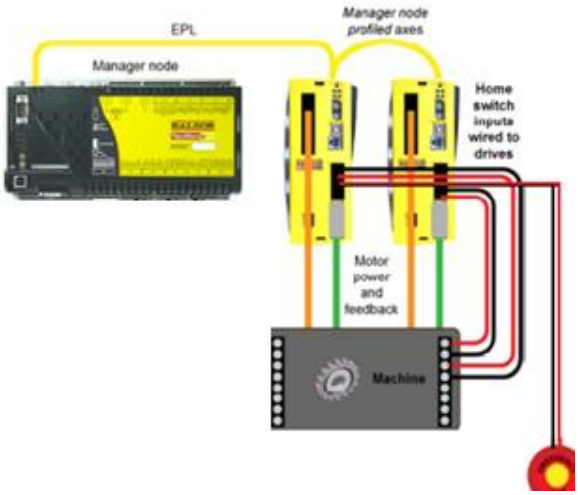


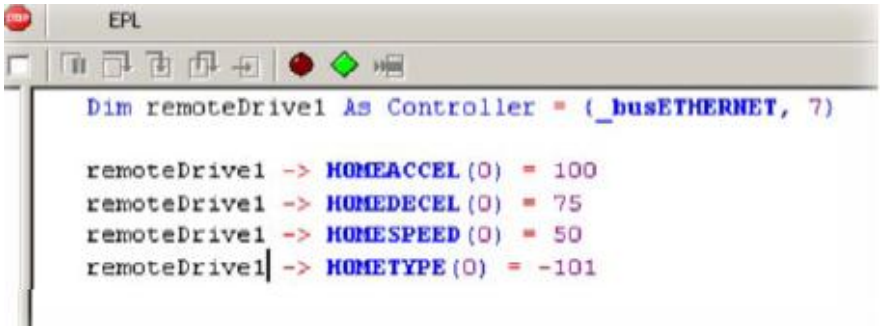
Ilustración 81. Homing

Es necesario hacer una rutina que permita colocar al robot en una posición conocida de referencia como punto de partida para poder hacer las demás ejecuciones. Ya que solo se cuenta con los encoders de los motores los cuales son incrementales y no cuentan con un cero absoluto, desde el punto de vista del robot.

Con las lecturas de las entradas se procedió a elaborar un algoritmo en el cual los brazos se extendieron paulatinamente de manera que presione correctamente el sensor posicionado sobre la horizontal de los brazos, ya que esta es la posición 0°, 0°,0° que adopta el robot.

Al llegar a esta posición los brazos activan la señal y cada uno es desacelerado para poder cambiar el valor de los encoders. Para ellos se usaron una serie de funciones que Baldor integra en el lenguaje de programación para poder realizar este proceso.

Para instrumentar este proceso, se usaron 3 limits witches normalmente abiertos, adquiridos por separado, así como cables y soldadura para hacer el circuito correspondiente. Una de las formas de ejecutar este proceso según Baldor es con la siguiente secuencia:



Pero no resultó favorable, para la implementación en visual Basic, así que diseñó otro algoritmo para funcionara tanto en Mint Workbench como en Visual Basic, ya que ahí se hará el programa final, como se ve a continuación para un motor:

```
Cls
Dim poAs Integer
Dim VV As Controller= {_busETHERNET, 3}    'Motor 3
VV->HOMEINPUT(0) = 0  'input 0, para el driver 3
Print VV->HOMESWITCH(0) 'imprimir el valor de la entrada
DRIVEENABLE(3)=1  'Configuración de movimiento absoluto
ACCEL(3)=10000
DECEL(3)=10000
SPEED(3)=5000
Loop
If (VV->HOMESWITCH(0)=1 ) Then  'Condicion
'  STOP3
Pause(IDLE(3))
Print "Movcompleto en 3"

POS(3) = 25000
Else
MOVER(3) = 500
GO(3)
  End If
Endl
'****************** fin del programa
```

4.6.- CODIFICACIÓN DEL ALGORITMO DE POSICIONAMIENTO INICIAL

Tomando el código en Mint WorkBench, bajo la misma lógica de programación se procedió a realizar el código en Visual Basic, ahora con los métodos del active X, primero se procuró hacer las lecturas del controlador, bajo el mismo esquema de leer 1's y 0's que se le introdujo mediante los interruptores de fin de carrera.

Ilustración 82.Botones agregados para la implementacion.

El programa se introdujo en un botón el cual se llama Posición inicial, para que el usuario oprima el botón y se inicie la secuencia de posicionamiento, el cual lleva el siguiente código para procesar el Homing en un servomotor.

```
a.DriveEnable(5) = 1
MsgBox("Iniciando Calibración,  Espere...")
DoWhile (VV = 0 Or VV2 = 0 Or VV3 = 0)
        VV = a.In(1) '#bank
        VV2 = a.In(2)
        VV3 = a.In(3)
If (VV = 1) Then'  STOP3
If (a.Idle(3) = True) Then
a.DoWait(200)
a.Pos(3) = 0
EndIf
Else
a.DoWait(200)
a.Accel(3) = 10000
a.Decel(3) = 10000
a.Speed(3) = 10000
a.MoveR(3) = 500  'Avanzar hasta topar con el sensor de fin de carrera
a.DoGo1(3)

EndIf
Loop
MsgBox("Finalizado...")
EndSub
```

Es necesario agregar condiciones que permitan tomar alternativas en caso de conflictos con el controlador, como la desconexión o des habilitación de drivers y solucionar problemas con diagnóstico de los errores. Se implementó una barra de estado que permita visualizar información de los motores en tiempo real, si presenta alguno error y si están habilitados o deshabilitados para proceder su accionamiento.

Se implementó un controlador virtual que permite correr el programa sin causar errores y realizar inspección de la interface sin necesidad de conectar el controlador Next-Move e100 a la computadora.

La interface final queda de la siguiente manera, donde se puede interactuar facilmente, cuenta con una seccion de informacion para aclarar cualquier duda que el usuario tenga, así mismo cuenta con una galería en menu proyecto, para visualizar simulaciones y gáficas del robot.

Al abrir la aplicación, aparecerá una ventana de generación de archivos .txt para registro del programa. De clic en aceptar.

Posteriormente aparece la siguiente ventana que es el área de trabajo de la aplicación:

Ilustración 83.Ventana de la aplicación.

Podemos ver una barra con tres menús.

- Proyecto
  - Gráficas .-Se pueden visualizar gráficas y simulaciones del robot

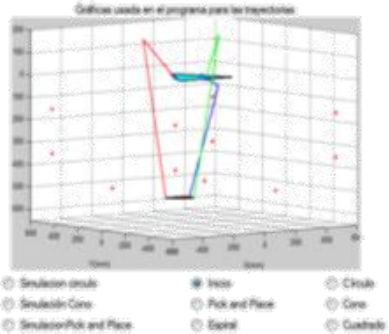


Ilustración 84. Ventana de simulaciones

- Salir.- Permite abandonar el proyecto de manera Segura. Con esto se deshabilitan los motores al salir de la aplicación.
  - Herramientas

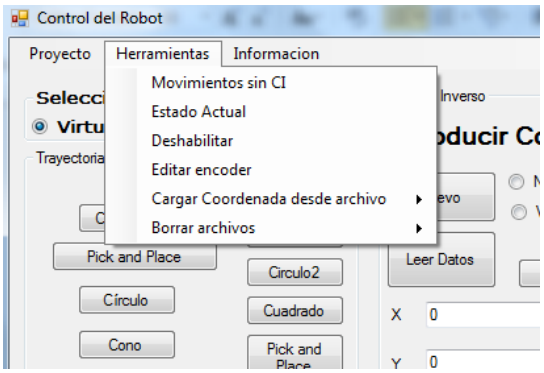


Ilustración 85.Barra de menús de la aplicación

- Movimientos sin Cinemática inversa.- Permite desplazar el robot de manera absoluta por medio de Splines **sin analizarla con el modelo geométrico inverso**.
- Estado actual.- Permite visualizar cuatro datos de los drivers para ver si alguno tiene errores que puedan alterar el funcionamiento del robot o la interface.
- Deshabilitar.- Deshabilita los motores.
- Editar encoder.- Permite Cambiar los valores del encoder sin mover el motor, haciendo un ajuste a la posición puesta. Se puede probar haciendo movimientos relativos con el botón MOVER.
- Cargar Coordenadas desde archivo.
  - Cuentas, permite cargar tres archivos, (Motor1, Motor2 y Motor3)
    Los cuales deberán tener las posiciones que se deseen introducir al robot. estos datos deberán estar posicionados y analizados con el modelo geométrico del robot ya serán movimientos directamente al robot sin procesamiento alguno.

  - Coordenadas.- Permite agregar tres archivos con X, Y y Z para movilizarse analizando tales puntos con la cinemática inversa. Si alguna posición estuviese fuera del espacio de trabajo se pondrá a 0, posición inicial del robot.

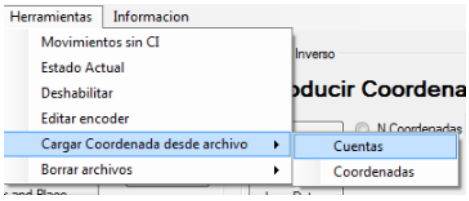


Ilustración 86. Barra de menús

- Borrar archivos.- Borra los archivos generados para el cálculo de la cinemática inversa y registros mostrados con el botón leer Datos, los cuales se van a cumulando y con esta sección se limpia para evitar problemas de conflictos de datos. Si requiere usar tal historial puede ir directamente a C:\ archivos.txt para visualizar o bien con el botón “Leer datos”.

- Información.- Despliega un mensaje con cada parte del programa, para aclarar cualquier duda en las diversas áreas de la interface.

Por default la aplicación se inicia con el controlador Virtual de Mint, para cambiarlo al controlador físico asegúrese de que esté conectado y encendido. De lo contrario dar no para evitar conflictos con los controladores.

Ya sea con el controlador virtual o con el NextMove e100, se puede operar en la interface pudiéndose encontrar:

- Trayectorias

Presenta diversas trayectorias analizadas en Matlab® para su ejecución. De preferencia usar las del lado derecho a menos que se tenga una calibración correcta de motores y contrapesos correctos.

- Modelo inverso

Permite introducir coordenadas ya sea con un numero finito de elementos o los N número de veces que se introduzcan para analizarlas con el modelo inverso.

- No Coordenadas.
  - Ingrese el número de posiciones y de Enter.
  - Introduzca los valores X presione Enter, Y presione Enter y Z presione Enter, al llegar a al número automáticamente le solicitará si desea mover los motores en tales posiciones calculadas.

- Variantes.-introducir múltiples coordenadas. Cuando finalice dar clic en Ejecutar movimiento y se procederá con el movimiento.

- Estado de motores.- se visualiza información de los encoders errores y demás datos necesarios para que se puedan ejecutar los movimientos.

4.7.-GRÁFICAS DE LA RESPUESTA DEL SISTEMA

Durante las pruebas realizadas en las etapas de construccion se obtuvieron diversas gráficas del comportamiento de los motores conforme se iban agregando peso o carga a los eslabones. Por ejemplo no era la misma respuesta de la velocidad- posicion cuando estaba el eje solo, a cuando se le agregó el brazo y posteriormente se le agregó la cadena cinemática cerrada.

A continuacion se visualiza las mas representativas. Los tres motores operaban de manera distinta, a pesar de tener las mismas carácterísticas electricas por ello se sintonizaron las ganancias del PID hasta lograr estabilidad de los motores como se ve en las gráficas.

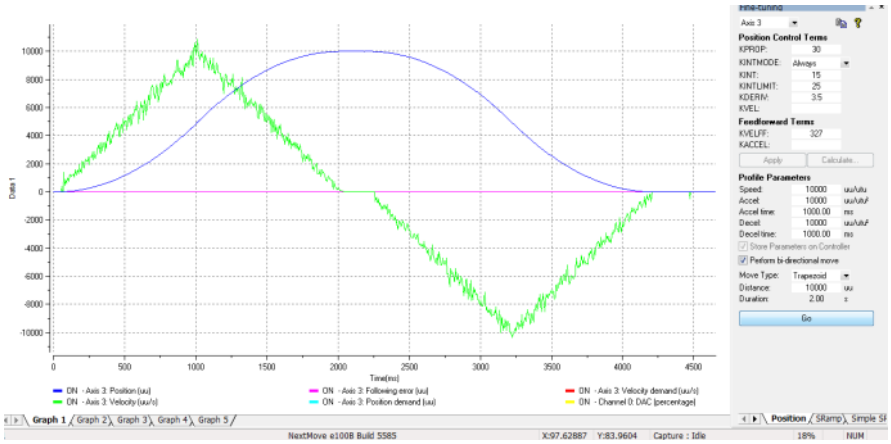


Ilustración 87.Respuesta de los motores con carga ayudado del resorte

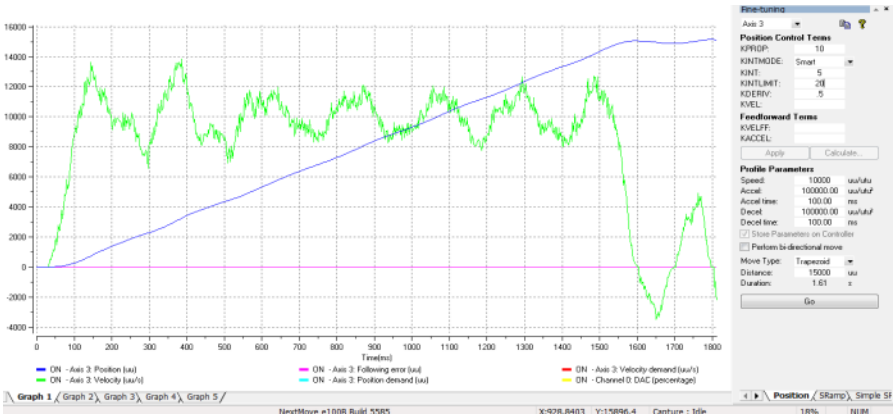


Ilustración 88.Oscilaciones por sobrecarga sin accion del resorte

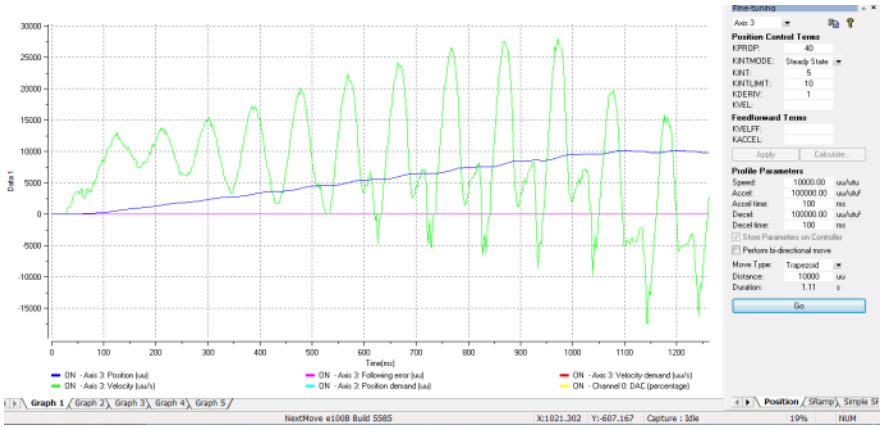


Ilustración 89.Oscilaciones por carga total del sistema

Cabe resaltar que al someter los servomotores con las cargas, se generaron oscilaciones que el PID no podía resolver, teniendo que ajustar de manera manual los valores de las constantes K para así poder disminuir o eliminar tales curvas que provocaban movimientos innecesarios en todo el robot, por la cadena cinemática cerrada. Al finalizar la calibración manual, se obtuvo un conjunto de valores para cada zona de interacción del robot.

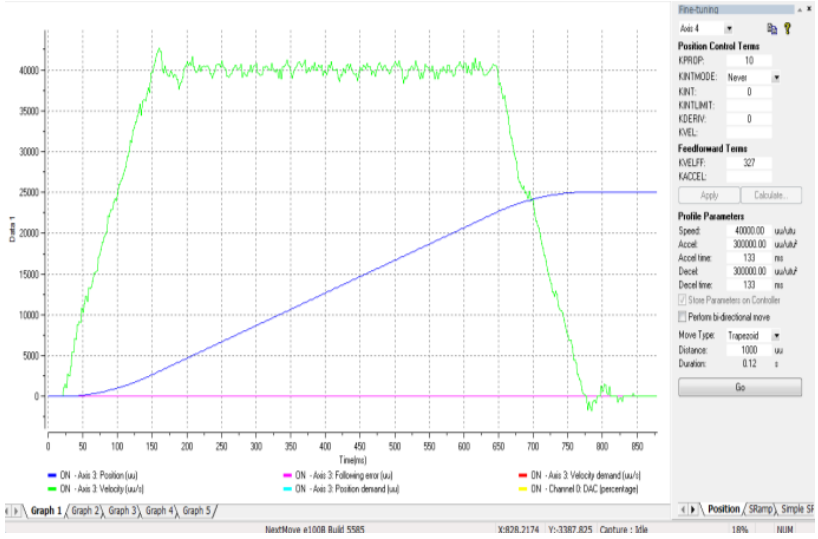


Ilustración 90.Gráfica ideal para de sintonización del PID.

CONCLUSIONES Y RECOMENDACIONES

Se presentó el desarrollo de un robot paralelo tipo delta, desde el diseño hasta la construcción, integración de los controladores y la instrumentación con equipo industrial, así como la programación de los controladores mediante una interface de usuario con trayectorias dentro de su espacio de trabajo.

Los resultados son muy alentadores ya que es una línea de investigación en la que se le integrará un modelo reconfigurable y con estos avances se tiene grandes logros en cuanto a posicionamiento en las tres coordenadas analizadas en este documento.

El análisis de las trayectorias es de suma importancia para manejar los motores con la cinemática inversa, con los cálculos realizados y a realizar, ya que se tiene que comprobar y analizar la cinemática total en el trazo de trayectorias como se hizo en esta parte del proyecto.

Con el uso de motores y controladores de la marca Baldor se obtuvieron herramientas de alta tecnología que permiten realizar movimientos muy precisos y de gran velocidad para realizar operaciones delicadas y repetitivas por ello la importancia de usarlos correctamente para sacar el máximo provecho.

El diseño en software asistido por computadora permitió analizar la estructura visualizando las ventajas y desventajas que tiene el prototipo y así realizar los ajustes necesarios antes de manufacturar las piezas en metal, y tener una plataforma con base a los cálculos realizados para la cinemática inversa del robot paralelo.

La programación en Visual Basic favoreció al accionar los motores con el análisis de la cinemática inversa al momento, dándonos los ángulos correctos que el robot puede ejecutar y llevar a cabo las trayectorias que el usuario le introduzca por su cuenta.

Con esto se puede realizar la implementación de los movimientos y análisis de trayectorias que permitirán hacer diversas funciones como carga y descarga de objetos.

Las trayectorias analizadas permiten visualizar e implementarlas en el prototipo para evitar problemas como colisiones, singularidades y otros problemas mecánicos, así mismo da una idea del comportamiento del sistema con las dimensiones del rediseño.

Se recomienda hacer una sintonización más cuidadosa, cuando se use el robot en posiciones donde pierde estabilidad, ya que en ésta sección del proyecto se usaron resortes para validar las posiciones y al llegar a los ángulos donde estos no actúan, existen oscilaciones que desfavorecen completamente al robot, aspecto erradicado con el diseño de contrapesos correctos.

REFERENCIAS BIBLIOGRÁFICAS

(Balmaceda, 2011) Balmaceda Santamaría, Albert Lester, Metodología De Rediseño De Un RPTD, de tres de grados de libertad En Función De Un Espacio De Trabajo prescrito. (Tesis de Maestría IPN), México, Noviembre 2011.

(Castañeda, 2006) Castillo Castañeda, Eduardo "Diseño, Construcción y control de un robot paralelo de tres grados de libertad", Revista SUPAUAQ(#35, 2006) pp 4-11

(Ceccarelli, 2004) Marco Ceccarelli, Chiara Lanni, "A Multi-bjetiveOptimumDesign Of General 3r Manipulators For Prescribed Workspace Limits", Mechanism And Machine Theory(No.392004), pp119-132

(Gomez, 2008) Gómez González,Sergio,El gran libro de SolidWorks, (1ª edición) Alfaomega, Barcelona España, 2008.

(Marquez, 2002) Márquez Santoyo Paulina Diseño y construcción de una tarjeta de control de movimiento para un robot paralelo 3 GDL. Tesis de licenciatura,Fac. de Ingeniería UAQ,Querétaro, México. Febrero 2002

(Ottaviano, 2002) Ottaviano Erika, Ceccarelli Marco, Optimal Design Of CAPAMAN With Prescribed Workspace, International Journal Robotic, 2002.

(Velázquez 2003) Velazques Camacho,María del Rosario,Estudio y análisis del espacio de trabajo de un manipulador paralelo, de 3 grados de libertad, Tesis de Licenciatura en Matemáticas, UAQ, Fac. Ingeniería, Septiembre del 2003

Referencias de contenido en internet.

Conceptos de Cinematica inversa, recuperado el 13 de agosto del 2012 de

Www.X-Robotics.Com/Cinematica.Htm

Diseño mecánico asistido por computadora, recuperado el 26 de septiembre del 2012 de http://www.solidcam.com/

Entorno de Solid Works, Recuperado el 20 de septiembre del 2012 http://alternativeto.net/software/solidworks/

Germán Andrés Ramos Fuentes Cinemática Inversa De Robots Industriales Recuperado de:Fgarciae.Mdl2.Com/Pluginfile.Php/.../Cinematica%20Robots.Pdf

Mint, the real time automation Language, programming guide for Mint WorkBench, Homing, recuperado el 13 de noviembre del 2012.

ANEXOS

Anexo 1

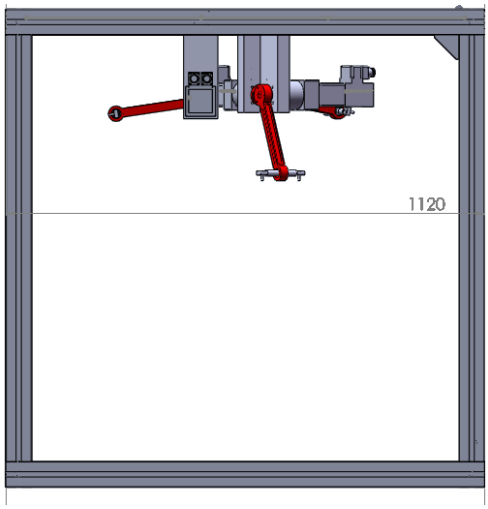


Ilustración 91.Armado del robot en Solid Works

Anexo 2

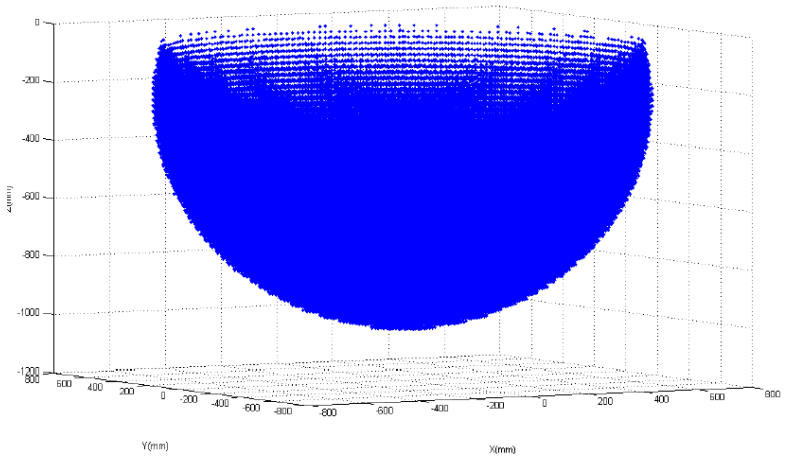


Ilustración 92.Nuevo espacio de trabajo en X,Y, Z

Anexo 3

Fotos del robot

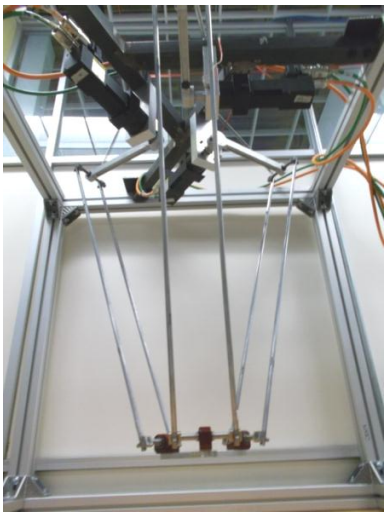


Ilustración 93. Robot

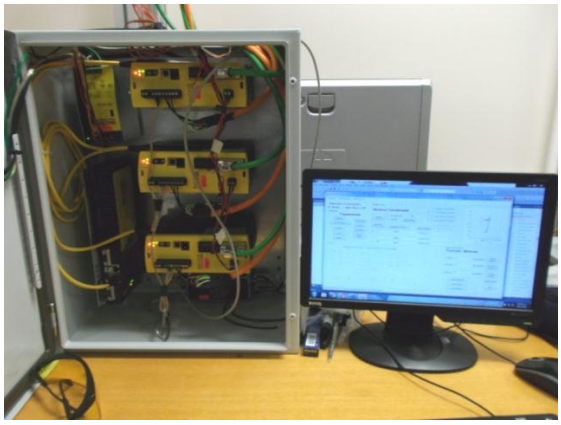
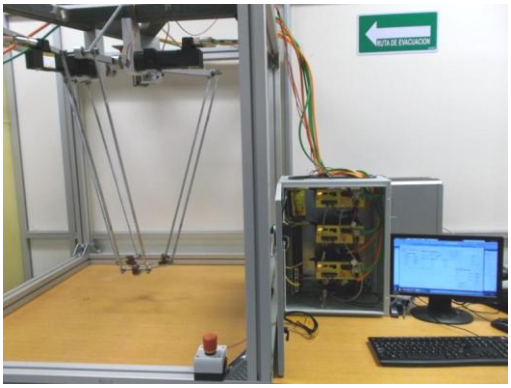


Ilustración 94.Gabinete



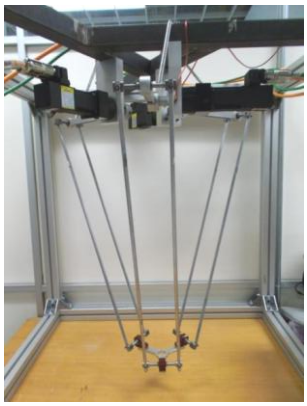


Ilustración 95. Vistas del robot

Anexo 4

Programa en Matlab con Algoritmo de programación para el barrido de puntos para calcular la región alcanzable del robot.

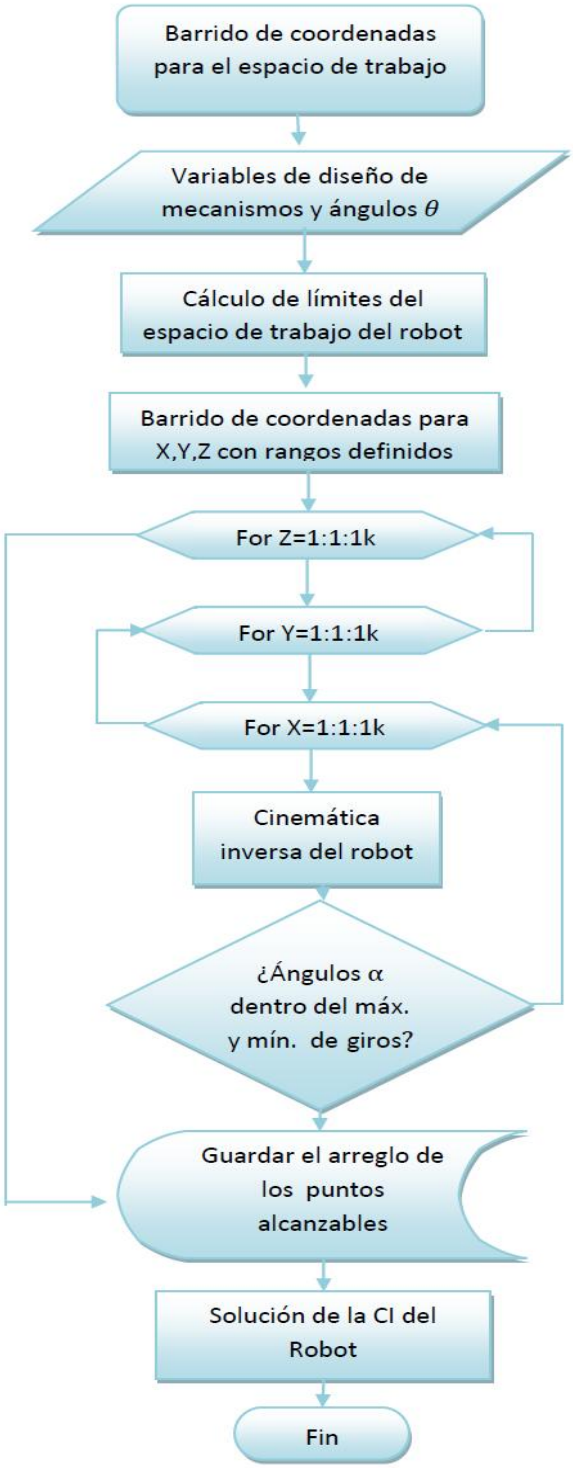


Ilustración 96. Algoritmo para el cálculo de la región alcanzable

```
%% Programa para el cálculo de la región  alcanzable del ROBOT DELTA

Clearall
clc
disp(Cálculo de la region alcanzable con dimensiones del robot')
la=200;lb=400;r=150;rpm=50;
teta(1)=pi/6;teta(2)=5*pi/6;teta(3)=-pi/2;
R=r-rpm;
 L=sqrt(lb^2+la^2-2*la*lb);
beta=asin(R/L);
alfamin=-((pi/2)-beta);
alfamax=pi/2;
ct=cos(teta);
st=sin(teta);
 lb2=lb*lb;
la2=la*la;
r2=r*r;
b2=4*r2;
j=1;
tic;
 %%
for k=-600:20:-150;
for h=-300:20:300;
fori=-400:20:400;
x0=i; y0=h;z0=k;
x=x0+rpm*ct';
y=y0+rpm*st';
        z=[z0;z0;z0];
        q=2*x.*ct'+2*y.*st';
        s=(1/la)*(-x.*x-y.*y-z.*z+lb2-la2-r2);
        b1=4*z.*z;
        b3=q.*q*(1-(r2/la2));
        b4=q.*((-2*r*s/la)-4*r);
b5=q*(r/la-1);
        b6=(-2*r-b5-s);
if (b6==0)
aa=alfamax+10;
else b6 ~= 0;
alfat=(-2*z-sqrt(b1+b2-s.*s+b3+b4))./(-2*r-b5-s);
aa=2*atan(alfat);
end
if ((aa>alfamin) & (aa<alfamax))
xesp(j)=x0;
yesp(j)=y0;
zesp(j)=z0;
          j=j+1;
end
end%para k
```

```
end%para h
end%parai
  %%
toc;
 X=[xesp' yesp' zesp'];
save'trabajo1_p.txt'-ASCIIX
plot(xesp', yesp','.b')
xlabel('X(mm)')
ylabel('y(mm)')
gridon

figure
plot(xesp', zesp','.b')
xlabel('X(mm)')
ylabel('z(mm)')
gridon

figure
plot3(xesp', zesp',zesp','.b')
grid
xlabel('X(mm)')
ylabel('Y(mm)')
zlabel('Z(mm)')
```

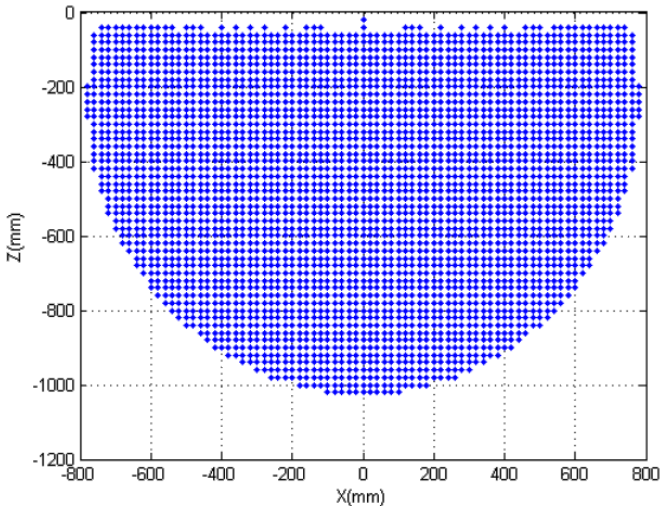


Ilustración 97. Graficas del barrido de puntos x, z

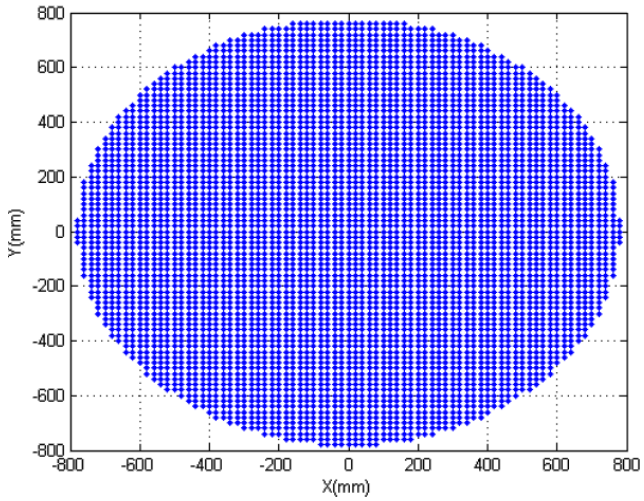


Ilustración 98. Graficas del barrido de puntos x, y

Anexo 5

Código simulación para posicionar en X, Y, Z, dando posición de motores en ángulos y cuentas del encoder.

```
%Posicionamiento  con X, Y y Z
clear; clc;clf;
tic;
load('Datos2.txt');
ac=Datos2(:,1);
ba0=Datos2(:,2);
f0=find(ac<ba0);
[f,c,amin]=find(min(ac(f0)));
f2=find(ac(f0)==amin);
m1=Datos2(f2,:);

ba=m1(:,2);
[f3,c3,bmin]=find(min(ba));
f4=find(ba==bmin);
m2=m1(f4,:);

ra=m2(:,3);
[f5,c5,rmin]=find(min(ra));
f6=find(ra==rmin);
m3=m2(f6,:);

rpma=m3(:,4);
[f7,c7,rpmin]=find(min(rpma));
f8=find(rpma==rpmin);
m4=m3(f8,:);
```

```
ha=m4(:,5);
[f9,c9,hmin]=find(min(ha));
%Dimensiones del parallix
la=240
lb=790
r=100
rpm=90
h=550 %%%%%
%% figura cuadrado
xd= input(' Xd  ') ;
yd=input('Yd ');
 zd=input('Zd ');
R=r-rpm;
teta(1)=pi/6;teta(2)=5*pi/6;teta(3)=-pi/2;
alfamax=pi/2;
 L=sqrt(lb^2+la^2-2*la*lb);
 beta=asin(R/L);
 alfamin=-((pi/2)-beta);
ct=cos(teta);
st=sin(teta);
lb2=lb*lb;
la2=la*la;
r2=r*r;
b2=4*r2;
%punto A
A=[r*cos(teta); r*sin(teta);0 0 0];

%motor1
Am1=A(:,1);
am1x=Am1(1,:);
am1y=Am1(2,:);
am1z=Am1(3,:);

%m2
Am2=A(:,2);
am2x=Am2(1,:);
am2y=Am2(2,:);
am2z=Am2(3,:);
%m3
Am3=A(:,3);
am3x=Am3(1,:);
am3y=Am3(2,:);
am3z=Am3(3,:);
%%
i=1;
        x0=xd(i); y0=yd(i);z0=zd(i);
x=x0+rpm*ct';
```

```
        y=y0+rpm*st';
        z=[z0;z0;z0];
        q=2*x.*ct'+2*y.*st';
        s=(1/la)*(-x.*x-y.*y-z.*z+lb2-la2-r2);
        b1=4*z.*z;
        b3=q.*q*(1-(r2/la2));
        b4=q.*((-2*r*s/la)-4*r);
        b5=q*(r/la-1);
       b6=(-2*r-b5-s);

        b7=b1+b2-s.*s+b3+b4; %es el radical

if ((b6 ~= 0) & (b7(1)>=0) & (b7(2)>=0) & (b7(3)>=0))
             alfat=(-2*z-sqrt(b7))./b6;
aa=2*atan(alfat)
            angulo1=180*aa(1)/pi
            angulo2=180*aa(2)/pi
angulo3=180*aa(3)/pi
            Motor1=(angulo1*100000)/360
            Motor2=(angulo2*100000)/360
            Motor3=(angulo3*100000)/360

else
              disp('verificar esta coordenada ')
x0=xd(i)
             y0=yd(i)
             z0=zd(i)
              aa=alfamax+10

end
if ((aa>alfamin) & (aa< alfamax))
%punto c
   Lx=r+la+cos(aa);
   Lz=-la*sin(aa);
   C=[Lx'.*cos(teta);Lx'.*sin(teta);Lz'];
%puntos B
   B=[xd(i)'+rpm.*cos(teta);yd(i)'+rpm.*sin(teta); zd(i) zd(i) zd(i)];
%%
%motor1
   Cm1=C(:,1);
   cm1x=Cm1(1,:);
   cm1y=Cm1(2,:);
cm1z=Cm1(3,:);
   Bm1=B(:,1);
   bm1x=Bm1(1,:);
bm1y=Bm1(2,:);
bm1z=Bm1(3,:);
%motor2
```

```
    Cm2=C(:,2);
cm2x=Cm2(1,:);
    cm2y=Cm2(2,:);
cm2z=Cm2(3,:);
Bm2=B(:,2);
    bm2x=Bm2(1,:);
bm2y=Bm2(2,:);
    bm2z=Bm2(3,:);
%motor3
    Cm3=C(:,3);
cm3x=Cm3(1,:);
    cm3y=Cm3(2,:);
cm3z=Cm3(3,:);
Bm3=B(:,3);
    bm3x=Bm3(1,:);
    bm3y=Bm3(2,:);
bm3z=Bm3(3,:);
    pause(.1)
    clf
    fill3([am3x,-am1x,-am2x]*2,[-am3y,-am1y,-am2y]*2,[am3z,am1z,am2z]*2,'c','linewidth',2)
%Plataforma fija del parallix
axis equal
    axis([-600,600,-600,600,-900,400]);
    view(-37.5,5);
    hold on
    grid
%eslabones
line([am1x,cm1x,bm1x],[am1y,cm1y,bm1y],[am1z,cm1z,bm1z],'color','b','linewidth',1.5)
line([am2x,cm2x,bm2x],[am2y,cm2y,bm2y],[am2z,cm2z,bm2z],'color','r','linewidth',1.5)
line([am3x,cm3x,bm3x],[am3y,cm3y,bm3y],[am3z,cm3z,bm3z],'color','g','linewidth',1.5)
xlabel('X(nm)')
ylabel('Y(nm)')
zlabel('z(mn)')
%plataforma movil
line([bm1x,bm2x,bm3x,bm1x],[bm1y,bm2y,bm3y,bm1y],[bm1z,bm2z,bm3z,bm1z],'color','k','linew
idth',2)
plot3(xd,yd,zd,'*r')
plot3(am1x,am1y,am1z,'*')
plot3(am2x,am2y,am2z,'*')
plot3(am3x,am3y,am3z,'*')
i=i+1;
pause(.1)
end

toc;
```

Anexo 6

Código en Matlab para la simulación e implementación de la trayectoria Círculo

```
%% Generar los puntos del  circulo
xcentro= input('X centro');
ycentro= input('Y centro');
rcir= input('Radio');
pts= input('No. Puntos');

t=linspace(0,2*pi,pts)

xd=xcentro+rcir*cos(t)
yd=ycentro+rcir*sin(t)

plot(xd,yd),axis('equal'),title('Grafica de un circulo')

grid

%%puntos
zd(1)=-550;   %0
for(pt=2:1:pts)
zd(pt)=-650;%1

end
R=r-rpm;
teta(1)=pi/6;teta(2)=5*pi/6;teta(3)=-pi/2;
alfamax=pi/2;
 L=sqrt(lb^2+la^2-2*la*lb);
beta=asin(R/L);
alfamin=-((pi/2)-beta);
ct=cos(teta);
st=sin(teta);
lb2=lb*lb;
la2=la*la;
r2=r*r;
b2=4*r2;
%punto A
A=[r*cos(teta); r*sin(teta);0 0 0];
%motor1
Am1=A(:,1);
am1x=Am1(1,:);
am1y=Am1(2,:);
am1z=Am1(3,:);
%m2
Am2=A(:,2);
am2x=Am2(1,:);
am2y=Am2(2,:);
```

```
am2z=Am2(3,:);
%m3
Am3=A(:,3);
am3x=Am3(1,:);
am3y=Am3(2,:);
am3z=Am3(3,:);
fori=1:1:pts;
x0=xd(i); y0=yd(i);z0=zd(i);
x=x0+rpm*ct';
        y=y0+rpm*st';
        z=[z0;z0;z0];
        q=2*x.*ct'+2*y.*st';
        s=(1/la)*(-x.*x-y.*y-z.*z+lb2-la2-r2);
        b1=4*z.*z;
        b3=q.*q*(1-(r2/la2));
        b4=q.*((-2*r*s/la)-4*r);
        b5=q*(r/la-1);
       b6=(-2*r-b5-s);
        b7=b1+b2-s.*s+b3+b4; %es el radical

if ((b6 ~= 0) & (b7(1)>=0) & (b7(2)>=0) & (b7(3)>=0))
alfat=(-2*z-sqrt(b7))./b6;
aa=2*atan(alfat)

else
disp('verificar esta coordenada ')
x0=xd(i)
              y0=yd(i)
              z0=zd(i)

aa=alfamax+10

end
if ((aa>alfamin) & (aa<alfamax))
%punto c
   Lx=r+la+cos(aa);
Lz=-la*sin(aa);
   C=[Lx'.*cos(teta);Lx'.*sin(teta);Lz'];

%puntos B
   B=[xd(i)'+rpm.*cos(teta);yd(i)'+rpm.*sin(teta); zd(i) zd(i) zd(i)];
%%            angulo1=180*aa(1)/pi
          angulo2=180*aa(2)/pi
          angulo3=180*aa(3)/pi
          Motor1=(angulo1*100000)/360
          Motor2=(angulo2*100000)/360
          Motor3=(angulo3*100000)/360
%motor1
```

```
    Cm1=C(:,1);
    cm1x=Cm1(1,:);
    cm1y=Cm1(2,:);
cm1z=Cm1(3,:);
Bm1=B(:,1);
    bm1x=Bm1(1,:);
bm1y=Bm1(2,:);
bm1z=Bm1(3,:);
%motor2
    Cm2=C(:,2);
cm2x=Cm2(1,:);
    cm2y=Cm2(2,:);
cm2z=Cm2(3,:);


Bm2=B(:,2);
    bm2x=Bm2(1,:);
bm2y=Bm2(2,:);
bm2z=Bm2(3,:);

%motor3
    Cm3=C(:,3);
cm3x=Cm3(1,:);
    cm3y=Cm3(2,:);
cm3z=Cm3(3,:);

Bm3=B(:,3);
    bm3x=Bm3(1,:);
    bm3y=Bm3(2,:);
bm3z=Bm3(3,:);
pause(.3)
clf
fill3([am3x,-am1x,-am2x]*2,[-am3y,-am1y,-am2y]*2,[am3z,am1z,am2z]*2,'c','linewidth',2)
%Plataformafija del parallix
axisequal
axis([-600,600,-600,600,-900,400]);
view(-37.5,5);
holdon
grid
%eslabones
line([am1x,cm1x,bm1x],[am1y,cm1y,bm1y],[am1z,cm1z,bm1z],'color','b','linewidth',1.5)
line([am2x,cm2x,bm2x],[am2y,cm2y,bm2y],[am2z,cm2z,bm2z],'color','r','linewidth',1.5)
line([am3x,cm3x,bm3x],[am3y,cm3y,bm3y],[am3z,cm3z,bm3z],'color','g','linewidth',1.5)
%

xlabel('X(nm)')
ylabel('Y(nm)')
zlabel('z(mn)')
```

```
%plataforma movil
line([bm1x,bm2x,bm3x,bm1x],[bm1y,bm2y,bm3y,bm1y],[bm1z,bm2z,bm3z,bm1z],'color','k','linew
idth',2)
plot3(xd,yd,zd,'*r')
plot3(am1x,am1y,am1z,'*')
plot3(am2x,am2y,am2z,'*')
plot3(am3x,am3y,am3z,'*')
i=i+1;
pause(.3)
end
end
toc;
```

Anexo 7

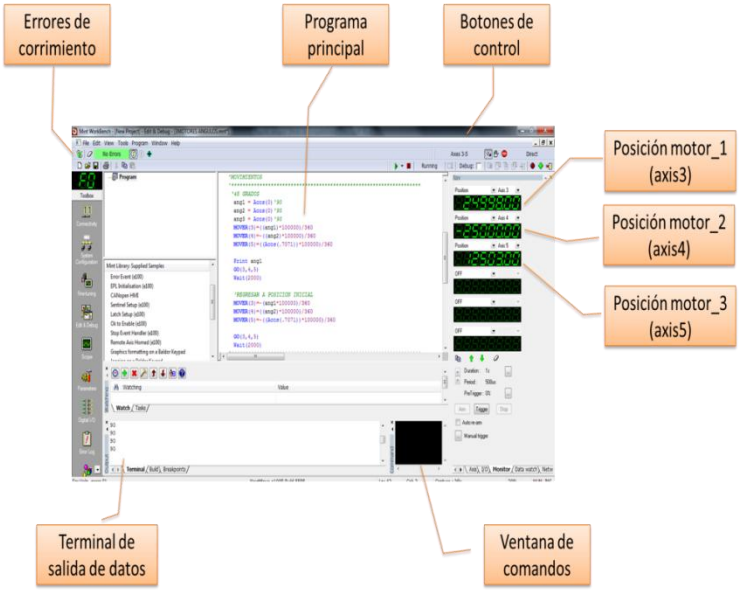


Ilustración 99. Interface de Mint Work Bench

Anexo 8

Programa para la interface de control en Visual Basic

```
PublicClassForm1
Dim Errorcito AsString
Dim a AsNew MintControls5702Lib.MintController
Dim poss, controlito AsInt32
Dim poss2, total, totale AsInt32
Dim poss3 AsInt32
Dim matrizCu(0 To 2) AsDouble
Dim matcono1(0 To 44) AsDouble
Dim matcono2(0 To 44) AsDouble
Dim matcono3(0 To 44) AsDouble
Dim abc1 AsString
Dim abc2 AsString
Dim abc3 AsString
Dim puntosa(0 To 100) AsSingle
Dim puntosb(0 To 100) AsSingle
Dim puntosc(0 To 100) AsSingle
PrivateSub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
    a.Accel(3) = 10000
    a.Decel(3) = 10000
    a.Speed(3) = 10000
    a.Accel(4) = 10000
    a.Decel(4) = 10000
    a.Speed(4) = 10000
    a.Accel(5) = 10000
    a.Decel(5) = 10000
    a.Speed(5) = 10000
    a.DriveEnable(3) = 1
    a.MoveR(3) = CInt(((Movrelat1.Text) * -100000) / 360)
    a.DriveEnable(4) = 1
    a.MoveR(4) = CInt(((Movrelat2.Text) * -100000) / 360)
    a.DriveEnable(5) = 1
    a.MoveR(5) = CInt(((Movrelat3.Text) * -100000) / 360)
    a.DoGo3(3, 4, 5)
EndSub
PrivateSub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
HandlesMyBase.Load
    PictureBox1.Visible = True
Try
If RadioButton1.Checked Then
        a.SetVirtualControllerLink()
ElseIf RadioButton2.Checked Then
a.SetUSBControllerLink(1)
```

```
Else
            MsgBox("Seleccione un controlador")
EndIf
Catch ex AsException
MsgBox("ERROR:  Revisar conexiones y Encendido del sistema")
EndTry
Dim file As System.IO.FileStream
      file = System.IO.File.Create("3.txt")
      file = System.IO.File.Create("4.txt")
      file = System.IO.File.Create("5.txt")
      file = System.IO.File.Create("ABC.txt")
      file = System.IO.File.Create("File1.txt")
      file = System.IO.File.Create("File2.txt")
      file = System.IO.File.Create("File3.txt")
      file = System.IO.File.Create("M1.txt")
      file = System.IO.File.Create("M2.txt")
      file = System.IO.File.Create("M3.txt")
      MsgBox("Generando archivos")
EndSub

PrivateSub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
      a.DriveEnable(3) = 0
      a.DriveEnable(4) = 0
      a.DriveEnable(5) = 0
      MsgBox("Motores deshabilitados")
      Close()
EndSub
PrivateSub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
EndSub
PrivateSub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
Dim pp(0 ToCInt(Text2.Text)) AsSingle
Dim null AsShort = 0
'negativos por el  cambio de la configuracion de robot
pp(0) = CInt(Text2.Text)
      pp(1) = CInt(((TextBox2.Text) * -100000) / 360)
      pp(2) = CInt(((TextBox3.Text) * -100000) / 360)
      pp(3) = CInt(((TextBox4.Text) * -100000) / 360)
      a.SplineTable(3, pp, null, null)
      a.DriveEnable(3) = 1
      a.SplineTime(3) = 500
      a.Spline(3) = 1 Or 2
      a.DoGo1(3)
Dim p2(0 ToCInt(TextBox20.Text)) AsSingle
      p2(0) = CInt(TextBox20.Text)
      p2(1) = CInt(((TextBox17.Text) * -100000) / 360)
      p2(2) = CInt(((TextBox18.Text) * -100000) / 360)
```

```
        p2(3) = CInt(((TextBox19.Text) * -100000) / 360)
        a.SplineTable(4, p2, null, null)
        a.DriveEnable(4) = 1
        a.SplineTime(4) = 500
        a.Spline(4) = 1 Or 2
        a.DoGo1(4)
Dim p3(0 ToCInt(TextBox21.Text)) AsSingle
        p3(0) = CInt(TextBox21.Text)
        p3(1) = CInt(((TextBox22.Text) * -100000) / 360)
        p3(2) = CInt(((TextBox23.Text) * -100000) / 360)
        p3(3) = CInt(((TextBox24.Text) * -100000) / 360)
        a.SplineTable(5, p3, null, null)
        a.DriveEnable(5) = 1
        a.SplineTime(5) = 500
        a.Spline(5) = 1 Or 2
        a.DoGo1(5)
EndSub
PrivateSub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button4.Click
        PictureBox1.Image = System.Drawing.Bitmap.FromFile("caida.jpg")
        a.DriveEnable(3) = 0
        a.DriveEnable(4) = 0
        a.DriveEnable(5) = 0
        MsgBox("Motores deshabilitados")
EndSub
PrivateSub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button5.Click
        PictureBox1.Image = System.Drawing.Bitmap.FromFile("esp.png")
Dim matrizT(0 To 2) AsDouble
Dim a1(0 To 2) AsDouble
Dim Resul AsBoolean
        Resul = cinematica(TextBox8.Text, TextBox9.Text, TextBox10.Text, matrizT)
If (Not (Resul)) Then
MsgBox("Posición inicial")
a1(0) = 0
            a1(1) = 0
            a1(2) = 0
            TextBox11.Text = a1(0)    'Angulo en grados
TextBox12.Text = a1(1)
            TextBox13.Text = a1(2)
Exit Sub
EndIf
        a1(0) = matrizT(0) * 180 / 3.1415926535897931
a1(1) = matrizT(1) * 180 / 3.1415926535897931
        a1(2) = matrizT(2) * 180 / 3.1415926535897931

        TextBox11.Text = a1(0)    'Angulo en grados
TextBox12.Text = a1(1)
```

```
        TextBox13.Text = a1(2)
EndSub
PrivateSub Label13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Label13.Click
EndSub
PrivateSub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button6.Click
        PictureBox1.Image = System.Drawing.Bitmap.FromFile("esp.png")
angulos(CDbl(TextBox8.Text), CDbl(TextBox9.Text), CDbl(TextBox10.Text), matrizCu)

        TextBox14.Text = CInt(matrizCu(0)) * -1 'Cuentas del Motor
TextBox15.Text = CInt(matrizCu(1)) * -1
        TextBox16.Text = CInt(matrizCu(2)) * -1
        a.Accel(3) = 10000
        a.Decel(3) = 10000
        a.Speed(3) = 10000
        a.Accel(4) = 10000
        a.Decel(4) = 10000
        a.Speed(4) = 10000
        a.Accel(5) = 10000
        a.Decel(5) = 10000
        a.Speed(5) = 10000
        a.DriveEnable(3) = 1
        a.DriveEnable(4) = 1
        a.DriveEnable(5) = 1
If (a.ErrData(1) > 0) Then
            MsgBox(a.ErrString, Title:="Error")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")
            MsgBox("¿Limpiar Errores?", MsgBoxStyle.YesNo, Title:="Error en Axis")
If (MsgBoxResult.Yes) Then
                a.DoErrorClear(3, -1)
Else
                MsgBox("No se podra ejecutar movimientos"& a.ErrString, Title:="Error")

EndIf
Else
            a.MoveA(3) = matrizCu(0) * -1
           a.MoveA(4) = matrizCu(1) * -1
            a.MoveA(5) = matrizCu(2) * -1
a.DoGo3(3, 4, 5)
EndIf
EndSub
PrivateSub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button7.Click
' Generar el cuadrado
        PictureBox1.Image = System.Drawing.Bitmap.FromFile("cuadFin.jpg")
Dim pp(0 To 7) AsSingle
Dim null AsShort = 0
```

```
    a.DriveEnable(3) = 1
    a.DriveEnable(4) = 1
    a.DriveEnable(5) = 1
If (a.ErrData(1) > 0) Then
      MsgBox(a.ErrString, Title:="Error")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")
Else
If (MsgBox("¿Ejecutar Cuadrado?", MsgBoxStyle.YesNo, Title:="Cuadrado") = MsgBoxResult.Yes)
Then


pp(0) = CInt(7)
        pp(1) = CInt(12449) 'Posicion inicial z=-600
        pp(2) = CInt(-712.14) * -1
        pp(3) = CInt(-7562.85) * -1
        pp(4) = CInt(4826.84) * -1
        pp(5) = CInt(11029.52) * -1
        pp(6) = CInt(-712.14) * -1
        pp(7) = CInt(12449) ' regresar a p1
        a.SplineTable(3, pp, null, null)
        a.SplineTime(3) = 900
        a.Spline(3) = 1 Or 2
        a.DoGo1(3)
'Motor 2-> axis 4
Dim p2(0 To 7) AsSingle
        p2(0) = CInt(7)
        p2(1) = CInt(12449)
        p2(2) = CInt(11029.52) * -1
        p2(3) = CInt(4826.84) * -1
        p2(4) = CInt(-7562.85) * -1
        p2(5) = CInt(-712.14) * -1
        p2(6) = CInt(11029.52) * -1
        p2(7) = CInt(12449)
        a.SplineTable(4, p2, null, null)
        a.SplineTime(4) = 900
        a.Spline(4) = 1 Or 2
        a.DoGo1(4)
'Motor 3 -> axis 5
Dim p3(0 To 7) AsSingle
        p3(0) = CInt(7)
        p3(1) = CInt(12449)
        p3(2) = CInt(-5274.92) * -1
        p3(3) = CInt(9025.88) * -1
        p3(4) = CInt(9025.88) * -1
        p3(5) = CInt(-5274.92) * -1
        p3(6) = CInt(-5274.9) * -1
        p3(7) = CInt(12449)
        a.SplineTable(5, p3, null, null)
```

```
            a.SplineTime(5) = 900
            a.Spline(5) = 1 Or 2
a.DoGo1(5)


Else
            MsgBox("  No ejecutado  ", Title:="Cuadrado")
EndIf

EndIf
EndSub

PrivateSub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button8.Click
' Generar Movimientos pick and place
      PictureBox1.Image = System.Drawing.Bitmap.FromFile("Pick&Place.jpg")
Dim pp(0 To 26) AsSingle
Dim null AsShort = 0
      a.DriveEnable(3) = 1
      a.DriveEnable(4) = 1
      a.DriveEnable(5) = 1
If (a.ErrData(1) > 0) Then
         MsgBox(a.ErrString, Title:="Error P & P")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")
Else
If (MsgBox("¿Ejecutar Pick and place?", MsgBoxStyle.YesNo, Title:="Pick and place") =
MsgBoxResult.Yes) Then
            pp(0) = CInt(26)
            pp(1) = CInt(12449) 'Posicion inicial
            pp(2) = CInt(19986.755036023) * -1
            pp(3) = CInt(15272.5553362053) * -1
            pp(4) = CInt(-11642.2665658061) * -1
            pp(5) = CInt(19986.755036023) * -1
            pp(6) = CInt(15272.5553362053) * -1
            pp(7) = CInt(-9597.77630483205) * -1
            pp(8) = CInt(19986.755036023) * -1
            pp(9) = CInt(15272.5553362053) * -1
            pp(10) = CInt(-12191.0631227619) * -1
            pp(11) = CInt(19986.755036023) * -1
            pp(12) = CInt(15272.5553362053) * -1
            pp(13) = CInt(-10012.6307529927) * -1
            pp(14) = CInt(19986.755036023) * -1
            pp(15) = CInt(15272.5553362053) * -1
            pp(16) = CInt(-11760.0844952138) * -1
            pp(17) = CInt(19986.755036023) * -1
            pp(18) = CInt(15272.5553362053) * -1
            pp(19) = CInt(-9526.12515433106) * -1
            pp(20) = CInt(19986.755036023) * -1
```

```
        pp(21) = CInt(15272.5553362053) * -1
        pp(22) = CInt(-10216.0017694698) * -1
        pp(23) = CInt(19986.755036023) * -1
        pp(24) = CInt(15272.5553362053) * -1
        pp(25) = CInt(-8030.74276848994) * -1
        pp(26) = CInt(12449) ' fin

        a.SplineTable(3, pp, null, null)
        a.SplineTime(3) = 800
        a.Spline(3) = 1 Or 2
        a.DoGo1(3)
'Motor 2-> axis 4

Dim p2(0 To 26) AsSingle

        p2(0) = CInt(26)
        p2(1) = CInt(12449) 'Posicion inicial
        p2(2) = CInt(2227.1) * -1
        p2(3) = CInt(651.5897) * -1
        p2(4) = CInt(3351.2) * -1
        p2(5) = CInt(2227.1) * -1
        p2(6) = CInt(651.5897) * -1
        p2(7) = CInt(9450.1) * -1
        p2(8) = CInt(2227.1) * -1
        p2(9) = CInt(651.5897) * -1
        p2(10) = CInt(3064.9) * -1
        p2(11) = CInt(2227.1) * -1
        p2(12) = CInt(651.5897) * -1
        p2(13) = CInt(9014.6) * -1
        p2(14) = CInt(2227.1) * -1
        p2(15) = CInt(651.5897) * -1
        p2(16) = CInt(3804) * -1
        p2(17) = CInt(2227.1) * -1
        p2(18) = CInt(651.5897) * -1
        p2(19) = CInt(9532.2) * -1
        p2(20) = CInt(2227.1) * -1
        p2(21) = CInt(651.5897) * -1
        p2(22) = CInt(5480.4) * -1
        p2(23) = CInt(2227.1) * -1
        p2(24) = CInt(651.5897) * -1
        p2(25) = CInt(10896.0) * -1
        p2(26) = CInt(12449) ' fin
        a.SplineTable(4, p2, null, null)
        a.SplineTime(4) = 800
        a.Spline(4) = 1 Or 2
        a.DoGo1(4)

'Motor 3 -> axis 5
```

```
Dim p3(0 To 26) AsSingle

        p3(0) = CInt(26)
        p3(1) = CInt(12449) 'Posicion inicial
        p3(2) = CInt(22457.0) * -1
        p3(3) = CInt(-1946.4) * -1
        p3(4) = CInt(3271.2) * -1
        p3(5) = CInt(22457.0) * -1
        p3(6) = CInt(-1946.4) * -1
        p3(7) = CInt(7456.0) * -1
        p3(8) = CInt(22457.0) * -1
        p3(9) = CInt(-1946.4) * -1
        p3(10) = CInt(-1062.8) * -1
        p3(11) = CInt(22457.0) * -1
        p3(12) = CInt(-1946.4) * -1
        p3(13) = CInt(3326.1) * -1
        p3(14) = CInt(22457.0) * -1
        p3(15) = CInt(-1946.4) * -1
        p3(16) = CInt(-4438.8) * -1
        p3(17) = CInt(22457.0) * -1
        p3(18) = CInt(-1946.4) * -1
        p3(19) = CInt(49.7347) * -1
        p3(20) = CInt(22457.0) * -1
        p3(21) = CInt(-1946.4) * -1
        p3(22) = CInt(-6566.1) * -1
        p3(23) = CInt(22457.0) * -1
        p3(24) = CInt(-1946.4) * -1
        p3(25) = CInt(-2145.6) * -1
        p3(26) = CInt(12449) ' fin

        a.SplineTable(5, p3, null, null)
        a.SplineTime(5) = 800
        a.Spline(5) = 1 Or 2
        a.DoGo1(5)

Else
        MsgBox("Movimiento cancelado ", Title:="Warning")
Exit Sub

EndIf
EndIf

EndSub

PrivateSub Button10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button10.Click
'Rutina para posicion inicial, HOMING
```

```
Dim VV AsShort
Dim VV2 AsShort
Dim VV3 AsShort
      a.DriveEnable(3) = 1
a.DriveEnable(4) = 1
      a.DriveEnable(5) = 1
      MsgBox("Iniciando Calibración,   Espere...")

DoWhile (VV = 0 Or VV2 = 0 Or VV3 = 0)
         VV = a.In(1) '#bank
         VV2 = a.In(2)
         VV3 = a.In(3)
If (VV = 1) Then
'  STOP3
If (a.Idle(3) = True) Then
               a.DoWait(200)
               a.Pos(3) = 0
EndIf
Else
            a.DoWait(200)
            a.Accel(3) = 10000
a.Decel(3) = 10000
a.Speed(3) = 10000
            a.MoveR(3) = 500  'Avanzar hasta topar con el sensor de fin de carrera
a.DoGo1(3)

EndIf

If (VV2 = 1) Then
'  STOP3
If (a.Idle(4) = True) Then
               a.DoWait(200)
               a.Pos(4) = 0
EndIf
Else
            a.DoWait(200)
            a.Accel(4) = 10000
a.Decel(4) = 10000
a.Speed(4) = 10000
            a.MoveR(4) = 500  'Avanzar hasta topar con el sensor de fin de carrera
a.DoGo1(4)

EndIf
If (VV3 = 1) Then
'  STOP 5
If (a.Idle(5) = True) Then
               a.DoWait(200)
               a.Pos(5) = 0
```

```
EndIf
Else
            a.DoWait(200)

            a.Accel(5) = 10000
a.Decel(5) = 10000
a.Speed(5) = 10000
            a.MoveR(5) = 500  'Avanzar hasta topar con el sensor de fin de carrera
a.DoGo1(5)
EndIf

Loop
      MsgBox("Finalizado...")
EndSub

PrivateSub Timer1_Tick_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer1.Tick
      Timer1.Enabled = False
If (RadioButton3.Checked) Then
          TextBox5.Text = total
ElseIf (RadioButton4.Checked) Then
          TextBox5.Text = totale
EndIf

      Errorcito = a.ErrorReadNext(0, -1)

If (a.ErrData(1) > 0) Then
          MsgBox(a.ErrString, Title:="Error en Driver_timer")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error en movimiento_timer")

If (MsgBox("¿Limpiar Errores?...", MsgBoxStyle.YesNo, Title:="Error en Axis_timer") =
MsgBoxResult.Yes) Then
            a.DoErrorClear(3, -1)

Else
            MsgBox("No se podra ejecutar movimientos"& a.ErrString, Title:="Error timer")

EndIf
Else

          ToolStripStatusLabel1.Text = " Estado del sistema: "
ToolStripStatusLabel2.Text = a.ErrString

          posi.Text = a.Pos(3)
posi2.Text = a.Pos(4)
          Posi3.Text = a.Pos(5)
If (a.DriveEnable(3) = True) Then
            Label6.Text = "Habilitado"
```

```
            ToolStripStatusLabel3.Text = "Drivers Habilitados"
Else
            Label6.Text = "Deshabilitado"
            ToolStripStatusLabel3.Text = "Drivers deshabilitados"
EndIf

If (a.DriveEnable(4) = True) Then
Label34.Text = "Habilitado"
Else
            Label34.Text = "Deshabilitado"
EndIf
If (a.DriveEnable(5) = True) Then
Label35.Text = "Habilitado"
Else
            Label35.Text = "Deshabilitado"
EndIf

EndIf

    Timer1.Enabled = True

EndSub

PrivateSub Button13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button13.Click
Dim VV AsShort
Dim VV2 AsShort
Dim VV3 AsShort
    VV = a.In(1)
    VV2 = a.In(2)
    VV3 = a.In(3)
MsgBox("Drive 1 en:  "& VV)
    MsgBox("Drive 2 en:  "& VV2)
MsgBox("Drive 3 en:  "& VV3)

EndSub

PrivateSub Button14_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button14.Click
posi.Text = a.Pos(3)
    posi2.Text = a.Pos(4)
Posi3.Text = a.Pos(5)
EndSub

PrivateSub Button15_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button15.Click

Try
```

```
        a.DoErrorClear(3, -1)
Catch ex AsException
MsgBox(" Error persistente, reinicie los Drivers")
EndTry
EndSub

PrivateSub Leerror_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Leerror.Click
    Label7.Text = a.ErrCode
    Label8.Text = a.ErrString
    Label32.Text = a.ErrData(1)
    Label33.Text = a.ErrLine
EndSub
PrivateSub Button16_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button16.Click

If (a.ErrData(1) > 0) Then
        MsgBox(a.ErrString, Title:="Error")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")

Else
        a.DriveEnable(3) = 1
        a.DriveEnable(4) = 1
        a.DriveEnable(5) = 1

EndIf

EndSub
PrivateSub RadioButton2_Checked(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles RadioButton2.CheckedChanged
If (MsgBox("Asegurese de que esté conectado el dispositivo", MsgBoxStyle.YesNo,
Title:="¿Controlador Conectado?") = MsgBoxResult.Yes) Then

        a.SetUSBControllerLink(1)
Else
        RadioButton1.Checked = True
        RadioButton2.Checked = False

EndIf
EndSub

PrivateSub RadioButton1_Checked(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles RadioButton1.CheckedChanged

    a.SetVirtualControllerLink()
EndSub
PrivateSub ControlToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ControlToolStripMenuItem.Click
```

```
EndSub

PrivateSub EstadoActualToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles EstadoActualToolStripMenuItem.Click
    Label7.Text = a.ErrCode
    Label8.Text = a.ErrString
    Label32.Text = a.ErrData(1)
    Label33.Text = a.ErrLine
EndSub
PrivateSub AcercaDeToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AcercaDeToolStripMenuItem1.Click
Dialog1.Show()
EndSub

PrivateSub DeshabilitarToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles DeshabilitarToolStripMenuItem.Click
    a.DriveEnable(3) = 0
    a.DriveEnable(4) = 0
    a.DriveEnable(5) = 0
    MsgBox("Motores deshabilitados")
EndSub
PrivateSub AjustarPIDToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AjustarPIDToolStripMenuItem.Click
If (MsgBox("Esto solo debe aplicarse en caso de conocer la posición absoluta del motor respecto al
robot", MsgBoxStyle.YesNo, Title:="¿Editar encoder?") = MsgBoxResult.Yes) Then
        GroupBox4.Visible = True
        GroupBox1.Enabled = False
        GroupBox3.Enabled = False
        GroupBox6.Enabled = False
        PictureBox1.Visible = False
Else
        MsgBox("Cancelado")
EndIf
EndSub

PrivateSub Button21_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button21.Click

    a.Pos(3) = PID1.Text
    a.Pos(4) = PID2.Text
    a.Pos(5) = PID3.Text
EndSub
PrivateSub MovimientosSinCIToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MovimientosSinCIToolStripMenuItem.Click
    GroupBox1.Enabled = True
EndSub
```

```
PrivateSub Button22_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button22.Click
'Profile parameters
    a.DriveEnable(PID4.Text) = 1
    a.Speed(PID4.Text) = 10000.0
    a.Accel(PID4.Text) = 10000.0
    a.Decel(PID4.Text) = 10000.0
    a.MoveR(PID4.Text) = PID5.Text
    a.DoGo1(PID4.Text)
EndSub

PrivateSub SalirToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles SalirToolStripMenuItem.Click

    a.DriveEnable(3) = 0
    a.DriveEnable(4) = 0
    a.DriveEnable(5) = 0
    Close()
EndSub
PrivateSub Movrelat1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Movrelat1.TextChanged
EndSub
PrivateSub GráficasToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles GráficasToolStripMenuItem.Click
Form2.Show()
EndSub
PrivateSub Button23_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button23.Click
    GroupBox4.Visible = False
    GroupBox1.Enabled = False
    GroupBox3.Enabled = True
    GroupBox6.Enabled = True
    PictureBox1.Visible = True
EndSub
PrivateSub ConexionesToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ConexionesToolStripMenuItem.Click
Dialog2.Show()
EndSub

PrivateSub Button24_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button24.Click
    TextBox8.Enabled = True
Dim puntosx(0 To (TextBox1.Text)) AsSingle
Dim puntosy(0 To (TextBox1.Text)) AsSingle
Dim puntosz(0 To (TextBox1.Text)) AsSingle
Dim puntos3(0 To (TextBox1.Text)) AsSingle
Dim puntos4(0 To (TextBox1.Text)) AsSingle
Dim puntos5(0 To (TextBox1.Text)) AsSingle
```

```
Dim textito1, textito2, textito3 AsString
If (RadioButton3.Checked) Then
puntos3(0) = CInt(TextBox1.Text) ' total de datos
puntos4(0) = CInt(TextBox1.Text)
        puntos5(0) = CInt(TextBox1.Text)
        textito1 = CStr(puntos3(0))
        textito2 = CStr(puntos4(0))
textito3 = CStr(puntos5(0))
My.Computer.FileSystem.WriteAllText("File1.txt", _
"Datos totales: "& textito1 & vbCrLf, True) 'con retorno de carrro vbCrLf
My.Computer.FileSystem.WriteAllText("File2.txt", _
"Posiciones Totales en Cuentas: "& textito2 & vbCrLf, True)
TextBox8.Focus()
        total = 1
ElseIf (RadioButton4.Checked) Then
My.Computer.FileSystem.WriteAllText("M1.txt", _
      vbCrLf, False)
My.Computer.FileSystem.WriteAllText("M2.txt", _
   vbCrLf, False)
My.Computer.FileSystem.WriteAllText("M3.txt", _
vbCrLf, False)
        TextBox8.Focus()
Else
        MsgBox("Seleccione tipo de Coordenadas")
EndIf
EndSub
PrivateSub Button25_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button25.Click
If RadioButton3.Checked Then
Dim fileReader1, fileReader2 AsString
        fileReader1 = My.Computer.FileSystem.ReadAllText("File1.txt")
        fileReader2 = My.Computer.FileSystem.ReadAllText("File2.txt")
MsgBox(fileReader1, Title:="Coordenadas")
        MsgBox(fileReader2, Title:="Cuentas de los motores")
ElseIf RadioButton4.Checked Then
Dim fileReader1, fileReader2, ar3 AsString
        fileReader1 = My.Computer.FileSystem.ReadAllText("File1.txt")
        fileReader2 = My.Computer.FileSystem.ReadAllText("File3.txt")
        MsgBox(fileReader1, Title:="Coordenadas")
        MsgBox(fileReader2, Title:="Cuentas")
EndIf
EndSub
PrivateSub TextBox8_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox8.KeyPress
If e.KeyChar = ChrW(Keys.Enter) Then
        abc1 = TextBox8.Text
        TextBox8.Enabled = False
        TextBox9.Enabled = True
```

```
        TextBox9.Focus()
EndIf
EndSub
PrivateSub TextBox9_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox9.KeyPress
If e.KeyChar = ChrW(Keys.Enter) Then

        abc2 = TextBox9.Text
        TextBox9.Enabled = False
        TextBox10.Enabled = True
        TextBox10.Focus()
EndIf
EndSub
PrivateSub TextBox10_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox10.KeyPress
If e.KeyChar = ChrW(Keys.Enter) Then
If (TextBox10.Text >= -550) Then
            MsgBox("Coordenada Z  no alcanzable", Title:="Error")
            TextBox10.Enabled = False
            TextBox8.Enabled = True
TextBox8.Focus()
Else
            abc3 = TextBox10.Text
            TextBox10.Enabled = False
            TextBox8.Enabled = True
            TextBox8.Focus()
If (RadioButton3.Checked) Then
Dim puntosx(0 To (TextBox1.Text)) AsSingle
Dim puntosy(0 To (TextBox1.Text)) AsSingle
Dim puntosz(0 To (TextBox1.Text)) AsSingle
Dim puntos3(0 To (TextBox1.Text)) AsSingle
Dim puntos4(0 To (TextBox1.Text)) AsSingle
Dim puntos5(0 To (TextBox1.Text)) AsSingle
Dim textito1, textito2, textito3 AsString
puntosx(total) = abc1
                puntosy(total) = abc2
                puntosz(total) = abc3
                angulos(CDbl(puntosx(total)), CDbl(puntosy(total)), CDbl(puntosz(total)), matrizCu)
                puntos3(total) = CInt(matrizCu(0)) * -1 'Cuentas del Motor
                puntos4(total) = CInt(matrizCu(1)) * -1
                puntos5(total) = CInt(matrizCu(2)) * -1
                textito1 = CStr(puntos3(total))
                textito2 = CStr(puntos4(total))
textito3 = CStr(puntos5(total))
My.Computer.FileSystem.WriteAllText("File1.txt", _
total &".- "&"X= "& abc1 &", Y= "& abc2 &", Z= "& abc3 & vbCrLf, True) 'con retorno de carrro
vbCrLf
My.Computer.FileSystem.WriteAllText("File2.txt", _
```

```
        total &".- "&"M3= "& textito1 &", M4="& textito2 &", M5="& textito3 & vbCrLf, True)
If (total = TextBox1.Text) Then
If (MsgBox("Se ha calculado el modelo inverso para los puntos, ¿Desea ejecutarlos?",
MsgBoxStyle.YesNo, Title:="¿Ejecutar movimientos?") = MsgBoxResult.Yes) Then
                    a.DriveEnable(3) = 1
                    a.DriveEnable(4) = 1
                    a.DriveEnable(5) = 1
If (a.ErrData(1) > 0) Then
                      MsgBox(a.ErrString, Title:="Error")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")
                      MsgBox("¿Limpiar Errores?", MsgBoxStyle.YesNo, Title:="Error en Axis")
If (MsgBoxResult.Yes) Then
                        a.DoErrorClear(3, -1)
Else
                        MsgBox("No se podra ejecutar movimientos"& a.ErrString, Title:="Error")
EndIf
Else
                      puntos3(0) = CInt(TextBox1.Text) ' total de datos
                      puntos4(0) = CInt(TextBox1.Text)
                      puntos5(0) = CInt(TextBox1.Text)
a.SplineTable(3, puntos3, 0, 0)
                      a.SplineTime(3) = 500
                      a.Spline(3) = 1 Or 2
                      a.SplineTable(4, puntos4, 0, 0)
                      a.SplineTime(4) = 500
                      a.Spline(4) = 1 Or 2
                      a.SplineTable(5, puntos5, 0, 0)
                      a.SplineTime(5) = 500
                      a.Spline(5) = 1 Or 2

a.DoGo3(3, 4, 5)
EndIf
Else
                    MsgBox("Movimiento cancelado")
EndIf
                total = 0

EndIf
            total = total + 1
ElseIf (RadioButton4.Checked) Then
            totale = total - 1
Dim puntosx(0 To total) AsSingle
Dim puntosy(0 To total) AsSingle
Dim puntosz(0 To total) AsSingle
Dim textito1, textito2, textito3 AsString
puntosx(totale) = abc1
            puntosy(totale) = abc2
            puntosz(totale) = abc3
```

```
 angulos(CDbl(puntosx(totale)), CDbl(puntosy(totale)), CDbl(puntosz(totale)), matrizCu)
                puntosa(totale) = CInt(matrizCu(0)) * -1 'Cuentas del Motor
                puntosb(totale) = CInt(matrizCu(1)) * -1
                puntosc(totale) = CInt(matrizCu(2)) * -1
                textito1 = CStr(puntosa(totale))
                textito2 = CStr(puntosb(totale))
textito3 = CStr(puntosc(totale))
My.Computer.FileSystem.WriteAllText("File1.txt", _
totale &".- "&"X= "& abc1 &", Y= "& abc2 &", Z= "& abc3 & vbCrLf, True) 'con retorno de carrro
vbCrLf
My.Computer.FileSystem.WriteAllText("File3.txt", _
            totale &".- M1= "& textito1 &", M2= "& textito2 &",M3= "& textito3 & vbCrLf, True)
My.Computer.FileSystem.WriteAllText("M1.txt", _
          textito1 & vbCrLf, True)
My.Computer.FileSystem.WriteAllText("M2.txt", _
          textito2 & vbCrLf, True)
My.Computer.FileSystem.WriteAllText("M3.txt", _
          textito3 & vbCrLf, True)
                total = total + 1
Else
                MsgBox("Seleccione tipo de Coordenadas")
EndIf
EndIf
EndIf
EndSub
PrivateSub RadioButton4_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs)
EndSub
PrivateSub RadioButton3_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton3.CheckedChanged
      TextBox8.Enabled = True
      TextBox9.Enabled = False
      TextBox10.Enabled = False
      Button27.Enabled = False
      Button24.Enabled = True
      Button25.Enabled = True
      TextBox1.Enabled = True
      TextBox1.Focus()
total = 1
EndSub
PrivateSub TextBox1_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox1.KeyPress

If e.KeyChar = ChrW(Keys.Enter) Then
Dim puntosx(0 To (TextBox1.Text)) AsSingle
Dim puntosy(0 To (TextBox1.Text)) AsSingle
Dim puntosz(0 To (TextBox1.Text)) AsSingle
Dim puntos3(0 To (TextBox1.Text)) AsSingle
```

```
Dim puntos4(0 To (TextBox1.Text)) AsSingle
Dim puntos5(0 To (TextBox1.Text)) AsSingle
Dim textito1, textito2, textito3 AsString
If (RadioButton3.Checked) Then
puntos3(0) = CInt(TextBox1.Text) ' total de datos
puntos4(0) = CInt(TextBox1.Text)
            puntos5(0) = CInt(TextBox1.Text)
            textito1 = CStr(puntos3(0))
            textito2 = CStr(puntos4(0))
textito3 = CStr(puntos5(0))
My.Computer.FileSystem.WriteAllText("File1.txt", _
"Datos totales: "& textito1 & vbCrLf, True) 'con retorno de carrro vbCrLf
My.Computer.FileSystem.WriteAllText("File2.txt", _
"Posiciones Totales en Cuentas: "& textito2 & vbCrLf, True)
            TextBox8.Focus()
            total = 1
Else
            MsgBox("Seleccione tipo de Coordenadas")
EndIf
EndIf
EndSub
PrivateSub RadioButton4_CheckedChanged_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton4.CheckedChanged
      total = 2
      Button27.Enabled = True
      TextBox1.Enabled = False
      TextBox8.Focus()
My.Computer.FileSystem.WriteAllText("File3.txt", _
  vbCrLf, True)
My.Computer.FileSystem.WriteAllText("M1.txt", _
        vbCrLf, False)
My.Computer.FileSystem.WriteAllText("M2.txt", _
 vbCrLf, False)
My.Computer.FileSystem.WriteAllText("M3.txt", _
vbCrLf, False)
EndSub
PrivateSub Button27_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button27.Click
      TextBox8.Enabled = False
If (totale <= 1) Then
MsgBox("Debe agregar mas de un conjunto de coordenadas", Title:="Advertencia")
Else
Dim puntosal(0 To totale) AsSingle
Dim puntosbl(0 To totale) AsSingle
Dim puntoscl(0 To totale) AsSingle
Dim conter1 AsInteger = 1
Dim conter2 AsInteger = 1
Dim conter3 AsInteger = 1
```

```
Using MyReader AsNew  _
   Microsoft.VisualBasic.FileIO.TextFieldParser("M1.txt")
            MyReader.TextFieldType = FileIO.FieldType.Delimited
            MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
                currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
puntosal(conter1) = currentField
                    conter1 = conter1 + 1
Next
EndWhile
EndUsing
Using MyReader AsNew  _
   Microsoft.VisualBasic.FileIO.TextFieldParser("M2.txt")
            MyReader.TextFieldType = FileIO.FieldType.Delimited
            MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
                currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                    puntosbl(conter2) = currentField
                    conter2 = conter2 + 1
Next
EndWhile
EndUsing
Using MyReader AsNew  _
   Microsoft.VisualBasic.FileIO.TextFieldParser("M3.txt")
            MyReader.TextFieldType = FileIO.FieldType.Delimited
            MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
                currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                    puntoscl(conter3) = currentField
                    conter3 = conter3 + 1
Next
EndWhile
EndUsing
          a.DriveEnable(3) = 1
          a.DriveEnable(4) = 1
          a.DriveEnable(5) = 1
If (a.ErrData(1) > 0) Then
            MsgBox(a.ErrString, Title:="Error")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")
```

```
            MsgBox("¿Limpiar Errores?", MsgBoxStyle.YesNo, Title:="Error en Axis")
If (MsgBoxResult.Yes) Then
                a.DoErrorClear(3, -1)
Else
                MsgBox("No se podra ejecutar movimientos"& a.ErrString, Title:="Error")
EndIf
Else
            puntosal(0) = CInt(totale) ' total de datos
             puntosbl(0) = CInt(totale)
             puntoscl(0) = CInt(totale)
a.SplineTable(3, puntosal, 0, 0)
             a.SplineTime(3) = 500
             a.Spline(3) = 1 Or 2
             a.SplineTable(4, puntosbl, 0, 0)
             a.SplineTime(4) = 500
             a.Spline(4) = 1 Or 2
             a.SplineTable(5, puntoscl, 0, 0)
             a.SplineTime(5) = 500
             a.Spline(5) = 1 Or 2
             a.DoGo3(3, 4, 5)
EndIf
          total = 2
EndIf
EndSub
PrivateSub Button26_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button26.Click
Dim gg AsInteger = 0
Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox6.Text)
          MyReader.TextFieldType = FileIO.FieldType.Delimited
          MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
             currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                gg = gg + 1
Next
EndWhile
EndUsing
Dim puntosal(0 To gg) AsSingle
Dim puntosbl(0 To gg) AsSingle
Dim puntoscl(0 To gg) AsSingle
Dim conter1 AsInteger = 1
Dim conter2 AsInteger = 1
Dim conter3 AsInteger = 1

Using MyReader AsNew  _
```

```
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox6.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                puntosal(conter1) = currentField
                conter1 = conter1 + 1
Next
EndWhile
EndUsing
Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox7.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                puntosbl(conter2) = currentField
                conter2 = conter2 + 1
Next
EndWhile
EndUsing

Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox25.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                puntoscl(conter3) = currentField
conter3 = conter3 + 1
Next
EndWhile
EndUsing
     a.DriveEnable(3) = 1
      a.DriveEnable(4) = 1
      a.DriveEnable(5) = 1
If (a.ErrData(1) > 0) Then
        MsgBox(a.ErrString, Title:="Error")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")
```

```
        MsgBox("¿Limpiar Errores?", MsgBoxStyle.YesNo, Title:="Error en Axis")
If (MsgBoxResult.Yes) Then
            a.DoErrorClear(3, -1)
Else
            MsgBox("No se podra ejecutar movimientos"& a.ErrString, Title:="Error")
EndIf
Else
        puntosal(0) = CInt(conter1 - 1) ' total de datos
        puntosbl(0) = CInt(conter2 - 1)
        puntoscl(0) = CInt(conter3 - 1)
        a.SplineTable(3, puntosal, 0, 0)
        a.SplineTime(3) = 500
        a.Spline(3) = 1 Or 2
        a.SplineTable(4, puntosbl, 0, 0)
        a.SplineTime(4) = 500
        a.Spline(4) = 1 Or 2
        a.SplineTable(5, puntoscl, 0, 0)
        a.SplineTime(5) = 500
        a.Spline(5) = 1 Or 2
a.DoGo3(3, 4, 5)
EndIf
     total = 2

EndSub
PrivateSub Button28_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button28.Click
Dim gg AsInteger = 0
Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox6.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                gg = gg + 1
Next
EndWhile
EndUsing
Dim puntosx(0 To gg) AsSingle
Dim puntosy(0 To gg) AsSingle
Dim puntosz(0 To gg) AsSingle
Dim conter1 AsInteger = 1
Dim conter2 AsInteger = 1
Dim conter3 AsInteger = 1
Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox6.Text)
```

```
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                puntosx(conter1) = currentField
                conter1 = conter1 + 1
Next
EndWhile
EndUsing

Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox7.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                puntosy(conter2) = currentField
conter2 = conter2 + 1
Next
EndWhile
EndUsing
Using MyReader AsNew  _
Microsoft.VisualBasic.FileIO.TextFieldParser(TextBox25.Text)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")
Dim currentRow AsString()
WhileNot MyReader.EndOfData
            currentRow = MyReader.ReadFields()
Dim currentField AsString
ForEach currentField In currentRow
                puntosz(conter3) = currentField
                conter3 = conter3 + 1
Next
EndWhile
EndUsing
Dim puntos3(0 To gg) AsSingle
Dim puntos4(0 To gg) AsSingle
Dim puntos5(0 To gg) AsSingle
Dim ss AsInteger
Dim textito1, textito2, textito3 AsString
For ss = 1 To gg
angulos(CDbl(puntosx(ss)), CDbl(puntosy(ss)), CDbl(puntosz(ss)), matrizCu)
```

```
puntos3(ss) = CInt(matrizCu(0)) * -1 'Cuentas del Motor
        puntos4(ss) = CInt(matrizCu(1)) * -1
        puntos5(ss) = CInt(matrizCu(2)) * -1
        textito1 = CStr(puntos3(ss))
        textito2 = CStr(puntos4(ss))
textito3 = CStr(puntos5(ss))
My.Computer.FileSystem.WriteAllText("File1.txt", _
ss &".- "&"X= "& puntosx(ss) &", Y= "& puntosy(ss) &", Z= "& puntosz(ss) & vbCrLf, True) 'con
retorno de carro vbCrLf
My.Computer.FileSystem.WriteAllText("ABC.txt", _
ss &".- "&"M3= "& textito1 &", M4="& textito2 &", M5="& textito3 & vbCrLf, True) 'guarda las
cuentas con separador
My.Computer.FileSystem.WriteAllText("3.txt", _
textito1 & vbCrLf, True)  'guarda las cuentas sin formato
My.Computer.FileSystem.WriteAllText("4.txt", _
textito2 & vbCrLf, True)  'guarda las cuentas sin formato
My.Computer.FileSystem.WriteAllText("5.txt", _
textito3 & vbCrLf, True)  'guarda las cuentas sin formato
Next
    a.DriveEnable(3) = 1
     a.DriveEnable(4) = 1
     a.DriveEnable(5) = 1
If (a.ErrData(1) > 0) Then
        MsgBox(a.ErrString, Title:="Error")
MsgBox("Revisar Axis:  "& a.ErrData(1), Title:="Error")
        MsgBox("¿Limpiar Errores?", MsgBoxStyle.YesNo, Title:="Error en Axis")
If (MsgBoxResult.Yes) Then
           a.DoErrorClear(3, -1)
Else
           MsgBox("No se podra ejecutar movimientos"& a.ErrString, Title:="Error")
EndIf
Else
        puntos3(0) = CInt(gg) ' total de datos
        puntos4(0) = CInt(gg)
        puntos5(0) = CInt(gg)
       a.SplineTable(3, puntos3, 0, 0)
        a.SplineTime(3) = 500
         a.Spline(3) = 1 Or 2
        a.SplineTable(4, puntos4, 0, 0)
        a.SplineTime(4) = 500
        a.Spline(4) = 1 Or 2
        a.SplineTable(5, puntos5, 0, 0)
        a.SplineTime(5) = 500
        a.Spline(5) = 1 Or 2
        a.DoGo3(3, 4, 5)
EndIf
    total = 2
EndSub
```

```
PrivateSub CuentasToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CuentasToolStripMenuItem.Click
        TextBox6.Enabled = True
        TextBox7.Enabled = True
        TextBox25.Enabled = True
        TextBox6.Text = "3.txt"
        TextBox7.Text = "4.txt"
        TextBox25.Text = "5.txt"
        Button26.Enabled = True
        Button28.Enabled = False
        Label42.Enabled = True
        Label43.Text = "M_3"
        Label44.Text = "M_4"
        Label45.Text = "M_5"
EndSub

PrivateSub CoordenadasToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CoordenadasToolStripMenuItem1.Click
        Label43.Text = "X"
        Label44.Text = "Y"
        Label45.Text = "Z"
        TextBox6.Enabled = True
        TextBox7.Enabled = True
        TextBox25.Enabled = True
TextBox6.Text = "X.txt"
        TextBox7.Text = "X.txt"
        TextBox25.Text = "Z.txt"
        Button26.Enabled = False
        Button28.Enabled = True
        Label42.Enabled = True
EndSub
PrivateSub TodosLosRegistrosToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TodosLosRegistrosToolStripMenuItem.Click
My.Computer.FileSystem.WriteAllText("M1.txt", _
vbCrLf, False)
My.Computer.FileSystem.WriteAllText("M2.txt", _
 vbCrLf, False)
My.Computer.FileSystem.WriteAllText("M3.txt", _
vbCrLf, False)
My.Computer.FileSystem.WriteAllText("file1.txt", _
        vbCrLf, False)
My.Computer.FileSystem.WriteAllText("file2.txt", _
 vbCrLf, False)
My.Computer.FileSystem.WriteAllText("file3.txt", _
vbCrLf, False)
My.Computer.FileSystem.WriteAllText("ABC.txt", _
vbCrLf, False)
My.Computer.FileSystem.WriteAllText("3.txt", _
```

```
vbCrLf, False)
My.Computer.FileSystem.WriteAllText("4.txt", _
vbCrLf, False)
My.Computer.FileSystem.WriteAllText("5.txt", _
vbCrLf, False)
EndSub
PrivateSub FilesToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles FilesToolStripMenuItem.Click
My.Computer.FileSystem.WriteAllText("file1.txt", _
    vbCrLf, False)
My.Computer.FileSystem.WriteAllText("file2.txt", _
 vbCrLf, False)
My.Computer.FileSystem.WriteAllText("file3.txt", _
vbCrLf, False)
EndSub
PrivateSub DeMotoresToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles DeMotoresToolStripMenuItem.Click
My.Computer.FileSystem.WriteAllText("M1.txt", _
  vbCrLf, False)
My.Computer.FileSystem.WriteAllText("M2.txt", _
 vbCrLf, False)
My.Computer.FileSystem.WriteAllText("M3.txt", _
vbCrLf, False)
EndSub
PrivateSub TextBox10_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TextBox10.TextChanged
EndSub
EndClass
```