Universidad Autónoma Metropolitana
División de Ciencias Básicas e Ingeniería
Unidad Azcapotzalco

Proyecto Terminal:

Diseño, fabricación y simulación numérica de la estructura de un robot PUMA (5 GDL).

Un modelo didáctico.

Asesor:

Ing. Romy Pérez Moreno

Alumnos:

Abreu Díaz Marco Antonio

Camargo Rodríguez Jesús Alejandro

Junio 2008

Agradecimientos.

Por Marco Antonio Abreu Díaz:

En primer lugar a mi madre, gracias por tenerme tanta paciencia y por creer en mi en todo momento y sobre todo por tu amor. Siempre estaré en deuda contigo.

A ti tía por tus consejos, por tu apoyo y cariño.

A Gissel, todo es más fácil contigo, gracias por tu amor, por apoyarme y darme ánimos en la recta final.

Por Jesús Alejandro Camargo Rodríguez:

Gracias a todas las personas que han hecho posible este sueño pero principalmente a mis queridos padres, que me han comprendido y apoyado en todo momento.

Gracias a mis amigos que han estado ahí cuando me han hecho falta porque sin su apoyo todo hubiera sido más difícil.

Gracias a Laura que llena mi vida. Desde que llegaste no me falta nada, eres lo que hace que todo funcione bien.

Gracias a todos los que creen en mi, gracias.

Gracias a Dios.

Tabla de contenido.

I. Resumen. 5

II. Introducción. 6

La robótica en México. 6

Fundamento teórico.... 7

Brazo de robot.... 8

Robot angular o antropomórfico (PUMA).... 10

Configuraciones de la muñeca. 11

Medida de la posición. 11

Potenciómetro. 12

Matriz de transformación homogénea. 12

Modelo cinemático. 15

Resolución al problema cinemático directo mediante matrices de transformación homogéneas. 15

Resolución al problema cinemático inverso por métodos geométricos y desacoplo cinemático.19

Método geométrico. 20

Desacoplo cinemático. 22

III. Manufactura de la estructura de robot. 26

Materiales utilizados. 26

Maquinaria utilizada y accesorios. 26

Instrumentos de medición. 26

Maquinado para realizar las distintas piezas: 26

Rodamientos INA modelo HK1210.... 31

IV. Modelado matemático del robot. 33

Cinemática inversa (CI).... 33

Posición. 33

V. Programación de la interfaz gráfica en MatLab. 35

Descripción de la pantalla del "KINEMATIC SOLVER (KS)". 38

VI. Manual de operación. 40

Partes del sistema 40

Conexión de la interfaz 41

Ejecución del Kinematic Solver con Matlab. 42

Pasos para calibrar. 45

Pasos para calcular la posición de la estructura usando las lecturas de los potenciómetros. 48

Pasos para calcular la posición de la estructura por cinemática directa o inversa. 49

Como calcular los errores porcentuales entre la CD experimental y la CD teórica 50

VII. Experimentación y resultados. 51

Prueba 1. 53

Prueba 2 54

Prueba 3 55

Prueba 4 56

Prueba 5. 57

Prueba 6. 58

VIII. Discusión de resultados. 59

IX. Conclusiones. 61

X. Bibliografía. 63

X. Anexo 1. – Dibujos de definición. 64

X. Anexo 2. – Código de MatLab. 81

I. Resumen.

El proyecto presente es producto de la búsqueda de un modelo de brazo de robot que facilite el entendimiento de los métodos matemáticos Denavit-Hartenberg que son el fundamento para describir las condiciones cinemáticas en un robot.

Las limitaciones más importantes para la construcción de este modelo fueron:

1. El reducido espacio para diseñar las articulaciones, aunado a la dificultad de conectar potenciómetros a las mismas con el objetivo de registrar una señal electrónica que será interpretada en una PC.

2. Un sistema de apriete que garantice una correcta sujeción entre los elementos de la articulación y que permita una medición precisa por parte de los potenciómetros.

La solución para la articulación es la aplicación de un rodamiento de agujas (cuya principal ventaja son sus reducidas dimensiones) insertado en una caja simétrica de acero donde se introduce el perno de la articulación. Por otro lado el sistema de apriete fue diseñado a partir de un cubo de acero que cuenta en una de sus caras con un cilindro de extremo roscado (perno) para la inserción de una rondana y una tuerca que constituyen el apriete para la fijación de las articulaciones del brazo de robot.

Para la operación de la estructura, la principal problemática se encontró al momento de fijar la estructura en la posición de *home*. La posición de *home* es donde se ajusta la referencia para realizar el resto de las mediciones. En esta parte se necesita enfocar toda la atención posible al momento de calibrar los potenciómetros.

II. Introducción.

La automatización constituye un elemento muy importante para el desarrollo y la competitividad en las empresas. El empleo de un robot o un brazo de robot dentro de las líneas de producción propone una disminución significativa en los tiempos de fabricación entre otros beneficios, por tanto la economía y la propia supervivencia de una empresa puede estar sustentada y/o fundamentada en la apuesta hacia dichos elementos.

Debido a esta situación, es necesario abordar estos temas dentro de las universidades.

La robótica en México.

México no es ajeno al avance de los robots industriales, aunque su despliegue ha sido muy disparejo. Algunas industrias los adoptan sin prejuicios, y otras ni siquiera se han acercado a evaluar su contribución a la productividad. Con todo, su número sigue creciendo a buen paso y, según algunos proveedores, en los últimos cinco años se ha duplicado debido al empuje de las empresas internacionales.

El uso y la difusión de los robots industriales inició no tanto como una manera de bajar costos, sino por la necesidad de alcanzar una calidad homogénea y garantizar la repetitibilidad, es decir, que todas las piezas o componentes se hicieran igual y cumplieran con las mismas normas. En México, las aplicaciones más tempranas se relacionan con las cabinas o túneles de pintura donde existían restricciones en materia de medio ambiente y salud ambiental, soldadura por resistencia y, una gran triunfadora, la soldadura de arco eléctrico.

La adopción de robots es bien entendida en las empresas globales, que suelen administrar sus tecnologías de una manera más racional debido al conocimiento y experiencias que tienen en diferentes latitudes. Por eso son las más rápidas en incorporarlos a sus líneas de producción. Las empresas mexicanas más pequeñas y de estructura familiar, en cambio, suelen tener menores conocimientos y, por lo tanto, enfrentan más dudas a la hora de tomar decisiones de inversión o reconversión industrial. Son las que requieren más información y apoyo por parte de los técnicos y proveedores de equipo y soluciones.

“Los industriales tienen que aceptar que incorporar un robot no es un lujo o una excentricidad, algo que deba llamar la atención —añade Campos—. Mientras ellos se tardan en tomar la decisión, en China se utilizan por millares porque los necesitan para participar en industrias demandantes como la automotriz, maquinaria y electrónica. Se han dado cuenta más rápido que no es un asunto de mano de obra cara o barata, sino de calidad en los procesos y los productos terminados” [1]

Los precios han venido bajando de manera consistente, ya que si hace 10 años un modelo mediano podía costar entre 100,000 y 120,000 dólares, ahora está en el rango de los 40,000 y 50,000 dólares, lo que para consumidores de grandes volúmenes puede reducirse a 30,000. Pero si los precios han bajado, los empresarios tampoco deben creer que un robot hará milagros. Hay empresas que no están preparadas para recibir la robótica y deben actualizar sus antiguos procesos e implantar mejores prácticas. Los especialistas señalan que los robots deben instalarse en un medio donde contribuyan a la productividad, y que no se conviertan en un cuello de botella o un elefante blanco.

"Si un industrial quiere mejorar su competitividad, la robótica puede ayudarle siempre y cuando también haga una inversión para actualizar el resto de los procesos. El robot no resuelve los problemas, pero cuando las cosas se hacen con inteligencia, el resultado es fantástico." [1]

Fundamento teórico.

Dentro de las muchas definiciones formales propuestas o establecidas de lo que es un robot industrial, la ISO (International Standards Organization) ha adoptado con ligeras modificaciones la definición de la RIA (Robotics Industrial Association). Según la ISO, un robot industrial es:

Un manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular cargas, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas.

El avance de los distintos aspectos de la robótica está siendo mucho más lento de lo que se preveía. Esto es debido en parte al hecho de que la robótica es un área interdisciplinar, por lo que sus avances están supeditados a avances en disciplinas tales como el diseño mecánico, la electrónica, el control, el desarrollo de software, la visión artificial, el sistema sensorial y los nuevos materiales, entre otros.

El robot ha demostrado su capacidad para realizar tareas simples de carácter repetitivo, pero no ha superado aun la barrera que suponen sus limitaciones de percepción del medio. Al carecer de sentidos como la vista o el tacto, las tareas que tienen una fácil solución robotizada son aquellas en las que el entorno del robot está perfectamente definido, estructurado y permanece invariante durante su tarea. La mayor parte de los robots instalados hoy en día trabajan en estas condiciones, pero ya existen soluciones sensorizadas mas o menos universales para un buen numero de aplicaciones, como por ejemplo, ensamblado, pulido o desbardado (sensores de esfuerzos), inspección o control de calidad (visión artificial).

A pesar de sus limitaciones, el campo de los procesos abordables por un robot en la industria manufacturera, especialmente en el sector del automóvil, hace que hoy en día

resulte raro encontrar una línea de fabricación en serie en la que no haya un robot. Las expectativas de crecimiento de la robótica se centran en la consolidación en otros sectores, como la industria alimenticia y fabricación de productos plásticos.

Es importante resaltar que el impacto negativo que la introducción de la robótica industrial tiene socialmente, debido a la reducción de empleo asociada, tiene dos contraposiciones positivas: la creación de puestos de trabajo relacionados con la robótica y accesorios y la mejora de condiciones laborales. Esto último se debe a que los robots se dedican principalmente a tareas que se encuentran dentro de la regla de las 3D, del inglés Dull-Dirty-Demeaning (aburridas o repetitivas, en ambientes o trabajos sucios y degradantes), o de las 3H, del inglés Hot-Heavy-Hazardous (temperaturas elevadas, piezas pesadas, ambientes peligrosos).

En determinadas aplicaciones no hay alternativa posible, generalmente debido a la dificultad de la tarea, en uno o varios de los aspectos siguientes.

- Complejidad de movimientos.
- Gran diversidad de piezas a tratar.
- Necesidad de comunicación con otras maquinas.
- Requerimiento de procesado de información de sensores externos.

Brazo de robot.

Un robot industrial está constituido por dos sistemas principales de componentes. El primer sistema, el manipulador propiamente dicho, lo constituyen las partes móviles: el brazo y la muñeca. Lo que podría considerarse "mano" no forma parte de del robot industrial, tal y como lo suministra el fabricante, sino que será uno de los accesorios que dependen de la aplicación a la que se va a aplicar el robot. De este modo se pueden encontrar robots dedicados a la manipulación con "manos" o dedos, o robots de pintura, que en lugar de "mano" tienen un pulverizador.

El segundo sistema el de control, desde el que se procesa la información necesaria para gobernar el movimiento del manipulador. La situación más común es un armario controlador por cada brazo manipulador, aunque hay fabricantes de robots que ofrecen un controlador que gobierna dos brazos manipuladores.

La estructura mecánica o brazo de un robot consiste en una cadena formada por eslabones (ejes) consecutivos, unidos entre sí por medio de articulaciones que permiten el movimiento relativo entre ellos. Generalmente, la cadena tiene uno de sus extremos fijos, también llamado base, y el extremo opuesto libre para la fijación del mecanismo con el que el robot realiza su trabajo.

Una parte de las características del robot quedan determinadas por su estructura, tales como

su configuración, espacio en planta y volumen de trabajo o alcance de robot. Otras características, como por ejemplo, la velocidad y la capacidad de carga, dependen de los sistemas de accionamiento de sus articulaciones.

Por lo tanto, la mayor parte de las características del robot residen en su brazo, que consta de los siguientes elementos:

- Carcasa o chasis, generalmente formada por elementos de acero o aluminio.
- Sistema de accionamiento de los ejes: actuadores, transmisores, sensores de posición y velocidad.
- Cableado, conectores, fines de carrera y otros elementos, como topes mecánicos, compensadores.

Aunque el brazo de robot es flexible en cierto grado, la mayor parte de los robots industriales son tratados como estructuras rígidas, asumiendo que las desviaciones debidas a la flexión de la estructura del brazo no tienen un efecto significativo en su posicionamiento. En general, los robots flexibles se encuentran únicamente en aplicaciones especiales.

Configuraciones típicas de las articulaciones.

Las articulaciones son los elementos de unión entre los ejes del robot y es en ellas donde se origina el movimiento del mismo. El movimiento de cada articulación puede ser de desplazamiento, de giro o de una combinación de los dos tipos de movimiento. Se distinguen seis tipos de articulaciones: prismáticas, de rotación, cilíndrica, esférica o rotula, planar y de tornillo (Fig. 1).

Figura 1. Tipos de articulaciones

Los tipos de rotación y prismática (o lineal) son los que se utilizan mayoritariamente en los robots industriales. Las articulaciones prismáticas ofrecen un cálculo sencillo para su posicionamiento, alta precisión y gran robustez, y las de rotación son más fáciles de construir y poseen envolventes de trabajo mayores con un menor espacio de planta.

A cada movimiento independiente que es capaz de realizar una articulación se le denomina grado de libertad (GDL). Puesto que en el caso de las articulaciones de rotación y prismáticas el GDL es uno, en los robots industriales el numero de GDL del robot suele

coincidir con el de la suma de sus articulaciones. Estrictamente, el GDL de manipulador es el número de movimientos independientes que se puede realizar. Considerando en un espacio 3D, el máximo GDL es seis, tres desplazamientos y tres giros, de ahí que la mayor parte de los robots industriales tenga seis articulaciones.

Los ejes se dividen comúnmente en dos grupos:

- Ejes principales (1, 2 y 3), mayoritariamente responsables de la posición del objeto.
- Ejes de la muñeca (del 4 en adelante), como los responsables de la orientación.

El empleo de combinaciones de los diferentes tipos de articulaciones en los primeros tres ejes del robot da lugar a lo que se denomina configuración del robot. Se nombra las configuraciones encadenando las iníciales de sus articulaciones de la base a la muñeca, por ejemplo, RPR (rotación, prismática, rotación).

El tipo de configuración determina, entre otras cosas, el campo de trabajo del robot, es decir, el volumen de espacio en el que el robot puede posicionar su muñeca. El campo de trabajo se obtiene de trazar las envolventes de las posiciones alcanzadas por la muñeca del robot como combinación de los movimientos en las articulaciones de sus ejes principales.

Robot angular o antropomórfico (PUMA)

Está formado por tres ejes rotacionales, como se muestra en la Fig. 2, con el primer eje perpendicular al suelo y los otros dos perpendiculares a este y paralelos entre sí. Los robots con configuración angular presentan una gran maniobrabilidad y accesibilidad a zonas con obstáculos, ocupan muy poco espacio en relación a su alcance, son robots muy rápidos, que permiten trayectorias muy complejas. Estas características son la que hacen que la mayor parte de los robots industriales presenten esta configuración de sus tres ejes principales.

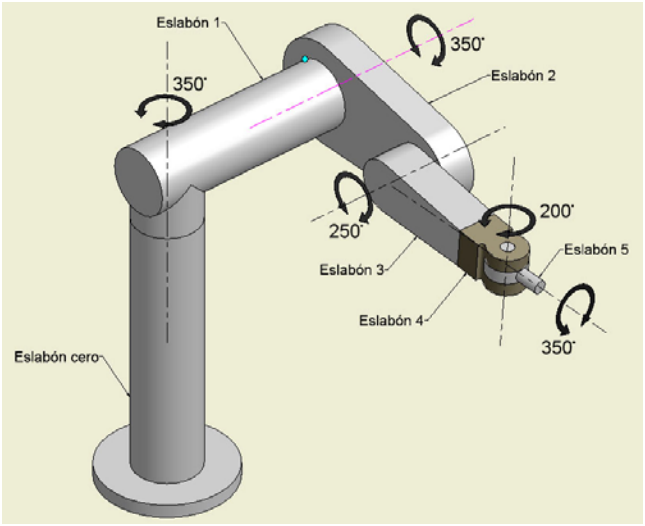


Figura 2. Robot Puma

En la clasificación del parque de robots según el tipo de estructura se observa que el robot angular es el más extendido, con gran diferencia frente al resto.

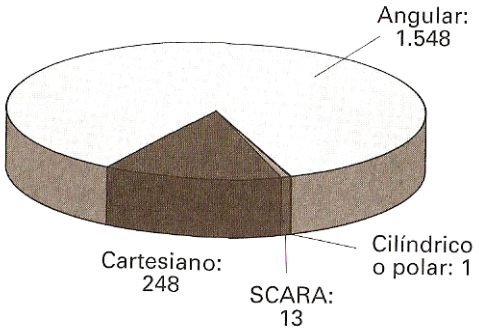


Figura 3. Distribución de robots en España en 1998 por configuración. Fuente AER.[2]

Configuraciones de la muñeca.

La muñeca está formada por las articulaciones del robot, que tienen como misión orientar el objeto en el espacio. Uno de los extremos de la muñeca va unido al brazo, y en el extremo opuesto se fija la herramienta de trabajo del robot. La muñeca consta de hasta tres ejes, dotados con articulaciones de rotación. Una configuración básica es la formada por articulaciones de rotación que producen giros sobre tres ejes perpendiculares que se cortan en un punto. En algunos modelos de robot, los fabricantes ofrecen el mismo brazo con distintas opciones de muñeca, que se seleccionara de acuerdo con la aplicación a la que se quiere destinar el robot.

Medida de la posición.

Para controlar el movimiento de cada uno de los ejes de un robot es necesario conocer en todo instante su posición. Por tanto es necesario incorporar al sistema de accionamiento de cada eje un dispositivo que nos dé una medida de la posición de dicho eje. La precisión en el posicionamiento no solo dependerá de la resolución del sensor de posición, sino también de efectos no considerados como flexión de la estructura.

Los transductores de posición son captadores capaces de medir la posición absoluta de un objeto, o la distancia recorrida desde un punto de referencia, generando a la salida una señal eléctrica proporcional a la magnitud que se desea medir. Los transductores pueden clasificarse atendiendo a dos criterios:

- En función del tipo de señal de salida que originen, se clasificaran en analógicos y digitales.
- En función del tipo de movimiento, en angulares y lineales.

Potenciómetro.

Consiste en una resistencia sobre la que se desliza un contacto móvil. El contacto móvil divide la caída de tensión en la resistencia en dos. El valor de la tensión en el contacto móvil es proporcional a su posición. Entre las ventajas que presenta tenemos que es muy barato y de tamaño muy reducido, aunque las desventajas pueden ser la linealidad y las incertidumbres mecánicas que se pueden encontrar.

Matriz de transformación homogénea.

Una de las representaciones más conocidas para una posición que incluye la orientación son las matrices homogéneas. La mayor ventaja de esta aplicación es que al mismo tiempo un operador permite convertir las coordenadas de un punto dado en el sistema de referencia origina en el que esta expresada la matriz al nuevo sistema de referencia, sin más que realizar el producto de la matriz por el vector de posición del punto.

Para un robot, desde el punto de vista interno, una posición, se entiende como la relación de las posiciones de cada una de sus articulaciones.

Finalmente, el movimiento del extremo del robot será el resultado de la coordinación de los movimientos de cada una de las articulaciones, que a su vez se mueven debido al par/fuerza que imprimen sus respectivos actuadores.

Es el método que permite una representación conjunta de la posición y la orientación de un sólido en el espacio. Es una matriz de dimensión 4X4 que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro.

La representación mediante coordenadas homogéneas de la localización de sólidos en un espacio n-dimensional se realiza a través de un espacio (n+1) dimensional. Es decir, un espacio n-dimensional se encuentra representado en coordenadas homogéneas por (n+1) dimensiones, de tal forma que un vector $p=(x,\ y,\ z)$ vendrá representado por $p=(wx,\ wy,\ wz,\ w)$, donde w tiene un valor arbitrario y representa un factor de escala. De forma general, un vector $p=ai+bj+ck$, donde i, j y k son los vectores unitarios de los ejes OX, OY y OZ del sistema de referencia OXYZ, se representa en coordenadas homogéneas mediante el vector columna:

$$p=\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}=\begin{bmatrix} aw \\ bw \\ cw \\ w \end{bmatrix}=\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \qquad (1)$$

Por ejemplo, el vector $2i+3j+4k$ se puede representar en coordenadas homogéneas como

$[2,3,4,1]^T$ o como $[4,6,8,2]^T$ o también $[6,9,12,3]^T$, etc. A partir de la definición de las coordenadas homogéneas surge inmediatamente el concepto de matriz de transformación homogénea.

$$T = \begin{bmatrix} R_{3X3} & p_{3X1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} Rotación & Translación \\ Perspectiva & Escalado \end{bmatrix} \tag{2}$$

Se puede considerar que una matriz homogénea se haya compuesta por cuatro submatrices de distinto tamaño: una submatriz R_{3X3} que corresponde a una matriz de rotación; una submatriz p_{3X1} que corresponde al vector de translación; una submatriz f_{3X1} que representa una transformación de perspectiva, y una submatriz w_{1X1} que representa un escalado global. En robótica generalmente solo interesara conocer el valor R_{3X3} y de p_{3X1}, considerándose las componentes de f_{1x3} nulas y la de w_{1X1} la unidad. Al tratarse de una matriz 4x4, los vectores sobre los que se aplique deberán contar con 4 dimensiones, que serán las coordenadas homogéneas del vector tridimensional de que se trate.

De esta manera la matriz homogénea T resultará ser de la siguiente forma:

$$T = \begin{bmatrix} R_{3X3} & p_{3X1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Rotación & Translación \\ 0 & 1 \end{bmatrix} \tag{3}$$

Que representa la orientación y posición de un sistema O'UVW rotado y trasladado con respecto al sistema de referencia OXYZ. Esta matriz sirve para conocer las coordenadas (r_x, r_y, r_z) del vector r en el sistema OXYZ a partir de sus coordenadas (r_u, r_v, r_w) en el sistema O'XYZ:

$$\begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = T \begin{bmatrix} r_u \\ r_v \\ r_w \\ 1 \end{bmatrix} \tag{4}$$

También se puede utilizar para expresar la rotación y translación de un vector respecto de un sistema de referencia fijo OXYZ, de tal manera que un vector r_{xyz} rotado según R_{3x3} y trasladado según P_{3x1} se convierte en el vector r'_{xyz} dado por:

$$\begin{bmatrix} r'_x \\ r'_y \\ r'_z \\ 1 \end{bmatrix} = T \begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} \tag{5}$$

En resumen una matriz homogénea se puede aplicar para:

1. Representar la posición y orientación de un sistema girado y trasladado O'UVW con respecto a un sistema de referencia fijo OXYZ, que es lo mismo que representar una rotación y translación realizada sobre un sistema de referencia.
2. Transformar un vector expresado en coordenadas con respecto a un sistema O'UVW, a su expresión en coordenadas del sistema de referencia OXYZ.
3. Rotar y trasladar un vector con respecto a un sistema de referencia fijo OXYZ.

Se hace notar que se utilizan coordenadas homogéneas con factor de escalado igual a la unidad, y que por tanto los vectores que intervienen en las transformaciones han de poseer cuatro componentes. En otras palabras, se elige el factor de escalado w=1.

La utilidad de las matrices homogéneas cobra aun más importancia cuando se componen las matrices homogéneas para describir diversos giros y translaciones consecutivos sobre un sistema de referencia determinado. De esta forma, una transformación compleja podrá descomponerse en la aplicación consecutiva de transformaciones simples (giros básicos y traslaciones).

Por ejemplo, una matriz que representa un giro de un ángulo α sobre el eje OX, seguido de un giro de ángulo ϕ sobre el eje OY y de un giro de un ángulo θ sobre el eje OZ, puede obtenerse por la composición de las matrices básicas de rotación:

$$T = T(z,\theta)T(y,\phi)T(x,\alpha) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C\phi & 0 & S\phi & 0 \\ 0 & 1 & 0 & 0 \\ -S\phi & 0 & C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha & -S\alpha & 0 \\ 0 & S\alpha & C\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\phi C\theta & -S\theta C\alpha + C\theta S\phi S\alpha & S\theta S\alpha + C\theta S\phi C\alpha & 0 \\ S\theta C\phi & C\theta C\alpha + S\theta S\phi S\alpha & -C\theta S\alpha + S\theta S\phi C\alpha & 0 \\ -S\phi & C\phi S\alpha & C\alpha C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Las principales ventajas de este método residen en su capacidad de representación conjunta de posición y orientación y en la comodidad con la que se puede realizar la composición de transformaciones. Para ello bastará únicamente multiplicar, en el orden adecuado, las matrices de transformación correspondientes. Es posible además, la aplicación de una transformación sobre un vector referido a un sistema fijo únicamente multiplicando la matriz de transformación correspondiente por el vector.

Como principal inconveniente presenta su alto nivel de redundancia (necesita definir doce componentes para los 6 grados de libertad). Esto dificulta su implementación en computadora.

Modelo cinemático.

Puesto que la orientación del extremo del robot es uno de los factores a controlar, las posiciones de robot se especifican en forma de sistema de referencia, con un punto origen, expresado en coordenadas cartesianas, y una terna de vectores que definen la orientación deseada para el extremo del robot en ese punto. De manera general, para especificar la posición y orientación de un objeto se liga a él un sistema de referencia.

La cinemática del brazo de un robot trata del estudio analítico de su movimiento con respecto de un sistema de referencia fijo como una función del tiempo sin tener en cuenta las fuerzas que originan el movimiento. Clásicamente se considera que hallar el modelo cinemático del robot es resolver el problema de cinemático directo e inverso.

El problema cinemático directo trata de obtener la posición y orientación del extremo final del robot, a partir de las coordenadas articulares y de parámetros geométricos del brazo. Este problema siempre tiene solución, y además es única.

El problema cinemático inverso trata de obtener las coordenadas articulares que se corresponden con una orientación y una posición dados el extremo móvil del robot. Este problema puede no tener solución (por ejemplo, posiciones no alcanzables por el robot), o en caso de tenerla, puede que esta no sea única.

Denavit y Hartenberg propusieron un método sistemático para describir y representar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo [3]. Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea 4x4 que relacione la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base.

Resolución al problema cinemático directo mediante matrices de transformación homogéneas.

La resolución del problema cinematico directo consiste en encontrar las relaciones que permiten conocer la localización espacial del extremo del robot a partir de los valores de sus coordenadas articulares.

Así, si se han escogido coordenadas cartesianas y ángulos de Euler para representar la posición y orientación del extremo de un robot de seis grados de libertad, la solución del problema cinemático directo vendrá dada por las relaciones:

$$\begin{aligned} x &= f_x(q_1,\ q_2,\ q_3,\ q_4,\ q_5,\ q_6) \\ y &= f_y(q_1,\ q_2,\ q_3,\ q_4,\ q_5,\ q_6) \\ z &= f_z(q_1,\ q_2,\ q_3,\ q_4,\ q_5,\ q_6) \\ \alpha &= f_\alpha(q_1,\ q_2,\ q_3,\ q_4,\ q_5,\ q_6) \\ \beta &= f_\beta(q_1,\ q_2,\ q_3,\ q_4,\ q_5,\ q_6) \\ \gamma &= f_\gamma(q_1,\ q_2,\ q_3,\ q_4,\ q_5,\ q_6) \end{aligned} \tag{7}$$

En general, un robot de de n grados de libertad está formado por n eslabones unidos por n articulaciones, de forma que cada par articulación-eslabón constituye un grado de libertad. A cada eslabón se le puede asociar un sistema de referencia solidario a él y, utilizando las transformaciones homogéneas, es posible representar las rotaciones y translaciones relativas entre los dos distintos eslabones que componen el robot. Normalmente, la matriz de transformación homogénea que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot se suele denominar matriz ${}^{i-1}A_i$. Así pues, ${}^{0}A_1$ describe la posición y orientación del sistema de referencia solidario al primer eslabón con respecto al sistema de referencia solidario a la base, ${}^{1}A_2$, describe la posición y orientación del segundo eslabón respecto del primero, etc. Del mismo modo, denominando ${}^{0}A_k$ a las matrices resultantes del producto de las matrices ${}^{i-1}A_i$ con i desde 1 hasta k, se puede representar de forma total o parcial la cadena cinemática que forma el robot. Así por ejemplo, la posición y orientación del sistema solidario con el segundo eslabón del robot con respecto al sistema de coordenadas de la base se puede expresar mediante la matriz ${}^{0}A_2$:

$${}^{0}A_2 = {}^{0}A_1 {}^{1}A_2 \tag{8}$$

De manera análoga, la matriz ${}^{0}A_3$ representa la localización del sistema del tercer eslabon:

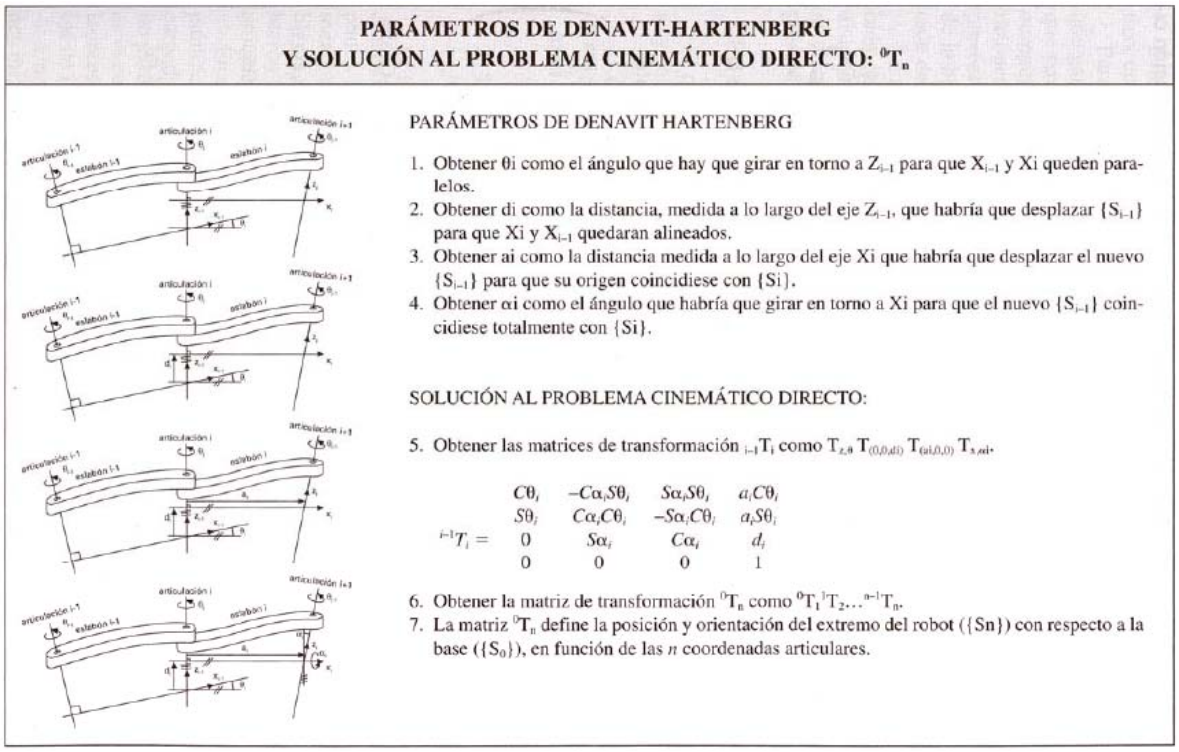
$${}^{0}A_3 = {}^{0}A_1 {}^{1}A_2 {}^{2}A_3 \tag{9}$$

Cuando se consideran todos los grados de libertad, a la matriz ${}^{0}A_n$ se le suele denominar *T*. Así, dado un robot de seis grados de libertad, se tiene que la posición y orientación del eslabon final vendrá dada por la matriz *T*:

$$T = {}^{0}A_6 = {}^{0}A_1 {}^{1}A_2 {}^{2}A_3 {}^{3}A_4 {}^{4}A_5 {}^{5}A_6 \tag{10}$$

Aunque para describir la relación que existe entre dos elementos contiguos se puede hacer uso de cualquier sistema de referencia ligado a cada elemento, la forma habitual que se suele utilizar en robótica es la representación de Denavit-Hartenberg (D-H). Denavit y Hartenberg propusieron en 1955 un método matricial que permite establecer de manera sistemática un sistema de coordenadas $\{S_i\}$ ligado a cada eslabón i de una cadena

articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de la cadena completa.



PARÁMETROS DE DENAVIT-HARTENBERG
Y SOLUCIÓN AL PROBLEMA CINEMÁTICO DIRECTO: ${}^0T_n$

PARÁMETROS DE DENAVIT HARTENBERG

1. Obtener θi como el ángulo que hay que girar en torno a Z_{i-1} para que X_{i-1} y Xi queden paralelos.
2. Obtener di como la distancia, medida a lo largo del eje Z_{i-1}, que habría que desplazar $\{S_{i-1}\}$ para que Xi y X_{i-1} quedaran alineados.
3. Obtener ai como la distancia medida a lo largo del eje Xi que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con {Si}.
4. Obtener αi como el ángulo que habría que girar en torno a Xi para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con {Si}.

SOLUCIÓN AL PROBLEMA CINEMÁTICO DIRECTO:

5. Obtener las matrices de transformación ${}_{i-1}T_i$ como $T_{z,\theta}\, T_{(0,0,di)}\, T_{(ai,0,0)}\, T_{x,\alpha i}$.

$$ {}^{i-1}T_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

6. Obtener la matriz de transformación ${}^0T_n$ como ${}^0T_1\,{}^1T_2 \ldots {}^{n-1}T_n$.
7. La matriz ${}^0T_n$ define la posición y orientación del extremo del robot ({Sn}) con respecto a la base ({S_0}), en función de las *n* coordenadas articulares.

Figura 4. Parámetros Denavit-Hartenberg. [3]

Según la representación de D-H, escogiendo adecuadamente los sistemas de coordenadas asociados a cada eslabón, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las cadenas geométricas del eslabón.

Estas transformaciones básicas consisten en una sucesión de rotaciones y translaciones que permiten relacionar el sistema de referencia del elemento i con el sistema del elemento i-1. Las transformaciones en cuestión son las siguientes (es importante recordar que el paso del sistema $\{S_{i-1}\}$ al $\{S_i\}$ mediante estas 4 transformaciones está garantizado solo si los sistemas $\{S_{i-1}\}$ y $\{S_i\}$ han sido definidos de acuerdo a las normas determinadas que se expondrán posteriormente):

1. Rotación alrededor del eje z_{i-1} un ángulo θ_i.
2. Translación a lo largo de z_{i-1} una distancia d_i; vector d_i (0, 0, d_i).
3. Translación a lo largo de x_i una distancia a_i; vector a_i (0, 0, a_i).
4. Rotación alrededor del eje x_{i-1} un ángulo α_i.

Dado que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene que:

$$ {}^{i-1}A_i = T(z,\theta_i)\,T(0,0,d_i)\,T(a_i,0,0)\,T(x,\alpha_i) \tag{11} $$

Y realizando el producto entre matrices:

$$ {}^{i-1}A_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12} $$

Donde θ_i, a_i, d_i, α_i son los parámetros D-H del eslabón i. De este modo, basta con identificar los parámetros θ_i, a_i, d_i, α_i para obtener las matrices A y relacionar así todos y cada uno los eslabones del robot.

Como se ha indicado, para que la matriz $^{i-1}A_i$, definida en (12) relacione los sistemas $\{S_i\}$ y $\{S_{i-1}\}$, es necesario que los sistemas se hayan escogido de acuerdo a las normas definidas a continuación:

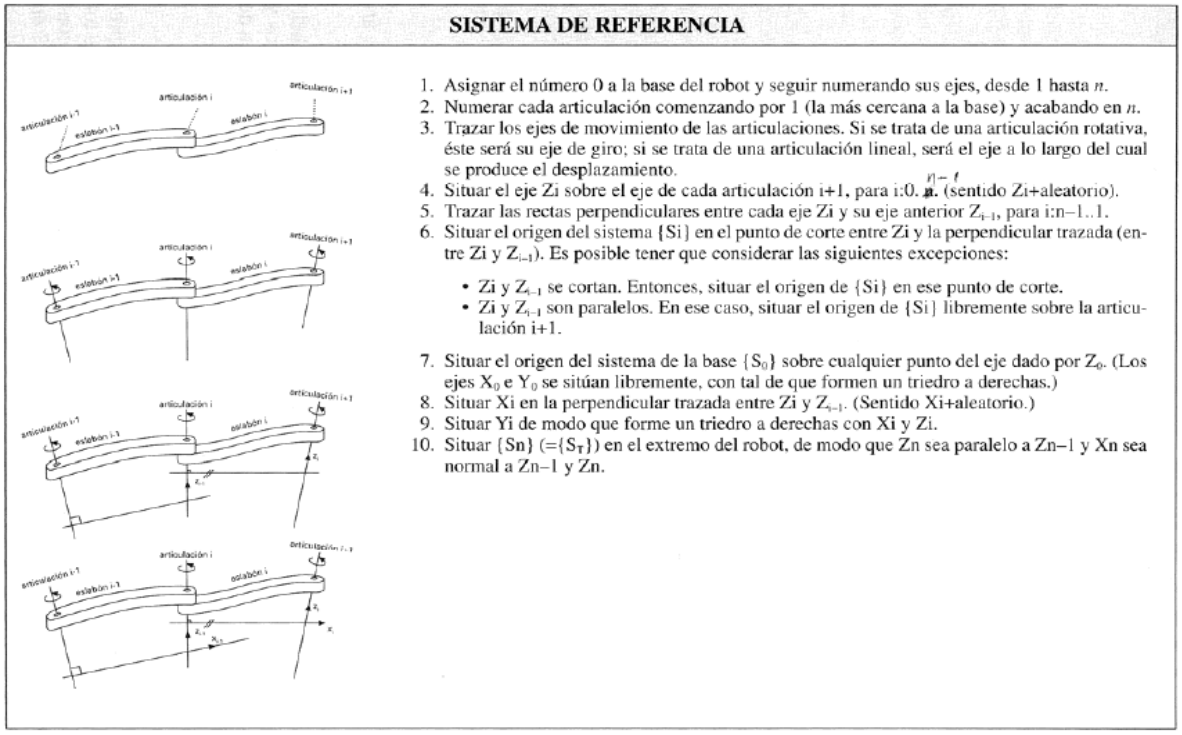


Figura 5. Sistema de referencia. [3]

De modo que basta con identificar los parámetros DH y resolver (12) para cada articulación y finalmente realizar el producto de todas las transformaciones para obtener la matriz de transformación que expresará la orientación (submatriz 3x3 de rotación) y posición (submatriz 3x1 de translación) del extremo del robot en función de sus coordenadas articulares, con lo que quedará resuelto el problema cinemático directo.

Resolución al problema cinemático inverso por métodos geométricos y desacoplo cinemático.

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $q = \left[q_1, \quad q_2, \quad ..., \quad q_n \right]^T$ para que su extremo se posicione y oriente según una determinada localización espacial.

Así como es posible abordar el problema cinemático directo de una manera sistemática a partir de la utilización de matrices, e independientemente de la configuración del robot, no ocurre lo mismo con el problema cinemático inverso, siendo el procedimiento de obtención de las ecuaciones fuertemente dependiente de la configuración del robot.

Se han desarrollado algunos procedimientos genéricos susceptibles de ser programados, de modo que una computadora pueda, a partir del conocimiento de la cinemática del robot (con sus parámetros de D-H, por ejemplo) obtener el conjunto de valores articulares que posicionan y orientan su extremo. El inconveniente de estos procedimientos es que se trata de métodos numéricos iterativos cuya velocidad de convergencia incluso su convergencia en sí no está siempre garantizada.

A la hora de resolver el problema cinemático inverso es mucho más adecuado encontrar una solución cerrada. Esto es, encontrar una relación matemática explicita de la forma:

$$\begin{aligned} q_k &= f_k\left(x, y, z, \alpha, \beta, \gamma\right) \\ k &= 1...n \end{aligned} \tag{13}$$

Este tipo de solución presenta, entre otras, las siguientes ventajas:

1. En muchas aplicaciones, el problema cinemático inverso ha de resolverse en tiempo real (por ejemplo, en el seguimiento de una trayectoria determinada). Una solución de tipo iterativo no garantiza tener la solución en el momento adecuado.
2. Al contrario de lo que ocurriría en el problema cinemático directo, con cierta frecuencia la solución del problema cinemático inverso no es única; existiendo diferentes soluciones $q = \left[q_1, \quad q_2, \quad ..., \quad q_n \right]^T$ que posicionan y orientan el extremo del robot del mismo modo. En estos casos una solución cerrada permite incluir determinadas restricciones que aseguren que la solución obtenida sea la más adecuada de entre las posibles (por ejemplo, límites en los recorridos articulares).

No obstante, a pesar de las dificultades comentadas, la mayor parte de los robots poseen cinemáticas relativamente simples que facilitan en cierta medida la resolución de su problema cinemático inverso. Por ejemplo, si se consideran sólo los tres primeros grados de libertad de muchos robots, estos tienen una estructura planar, esto es, los tres primeros elementos quedan contenidos en un plano. Esta circunstancia facilita la resolución del problema. Asimismo, en muchos robots se da la circunstancia de que los tres últimos grados de libertad, dedicados fundamentalmente a orientar el extremo del robot, corresponden a giros sobre ejes que se cortan en un punto. De nuevo esta situación facilita el cálculo de la matriz $q = [q_1, \quad q_2, \quad ..., \quad q_n]^T$ correspondiente a la posición y orientación deseadas. Por lo tanto, para los casos citados y otros, es posible establecer ciertas pautas generales que permitan plantear y resolver el problema cinemático inverso de una manera sistemática.

Método geométrico.

Los métodos geométricos permiten obtener normalmente los valores de las primeras variables articulares, que son las que consiguen posicionar el robot (prescindiendo de la orientación de su extremo). Para ello utilizan relaciones trigonométricas y geométricas sobre los elementos del robot. Se suele recurrir a la resolución de triángulos formados por los elementos y articulaciones de robot.

El procedimiento en sí se basa en encontrar suficiente número de relaciones geométricas en las que intervendrán las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos.

Para mostrar el procedimiento a seguir se va aplicar el método a la resolución del problema cinemático inverso de un robot de 3GDL de rotación (estructura típica articular). La Fig. 6 muestra la configuración del robot. El dato de partida son las coordenadas (p_x, p_y, p_z) referidas a $\{S_0\}$ en las que se quiere posicionar su extremo.

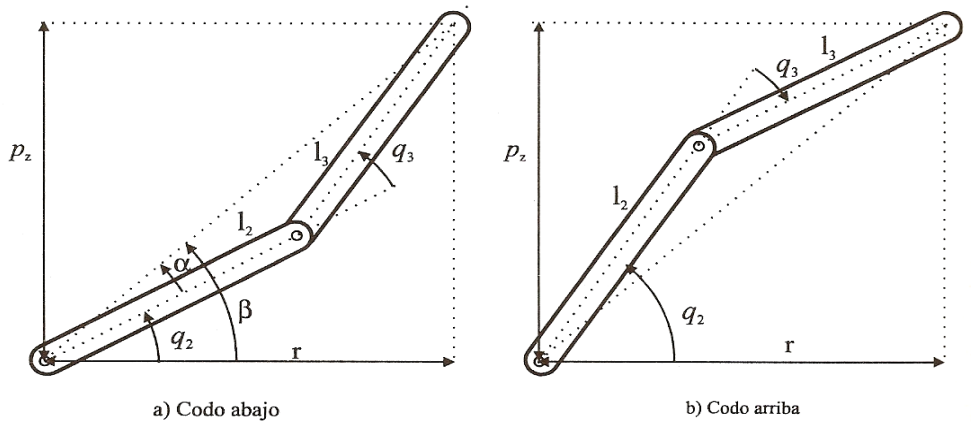


Figura 6. Configuraciones del codo

Como se ve, este robot posee una estructura planar, quedando este plano definido por el ángulo de la primera variable articular q_1.

El valor q_1 se obtiene inmediatamente como:

$$q_1 = arctg\left(\frac{p_y}{p_x}\right) \tag{14}$$

Considerando ahora únicamente los elementos 2 y 3 que están situados en un plano (Fig. 6), y utilizando el teorema del coseno, se tendrá:

$$r^2 = p_x{}^2 + p_y{}^2$$

$$r^2 + p_z{}^2 = l_2{}^2 + l_3{}^2 + 2l_2 l_3 \cos q_3$$

$$\cos q_3 = \frac{p_x{}^2 + p_y{}^2 + p_z{}^2 - l_2{}^2 - l_3{}^2}{2l_2 l_3} \tag{15}$$

Esta expresión permite obtener q_3 en función del vector de posición del extremo p. No obstante, y por motivos de ventajas computacionales, es más conveniente utilizar la expresión de la arcotangente en lugar del arco coseno.

Puesto que

$$sen q_3 = \pm\sqrt{1 - \cos^2 q_3} \tag{16}$$

Se tendrá que

$$q_3 = arctg\left(\frac{\pm\sqrt{1 - \cos^2 q_3}}{\cos q_3}\right) \tag{17}$$

Como se aprecia, existen dos posibles soluciones para q_3 según se tome el signo positivo o el signo negativo en la raíz. Éstas corresponden a las configuraciones de codo arriba y codo abajo del robot.

El cálculo de q_2 se hace a partir de la diferencia entre β y α:

$$q_2 = \beta - \alpha \tag{18}$$

Siendo:

$$\beta = arctg\left(\frac{p_z}{r}\right) = arctg\left(\frac{p_z}{\pm\sqrt{{p_x}^2 + {p_y}^2}}\right)$$
$$\alpha = arctg\left(\frac{l_3 senq_3}{l_2 + l_3\cos q_3}\right) \tag{19}$$

Luego, finalmente

$$q_2 = arctg\left(\frac{p_z}{\pm\sqrt{{p_x}^2 + {p_y}^2}}\right) - arctg\left(\frac{l_3 senq_3}{l_2 + l_3\cos q_3}\right) \tag{20}$$

De nuevo los dos posibles valores según la elección del signo dan lugar a dos valores diferentes de q_2 correspondientes a las configuraciones codo arriba y abajo (Fig. 6).

Desacoplo cinemático.

El procedimiento estudiado anteriormente permite obtener los valores de las tres primeras variables articulares del robot, aquellas que posicionan su extremo en unas coordenadas (p_x, p_y, p_z) determinadas, aunque puede ser igualmente utilizadas para la obtención de las seis a costa de una mayor complejidad.

Ahora bien, como es sabido, en general no basta con posicionar el extremo del robot en un punto del espacio, sino que casi siempre es preciso también conseguir que la herramienta que aquél porta se oriente de una manera determinada. Para ello, los robots cuentan con otros tres grados de libertad adicionales, situados al final de la cadena cinemática y cuyos ejes, generalmente, se cortan en un punto, que informalmente se denomina *muñeca del robot.* Si bien la variación de estos tres últimos grados de libertad origina un cambio en la posición final del extremo real del robot, su verdadero objetivo es poder orientar la herramienta del robot libremente en el espacio.

El método de desacoplo cinemático saca partido de este hecho, separando ambos problemas: posición y orientación. Para ello, dada una posición y orientación final deseadas, establece las coordenadas del punto de corte de los tres últimos ejes (*muñeca del robot*) calculándose (q_1, q_2, q_3) que consiguen posicionar este punto.

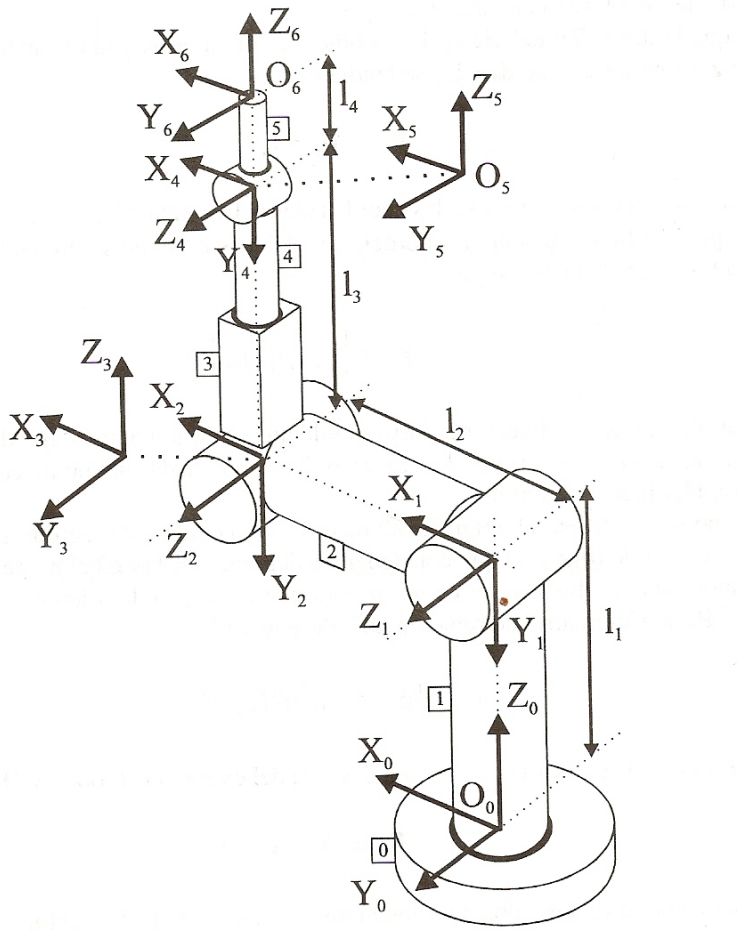


Figura 7. Cinemática del robot

El punto central de la muñeca del robot corresponde al origen del sistema $\{S_5\}$:O_5 (Fig. 7). Por su parte, el punto final del robot será el origen del sistema $\{S_6\}$:O_6. En lo que sigue se utilizarán los vectores:

$$p_m = \overline{O_0 O_5}$$
$$p_r = \overline{O_o O_6} \tag{21}$$

Que van desde el origen del sistema asociado a la base del robot $\{S_0\}$ hasta los puntos centro de la muñeca y fin del robot, respectivamente.

Puesto que la dirección del eje z_6 debe coincidir con la del z_5 y la distancia entre O_5 y O_6 medida a lo largo de z_5 es precisamente $d_4=l_4$, se tendrá que:

$$p_m = p_r - l_4 z_6 \tag{22}$$

Estando todos los vectores referidos a las coordenadas del sistema $\{S_0\}$.

Sabemos que p_r son las coordenadas del punto donde se pretende que se posicione el robot expresadas en $\{S_0\}$. Por lo tanto

$$p_r = \left[p_x, p_y, p_z \right]^T \tag{23}$$

El vector director z_6 es el vector a correspondiente a la orientación deseada $z_6 = \left[a_x, \ a_y, \ a_z \right]^T$ y l_4 es un parámetro asociado con el robot. Por lo tanto, las coordenadas del punto central de la muñeca (p_{mx}, p_{my}, p_{mz}) son fácilmente obtenibles.

Tal y como se mostró en el método geométrico es posible, calcular los valores (q_1, q_2, q_3) que consisten en posicionar el robot en el p_m deseado.

Queda ahora obtener los valores de q_4, q_5 y q_6 que consiguen la orientación deseada. Para ello, denominando ${}^0R_6$ a la submatriz de rotación ${}^0T_6$ se tendrá:

$${}^0R_6 = \left[n \quad o \quad a \right] = {}^0R_3 \quad {}^3R_6 \tag{24}$$

Donde ${}^0R_6$ es conocida por ser la orientación deseada del extremo del robot, y ${}^0R_3$ definida por:

$${}^0R_3 = {}^0A_1 \quad {}^1A_2 \quad {}^2A_3 \tag{25}$$

También lo será a partir de los valores ya obtenidos de q_1, q_2, q_3. Por lo tanto:

$${}^3R_6 = \left[r_{ij} \right] = \left({}^oR_3 \right)^{-1} \qquad {}^0R_6 = \left({}^0R_3 \right)^T \left[n \quad o \quad a \right] \tag{26}$$

Tendrá sus componentes numéricas conocidas.

Por otra parte, ${}^3R_6$ corresponde con la submatriz (3x3) de rotación de la matriz de transformación homogénea ${}^3T_6$ que relaciona el sistema $\{S_3\}$ con el $\{S_6\}$. Por lo tanto:

$${}^3R_6 = {}^3A_4 \quad {}^4A_5 \quad {}^5A_6 \tag{27}$$

Donde ${}^{i-1}R_i$ es la submatriz de rotación de la matriz de Denavit-Hartenberg ${}^{i-1}A_i$, cuyos valores son:

$${}^3R_4 = \begin{bmatrix} C_4 & 0 & -S_4 \\ S_4 & 0 & C_4 \\ 0 & -1 & 0 \end{bmatrix} \qquad {}^4R_5 = \begin{bmatrix} C_4 & 0 & S_5 \\ S_5 & 0 & -C_5 \\ 0 & 1 & 0 \end{bmatrix} \qquad {}^5R_6 = \begin{bmatrix} C_6 & -S_6 & 0 \\ S_6 & C_6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{28}$$

Luego se tiene que:

$$ {}^{3}R_6 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & C_4S_5 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 \\ -S_5C_6 & S_5S_6 & C_5 \end{bmatrix} \tag{29} $$

Donde r_{ij} debido a (26) serán valores numéricos conocidos:

$$ {}^{3}R_6 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & C_4S_5 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 \\ -S_5C_6 & S_5S_6 & C_5 \end{bmatrix} \tag{30} $$

De las nueve relaciones expresadas en la ecuación anterior se pueden tomas las correspondientes a r_{13}, r_{23}, r_{33}, r_{31} y r_{32}:

$$ \begin{aligned} r_{13} &= C_4S_5 & r_{23} &= S_4S_5 & r_{33} &= C_5 \\ r_{31} &= -S_5C_6 & r_{32} &= S_5S_6 \end{aligned} \tag{31} $$

Del conjunto de ecuaciones anterior es inmediato obtener los valores de los parámetros articulares (se recomienda convertir todas las funciones trigonométricas inversas en su arcotangente, por ser ésta computacionalmente más robusta).

$$ \begin{aligned} q_4 &= \text{arc}\,sen\left(\frac{r_{23}}{r_{33}}\right) \\ q_5 &= ar\cos\left(r_{33}\right) \\ q_6 &= \arctan\left(-\frac{r_{32}}{r_{31}}\right) \end{aligned} \tag{32} $$

Las ecs. (32) junto con (14), (17) y (20), y teniendo en cuenta que las posiciones de cero son distintas, constituyen la solución completa del problema cinemático inverso para un robot de 6 GDL.

III. Manufactura de la estructura de robot.

Materiales utilizados.

- 1 barra cuadrada de acero cold rolled de 1m de longitud y 31.7mm de lado
- 1 solera de aluminio 1018 de 1.8m de largo, de 31.7mm x 6.1mm por lado.
- Tornillos de acero de cabeza plana:
  - 24 tornillos M3x0.5x12
  - 48 tornillos M4x0.7x8
  - 9 tornillos M3x0.5x8
- 10 rodamientos de agujas modelo HK 1210 de la marca INA
- 5 Potenciómetros de 1kΩ de una vuelta
- 1 machuelo para rosca M3x0.5 y M4x0.7
- 2 juntas de neopreno de 30mm de diámetro x 2 mm de espesor
- 5 tuercas de acero de M12x1.75
- 5 rondanas de 20mm de diámetro x 2mm de espesor
- 1 base cuadrada de 700x450 mm

Maquinaria utilizada y accesorios.

- 1 Torno [Birminham CT-1440G, 1999, Taiwán, 70-1350 rpm, 220V]
- 1 Fresadora [Gate Giewont FWD32, Polonia, 1979, 56 a 1800 rpm, 7.1 kW]
- 1 Taladro [IR2 Mas, 1980, Checoslovaquia, 140 a 4500 rpm, 220V]
- 1 Sierra [Doroco 872MY, USSR]
- Broca [1/8”, 5/16”]
- Buril punta de tungsteno y acero rapido de 1/2”
- Cortador [3/8”,1/2”,5/8”,1”]

Instrumentos de medición.

- Calibrador [Phase, Resolución 1/50]
- Flexometro [Stanley, Galaxy 3m/10’, Resolución 1]

Maquinado para realizar las distintas piezas:

I. Perno 1

a. Se corta un tramo de la barra cuadrada de 85mm de longitud
b. Se maquina el área transversal recién cortada mediante la fresadora con el objetivo de obtener una superficie perpendicular con respecto a los costados.

c. Se maquina en el torno un cilindro de 12mm de diámetro x 50mm de extensión en uno de los extremos de la barra y a lo largo de ésta. En este proceso se debe tener un especial cuidado para lograr los intervalos de tolerancia necesarios.
d. Se maquina en el torno una rosca M12x1.75-6g de 17.3mm de extensión en el extremo del cilindro maquinado en el paso anterior.
e. Se maquina con la fresadora en el extremo plano del cilindro un prisma rectangular alineado con el centro del cilindro de 1.2mm de ancho por 2mm de alto, que servirá de conexión con la cavidad en la punta del potenciómetro
f. Se maquinan sobre una de las caras planas perpendiculares al área transversal de la pieza 4 agujeros simétricos (ver dibujo 11 "PERNO 1" en el Anexo 1) con la ayuda del taladro. Hacer lo mismo con la cara posterior a la que se trabajo.
g. Se maquina con la ayuda de un machuelo una cuerda interior [M4x0.7] sobre los ocho agujeros hechos en el paso anterior.

II. Perno 2.

a. Se corta un tramo de la barra cuadrada de 65mm de longitud
b. Se maquina el área transversal recién cortada mediante la fresadora con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina en el torno un cilindro de 12mm de diámetro x 50mm de extensión en uno de los extremos de la barra y a lo largo de esta. En este proceso se debe tener un especial cuidado para lograr los intervalos de tolerancia necesarios.
d. Se maquina en el torno una rosca M12x1.75-6g de 17.3mm de extensión en el extremo del cilindro maquinado en el paso anterior.
e. Se maquina con la fresadora en el extremo plano del cilindro un prisma rectangular alineado con el centro del cilindro de 1.2mm de ancho por 2mm de alto, que servirá de conexión con la cavidad en la punta del potenciómetro
f. Se maquina sobre la cara de donde nace el cilindro 4 agujeros simétricos (ver dibujo "PERNO 2" en el Anexo 1) con la ayuda del taladro.
g. Se maquina con la ayuda de un machuelo una cuerda interior [M4x0.7] sobre los cuatro agujeros hechos en el paso anterior.

III. Perno 3.

a. Se corta un tramo de la barra cuadrada de 100mm de longitud
b. Se maquina el área transversal recién cortada mediante la fresadora con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina en el torno un cilindro de 12mm de diámetro x 50mm de extensión en uno de los extremos de la barra y a lo largo de esta. En este proceso se debe tener un especial cuidado para lograr los intervalos de tolerancia necesarios.
d. Se maquina en el torno una rosca M12x1.75-6g de 17.3mm de extensión en el extremo del cilindro maquinado en el paso anterior.
e. Se maquina con la fresadora en el extremo plano del cilindro un prisma rectangular alineado con el centro del cilindro de 1.2mm de ancho por 2mm de alto, que servirá de conexión con la cavidad en la punta del potenciómetro

f. Se maquinan sobre una de las caras planas perpendiculares al área transversal de la pieza 4 agujeros simétricos (ver dibujo "PERNO 3" en el Anexo 1) con la ayuda del taladro. Hacer lo mismo con la cara posterior a la que se trabajo.
g. Se maquina con la ayuda de un machuelo una cuerda interior [M4x0.7] sobre los ocho agujeros hechos en el paso anterior.
h. Se maquina con la fresadora y con un cortador una ranura de 12.7mm de ancho x 35mm de largo y 11mm de profundidad como se indica en el perno 3

IV. Perno 4.

a. Se corta un tramo de la barra cuadrada de 73mm de longitud
b. Se maquina el área transversal recién cortada mediante la fresadora con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina en el torno un cilindro de 12mm de diámetro x 50mm de extensión en uno de los extremos de la barra y a lo largo de esta. En este proceso se debe tener un especial cuidado para lograr los intervalos de tolerancia necesarios.
d. Se maquina en el torno una rosca M12x1.75-6g de 17.3mm de extensión en el extremo del cilindro maquinado en el paso anterior.
e. Se maquina con la fresadora en el extremo plano del cilindro un prisma rectangular alineado con el centro del cilindro de 1.2mm de ancho por 2mm de alto, que servirá de conexión con la cavidad en la punta del potenciómetro
f. Se maquinan sobre la cara correspondiente al extremo opuesto de donde nace el cilindro 4 agujeros simétricos (ver dibujo "PERNO 4" en el Anexo 1) con la ayuda del taladro. Hacer lo mismo con la cara posterior a la que se trabajo.
g. Se maquina con la ayuda de un machuelo una cuerda interior [M4x0.7] sobre los cuatro agujeros hechos en el paso anterior.
h. Se maquina con la ayuda del taladro un agujero pasado de 8mm sobre el área perpendicular al área transversal de la pieza.
i. Se maquina mediante la fresadora y con un cortador un agujero pasado de 12mm de diámetro usando como guía el barreno pasado en el paso anterior, después de la misma forma se maquina otro agujero de 16mm de diámetro por 10mm de profundidad en ambos extremos de la pieza (ver dibujo "PERNO 4" en el Anexo 1). En este proceso se debe tener un especial cuidado para lograr los intervalos de tolerancia necesarios.
j. Se maquina con la fresadora y con un cortador una ranura de 11.7mm de ancho x 35mm de largo y 18mm de profundidad como se indica en el dibujo Perno 4 (consultar Anexo 1)
k. Se maquina con la fresadora en el extremo plano del cilindro un prisma rectangular alineado con el centro del cilindro de 1.2mm de ancho por 2mm de alto, que servirá de conexión con la cavidad en la punta del potenciómetro

V. Perno herramienta.

a. Se corta un tramo de la barra cuadrada de 50mm de longitud

b. Se maquina el área transversal recién cortada mediante la fresadora con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina en el torno un cilindro de 12mm de diámetro x 47mm de extensión, o sea a lo largo de toda la barra, obteniendo únicamente un cilindro macizo. En este proceso se debe tener un especial cuidado para lograr los intervalos de tolerancia necesarios.
d. Se maquina en el torno una rosca M12x1.75-6g de 17.3mm de extensión en el extremo del cilindro maquinado en el paso anterior.
e. Se maquina con la fresadora en el extremo plano del cilindro un prisma rectangular alineado con el centro del cilindro de 1.2mm de ancho por 2mm de alto, que servirá de conexión con la cavidad en la punta del potenciómetro

VI. Alojamientos

a. Se corta un tramo de la barra cuadrada de 31.7 mm de longitud
b. Se maquina el área transversal recién cortada mediante la fresadora con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina con la ayuda del taladro un agujero pasado de 8mm sobre el área transversal de la pieza.
d. Se maquina mediante la fresadora y con un cortador un agujero pasado de 12mm de diámetro usando como guía el barreno pasado en el paso anterior, después de la misma forma se maquina otro agujero de 16mm de diámetro por 10mm de profundidad en ambos extremos de la pieza (ver dibujo "Alojamiento" en el Anexo 1). En este proceso se debe tener un especial cuidado para lograr los intervalos de tolerancia necesarios.
e. Se maquina sobre una de las caras planas de la pieza 4 agujeros simétricos (ver dibujo "alojamiento") con la ayuda del taladro. Hacer lo mismo con la cara posterior a la que se trabajo.
f. Se maquina con la ayuda de un machuelo una cuerda interior [M4x0.7] sobre los ocho agujeros hechos en el paso anterior

VII. Soporte principal

a. Se corta un tramo de la barra cuadrada de 47mm de longitud
b. Se maquina el área transversal recién cortada mediante la Fresa con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina sobre una de las caras planas de la pieza 4 agujeros simétricos (ver dibujo "Soporte principal" en el Anexo 1) con la ayuda del taladro. Hacer lo mismo con la cara posterior a la que se trabajo y también en la cara inferior de la pieza.
d. Se maquina con la ayuda de un machuelo una cuerda interior [M4x0.7] sobre los doce agujeros hechos en el paso anterior

VIII. Solera 1

a. Se corta un tramo de solera de 160mm de longitud
b. Se maquina el área transversal recién cortada mediante la Fresa con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina sobre una de las caras planas de la pieza 11 agujeros pasados y simétricos (ver dibujo "Solera l_0" en el Anexo 1) con la ayuda del taladro.

IX. Solera 2

a. Se corta un tramo de solera de 160mm de longitud
b. Se maquina el área transversal recién cortada mediante la Fresa con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina sobre una de las caras planas de la pieza 11 agujeros pasados y simétricos (ver dibujo "Solera l_2" en el Anexo 1) con la ayuda del taladro.

X. Solera 3

a. Se corta un tramo de solera de 140mm de longitud
b. Se maquina el área transversal recién cortada mediante la Fresa con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina sobre una de las caras planas de la pieza 14 agujeros pasados y simétricos (ver dibujo "Solera l_4" en el Anexo 1) con la ayuda del taladro.

XI. Base interna de potenciómetro.

a. Se corta un tramo de solera de 31.7mm de longitud
b. Se maquina el área transversal recién cortada mediante la Fresa con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina sobre una de las caras planas de la pieza un agujero pasado y simétrico (ver dibujo "Base 1 & 4" en el Anexo 1) de 9.5mm de diámetro con la ayuda del taladro.

XII. Base externa de potenciómetro.

a. Se cortan dos tramos de solera de 65.5mm (pieza A) y otro de 38.1mm (pieza B) de longitud
b. Se maquina el área transversal recién cortada mediante la Fresa con el objetivo de obtener una superficie perpendicular con respecto a los costados.
c. Se maquina en la pieza A sobre una de las caras laterales 6 agujeros simétricos de 2.5mm de diámetro (ver dibujo "base 2 y 3" en el Anexo 1) con la ayuda del taladro.
d. Se maquinan sobre una de las caras planas de la pieza B 3 agujeros simétricos de 2.5mm de diámetro (ver dibujo "base 2 y 3" en el Anexo 1) con la ayuda del taladro.

e. Se maquina con la ayuda de un machuelo una cuerda interior [M4x0.7] sobre los 9 agujeros hechos en los pasos anteriores.
f. Se maquina sobre la pieza A en una de las caras perpendiculares a su área transversal un agujero pasado y simétrico de 9.5mm de diámetro (ver dibujo "base 2 y 3" en el Anexo 1) con la ayuda del taladro.

Todos los dibujos de las piezas se encuentran en el Anexo 1 1.

Rodamientos INA modelo HK1210

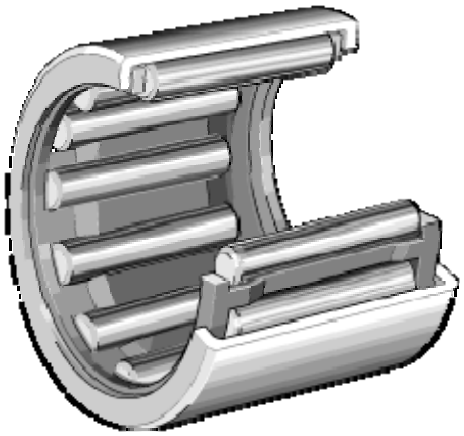


Figura 8. Rodamiento de agujas INA HK1210.

Características.

- Son unidades constructivas formadas por anillos exteriores de pared delgada, conformados sin arranque de viruta, y coronas de agujas
- Permiten ahorrar espacio constructivo en dirección radial
- Son fáciles de montar.
- Se introducen a presión en el agujero del alojamiento
- No requieren ninguna otra fijación axial

Es importante señalar cuáles son los intervalos de tolerancia necesarios para el árbol y el alojamiento que estarán en contacto con el rodamiento, estos son recomendados por el mismo fabricante de los rodamientos, en este caso para el modelo INA HK1210 [5] son:

- Árbol.

IT Axial= k5

IT Radial=h5

- Alojamiento.

IT Radial=H6

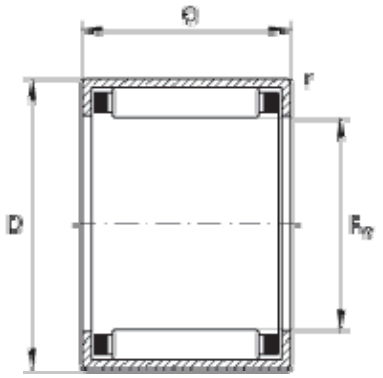


Figura 9. Dimensiones del rodamiento [5].

Dimensiones		
1	$\mathbf{F_w}$	12 mm
2	**D**	16 mm
3	**C**	10 mm
4	$\mathbf{r_{min}}$	0,4 mm
5	**m**	4,6 g

IV. Modelado matemático del robot.

Cinemática inversa (CI).

Para facilitar el análisis de la cinemática inversa decidimos dividir el problema en dos partes. Naturalmente la primera parte es localizar la posición del brazo y la otra es describir la orientación del efector final.

Posición.

Para encontrar la posición del brazo en el espacio basta con conocer los tres primeros ángulos de rotación (θ_1, θ_2, θ_3).

Para dar solución a este problema se eligió el Método Geométrico que consiste en encontrar una ecuación para cada ángulo que describa su comportamiento analizando geométricamente el brazo en dos posiciones cualesquiera distintas. A continuación se describe la metodología que se utilizó.

Para el análisis, por conveniencia, se desplazó el brazo de home a un punto cualquiera en el cual los ángulos θ_1, θ_2 y θ_3 fueran distintos de su posición original como se muestra

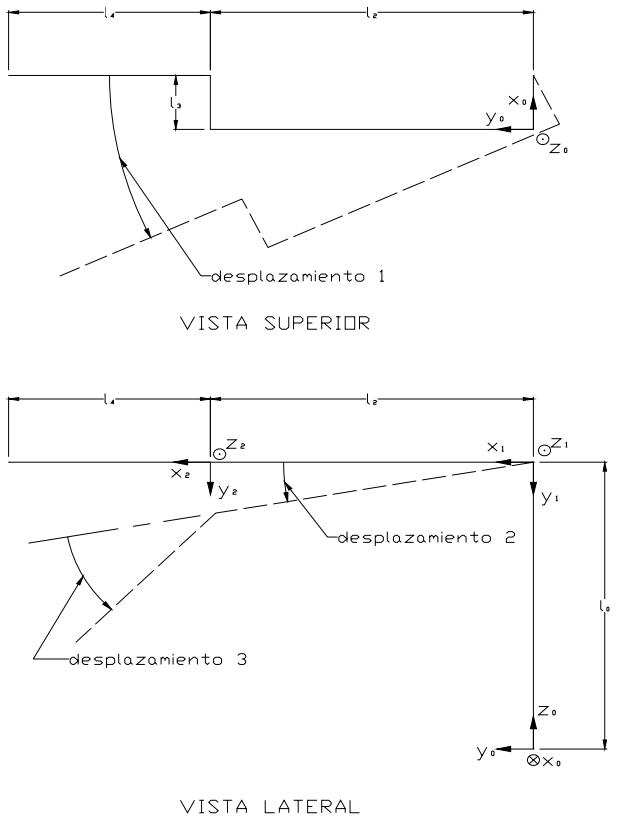


Figura 10. Cinemática inversa. Método geométrico

A continuación se hizo el análisis geométrico de cada ángulo trazando un vector que localiza la posición de la muñeca (R_m) con respecto al origen del sistema de coordenadas S_0 (x_0, y_0, z_0).

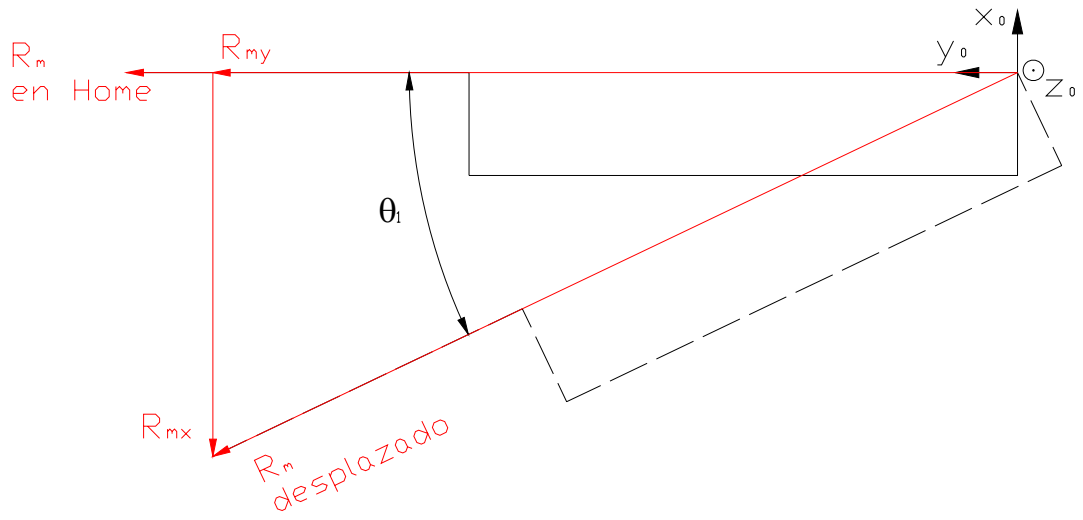


Figura 11. Cinemática inversa. Método geométrico

Partiendo de que para resolver la CI conocemos los valores de la posición (P) y orientación (n, o, a) del efector final es fácil definir el vector R_m.

V. Programación de la interfaz gráfica en MatLab.

La solución de la cinemática de un mecanismo como Estructura de robot PUMA (ErP) es un problema muy complejo que se puede descomponer en subproblemas más simples. A continuación se presenta a grandes rasgos dichos subproblemas.

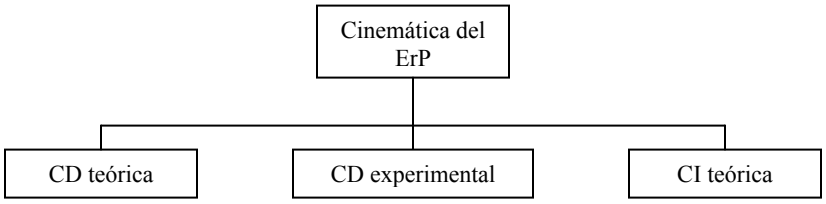


Figura12. Subproblemas

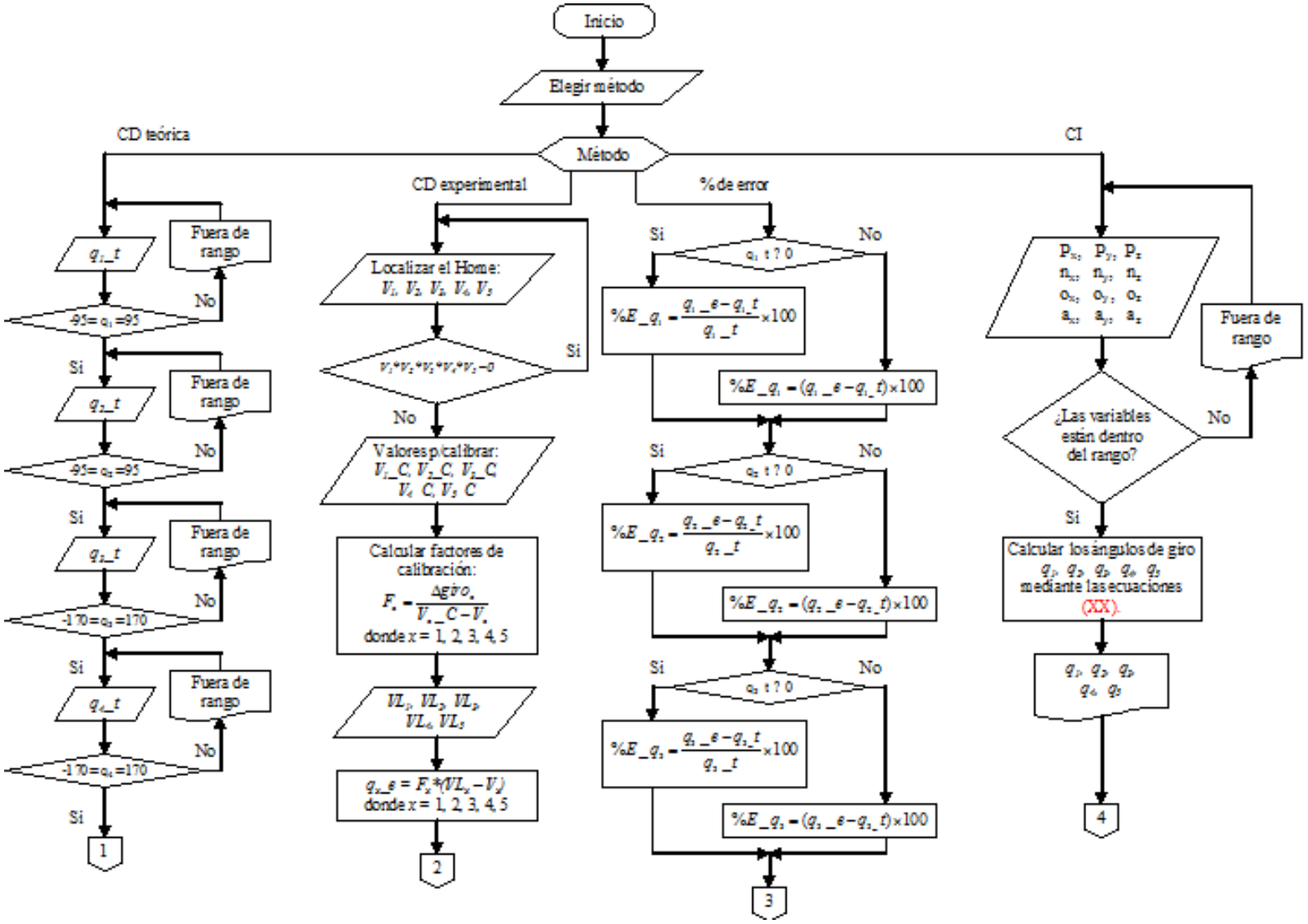


Figura 13. Diagrama de flujo

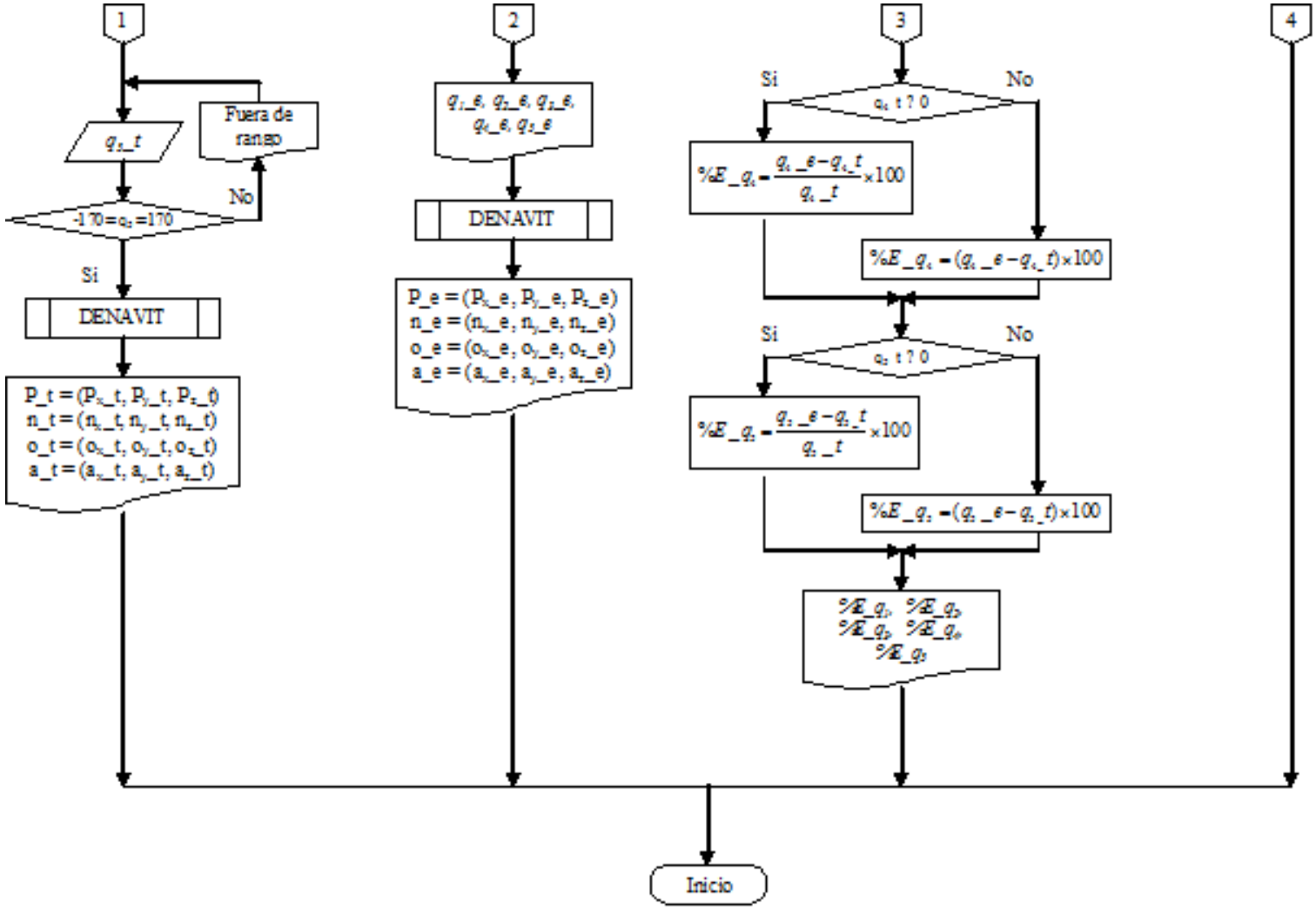


Figura 14. Diagrama de flujo (Continuación)

Descripción de la pantalla del "KINEMATIC SOLVER (KS)".

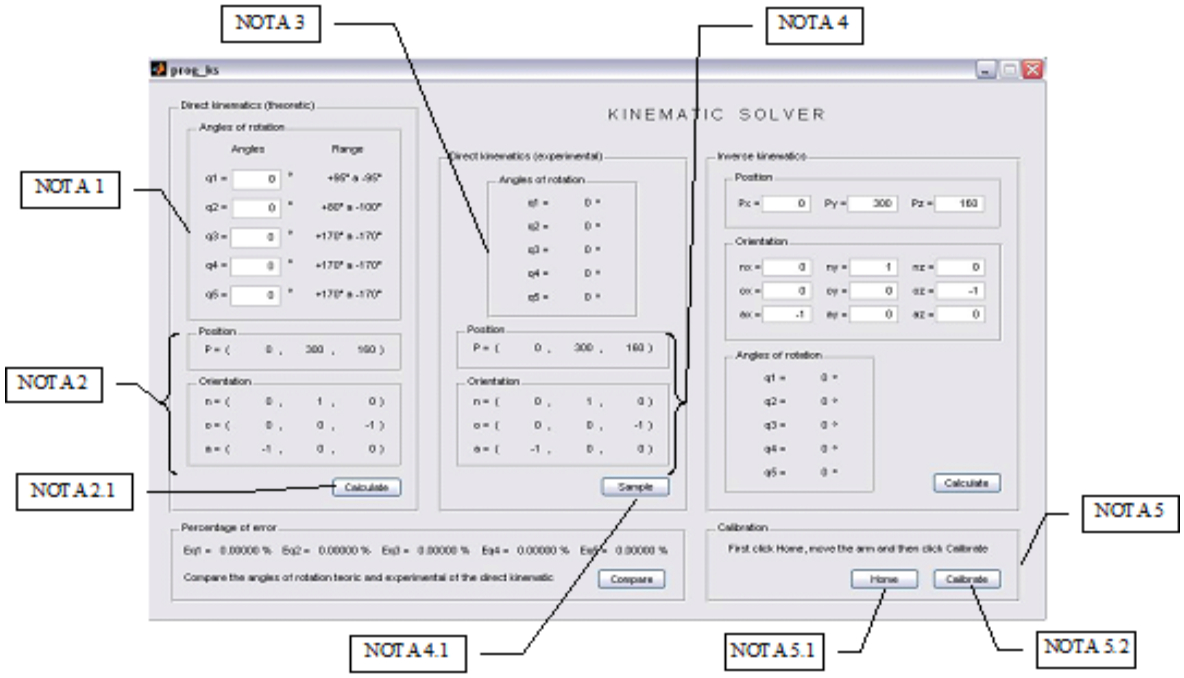


Figura 15. Kinematic Solver.

NOTA 1: en esta sección se leen los ángulos de giro para calcular la CD teórica. Los valores deben estar dentro de los rangos especificados.

NOTA 2: en esta sección se presentan los resultados para la CD con datos de entrada teóricos. P son las coordenadas de efector final, mientras que n, o y a describen la orientación de efector final. Estos resultados se obtienen con el método de Denavit-Hartemberg.

NOTA 2.1: el cálculo de la CD teórica se inicia cuando se da clic este botón (Calculate).

NOTA 3: en esta sección se leen los datos directamente de los censores. Los datos son leídos en volts y se transforman en grados para presentarlos en la pantalla y para hacer los cálculos posteriores.

NOTA 4: en esta sección se presentan los resultados para la CD con los datos de entrada obtenidos de los censores. P son las coordenadas de efector final, mientras que n, o y a describen la orientación de efector final. Estos resultados se obtienen también usando el método de Denavit-Hartemberg.

NOTA 4.1: la lectura de los datos de los censores y el cálculo de la CD experimental se inicia cuando se da clic este botón (Sample). Es muy importante haber hecho una buena calibración del brazo antes de empezar a obtener muestras.

NOTA 5: para obtener resultados satisfactorios de la CD experimental es necesario calibrar el brazo. Para tal efecto están estos dos botones.

NOTA 5.1: para obtener resultados satisfactorios de la CD experimental es necesario calibrar el brazo. Para tal efecto están estos dos botones.

NOTA 5.2: Al dar clic en el botón "Calibrate" el programa lee directamente de los censores la posición de calibración que es conocida:

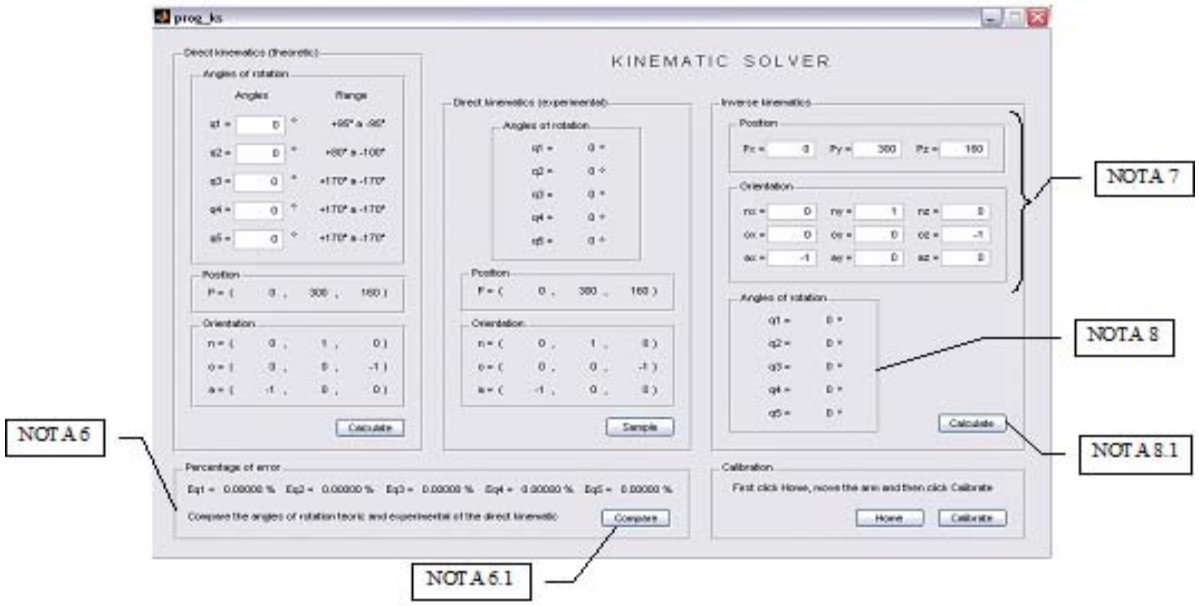


Figura 16. Kinematic Solver.

NOTA 6: en esta sección se presentan los porcentajes de error entre los ángulos de giro teóricos y los experimentales.

NOTA 6.1: al dar clic en el botón "Compare" de inicia el cálculo de los porcentajes de error mencionados en la NOTA 6

NOTA 7: en esta sección se leen la posición y la orientación para calcular la CI teóricamente.

NOTA 8: en esta sección se presentan los resultados para la CI con datos de entrada teóricos. Estos resultados se obtienen mediante las ecuaciones del método geométrico.

NOTA 8.1: el cálculo de la CI se inicia cuando se da clic este botón "Calculate".

Todas estas notas se encuentran en el código fuente del programa. También hay otras notas que no se enlistan aquí ya que son acerca de cálculos internos.

VI. Manual de operación.

En este manual se explica de manera muy sencilla los pasos que se deben de seguir para la calibración y operación de la estructura de robot.

Antes de operar el robot por primera vez es necesario tener en cuenta las siguientes condiciones:

- La estructura cuenta con potenciómetros de una vuelta con tope, por lo tanto se debe tener especial cuidado al girar las articulaciones para no girar más allá del tope del potenciómetro.
- La interfaz debe estar conectada a sus dos puertos izquierdo y derecho antes de conectarla mediante el cable USB a la computadora.

Partes del sistema (ver Fig. 17)

1. Estructura de robot con base graduada
2. Banco para home
3. Apoyo para calibración
4. DAQ USB-6009 de NI.
5. Terminal A (cables rojos)
6. Terminal B (cable blanco y rojo)
7. Cable USB

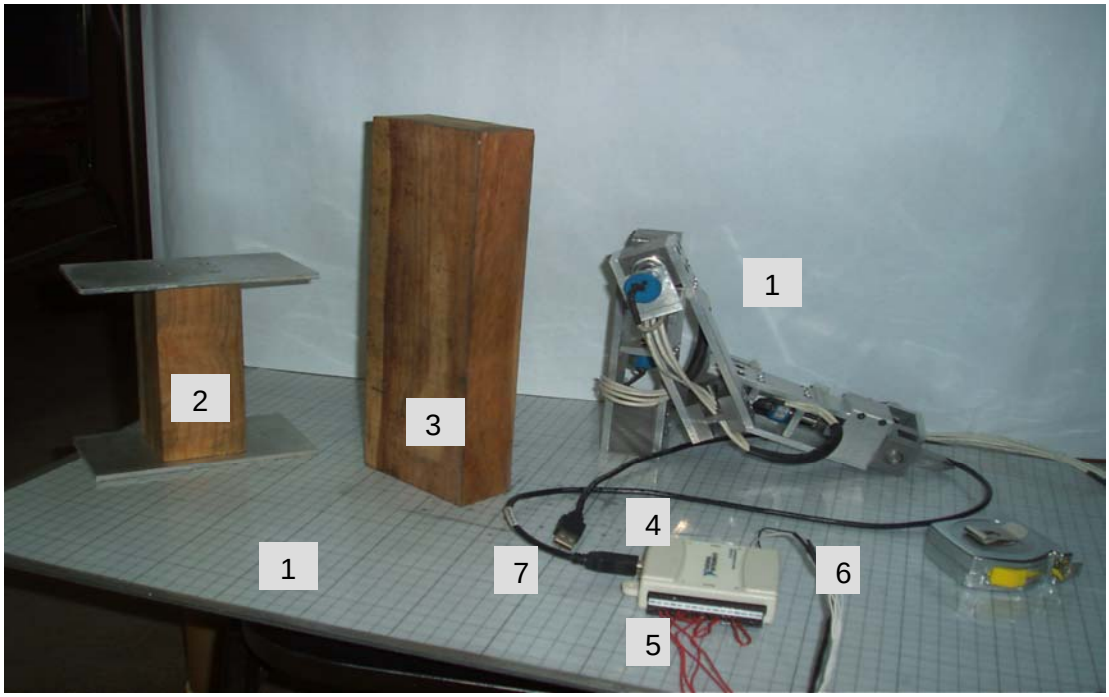


Figura 17. Partes del sistema

Conexión de la interfaz.

Este procedimiento se relaciona con la Fig. 18.

1. Conectar la terminal A del robot a la terminal análoga de la interfaz USB-6009.
2. Conectar la terminal B del robot a la terminal digital de la interfaz USB-6009.
3. Conectar el puerto de menor tamaño del cable USB a la interfaz USB-6009.
4. Conectar el cable USB a la computadora.

NOTA: Es posible que aparezca un cuadro de dialogo ofreciendo abrir o ejecutar los programas de National Instruments, ignorar esta oferta y cerrar el cuadro de dialogo sin elegir ninguna opción.
Ahora la interfaz "NI USB-6009" esta lista para su uso con la PC.

PRECAUCIÓN: No hacer conexiones entre las terminales mientras la interfaz este conectada con la PC ya que podría causar daños irreparables en la interfaz.

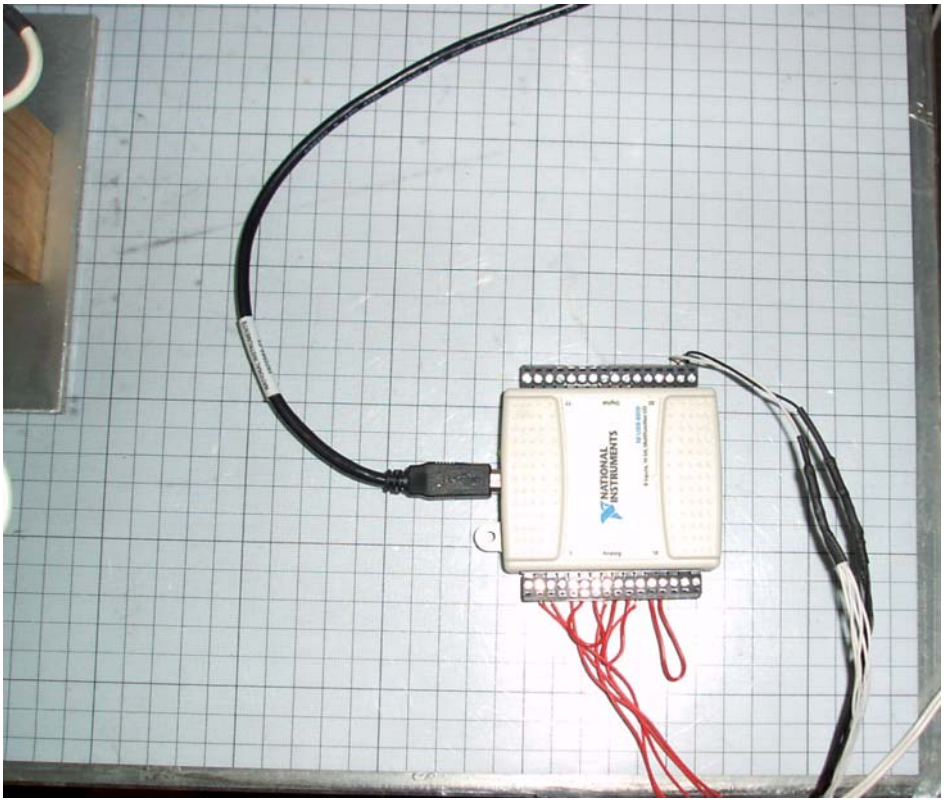


Figura 18. Conexión interfaz

Una vez conectada la interfaz al puerto USB de la computadora, es necesario abrir el cuadro de operación llamado "Kinematic Solver".

Ejecución del Kinematic Solver con Matlab.

1. Ejecutar MatLab desde donde este instalado en su PC

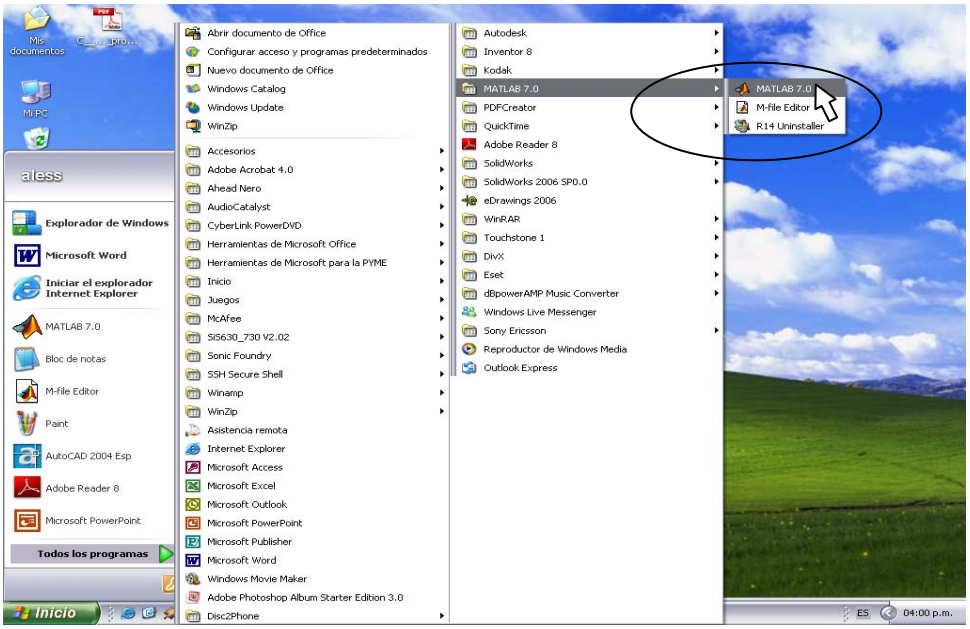


Figura 19. Ejecución de Matlab

2. Una vez abierta la ventana de Matlab indicar en la pestaña "Current Directory" la ubicación o dirección de la carpeta en donde se encuentra guardado el programa. Por ejemplo: C:\Documents and Settings\Unknown User\Escritorio\programa

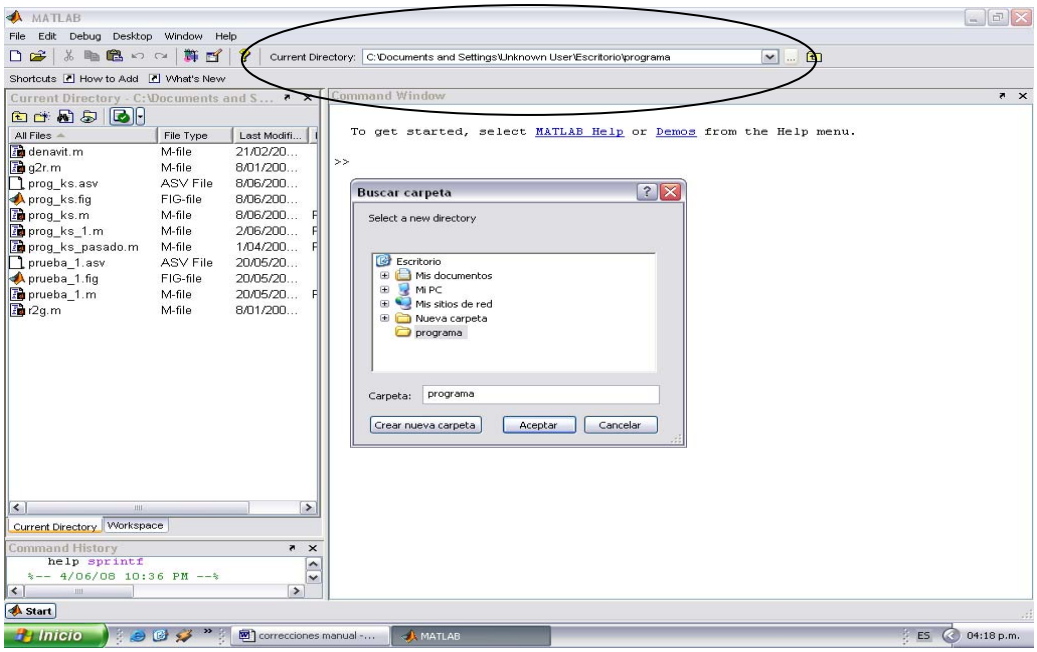


Figura 20. Ubicación del directorio en Matlab

3. Ahora dar clic en el icono GUIDE  que se encuentra en la barra de herramientas

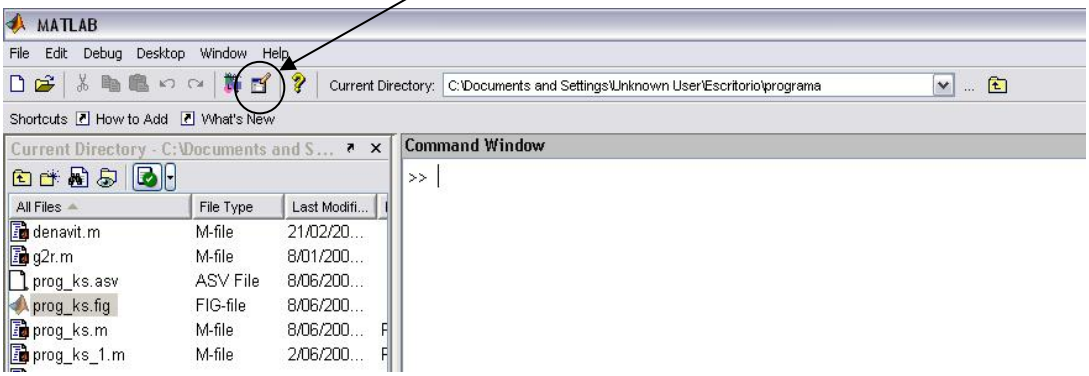


Figura 21. Ejecución de Matlab

4. Se desplegará un cuadro de dialogo (Fig. 22). Elegir la pestaña "Open Existing GUI". Ahí elija o busque el archivo prog_ks.fig (para buscar el archivo de clic en el "Browse..."

   Ejemplo de una ubicación:

   C:\Documents and Settings\Unknown User\Escritorio\programa\prog_ks.fig

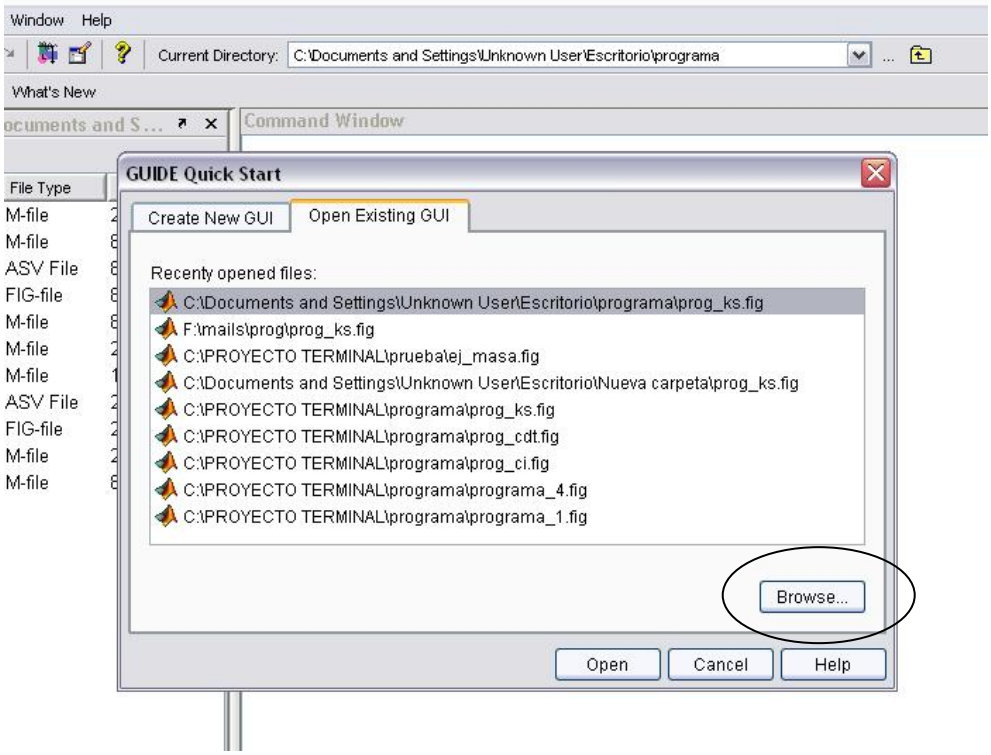


Figura 22. Ejecución de Matlab

5. Se abrirá la ventana Kinematic Solver (ver Fig. 23), dar clic en el icono Run que se encuentra en la barra de herramientas

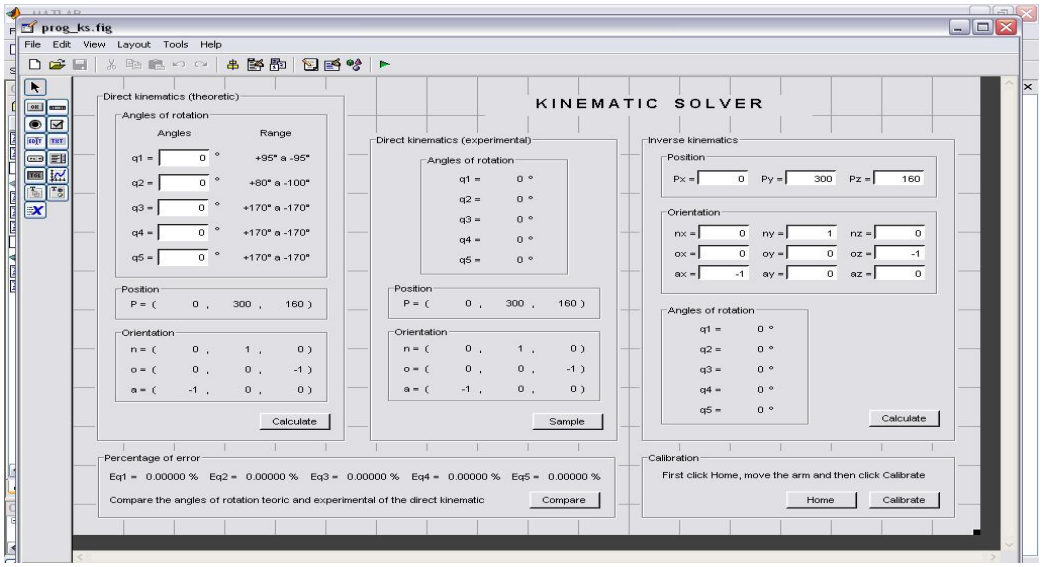


Figura 23. Kinematic Solver

6. Se ejecutara la ventana gráfica del Kinematic Solver. Ahora esta listo para utilizarlo con la estructura ErP (Fig. 24)

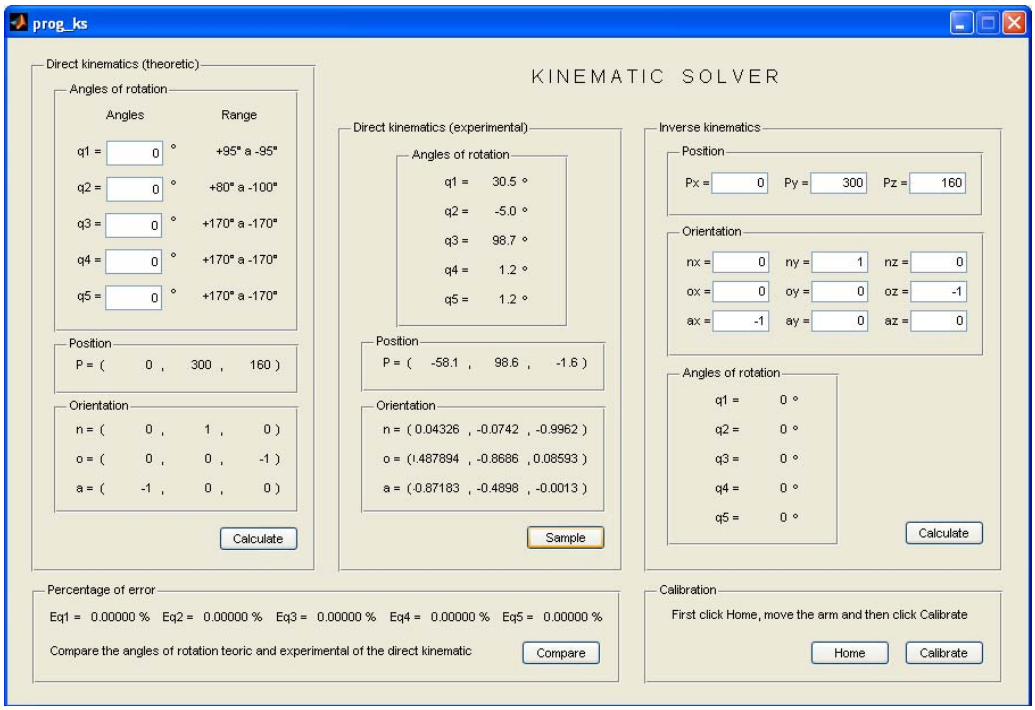


Fig. 24. Cuadro de operación

Ya está todo conectado, antes de empezar a operar la estructura de robot, es necesario y esencial calibrar los giros de las articulaciones.

Pasos para calibrar.

1. Colocar la estructura de robot en su posición de home (ver Fig. 25 y 26), en esta posición es necesario hacer uso del "banco para home", dicho banco tiene una etiqueta en una de las caras de las placas de aluminio atornilladas a sus extremos que indica hacia donde se debe orientar el banco.

Figura 25. Robot en posición de Home

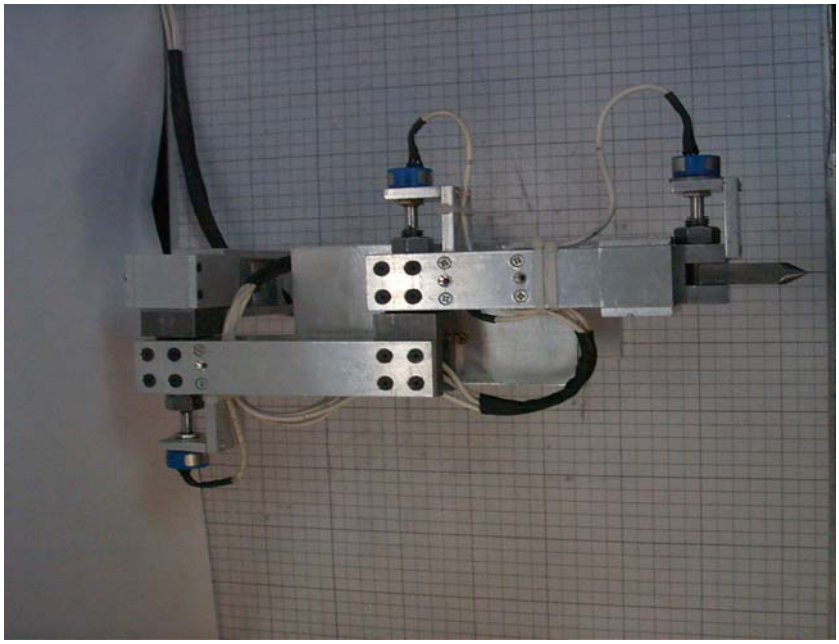


Figura 26. Robot en posición de Home

2. Una vez que está perfectamente alineada la estructura, se presiona el botón de 

en el cuadro de operación en la computadora.
3. Coloca la estructura de robot en su posición de calibración, para este paso usamos el "apoyo para calibración" (ver Fig. 27 y 28). Los giros que se deben dar son los siguientes:

   q1 = 90°; q2 = -90°; q3 = 90°; q4 = 90°; q5 = -70°.

Figura 27. Calibracion

Figura 28. Calibracion

La función de este apoyo es la de proporcionar un paralelismo vertical más preciso entre las caras de los primeros dos eslabones, es decir entre la soleras de la base y la solera siguiente. También otra función que cumple es la de garantizar una perpendicularidad entre las caras de los eslabones dos y tres.
Debe mencionarse que por la posición de los cables, es necesario sujetar con fuerza con nuestras manos este apoyo, presionándolo en contra de las caras donde se busca el paralelismo.

4. Una vez sujetada perfectamente la estructura y estando seguros de que se garantiza las condiciones de paralelismo y perpendicularidad entre las caras mencionadas, presionar

el botón [Calibrate] en el cuadro de operación en la computadora. Ahora nuestra estructura esta calibrada y lista para calcular cualquier posición.

Pasos para calcular la posición de la estructura usando las lecturas de los potenciómetros.

1. Mover la estructura a la posición deseada y fijarla, para fijar la posición es necesario girar las tuercas hasta conseguir un apriete suficiente.
2. Presionar el botón [Sample] en el cuadro de operación y observar los resultados.

Los resultados que arroja el programa se encuentran en la parte media del cuadro de operación:

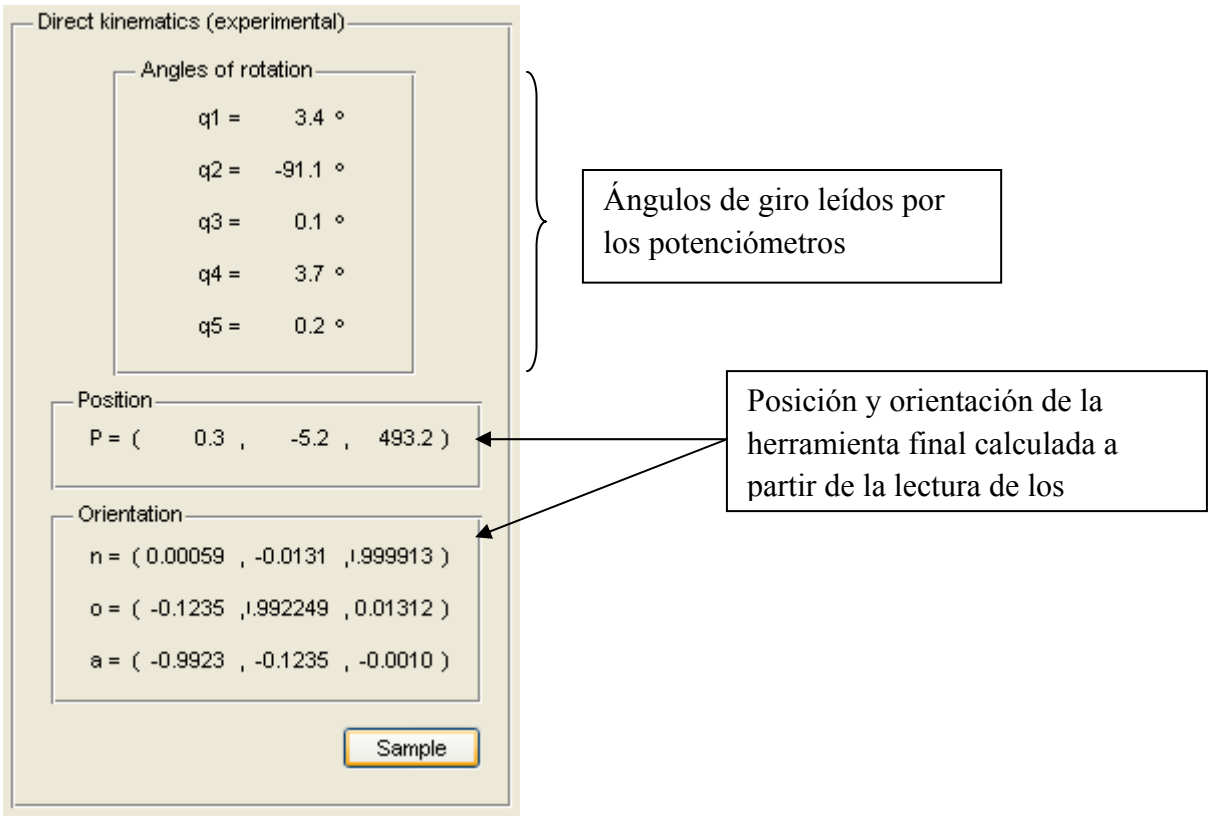


Figura 29. Cinematica experimental

Tenemos ya un resultado experimental, medido a partir de los propios instrumentos de medición proporcionados por la misma estructura. Ahora se procede a tomar distintas mediciones físicas, por medio de por ejemplo un flexometro, una escuadra o un transportador. Estas mediciones nos servirán para calcular la posición de la estructura ya sea por cinemática directa o cinemática inversa.

Pasos para calcular la posición de la estructura por cinemática directa o inversa.

1. Mover la estructura a la posición deseada y fijarla, para fijar la posición es necesario girar las tuercas hasta conseguir un apriete suficiente.
2. Medir físicamente los ángulos entre los eslabones (cinemática directa) o medir la posición (x, y, z) y calcular la orientación (cinemática inversa).
3. Presionar el botón [Calculate] en el cuadro de operación y observar los resultados.

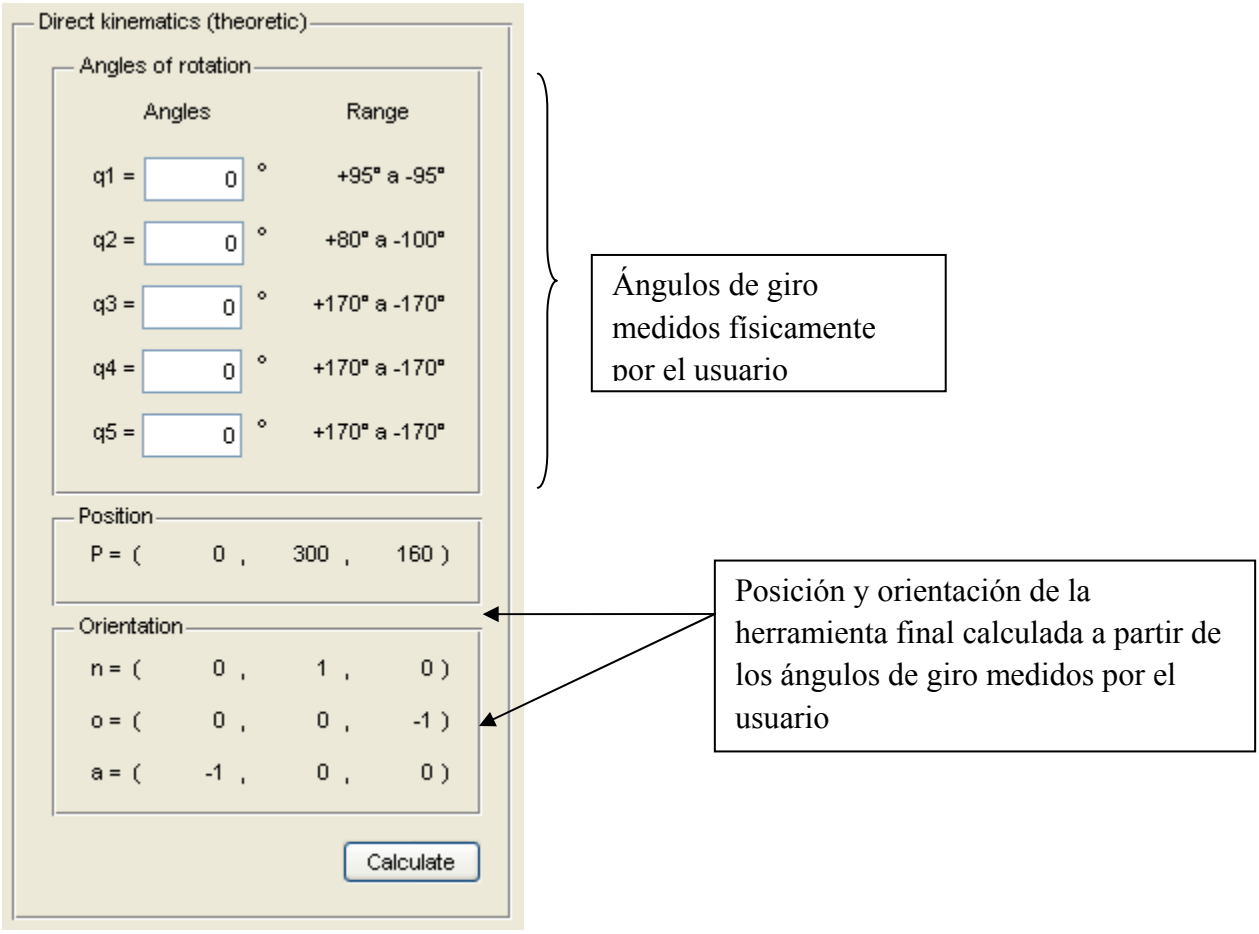


Figura 30. Cinematica directa

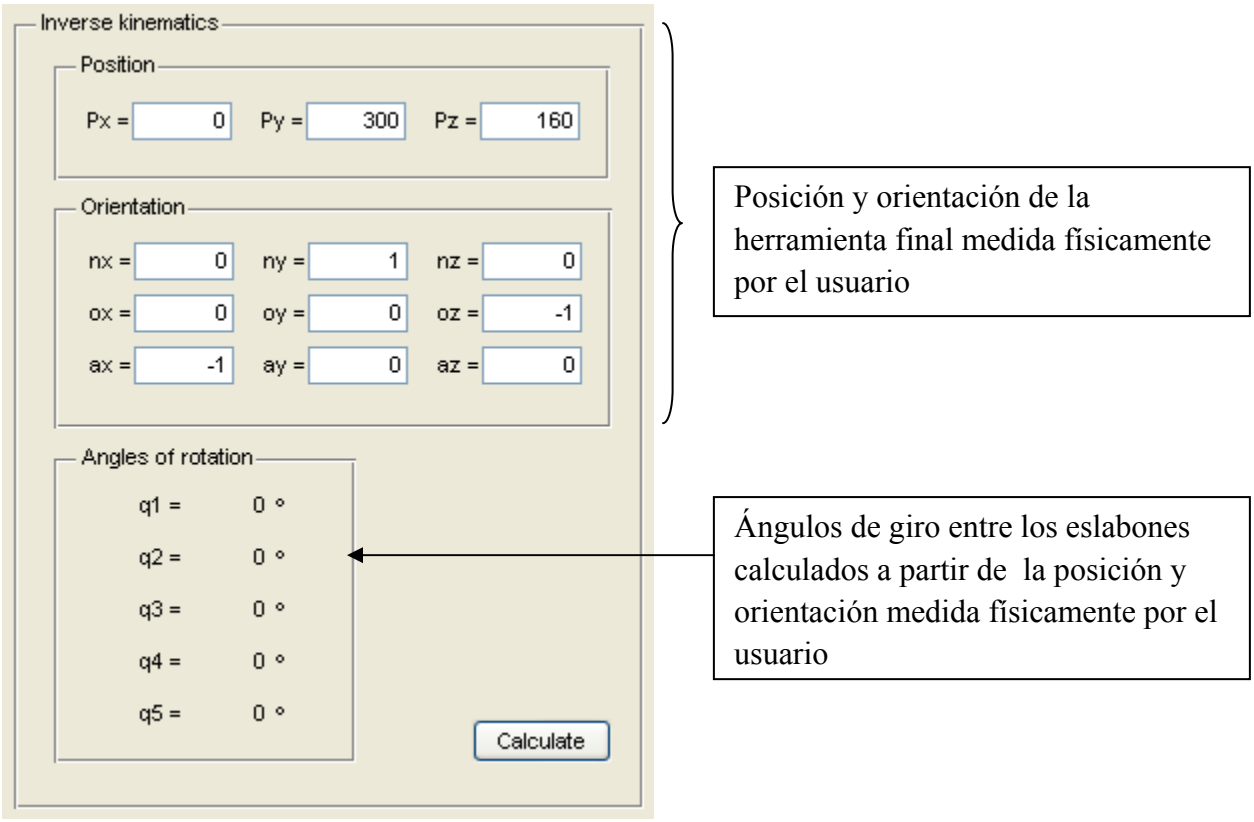


Figura 31. Cinematica inversa

Como calcular los errores porcentuales entre la CD experimental y la CD teórica

Para calcular los errores porcentuales entre las CD experimental y la CD teórica basta con dar clic en el botón "Compare" y se desplegaran los errores correspondientes a cada articulación.

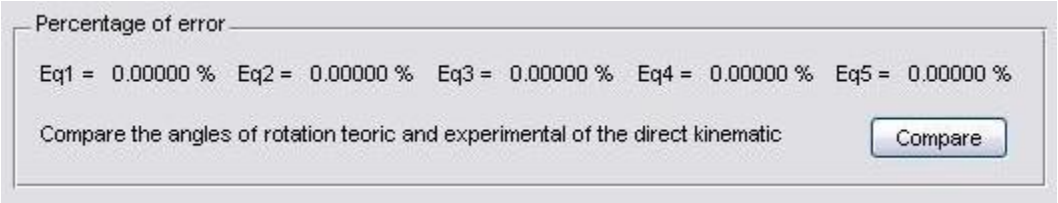


Figura 32. Errores porcentuales

VII. Experimentación y resultados.

Antes de llevar a cabo cualquier prueba es necesario como se menciona en el manual de operación, calibrar la estructura. Ahora para distinguir entre una buena o una mala calibración, configuramos nuevamente la estructura en el home, utilizando el banco para home y alineando perfectamente el eje central de cada uno de de los eslabones entre ellos.

Calibración A

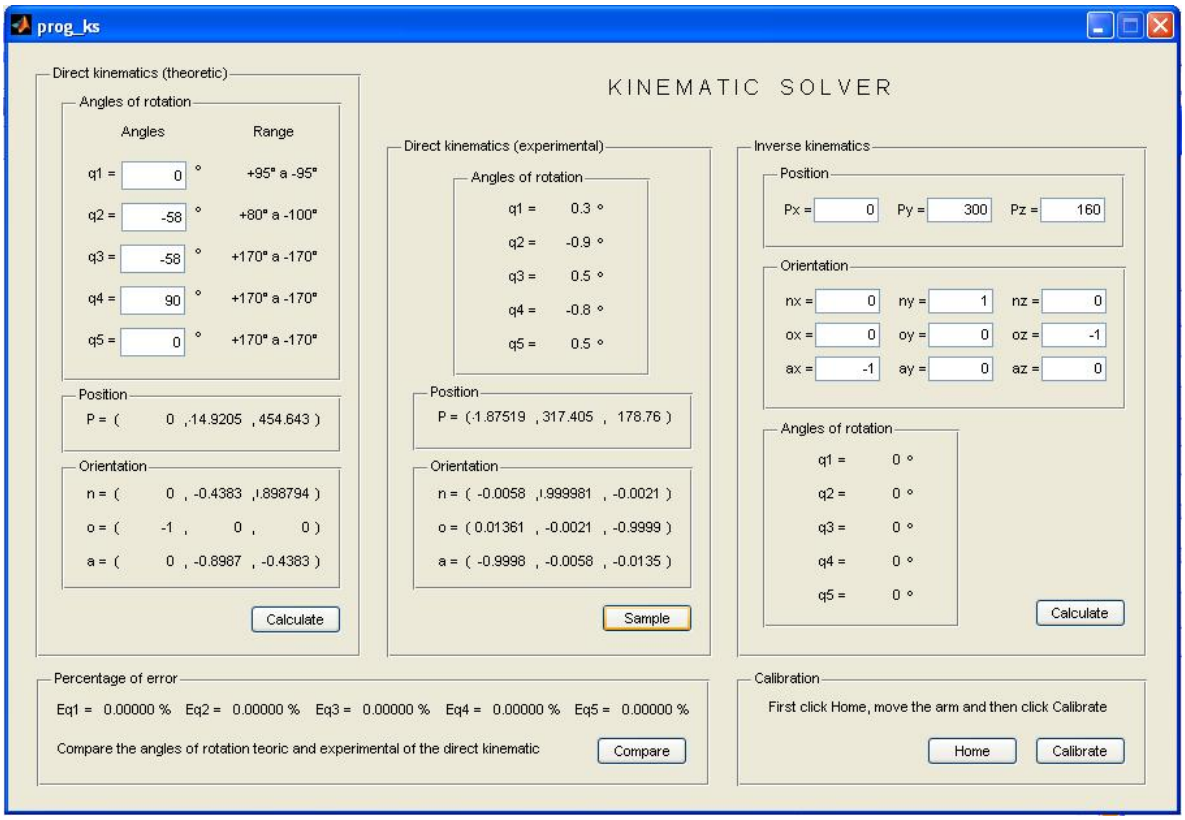


Figura 33. Calibración A

Calibración B

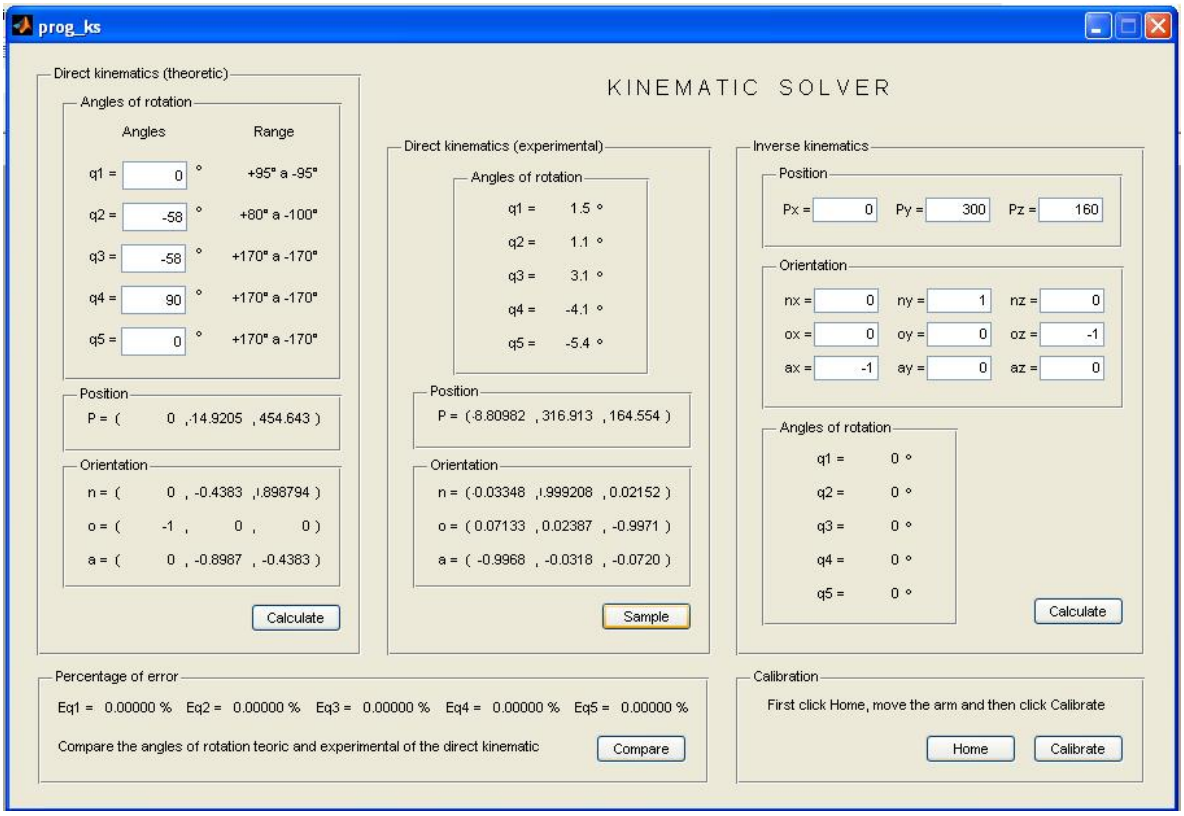


Figura 34. Calibración B

Las pruebas siguientes se realizaron después de llevar a cabo la calibración A, las razones se explicaran posteriormente en la discusión de resultados.

Medida física: Medición física del punto (x, y, z) de la herramienta final

Medida experimental: Medición del punto (x, y, z) de la herramienta final a partir de la lectura de los potenciómetros.

θ: Angulo formado por las componentes x & y situado en el origen de la estructura (q1)

Prueba 1.

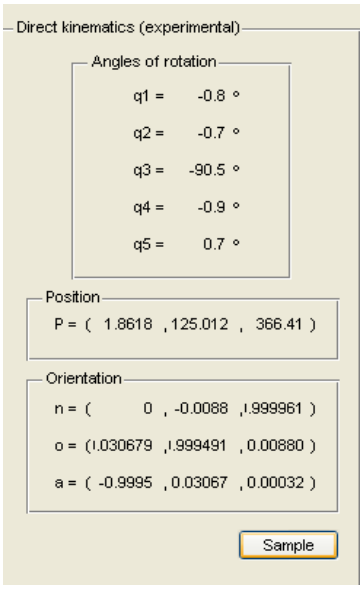


Figura 35. Prueba 1

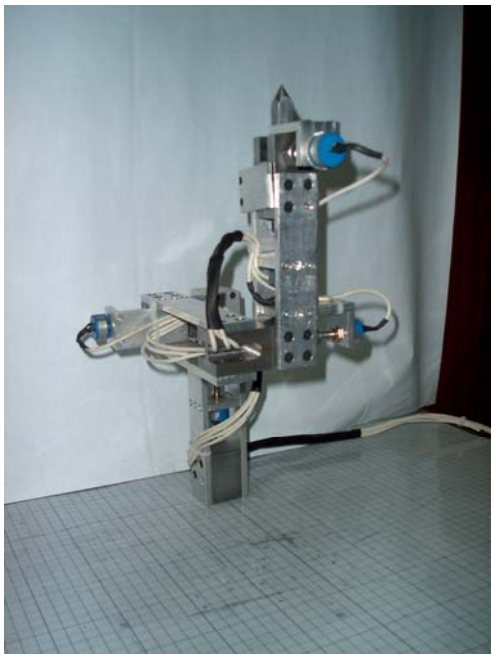


Figura 36. Prueba 1

Tabla de resultados

Prueba 1		x [mm]	y [mm]	θ[º]	z [mm]
1	Medida física	5	127	2.25	364
2	Medida experimental	1.8618	125.012	0.853	366.41

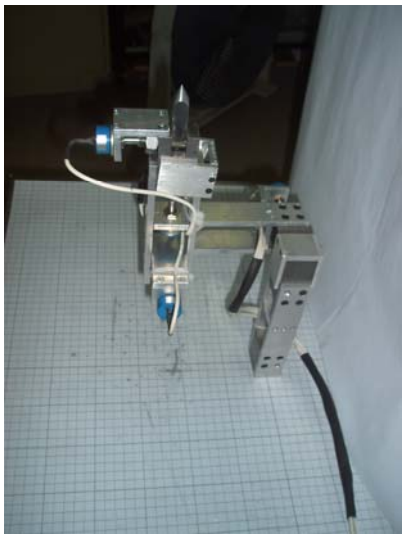


Figura 37. Prueba 1

Prueba 2

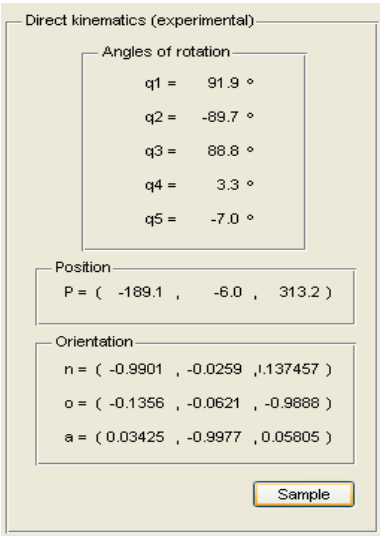


Figura 38. Prueba 2

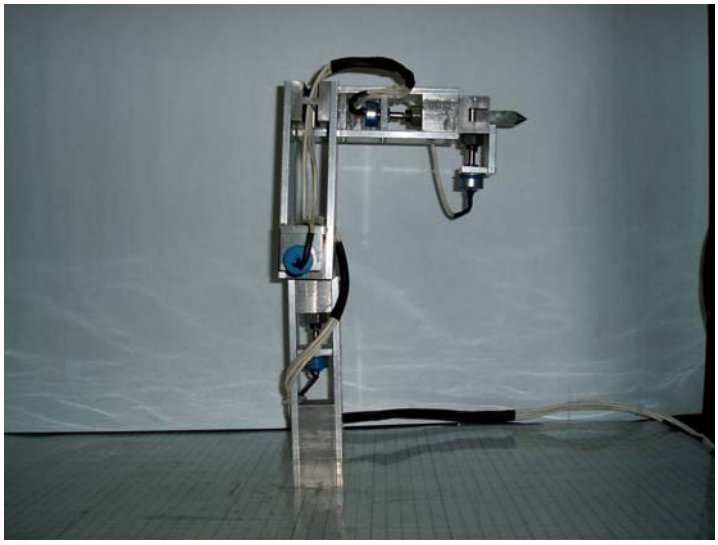


Figura 39. Prueba 2

Tabla de resultados

Prueba 2		x [mm]	y [mm]	θ[°]	z [mm]
1	Medida física	-190	5	-88.493	303
2	Medida experimental	-189.1	-6.0	-91.817	313.2

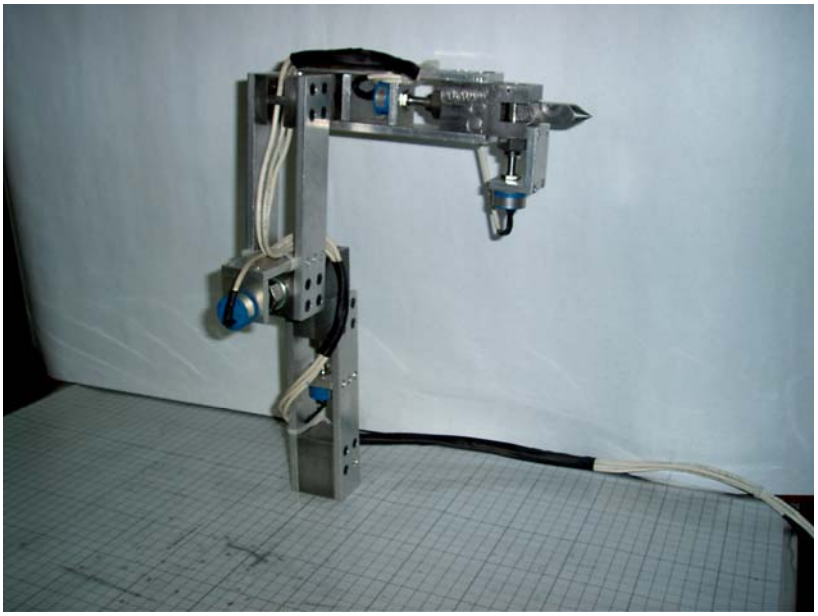


Figura 40. Prueba 2

Prueba 3

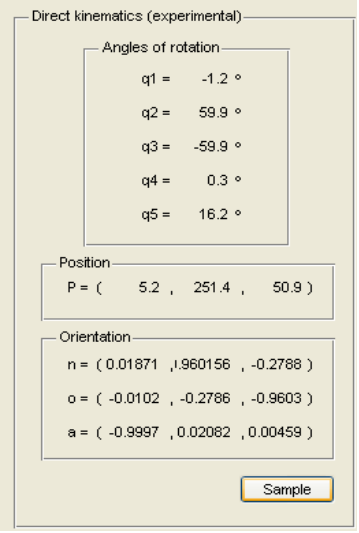


Figura 41. Prueba 3

Figura 42. Prueba 3

Tabla de resultados

Prueba 3		x [mm]	y [mm]	θ[º]	z [mm]
1	Medida física	3	252	0.682	57
2	Medida experimental	5.2	251.4	1.185	50.9

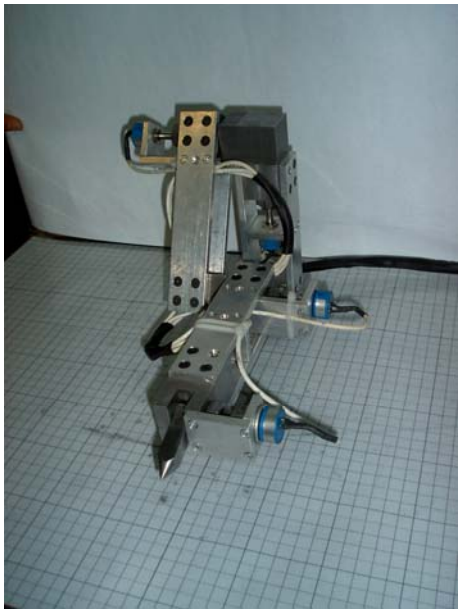


Figura 43. Prueba 3

Prueba 4

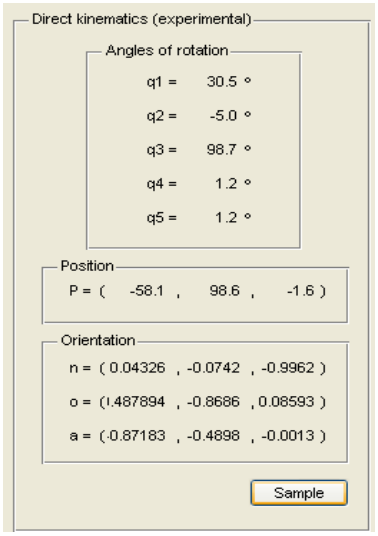


Figura 44. Prueba 4

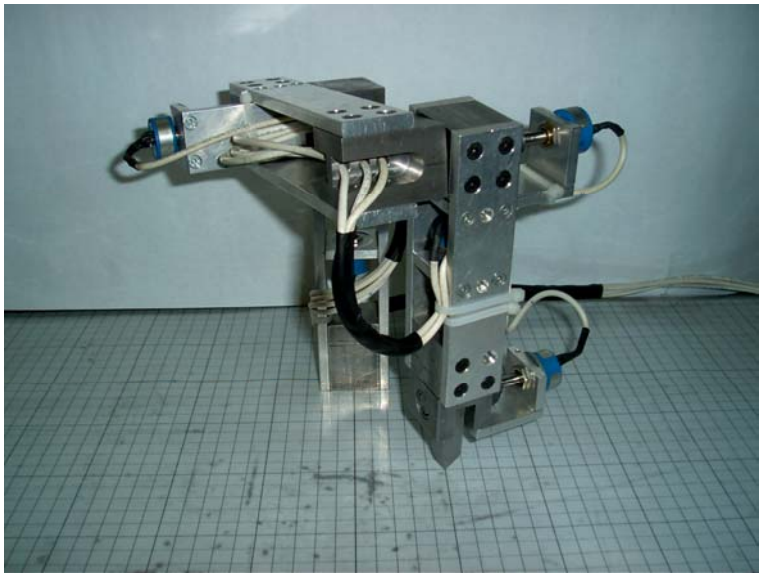


Figura 45. Prueba 4

Tabla de resultados

Prueba 4		x [mm]	y [mm]	θ[º]	z [mm]
1	Medida física	-50	100	-26.565	0
2	Medida experimental	-58.1	98.6	-30.501	-1.6

Prueba 5.

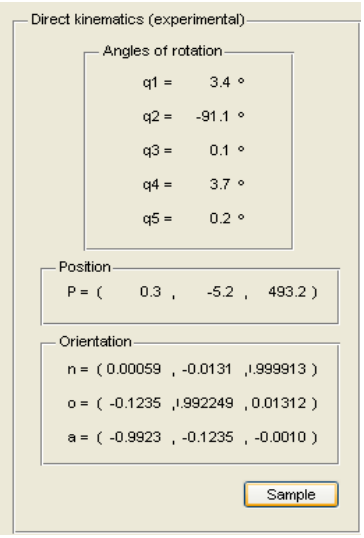


Figura 46. Prueba 5

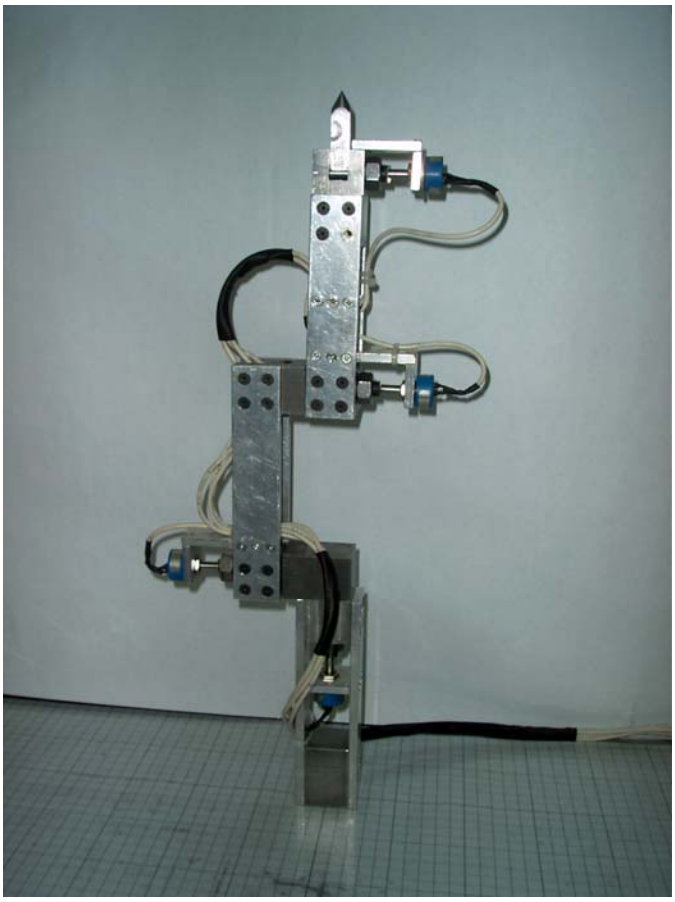


Figura 47. Prueba 5.

Tabla de resultados

Prueba 5		x [mm]	y [mm]	θ[º]	z [mm]
1	Medida física	0	0	0	492
2	Medida experimental	0.3	-5.2	-3.302	493.2

Prueba 6.

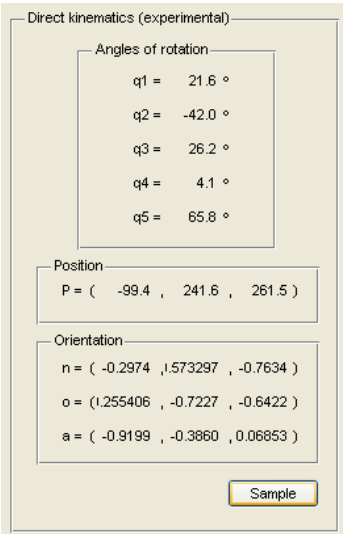


Figura 48. Prueba 6.

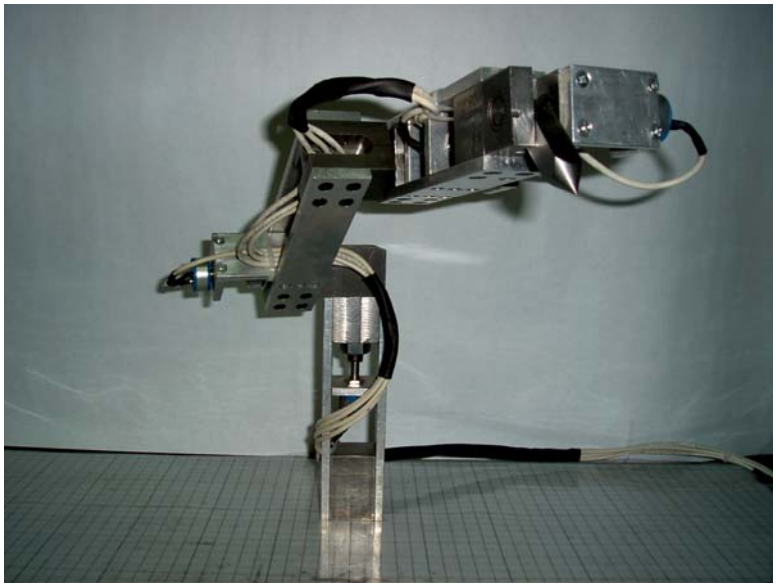


Figura 49. Prueba 6.

Tabla de resultados

Prueba 6		x [mm]	y [mm]	θ[º]	z [mm]
1	Medida física	-80	255	-17.418	271
2	Medida experimental	-99.4	241.6	-22.363	261.5

VIII. Discusión de resultados.

La calibración que se tomo para realizar las pruebas fue la opción A debido a que presenta ligeras desviaciones entre la lectura experimental y la lectura teórica que se espera en el *home*, es decir:

Calibración		x [mm]	y [mm]	θ[º]	z [mm]
1	Medida física	0	317.6	0	175
2	Medida experimental	-1.875	317.405	-0.338	178.76

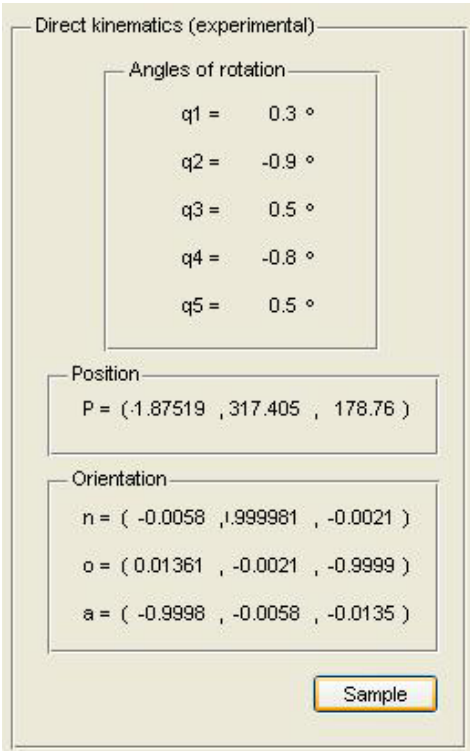


Figura 50. Calibración.

En este punto tambien es importante mencionar que la lectura de los potenciometros en el home debe registrar angulos de rotacion iguales a cero, la diferencia mayor la encontramos en q_2=-0.9º.

Las pruebas 1 a 5 caen dentro de una variación aceptable entre lo que se mide físicamente y lo que se registra, mientras que la prueba 6 tiene una diferencia notable de casi 20mm en la coordenada x, lo que se traduce en un 19.52% de error.

Otro aspecto importante que debemos señalar es que la lectura de los potenciómetros no permanece invariable con el tiempo. Es decir, si nosotros garantizamos que la estructura permanece estática en alguna configuración, por ejemplo recargándola en el banco de apoyo, esperaríamos que la lectura del dispositivo no variase, o que por lo menos la variación sea mínima, sin en cambio nos encontramos que al registrar en múltiples ocasiones la lectura de los potenciómetros en esta condición estática "rigurosa", se presenta una variación de ± 6mm en general para las lecturas de los puntos x, y, z.

IX. Conclusiones.

La manufactura de la estructura del robot no es de ninguna manera algo trivial ni ordinaria. Los ajustes necesarios, la concentricidad entre perno y alojamiento, el paralelismo o la perpendicularidad entre las caras de los elementos, entre otras cosas, demandan una gran habilidad por parte del operador por un lado, sin tener en cuenta la necesidad de contar con maquinaria realmente de precisión y que trabaje correctamente.

Los errores apreciados en las lecturas del punto x, y, z en las pruebas realizadas se pueden explicar desde distintos enfoques:

- Una mala calibración.
- La misma operación del robot con el paso del tiempo, provoca que los potenciómetros se desplacen de su posición inicial. Esto se puede corroborar configurando la estructura en el home nuevamente, se aprecia una variación de 1 a dos grados. A simple vista parece no repercutir mucho aunque como es evidente en la prueba 6 existen variaciones notables (20mm)
- La misma variación en la lectura por parte de los potenciómetros. La variación de voltaje normal en los potenciómetros es de $\pm$ 0.01V de un máximo de 5V
- La diferencias entre el modelo virtual y el modelo físico, provocadas principalmente por una deficiente manufactura.
- Un apriete excesivo, provocando una deformación en el material, que puede repercutir mas cuando tenemos una junta de neopreno de por medio.
- Algún desnivel en el apoyo de la estructura

Las variaciones en las lecturas del programa cuando se calibra mal la estructura son notables por lo que definitivamente será necesario diseñar un nuevo dispositivo de calibración, ya que el que utilizamos no garantiza la repetitibilidad en el funcionamiento correcto del programa, o sea las mediciones físicas, los valores esperados presentan diferencias claras entre lo que se mide físicamente y lo que se registra en la ventana experimental del cuadro de operación (*Kinematic Solver*)

Para un valor de θ, la lectura de la distancia en el eje x y en el eje y varia $\pm$ 6mm como se menciono y explicó anteriormente, esto nos provocara una variación "normal" en cualesquiera de nuestras mediciones dependiendo de la posición fijada.

El uso de Matlab para crear el software Kinematic Solver facilitó en gran medida las cosas ya que es un lenguaje de programación muy accesible en el se pueden crear ambientes

gráficos muy amigables para el usuario. El no contar con experiencia previa con el Matlab se convirtió en una limitante considerable ya que no pudimos crear menús como los deseábamos.

El uso de las tecnologías de National Instrument y MatLab, así como el uso de sensores de alta precisión puede hacer pensar que es una gran ventaja, pero al no contar con el mismo nivel de tecnología en maquinaria para la manufactura de la estructura ErP se presenta un problema: los errores que tenemos son altos ya que al tener tan alta precisión en la mediciones electrónicas, se hacen evidentes las fallas en las piezas que componen la estructura. Otro problema importante es que contamos con instrumentos de medición (mecánica) cuya resolución no se compara con la de los sensores.

X. Bibliografía.

[1] Revista Manufactura. Soluciones integrales para el profesional de planta Articulo Robots. Soluciones a la mano Por: Dino Rozenberg. 2/1/2008. http://www.manufacturaweb.com/buscar.asp?xsl=indice.xsl

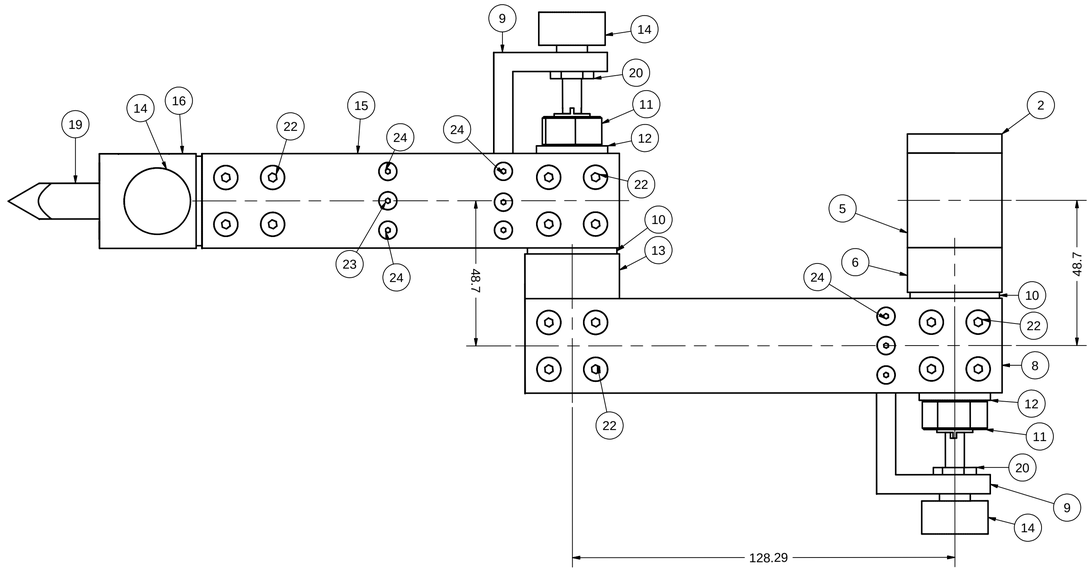
[2] http://www.aeratp.com/documentos/revista/boletin.pdf

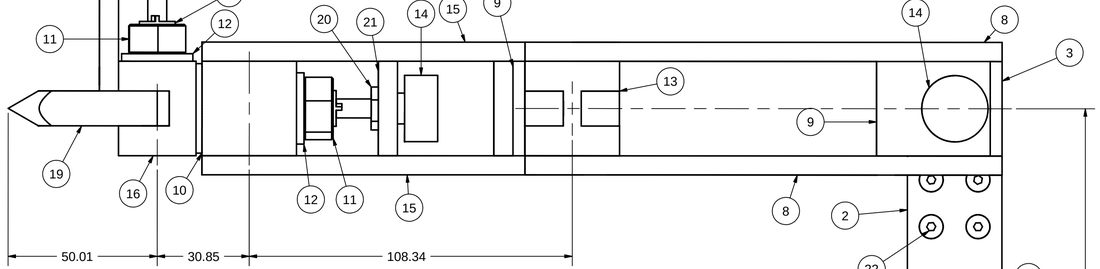
[3] Robótica Industrial. Fundamentos y aplicaciones. Arantxa y Rivas. Año 2000 España. Mc Graw Hill

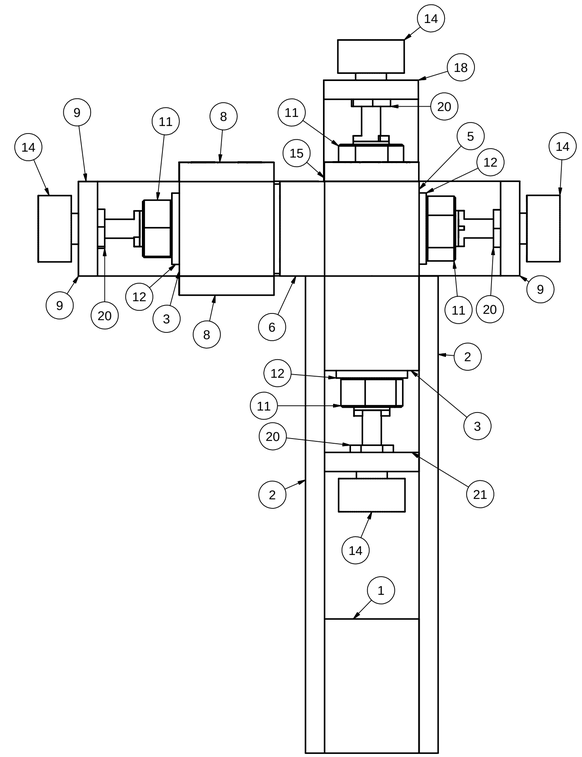
[4] Referencia: Manual SKF. Página 248

X. Anexo 1. – Dibujos de definición.

En este capítulo se incluyen los dibujos de conjunto y de definición de todas las piezas mecánicas que se maquinaron para elaborar la estructura del robot.

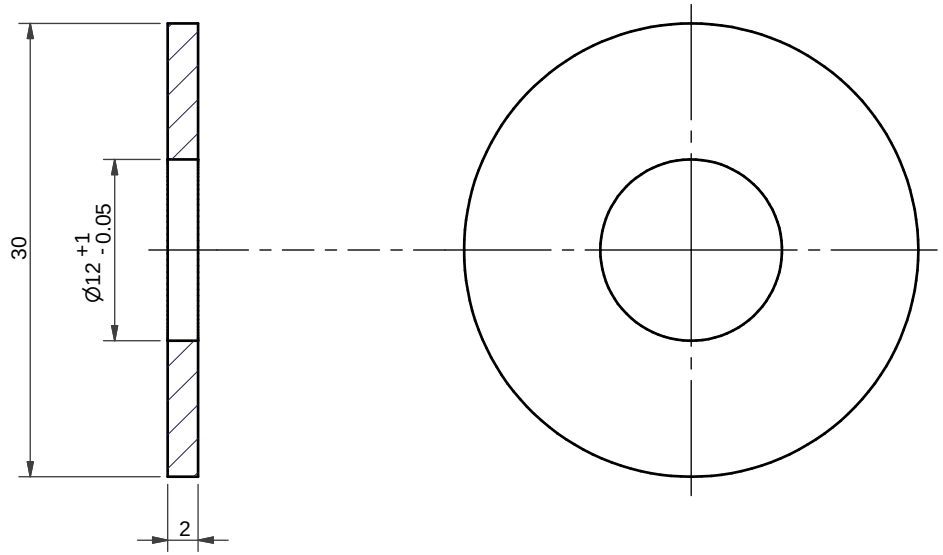


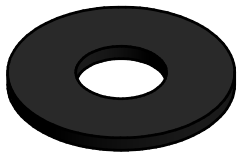




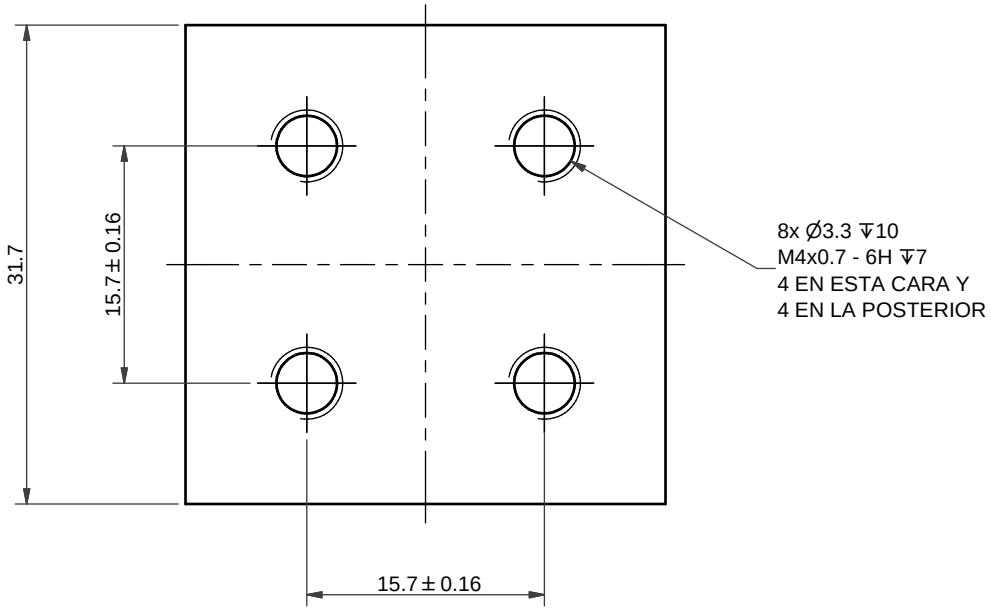
Parts List			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	SOPORTE PRINCIPAL	
2	2	SOLERA L0	
3	4	ALOJAMIENTO	
4	10	RODAMIENTO DE AGUJAS HK1210	
5	1	PERNO 1	
6	1	PERNO 2	
7	5	TONILLO M4x0.7 x 25	
8	2	SOLERA L2	
9	2	BASE 2 y 3	
10	3	JUNTA DE NEOPRENO	
11	5	TUERCA M12 x 1.75	
12	5	ARANDELA DE 1/2	
13	1	PERNO 3	
14	5	POTENCIOMETRO 1 Kohm	
15	2	SOLERA L3	
16	1	PERNO 4	
17	1	PERNO HERRAMIENTA	
18	1	BASE 5	
19	1	HERRAMIENTA FINAL	
20	5	TUERCA DE POTENCIOMETRO	
21	2	BASE 1 y 4	
22	48	TONILLO M4x0.7 x 12	
23	6	TONILLO M3x0.5 x 12	
24	20	TONILLO M3x0.7 x 16	

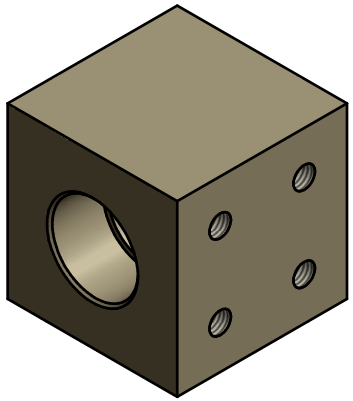
UNIVERSIDAD AUTÓNOMA METROPOLITANA Azcapotzalco Casa abierta al tiempo		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 07-nov-07	DIBUJO DE ENSAMBLE	acot: mm
revisó: G.A.M.		escala: 1:1
material: ACERO 1018 ALUMINIO 3660	dibujo 1 de 16	

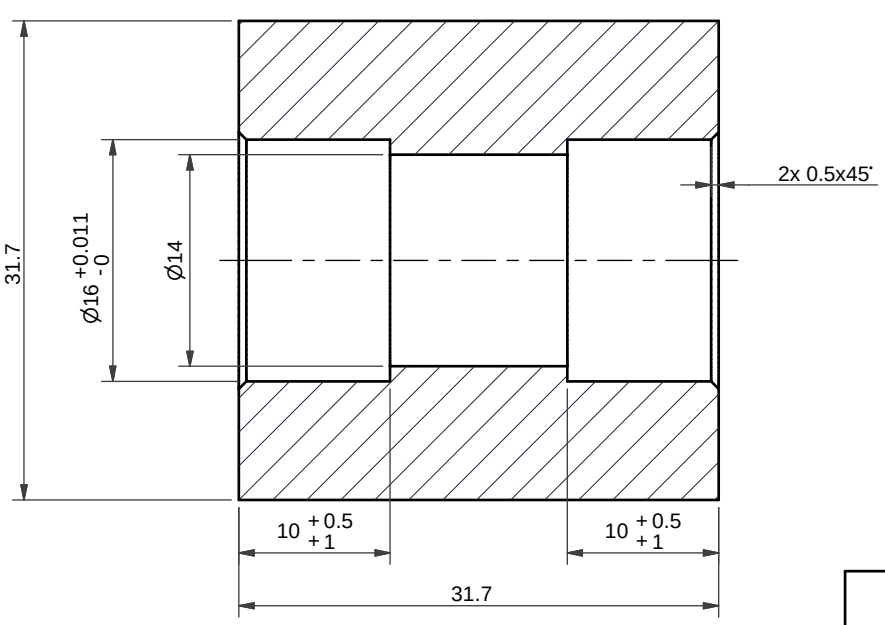


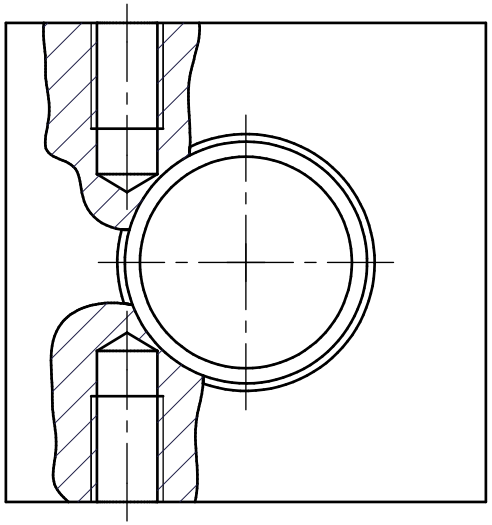


UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 07-nov-07	JUNTA DE NEOPRENO	acot: mm
revisó: G.A.M.		escala: 2:1
material: neopreno	dibujo 2 de 16	

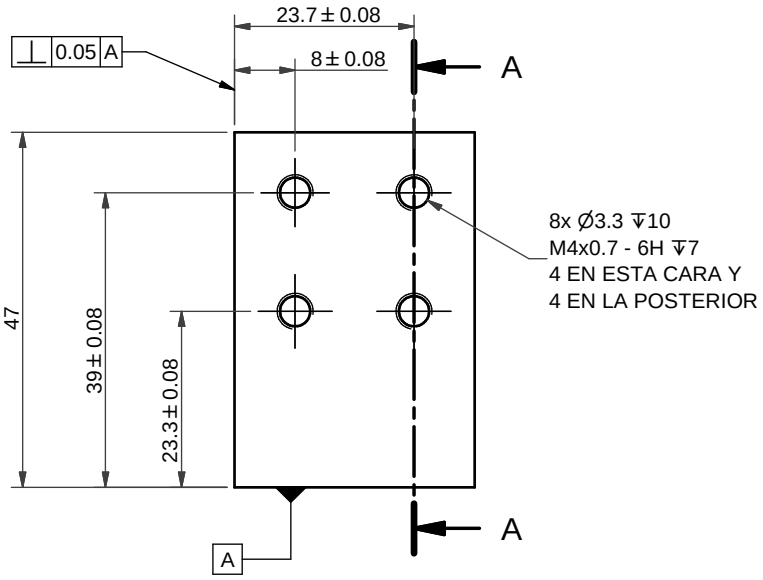


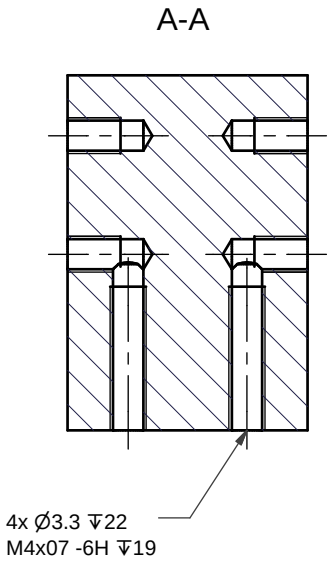


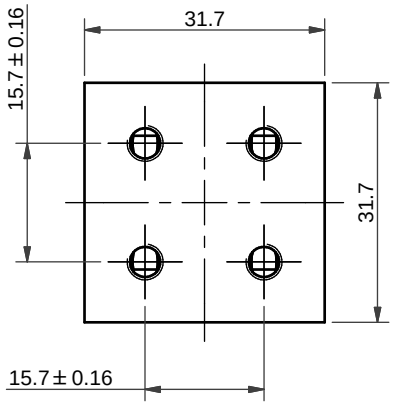


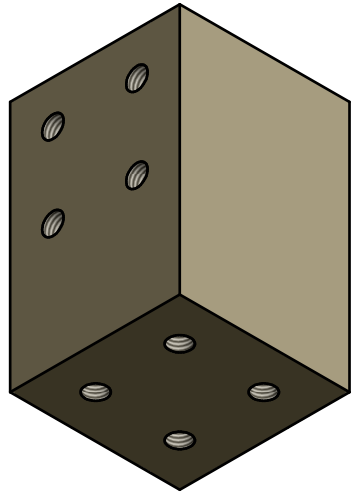


UNIVERSIDAD AUTÓNOMA METROPOLITANA Azcapotzalco Casa abierta al tiempo		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	ALOJAMIENTO	acot: mm
revisó: G.A.M.		escala: 2:1
material: acero 1018	dibujo 3 de 16	

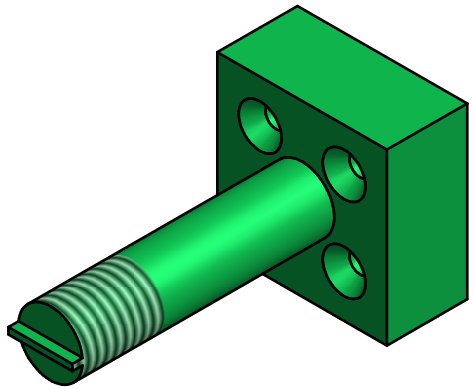


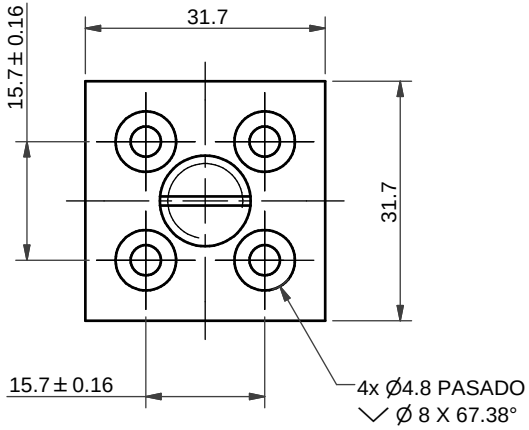


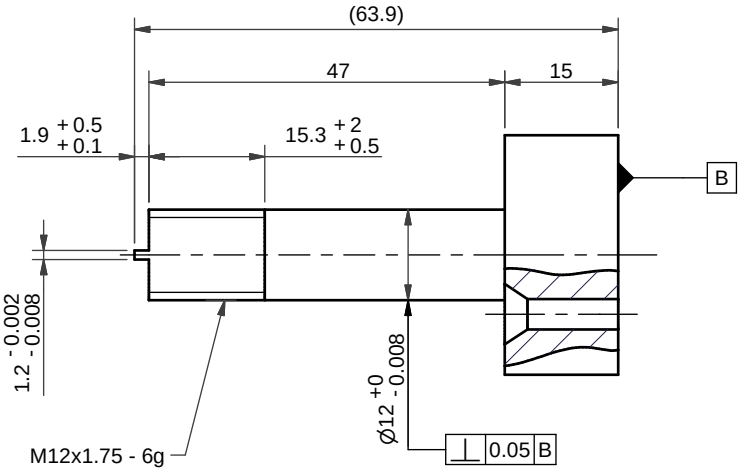




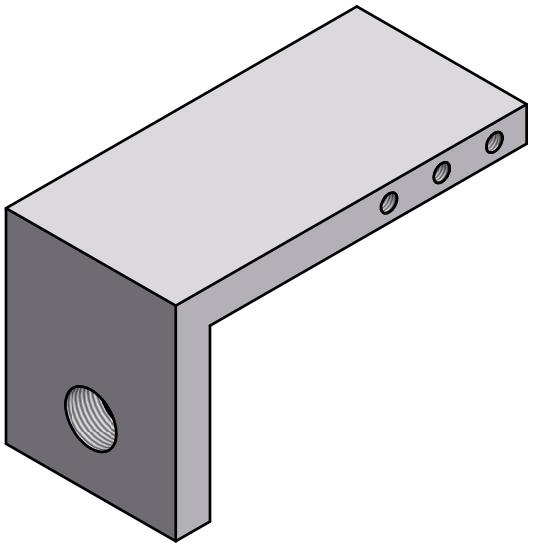
UNIVERSIDAD AUTÓNOMA METROPOLITANA Azcapotzalco Casa abierta al tiempo		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	SOPORTE PRINCIPAL	acot: mm
revisó: G.A.M.		escala: 1:1
material: acero 1018	dibujo 4 de 16	

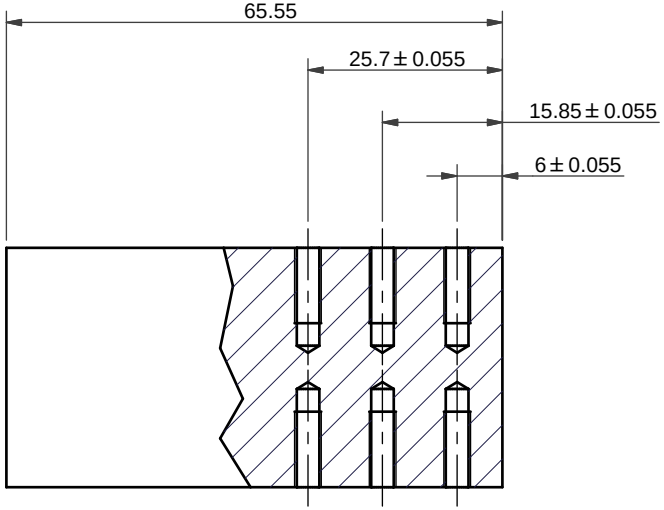


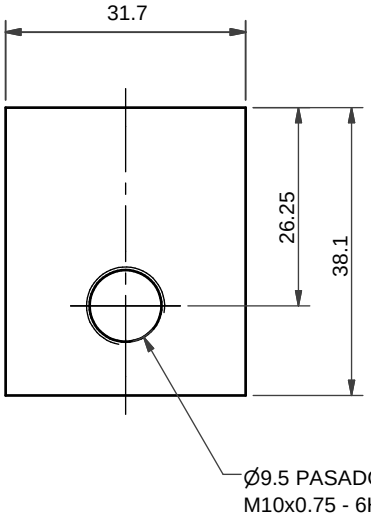


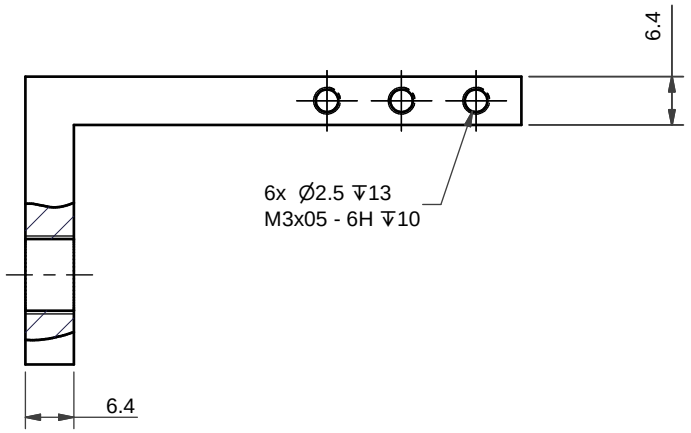


UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	PERNO 2	acot: mm
revisó: G.A.M.		escala: 1:1
material: acero 1018	dibujo 5 de 16	

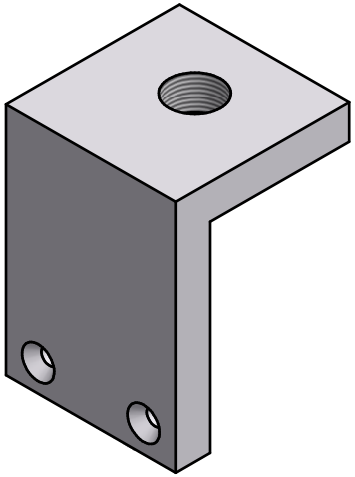


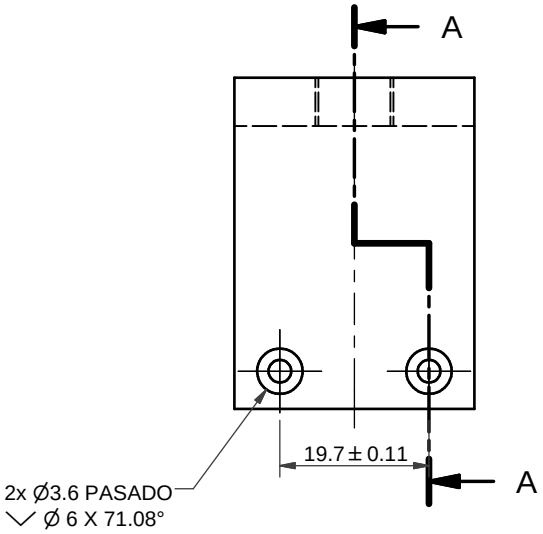


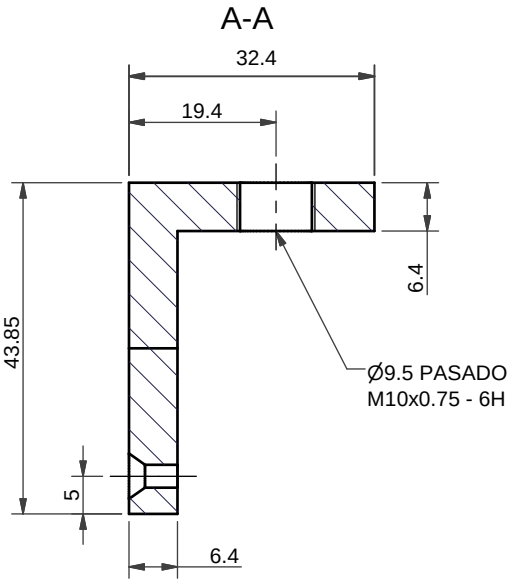




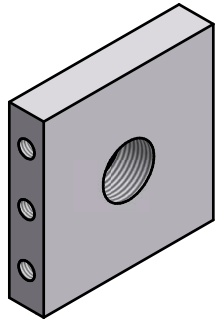
UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	BASE 2 & 3	acot: mm
revisó: G.A.M.		escala: 1:1
material: aluminio 3660	dibujo 6 de 16	

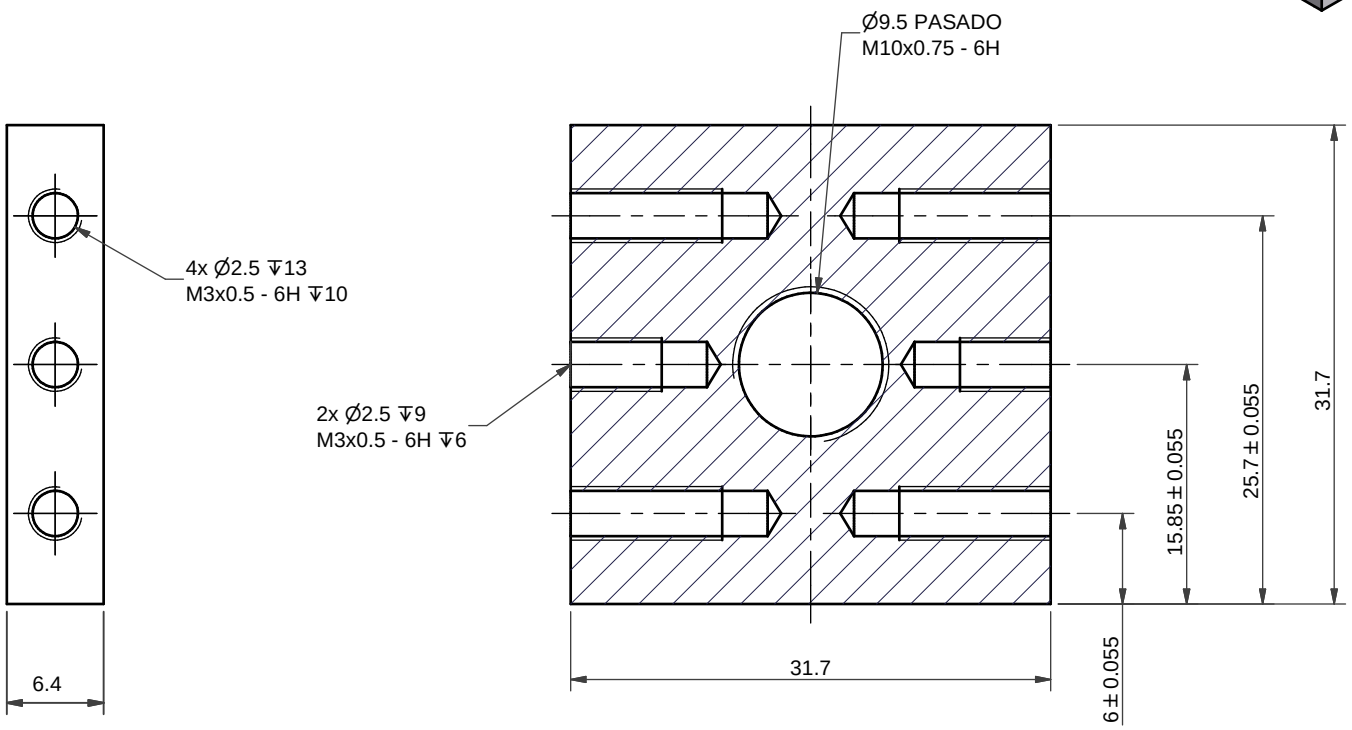




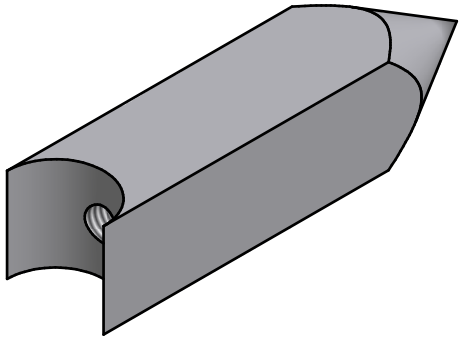


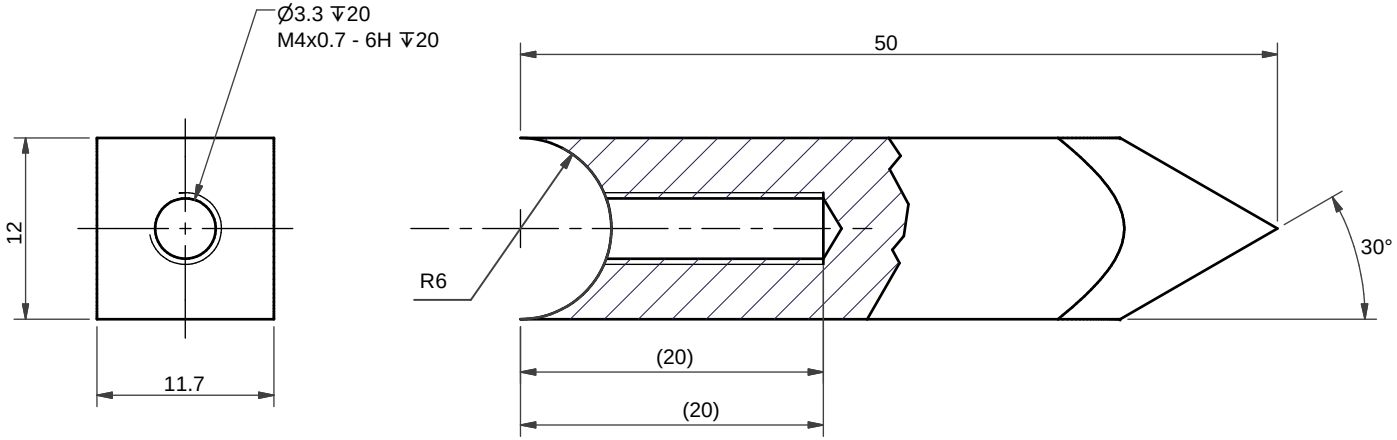
UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	BASE 5	acot: mm
revisó: G.A.M.		escala: 1:1
material: aluminio 3660	dibujo 7 de 16	



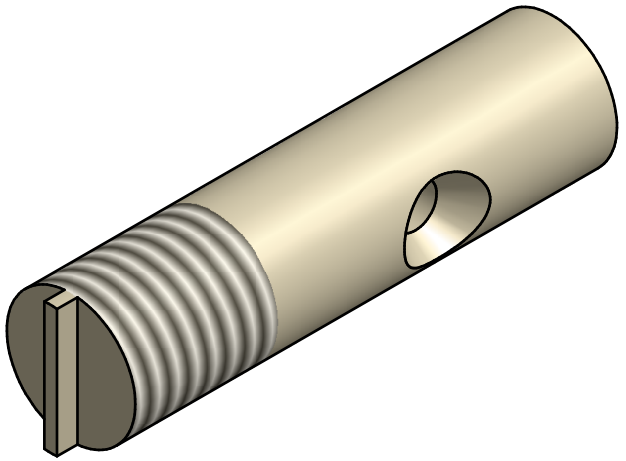


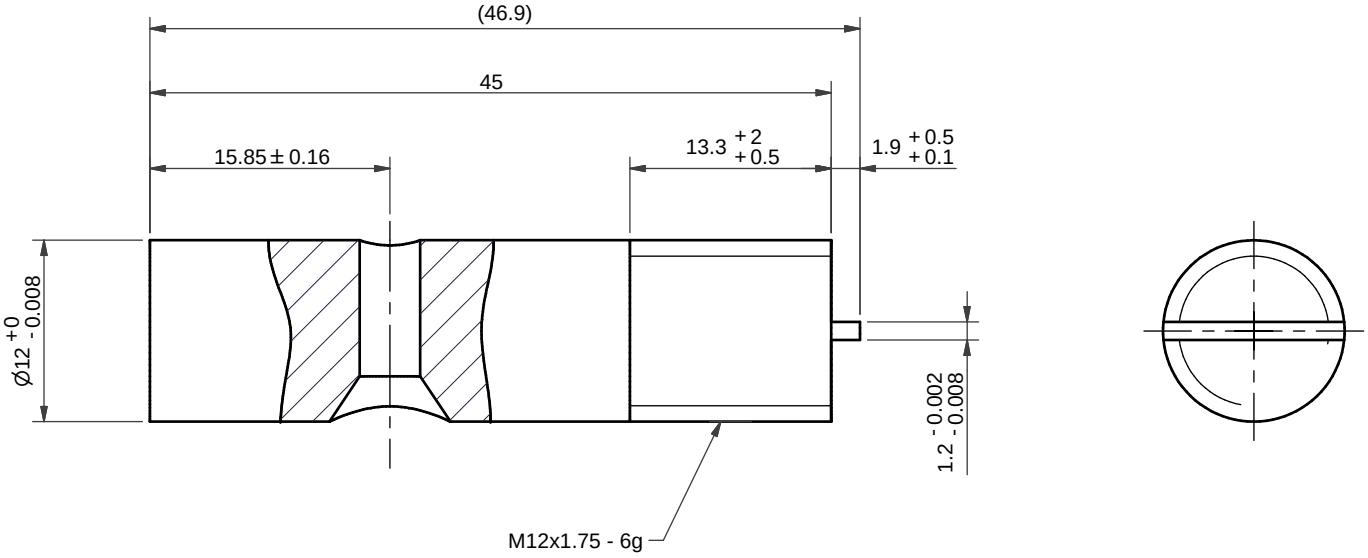
UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	BASE 1 & 4	acot: mm
revisó: G.A.M.		escala: 2:1
material: aluminio 3660	dibujo 8 de 16	



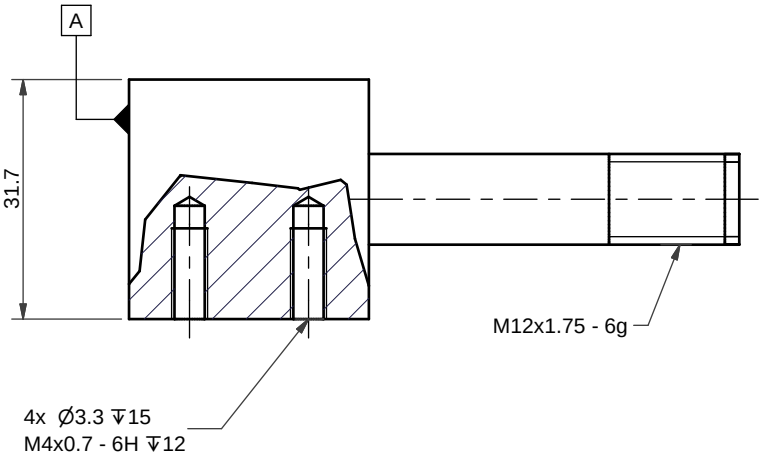


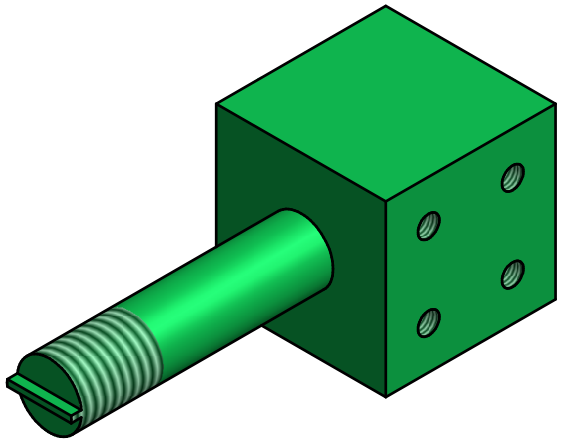
UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	HERRAMIENTA FINAL	acot: mm
revisó: G.A.M.		escala: 2:1
material: acero 1018	dibujo 9 de 16	

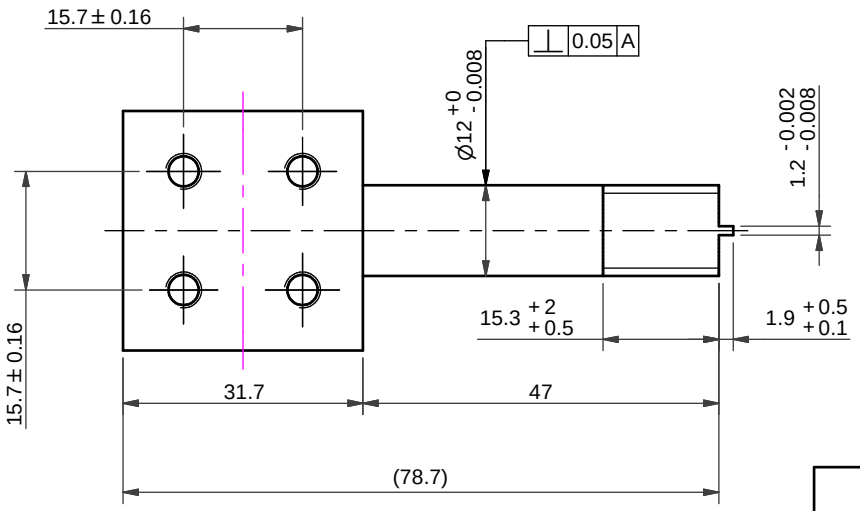


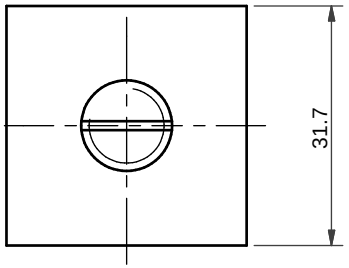


UNIVERSIDAD AUTÓNOMA METROPOLITANA Azcapotzalco Casa abierta al tiempo		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	PERNO HERRAMIENTA	acot: mm
revisó: G.A.M.		escala: 2:1
material: acero 1018	dibujo 10 de 16	

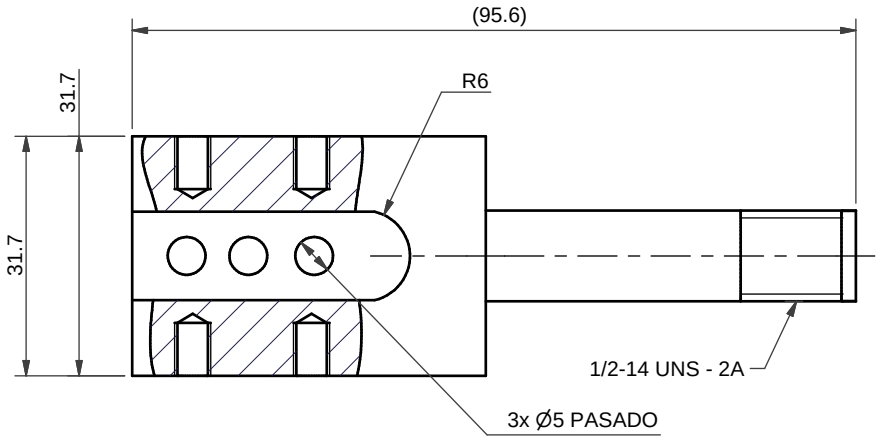


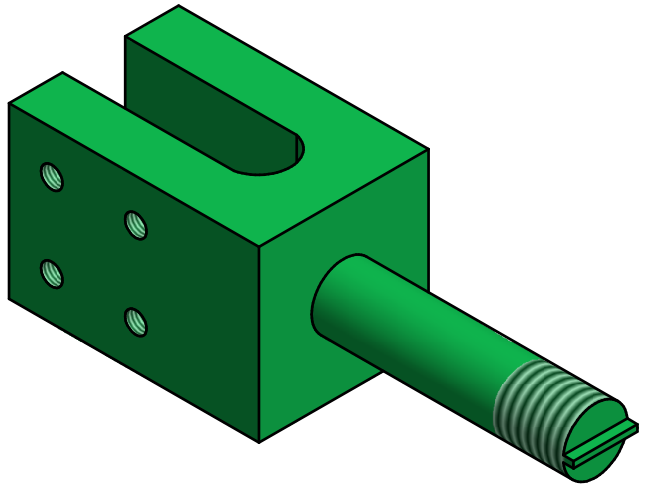


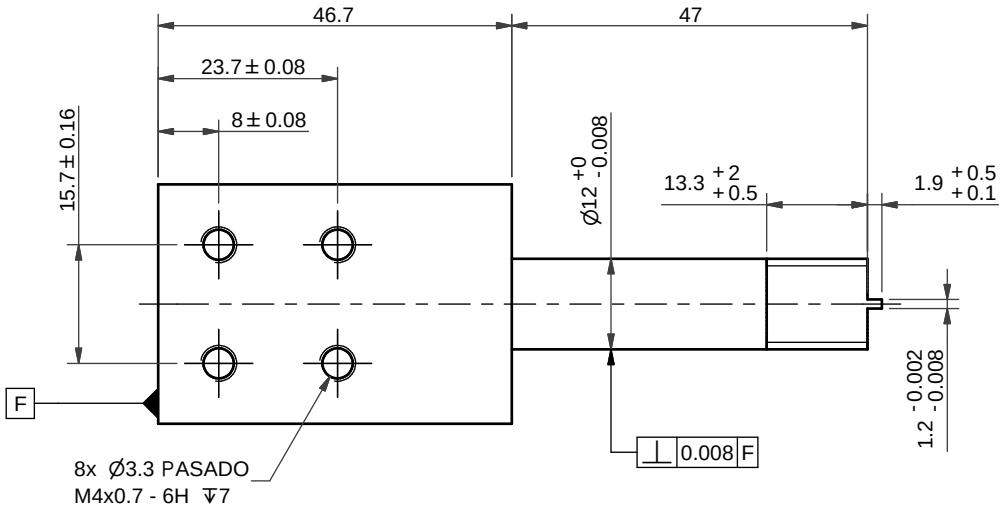


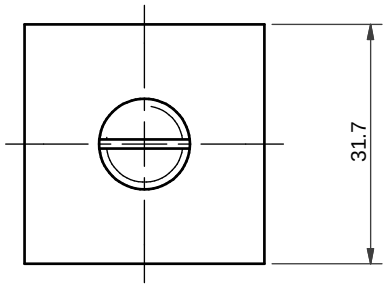


UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	PERNO 1	acot: mm
revisó: G.A.M.		escala: 1:1
material: acero 1018	dibujo 11 de 16	

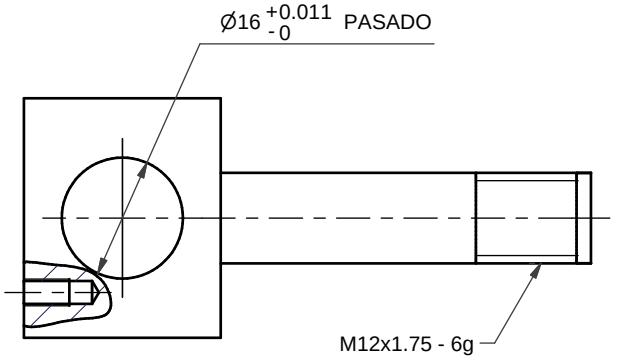


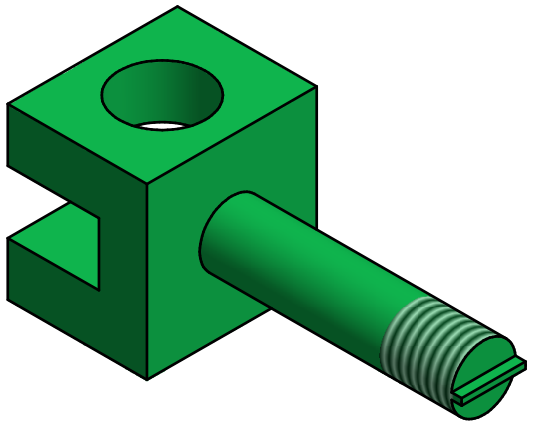


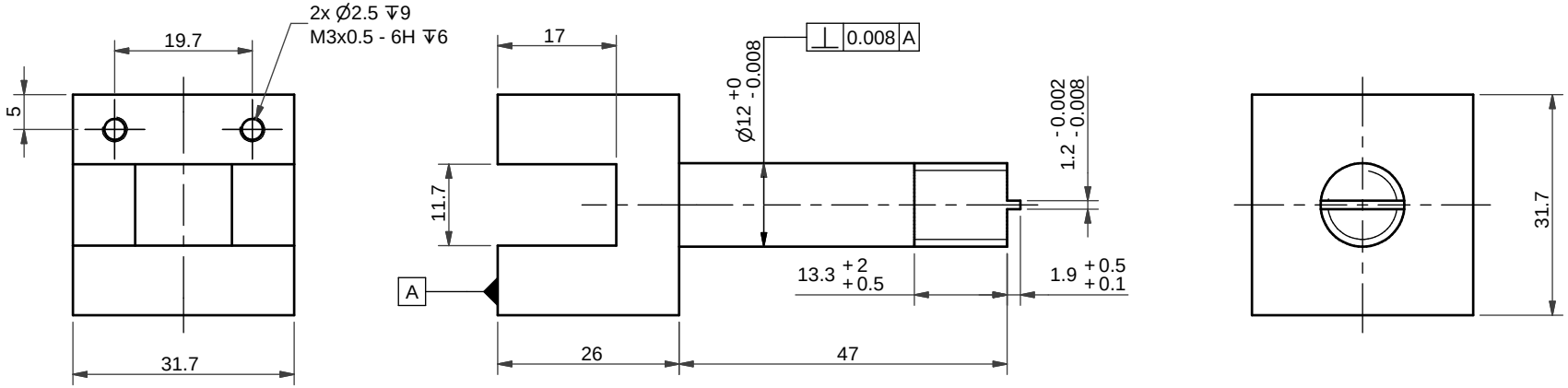




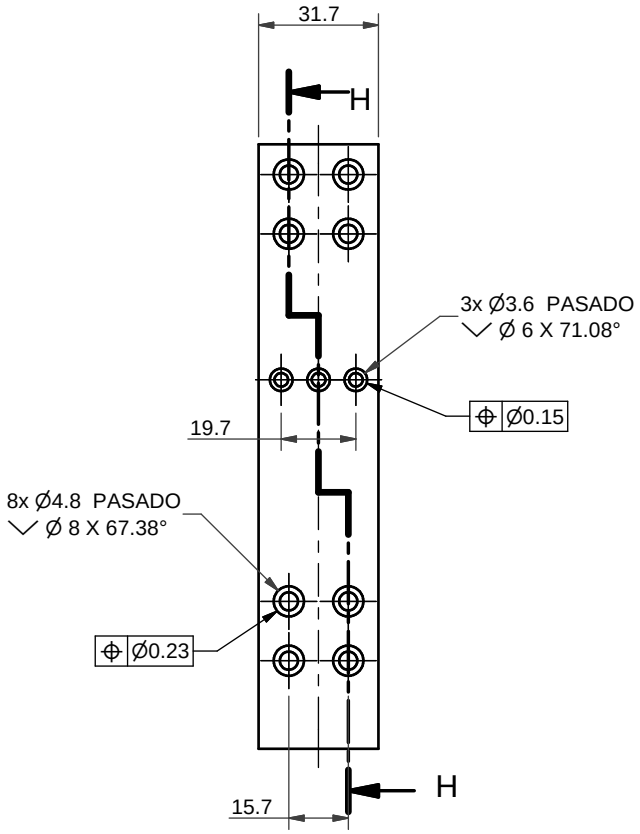
UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	PERNO 3	acot: mm
revisó: G.A.M.		escala: 1:1
material: acero 1018	dibujo 12 de 18	

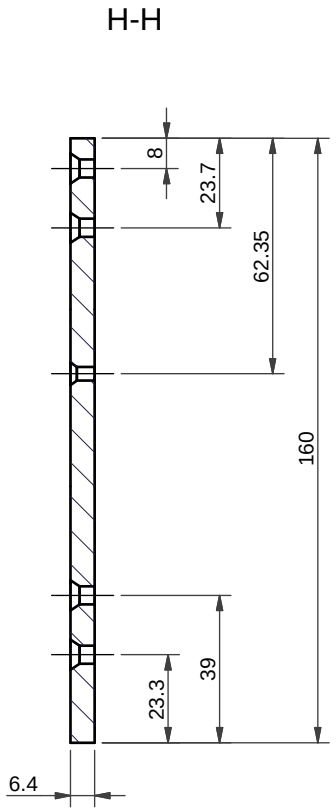


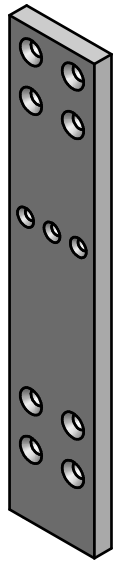




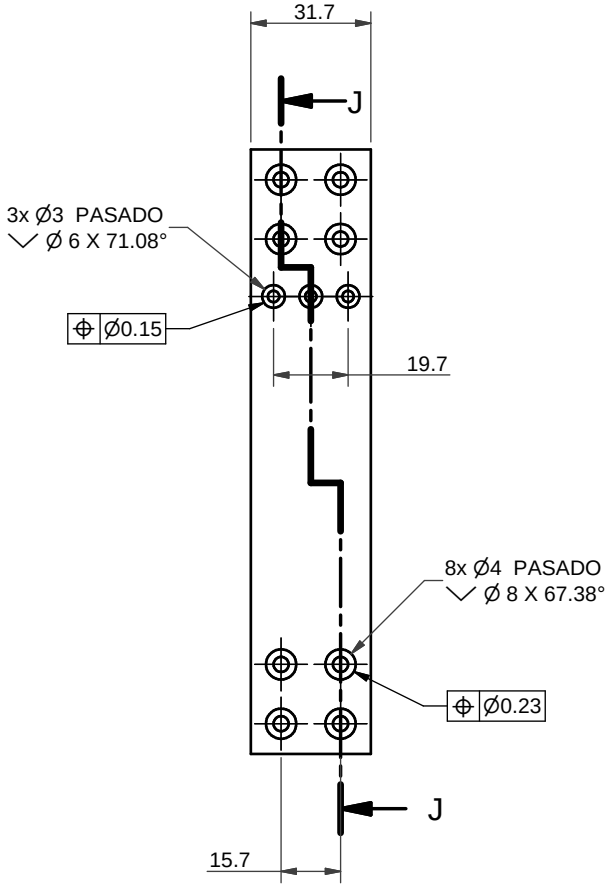
UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	PERNO 4	acot: mm
revisó: G.A.M.		escala: 1:1
material: acero 1018	dibujo 13 de 16	

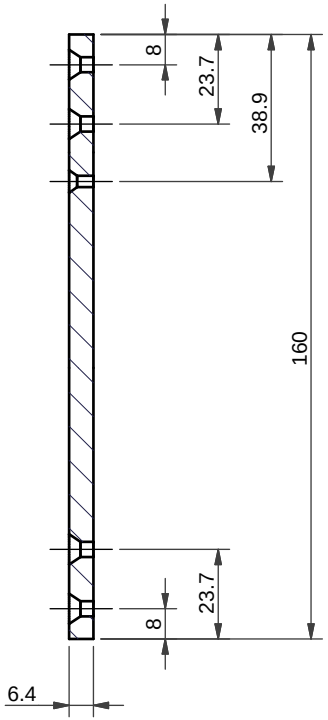


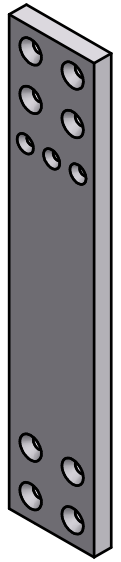




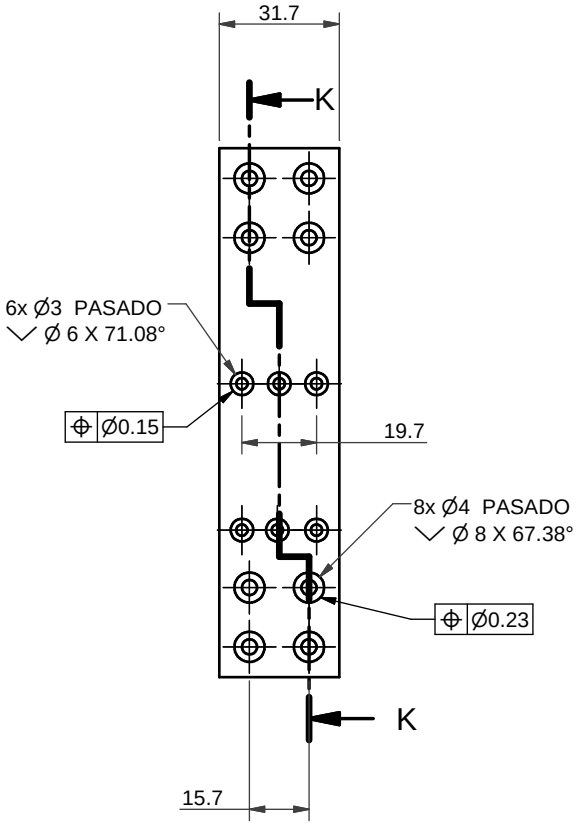
UNIVERSIDAD AUTÓNOMA METROPOLITANA Azcapotzalco Casa abierta al tiempo		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	SOLERA L0	acot: mm
revisó: G.A.M.		escala: 1:2
material: aluminio 3660	dibujo 14 de 16	

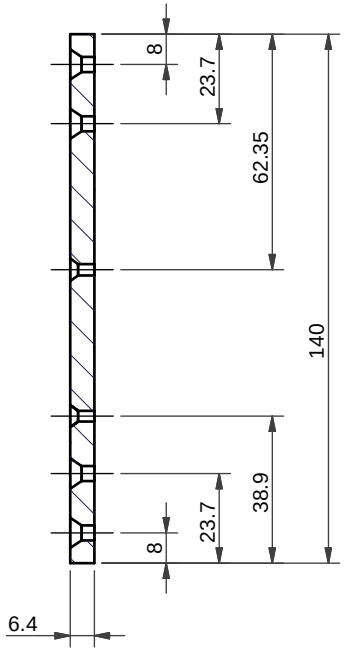


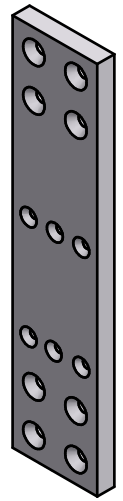




UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	SOLERA L2	acot: mm
revisó: G.A.M.		escala: 1:2
material: aluminio 3660	dibujo 15 de 16	







UNIVERSIDAD AUTÓNOMA METROPOLITANA Casa abierta al tiempo Azcapotzalco		Elaboró: ABREU DÍAZ CAMARGO R.
fecha: 12-nov-07	SOLERA L4	acot: mm
revisó: G.A.M.		escala: 1:2
material: aluminio 3660	dibujo 16 de 16	

X. Anexo 2. – Código de MatLab.

En este capítulo se incluye el código de MatLab elaborado para la interfaz gráfica que registra la posición actual de la estructura del robot y resuelve la cinemática directa e inversa de los puntos solicitados.

```
function varargout = prog_ks(varargin)
% PROG_KS M-file for prog_ks.fig
%      PROG_KS, by itself, creates a new PROG_KS or raises the existing
%      singleton*.
%
%      H = PROG_KS returns the handle to a new PROG_KS or the handle to
%      the existing singleton*.
%
%      PROG_KS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROG_KS.M with the given input arguments.
%
%      PROG_KS('Property','Value',...) creates a new PROG_KS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before prog_ks_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to prog_ks_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help prog_ks

% Last Modified by GUIDE v2.5 22-May-2008 17:34:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @prog_ks_OpeningFcn, ...
                   'gui_OutputFcn',  @prog_ks_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before prog_ks is made visible.
function prog_ks_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to prog_ks (see VARARGIN)

% Choose default command line output for prog_ks
handles.output = hObject;

% NOTA 9: aquí se inicializan las variables (cuyos datos se obtienen de los
%         potenciometros en volts) que sirven para encontrar el "Home".
handles.ca.v1 = 0;
handles.ca.v2 = 0;
handles.ca.v3 = 0;
handles.ca.v4 = 0;
handles.ca.v5 = 0;
%--------------------------------------------------------------------------
% Update handles structure
guidata(hObject, handles);

initialize_gui(hObject, handles, 1);

% UIWAIT makes prog_ks wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = prog_ks_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%_____________________________________________________________________________
% NOTA 7: en esta sección se leen la posición y la orientación  para
% calcular la CI.
%--------------------------------------------------------------------------
function nx_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to nx_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nx_ci_e as text
%        str2double(get(hObject,'String')) returns contents of nx_ci_e as a double
nx_ci_e = str2double(get(hObject, 'String'));
if isnan(nx_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif nx_ci_e>1 | nx_ci_e<-1
```

```
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new nx_ci_e value
handles.ci.nx_ci_e = nx_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function nx_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nx_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function ny_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to ny_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ny_ci_e as text
%        str2double(get(hObject,'String')) returns contents of ny_ci_e as a double
ny_ci_e = str2double(get(hObject, 'String'));
if isnan(ny_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif ny_ci_e>1 | ny_ci_e<-1
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new ny_ci_e value
handles.ci.ny_ci_e = ny_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function ny_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ny_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
```

```
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function nz_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to nz_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nz_ci_e as text
%        str2double(get(hObject,'String')) returns contents of nz_ci_e as a double
nz_ci_e = str2double(get(hObject, 'String'));
if isnan(nz_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif nz_ci_e>1 | nz_ci_e<-1
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new nz_ci_e value
handles.ci.nz_ci_e = nz_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function nz_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nz_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function ox_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to ox_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ox_ci_e as text
%        str2double(get(hObject,'String')) returns contents of ox_ci_e as a double
ox_ci_e = str2double(get(hObject, 'String'));
if isnan(ox_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
```

```
elseif ox_ci_e>1 | ox_ci_e<-1
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new ox_ci_e value
handles.ci.ox_ci_e = ox_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function ox_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ox_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function oy_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to oy_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of oy_ci_e as text
%        str2double(get(hObject,'String')) returns contents of oy_ci_e as a double
oy_ci_e = str2double(get(hObject, 'String'));
if isnan(oy_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif oy_ci_e>1 | oy_ci_e<-1
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new oy_ci_e value
handles.ci.oy_ci_e = oy_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function oy_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to oy_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function oz_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to oz_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of oz_ci_e as text
%        str2double(get(hObject,'String')) returns contents of oz_ci_e as a double
oz_ci_e = str2double(get(hObject, 'String'));
if isnan(oz_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif oz_ci_e>1 | oz_ci_e<-1
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new oz_ci_e value
handles.ci.oz_ci_e = oz_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function oz_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to oz_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function ax_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to ax_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ax_ci_e as text
%        str2double(get(hObject,'String')) returns contents of ax_ci_e as a double
ax_ci_e = str2double(get(hObject, 'String'));
if isnan(ax_ci_e)
    set(hObject, 'String', 0);
```

```
    errordlg('Input must be a number','Error');
elseif ax_ci_e>1 | ax_ci_e<-1
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new ax_ci_e value
handles.ci.ax_ci_e = ax_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function ax_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ax_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function ay_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to ay_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ay_ci_e as text
%        str2double(get(hObject,'String')) returns contents of ay_ci_e as a double
ay_ci_e = str2double(get(hObject, 'String'));
if isnan(ay_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif ay_ci_e>1 | ay_ci_e<-1
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new ay_ci_e value
handles.ci.ay_ci_e = ay_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function ay_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ay_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function az_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to az_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of az_ci_e as text
%        str2double(get(hObject,'String')) returns contents of az_ci_e as a double
az_ci_e = str2double(get(hObject, 'String'));
if isnan(az_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif az_ci_e>1 | az_ci_e<-1
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new az_ci_e value
handles.ci.az_ci_e = az_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function az_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to az_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function px_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to px_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of px_ci_e as text
%        str2double(get(hObject,'String')) returns contents of px_ci_e as a double
px_ci_e = str2double(get(hObject, 'String'));
if isnan(px_ci_e)
```

```
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif px_ci_e>300 | px_ci_e<-300
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new px_ci_e value
handles.ci.px_ci_e = px_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function px_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to px_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function py_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to py_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of py_ci_e as text
%        str2double(get(hObject,'String')) returns contents of py_ci_e as a double
py_ci_e = str2double(get(hObject, 'String'));
if isnan(py_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif py_ci_e>300 | py_ci_e<-52
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new py_ci_e value
handles.ci.py_ci_e = py_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function py_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to py_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function pz_ci_e_Callback(hObject, eventdata, handles)
% hObject    handle to pz_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pz_ci_e as text
%        str2double(get(hObject,'String')) returns contents of pz_ci_e as a double
pz_ci_e = str2double(get(hObject, 'String'));
if isnan(pz_ci_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif pz_ci_e>460 | pz_ci_e<0
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new pz_ci_e value
handles.ci.pz_ci_e = pz_ci_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function pz_ci_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pz_ci_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% fin de la lectura la posición y orientación-----------------------------
%_________________________________________________________________________


%_________________________________________________________________________
% NOTA 1: en esta sección se leen los ángulos de giro para la CD teórica
% q1, q2, q3, q4, q5
%-------------------------------------------------------------------------
function q1_cdt_e_Callback(hObject, eventdata, handles)
% hObject    handle to q1_cdt_e (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of q1_cdt_e as text
%        str2double(get(hObject,'String')) returns contents of q1_cdt_e as a double
q1_cdt_e = str2double(get(hObject, 'String'));
if isnan(q1_cdt_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif q1_cdt_e>95 | q1_cdt_e<-95
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new q1_cdt_e value
handles.cdt.q1_cdt_e = q1_cdt_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function q1_cdt_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to q1_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function q2_cdt_e_Callback(hObject, eventdata, handles)
% hObject    handle to q2_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of q2_cdt_e as text
%        str2double(get(hObject,'String')) returns contents of q2_cdt_e as a double
q2_cdt_e = str2double(get(hObject, 'String'));
if isnan(q2_cdt_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif q2_cdt_e>95 | q2_cdt_e<-95
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new q2_cdt_e value
handles.cdt.q2_cdt_e = q2_cdt_e;
guidata(hObject,handles)
```

```
% --- Executes during object creation, after setting all properties.
function q2_cdt_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to q2_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function q3_cdt_e_Callback(hObject, eventdata, handles)
% hObject    handle to q3_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of q3_cdt_e as text
%        str2double(get(hObject,'String')) returns contents of q3_cdt_e as a double
q3_cdt_e = str2double(get(hObject, 'String'));
if isnan(q3_cdt_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif q3_cdt_e>170 | q3_cdt_e<-170
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new q3_cdt_e value
handles.cdt.q3_cdt_e = q3_cdt_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function q3_cdt_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to q3_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function q4_cdt_e_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to q4_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of q4_cdt_e as text
%        str2double(get(hObject,'String')) returns contents of q4_cdt_e as a double
q4_cdt_e = str2double(get(hObject, 'String'));
if isnan(q4_cdt_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif q4_cdt_e>170 | q4_cdt_e<-170
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new q4_cdt_e value
handles.cdt.q4_cdt_e = q4_cdt_e;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function q4_cdt_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to q4_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function q5_cdt_e_Callback(hObject, eventdata, handles)
% hObject    handle to q5_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of q5_cdt_e as text
%        str2double(get(hObject,'String')) returns contents of q5_cdt_e as a double
q5_cdt_e = str2double(get(hObject, 'String'));
if isnan(q5_cdt_e)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
elseif q5_cdt_e>170 | q5_cdt_e<-170
    set(hObject, 'String', 0);
    errordlg('Input must be within the range','Error');
end

% Save the new q5_cdt_e value
handles.cdt.q5_cdt_e = q5_cdt_e;
```

```
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function q5_cdt_e_CreateFcn(hObject, eventdata, handles)
% hObject    handle to q5_cdt_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% fin de la lectura de los ángulos de giro para la CD teórica
%q1, q2, q3, q4, q5-------------------------------------------------------
%_____________________________________________________________________________



%_____________________________________________________________________________
%NOTA 8: en esta sección se hace el calculo para resolver la CI con datos
%de entrada  teóricos. Estos resultados se obtienen mediante las ecuaciones
%del método geométrico.
%NOTA 8.1: el calculo de la CI se inicia cuando se da click este botón
%"Calculate".
%-----------------------------------------------------------------------------
% --- Executes on button press in calcular_ci.
function calcular_ci_Callback(hObject, eventdata, handles)
% hObject    handle to calcular_ci (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% valores reales de los eslabones de la estructura
% ¡¡¡¡¡cambiar si es necesario!!!!!!
l0 = 175.85; %160;
l1 = 48.7; %30;
l2 = 128.3; %160;
l3 = -48.7; %-30;
l4 = 139.13; %100;
l5 = 50; %40;

%¡¡¡¡¡¡¡¡¡cambiar el valor del largo maximo del brazo(300)!!!!!!!!!!!!!
pxmax_py = sqrt(300^2 - handles.ci.py_ci_e^2);  %valor max que puede alcanzar px con un↙
valor dado de py
pxmax_pz = sqrt(300^2 - (handles.ci.pz_ci_e-l0)^2);  %valor max que puede alcanzar px con↙
un valor dado de pz
pymax_px = sqrt(300^2 - handles.ci.px_ci_e^2);  %valor max que puede alcanzar py con un↙
valor dado de px
pymax_pz = sqrt(300^2 - (handles.ci.pz_ci_e-l0)^2);  %valor max que puede alcanzar py con↙
un valor dado de pz
```

```
if handles.ci.px_ci_e>pxmax_py | handles.ci.px_ci_e<-pxmax_py
    errordlg('Invalid configuration1','Error');
elseif handles.ci.px_ci_e>pxmax_pz | handles.ci.px_ci_e<-pxmax_pz
    errordlg('Invalid configuration2','Error');
elseif handles.ci.py_ci_e>pymax_px | handles.ci.py_ci_e<-pymax_px
    errordlg('Invalid configuration3','Error');
elseif handles.ci.py_ci_e>pymax_pz | handles.ci.py_ci_e<-pymax_pz
    errordlg('Invalid configuration4','Error');
else
%Rp = (handles.ci.px_ci_e, handles.ci.py_ci_e, handles.ci.pz_ci_e)
Rmx = handles.ci.px_ci_e - l5*handles.ci.nx_ci_e;
Rmy = handles.ci.py_ci_e - l5*handles.ci.ny_ci_e;
Rmz = handles.ci.pz_ci_e - l5*handles.ci.nz_ci_e;

nx = handles.ci.nx_ci_e;
ny = handles.ci.ny_ci_e;
nz = handles.ci.nz_ci_e;

ox = handles.ci.ox_ci_e;
oy = handles.ci.oy_ci_e;
oz = handles.ci.oz_ci_e;

ax = handles.ci.ax_ci_e;
ay = handles.ci.ay_ci_e;
az = handles.ci.az_ci_e;

%calculos auxiliares-------------
r1 = l0 - Rmz;
Rmxy = sqrt(Rmx^2+Rmy^2);
r2 = sqrt((Rmxy^2)+(r1^2));

%calculo de q1-------------------
q1 = atan2(-Rmx,Rmy);
% if Rmx<0
%     q1 = atan2(-Rmx,Rmy);
% else
%     %q1 = 2*pi+(atan2(-Rmx,Rmy));
% end
q1_ci_s = r2g(q1);
if q1_ci_s<0.0001 & q1_ci_s>-0.0001
    q1_ci_s=0;
end
set(handles.q1_ci_s, 'String', q1_ci_s);

%calculo de q2-------------------
alfa = atan2(Rmxy,r1);
cosgama = (((l2^2)+(r2^2)-(l4^2))/(2*l2*r2));
c1 = sqrt(1-cosgama^2);
c2 = cosgama;
gama = atan2(c1,c2);
```

```
q2 = (pi/2)-(alfa+gama);

q2_ci_s = r2g(q2);
if q2_ci_s<0.0001 & q2_ci_s>-0.0001
    q2_ci_s=0;
end
set(handles.q2_ci_s, 'String', q2_ci_s);

%cálculo de q3-------------------
cosbeta = (((l4^2)+(l2^2)-(r2^2))/(2*l4*l2));
cc1 = sqrt(1-cosbeta^2);
cc2 = cosbeta;
beta = atan2(cc1,cc2);

q3 = pi-beta;

q3_ci_s = r2g(q3);
if q3_ci_s<0.0001 & q3_ci_s>-0.0001
    q3_ci_s=0;
end
set(handles.q3_ci_s, 'String', q3_ci_s);

%cálculo de q4-------------------
cosq4 = (-ax*cos(q1)-ay*sin(q1));
sinq4 = -((sin(q1)*cos(q2)*sin(q3)+sin(q1)*sin(q2)*cos(q3))*ax+(-cos(q1)*cos(q2)*sin(q3)-↙
cos(q1)*sin(q2)*cos(q3))*ay+(sin(q2)*sin(q3)-cos(q2)*cos(q3))*az);

q4 = atan2(sinq4,cosq4);

q4_ci_s = r2g(q4);
if q4_ci_s<0.0001 & q4_ci_s>-0.0001
    q4_ci_s=0;
end
set(handles.q4_ci_s, 'String', q4_ci_s);

%cálculo de q5-------------------
cosq5 = ((-sin(q1) *cos(q2)* cos(q3) + sin(q1) *sin(q2) *sin(q3)) *nx + (cos(q1) *cos(q2)*↙
cos(q3) - cos(q1)* sin(q2) *sin(q3)) *ny + (-sin(q2)* cos(q3) - cos(q2)* sin(q3))* nz);
sinq5=-((-sin(q1) *cos(q2) *cos(q3) + sin(q1)* sin(q2)* sin(q3))* ox+ (cos(q1)* cos(q2)*↙
cos(q3) - cos(q1) *sin(q2)* sin(q3))* oy + (-sin(q2)* cos(q3) - cos(q2) *sin(q3)) *oz);

q5 = atan2(sinq5,cosq5);

q5_ci_s = r2g(q5);
if q5_ci_s<0.0001 & q5_ci_s>-0.0001
    q5_ci_s=0;
end
ttt = vpa(q5_ci_s,2)
%set(handles.q5_ci_s, 'String %.2f', q5_ci_s);
set(handles.q5_ci_s, 'String %.2f', ttt);
end
%--------------------------------------------------------------------------
```

```
%_____________________________________________________________________________



%_____________________________________________________________________________
% NOTA 2: en esta sección se realiza el calculo de la posición y la
% orientación de efector final por medio del metodo de Denavit-Hartemberg
% el calculo se inicia cuando se da click en el boton "Calculate"
%-----------------------------------------------------------------------------
% --- Executes on button press in calcular_cdt.
function calcular_cdt_Callback(hObject, eventdata, handles)
% hObject    handle to calcular_cdt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

l0 = 175.85; %160;
l1 = 48.7; %30;
l2 = 128.3; %160;
l3 = -48.7; %-30;
l4 = 139.13; %100;
l5 = 50; %40;

T01 = denavit( g2r(handles.cdt.q1_cdt_e)+pi/2,  l0,   0, -pi/2 );
T12 = denavit( g2r(handles.cdt.q2_cdt_e),       l1,  l2,     0 );
T23 = denavit( g2r(handles.cdt.q3_cdt_e)+pi/2,  l3,   0,  pi/2 );
T34 = denavit( g2r(handles.cdt.q4_cdt_e),       l4,   0, -pi/2 );
T45 = denavit( g2r(handles.cdt.q5_cdt_e)-pi/2,   0,  l5,     0 );

T05 = T01 * T12 * T23 * T34 * T45;

%cálculo de la posición-----------------------
px_cdt_s = T05(1,4);
if px_cdt_s<0.0001 & px_cdt_s>-0.0001
    px_cdt_s=0;
end
set(handles.px_cdt_s, 'String', px_cdt_s);

py_cdt_s = T05(2,4);
if py_cdt_s<0.0001 & py_cdt_s>-0.0001
    py_cdt_s=0;
end
set(handles.py_cdt_s, 'String', py_cdt_s);

pz_cdt_s = T05(3,4);
if pz_cdt_s<0.0001 & pz_cdt_s>-0.0001
    pz_cdt_s=0;
end
set(handles.pz_cdt_s, 'String', pz_cdt_s);

%cálculo de la orientación (parmetro "n")-----
%   "nx"
nx_cdt_s = T05(1,1);
```

```
if nx_cdt_s<0.0001 & nx_cdt_s>-0.0001
    nx_cdt_s=0;
end
set(handles.nx_cdt_s, 'String', nx_cdt_s);
%   "ny"
ny_cdt_s = T05(2,1);
if ny_cdt_s<0.0001 & ny_cdt_s>-0.0001
    ny_cdt_s=0;
end
set(handles.ny_cdt_s, 'String', ny_cdt_s);
%   "nz"
nz_cdt_s = T05(3,1);
if nz_cdt_s<0.0001 & nz_cdt_s>-0.0001
    nz_cdt_s=0;
end
set(handles.nz_cdt_s, 'String', nz_cdt_s);

%cálculo de la orientación (parmetro "o")-----
%   "ox"
ox_cdt_s = T05(1,2);
if ox_cdt_s<0.0001 & ox_cdt_s>-0.0001
    ox_cdt_s=0;
end
set(handles.ox_cdt_s, 'String', ox_cdt_s);
%   "oy"
oy_cdt_s = T05(2,2);
if oy_cdt_s<0.0001 & oy_cdt_s>-0.0001
    oy_cdt_s=0;
end
set(handles.oy_cdt_s, 'String', oy_cdt_s);
%   "oz"
oz_cdt_s = T05(3,2);
if oz_cdt_s<0.0001 & oz_cdt_s>-0.0001
    oz_cdt_s=0;
end
set(handles.oz_cdt_s, 'String', oz_cdt_s);

%cálculo de la orientacion (parmetro "a")-----
%   "ax"
ax_cdt_s = T05(1,3);
if ax_cdt_s<0.0001 & ax_cdt_s>-0.0001
    ax_cdt_s=0;
end
set(handles.ax_cdt_s, 'String', ax_cdt_s);
%   "ay"
ay_cdt_s = T05(2,3);
if ay_cdt_s<0.0001 & ay_cdt_s>-0.0001
    ay_cdt_s=0;
end
set(handles.ay_cdt_s, 'String', ay_cdt_s);
%   "az"
az_cdt_s = T05(3,3);
```

```
if az_cdt_s<0.0001 & az_cdt_s>-0.0001
    az_cdt_s=0;
end
set(handles.az_cdt_s, 'String', az_cdt_s);
%aquí finaliza el calculo de la posición y la orientación de efector final
%de la CD teorica-----------------------------------------------------------
%_____________________________________________________________________________



%_____________________________________________________________________________
% NOTA 3: en esta sección se leen los datos directamente de los censores.
% Los datos son leídos en volts y se transformados en grados.
% NOTA 4: en esta sección tambien se realiza el calculo de la posición y la
% orientación de efector final por medio del metodo de Denavit-Hartemberg
% NOTA 4.1: la lectura de los datos de los censores y el cálculo de la CD
% experimental se inicia cuando se da click este botón (Sample).
% Es muy importante haber hecho una buena calibración del brazo antes de
% empezar a obtener muestras.
%-----------------------------------------------------------------------------
% --- Executes on button press in sample_cde.
function sample_cde_Callback(hObject, eventdata, handles)
% hObject    handle to sample_cde (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai0', 0, 5, 2, 5, -1);
%data=1.25;
vl1 = data(1);% + 1.4
q1_cde_e = ((vl1 - handles.ca.v1)*(handles.ca.f1));
%q1_cde_e = 0; % ¡¡¡¡¡¡se debe sustituir este valor por el que se obtenga del↙
potenciometro correspondiente!!!!!!!!!
algodon = sprintf('%0.1f', q1_cde_e);
set(handles.q1_cde_e, 'String', algodon);
handles.cde.q1_cde_e = q1_cde_e;
guidata(hObject,handles)

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai1', 0, 5, 2, 5, -1);
%data=1.25;
vl2 = data(1)% + 1.4
q2_cde_e = ((vl2 - handles.ca.v2)*(handles.ca.f2));
%q2_cde_e = 0.; % ¡¡¡¡¡¡se debe sustituir este valor por el que se obtenga del↙
potenciometro correspondiente!!!!!!!!!
algodon = sprintf('%0.1f', q2_cde_e);
set(handles.q2_cde_e, 'String', algodon);
handles.cde.q2_cde_e = q2_cde_e;
guidata(hObject,handles)

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai2', 0, 5, 2, 5, -1);
%data=1.25;
vl3 = data(1)% + 1.4
q3_cde_e = ((vl3 - handles.ca.v3)*(handles.ca.f3));
```

```
%q3_cde_e = 0; % ¡¡¡¡¡¡se debe sustituir este valor por el que se obtenga del↙
potenciometro correspondiente!!!!!!!!!
algodon = sprintf('%0.1f', q3_cde_e);
set(handles.q3_cde_e, 'String', algodon);
handles.cde.q3_cde_e = q3_cde_e;
guidata(hObject,handles)

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai3', 0, 5, 2, 5, -1);
%data=1.25;
vl4 = data(1)% + 1.4
q4_cde_e = ((vl4 - handles.ca.v4)*(handles.ca.f4));
%q4_cde_e = 0; % ¡¡¡¡¡¡se debe sustituir este valor por el que se obtenga del↙
potenciometro correspondiente!!!!!!!!!
algodon = sprintf('%0.1f', q4_cde_e);
set(handles.q4_cde_e, 'String', algodon);
handles.cde.q4_cde_e = q4_cde_e;
guidata(hObject,handles)

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai4', 0, 5, 2, 5, -1);
%data=1.25;
vl5 = data(1)% + 1.4
q5_cde_e = ((vl5 - handles.ca.v5)*(handles.ca.f5));
%q5_cde_e = 0; % ¡¡¡¡¡¡se debe sustituir este valor por el que se obtenga del↙
potenciometro correspondiente!!!!!!!!!
algodon = sprintf('%0.1f', q5_cde_e);
%set(handles.q5_cde_e, 'String', q5_cde_e);
set(handles.q5_cde_e, 'String', algodon);
handles.cde.q5_cde_e = q5_cde_e;
guidata(hObject,handles)

l0 = 175.85; %160;
l1 = 48.7; %30;
l2 = 128.3; %160;
l3 = -48.7; %-30;
l4 = 139.13; %100;
l5 = 50; %40;

T01 = denavit( g2r(handles.cde.q1_cde_e)+pi/2,  l0,   0, -pi/2 );
T12 = denavit( g2r(handles.cde.q2_cde_e),       l1,  l2,     0 );
T23 = denavit( g2r(handles.cde.q3_cde_e)+pi/2,  l3,   0,  pi/2 );
T34 = denavit( g2r(handles.cde.q4_cde_e),       l4,   0, -pi/2 );
T45 = denavit( g2r(handles.cde.q5_cde_e)-pi/2,   0,  l5,     0 );

T05 = T01 * T12 * T23 * T34 * T45;

%cálculo de la posición-----------------------
px_cde_s = T05(1,4);
if px_cde_s<0.0001 & px_cde_s>-0.0001
    px_cde_s=0;
end
algodon = sprintf('%0.1f', px_cde_s);
set(handles.px_cde_s, 'String', algodon);
```

```
py_cde_s = T05(2,4);
if py_cde_s<0.0001 & py_cde_s>-0.0001
    py_cde_s=0;
end
algodon = sprintf('%0.1f', py_cde_s);
set(handles.py_cde_s, 'String', algodon);

pz_cde_s = T05(3,4);
if pz_cde_s<0.0001 & pz_cde_s>-0.0001
    pz_cde_s=0;
end
algodon = sprintf('%0.1f', pz_cde_s);
set(handles.pz_cde_s, 'String', algodon);

%cálculo de la orientación (parmetro "n")-----
%   "nx"
nx_cde_s = T05(1,1);
if nx_cde_s<0.0001 & nx_cde_s>-0.0001
    nx_cde_s=0;
end
set(handles.nx_cde_s, 'String', nx_cde_s);
%   "ny"
ny_cde_s = T05(2,1);
if ny_cde_s<0.0001 & ny_cde_s>-0.0001
    ny_cde_s=0;
end
set(handles.ny_cde_s, 'String', ny_cde_s);
%   "nz"
nz_cde_s = T05(3,1);
if nz_cde_s<0.0001 & nz_cde_s>-0.0001
    nz_cde_s=0;
end
set(handles.nz_cde_s, 'String', nz_cde_s);

%cálculo de la orientación (parmetro "o")-----
%   "ox"
ox_cde_s = T05(1,2);
if ox_cde_s<0.0001 & ox_cde_s>-0.0001
    ox_cde_s=0;
end
set(handles.ox_cde_s, 'String', ox_cde_s);
%   "oy"
oy_cde_s = T05(2,2);
if oy_cde_s<0.0001 & oy_cde_s>-0.0001
    oy_cde_s=0;
end
set(handles.oy_cde_s, 'String', oy_cde_s);
%   "oz"
oz_cde_s = T05(3,2);
if oz_cde_s<0.0001 & oz_cde_s>-0.0001
    oz_cde_s=0;
```

```
end
set(handles.oz_cde_s, 'String', oz_cde_s);

%cálculo de la orientacion (parmetro "a")-----
%   "ax"
ax_cde_s = T05(1,3);
if ax_cde_s<0.0001 & ax_cde_s>-0.0001
    ax_cde_s=0;
end
set(handles.ax_cde_s, 'String', ax_cde_s);
%   "ay"
ay_cde_s = T05(2,3);
if ay_cde_s<0.0001 & ay_cde_s>-0.0001
    ay_cde_s=0;
end
set(handles.ay_cde_s, 'String', ay_cde_s);
%   "az"
az_cde_s = T05(3,3);
if az_cde_s<0.0001 & az_cde_s>-0.0001
    az_cde_s=0;
end
set(handles.az_cde_s, 'String', az_cde_s);
%aquí finaliza el calculo de la posición y la orientación de efector final
%de la CD
%experimental-----------------------------------------------------------
%_________________________________________________________________________



%_________________________________________________________________________
% NOTA 5.1: para obtener resultados satisfactorios de la CD experimental es
% necesario calibrar el brazo.
% Al dar click en el boton "Calibrate" el programa lee directamente de los
% censores la posición de calibración que es conocida (colocada manualmente)
%-------------------------------------------------------------------------
% --- Executes on button press in calibrar_ca.
function calibrar_ca_Callback(hObject, eventdata, handles)
% hObject    handle to calibrar_ca (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

bandera = handles.ca.v1*handles.ca.v2*handles.ca.v3*handles.ca.v4*handles.ca.v5;

if bandera == 0;
    errordlg('First click Home','Error');
    return;
end

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai0', 0, 5, 2, 5, -1);
%data=0;
vc1 = data(1)% + 1.4 % lectura de referencia del potenciometro 1 en volts
f1 = ( 90/( vc1 - handles.ca.v1))
```

```
[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai1', 0, 5, 2, 5, -1);
%data=0;
vc2 = data(1)% + 1.4 % lectura de referencia del potenciometro 2 en volts
f2 = ( -90/( vc2 - handles.ca.v2))

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai2', 0, 5, 2, 5, -1);
%data=0;
vc3 = data(1)% + 1.4 % lectura de referencia del potenciometro 3 en volts
f3 = ( 90/( vc3 - handles.ca.v3))

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai3', 0, 5, 2, 5, -1);
%data=0;
vc4 = data(1)% + 1.4 % lectura de referencia del potenciometro 4 en volts
f4 = ( 90/( vc4 - handles.ca.v4))

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai4', 0, 5, 2, 5, -1);
%data=0;
vc5 = data(1)% + 1.4 % lectura de referencia del potenciometro 5 en volts
f5 = ( -70/( vc5 - handles.ca.v5))

%ahora se guardan los datos
handles.ca.f1 = f1;
guidata(hObject,handles)

handles.ca.f2 = f2;
guidata(hObject,handles)

handles.ca.f3 = f3;
guidata(hObject,handles)

handles.ca.f4 = f4;
guidata(hObject,handles)

handles.ca.f5 = f5;
guidata(hObject,handles)
%--------------------------------------------------------------------------



%_____________________________________________________________________________
%NOTA 6: en esta sección se calculan los porcentajes de error entre los
%ángulos de giro teóricos y los experimentales.
%NOTA 6.1: al dar click en el botón “Compare” de inicia el calculo de los
%porcentajes de error mencionados en la NOTA 6
%--------------------------------------------------------------------------
% --- Executes on button press in comparar_pe.
function comparar_pe_Callback(hObject, eventdata, handles)
% hObject    handle to comparar_pe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
if handles.cdt.q1_cdt_e ~= 0
    eq1_pe_s = 100*abs((handles.cde.q1_cde_e-handles.cdt.q1_cdt_e)/handles.cdt.q1_cdt_e);
else
    eq1_pe_s = 100*abs(handles.cde.q1_cde_e-handles.cdt.q1_cdt_e);
end
set(handles.eq1_pe_s, 'String', eq1_pe_s);

if handles.cdt.q2_cdt_e ~= 0
    eq2_pe_s = 100*abs((handles.cde.q2_cde_e-handles.cdt.q2_cdt_e)/handles.cdt.q2_cdt_e);
else
    eq2_pe_s = 100*abs(handles.cde.q2_cde_e-handles.cdt.q2_cdt_e);
end
set(handles.eq2_pe_s, 'String', eq2_pe_s);

if handles.cdt.q3_cdt_e ~= 0
    eq3_pe_s = 100*abs((handles.cde.q3_cde_e-handles.cdt.q3_cdt_e)/handles.cdt.q3_cdt_e);
else
    eq3_pe_s = 100*abs(handles.cde.q3_cde_e-handles.cdt.q3_cdt_e);
end
set(handles.eq3_pe_s, 'String', eq3_pe_s);

if handles.cdt.q4_cdt_e ~= 0
    eq4_pe_s = 100*abs((handles.cde.q4_cde_e-handles.cdt.q4_cdt_e)/handles.cdt.q4_cdt_e);
else
    eq4_pe_s = 100*abs(handles.cde.q4_cde_e-handles.cdt.q4_cdt_e);
end
set(handles.eq4_pe_s, 'String', eq4_pe_s);

if handles.cdt.q5_cdt_e ~= 0
    eq5_pe_s = 100*abs((handles.cde.q5_cde_e-handles.cdt.q5_cdt_e)/handles.cdt.q5_cdt_e);
else
    eq5_pe_s = 100*abs(handles.cde.q5_cde_e-handles.cdt.q5_cdt_e);
end
set(handles.eq5_pe_s, 'String', eq5_pe_s);
%--------------------------------------------------------------------------



% --------------------------------------------------------------------
function initialize_gui(fig_handle, handles, x)
% If the metricdata field is present and the reset flag is false, it means
% we are we are just re-initializing a GUI by calling it from the cmd line
% while it is up. So, bail out as we dont want to reset the data.
% if isfield(handles, 'cdt') && ~isreset
%     return;
% end

handles.ci.nx_ci_e = 0;
handles.ci.ny_ci_e = 1;
handles.ci.nz_ci_e = 0;

handles.ci.ox_ci_e = 0;
```

```
handles.ci.oy_ci_e = 0;
handles.ci.oz_ci_e = -1;

handles.ci.ax_ci_e = -1;
handles.ci.ay_ci_e = 0;
handles.ci.az_ci_e = 0;

handles.ci.px_ci_e = 0;
handles.ci.py_ci_e = 300;
handles.ci.pz_ci_e = 160;

%------------------------
handles.cdt.q1_cdt_e = 0;
handles.cdt.q2_cdt_e = 0;
handles.cdt.q3_cdt_e = 0;
handles.cdt.q4_cdt_e = 0;
handles.cdt.q5_cdt_e = 0;

set(handles.nx_ci_e, 'String', handles.ci.nx_ci_e);
set(handles.ny_ci_e, 'String', handles.ci.ny_ci_e);
set(handles.nz_ci_e, 'String', handles.ci.nz_ci_e);
set(handles.ox_ci_e, 'String', handles.ci.ox_ci_e);
set(handles.oy_ci_e, 'String', handles.ci.oy_ci_e);
set(handles.oz_ci_e, 'String', handles.ci.oz_ci_e);
set(handles.ax_ci_e, 'String', handles.ci.ax_ci_e);
set(handles.ay_ci_e, 'String', handles.ci.ay_ci_e);
set(handles.az_ci_e, 'String', handles.ci.az_ci_e);
set(handles.px_ci_e, 'String', handles.ci.px_ci_e);
set(handles.py_ci_e, 'String', handles.ci.py_ci_e);
set(handles.pz_ci_e, 'String', handles.ci.pz_ci_e);

set(handles.q1_cdt_e, 'String', handles.cdt.q1_cdt_e);
set(handles.q2_cdt_e, 'String', handles.cdt.q1_cdt_e);
set(handles.q3_cdt_e, 'String', handles.cdt.q1_cdt_e);
set(handles.q4_cdt_e, 'String', handles.cdt.q1_cdt_e);
set(handles.q5_cdt_e, 'String', handles.cdt.q1_cdt_e);

% Update handles structure
guidata(handles.figure1, handles);


%_____________________________________________________________________________
% NOTA 5.1: para obtener resultados satisfactorios de la CD experimental es
% necesario calibrar el brazo.
% Al dar click en el boton "Home" el programa lee directamente de los
% sensores la posicion de home (colocada manualmente )
%-----------------------------------------------------------------------------

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai0', 0, 5, 2, 5, -1);
%data=2.5;
v1 = data(1)% + 1.4 % lectura del potenciometro 1 en volts

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai1', 0, 5, 2, 5, -1);
%data=2.5;
v2 = data(1)% + 1.4 % lectura del potenciometro 2 en volts

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai2', 0, 5, 2, 5, -1);
%data=2.5;
v3 = data(1)% + 1.4 % lectura del potenciometro 3 en volts

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai3', 0, 5, 2, 5, -1);
%data=2.5;
v4 = data(1)% + 1.4 % lectura del potenciometro 4 en volts

[data, time, initial_time, timed_out] = Acq_N_Updates('dev1/ai4', 0, 5, 2, 5, -1);
%data=2.5;
v5 = data(1)% + 1.4 % lectura del potenciometro 5 en volts

%ahora se guardan los datos
handles.ca.v1 = v1;
guidata(hObject,handles)

handles.ca.v2 = v2;
guidata(hObject,handles)

handles.ca.v3 = v3;
guidata(hObject,handles)

handles.ca.v4 = v4;
guidata(hObject,handles)

handles.ca.v5 = v5;
guidata(hObject,handles)
```