DISEÑO Y CONSTRUCCIÓN DE UN MANIPULADOR TIPO PUMA

Director

Ingeniero JHON FABER ARCHILA

RAUL MUNIVE HERRERA

JOEL RIVERA PEÑA

JOHN W. MORENO VERGEL

UNIVERSIDAD AUTONOMA DE BUCARAMANGA

FACULTAD DE INGENIERIA MECATRONICA

ESCUELA DE CIENCIAS NATURALES E INGENIERIA

BUCARAMANGA

2005

DISEÑO Y CONSTRUCCIÓN DE UN MANIPULADOR TIPO PUMA

RAUL MUNIVE HERRERA

JOEL RIVERA PEÑA

JOHN W. MORENO VERGEL

UNIVERSIDAD AUTONOMA DE BUCARAMANGA

FACULTAD DE INGENIERIA MECATRONICA

ESCUELA DE CIENCIAS NATURALES E INGENIERIA

BUCARAMANGA

2005

Nota de aceptación

_____

_____

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

_____

Ciudad y fecha (día, mes, año)

AGRADECIMIENTOS

Estos agradecimientos van dirigidos a todos aquellos que nos colaboraron durante la realización de nuestro proyecto y a los que lo hicieron posible.

Gracias de antemano a nuestro Decano, Ingeniero Eduardo Calderón Porras, por que fue quien nos asignó este proyecto y por la asesoría prestada. A nuestro director de proyectos, Ingeniero John Faber Archila, quien nos apoyó durante la realización del mismo y nos orientó en el proceso correspondiente. Al Doctor Faustino Muñoz, por su asesoría prestada y por su interés en el progreso de diseño. Al Ingeniero Mauricio, director del taller de mecánica, por todo su apoyo, colaboración y conocimientos compartidos.

A los Docentes que nos apoyaron y asesoraron para llevar a cabo nuestro proyecto y a la gran importancia de la calidad de sus enseñanzas, las cuales fueron de base y fundamento para la realización del proyecto.

A todos aquellos que nos colaboraron, nos prestaron su apoyo y nos aconsejaron en el proceso.

Finalmente gracias a Dios por permitirnos alcanzar nuestras metas y a nuestros padres y amigos por todo el apoyo recibido.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCION

Uno de los manipuladores más versátiles de la industria es el tipo PUMA, *Programmable Universal Machine for Assambly*, o *Brazo Manipulador Universal Programable*. Puede tener hasta seis grados de libertad, lo que lo hace ser robusto en aplicaciones complejas. El diseño PUMA es una estructura inspirada en el brazo humano, con el fin de serle de ayuda al hombre en trabajos automatizados con una complejidad y precisión similar o superior al mismo.

En primera instancia se citará un breve marco teórico que trate los fundamentos de los conceptos y dispositivos utilizados en el proyecto y se desarrollarán diagramas correspondientes a la metodología de diseño mecatrónico que se estableció para el proyecto.

Después se desarrollarán los cálculos respectivos al sistema, como cálculos cinemáticos y dinámicos, con el fin de encontrar las características mecánicas y cálculos de posicionamiento del manipulador. Luego se describirá en detalle las características físicas, como la estructura, materiales, potencia y sistemas de transmisión.

Más adelante se hablará de los sensores y de la metodología de diseño del control general del sistema. Entonces se describirán los sistemas electrónicos usados para el control del sistema y las fuentes de alimentación correspondientes. A continuación se describirá el microcontrolador, el diagrama de flujo de su algoritmo y el código de programación de dicho algoritmo en lenguaje *Assembler*.

Siguiendo finalmente con la construcción y montaje del manipulador y el software de control por computador desarrollado en MatLAB.

PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACION

El objetivo de la realización de este proyecto surge como una solución a un problema que actualmente tiene la facultad de ingeniería mecatrónica de la universidad autónoma de Bucaramanga (UNAB) en el mejoramiento del laboratorio de robótica.

El problema que se quiere afrontar con este proyecto, es la realización de un manipulador didáctico o experimental tipo PUMA el cual se podrá instalar en el laboratorio de robótica con fines dedicados a la enseñanza y aprendizaje de la ciencia de la robótica.

La importancia de la elaboración de este proyecto es la de mejorar la infraestructura con que actualmente cuenta la universidad. La principal aplicación de este tipo de manipulador experimental es aprender sobre la estructura, el funcionamiento y la programación de los robots industriales, ya que posiblemente son estos el principal tipo de robots que han tenido una aplicación útil y practica en la industria. Principalmente en el sector automotriz, aunque también se han extendido a otros tipos de industria como la industria electrónica en donde se utilizan para ensamblar automáticamente circuitos en general, equipos electrónicos, armar maquinas o sus parte y en fin para diferentes usos.

El problema que anteriormente se ha expuesto se ha afrontado en diferentes planteles educativos como un problema que debe ser atendido ya es muy importante tener buenos sistemas de aprendizaje que faciliten mejor el desarrollo de destrezas y capacidades en cada estudiante, por tanto seria de gran

importancia para la facultad de ingeniería mecatrónica formar profesionales con excelentes perfiles y capacidades que puedan dar solución a cualquier problema que se presente en la industria moderna.

OBJETIVOS

OBJETIVO GENERAL

DISEÑAR Y CONSTRUIR UN MANIPULADOR TIPO PUMA PARA EL LABORATORIO DE ROBÓTICA DE LA FACULTAD DE MECATRÓNICA DE LA UNAB.

OBJETIVOS ESPECÍFICOS

Fase de Investigación:

- Realizar una investigación acerca de los fundamentos y estándares que rigen el diseño y funcionamiento de los manipuladores.
- Realizar cálculos de posicionamiento para el manipulador PUMA necesarios para su control.
- Desarrollar cálculos cinéticos fundamentales en el diseño previo del manipulador.

Fase de Diseño:

- Implementar herramientas CAE, como *CosmosWorks* en la selección de los materiales adecuados para el brazo robótico.
- Desarrollo del algoritmo de control para el manipulador.
- Identificar la plataforma para implementar el algoritmo de control en el diseño final del manipulador.

- Modelar un manipulador PUMA con sus componentes principales por medio de software de diseño asistido por computador (CAD) como *SolidWorks*.

Fase de Construcción:

- Seleccionar el tipo y cantidad de motores necesarios para suplir los grados de libertad del manipulador PUMA.
- Diseñar los circuitos electrónicos que se implementarán para el control de los diferentes motores articuladores del brazo robótico.
- Seleccionar fuentes de alimentación para el sistema electrónico de control.
- Implementar el software de control para mover cada una de las articulaciones (motores) del manipulador.
- Ensamblar cada una de las partes que conforman el brazo mecánico.
- Seleccionar sensores para el sistema de percepción del manipulador.

MARCO TEORICO

ESTÁNDARES

1. RIA (ROBOTIC INDUSTRY ASSOCIATION)

La definición de la Asociación de Industrias Robóticas (RIA) de robot es la siguiente: Un robot industrial es un manipulador programable multifuncional, diseñado para mover piezas, herramientas, dispositivos especiales mediante movimientos variados, programados para la ejecución de diversas tareas.

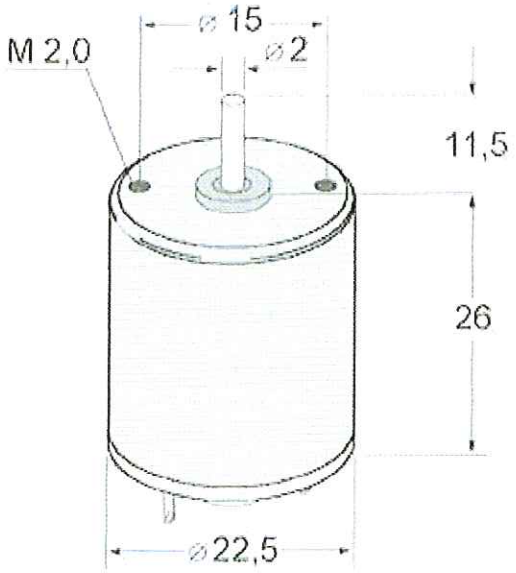
2. ISO (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION)

La definición de la Organización Internacional de Normas (ISO) es: un robot industrial es un manipulador automático reprogramable y multifuncional, que posee ejes capaces de agarrar materiales, objetos, herramientas mecanismos especializados a través de operaciones programadas para la ejecución de una variedad de tareas. Como se puede apreciar, estas definiciones se ajustan a la mayoría de las aplicaciones industriales de robots, salvo para las aplicaciones de inspección y para los robots móviles (autónomos) o robots personales

MOTORES CD

Descripción: El motor eléctrico es un dispositivo electromotriz, esto quiere decir que convierte la energía eléctrica en energía motriz. Todos los motores disponen

de un eje de salida para acoplar un engranaje, polea o mecanismo capaz de transmitir el movimiento creado por el motor.



Funcionamiento: El funcionamiento de un motor se basa en la acción de campos magnéticos opuestos que hacen girar el rotor (eje interno) en dirección opuesta al estator (imán externo o bobina), con lo que si sujetamos por medio de soportes o bridas la carcasa del motor el rotor con el eje de salida será lo único que gire.

Para cambiar la dirección de giro en un motor de Corriente Continua tan solo tenemos que invertir la polaridad de la alimentación del motor.

Para modificar su velocidad podemos variar su tensión de alimentación con lo que el motor perderá velocidad, pero también perderá par de giro (fuerza) o para no perder par en el eje de salida podemos hacer un circuito modulador de anchura de pulsos (pwm) con una salida a transistor de mas o menos potencia según el motor utilizado.

SISTEMAS DE CONTROL

Se define control como el conjunto de procedimientos que aplicamos para que un

sistema, desde un estado inicial, alcance cierto estado final y se mantenga en el o muy próximo, independientemente de los cambios en magnitudes externas o internas que puedan afectar.

El control de un sistema puede tener 2 finalidades distintas:

- Regulación: consiste en mantener la salida constante independientemente de la variación de magnitudes externas. Ej: Mantener constante la orientación de una antena a un satélite.
- Seguimiento de trayectorias: hacer que la variable de salida tenga en todo momento un valor tan próximo como sea posible al de alguna variable de entrada. Ej: Controlar un robot móvil para que siga un camino establecido.

Estos son algunos de los términos básicos utilizados en control:

Señal de salida: es la variable que se desea controlar (posición, velocidad, presión, temperatura, etc.). También se denomina *variable controlada*.

Señal de referencia: es el valor que se desea que alcance la señal de salida.

Error: es la diferencia entre la señal de referencia y la señal de salida real.

Señal de control: es la señal que produce el controlador para modificar la variable controlada de tal forma que se disminuya, o elimine, el error.

Señal análoga: es una señal continua en el tiempo.

Señal digital: es una señal que solo toma valores de 1 y 0. El PC solo envía y/o recibe señales digitales.

Conversor análogo/digital: es un dispositivo que convierte una señal analógica en una señal digital (1 y 0).

Conversor digital/análogo: es un dispositivo que convierte una señal digital en

una señal analógica (corriente o voltaje).

Planta: es el elemento físico que se desea controlar. Planta puede ser: un motor, un horno, un sistema de disparo, un sistema de navegación, un tanque de combustible, etc.

Proceso: operación que conduce a un resultado determinado.

Sistema: consiste en un conjunto de elementos que actúan coordinadamente para realizar un objetivo determinado.

Perturbación: es una señal que tiende a afectar la salida del sistema, desviándola del valor deseado.

Sensor: es un dispositivo que convierte el valor de una magnitud física (presión, flujo, temperatura, etc.) en una señal eléctrica codificada ya sea en forma analógica o digital. También es llamado *transductor*. Los sensores, o transductores, analógicos envían, por lo regular, señales normalizadas de 0 a 5 voltios, 0 a 10 voltios o 4 a 20 mA.

Sistema de control en lazo cerrado: es aquel en el cual continuamente se está monitoreando la señal de salida para compararla con la señal de referencia y calcular la señal de error, la cual a su vez es aplicada al controlador para generar la señal de control y tratar de llevar la señal de salida al valor deseado. También es llamado *control realimentado*.

Sistema de control en lazo abierto: en estos sistemas de control la señal de salida no es monitoreada para generar una señal de control.

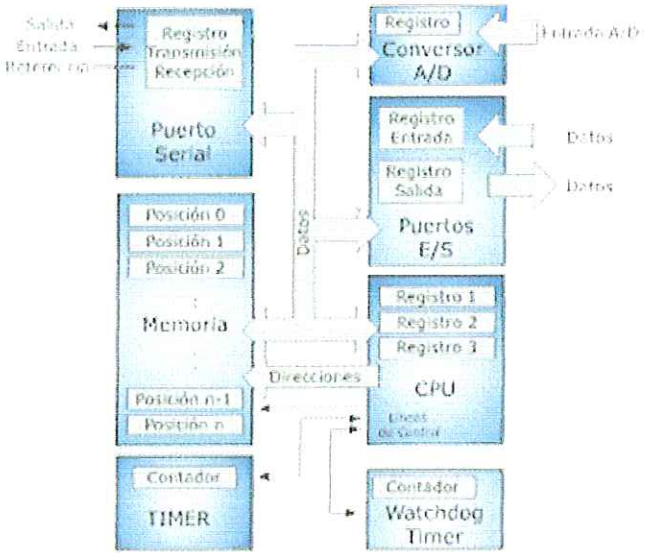
MICROCONTROLADORES

Muchos de los sistemas digitales pueden diseñarse empleando procesadores o

microcontroladores, la selección del dispositivo depende del tipo de aplicación y la diferencia básica que existe entre estos dos dispositivos se explica a continuación:

- Los procesadores son empleados para procesar información. A manera de ejemplo, con un procesador se podría hallar todas las personas en Bogotá entre edades de 15 y 30 años, de sexo femenino, y ordenados por el Apellido. Los procesadores usan un conjunto de instrucciones útiles para procesar datos, lo cual los hace muy versátiles para manejar información.
- Los microcontroladores son utilizados como su nombre lo indica para controlar. Son muy utilizados para implementar controles automáticos. Como ejemplo, un microcontrolador puede sensar la temperatura de un proceso, compararla con un valor almacenado en memoria y tomar la decisión de encender un equipo de calefacción si la temperatura baja de cierto valor, y además de ello mostrar el valor en un *display*.

Los microcontroladores generalmente tienen instrucciones especiales que permiten controlar procesos como el indicado anteriormente y otros más complejos; todo depende de la habilidad del programador para generar el código para manejar el proceso. Un microcontrolador es simplemente un procesador con memoria *ROM* y *RAM*, puertos de *E/S* y otros dispositivos de propósito especial como conversores *A/D*, contadores, temporizadores y puertos de comunicación, o en otras palabras es un microcomputador con funciones especiales. En la figura siguiente se indica la estructura interna típica de un microcontrolador:



Estos dispositivos generalmente incluyen variedad de funciones especiales que se pueden utilizar gracias a los dispositivos internos incluidos dentro de ellos. Entre las características mas relevantes de un microcontrolador, se pueden enunciar las siguientes:

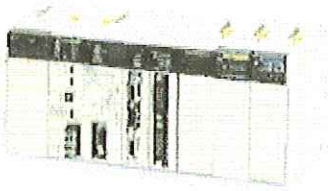
- La memoria de programa generalmente es una Flash EEPROM.
- Tiene puertos de Entrada y Salida (Configurables por software).
- Poseen contadores de propósito especial.
- Tiene incluido un reloj del sistema que permite contabilizar tiempo.
- Algunos modelos incluyen conversores A/D.
- Tiene Memoria EEPROM para almacenar datos.
- Tiene puerto de comunicaciones.
- Manejan velocidades de operación hasta 20 MHz.
- Algunos de estos dispositivos tienen puerto de comunicaciones serial.
- Tienen entradas para interrupción.
- La programación es rápida.
- Las herramientas de desarrollo son económicas y se encuentran disponibles en a red, las cuales incluyen el ensamblador y simulador.

Los microcontroladores se pueden encontrar en varias aplicaciones que se relacionen con medida, almacenamiento, control, cálculo entre otras. También se

pueden encontrar dentro de los teclados, módems, impresoras y otros periféricos. Como se puede notar los microcontroladores son dispositivos muy versátiles que pueden ser utilizados en muchas aplicaciones, donde todo el potencial se encuentra en la programación.

PLC (AUTÓMATAS PROGRAMABLES)

Un autómata programable industrial (API) o Programable logic controller (PLC), es un equipo electrónico, programable en lenguaje no informático, diseñado para controlar en tiempo real y en ambiente de tipo industrial, procesos secuenciales.
Un PLC trabaja en base a la información recibida por los captadores y el programa lógico interno, actuando sobre los accionadores de la instalación.



Campos de aplicación

El PLC por sus especiales características de diseño tiene un campo de aplicación muy extenso. La constante evolución del hardware y software amplía constantemente este campo para poder satisfacer las necesidades que se detectan en el espectro de sus posibilidades reales.
Su utilización se da fundamentalmente en aquellas instalaciones en donde es necesario un proceso de maniobra, control, señalización, etc. , por tanto, su aplicación abarca desde procesos de fabricación industriales de cualquier tipo a transformaciones industriales, control de instalaciones, etc.
Sus reducidas dimensiones, la extremada facilidad de su montaje, la posibilidad de almacenar los programas para su posterior y rápida utilización, la modificación o alteración de los mismos, etc., hace que su eficacia se aprecie fundamentalmente

en procesos en que se producen necesidades tales como:

- Espacio reducido
- Procesos de producción periódicamente cambiantes
- Procesos secuenciales
- Maquinaria de procesos variables
- Instalaciones de procesos complejos y amplios
- Chequeo de programación centralizada de las partes del proceso

Ejemplos de aplicaciones generales:

- Maniobra de máquinas
- Maquinaria industrial de plástico
- Máquinas transfer
- Maquinaria de embalajes
- Maniobra de instalaciones:
- Instalación de aire acondicionado, calefacción...
- Instalaciones de seguridad
- Señalización y control:
- Chequeo de programas
- Señalización del estado de procesos

Ventajas e inconvenientes

No todos los autómatas ofrecen las mismas ventajas sobre la lógica cableada, ello es debido, principalmente, a la variedad de modelos existentes en el mercado y las innovaciones técnicas que surgen constantemente. Tales consideraciones me obligan e referirme a las ventajas que proporciona un autómata de tipo medio.

Ventajas

- Menor tiempo empleado en la elaboración de proyectos debido a que:
- No es necesario dibujar el esquema de contactos

- No es necesario simplificar las ecuaciones lógicas, ya que, por lo general la capacidad de almacenamiento del módulo de memoria es lo suficientemente grande.
- La lista de materiales queda sensiblemente reducida, y al elaborar el presupuesto correspondiente eliminaremos parte del problema que supone el contar con diferentes proveedores, distintos plazos de entrega.
- Posibilidad de introducir modificaciones sin cambiar el cableado ni añadir aparatos.
- Mínimo espacio de ocupación.
- Menor coste de mano de obra de la instalación.
- Economía de mantenimiento. Además de aumentar la fiabilidad del sistema, al eliminar contactos móviles, los mismos autómatas pueden indicar y detectar averías.
- Posibilidad de gobernar varias máquinas con un mismo autómata.
- Menor tiempo para la puesta en funcionamiento del proceso al quedar reducido el tiempo cableado.
- Si por alguna razón la máquina queda fuera de servicio, el autómata sigue siendo útil para otra máquina o sistema de producción.

Inconvenientes

- Como inconvenientes podríamos hablar, en primer lugar, de que hace falta un programador, lo que obliga a adiestrar a uno de los técnicos en tal sentido, pero hoy en día ese inconveniente esta solucionado porque las universidades ya se encargan de dicho adiestramiento.
- El coste inicial también puede ser un inconveniente.

Funciones básicas de un PLC

- Detección:

  Lectura de la señal de los captadores distribuidos por el sistema de fabricación.

- Mando:

Elaborar y enviar las acciones al sistema mediante los accionadores y preaccionadores.

- Dialogo hombre maquina:

  Mantener un diálogo con los operarios de producción, obedeciendo sus consignas e informando del estado del proceso.

- Programación:

  Para introducir, elaborar y cambiar el programa de aplicación del autómata. El dialogo de programación debe permitir modificar el programa incluso con el autómata controlando la maquina.

Nuevas Funciones

- Redes de comunicación:

  Permiten establecer comunicación con otras partes de control. Las redes industriales permiten la comunicación y el intercambio de datos entre autómatas a tiempo real. En unos cuantos milisegundos pueden enviarse telegramas e intercambiar tablas de memoria compartida.

- Sistemas de supervisión:

  También los autómatas permiten comunicarse con ordenadores provistos de programas de supervisión industrial. Esta comunicación se realiza por una red industrial o por medio de una simple conexión por el puerto serie del ordenador.

- Control de procesos continuos:

  Además de dedicarse al control de sistemas de eventos discretos los autómatas llevan incorporadas funciones que permiten el control de procesos continuos. Disponen de módulos de entrada y salida analógicas y la posibilidad de ejecutar reguladores PID que están programados en el autómata.

- Entradas- Salidas distribuidas:

  Los módulos de entrada salida no tienen porqué estar en el armario del autómata. Pueden estar distribuidos por la instalación, se comunican con la unidad central del autómata mediante un cable de red.

- Buses de campo:

  Mediante un solo cable de comunicación se pueden conectar al bus captadores y accionadores, reemplazando al cableado tradicional. El autómata consulta cíclicamente el estado de los captadores y actualiza el estado de los accionadores.

MANIPULADORES CINEMÁTICOS

La tecnología robótica encontró su primer aplicación en la industria nuclear con el desarrollo de teleoperadores para manejar material radiactivo. Los robots más recientes han sido utilizados para soldar a control remoto y la inspección de tuberías en áreas de alta radiación. El accidente en la planta nuclear de Three Mile Island en Pennsylvania en 1979 estimuló el desarrollo y aplicación de los robots en la industria nuclear. El reactor numero 2 (TMI-2) predio su enfriamiento, y provocó la destrucción de la mayoría del reactor, y dejo grandes áreas del reactor contaminadas, inaccesible para el ser humano. Debido a los altos niveles de radiación las tareas de limpieza solo eran posibles por medios remotos. Varios robots y vehículos controlados remotamente han sido utilizados para tal fin en los lugares donde ha ocurrido una catástrofe de este tipo. Ésta clase de robots son equipados en su mayoría con sofisticados equipos para detectar niveles de radiación, cámaras, e incluso llegan a traer a bordo un minilaboratorio para hacer pruebas.

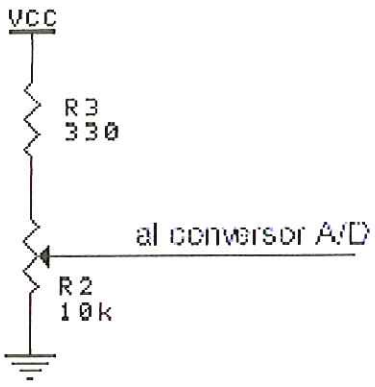
Potenciómetros:

Son de uso común y son muy útiles para medir movimientos y determinar la posición de un mecanismo determinado como por ejemplo el eje de una articulación de un brazo mecánico.

Debido a que los potenciómetros poseen un ángulo de giro de aproximadamente 270°, no es posible usarlos en mecanismos que deben realizar un giro completo o

bien mas de una vuelta sobre su eje.

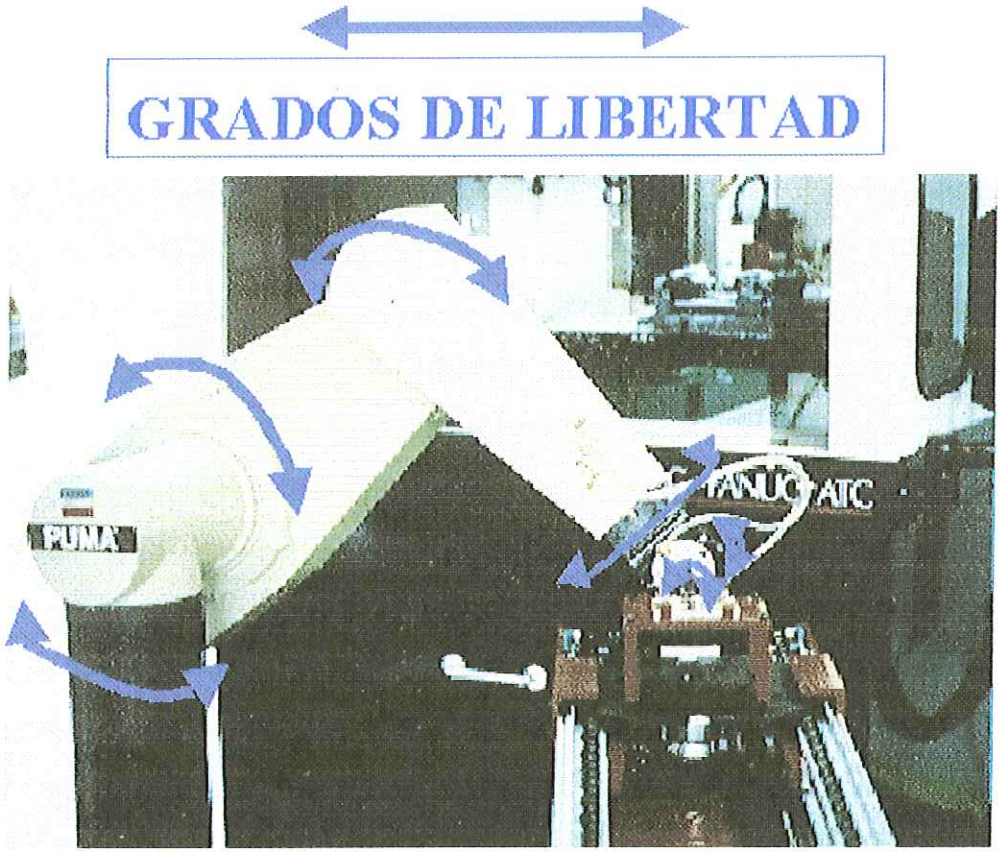
Como se aprecia en el diagrama siguiente, la forma de conexión es similar al caso del LDR, con la simple diferencia que en este caso el Potenciómetro es un divisor resistivo en si mismo y R3 se usa como simple limitador de corriente.
Los valores son a modo de ejemplo y pueden usarse cualquier valor dentro de rangos aceptables. No muy bajos para no provocar un elevado consumo (10K es lo mas bajo recomendable) y no muy elevado ya que la corriente sería demasiado baja (no mas de 1.5M).



Existen dos tipos de potenciómetros en el mercado: Lineares y Logarítmicos (estos últimos usados normalmente en audio).
Los del tipo linear varían su valor en forma constante (linealmente), los del tipo logarítmicos poseen una curva de variación del tipo logarítmica, esto es decir que su valor aumenta lentamente en los extremos y luego los valores cambian cada vez mas rápidamente.
Los mas recomendados a la hora de sensar posiciones de mecanismos son los del tipo linear.

MANIPULADOR PUMA (Programmable Universal Manipulator for Assembly)

Máquina Universal de Ensamblaje. Este robot presenta una configuración angular, tiene 3 grados de libertad en el cuerpo y brazo y 3 en la muñeca, dando un total de 6 grados de libertad.



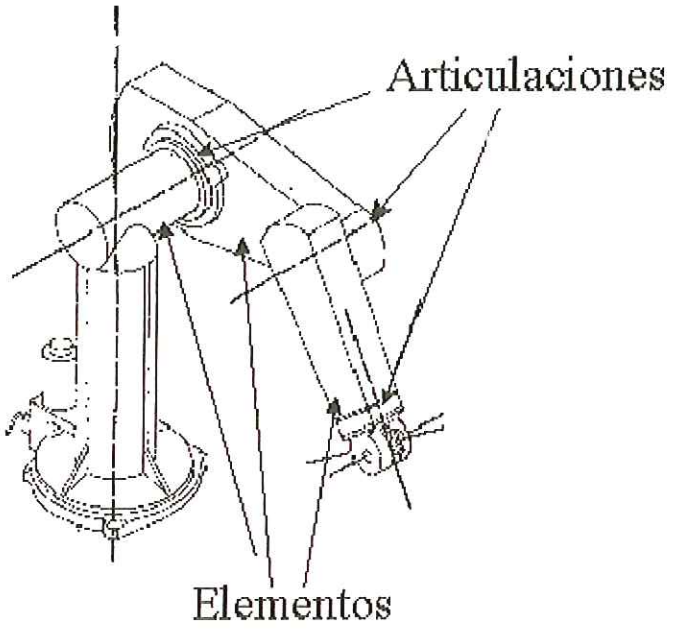
Mikell Groover, en su libro *Automation, Production Systems and Computer Integrated Manufacturing*, define al robot industrial como **"...una máquina programable, de propósito general, que posee ciertas características antropomórficas, es decir, con características basadas en la figura humana..."**.

Cabe destacar que la característica antropomórfica más común en nuestros días es la de un brazo mecánico, el cual realiza diversas tareas industriales. Existen en el mercado diversas empresas dedicadas a la fabricación de robots industriales

por lo que existen diferentes marcas y modelos. Estos últimos son normalmente asignados para identificarlos o de acuerdo a su función.

Componentes:

El componente principal lo constituye el manipulador, el cual consta de varias articulaciones y sus elementos.

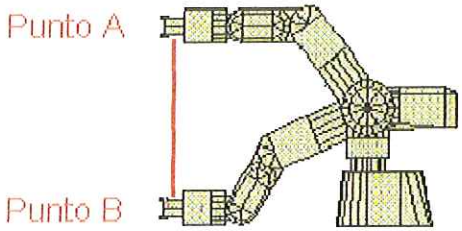


Las partes que conforman el manipulador reciben los nombres de: **cuerpo, brazo, muñeca y efector final.** Al efector final se le conoce comúnmente como sujetador o *gripper.*

Vamos a centrar nuestra atención en los elementos de las articulaciones. Cada articulación provee al robot de al menos un *"grado de libertad"*. En otras palabras, las articulaciones permiten al manipulador realizar movimientos:

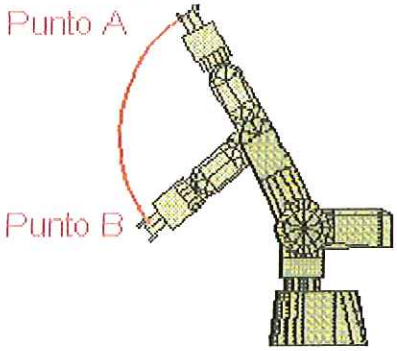
- Lineales que pueden ser horizontales o verticales.

Movimiento lineal entre los puntos A-B

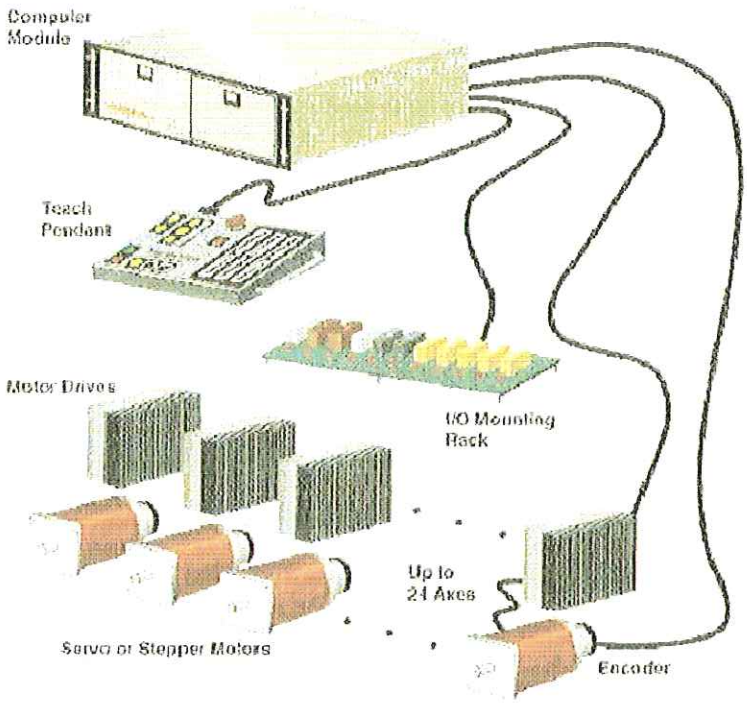


- Por articulación.

Movimiento angular (por articulación) entre los puntos A-B

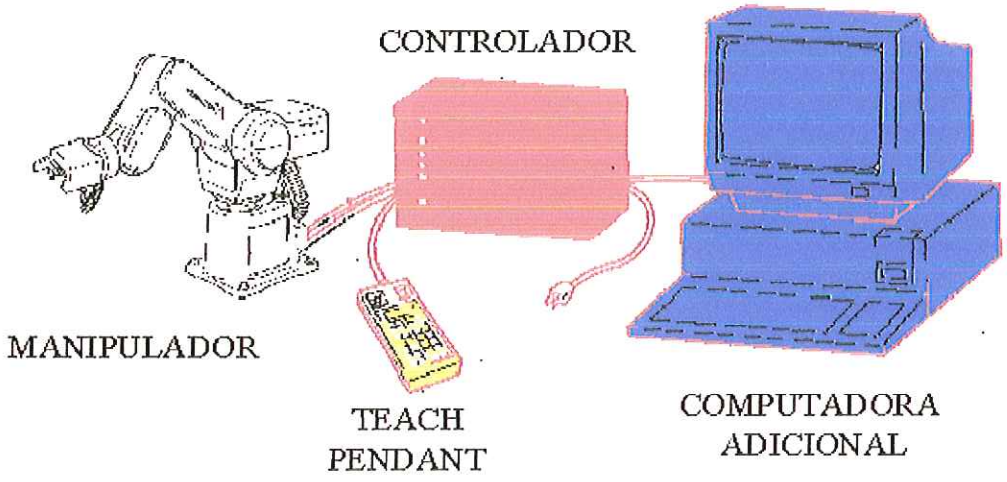


(En los dos casos la línea roja representa la trayectoria seguida por el robot). Además del manipulador, los otros elementos que forman parte del robot son un controlador, mecanismos de entrada y salida de datos y dispositivos especiales. El controlador del robot, como su nombre lo indica, es el que controla cada uno de los movimientos del manipulador y guarda sus posiciones.



El controlador recibe y envía señales a otras máquinas-herramientas (por medio de señales de entrada/salida) y almacena programas. Los mecanismos de entrada y salida, más comunes son: **teclado, monitor y caja de comandos llamada "teach pendant".** En el dibujo anterior tenemos un controlador **(computer module)** que envía señales a los motores de cada uno de los ejes del robot, la caja de comandos **("teach pendant")** la cual sirve para enseñarle las posiciones al manipulador del robot.

COMPONENTES DEL ROBOT



Los dispositivos de entrada y salida permiten introducir y, a su vez, ver los datos del controlador. Para mandar instrucciones al controlador y para dar de alta programas de control, comúnmente se utiliza una computadora adicional.

Es necesario aclarar que algunos robots únicamente poseen uno de estos componentes. En estos casos, uno de los componentes de entrada y salida permite la realización de todas las funciones.

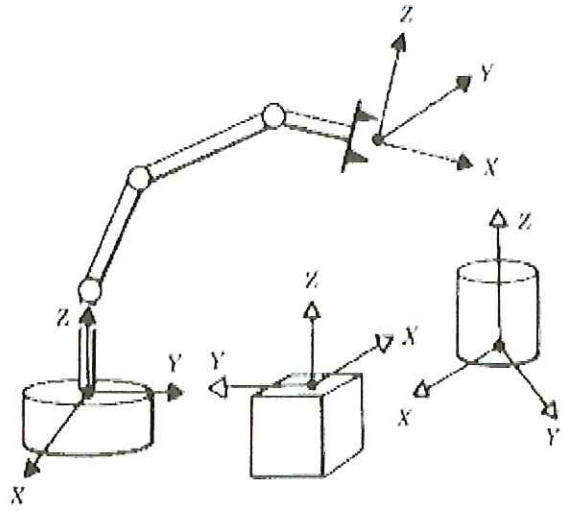
Leyes de la Robótica

Isaac Asimov ha contribuido con varias narraciones a la ciencia ficción con el tema de los robots y a él se le atribuye el acuñamiento del término robótica. Además fue él quien propuso las tres leyes de la Robótica con las que se garantiza que una máquina esté bien diseñada y sea segura; estas leyes son:

1. Un robot no puede actuar contra un ser humano o mediante la inacción, permitir que un ser humano sufra daños.
2. Un robot debe obedecer las órdenes dadas por los seres humanos, salvo que estén en conflicto con la primera ley.
3. Un robot debe proteger su propia existencia, a no ser que este en conflicto con las dos primeras leyes.

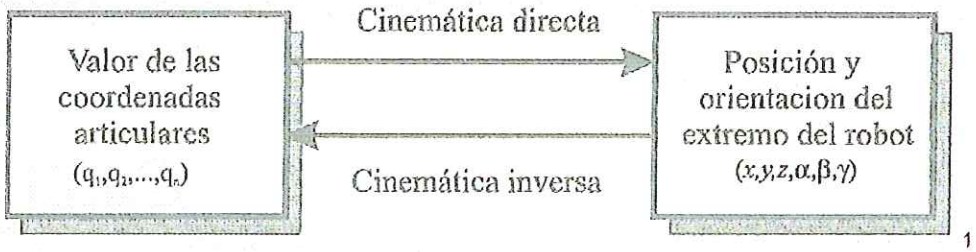
Descripción de la Posición y de la Orientación

En el estudio de la robótica es importante prestar una atención constante a la localización de objetos en el espacio. Para describir la posición y la orientación de un cuerpo en el espacio asociaremos siempre un sistema de coordenadas o marco al objeto. Describimos entonces la posición y la orientación de este objeto con respecto a un sistema de coordenadas de referencia.



Cinemática del Robot

Se entiende por cinemática el estudio del movimiento sin considerar las fuerzas que lo producen. La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia. Existen dos problemas fundamentales a resolver en la cinemática del robot; el primero es el problema cinemático directo, y consiste en determinar cual es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot; y el segundo; denominado problema cinemático inverso, resuelve la configuración que debe adoptar el robot para una posición y orientación del extremo conocidas.



[1]

[1] Antonio Barrientos, "Fundamentos de Robótica". Diagrama de relación entre cinemática directa e inversa. Pagina 94.

Problema Cinemático Directo

Dado que un robot se puede considerar como una cadena cinemática formada por objetos rígidos o eslabones unidos entre si mediante articulaciones, se puede establecer un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia.

ALGORITMO DE DENAVIT-HARTENBERG

Consiste en un método matricial que permite establecer de manera sistemática un sistema de coordenadas ligado a cada eslabón de una cadena articulada, pudiéndose determinar las ecuaciones cinemáticas de la cadena completa.

De acuerdo con esto será posible pasar de un eslabón a otro mediante 4 transformaciones básicas que dependerán de las características geométricas de cada uno de ellos. Los cuatro parámetros D-H son los siguientes:

θ_i : Rotación alrededor del eje z_{i-1} un ángulo θ_1

d_i : Traslación a lo largo de z_{i-1} una distancia d_i ; vector $d_i(0,0,d_i)$

a_i : Traslación a lo largo de x_i una distancia a_i; vector $a_i(a_i,0,0)$

α_i : Rotación alrededor del eje x_i un ángulo α_i.

Para obtener la matriz de transformación **T** que relacione la posición y orientación del extremo del robot respecto al sistema de referencia fijo situado en la base del robot; las transformaciones se realizan en el siguiente orden:

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

Según la anterior expresión; una vez realizado el producto entre matrices, se obtienen las matrices **A,** cuyo producto final dará como resultado la matriz de

transformación **T**.

Problema Cinemático Inverso

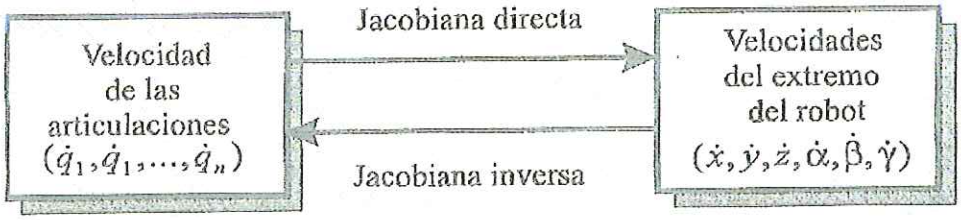
Para abordar este problema el procedimiento de obtención de las ecuaciones depende de la configuración del robot.

La solución por métodos geométricos es ideal para robots de pocos grados de libertad; el método consiste en encontrar suficiente numero de relaciones geométricas en las que intervengan las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos.

Matríz Jacobiana

Dentro del modelado cinemático de un robot también es importante encontrar la relación entre las velocidades de las coordenadas articulares y las de posición y orientación de su extremo; esto se obtiene mediante relaciones diferenciales a partir de la Matríz Jacobiana.

Existen dos problemas a resolver; la Matriz Jacobiana Directa y la Matriz Jacobiana Inversa; el primero busca conocer las velocidades del extremo del robot a partir de las velocidades de cada articulación; y el segundo permite conocer las velocidades articulares necesarias para obtener unas velocidades determinadas en el extremo del robot.



[2]

Matríz Jacobiana Directa

Una vez conocidas las ecuaciones que resuelven el problema cinemático directo de un robot de determinado número de grados de libertad, se procede a derivar con respecto al tiempo cada miembro del conjunto de ecuaciones para obtener la siguiente relación diferencial y así obtener la velocidad del extremo del robot:

$$\begin{vmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{vmatrix} = J \cdot \begin{vmatrix} \dot{q} \\ \\ \vdots \\ \vdots \\ \\ \dot{q}_n \end{vmatrix}$$

con:

[2] Antonio Barrientos, "Fundamentos de Robótica". Diagrama de relación entre Matriz Jacobiana Directa e Inversa. Pagina 122.

$$J = \begin{vmatrix} \frac{\partial f_x}{\partial q_1} & \cdots & \frac{\partial f_x}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_\gamma}{\partial q_1} & \cdots & \frac{\partial f_\gamma}{\partial q_n} \end{vmatrix}$$

Donde 'J ' se denomina la Matríz Jacobiana.

Matríz Jacobiana Inversa

Del mismo modo que se obtiene la relación directa, puede obtenerse la relación inversa invirtiendo simbólicamente la Matriz Jacobiana Directa. El procedimiento a seguir es a partir de las ecuaciones obtenidas del modelo cinemático inverso; se procede a derivar con respecto al tiempo cada miembro de la igualdad; para obtener la siguiente relación diferencial y así calcular las velocidades articulares partiendo de las del extremo del robot.

$$\begin{vmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \dot{q}_n \end{vmatrix} = J^{-1} \cdot \begin{vmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{vmatrix}$$

con:

$$J^{-1} = \begin{vmatrix} \frac{\partial f_1}{\partial x} & \cdots & \cdots & \cdots & \frac{\partial f_1}{\partial \gamma} \\ \vdots & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \frac{\partial f_n}{\partial x} & \cdots & \cdots & \cdots & \frac{\partial f_n}{\partial \gamma} \end{vmatrix}$$

Donde 'J^{-1}' es la Matríz Jacobiana invertida.

Dinámica del Robot

La dinámica se ocupa de la relación entre las fuerzas que actúan sobre un cuerpo y el movimiento que en él se origina. El modelo dinámico de un robot tiene por objetivo conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo. Este modelo relaciona matemáticamente lo siguiente:

La localización del robot definidla por sus variables articulares, y sus derivadas: velocidad y aceleración.

Las fuerzas y pares aplicados en las articulaciones o el extremo del robot.

Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

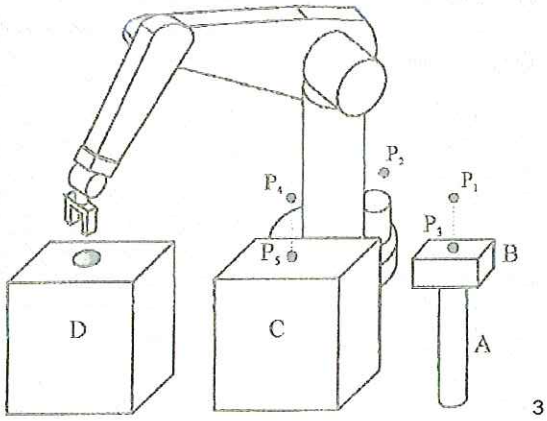
El modelo dinámico se resuelve de manera iterativa utilizando procedimientos numéricos y se expresa mediante una serie de ecuaciones de tipo diferencial de 2º orden, cuya integración permite conocer qué movimiento surge al aplicar unas fuerzas; o qué fuerzas hay que aplicar para obtener un movimiento determinado; además es imprescindible para conseguir los siguientes fines:

Simulación y movimiento del robot.

Diseño y evaluación de la estructura mecánica del robot.

Dimensionamiento de los actuadotes.

Diseño y evaluación del control dinámico del robot.



[3]

[3] Antonio Barrientos, "Fundamentos de la Robótica". Ejemplo de una tarea programada realizada por un robot. Pagina 224.

1. DISEÑO MECATRÓNICO

El manipulador automático aplicado al proceso de ensamble cuenta con:

- Un sistema mecánico compuesto de 5 DOF activadas mediante motoreductores que realizan la transmisión del movimiento por medio de engranajes.

- Un sistema sensorial que permite evaluar la posición y orientación del manipulador con el fin de obtener exactitud en sus movimientos y realizar un adecuado control sobre las articulaciones. los potenciómetros permiten conocer la posición de cada una de las articulaciones y los sensores de presencia (finales de carrera) que no solo indican la posición extrema del manipulador también se usa como protección para evitar mal uso del los controles manuales.

- Un sistema eléctrico que permite ofrecer la respectiva alimentación a cada uno de los componentes eléctricos del manipulador (motoreductores, sensores, controladores, las interfaces entre los sensores con el sistema de control, las fuentes de alimentación y las respectivas derivaciones para cada uno de los componentes).

- Un sistema electrónico que consta de los dispositivos o interfaces entre sensores – actuadores y sistema de control.

- Un sistema de control (microcontrolador) que se encarga de tomar las respectivas acciones de control una vez evaluada la información que brindan los sensores y las especificaciones dadas por el operario o el PC.

1.1 METODOLOGÍA DEL DISEÑO MECATRÓNICO

1.1.1 Principio del diseño mecatrónico

El diseño mecatrónico del prototipo en principio se basa de los saberes fundamentales de las ingenierías Mecánica, Electrónica, Sistemas y Control. Estas ciencias se integran sinérgicamente para dar origen a sistemas integrales y automatizados, como se puede ver en la siguiente figura:

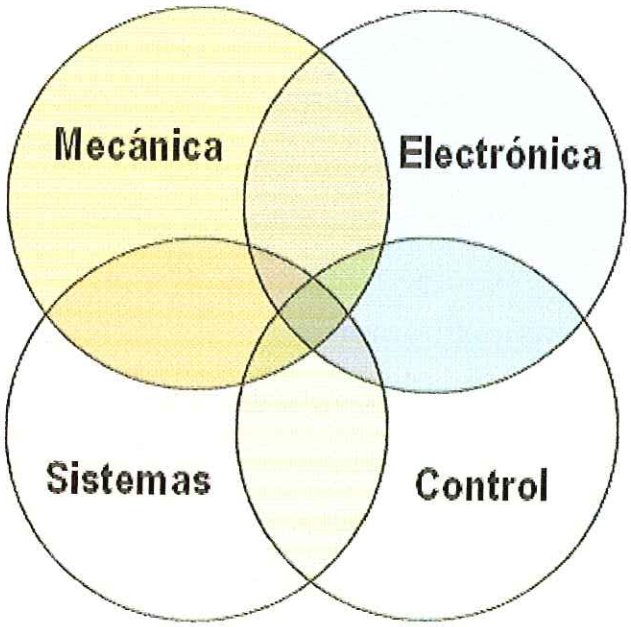


Fig. 1. Diseño Mecatrónico

El diseño mecatrónico tiene una visión integral en el diseño de productos, se basa en alta tecnología y es integral debido a que no se manejan procesos separados por barreras de Conocimientos diferentes, más bien se diseña íntegramente a partir de conocimientos fundamentales de ingenierías básicas.

1.1.2 Metodología de diseño propuesto

Esta es una propuesta de diseño mecatrónico del Dr. José Emilio Vargas Soto, la cual se tomará como base para el diseño del manipulador PUMA. En el siguiente esquema se muestra el diagrama de diseño, en el que se ve una clara aplicación de la integración sinérgica de los saberes básicos de las diferentes ingenierías:

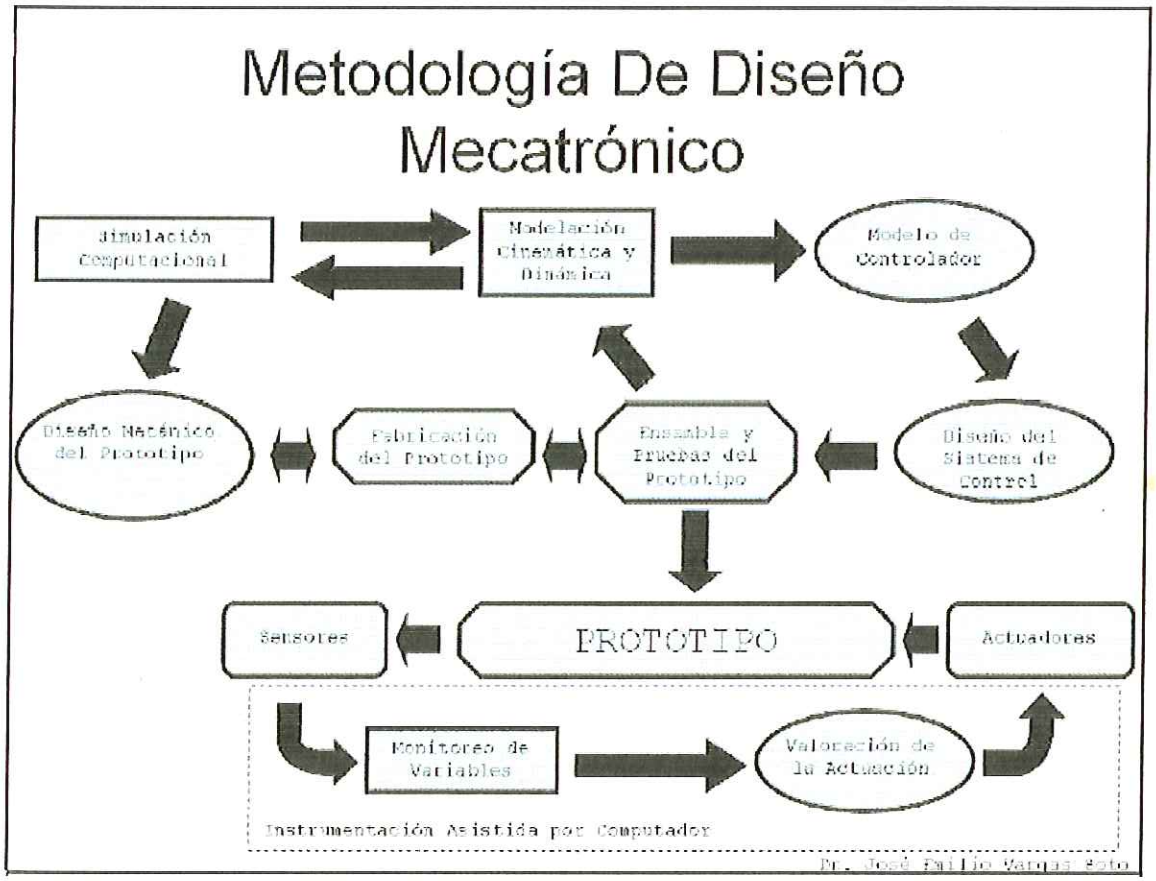


Figura 2. Metodología de diseño propuesto

1.2 METODOLOGÍA DE DISEÑO MECATRÓNICO GENERAL

La siguiente metodología de diseño corresponde al proceso de referencia de diseño mecatrónico en el que se parte de diseños, simulaciones y cálculos como base para el diseño real e implementación:

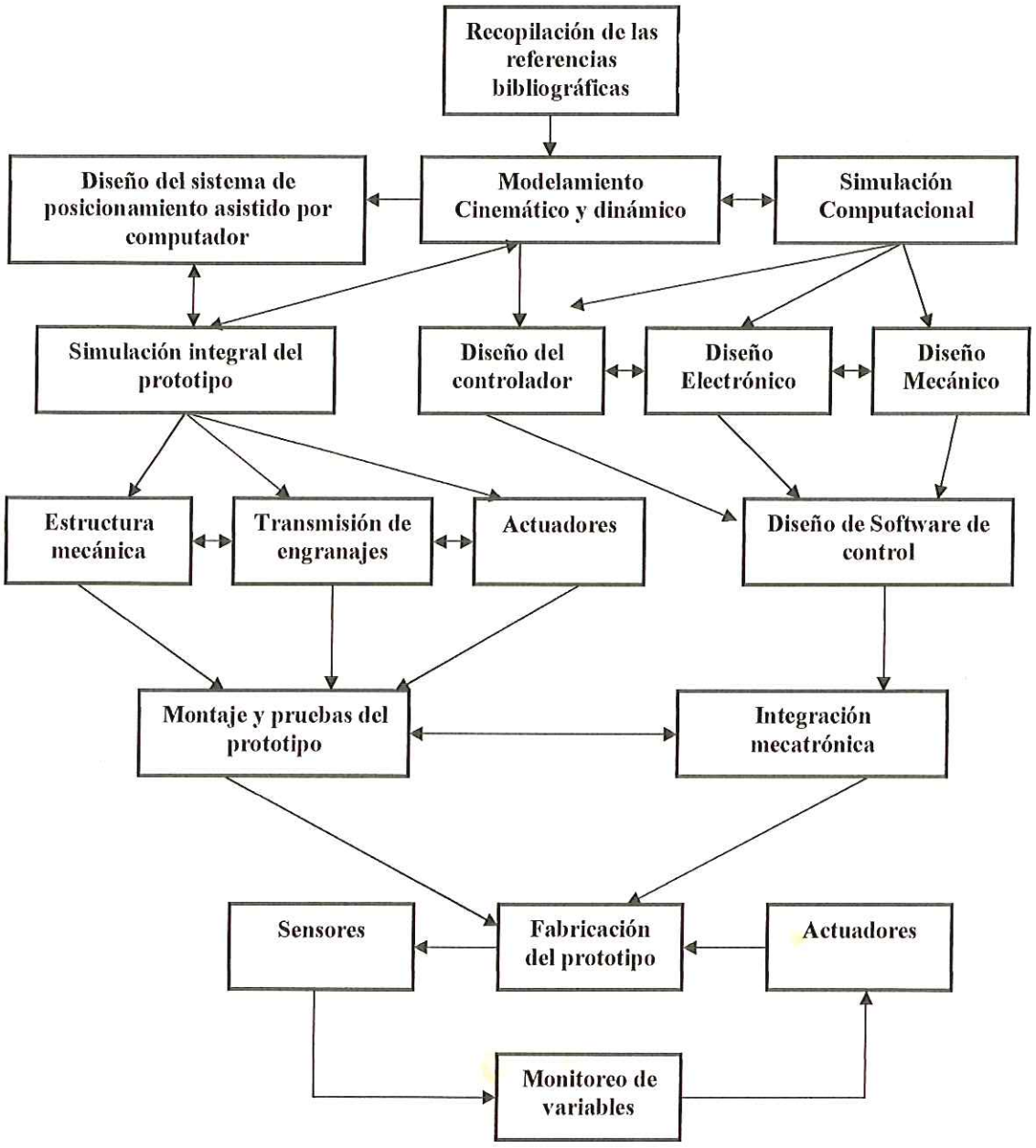


Figura 3. Metodología de diseño mecatrónico general

Este diseño está basado por el propuesto anteriormente y adaptado al prototipo del manipulador PUMA de manera general y tomado como referencia.

1.3 METODOLOGÍA DE DISEÑO MECATRÓNICO APLICADO

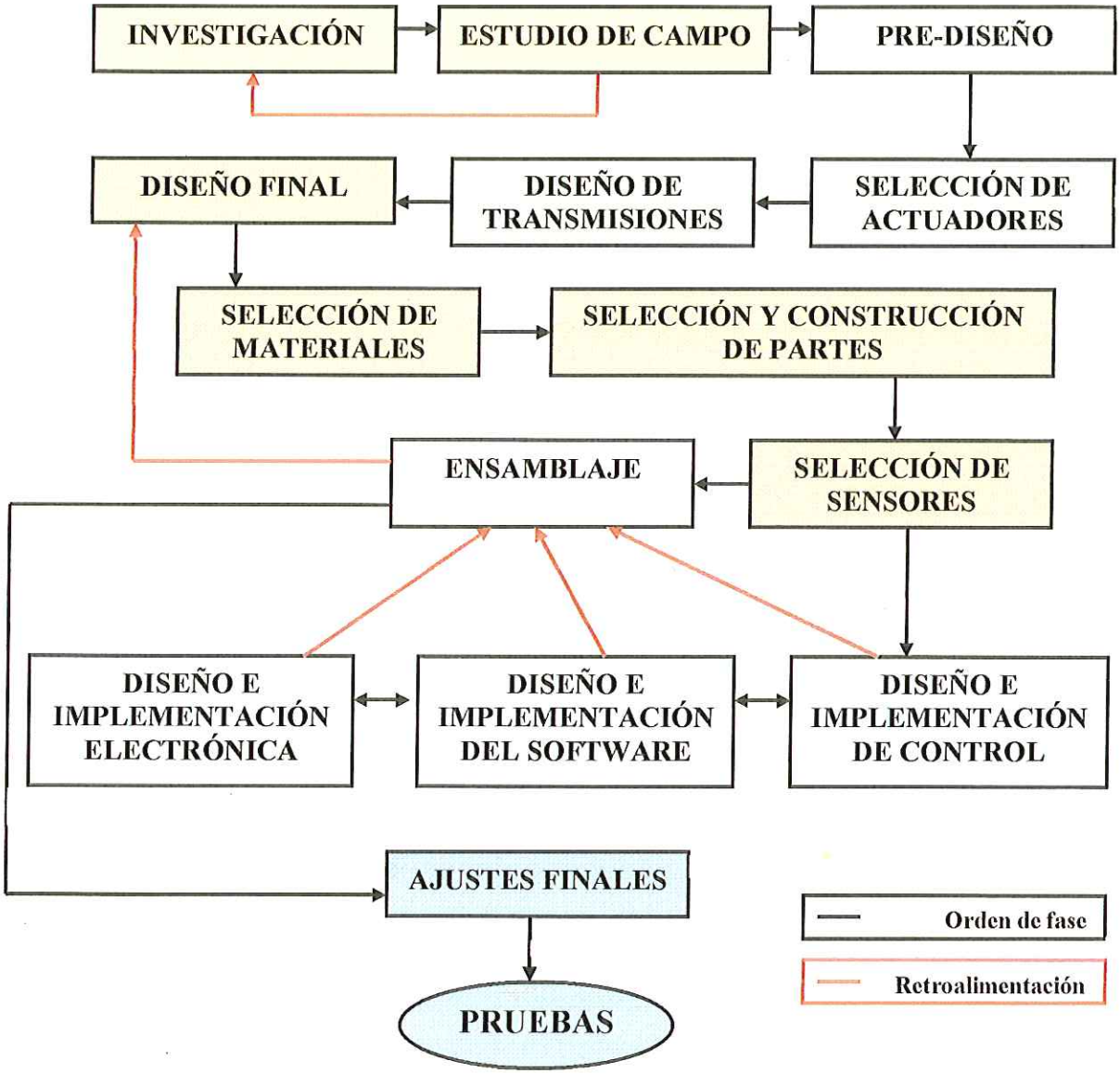


Figura 4. Proceso de diseño mecatrónico aplicado

1.3.1 Investigación:

Etapa en la cual se investigan los conceptos relacionados con el proyecto a realizar, en este caso, el manipulador PUMA, ya sea por medio de INTERNET o por fuentes bibliográficas correspondientes. Aquí se reúnen y se estudian todos los conceptos básicos y fundamentales que tienen que ver con el diseño del manipulador antes de realizar todo el proceso.

1.3.2 Estudio de campo:

Es también una fase importante y complementaria de la investigación. Es aquí donde se realiza una etapa de investigación a nuestro alrededor, ya sea por medio de personas experimentadas, maquinaria, tipos y formas de materiales existentes, estándares, etc. Permite tener un marco de diseño el cual se rige por los estándares actuales, específicamente en la localidad, de tal forma que nos permita saber que es posible hacer y que no. Se tienen listas de precios y lugares relacionados con el trabajo de diseño a realizar.

1.3.3 Pre-diseño:

Es un diseño general del manipulador, con un tamaño más grande de lo pensado realmente y con las partes fundamentales que conforman en diseño base, característico del manipulador PUMA. Aquí se debe tener en cuenta la etapa de investigación, por que esta da la pauta o el marco para diseñar de acuerdo a experiencias y estándares. Su tamaño es grande por que no se han tenido en cuenta aún los espacios que ocupan los mecanismos y actuadores, pero es una fase importante por que la respectiva simulación en CAD revela que fuerzas se van a manejar y que velocidades, de esta manera se pueden seleccionar el tipo de actuadores a usar. Los cálculos cinemáticos y dinámicos son importantes, pues son la base de los cálculos finales definitivos y contribuyen a la selección de las características de cada actuador.

1.3.4 Selección de actuadores:

Etapa en la cual se seleccionan los actuadores a usar, en este caso, motores de corriente directa, teniendo en cuenta los resultados obtenidos de las simulaciones respectivas realizadas en CAD. Con los cálculos cinemáticos y dinámicos se pueden hallar datos específicos para los motores y de esta manera saber más exactamente que tipo de motor usar. La etapa de Actuadores tiene que ver también con la elección correspondiente de las fuentes de alimentación para cada motor seleccionado.

1.3.5 Diseño de transmisiones mecánicas:

Después de la selección de motores para cada articulación del brazo mecánico se procede a diseñar el tren de engranajes correspondiente a la velocidad deseada para cada grado de libertad, esto depende de los parámetros iniciales del brazo y de las simulaciones hechas en el pre-diseño. En las trasmisiones mecánicas van comprendidos un número determinado de piñones los cuales deben cumplir con un torque y una velocidad especificados por los procesos anteriores, deben ir una serie de bujes y ejes para cada uno de estos piñones, los cuales van separados con unas distancias y distribución dependiente del diseño previo, deben tenerse en cuenta los rodamientos para las articulaciones, estos se rigen de unos estándares y deben seleccionarse antes de los ejes que van incrustados en ellos, pues los ejes pueden ajustarse a cualquier diámetro menor a él gracias a que puede ser torneado.

1.3.6 Diseño final:

Es la etapa final del diseño, finalmente se tienen en cuenta los espacios ocupados por los motores, trenes de engranajes, ejes y rodamientos para ajustar el pre-diseño a un diseño apropiado a los elementos hasta el momento contemplados. Se debe analizar el peso de todo el conjunto de piezas del brazo para hacer las

correcciones respectivas a los cálculos cinemáticos y dinámicos, logrando así el menor peso posible, conservando ante todo, la capacidad de carga de la estructura del manipulador. Las simulaciones se deben hacer con un poco más de peso, para que al obtener la pieza final no haya ningún problema de sobrecarga y se vean comprometidos los mecanismos ya diseñados, estos problemas se deben a que el material real no es a menudo de las características ideales, las variaciones que sufren las piezas al pasar por varios procesos, los fenómenos de contracción (en el caso de la fundición) y a que el diseño podría tener mejores capacidades de desempeño, pues ya que se simuló con cargas más altas de las esperadas en la realidad, el sistema puede tener mejores capacidades de carga o pude compensarse si por algún motivo el diseño real tiene más peso del que se tenía pensado. La inercia es otro problema, causado por las altas velocidades en las articulaciones, por eso generalmente se manejan bajas velocidades con motores cd, para evitar aceleraciones altas y asegurar buen torque.

1.3.7 Selección de materiales:

Tiene que ver con la fase de diseño físico, inmediatamente después del diseño en CAD anterior. Los materiales deben ser lo más livianos y resistentes posibles, por que el PUMA maneja cargas considerables, debido a su diseño, pues es muy versátil, y esto implica ligereza en su propia estructura, teniendo en cuenta que debe soportar cargas extras consideradas en los parámetros iniciales de diseño. No solo deben cumplir con bajo peso sino también con los requerimientos ante desgaste por fricción o resistencia al que puedan enfrentarse.

1.3.8 Selección y construcción de partes:

Cada una de las partes de la estructura del brazo debe ser analizada para decidir si adaptar piezas similares ya hechas o construirla. Piezas básicas como tubos, cajones, láminas, son básicamente obtenidas de las que ya existen, obviamente estas cumplen con unos estándares, los cuales ya fueron concebidos en la etapa

de diseño.

Lo contrario con piezas complejas, las cuales fueron diseñadas para contener un espacio definido y dimensiones específicas, son estas las que deben ser construidas a partir del diseño en CAD y generalmente se hacen por fundición, aquí entra en juego la selección de materiales realizada anteriormente.

1.3.9 Selección de sensores:

Corresponde directamente con la función a realizar. El manipulador puede requerir de sensores como: presión (para el *gripper* o elemento Terminal de agarre), presencia (para efectuar el agarre una ves el gripper detecte el objeto a mover) o posición (potenciómetros o *encoders*), pero deben ser seleccionados de acuerdo a la tarea a realizar y a su aporte al rendimiento.

1.3.10 Ensamblaje:

Después de esta etapa de selección se tienen finalmente todos los elementos que componen el brazo y se prosigue al ensamblaje de estos mismos. En el ensamblaje entra en juego la soldadura y los tornillos.

La soldadura es especial para cada tipo de material y los tornillos adecuados para cada superficie. Los mecanismos como engranajes y rodamientos por balines deben ser correctamente lubricados, con el fin de protegerlos contra la oxidación, hacer los mecanismos más suaves y disminuir el sonido que producen. Los procesos hasta el momento pueden causar cambios en el diseño real lo que conlleva a retroalimentar o actualizar el diseño en CAD, para así poder simular el diseño real por software.

1.3.11 Diseño e implementación Electrónica:

Tiene que ver con los circuitos, microcontroladores y *drivers* a usar. Se debe destinar un espacio en la estructura para los circuitos, en el caso de que se haga de forma interna, como en el caso de este proyecto. Se deben seleccionar que o cuales tipos de controlador usar, ya sean microcontroladores o PC, y los drivers para cada motor. La estructura debe tener los orificios por donde va el cableado interno, este debe comunicar y alimentar todos los circuitos.

1.3.12 Diseño e implementación del Software:

Es la etapa de programación tanto para microcontroladores como para software de computador. Se debe desarrollar un diagrama de flujo el cual contiene todas las tareas a realizar de forma general, las cuales se refieren a la tarea que vaya a realizar el manipular, para este caso, trasladar objetos de un punto a otro, a partir de esto se hace la programación a fondo de cada elemento programable. Se deben usar interfaces de protección entre el circuito y el PC o microcontrolador, para que no se vean involucrados por efectos indeseables producidos por los motores a usar.

1.3.13 Diseño e implementación del control

Para el sistema de control se implementó un control en lazo cerrado. Debido a que los motores CD manejan altas velocidades, se hace difícil implementar sistemas de control de lazo abierto y posicionarlo de manera exacta y confiable, pero al implementar un sistema de retroalimentación por medio de potenciómetros ubicados en cada una de las articulaciones, se hace posible tener un posicionamiento correcto y grandes velocidades, las cuales se pueden aplicar a trenes de engranajes robustos. Los potenciómetro en todo momento informan al sistema de control de la posición de cada articulación, a diferencia de los *encoders*, y se hace posible decidir en que sentido hacer girar el motor a partir de

la posición actual del potenciómetro. Los potenciómetros implementados al manipulador son lineales a 5kΩ. La linealidad en una característica absolutamente importante, pues, la posición es directamente proporcional al nivel de voltaje que este provee (0 a 5V), es así como se asegura un posicionamiento uniforme. De esta forma el programa de control puede activar determinado motor hasta que en el potenciómetro se llegue a la posición deseada.

Los sistemas de control basados en microcontroladores y ordenador, manejan señales digitales, por tanto se aplica un control en tiempo discreto al sistema. Como las salidas de estos dispositivos producen cambios en las articulaciones del manipulador de manera secuencial, se habla de un control por estados, estos se analizarán más adelante con el sistema de grafos de estado.

1.3.14 Ajustes finales:

Se hacen con el fin de que todo funcione sincronizada y correctamente, puede ocasionar cambios a nivel electrónico y programable, como también de cambios físicos mínimos. Tiene que ver también con la estética, los colores, acabados superficiales en la estructura, periféricos de control y observación de procesos, controles a distancia, accesorios, etc.

1.3.15 Pruebas:

Etapa en la cual se ponen a trabajar todos los elementos desarrollados hasta el momento para que trabajen en conjunto. Se hacen pruebas referentes a las tareas a realizar por el manipulador y se descubren capacidades adicionales que este es capas de desarrollar no conocidas con anterioridad.

1.4 CINEMÁTICA

1.4.1 Cinemática directa

Se utiliza fundamentalmente el álgebra vectorial y matricial para representar y describir la localización de un objeto en el espacio tridimensional con respecto aun sistema de referencia fijo.

Dado que un manipulador se puede considerar como una cadena cinemática formada por objetos rígidos o eslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia.

De esta forma, el problema cinematico directo se reduce a encontrar una matriz homogénea de transformación T que relacione la posición y orientación del extremo del robot respecto del sistema de referencia fijo situado en la base del mismo. Esta matriz T será función de las coordenadas articulares.

Teniendo en cuenta el algoritmo de *Denavit Hartenberg*; la resolución para el problema cinemático del manipulador Puma es la siguiente:

Parámetros D.H: Con estos parámetros se calculan las relaciones entre los eslabones consecutivos del robot.

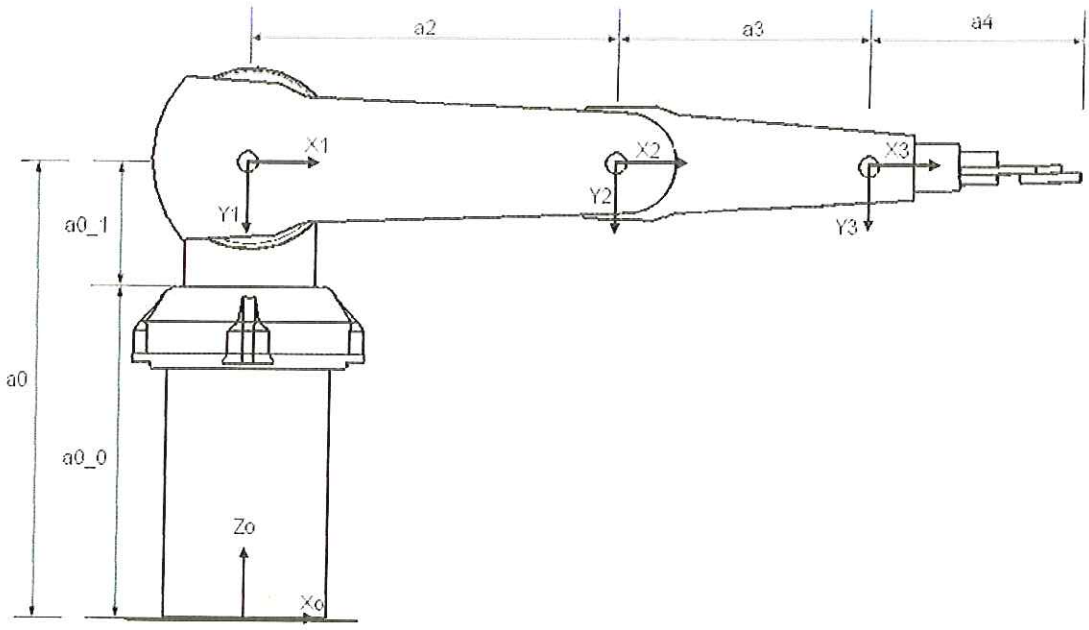


Figura 5. manipulador puma, cinemática

$$a_1 = 0$$

Articulación	α_i	a_i	d_i	θ_i
1	$-\frac{\pi}{2}$	0	a_0	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3
4	0	a_4	0	θ_4

Tabla 1.D-H Manipulador

Una vez hallados los parámetros de cada eslabón, se comienza a calcular las matrices ***A:***

$$A_1 = \begin{vmatrix} c\theta_1 & 0 & -s\theta_1 & 0 \\ s\theta_1 & 0 & c\theta_1 & 0 \\ 0 & -1 & 0 & a_0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$A_2 = \begin{vmatrix} c\theta_2 & -s\theta_2 & 0 & a_2 \cdot c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2 \cdot s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$A_3 = \begin{vmatrix} c\theta_3 & -s\theta_3 & 0 & a_3 \cdot c\theta_3 \\ s\theta_3 & c\theta_3 & 0 & a_3 \cdot s\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$A_4 = \begin{vmatrix} c\theta_4 & -s\theta_4 & 0 & a_4 \cdot c\theta_4 \\ s\theta_4 & c\theta_4 & 0 & a_4 \cdot s\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Realizando el producto entre matrices, se obtiene la matriz de transformación ***T*** que indica la localización del sistema final con respecto al sistema de referencia de la base del robot.

$$T = [A_1 \cdot A_2 \cdot A_3 \cdot A_4]$$

$$T = \begin{vmatrix} \frac{c(\theta_1 - \theta_2 - \theta_3 - \theta_4) + c(\theta_1 + \theta_2 + \theta_3 + \theta_4)}{2} & \frac{s(\theta_1 - \theta_2 - \theta_3 - \theta_4) - s(\theta_1 + \theta_2 + \theta_3 + \theta_4)}{2} & -s\theta_1 \\ \frac{s(\theta_1 - \theta_2 - \theta_3 - \theta_4) + s(\theta_1 + \theta_2 + \theta_3 + \theta_4)}{2} & \frac{-c(\theta_1 - \theta_2 - \theta_3 - \theta_4) - c(\theta_1 + \theta_2 + \theta_3 + \theta_4)}{2} & c\theta_1 \\ -s(\theta_2 + \theta_3 + \theta_4) & -c(\theta_2 + \theta_3 + \theta_4) & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

$$\begin{vmatrix} a_3 \cdot c\theta_1 \cdot c\theta_2 + a_3 \cdot c\theta_1(c\theta_2 \cdot c\theta_3 + s_2 \cdot s\theta_3) + a_4 \cdot c\theta_1 \cdot [c\theta_2 \cdot (c\theta_3 \cdot c\theta_4 - s\theta_3 \cdot s\theta_4 + s\theta_2(-c\theta_3 \cdot s\theta_4 - s\theta_3 \cdot c\theta_4))] \\ a_2 \cdot s\theta_1 \cdot c\theta_2 + a_3 \cdot s\theta_1(c\theta_2 \cdot c\theta_3 - s_2 \cdot s\theta_3) + a_4 \cdot s\theta_1 \cdot [c\theta_2 \cdot (c\theta_3 \cdot c\theta_4 - s\theta_3 \cdot s\theta_4 + s\theta_2(-c\theta_3 \cdot s\theta_4 - s\theta_3 \cdot c\theta_4))] \\ a_0 - a_2 \cdot s\theta_2 - a_4 \cdot s(\theta_2 + \theta_3 + \theta_4) + a_3(-s(\theta_2 + \theta_3)) \\ 1 \end{vmatrix}$$

1.4.2 Cinemática inversa.

El objetivo del problema cinematico inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot q=(q1, q2,..., qn) expresado en T para que su extremo se posicione y oriente según una determinada localización espacial.

Así como es posible abordar el problema cinematico directo de una manera sistemática a partir de la utilización de matrices de transformación homogéneas, 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.

El método utilizado para determinar los valores de las tres primeras articulaciones es el método geométrico. El procedimiento consiste en encontrar relaciones geométricas entre las coordenadas del extremo del manipulador, sus coordenadas articulares y las dimensiones de sus eslabones. La ventaja de utilizar este método es que se encuentra una solución cerrada para la cinemática inversa.

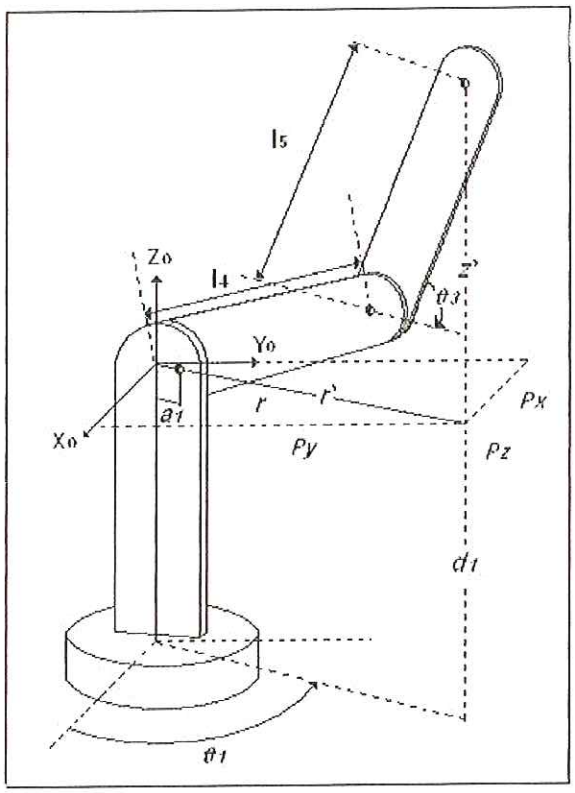


Figura 6. manipulador, cinemática inversa

Obtención de Θ_1 (Angulo Primera Articulación):

$$\theta_1 = arctg\left(\frac{Py}{Px}\right)$$

Obtención de Θ_3 (Angulo Tercera Articulación):

Relaciones Geométricas:

$$r = a_1 + r' \Rightarrow r' = r - a_1 \quad (11)$$

$$Pz = d_1 + z' \Rightarrow z' = Pz - d_1 \quad (12)$$

$$r'^2 = Px^2 + Py^2 \quad (13)$$

Aplicando el teorema del coseno se obtiene:

$$r''^2 + z'^2 \Rightarrow L_4^2 + L_5^2 + 2 \cdot L_4 \cdot L_5 \cdot \cos\theta_3 \quad (14)$$

$$\cos\theta_3 = \frac{r''^2 + z'^2 - L_4^2 - L_5^2}{2 \cdot L_4 \cdot L_5} \quad (15)$$

Sustituyendo las ecuaciones (11) y (12) en (15) se obtiene:

$$\cos\theta_3 = \frac{(r - a_1)^2 + (Pz - d_1)^2 - L_4^2 - L_5^2}{2 \cdot L_4 \cdot L_5} \quad (16)$$

Resolviendo los cuadrados se obtiene:

$$\cos\theta_3 = \frac{r'^2 - 2ra_1 + a_1^2 + Pz^2 - 2Pzd_1 + d_1^2 - L_4^2 - L_5^2}{2 \cdot L_4 \cdot L_5} \quad (17)$$

Sustituyendo la ecuación (13) se obtiene:

$$\cos\theta_3 = \frac{Px^2 + Py^2 - 2\sqrt{Px^2 + Py^2}a_1 + a_1^2 + Pz^2 - 2 \cdot Pz \cdot d_1 + d_1^2 - L_4^2 - L_5^2}{2 \cdot L_4 \cdot L_5} \quad (18)$$

$$\cos^2\theta + sen^2\theta = 1 \Rightarrow sen^2\theta = 1 - \cos^2\theta \Rightarrow sen\theta = \pm\sqrt{1-\cos^2\theta} \quad (19)$$

$$\frac{sen\theta_3}{\cos\theta_3} = \frac{\pm\sqrt{1-\cos^2\theta_3}}{\dfrac{Px^2 + Py^2 - 2\cdot\sqrt{Px^2+Py^2}\,a_1 + a_1^{\,2} + Pz^2 - 2\cdot Pz\cdot d_1 + d_1^{\,2} - L_4^2 - L_5^2}{2\cdot L_4\cdot L_5}} \quad (20)$$

$$\theta_3 = arctag\left(\frac{\pm\sqrt{1-\left(\dfrac{Px^2 + Py^2 - 2\cdot\sqrt{Px^2+Py^2}\,a_1 + a_1^{\,2} + Pz^2 - 2\cdot Pz\cdot d_1 + d_1^{\,2} - L_4^2 - L_5^2}{2\cdot L_4\cdot L_5}\right)^2}}{\dfrac{Px^2 + Py \;\; - 2\cdot\sqrt{Px^2+Py^2}\,a_1 + a_1^{\,2} + Pz^2 - 2\cdot Pz\cdot d_1 + d_1^{\,2} - L_4^2 - L_5^2}{2\cdot L_4\cdot L_5}}\right)$$

(21)

Obtención de Θ_2 (Angulo Segunda Articulación):

$$\theta_2 = \beta - \alpha \quad (22)$$

$$\beta = arctag\left(\frac{z'}{r'}\right) \quad (23)$$

Geometría para determinar Angulo Segunda Articulación

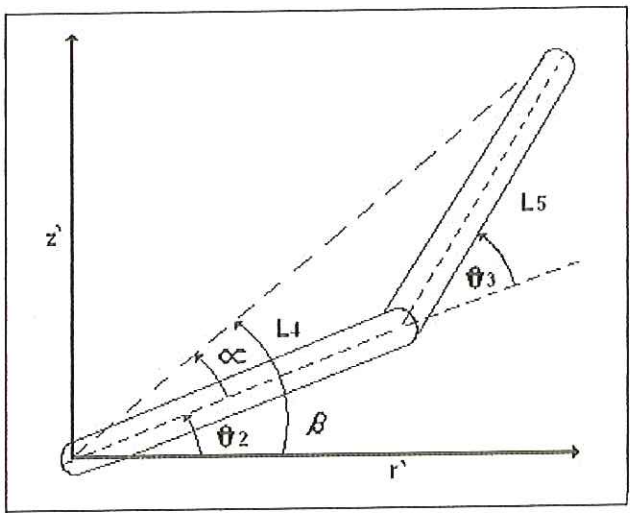


Figura 7. ángulos, cinemática inversa

Del Autor

De la ecuación (12) se obtiene:

$$z' = Pz - d_1 \Leftrightarrow d_1 = L_1 + L_2 \quad (23)$$

$$r' = r - a_1 \Leftrightarrow a_1 = L_3 \quad (24)$$

$$\beta = arctag\left(\frac{Pz - d_1}{r - a_1}\right) \quad (25)$$

Sustituyendo r se obtiene:

$$\beta = arctag\left(\frac{Pz - d_1}{\pm\sqrt{Px^2 + Py^2} - a_1}\right) \quad (26)$$

Del teorema del coseno se puede obtener:

$$x^2 = L_4^2 + L_5^2 - 2L_4L_5\cos(90-\theta_3) \Rightarrow x = \sqrt{L_4^2 + L_5^2 + 2L_4L_5\cos(\theta_3)} \quad (27)$$

Del teorema del seno se obtiene:

$$\frac{sen\alpha}{L_5} = \frac{sen(90-\theta_3)}{\sqrt{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3}} \quad (28)$$

$$sen(90-\theta_3) = sen\theta_3 \quad (29)$$

$$sen\alpha = \frac{L_5 sen\theta_3}{\sqrt{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3}} \quad (30)$$

$$sen^2\alpha + \cos^2\alpha = 1 \quad (31)$$

$$\cos\alpha = \sqrt{1 - sen^2\alpha} \quad (32)$$

$$\cos\alpha = \sqrt{1 - \left(\frac{L_5 sen\theta_3}{\sqrt{L_4^2 + L_5^2 + 2L_4L_5\cos(\theta_3)}}\right)^2} \quad (33)$$

$$\cos\alpha = \sqrt{1 - \frac{L_5{}^2 sen^2\theta_3}{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3}} \quad (34)$$

$$\cos\alpha = \sqrt{\frac{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3 - L_5{}^2 sen^2\theta_3}{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3}} \quad (35)$$

$$\frac{sen\alpha}{\cos\alpha} = \frac{\frac{L_5 sen\theta_3}{\sqrt{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3}}}{\sqrt{\frac{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3 - L_5{}^2 sen^2\theta_3}{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3}}} \quad (36)$$

$$\frac{sen\alpha}{\cos\alpha} = \frac{L_5 sen\theta_3}{\sqrt{L_4^2 + L_5^2 + 2L_4L_5\cos\theta_3 - L_5{}^2 sen^2\theta_3}} \quad (37)$$

$$\frac{sen\alpha}{\cos\alpha} = \frac{L_5 sen\theta_3}{\sqrt{L_4^2 + 2L_4L_5\cos\theta_3 + L_5{}^2\left(1 - sen^2\theta_3\right)}} \quad (38)$$

$$\cos^2\theta_3 = \left(1 - sen^2\theta_3\right) \quad (39)$$

$$\frac{sen\alpha}{\cos\alpha} = \frac{L_5 sen\theta_3}{\sqrt{L_4^2 + 2L_4L_5\cos\theta_3 + L_5{}^2\cos^2\theta_3}} \quad (40)$$

$$\frac{sen\alpha}{\cos\alpha} = \frac{L_5 sen\theta_3}{\sqrt{(L_4 + L_5\cos\theta_3)^2}} \quad (41)$$

$$tag\alpha = \frac{L_5 sen\theta_3}{L_4 + L_5\cos\theta_3} \quad (42)$$

$$\alpha = arctag\left(\frac{L_5 sen\theta_3}{L_4 + L_5\cos\theta_3}\right) \quad (43)$$

$$\theta_2 = \beta - \alpha \quad (44)$$

$$\theta_2 = arctag\left(\frac{Pz - d_1}{\pm\sqrt{Px^2 + Py^2} - a_1}\right) - arctag\left(\frac{L_5 sen\theta_3}{L_4 + L_5\cos\theta_3}\right) \quad (45)$$

1.4.3 Desacoplo Cinemático

Los procedimientos vistos en los apartados anteriores permiten obtener los valores de las 3 primeras variables articulares del robot, aquellas que posicionan su extremo en las coordenadas (Px, Py, Pz) determinadas, aunque pueden ser igualmente utilizadas para la obtención de las siguiente articulaciones 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 aquel 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 3 últimos ejes (muñeca del robot) calculándose los valores de las tres primeras variables articulares (q1, q2, q3) que consiguen posicionar este punto. A continuación, a partir de los datos de orientación y de los ya calculados (q1, q2, q3) obtiene los valores del resto de las variables articulares.

1.5 JACOBIANO

El modelo cinematico del manipulador puma busca las relaciones entre las variables articulares, la posición y orientación del extremo del robot.
En esta relación no se tienen en cuenta las fuerzas o pares que actúan sobre el manipulador (actuadores, cargas, fricciones, etc.) y que pueden originar el movimiento del mismo.

Sin embargo, debe permitir conocer, además de la relación entre las coordenadas articulares y del extremo, la relación entre sus respectivas derivadas. Así, el sistema de control del robot debe establecer las velocidades a cada articulación (a través de sus respectivos actuadores) para conseguir que el extremo desarrolle una trayectoria temporal concreta.

Para este y otros fines, es de gran utilidad disponer de la relación entre las velocidades de las coordenadas articulares y las de posición y orientación del extremo del robot. La relación entre ambos vectores de velocidad se obtiene a través de la denominada matriz Jacobiana.

La matriz jacobiana del manipulador relaciona las velocidades articulares con las velocidades cartesianas del extremo

la matriz Jacobiana inversa permitirá conocer las velocidades determinadas en el extremo del robot.

1.5.1 Matriz jacobiana directa

La matriz jacobiana directa permite conocer las velocidades del extremo del manipulador a partir de los valores de las velocidades de cada articulación.

Primero de halla los vectores de posición de los centros de masa de cada eslabón con respecto a la base.

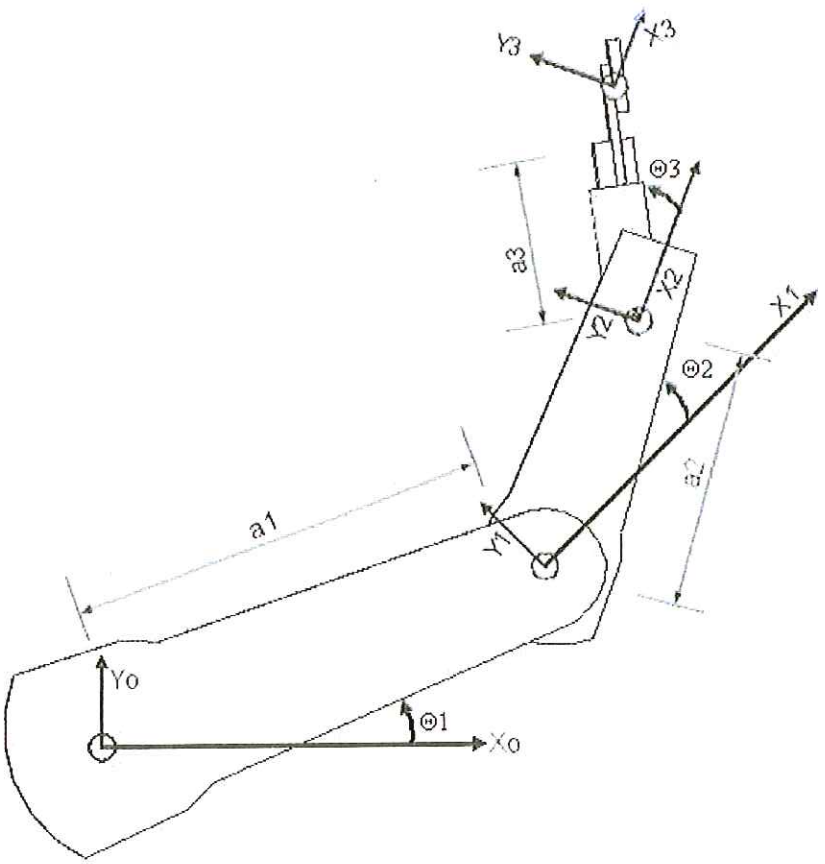


Figura 8. Manipulador, Jacobiano

$$ {}^{2}P_{3}^{*} = \begin{vmatrix} a_3 \cdot c\theta_{123} \\ a_3 \cdot s\theta_{123} \\ 0 \end{vmatrix} $$

$$ {}^{1}P_3^* = \begin{vmatrix} a_2 \cdot c\theta_{12} + a_3 \cdot c\theta_{123} \\ a_2 \cdot s\theta_{12} + a_3 \cdot s\theta_{123} \\ 0 \end{vmatrix} $$

$$ {}^{0}P_3^* = \begin{vmatrix} a_1 \cdot c\theta_1 + a_2 \cdot \theta_{12} + a_3 \cdot c\theta_{123} \\ a_1 \cdot s\theta_1 + a_2 \cdot s\theta_{12} + a_3 \cdot s\theta_{123} \\ 0 \end{vmatrix} $$

Donde $\theta_{12} = \theta_1 + \theta_2$ y $\theta_{123} = \theta_1 + \theta_2 + \theta_3$

$$ \begin{vmatrix} v_x \\ u_y \\ w_z \end{vmatrix} = J \begin{vmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{vmatrix} $$

Donde

$$ J = \begin{vmatrix} -(a_1 \cdot s\theta_1 + a_2 \cdot s\theta_{12} + a_3 \cdot s\theta_{123}) & -(a_2 \cdot s\theta_{12} + a_3 \cdot s\theta_{123}) & -a_3 \cdot s\theta_{123} \\ (a_1 \cdot c\theta_1 + a_2 \cdot c\theta_{12} + a_3 \cdot c\theta_{123}) & (a_2 \cdot c\theta_{12} + a_3 \cdot c\theta_{123}) & (a_3 \cdot c\theta_{123}) \\ 1 & 1 & 1 \end{vmatrix} $$

Si escogemos como punto de referencia el origen del gripper, en (X2,Y2) la matriz jacobiana se reduce a:

$$J=\begin{vmatrix} -(a_1\cdot s\theta_1+a_2\cdot s\theta_{12}) & -(a_2\cdot s\theta_{12}) & 0 \\ (a_1\cdot c\theta_1+a_2\cdot c\theta_{12}) & (a_2\cdot c\theta_{12}) & 0 \\ 1 & 1 & 1 \end{vmatrix}$$

1.5.2 Matriz jacobiana inversa

$$J^{-1}=\begin{vmatrix} -(a_1\cdot s\theta_1+a_2\cdot s\theta_{12}) & -(a_2\cdot s\theta_{12}) & 0 \\ (a_1\cdot c\theta_1+a_2\cdot c\theta_{12}) & (a_2\cdot c\theta_{12}) & 0 \\ 1 & 1 & 1 \end{vmatrix} \cdot \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{vmatrix}$$

1.6 DINAMICA

La Formulación Lagrangiana permite describir la dinámica del manipulador puma a partir de un balance de energía. Desde este punto de vista el manipulador se considera como una caja negra.

Las ecuaciones tienen únicamente en cuenta la energía almacenada, que se expresa en términos de energía cinética y potencial.

El Langrangiano es una función escalar que se define como la diferencia entre las energías cinética y potencial de un sistema mecánico, en función de las llamadas coordenadas generalizadas; que para nuestro caso son las variables articulares.

Este método conduce a unas ecuaciones finales bien estructuradas donde aparecen de manera clara los diversos pares y fuerzas que intervienen en el movimiento (inercia, Coriolis, gravedad).

1.6.1 Formulación de lagrange-euler

TABLA DE VARIABLES

G	Vector de fuerzas gravitacionales.
I_i	Matriz de Inercia de cada eslabón con respecto a la base.
J_i	Matriz Jacobiana de cada eslabón.
J_{vi}	Elementos de la Matriz Jacobiana asociados con la velocidad lineal del centro de masa de cada eslabón.
J_{wi}	Elementos de la Matriz Jacobiana asociados con el vector de velocidad angular de cada eslabón.
L	Función de Lagrange, $L = K - U$
M	Matriz de Inercia del manipulador.
M_{ij}	Elementos de la matriz de Inercia.
${}^{k}P_{ci}^{*}$	Vector de posición de los centros de masa de cada eslabón con respecto a la base.
Q_i	Fuerzas generalizadas activas.
Q	Vector de fuerzas generalizadas, $Q = [Q_1, Q_2 \ldots, Q_n]^T$
q_i	Coordenadas generalizadas
q	Vector de coordenadas generalizadas, $q = [q_1, q_2 \ldots, q_n]^T$
U	Energía potencial de un Sistema Mecánico
V	Vector velocidad
K	Energía cinética de un Sistema Mecánico

A continuación se describe el modelo dinámico del manipulador puma mediante Lagrange-Euler

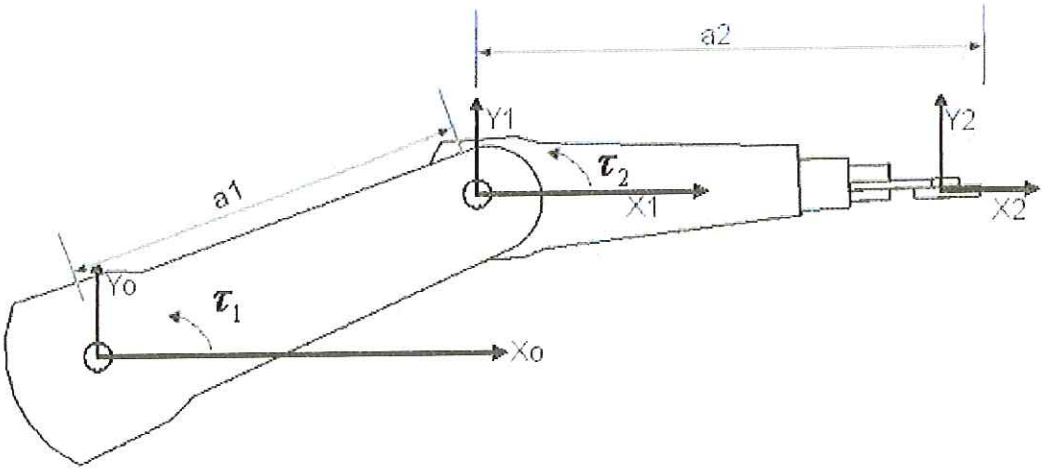


Figura 9. Manipulador dinámica

Matrices de inercia de las articulaciones

Se asume que los movimientos en las articulaciones son homogéneas y que el centro de masa se encuentra en el centro de cada articulación

$$I_i = \frac{1}{12} \cdot m_i \cdot a_i^2 \cdot \begin{vmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

De la ecuación anterior sacamos las matrices de inercia para las articulaciones:

$$I_1 = \frac{1}{12} \cdot m_1 \cdot a_1^2 \cdot \begin{vmatrix} s^2\theta_1 & -s\theta_1 \cdot c\theta_1 & 0 \\ -s\theta_1 \cdot \cos\theta_1 & c^2\theta_1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$I_2 = \frac{1}{12} \cdot m_2 \cdot a_2^2 \cdot \begin{vmatrix} s^2\theta_{12} & -s\theta_{12} \cdot c\theta_{12} & 0 \\ -s\theta_{12} \cdot c\theta_{12} & c^2\theta_{12} & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Se construye la Matriz Jacobiana teniendo en cuenta los vectores de posición de los centros de masa de cada eslabón respecto a su movimiento con el origen (Xo,Yo):

$$^0P_{c1}^* = \begin{vmatrix} \frac{1}{2} \cdot a_1 \cdot c\theta_1 \\ \frac{1}{2} \cdot a_1 \cdot s\theta_1 \\ 0 \end{vmatrix}$$

$$^1P_{c2}^* = \begin{vmatrix} \frac{1}{2} \cdot a_2 \cdot c\theta_{12} \\ \frac{1}{2} \cdot a_2 \cdot s\theta_{12} \\ 0 \end{vmatrix}$$

$$^0P_{c2}^* = \begin{vmatrix} a_1 \cdot c\theta_1 + \frac{1}{2} \cdot a_2 \cdot c\theta_{12} \\ a_1 \cdot s\theta_1 + \frac{1}{2} \cdot a_2 \cdot s\theta_{12} \\ 0 \end{vmatrix}$$

Se conforma cada elemento de la Matriz Jacobiana mediante valores de velocidad instantánea de cada centro de masa y la velocidad angular de cada eslabón:

$$J_{v1} = \begin{vmatrix} -\frac{1}{2} \cdot a_1 \cdot s\theta_1 & 0 \\ \frac{1}{2} \cdot a_1 \cdot c\theta_1 & 0 \\ 0 & 0 \end{vmatrix}$$

$$J_{w1} = \begin{vmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{vmatrix}$$

$$J_{v2} = \begin{vmatrix} -a_1 \cdot s\theta_1 - \frac{1}{2} \cdot a_2 \cdot s\theta_{12} & -\frac{1}{2} \cdot a_2 \cdot s\theta_{12} \\ a_1 \cdot c\theta_1 + \frac{1}{2} \cdot a_2 \cdot c\theta_{12} & \frac{1}{2} \cdot a_2 \cdot c\theta_{12} \\ 0 & 0 \end{vmatrix}$$

$$J_{w2} = \begin{vmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 2 \end{vmatrix}$$

Se obtiene la Matriz de Inercia del manipulador de la forma

$$M = J_{v1}^{T} \cdot m_1 J_{v1} + J_{w1}^{T} \cdot I_1 J_{w1} + J_{v2}^{T} \cdot m_2 J_{v2} + J_{w2}^{T} \cdot I_2 J_{w2}$$

$$M = \begin{vmatrix} \frac{1}{3} \cdot m_1 \cdot a_1 + m_2 \cdot \left(a_1^2 + a_1 \cdot a_2 \cdot c\theta_2 + \frac{1}{3} \cdot a_2^2 \right) & m_2 \cdot \left(\frac{1}{2} \cdot a_1 \cdot a_2 \cdot c\theta_2 + \frac{1}{3} a_2^2 \right) \\ m_2 \cdot \left(\frac{1}{2} \cdot a_1 \cdot a_2 \cdot c\theta_2 + \frac{1}{3} a_2^2 \right) & \frac{1}{3} \cdot m_2 \cdot a_2^2 \end{vmatrix}$$

Se deriva parcialmente la Matriz de Inercia del manipulador con respecto a θ las fuerzas de inercia de cada eslabón; para obtener las velocidades de ajuste

$$V_1 = \sum_{j=1}^{3} \sum_{k=1}^{3} \left(\frac{\partial M_{1j}}{\partial \theta_k} - \frac{1}{2} \cdot \frac{\partial M_{jk}}{\partial \theta_1} \right) \cdot \dot{\theta}_j \cdot \dot{\theta}_j$$

$$V_1 = -m_2 \cdot a_1 \cdot a_2 \cdot s\theta_2 \cdot \left(\dot{\theta}_1 \cdot \dot{\theta}_2 + \frac{1}{2} \cdot \dot{\theta}_2^2 \right)$$

$$V_2 = \sum_{j=1}^{2} \sum_{k=1}^{2} \left(\frac{\partial M_{2j}}{\partial \theta_k} - \frac{1}{2} \cdot \frac{\partial M_{jk}}{\partial \theta_2} \right) \cdot \dot{\theta}_j \cdot \dot{\theta}_k$$

$$V_2 = \frac{1}{2} m_2 \cdot a_1 \cdot a_2 \cdot s\theta_2 \cdot \dot{\theta}_1^2$$

Por ultimo se calcula el vector de fuerzas gravitacionales:

$$G_1 = -\sum_{j=1}^{2} m_j \cdot g^T \cdot J_{vj}^1$$

$$G_1 = \frac{1}{2} m_1 \cdot g_c \cdot a_1 \cdot c\theta_1 + m_2 \cdot g_c \cdot a_1 \cdot c\theta_1 + \frac{1}{2} m_2 \cdot g_c \cdot a_2 \cdot c\theta_{12}$$

$$G_2 = -\sum_{j=1}^{2} m_j \cdot g^T \cdot J_{vj}^1$$

$$G_2 = \frac{1}{2} m_2 \cdot g_c \cdot a_2 \cdot c\theta_{12}$$

$$\tau_1 = \left[\left(\frac{1}{3} \cdot m_1 + m_2 \right) \cdot a_1^2 + m_2 \cdot a_1 \cdot a_2 \cdot c\theta_2 + \frac{1}{3} \cdot m_2 a_2^2 \right] \cdot \ddot{\theta}_1$$

$$+ \left(\frac{1}{2} \cdot m_2 a_1 a_2 c\theta_2 + \frac{1}{3} \cdot m_2 a_2^2 \right) \ddot{\theta}_2 - m_2 \cdot a_1 \cdot a_2 \cdot s\theta_2 \left(\dot{\theta}_1 \cdot \dot{\theta}_2 + \frac{1}{2} \dot{\theta}_2^2 \right)$$

$$+ g_c \left[\left(\frac{1}{2} \cdot m_1 + m_2 \right) \cdot a_1 \cdot c\theta_1 + \frac{1}{2} \cdot m_2 a_2 \cdot c\theta_{12} \right]$$

$$\tau_2 = \left(\frac{1}{2}\cdot m_2 a_1 a_2 c\theta_2 + \frac{1}{3}\cdot m_2 a_2^2\right)\ddot{\theta}_1 - \frac{1}{3}\cdot m_2 \cdot a_2^2 \cdot \ddot{\theta}_2 + \frac{1}{2} m_2 \cdot a_1 \cdot a_2 \cdot s\theta_2 \cdot \dot{\theta}_1^2$$

$$+ \frac{1}{2} m_2 \cdot g_c \cdot a_2 \cdot c\theta_{12}$$

1.7 ESTRUCTURA

Las características más importantes que se tuvieron en cuenta para el diseño mecánico del manipulador fueron las de un dispositivo de utilidad que fuera seguro, eficiente y práctico. Algo importante para destacar antes de diseñar fue tener en cuenta la funcionalidad que iba a tener el manipulador puma en la tarea a realizar. Con estas herramientas se pudo concretar un primer diseño mecánico que trajo como consecuencia un esquema robusto, no atractivo físicamente pero en su momento ideal para lo que se estaba buscando.

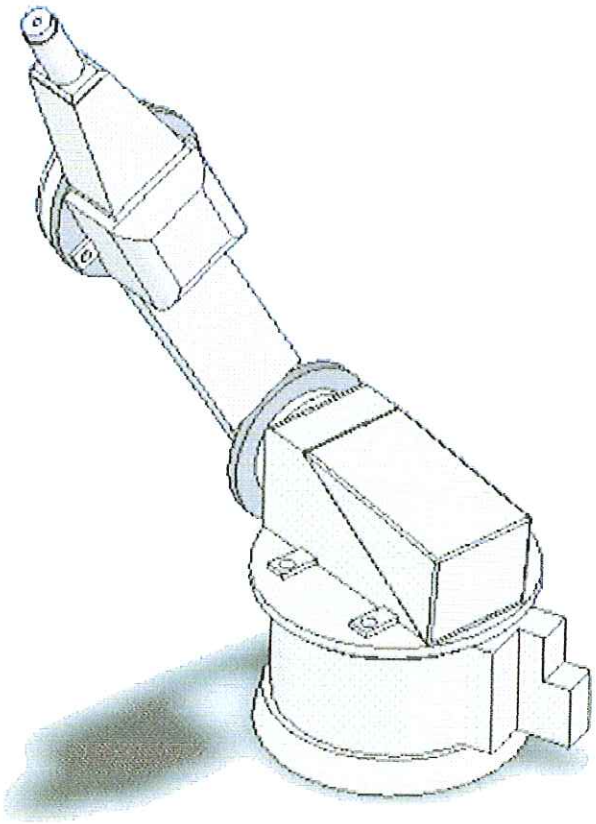


Figura 9. Diseño previo

Se decidió rediseñar el manipulador hacia un cambio total de diseño, logrando así

un esquema más práctico, resistente, moderno a la vista y con gran capacidad de carga, donde cada una de las piezas era fácil de construir y los motores permitían el libre movimiento del manipulador y gran espacio para las cajas de engranajes. Este diseño fue más óptimo al momento de construirlo y los materiales fueron fácilmente encontrados en el mercado. Además las piezas en conjunto gracias al fueron fáciles de ensamblar y los motores se adecuaron perfectamente al modelo permitiendo un satisfactorio funcionamiento del manipulador puma.

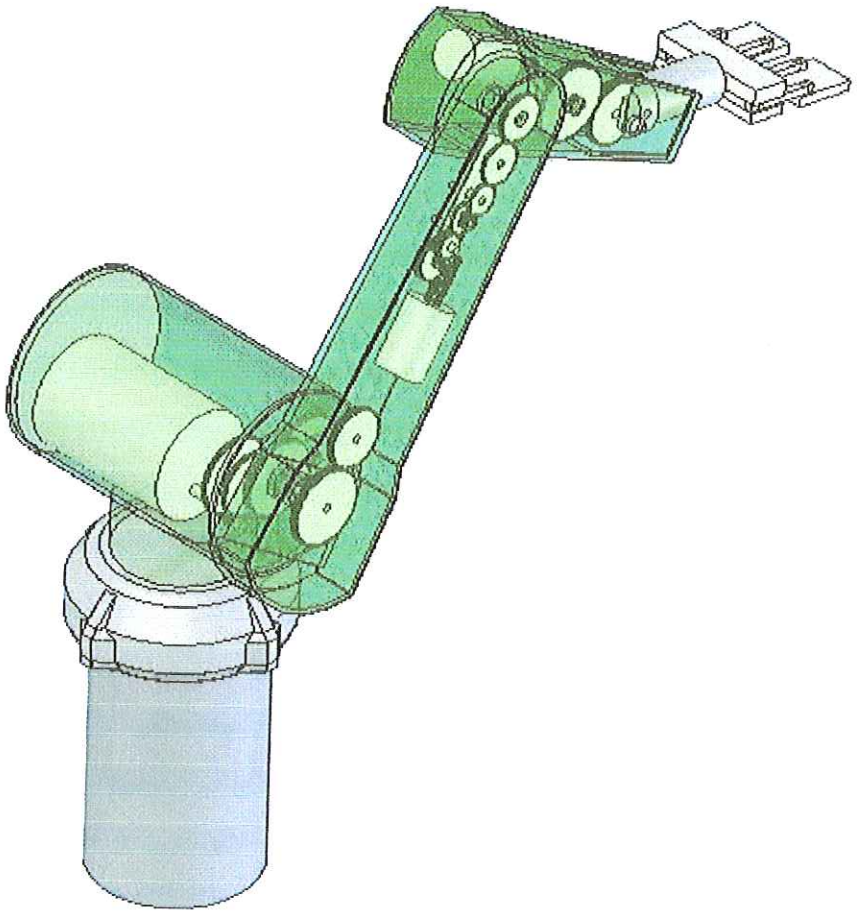


Figura 10. Diseño final

1.7.1 Selección de Materiales

El desempeño satisfactorio de partes de maquinaria y sistemas depende en gran medida de los materiales que elige quien diseña. Es importante tener en cuenta el comportamiento de los materiales, cuáles de sus propiedades afectan el desempeño de los elementos o piezas y sobretodo la disponibilidad de encontrarlos en el mercado comercial.

Los elementos de maquinaria se fabrican a menudo de uno de los metales o de aleaciones de metales tales como acero, aluminio o bronce.
Dentro de los diferentes tipos de materiales a emplear encontramos:

1.7.1.1 Acero al Carbón y Acero con Aleaciones: El acero es quizá el material que más se utiliza en elementos de maquinaria debido a sus propiedades de alta resistencia, extrema rigidez, durabilidad y relativa facilidad para fabricarlo; en el mercado es de gran facilidad encontrarlo. El término acero se refiere a una aleación de hierro, carbón, manganeso y uno o más elementos significativos. El carbón surte un efecto considerable en su resistencia, dureza y ductilidad.

En cuanto a las diversas aleaciones que encontramos tenemos elementos como el azufre, fósforo y plomo, los cuáles mejoran su maquinabilidad. A si mismo el níquel mejora su fortaleza, capacidad de endurecimiento y la resistencia a la corrosión.

Número AISI	*Aplicaciones*
1015	Partes de metal laminado; partes maquinadas (pueden ser carburizadas)
1030	Partes en forma de barra para uso general, palancas o manijas, eslabones o uniones, cuñas de unión
1045	Flechas o ejes, engranes
1080	Piezas para equipo agrícola (rejas, discos, dientes de rastrillos, dientes de podadoras de césped) que se someten a fricción; resortes.
1112	Piezas de tornillos para máquinas.
4140	Engranes, flechas o ejes, piezas forjadas.
4340	Engranes o ejes, piezas que requieren de un buen endurecimiento directo.
4640	Engranes, flechas o ejes, levas.
5150	Flechas o ejes para trabajo pesado, resortes, engranes.
52100	Pistas de rodamiento, bolas y baleros (acero para cojinetes)
6150	Engranes, piezas forjadas, flechas o ejes, resortes
8650	Engranes, flechas o ejes
9260	Resortes [4]

Tabla 2. Acero al Carbón y Acero con Aleaciones

1.7.1.2 Acero Inoxidable; El término acero inoxidable caracteriza al alto nivel de resistencia a la corrosión que ofrecen las aleaciones de este grupo; por lo menos la aleación debe tener un 10% de contenido de cromo.

[4] Robert L. Mott, P.E., "Diseño de Elementos de Maquinas 2ª Edición". Tabla correspondiente al uso de algunos aceros. Pagina 33.

Designación del material		Condición	Resistencia a la tracción		Resistencia a punto cedente		Ductibilidad (elongación porcentual en 2 pulg)
Número AISI	UNS		Ksi	MPa	Ksi	MPa	
Aceros austeníticos							
201	S20100	Recocido	115	793	55	379	55
		1/4 duro	125	862	75	517	20
		1/2 duro	150	1 030	110	758	10
		3/4 duro	175	1 210	135	931	5
		Totalmente duro	185	1 280	140	966	4
301	S30100	Recocido	110	758	40	276	60
		1/4 duro	125	862	75	517	25
		1/2 duro	150	1 030	110	758	15
		3/4 duro	175	1 210	135	931	12
		Totalmente duro	185	1 280	140	966	8
304	S30400	Recocido	85	586	35	241	60
310	S31000	Recocido	95	655	45	310	45
316	S31600	Recocido	80	552	30	207	60
Aceros ferríticos							
405	S40500	Recocido	70	483	40	276	30
430	S43000	Recocido	75	517	40	276	30
446	S44600	Recocido	80	552	50	345	25
Aceros martensíticos							
410	S41000	Recocido	75	517	40	276	30
416	S41600	Q&T 600	180	1 240	140	966	15
		Q&T 1000	145	1 000	115	793	20
		Q&T 1400	90	621	60	414	30
431	S43100	Q&T 600	195	1 344	150	1 034	15
440A	S44002	Q&T 600	280	1 930	270	1 860	3
Aceros endurecidos por precipitación							
17-4PH	S17400	H 900	200	1 380	185	1 280	14
		H 1150	145	1 000	125	862	19
17-7PH	S17700	RH 950	200	1 380	175	1 210	10
		TH 1050	175	1 210	155	1 070	12

[5]

Tabla 3. Acero Inoxidable

1.7.1.3 Acero Estructural: Los aceros estructurales se designan por medio de números ASTM que estableció la *American Society for Testing and Materials*. En el mercado se dispone de muchos grados de aceros estructurales de mayor resistencia para utilizarse en la construcción y aplicaciones vehiculares y de maquinaria; se diferencian por el rango de deformación que presentan dentro de sus características; y se fabrican casi siempre en forma de lámina y placa.

[5] Robert L. Mott, P.E., "Diseño de Elementos de Maquinas 2ª Edición". Tabla correspondiente a las propiedades de los aceros inoxidables. Apéndice A-15.

Número de designación del material (número ASTM)	*Grado o espesor*	*Resistencia a la tracción*		*Resistencia a punto cedente*		*Ductibilidad (elongación porcentual en 8 pulg)*
		Ksi	*MPa*	*Ksi*	*MPa*	
A36	$t \leq 8$"	58	400	36	248	20
A242	$t \leq 3/4$"	70	485	50	345	18
A242	$t \leq 1\frac{1}{2}$"	67	460	46	315	—
A242	$t \leq 4$"	63	435	42	290	—
A441	$t \leq 4$"	63	435	42	290	18
A514	Inmerso y templado, $t \leq 2\frac{1}{2}$"	115	800	100	700	18% (" 2")
A572	42, $t \leq 6$"	60	414	42	290	—
A572	50, $t \leq 4$"	65	448	50	345	—
A572	60, $t \leq 1\frac{1}{4}$"	75	517	60	414	—
A572	65, $t \leq 1\frac{1}{4}$"	80	552	65	448	—
A588	$t \leq 4$"	70	485	50	345	18

[6]

Tabla 4. Acero Estructural

1.7.1.4 Aluminio: El aluminio se utiliza en forma muy difundida en aplicaciones estructurales y mecánicas. Dentro de sus propiedades encontramos la ligereza, una buena resistencia a la corrosión, facilidad relativa para darle forma y maquinarlo y lo atractivo de su aspecto.

Algo importante para destacar es que su densidad equivale aproximadamente a una tercera parte de la del acero; sin embrago su resistencia es algo menor.

[6] Robert L. Mott, P.E., "Diseño de Elementos de Maquinas 2ª Edición". Tabla correspondiente a las propiedades de los aceros estructurales. Apéndice A-19.

Aleación	Aplicaciones	Formas
1060	Equipo químico y tanques	Lámina, placa, tubo
1350	Conductores eléctricos	Lámina, placa, tubo, varilla, barra, alambre, tubería, formas
2014	Estructuras para aeronaves y bastidores para vehículos	Lámina, placa, varilla, barra, alambre, formas, piezas forjadas (forjas)
2024	Estructuras para aeronaves, ruedas, partes de máquinas	Lámina, placa, tubo, varilla, barra, alambre, formas, remaches
2219	Partes sujetas a altas temperaturas (hasta 600°F)	Lámina, placa, tubo, varilla, barra, formas, forjas
3003	Equipo químico, tanques, utensilios de cocina, partes para arquitectura	Lámina, placa, tubo, varilla, barra, alambre, formas, tubería, remaches, forjas
5052	Tubería hidráulica, aparatos para el hogar (línea blanca), fabricación de láminas de metal	Lámina, placa, tubo, varilla, barra, alambre, remaches
6061	Estructuras, bastidores y partes para vehículos, usos marinos	Todas las formas
6063	Muebles, accesorios para arquitectura	Tubos, tubería, formas extruidas
7001	Estructuras de alta resistencia	Tubo, formas extruidas
7075	Estructuras para aeronaves y para trabajo pesado	Todas las formas, excepto tubería

7

Tabla 5 y 6. Aluminio

Con aleación y templado	Resistencia a la tracción		Resistencia a punto cedente		Ductibilidad (elongación porcentual en 2 pulg)	Resistencia al corte		Resistencia por durabilidad	
	Ksi	MPa	Ksi	MPa		Ksi	MPa	Ksi	MPa
1060-O	10	69	4	28	43	7	48	3	21
1060-H14	14	97	11	76	12	9	62	5	34
1060-H18	19	131	18	124	6	11	121	6	41
1350-O	12	83	4	28	28	8	55	—	—
1350-H14	16	110	14	97	—	10	69	—	—
1350-H19	27	186	24	165	—	15	103	7	48
2014-O	27	186	14	97	18	18	124	13	90
2014-T4	62	427	42	290	20	38	262	20	138
2014-T6	70	483	60	414	13	42	290	18	124
2024-O	27	186	11	76	22	18	124	13	90
2024-T4	68	469	47	324	19	41	283	20	138
2024-T361	72	496	57	393	12	42	290	18	124
2219-O	25	172	11	76	18	—	—	—	—
2219-T62	60	414	42	290	10	—	—	15	103
2219-T87	69	476	57	393	10	—	—	15	103
3003-O	16	110	6	41	40	11	121	7	48
3003-H14	22	152	21	145	16	14	97	9	62
3003-H18	29	200	27	186	10	16	110	10	69
5052-O	28	193	13	90	30	18	124	16	110
5052-H34	38	262	31	214	14	21	145	18	124
5052-H38	42	290	37	255	8	24	165	20	138
6061-O	18	124	8	55	30	12	83	9	62
6061-T4	35	241	21	145	25	24	165	14	97
6061-T6	45	310	40	276	17	30	207	14	97
6063-O	13	90	7	48	—	10	69	8	55
6063-T4	25	172	13	90	22	—	—	—	—
6063-T6	35	241	31	214	12	22	152	10	69
7001-O	37	255	22	152	14	—	—	—	—
7001-T6	98	676	91	627	9	—	—	22	152
7075-O	33	228	15	103	16	22	152	—	—
7075-T6	83	572	73	503	11	48	331	23	159

8

Robert L. Mott, P.E., "Diseño de Elementos de Maquinas 2ª Edición". Tabla correspondiente a las aleaciones de aluminio comunes y sus usos. Pagina 44.

1.7.2 Simulación en cosmosWorks

Básicamente para la construcción del prototipo de manipulador PUMA se tuvieron en cuenta dos tipos de metales: acero inoxidable y aluminio; ya que sus propiedades ofrecían mejores características de diseño y un funcionamiento satisfactorio del manipulador.

Ya conocidas las dimensiones de diseño para la base, el brazo y el antebrazo se entrará a evaluar el material de construcción para la misma.

La base es donde se encuentra aplicado todo el peso del manipulador aca el material a escoger debe ser mas resistente que en otras partes del manipulador Dicho material debe ser lo suficientemente resistente como para soportar cargas normales de hasta 15 kgf (el robot pesa casi 10 Kg), aún en los puntos donde se concentra el esfuerzo, que son los cambios de geometría y las secciones estrechas.

Para la selección del material se realizaron simulaciones computacionales con el software Cosmos para *Solidworks* el cual es una herramienta para simulación por elementos finitos que permite simular cargas y esfuerzos en estructuras modeladas en CAD.

Las dos variables de diseño que se tuvieron en cuenta fueron el grosor de la lámina del material y el material del cual esta constituido.

---

[8] Robert L. Mott, P.E., "Diseño de Elementos de Maquinas 2ª Edición". Tabla correspondiente a las propiedades típicas del aluminio. Apéndice A-19.

La base es donde se encuentra aplicado todo el peso del manipulador acá el material a escoger debe ser mas resistente que en otras partes del manipulador.

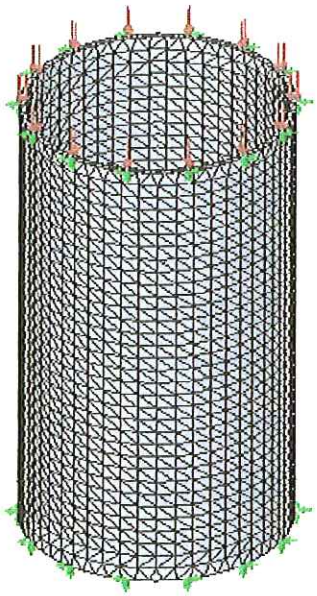


Figura 11. Enmallado base

Las pruebas consistieron en modificar el grosor de la pieza y/o variar el material para posteriormente exponer la estructura a una carga normal de 150 N sobre la cara donde iba a ser aplicada la presión, fijando la estructura en la parte inferior de la misma.

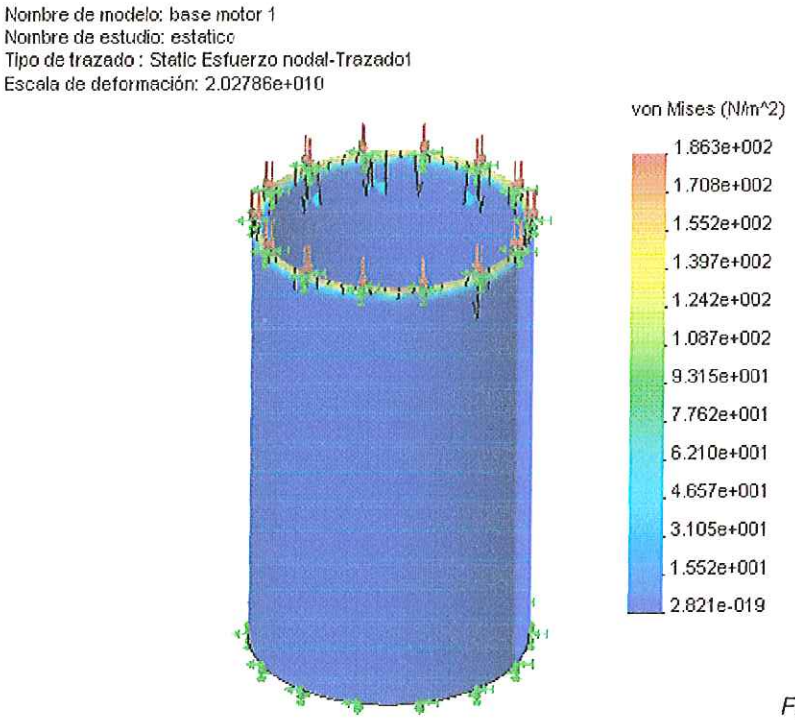


Figura 12. Análisis base

Las áreas en color azul es donde el factor de seguridad es alto, en cambio en las rojas es donde son más sensibles a deformación.

Cabe aclarar que la pieza en material de aluminio de aluminio presento características aceptable pues ofrecía una resistencia aproximada y un peso inferior que en otros materiales ante un cilindro de 3mm de espesor, pero sin embargo el costo de este material es 3 veces más que el acero.
Por tal razón se opto finalmente por el acero AISI 304 usando un cilindro de 10cm de diámetro externo y de 3mm de espesor.

La siguiente pieza analizada es la Base "T". Esta se fija en la parte inferior, como se puede ver en la gráfica con pequeñas flechas verdes, y se le aplica una fuerza de dentro hacia fuerza, en realidad hay una placa en este extremo y esta va acompañada de todo el peso de el brazo, antebrazo y carga adicional. Se aplica una aleación de Aluminio 1060. Este es el enmallado de la pieza:

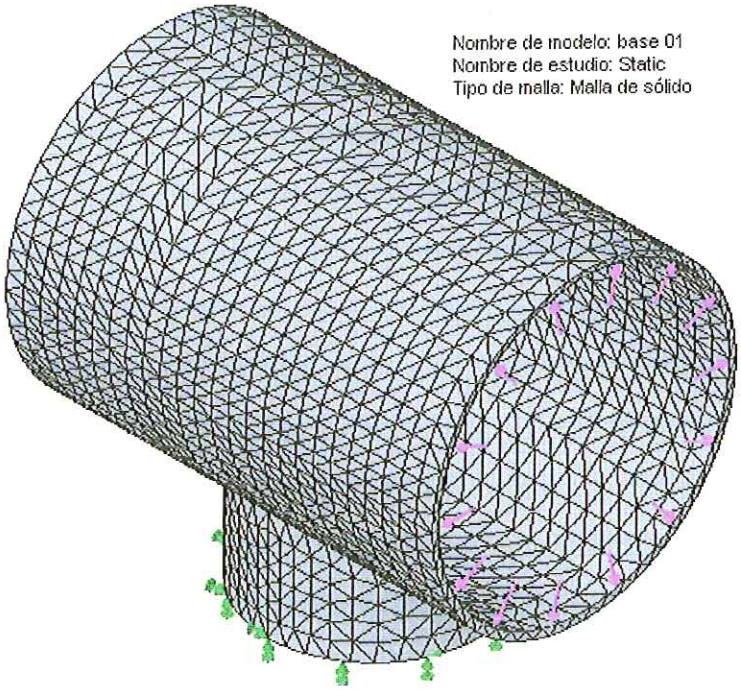


Figura 13. Enmallado T

En el diagrama de esfuerzos de la gráfica siguiente se ve como se expande por el efecto de una fuerza de 150 N, las partes azules tienen menor riego, con factor de seguridad más alto, y el rojo es el menor de seguridad más alto, por tanto el más propenso a deformación.

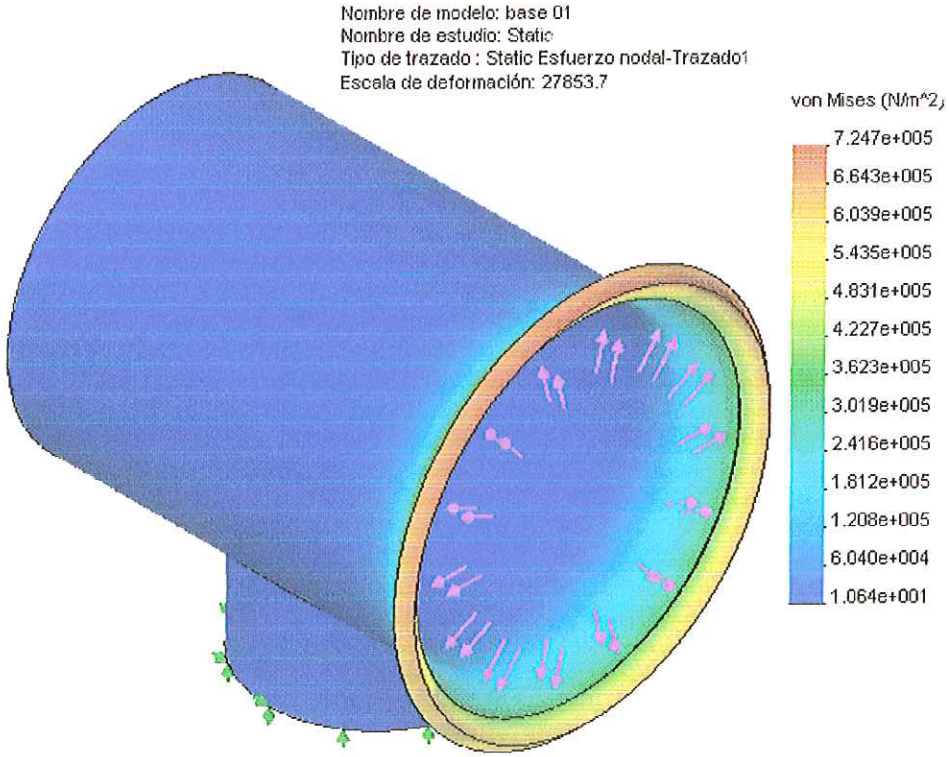


Figura 14. Análisis T

La siguiente pieza es el Brazo. El enmallado es el siguiente:

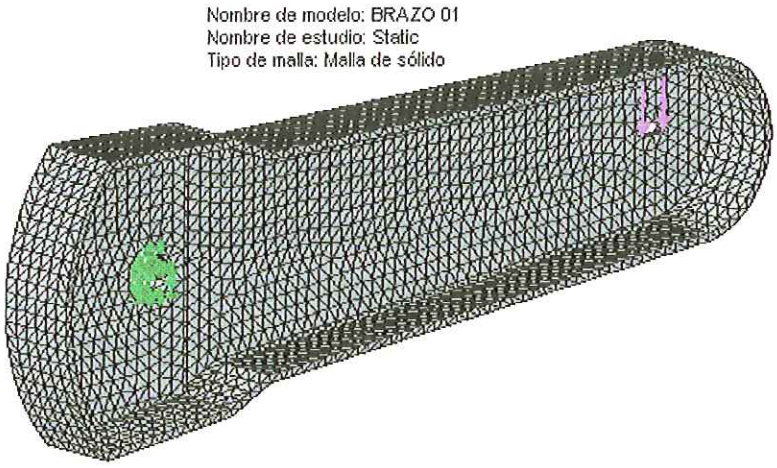


Figura 15. Enmallado brazo

De la misma manera se hace el análisis de deformación para esta pieza:

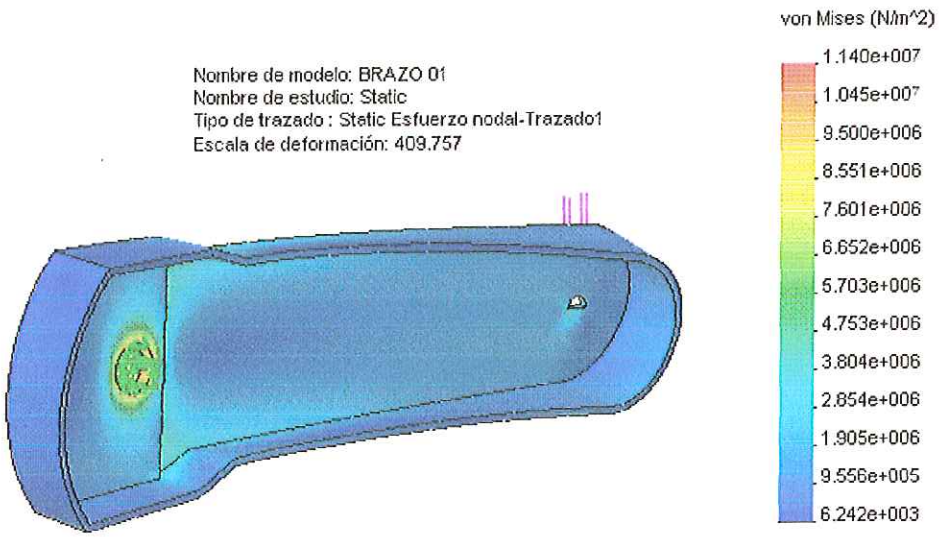


Figura 16. Análisis brazo

Se fija en el primer eje, el de color verde, y se aplica una fuerza de 100 N en el eje del extremo derecho. El material aplicado a la pieza es una aleación de Aluminic 1060.

La siguiente pieza es el Antebrazo. El enmallado es el siguiente:

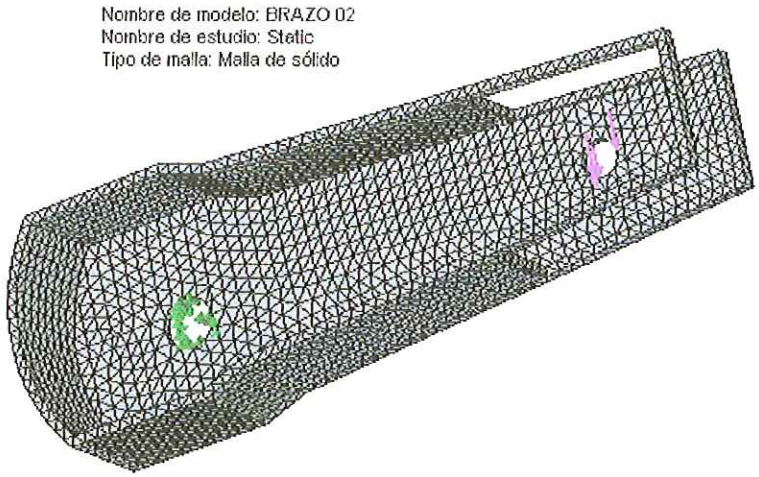


Figura 17. Enmallado Ante-brazo

El análisis de deformación es el siguiente:

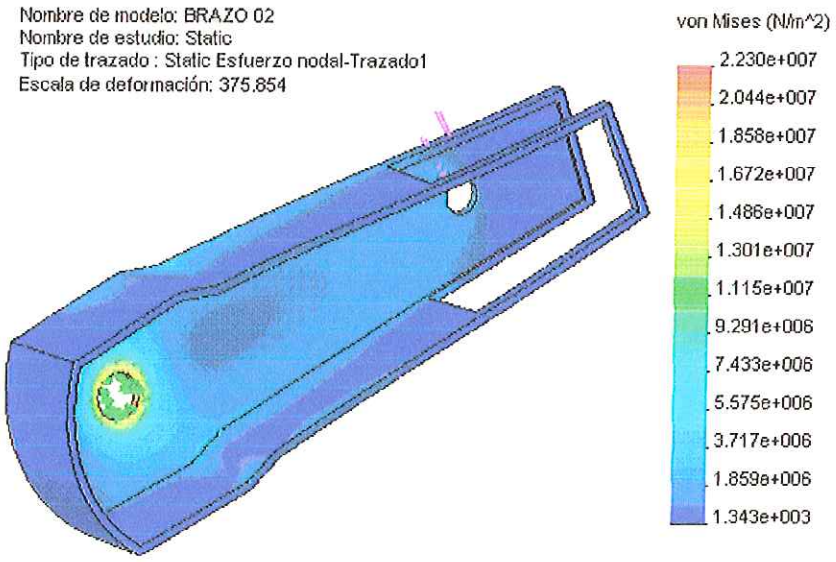


Figura 18. Análisis Ante-brazo

Se aplicó una fuerza en el extremo derecho de 100 N y se fija en el extremo izquierdo. Se aplica también una aleación de Aluminio 1060 como material.

1.8 COMPONENTES MECÁNICOS

Los mecanismos de transmisión se encargan de transmitir movimientos de giro entre ejes alejados. Están formados por un eje en el *motor* (*conductor*), un *eje resistente* (*conducido*) y otros elementos intermedios, que dependen del mecanismo particular.

Una *manivela* o un *motor* realizan el par motor necesario para provocar la rotación del eje motor.

Las diferentes piezas del mecanismo transmiten este movimiento al eje resistente, solidario a los elementos que realizan el trabajo útil. El mecanismo se diseña para que las velocidades de giro y los momentos de torsión implicados sean los deseados, de acuerdo con una relación de transmisión determinada.

1.8.1 Sistema de Transmisión Mecánica.

El tipo de transmisión mecánica con la que cuenta el manipulador automático es indirecta, los motores van acoplados a cada articulación a través de engranajes, con esto aseguramos una reducción en la inercia de las articulaciones debido a que se necesitan varias revoluciones del motor para que gire la articulación.

1.8.2 Transmisión de la articulación de la base

Sabemos que la base debe ser resistente ya que soporta todo el peso de la demás articulaciones a su vez su giro es hacia los lados, es decir casi no es necesario vencer la gravedad, puesto que no levanta ningún peso en esa dirección.

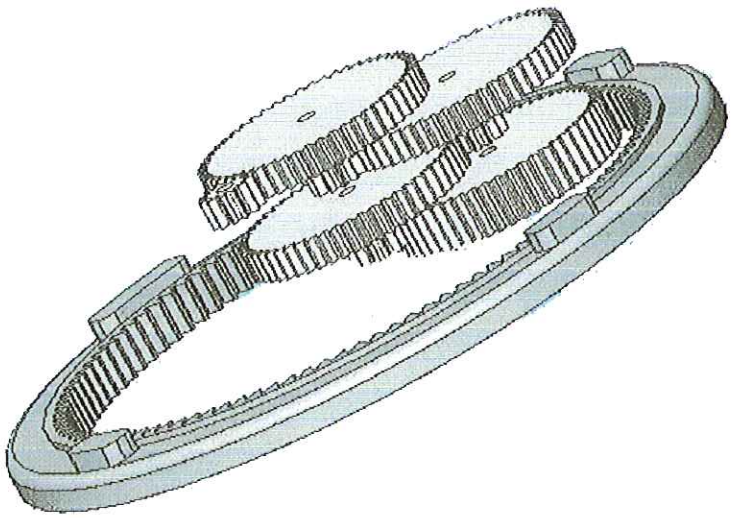


Figura 19. Transmisión de la articulación de la base 1

Sabemos que la base debe ser resistente ya que soporta todo el peso de la demás articulaciones a su vez su giro es hacia los lados, es decir casi no es necesario vencer la gravedad, puesto que no levanta ningún peso en esa dirección.
El motor seleccionado para esta parte del manipulador es un motor de corriente continua de 24 voltios que tiene una velocidad de 3895rpm (ver selección de actuadores)

Se eligió un engrane de dientes internos rectos por dos razones principales:

La primera para economizar espacio, con los rector ordinarios habría la necesidad de usar varios ejes, y el diámetro externo se incrementaría y el diámetro de uso seria relativamente pequeño al de el diámetro externo, es decir seria poco ingenieril.

La segunda razón es que de esta manera el peso de las demás

articulaciones del manipulador se distribuye alrededor de todo este engrane y no sobre un eje que se podría fácilmente partir.

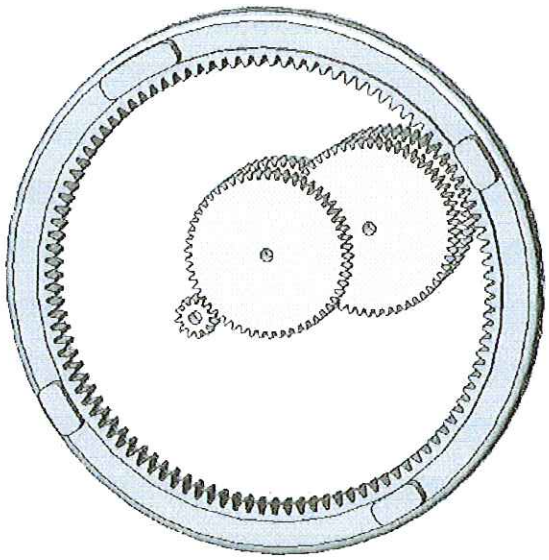


Figura 20. Transmisión de la articulación de la base 2

El engranaje de dientes internos esta soportado por unas balineras las cuales les da la facilidad de giro y soporte del peso distribuido. Se le adjunto unas guías en la superficie para la construcción de un soporte.

Para la transmisión interna se decidió usar un tren de engranaje compuesto, sobre dos ejes paralelos

De acuerdo a los parámetros de diseño todas las articulaciones tendrían una velocidad entre el rango de 4 a 5rpm, se inicio el proceso de diseño de engranajes rectos, que se ubicarían dentro del engranaje con dientes internos.

Puesto que se utilizaría ejes paralelos y la distancia entre estos ejes no cambiaria, los engranajes tendrían que cumplir con la siguiente condición.
La distancia entre los diámetros de los orificios de los engranajes debe ser la misma que la de los ejes. Por esta razón los diámetro externos tanto en el piñón (el que transmite el movimiento) como en el engrane (el conducido) deben ser iguales.

Sabiendo esto de decidió diseñar una relación de 1/779, luego de cálculos y análisis se Eligio los siguientes piñones y engranes

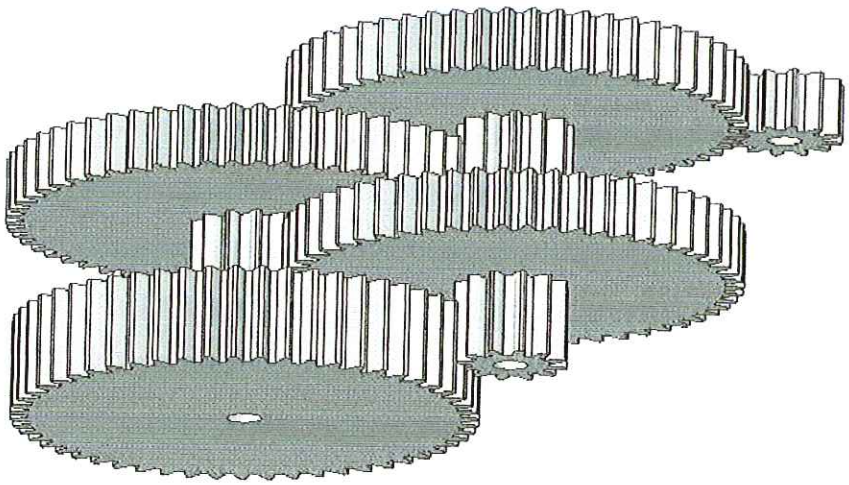


Figura 21. Transmisión de la articulación de la base. 3

El piñón que va en el eje del motor cuenta con doce dientes y con un modulo de 1, esta configuración se uso para todos los piñones. Los engranes cuentan con 54 dientes y con un modulo de 1.

Se unió un piñón y un engrane de manera que sus diámetros de los orificios fueran concéntricos y que giraran a la misma revolución

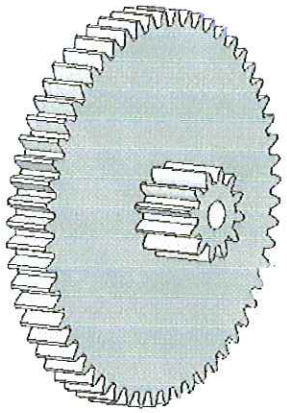


Figura 21. Transmisión de la articulación de la base. 4

La relación diseñada entre todos los engranaje fue la siguiente:

Numero de dientes del piñón / Numero de diente del engranaje= 1/relación=r

$$r = \frac{12}{54} \cdot \frac{12}{54} \cdot \frac{12}{54} \cdot \frac{12}{54} \cdot \frac{54}{107} = \frac{1}{779}$$

Siendo el engranaje de 107 dientes internos y 128mm de diámetro externo el último que gira y por lo tanto es el que otorga el primer grado de libertad al manipulador.

Entonces si el motor gira a 3895rpm con esta relación la articulación gira a 5 rpm la velocidad se controla por los drivers de potencia para los motores
Vale la pena nombrar que para que esta relación sea soportada, los engranajes se mandaron a hacer de acero

1.8.3 Transmisión de la articulación del brazo

Esta transmisión se puede considerar la mas importante puesto que es la que sostiene y hace girar el brazo que a su vez sostiene el antebrazo y este al gripper, por lo tanto es donde el torque debe ser mayor.

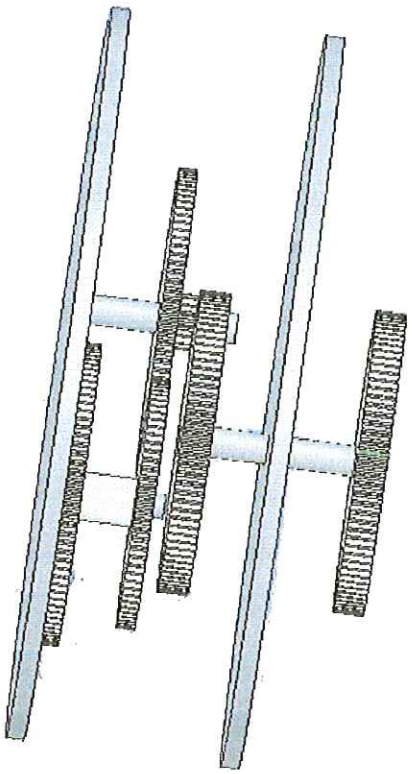


Figura 22. Transmisión del Brazo.1

El motor seleccionado para esta parte del manipulador es un motor de corriente continua de 24 voltios que tiene una velocidad de 150rpm (ver selección de actuadores) y un torque de 8kgf-cm

De acuerdo a los cálculos dinámicos (ver dinámica) sabemos que el torque min que debe tener esta articulación es de 14N-m o sea 142.8Kgf-cm, para que esto pase la relación debe ser de 1/17, pero no solo se busca que se pueda mantener las articulaciones, si no también que pueda levantar una carga.

Se decidió aumentar la relación a 1/27 para que de esta manera la carga a levantar sea considerable. El diseño de esta caja de transmisiones, es parecido al de la base, la diferencia es que no se manejo engranaje de dientes internos.

Para la transmisión interna se decidió usar un tren de engranaje compuesto, sobre dos ejes paralelos

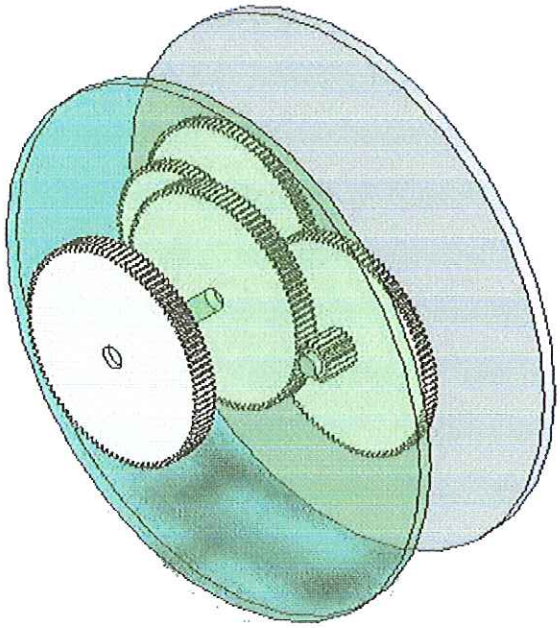


Figura 23. Transmisión del Brazo.2

De acuerdo a los parámetros de diseño todas las articulaciones tendrían una velocidad entre el rango de 4 a 5rpm, se inicio el proceso de diseño de engranajes rectos.

Se unió un piñón y un engrane de manera que sus diámetros de los orificios fueran concéntricos y que giraran a la misma revolución

Puesto que se utilizaría ejes paralelos y la distancia entre estos ejes no cambiaria, los engranajes tendrían que cumplir con la siguiente condición.

La distancia entre los diámetros de los orificios de los engranajes debe ser la misma que la de los ejes. Por esta razón los diámetro externos tanto en el piñón (el que transmite el movimiento) como en el engrane (el conducido) deben ser iguales.

El piñón que va en el eje del motor cuenta con 26 dientes y con un modulo de 0.5. El siguiente es un engranaje compuesto, por uno de 97 dientes y otro de 80 dientes ambos con un modulo de 0.5, este de 80 dientes va conectado a otro de 97 dientes, que hace girar a un piñón de 17 dientes, este mencionado piñón se conecta a un nuevo engranaje de 97 diente, con un eje fijo para que así transmita el movimiento al eje. A este eje se le conecta otro engranaje de 97 diente, que es el que va transmitir el movimiento al brazo y ofrece el segundo grado de libertad.

La relación diseñada entre todos los engranaje fue la siguiente:

Numero de dientes del piñón / Numero de diente del engranaje= 1/relación=r

$$r = \frac{26}{97} \cdot \frac{80}{97} \cdot \frac{16}{97} = \frac{1}{27.39}$$

Entonces si el motor gira a 150rpm con esta relación la articulación gira a 5.4 rpm como se busca que todas las articulaciones de muevan a la misma velocidad por medio de los driver de potencia se controla la velocidad para los motores.

Teniendo un torque de 8kgf-cm en el motor con esta relación el eje final tiene un torque de 219.12kgf-cm que es lo mismo que 21,4823529 N-m

Vale la pena nombrar que para que esta relación sea soportada, los engranajes que soportan mas carga (los últimos de las trasmisiones) se mandaron a hacer en acero

1.8.4 Transmisión de la articulación del ante-brazo

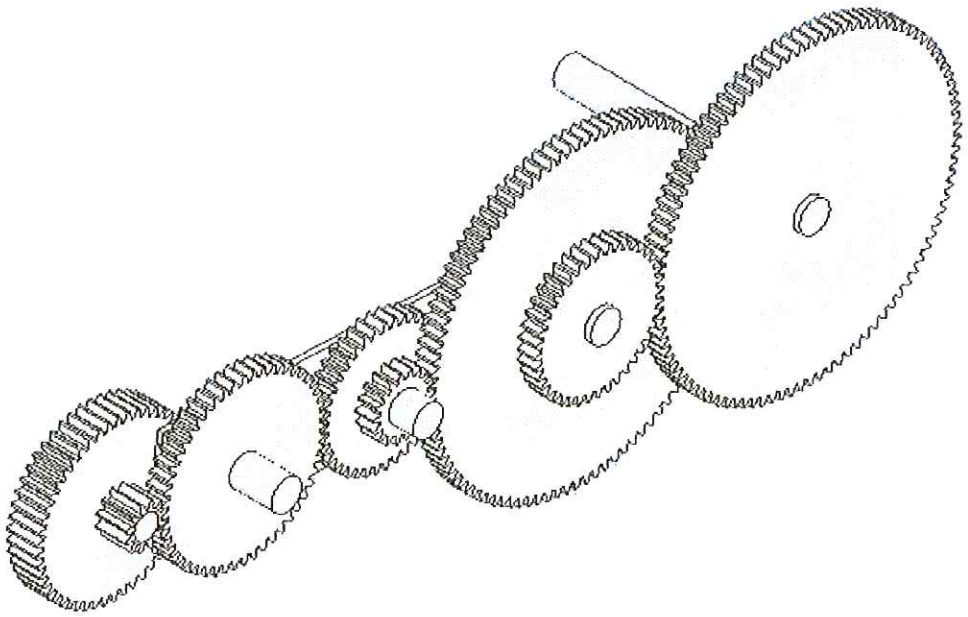


Figura 24. Transmisión del Ante-Brazo.1

Esta transmisión es la encargada de sostener y dar movimiento al antebrazo que a su vez sostiene el gripper.

Para diseñar esta trasmisión, contamos con suficiente espacio para montar varios ejes, pero debido al espesor del brazo (3 cm) no se pueden montar un engrane arriba de otro así que para el diseño se ubica un engrane por eje.
Debido a este espacio del brazo se opto en usar un mecanismo de tornillo sin fin que gire perpendicularmente a los demás engranajes pero en este tipo de mecanismos se sabe que por cada giro del tornillo sin fin, el engrane helicoidal solo gira un diente, por este motivo se uso un motor de buena velocidad.

El motor seleccionado para esta parte del manipulador es un motor de corriente continua de 24 voltios que tiene una velocidad de 4320 rpm (ver selección de actuadores) y un torque de 0.12kgf-cm

De acuerdo a los cálculos dinámicos (ver dinámica) sabemos que el torque min que debe tener esta articulación es de 7.2N-m o sea 73.4Kgf-cm, para que esto pase la relación debe ser de 1/ 611, pero no solo se busca que se pueda mantener las articulaciones, si no también que pueda levantar una carga.

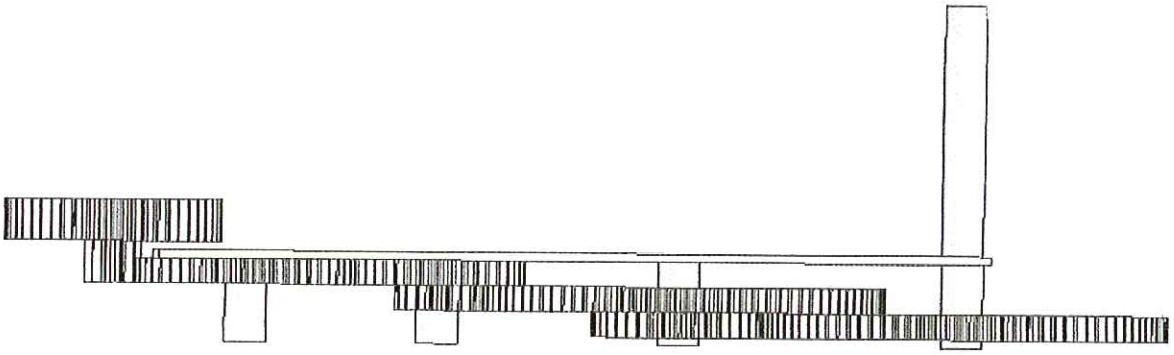


Figura 25. Transmisión del Ante-Brazo.2

Se decidió aumentar la relación a 1/1050 para que de esta manera la carga a levantar sea considerable. El diseño de esta caja de transmisiones, es parecido al de la base, la diferencia es que no se manejo engranaje de dientes internos.

De acuerdo a los parámetros de diseño todas las articulaciones tendrían una velocidad entre el rango de 4 a 5rpm, se inicio el proceso de diseño de engranajes rectos.

Se comienza con un engrane helicoidal de 34 dientes y un angulo de 20 grados el cual va girando de acuerdo aun tornillo sin fin que esta en el motor, acoplado al engrane helicoidal tenemos otro acoplado de 17 dientes con un modulo de 0.5 que engrana con otro de 52 dientes y este transmite movimiento a otro de 52 dientes

que hace girar a uno de 40 dientes que tiene acoplado otro de 18 dientes que transmite el movimiento a uno de 97 diente acoplado a un piñón de 40 dientes y este hace girar a un ultimo engrane de 97 dientes que el que esta conectado a un eje fijo y permite el movimiento del antebrazo y nos brinda el tercer grado de libertad

La relación diseñada entre todos los engranaje fue la siguiente:

Numero de dientes del piñón / Numero de diente del engranaje= 1/relación=r

$$r = \frac{1}{34.2} \cdot \frac{17}{52} \cdot \frac{52}{40} \cdot \frac{18}{97} \cdot \frac{40}{97} = \frac{1}{1053.25}$$

Entonces si el motor gira a 4320rpm con esta relación la articulación gira a 4.1 rpm como se busca que todas las articulaciones de muevan a la misma velocidad por medio de los driver de potencia se controla la velocidad para los motores.

Teniendo un torque de 0.12kgf-cm en el motor con esta relación el eje final tiene un torque de 126.36kgf-cm que es lo mismo que 12.388 N-m

Vale la pena nombrar que para que esta relación sea soportada, los engranajes que soportan mas carga (los últimos de las trasmisiones) se mandaron a hacer en acero

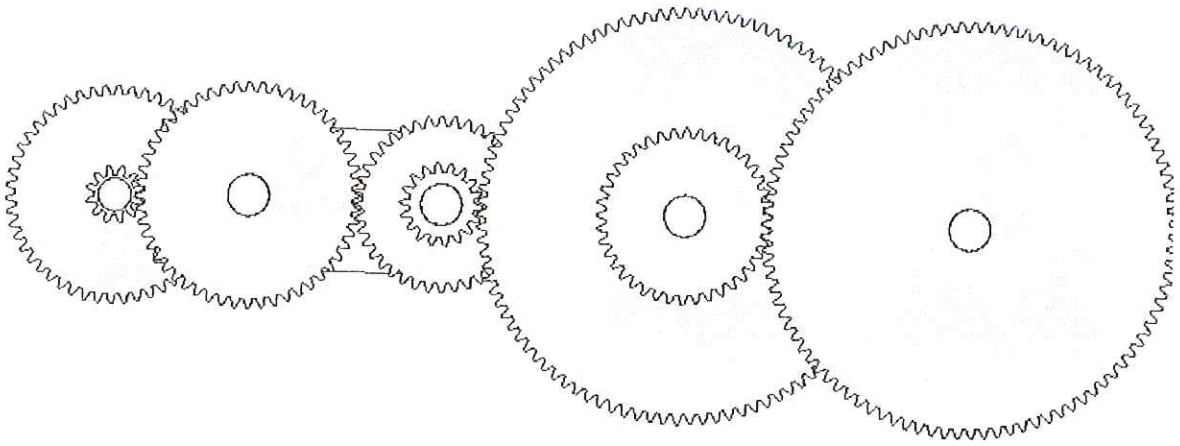


Figura 26. Transmisión del Ante-Brazo.3

1.8.5 Transmisión de la articulación del gripper

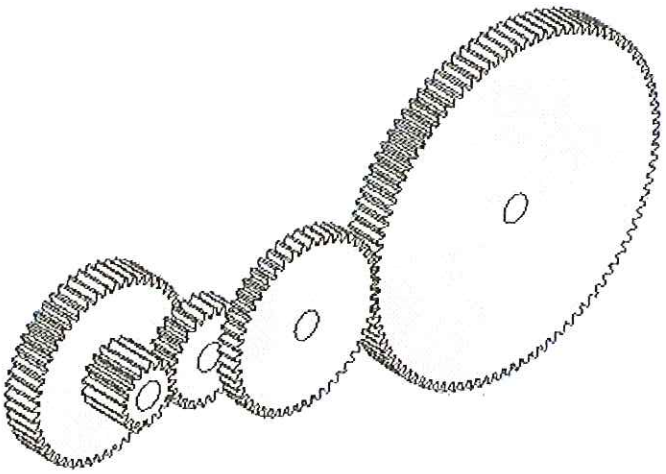


Figura 27. Transmisión del Gripper. 1

Esta transmisión es la encargada de sostener y dar movimiento gripper que a su vez sostiene una carga

Para diseñar esta trasmisión, contamos con suficiente espacio para montar varios ejes, pero debido al espesor del brazo (3 cm) no se pueden montar un

engrane arriba de otro así que para el diseño se ubica un engrane por eje.
Debido a este espacio del brazo se opto en tambien en usar un mecanismo de tornillo sin fin que gire perpendicularmente a los demás engranajes pero en este tipo de mecanismos se sabe que por cada giro del tornillo sin fin, el engrane helicoidal solo gira un diente, por este motivo se uso un motor de buena velocidad.

El motor seleccionado para esta parte del manipulador es un motor de corriente continua de 24 voltios que tiene una velocidad de 1850 rpm (ver selección de actuadores) y un torque de 0.08kgf-cm

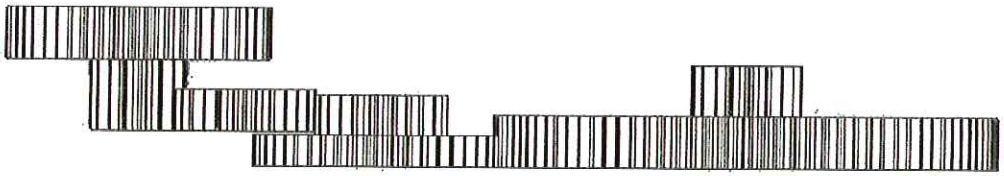


Figura 28. Transmisión del Gripper. 2

De acuerdo a los parámetros de diseño todas las articulaciones tendrían una velocidad entre el rango de 4 a 5rpm, se inicio el proceso de diseño de engranajes rectos.

Se comienza con un engrane helicoidal de 60 dientes y un angulo de 20 grados el cual va girando de acuerdo aun tornillo sin fin que esta en el motor, acoplado al engrane helicoidal tenemos otro acoplado de 17 dientes con un modulo de 0.5 que engrana con otro de 50 dientes y este transmite movimiento a otro de 20 dientes que hace girar a uno acoplado de de 41 dientes que hace gire e uno de 60 dientes y este hace girar a un ultimo engrane de 80 dientes que el que esta conectado al

gripper y nos brinda el cuarto grado de libertad

La relación diseñada entre todos los engranajes fue la siguiente:

Numero de dientes del piñón / Numero de diente del engranaje= 1/relación=r

$$r = \frac{1}{30.8} \cdot \frac{17}{50} \cdot \frac{20}{41} \cdot \frac{41}{60} \cdot \frac{60}{80} = \frac{1}{360}$$

Entonces si el motor gira a 1850rpm con esta relación la articulación gira a 5.13 rpm como se busca que todas las articulaciones de muevan a la misma velocidad por medio de los driver de potencia se controla la velocidad para los motores.

Teniendo un torque de 0.088kgf-cm en el motor con esta relación el eje final tiene un torque de 31.68kgf-cm que es lo mismo que 3.1 N-m

Vale la pena nombrar que para que esta relación sea soportada, los engranajes que soportan mas carga (los últimos de las trasmisiones) se mandaron a hacer en acero

1.9 SELECCIÓN DE ACTUADORES Y SENSORES

1.9.1 Actuadores

Los actuadores, también llamados accionamientos son las partes de un sistema de control que se encarga de regular la potencia de la planta. Los mismos pueden estar gobernados directamente por el controlador del sistema o requerir de algún tipo de preaccionamiento para amplificar la acción de mando. Los actuadores mas comúnmente utilizados en aplicaciones industriales son los destinados a producir movimiento (motores y cilindros).

Los dispositivos que pueden producir movimientos en un sistema industrial, se pueden clasificar en tres grandes grupos, dependiendo del tipo de energía que consuman.

Estos tres grandes grupos son:

- Hidráulicos.
- Neumáticos.
- Eléctricos.

Los elementos neumáticos e hidráulicos hacen uso del aire comprimido y de un Fluido a presión respectivamente. Los elementos eléctricos utilizan obviamente la energía eléctrica para su funcionamiento y son los conocidos motores eléctricos.

1.9.2 Selección de los actuadores

Para la selección de los actuadores se tuvieron en cuenta varios factores, como el costo, fácil adquisición, el torque, la velocidad (rpm) y lo más importante el sistema de control. Ya que para este manipulador se implementara un controlador el cual está incluido dentro de la categoría de controladores de lazo cerrado (*Figura 29. Control lazo cerrado*), pues resulta imposible establecer una relación fiable entre la señal aplicada y la posición final, es decir, los motores de corriente continua no pueden funcionar en lazo abierto. Para ello, se realimenta la variable de salida para generar a partir de esta y de la señal de control una función de error que el controlador tratará de minimizar. Para el correcto funcionamiento de estos motores, se necesita incorporar un detector de posicionamiento final del eje (potenciómetro o encoder). Por esta razón se ha decidido diseñar el manipulador con motores de corriente directa y no con motores paso a paso.

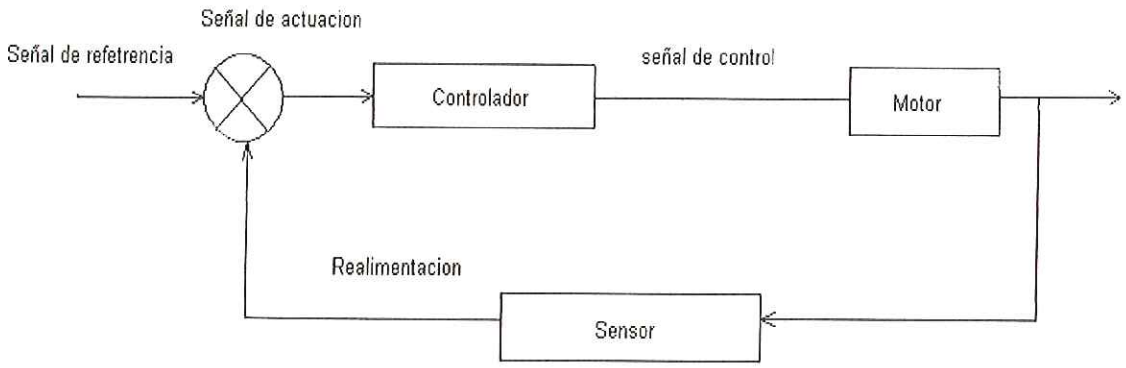


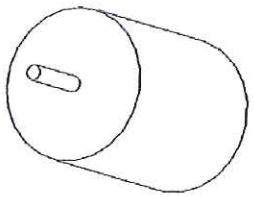
Figura 29. Control lazo cerrado

Teniendo como base los parámetros anteriores se procedió a la búsqueda de los actuadores que cumplieran las exigencias requeridas para poder dar movimiento al manipulador. Estos tenían que ser de masa muy pequeña, ya que de lo contrario aumentaba la masa de cada articulación y por ende el torque debería ser mayor para dar su movimiento. También se requería no ser tan grandes ya que nuestro diseño no permite una ubicación exacta para de actuadores grandes debido a su tamaño. Y algo muy importante capaces de generar los torques necesarios para mover las diferentes articulaciones y con una carga de 1kg de peso, los cuales tenemos según cálculos dinámicos, para la base un torque mínimo de 8N.m o sea 81Kgf-cm. Para la articulación del brazo un torque mínimo de 14N.m es decir 142.8Kgf-cm. Para la articulación del antebrazo un torque mínimo de 7.2N-m es decir 73.4Kgf-cm. Y para la articulación del gripper un torque mínimo de 3.1N.m lo mismo que 31.68Kgf-cm.

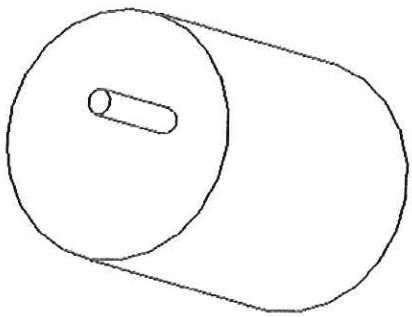
Teniendo en cuenta estos torques y el sistema de transmisión mecánica de las articulaciones se procedió a la búsqueda de los motores y auque no se consiguen con los torques exactos para nuestro diseño se buscaron torques parecidos, además con factor de seguridad elevado.

Posteriormente se busco que fueran económicamente asequibles ya que no contábamos con un presupuesto elevado, de esta forma se encontraron estos motores con las siguientes características.

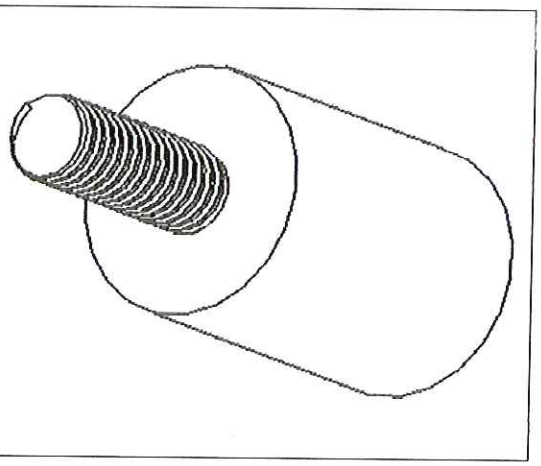
MOTOR NUMERO 1

Motor para la articulación de la base		
características	**Utilidad**	
Voltaje: 24v	Este motor esta destinado para dar movimiento a la transmisión de engranajes encargada de mover la base.	
Torque:0.58kgf-cm		
Masa:110gr		
Velocidad angular: 3895rpm		
Corriente:1.2A		

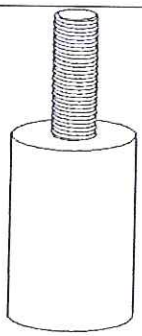
MOTOR NUMERO 2

Motor para la articulación del brazo		
características	**Utilidad**	
Voltaje: 24v	Este motor esta reservado para dar movimiento a la transmisión de engranajes encargada de mover el brazo.	
Torque:8kgf/cm.		
Masa:500gr		
Velocidad angular: 150rpm		
Corriente:2Amp nominal		

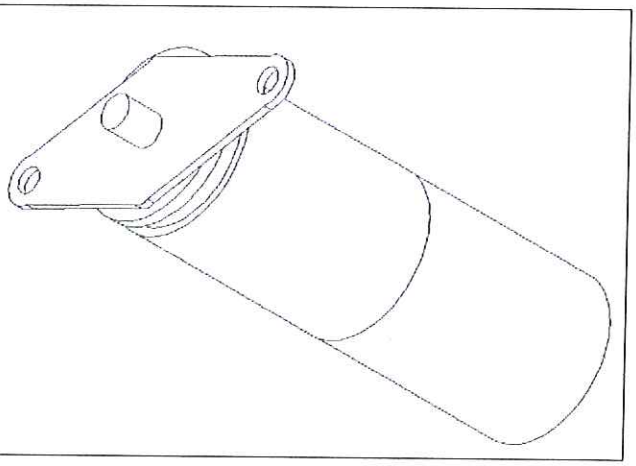
MOTOR NUMERO 3

Motor para la articulación del antebrazo		
características	**Utilidad**	
Voltaje: 24v	Motor encargado de dar movimiento a la transmisión de engranajes y tornillo sin fin encargada de mover el Antebrazo.	
Torque:0.12kgf/cm.		
Masa:194gr		
Velocidad angular: 4320rpm		
Corriente:800mAmp nominal		

MOTOR NUMERO 4

Motor para la articulación del gripper		
características	**Utilidad**	
Voltaje: 24v	Este motor esta reservado para dar movimiento a la transmisión de engranajes y tornillo sin fin encargada de mover el gripper.	
Torque:0.08kgf/cm.		
Masa:150gr		
Velocidad angular: 1850rpm		
Corriente:600mAmp nominal		

MOTOR NUMERO 5

Motor para la articulación del gripper		
características	**Utilidad**	
Voltaje: 24v	Este motor esta dedicado para dar movimiento al sistema de tornillo sin fin para abrir y cerrar el gripper.	
Torque: 0.12kgf/cm.		
Masa:60gr		
Velocidad angular: 118rpm		
Corriente: 500Amp nominal		

1.9.3 Sensores

Tanto la posición como los cambios de posición son factores de interés primario en muchos procesos. Las líneas de ensamblaje que utilizan robots, por ejemplo demandan el posicionamiento preciso de partes. De igual modo, el fresado, torneado y taladrado de piezas en una maquina de control numérico, requieren la medición y el control exacto de la posición. Existen varios sensores de posición y desplazamiento, tanto angular como lineal. Los mas usados en sistemas de control de movimiento son los codificadores ópticos, los potenciómetros y los transformadores diferenciales lineales (LVDT's).los codificadores magnéticos, los sincros, los resolucionadores (resolvers).

1.9.4 Potenciómetros

Los potenciómetros, son sensores de posición relativamente simples

formados por un elemento conductor de resistencia fija y uniforme, alimentado con un voltaje AC o DC constante y sobre el cual se desliza un cursor metálico. Este último es accionado por el eje de movimiento de sistema o el objeto bajo medición, de modo que a cada posición corresponde un voltaje equivalente entre el cursor y cualquiera de los extremos.

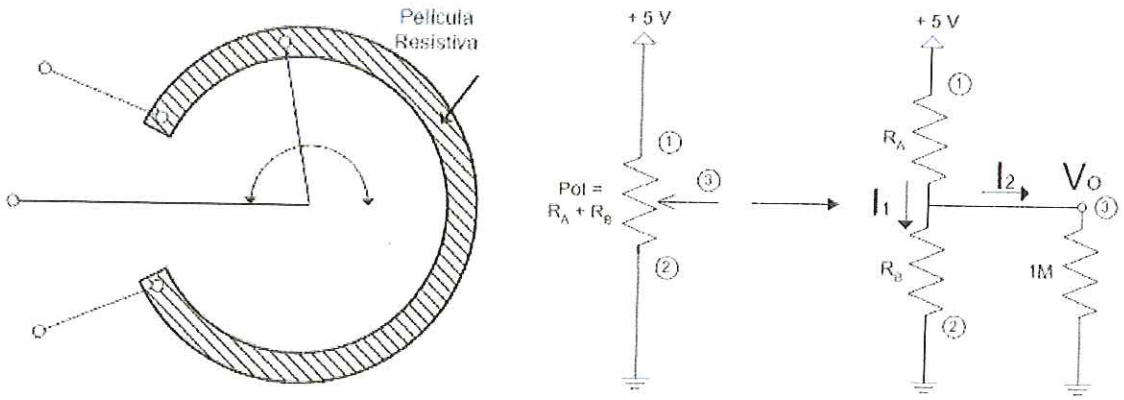


Figura 30. Esquema potenciómetro lineal

En la figura se ilustra el principio del potenciómetro lineal. Si se aplica una tensión Ve en la pista resistiva, se obtiene Vs en el contacto como función de la posición. El desplazamiento se define como

$$L_l = \frac{R_l L_T}{R_T} = \frac{V_s L_T}{V_e}$$

Siendo R_T la resistencia total entre extremos de la pista resistiva, R/ la resistencia entre el contacto deslizante y el extremo y L_T la longitud total.

Para realizar físicamente las pistas resistivas se emplean películas de material resistivo: carbón, metal, cerámica conductora, plástico conductor, etc. Los potenciómetros son sensores de relativamente bajo costo. Sin embargo, la precisión es limitada. En potenciómetros de calidad pueden conseguirse errores lineales del 0,1%.

En general, presentan problemas de fiabilidad debido a desgaste, fricciones, polvo, etc.

Nótese también que la salida que suministros los potenciómetros convencionales es analógica, por lo que es necesario digitalizar la señal para aplicar control digital.

1.9.5 Selección de sensores

Debido a su sencillez de funcionamiento se escogió el potenciómetro lineal como sensor de posicionamiento ya que los codificadores ópticos necesitan un circuito o un programa especial para decodificación de las señales. Además los potenciómetros lineales son más asequibles tanto en el mercado como económicamente.

El manipulador tendrá 4 potenciómetros lineales ubicados en la base, el brazo, el antebrazo y la muñeca, cada uno con las siguientes características:

- Potenciómetro lineal con resistencia de 5kΩ.
- Excelente linealidad.
- Alta precisión.
- Larga vida útil.
- un grado de libertad de 300°.

En cuanto al procedimiento de conversión análogo – digital, sabemos que el sistema requiere reconocer por lo menos 180 posiciones diferentes (para poder hacer un muestreo de 180° con resolución de 1 grado). Esta necesidad es cubierta por el microcontrolador 68HC908GP32 (previamente seleccionado), puesto que su

resolución es de 255 posiciones para un voltaje máximo de entrada de cinco voltios. Por lo cual, su resolución es de

5 V / 255 bits = 19.6 mV / bit

Los potenciómetros, deben ser lineales para que el voltaje entregado varíe de igual forma con al ángulo medido. Por otra parte, se debe verificar que la corriente entregada por el potenciómetro no exceda las capacidades del microcontrolador. En el caso del Motorola 68HC908GP32, los pines de entrada toleran hasta 25 mA (según la información suministrada por el fabricante).

Aplicando las leyes de voltaje de kirchoff, calculamos el mínimo valor que deberá tener el potenciómetro para que la corriente no exceda este umbral.

$$V = I.R \rightarrow \frac{V}{I} = R \rightarrow R = \frac{5v}{25e^{-3}A} = 200\Omega$$

Esto es: si la resistencia máxima del potenciómetro fuera de 200Ω (ò sea un potenciómetro de 200Ω), la corriente a través del mismo sería de 25 mA.
Conocido este mínimo de 200Ω, adquirimos los potenciómetros de 5 KΩ, los cuales están muy por encima del límite ya establecido, asegurando que el valor máximo de corriente que se le entregará al microcontrolador será de

$$\frac{V}{R} = I \rightarrow I = \frac{5v}{5e^{3}\Omega} = 1mA$$

Figura 31. potenciómetro lineal

1.9.5.1 Detectores de posición (finales de carrera)

1.9.5.2 Interruptores de posición

También llamados Finales de Carrera son utilizados para transformar un movimiento mecánico en una señal eléctrica. El movimiento mecánico en forma de leva o empujador actúa sobre la palanca o pistón de accionamiento del interruptor de posición haciendo abrir o cerrar un contacto eléctrico del interruptor.
Esta señal eléctrica se utiliza para posicionar, contar, parar o iniciar una secuencia operativa al actuar sobre los elementos de control de la máquina.

El manipulador estará dotado de finales de carrera tipo pulsadores de contacto NA (Normalmente abiertos). Los finales de carrera serán utilizados para colocar restricciones a los movimientos del manipulador cuando se este controlando de forma manual ya que se podrían presentar colisiones por mal uso del controlador. Cuando este es activado el mismo manda una señal al microcontrolador el cual es encargado de desactivar las señales que son enviadas a los drivers de los motores y así parar el manipulador y evitar accidentes y daños en el mismo

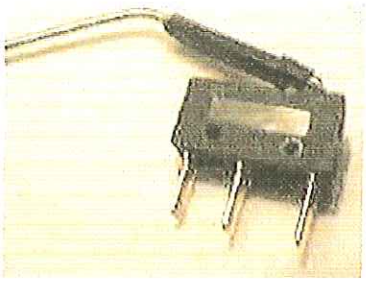


Figura 32. Final de carrera

1.10 DIAGRAMA DE CONTROL GENERAL

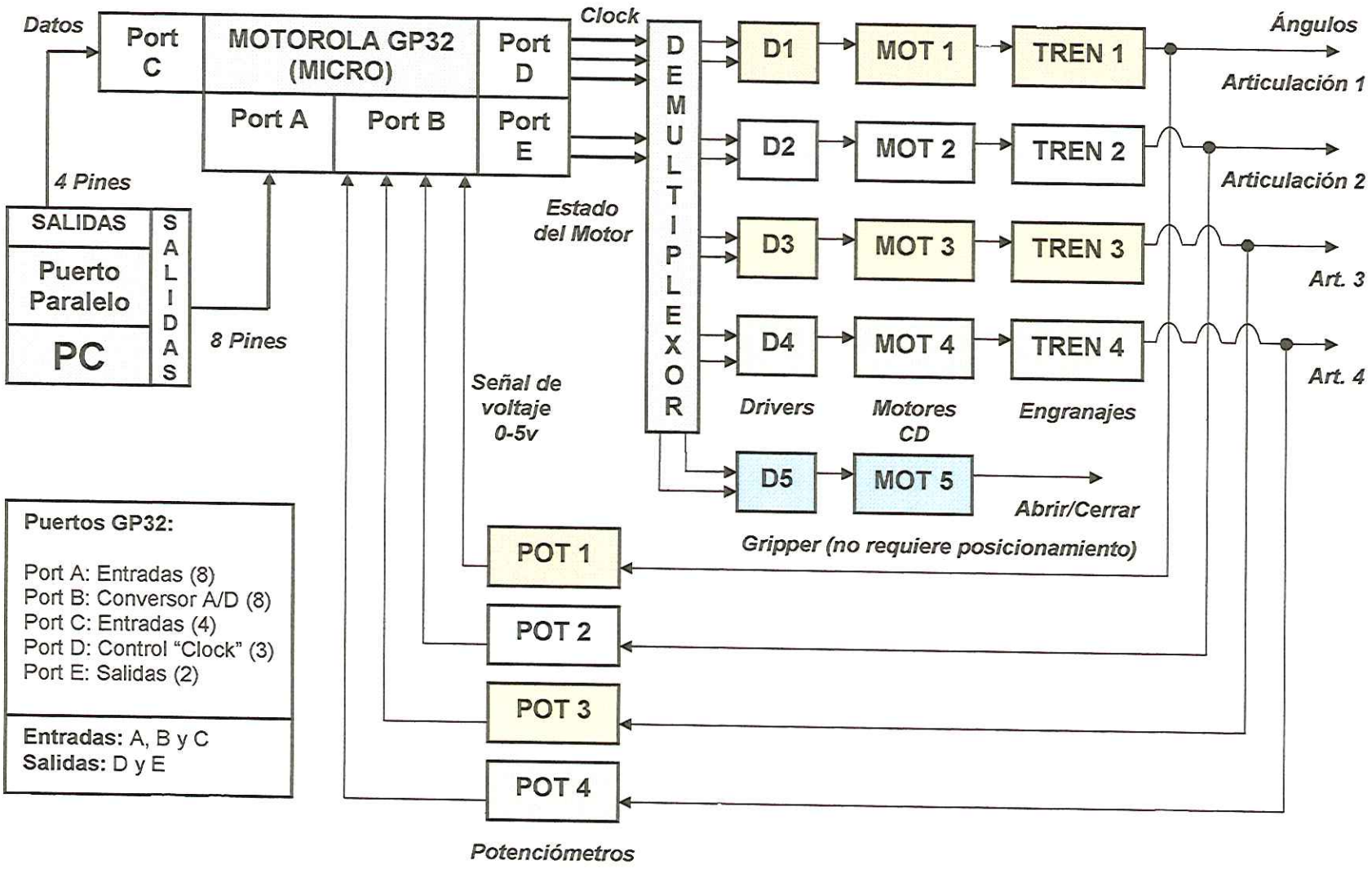


Figura 33. Diagrama de control General

1.10.1 Descripción del diagrama de control general

En el primer bloque encontramos el PC, aquí se maneja el software de control por MatLAB y el puerto paralelo como interfaz de entradas y salidas. Las salidas del puerto paralelo (puerto DATA) son 8 en total y vienen del pin 2 al 9, las cuales se conectan al **puerto A** (entradas) del microcontrolador, estas señales envían a este último un valor binario correspondiente al ángulo deseado, dentro del programa del micro se acumularán estos datos y se procesarán después de obtener los datos necesarios para realizar el movimiento. El puerto paralelo tiene otras salidas ubicadas en el puerto de CONTROL y comprenden los pines 1, 14, 16 y 17; estos pines se conectan con el **puerto C** del microcontrolador para decirle a éste qué número de motor corresponde el ángulo deseado, de esta manera el microcontrolador internamente va registrando los ángulos a mover en un vector de cuatro campos correspondientes a cada motor.

El microcontrolador utiliza el **puerto D** de salida para enviar una señal de reloj (clock), la cual es conectada a un *Demultiplexor,* con el fin de seleccionar secuencialmente los motores. Adicionalmente, a la entrada del demultiplexor se conectan dos salidas del micro, las cuales corresponden al puerto E, el cual se encarga de mandar un dato de 2 bits correspondiente al estado de activación del motor, este dato es recibido por cada *driver* y obtener los estados siguientes:

Bit 1	Bit 2	ESTADO
1	1	Freno
1	0	Izquierda
0	1	Derecha
0	0	Nulo

Estados de activación del motor por el driver

Los estados Izquierda y Derecha dependen de la conexión electrónica, el hecho es que si son bits diferentes motor gira en alguno de los dos sentidos.

El micro se encarga de activar cada uno de los 4 primeros motores en secuencia, es decir, se envían los datos al primero, luego al segundo, etc., y así sucesivamente en un tiempo muy corto, hasta completar el movimiento del manipulador, cada uno de los potenciómetros que retroalimentan el lazo cerrado de control está sujeto a cada articulación, cuando se alcance la lectura correspondiente al ángulo deseado, se detiene la articulación que detectó su respectivo potenciómetro, el ciclo sigue hasta completar el movimiento, de esta manera las articulaciones del brazo parecerán moverse al mismo tiempo, es una forma eficiente para llegar más rápido al objetivo.

Cada motor es conectado antes a un *Driver* (Toshiba TA7291P a 2.0A máx), el cual está encargado de mover el motor de un sentido a otro y frenarlo (*ver tabla: Estados de activación del motor por el driver*), este alimenta el motor hasta 20V máximo y regula la velocidad del mismo gracias a un voltaje de referencia que va de 0 a 20V. Cada uno de los motores va con su tren de engranajes respectivo, y las articulaciones están en el orden de 4 RPM aprox.

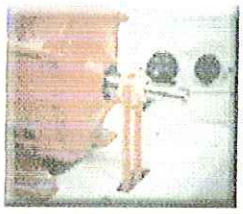
Finalmente se produce un ángulo en cada articulación, es entonces donde 4 potenciómetros (lineales, 5KΩ), mencionados anteriormente, mandan un voltaje de 0 a 5V al micro, dependiendo del ángulo, por el conversor A/D del **puerto B** y así detener el proceso de movimiento de acuerdo con el ángulo deseado ya enviado por el PC.

DISEÑO DE CONTROL DEL MANIPULADOR

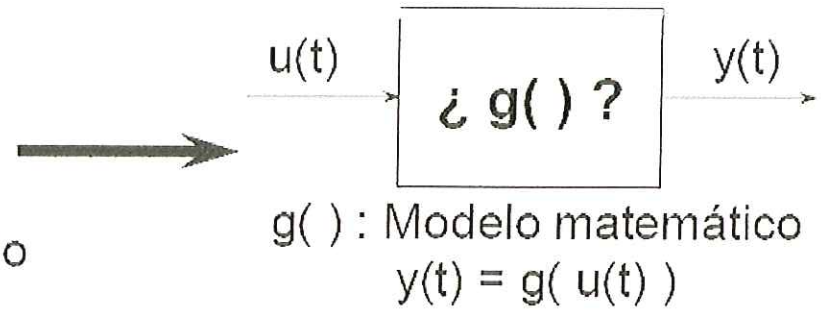
MODELO MATEMÁTICO

• Para realizar un sistema de control es conveniente disponer de un modelo matemático de la planta que permita diseñar el controlador adecuado.

• MODELO MATEMÁTICO: conjunto de ecuaciones que intentan aproximar el efecto que tienen unas variables (entradas: U) sobre otras variables (salidas: Y) en un sistema a lo largo del tiempo.



Planta o proceso



g( ) : Modelo matemático

$y(t) = g(u(t))$

• El modelo siempre es una aproximación y supone un compromiso entre exactitud y sencillez.

MÉTODOS DE OBTENCIÓN DE MODELOS MATEMÁTICOS

• **ANALÍTICO**: se estudia la constitución de la planta y se aplican las leyes físicas que caracterizan sus componentes para formular las ecuaciones del modelo. Para aplicarlo es necesario conocer los componentes que forman el sistema y el funcionamiento de los mismos.

• **EXPERIMENTAL**: se somete al sistema a pruebas en las variables de entrada y se observa el comportamiento de las salidas, tratando de establecer las ecuaciones que

determinarían ese mismo comportamiento. El sistema se observa como una "caja negra" sometida a pruebas, sin conocer sus componentes.

• Habitualmente los modelos matemáticos se expresan mediante:

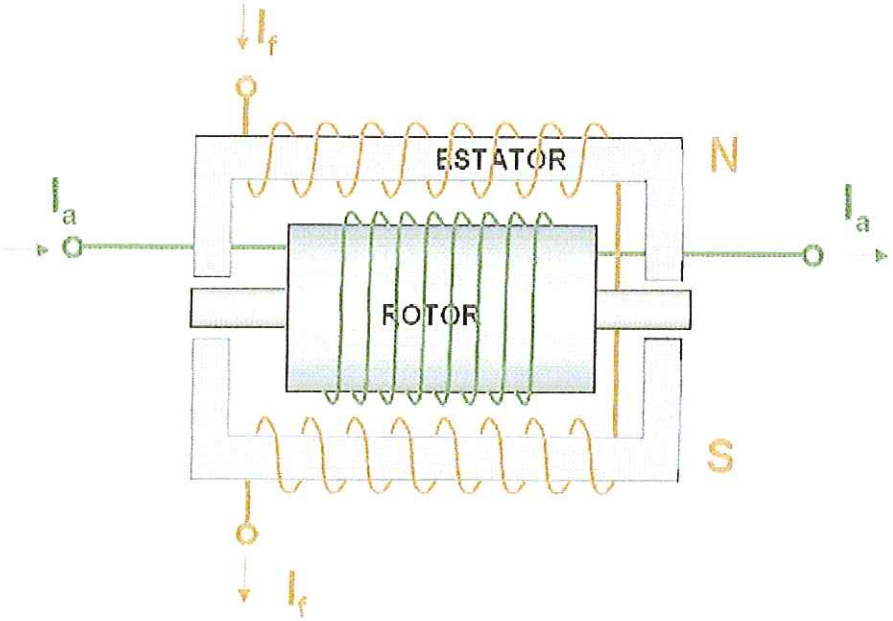
– Una ecuación diferencial (o en diferencias) de orden n.

– Sistemas de ecuaciones diferenciales (o en diferencias) de primer orden.

– Transformada de Laplace o Transformada Z.

SECCIÓN DE UN MOTOR DE CORRIENTE CONTINUA (M_{CC})

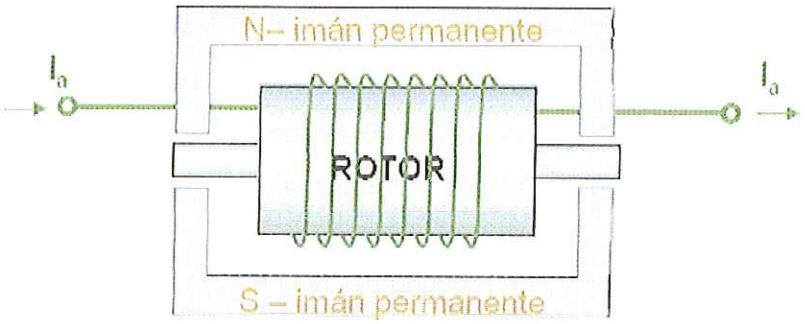
• MECANICO:



– ESTATOR: zona estática, se genera el campo magnético.

– ROTOR: zona móvil, se genera el campo eléctrico.

• ELÉCTRICO:



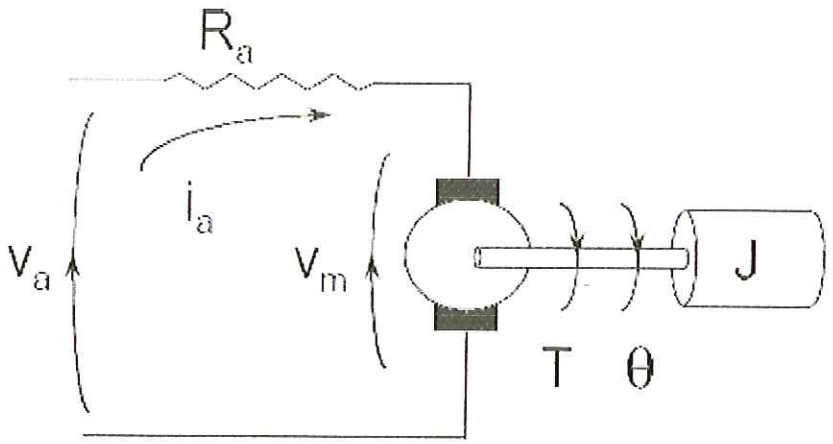
– INDUCTOR: bobinados que generan el campo magnético.

– INDUCIDO: bobinados que generan el campo eléctrico.

Hay dos formas para calcular el modelo matemático para los motores de corriente continua, una es por el método analítico y la otra es por el método experimental:

MÉTODO ANALÍTICO

Entrada: $V_a(t)$ Salida: $\theta(t)$



Par Generado:

$$K_1 \cdot i_a(t)$$

Par Resistente:

$$J\frac{d^2\theta(t)}{dt^2}$$

El equilibrio de pares:

$$J\frac{d^2\theta(t)}{dt^2}=K_1 \cdot i_a(t)$$

La tensión generada:

$$v_m(t)=K_2 \cdot \frac{d\theta(t)}{dt}$$

Circuito eléctrico:

$$v_a(t)=R \cdot i_a(t)+v_m(t)$$

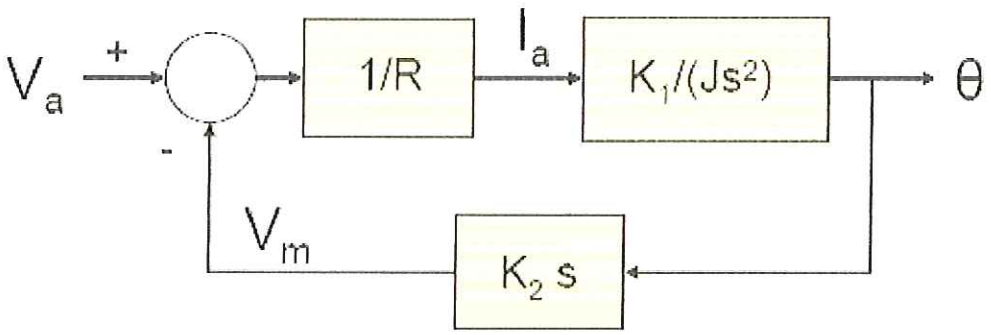
Estas tres ecuaciones caracterizan el comportamiento del motor de CC.

Diagrama de bloques interno:

$$\begin{cases} i_a(t)=\dfrac{v_a(t)-v_m(t)}{R} \\ J\dfrac{d^2\theta(t)}{dt^2}=K_1 \cdot i_a(t) \\ v_m(t)=K_2 \cdot \dfrac{d\theta(t)}{dt} \end{cases}$$

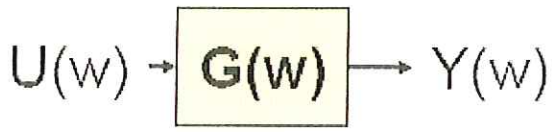
Laplace

$$\begin{cases} I_a(s)=\dfrac{V_a(s)-V_m(s)}{R} \\ Js^2\theta(s)=K_1 \cdot I_a(s) \\ V_m(s)=K_2 \cdot s \cdot \theta(s) \end{cases}$$



Función de Transferencia:

Expresa únicamente la relación entre la entrada y la salida, sin relación con otras variables internas.



En el motor de CC se puede obtener una única ecuación diferencial a partir del sistema de ecuaciones:

$$\frac{JR}{K_1}\frac{d^2\theta(t)}{dt^2}+K_2\frac{d\theta(t)}{dt}=v_a(t)$$

Utilizando la transformada de Laplace:

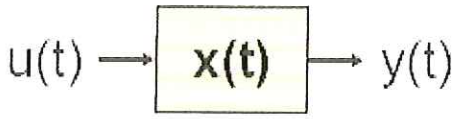
$$\frac{JR}{K_1}s^2\theta(s)+K_2 s\theta(s)=V_a(s)$$

La función de Transferencia es:

$$\frac{\theta(s)}{V_a(s)}=\frac{1}{\frac{JR}{K_1}s^2+K_2 s}=\frac{K_m}{s(\tau_m s+1)}$$

$$K_m=\frac{1}{K_2}\qquad \tau_m=\frac{JR}{K_1K_2}$$

Espacios de estados:



X: vector de variables de estado

$\{x_1(t), x_2(t), \ldots x_n(t)\}$

El sistema se modela mediante un conjunto de ecuaciones diferenciales (o en diferencias) de primer orden que relacionan la entrada, la salida y n de variables de estado.

En sistemas continuos:

$$\frac{dx}{dt} = f(x(t), u(t), t)$$
$$y(t) = g(x(t), u(t), t)$$

En sistemas discretos:

$$x[k+1] = f(x[k], u[k], k)$$
$$y[k] = g(x[k], u[k], k)$$

Espacios de estados del Motor de CC:

El motor de CC analizado se caracteriza por la ecuación diferencial:

$$\frac{JR}{K_1}\frac{d^2\theta(t)}{dt^2} + K_2\frac{d\theta(t)}{dt} = v_a(t)$$

Si se eligen como *variables de estado:*

$$x_1(t) = \theta(t)$$
$$x_2(t) = \frac{d\theta(t)}{dt}$$

Obtenemos un sistema de 2 ecuaciones diferenciales de primer orden:

$$\frac{dx_1(t)}{dt} = x_2(t)$$

$$\frac{dx_2(t)}{dt} = \frac{-k_1 k_2}{RJ} x_2(t) + \frac{k_1}{RJ} v_a(t)$$

La salida es:

$$y(t) = \theta(t) = x_1(t)$$

Control de Velocidad del Motor de CC:

La relación entre tensión Va y la posición angular del motor está calculada como:

$$\frac{\theta(s)}{V_a(s)} = \frac{K_m}{s(T_m s + 1)}$$

La velocidad angular es:

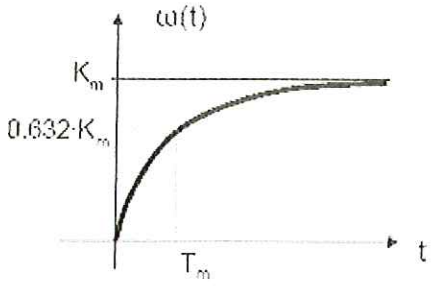
$$\omega(t) = \frac{d\theta(t)}{dt} \quad \Rightarrow \quad \omega(s) = s\theta(s)$$

La función de transferencia para control de velocidad:

$$\frac{\omega(s)}{V_a(s)} = \frac{K_m}{T_m s + 1}$$

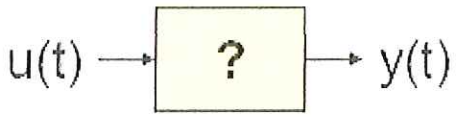
Si la entrada es un escalón unitario, la salida (velocidad) se puede calcular fácilmente aplicando la transformada inversa:

$$\omega(t) = K_m (1 - e^{-\frac{t}{T_m}})$$



Se obtuvo una respuesta similar a la carga de un Condensador.

MÉTODO EXPERIMENTAL



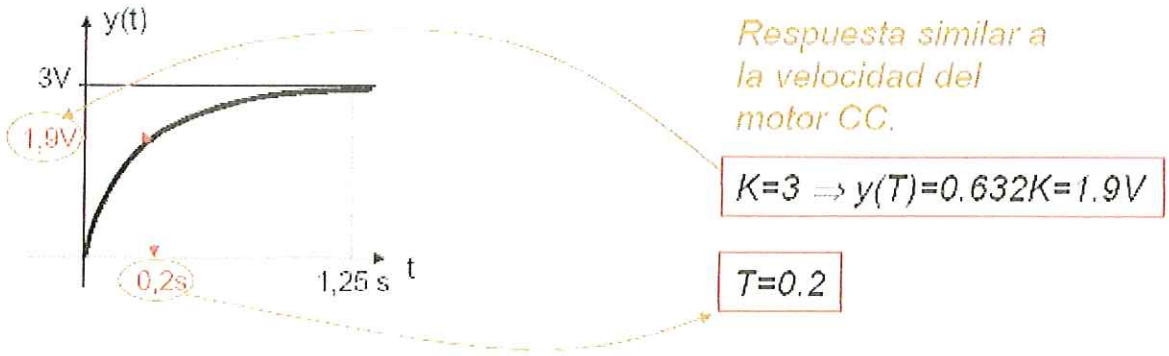
Consiste en:

1.- **Aplicar** al sistema determinadas entradas de prueba.

2.- **Registrar y analizar** la salida para inferir de qué tipo de sistema se trata.

3.- Realizar medidas en la salida para **calcular** los parámetros del sistema.

Ejemplo: *Supongamos un circuito electrónico desconocido, al que solo se puede acceder a sus bornes de entrada y de salida. Ponemos a la entrada un* ***escalón unitario de tensión*** *y obtenemos la siguiente salida:*



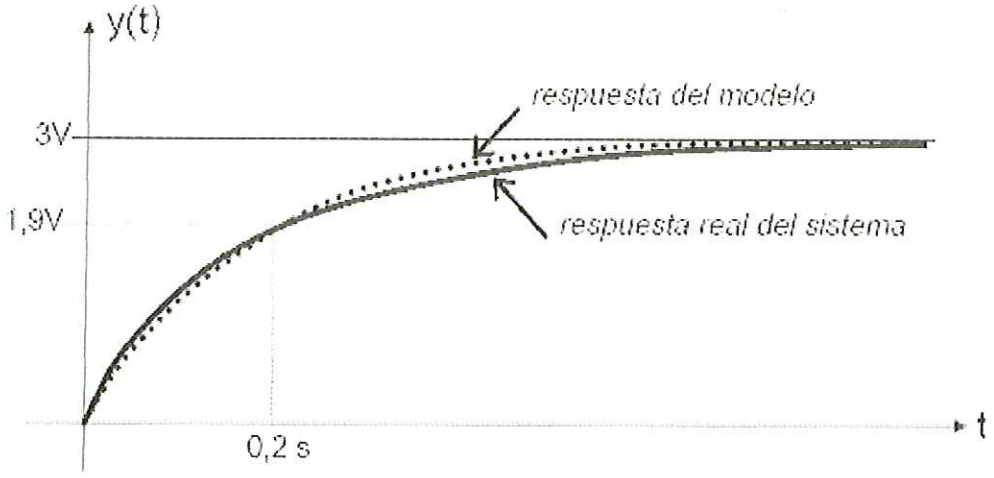
La señal de salida se puede aproximar por:

$$y(t) = K(1 - e^{-\frac{t}{T}}) = 3(1 - e^{-\frac{t}{0,2}})$$

Y la función de transferencia será:

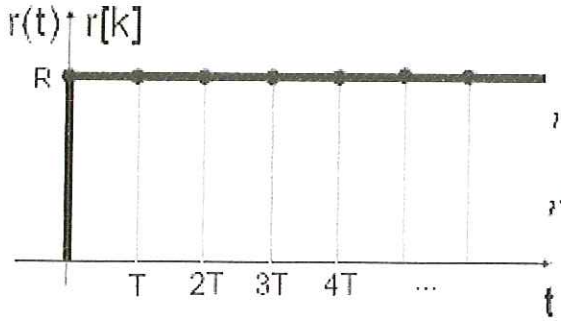
$$\frac{Y(s)}{U(s)} = \frac{K}{Ts+1} = \frac{3}{0,2s+1}$$

La respuesta real del sistema y la respuesta del modelo coincidirán en t = 0,2 y en t → ∞:



Entradas de Prueba:

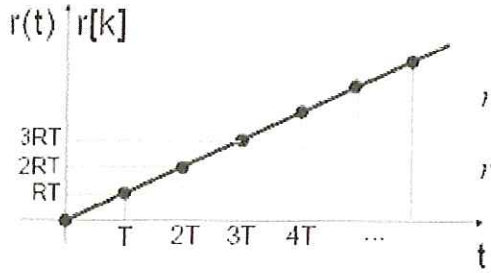
Entrada Escalón:



$$r(t) = Ru(t) \rightarrow R(s) = \frac{R}{s}$$

$$r[k] = Ru[k] \rightarrow R(z) = \frac{Rz}{z-1}$$

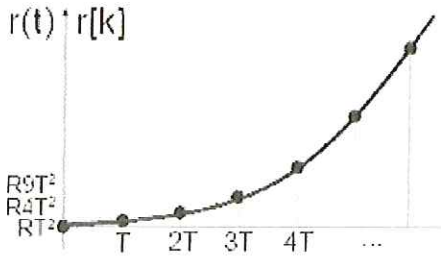
Entrada Rampa:



$$r(t) = Rtu(t) \rightarrow R(s) = \frac{R}{s^2}$$

$$r[k] = RTku[k] \rightarrow R(z) = \frac{RTz}{(z-1)^2}$$

Entrada Parábola:



$$r(t) = Rt^2 u(t) \rightarrow R(s) = \frac{2R}{s^3}$$

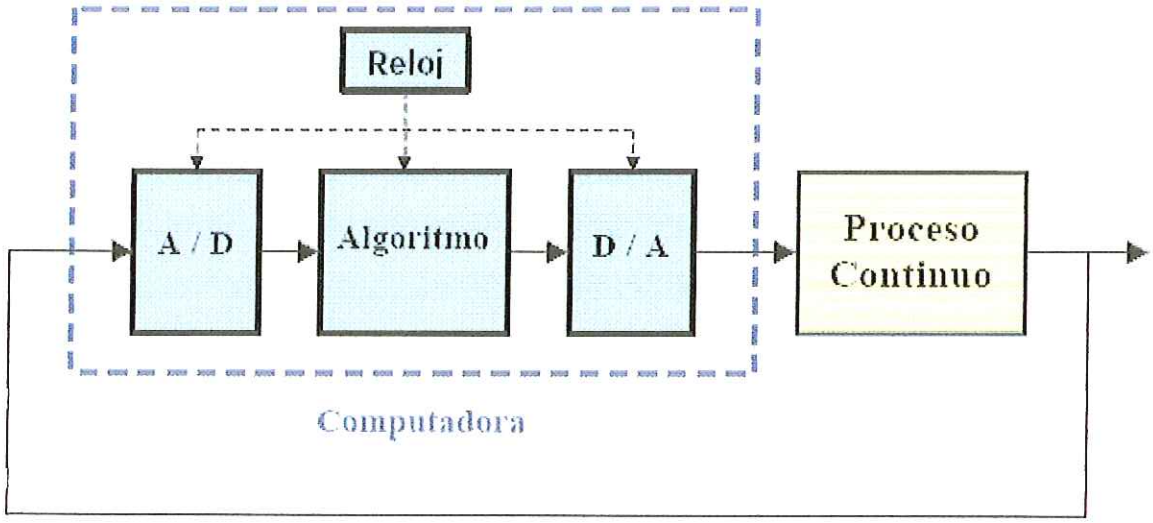
$$r[k] = R(Tk)^2 u[k] \rightarrow R(z) = \frac{RT^2(z^2+z)}{(z-1)^3}$$

MODELO DE SISTEMAS DE CONTROL DIGITAL

Comienzo por aclarar que el tipo de controladores al que me enfoco en estos apuntes es a los que realizan ***control retroalimentado de sistemas continuos,*** por ello se descartan los controladores en lazo abierto (salvo que sirvan para ilustrar aspectos básicos) y los controladores de secuencias discretas tales como los PLC's (o equivalentes) en su modalidad de control ON-OFF (La mayoría de los PLC's actuales contemplan también módulos analógicos que permiten utilizarlos en control retroalimentado de sistemas continuos).

En otras palabras, los controladores que se tratan aquí son computadoras, microcontroladores o cualquier sistema digital programable equipado con módulos de adquisición de datos analógicos (***Convertidores Analógico/Digital***) y Control analógico (***Convertidores Digital/Analógico***).

En la figura siguiente se muestran los elementos básicos que debe poseer un sistema de control retroalimentado basado en computadora.



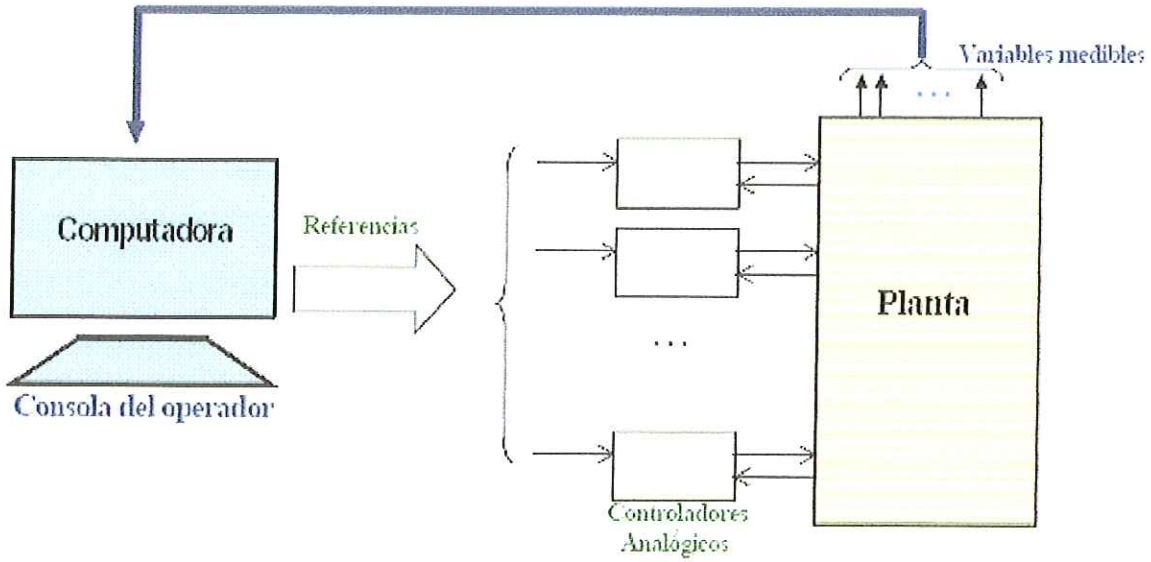
Esquema general de un control por computadora

Clasificación

Los esquemas generales de control digital retroalimentado han evolucionado considerablemente desde que comenzaron a utilizarse por allá en la década de los 60's. A continuación se presentan los esquemas más representativos utilizados en el control de procesos en industrias de gran envergadura.

Control Supervisorio

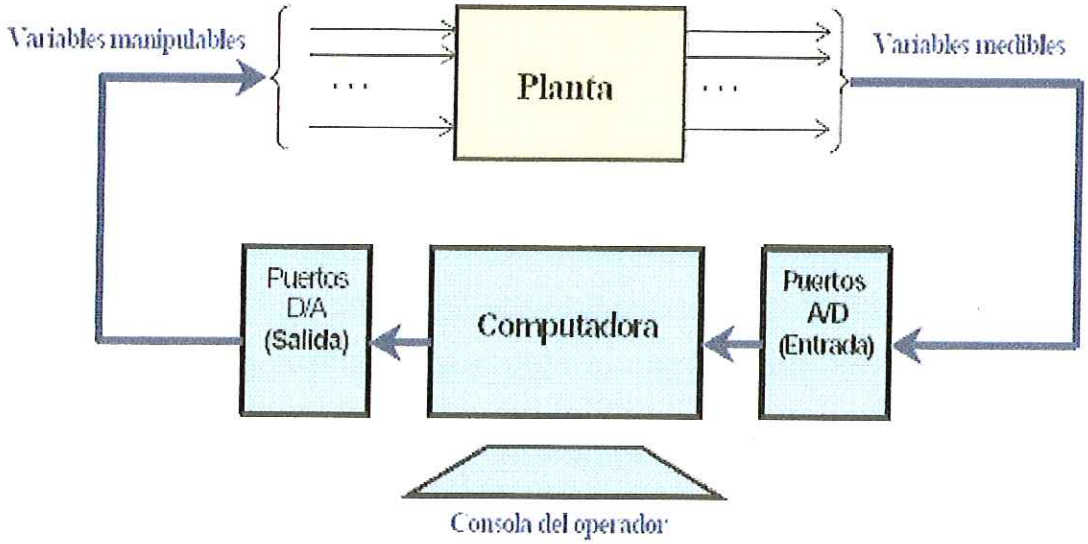
En la figura siguiente se muestra este esquema de control, el cual por orden histórico fue el primero en utilizarse. En este esquema la computadora juega solamente el papel de un supervisor, ya que no tiene acceso a ningún lazo de control y su única función es monitorear las variables controladas del proceso o bien, modificar las referencias de control (set points). Los lazos de control en este esquema se siguen realizando mediante ***controladores analógicos***.



Control Supervisorio

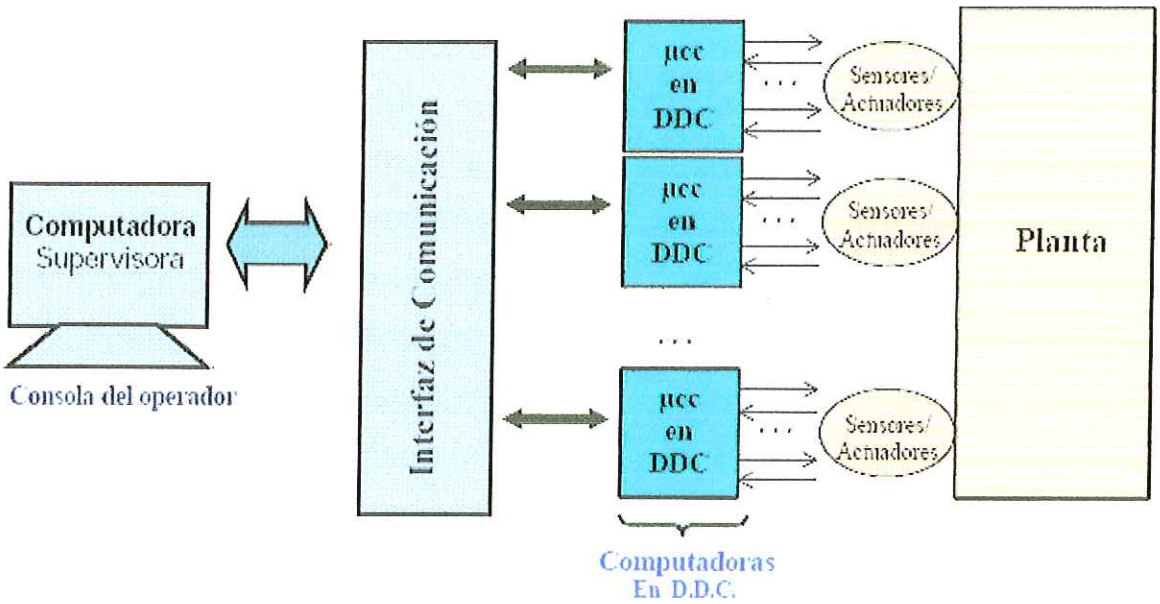
Control Digital Directo (DDC)

En la Figura siguiente se muestra el esquema de una computadora trabajando en control digital directo. En este esquema la computadora ejecuta uno o varios algoritmos de control para realizar directamente el control de una o varias variables de un proceso. Este esquema es al que se enfocan estos apuntes.



Control Digital Directo (DDC)

Control Distribuido



Control Jerárquico o Distribuido

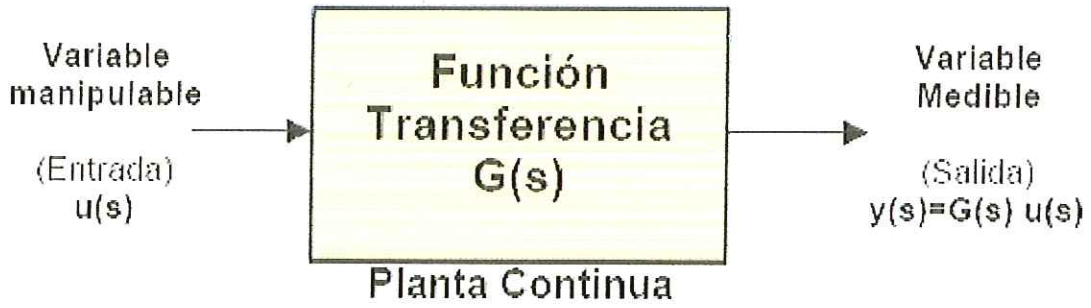
En este último esquema, que es el más difundido a nivel industrial en la actualidad se utilizan computadoras o microcontroladores para reemplazar los lazos de control individuales que en el esquema antiguo se implementaban con controladores analógicos. Además se usa una gran computadora de gran capacidad para realizar la función de supervisora que ya se describió en el esquema supervisor anterior, con la diferencia que en el nuevo esquema dicha computadora se auxilia de subsistemas que controlan una red local que sirve de interfaz de comunicación con cada controlador funcionando en control digital directo.

Modelos Matemáticos de Sistemas Continuos

Uno de los pasos que complica más el diseño de sistemas de control digital es la necesidad de modelar el proceso (continuo) que se desea controlar, ya que sin un modelo no podríamos decidir las acciones que el control deberá de tomar para corregir una desviación (respecto a lo deseado) de alguna variable del proceso.

Modelado Entrada- Salida

Teniendo en cuenta el último comentario uno de los enfoques de modelado más útiles para propósitos de control es el ***Modelado Externo*** o entrada/salida este tipo de modelo describe la relación estímulo - respuesta del proceso y conduce a la llamada **Función Transferencia** del proceso. Este enfoque de modelado se ilustra en la figura siguiente para una planta continua de una entrada y una salida.



Modelo Entrada-Salida para una planta continua

Aunque el modelado entrada – salida produce una representación muy conveniente, en el ***dominio del tiempo*** conduce a una relación expresada por la ***integral de convolución*** siguiente:

$$y(t) = \int_0^t g(t-\tau)u(\tau)d\tau \qquad (1)$$

Donde g(t) es la función de ***respuesta al impulso del sistema***. Buscando una relación más sencilla se expresa esta integral de convolución en el ***dominio de la frecuencia*** como una simple multiplicación de ***una función transferencia*** por la entrada para obtener la salida:

$$Y(s) = G(s)\ U(s) \qquad (2)$$

Ecuaciones Diferenciales

Todo sistema dinámico continuo (que contiene elementos almacenadores de energía) puede modelarse por una ecuación diferencial de orden n (donde n es el número de elementos almacenadores de energía). El caso más simple y más estudiado es el de una ***Ecuación Diferencial Lineal*** como la siguiente:

$$y^{(n)}(t)+a_1 y^{(n-1)}(t)+\ldots+a_{n-1}\dot{y}(t)+a_n y(t)=b_1 u^{(n-1)}(t)+\ldots+b_{n-1}\dot{u}(t)+b_n u(t) \quad (3)$$

Donde los superíndices indican derivación respecto al tiempo, y los coeficientes a1,...,an, b1,...,bn, son en general funciones del tiempo y corresponden a los parámetros del sistema (tales como masas, coeficientes de fricción, resistencias, capacitancias, inductancias, etc.). Si el sistema puede modelarse mediante una ecuación como la (2), se dice que es un ***Sistema Lineal***, y si los coeficientes a1,...,an, b1,...,bn son constantes se dice que es un ***Sistema Lineal Invariante en el Tiempo*** (***S.L.I.T.***).

Ecuaciones de Estado

Una ecuación diferencial de orden n como la (2) puede transformarse en un sistema de n ecuaciones diferenciales de primer orden mediante la introducción de n ***variables de estado***, x1(t), x2(t),...,xn(t). Así, en términos de estas variables de estado (si se eligen convenientemente, ya que hay una infinidad de maneras de hacerlo), la ecuación diferencial puede escribirse (bajo ciertas condiciones) como sigue:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \ldots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & -a_3 & \ldots & -a_n \\ 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \ldots \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} b_1 & b_2 & \ldots & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_3 \end{bmatrix} \quad (4)$$

La cual se denomina ***Forma Canónica Controlador*** y corresponde a la Función Transferencia en el dominio de Laplace siguiente:

$$\frac{y(s)}{u(s)} = G(s) = \frac{b_1 s^{n-1} + b_2 s^{n-2} + \ldots + b_{n-1} s^{n-1} + b_n s^n}{s^0 + a_1 s^{n-1} + \ldots + a_{n-1} s^{n-1} + a_n s^n} \qquad (5)$$

Ejemplo: Al sistema descrito por la siguiente ecuación diferencial:

$$\ddot{y} + 2\dot{y} + 3y = u(t)$$

Le corresponde el modelo en espacio de estado en su forma controlador siguiente:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -2 & -3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \qquad y(t) = x_2$$

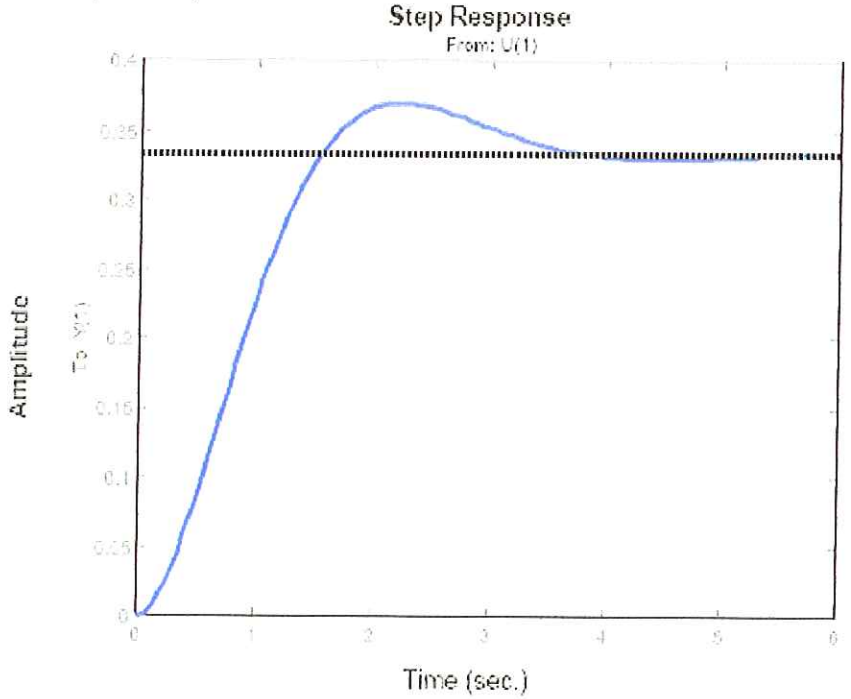
O bien, le corresponde la función transferencia:

$$G(s) = \frac{1}{s^2 + 2s + 3}$$

Por lo tanto, su respuesta al escalón puede ser obtenida mediante Matlab como sigue:

```
>>G=TF(1,[1 2 3]);
>>step(G);
```

Para obtener la respuesta que se muestra a continuación:



Respuesta al escalón unitario del sistema del ejemplo

PROGRAMACIÓN DIGITAL EN CASCADA

La manera más versátil de compensar un sistema de control de datos discretos es empleando un controlador digital. En general, éstos pueden implantarse con redes digitales, microprocesadores o procesadores de señales de señales digitales (PSD). En comparación con los controladores de datos continuos, los digitales permiten mejores desempeños. Otra ventaja es que el algoritmo de control puede modificarse con facilidad al alterar el programa del controlador de datos continuos es bastante difícil una vez que se encuentra implantado. La función de transferencia del controlador digital se describe mediante:

$$D(z) = \frac{E_2(z)}{E_1(z)} = \frac{b_m z^m + b_{m-1} z^{m-1} + \cdots + b_0}{a_n z^n + a_{n-1} z^{n-1} + \cdots + a_0} \quad (6)$$

La función de transferencia de un controlador físicamente realizable puede escribirse como:

$$D(z) = \frac{E_2(z)}{E_1(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} \quad (7)$$

La función de transferencia *D(z)* puede escribirse como el producto de varias funciones de transferencia más sencillas, cada una realizable con un programa digital. Con esto, la programación digital de *D(z)* puede representarse con una serie de programas digitales en cascada de funciones de transferencia más sencillas. Si se factoriza la ecuación (7), entonces:

$$D(z) = \prod_{k=1}^{p} D_k(z) \quad (8)$$

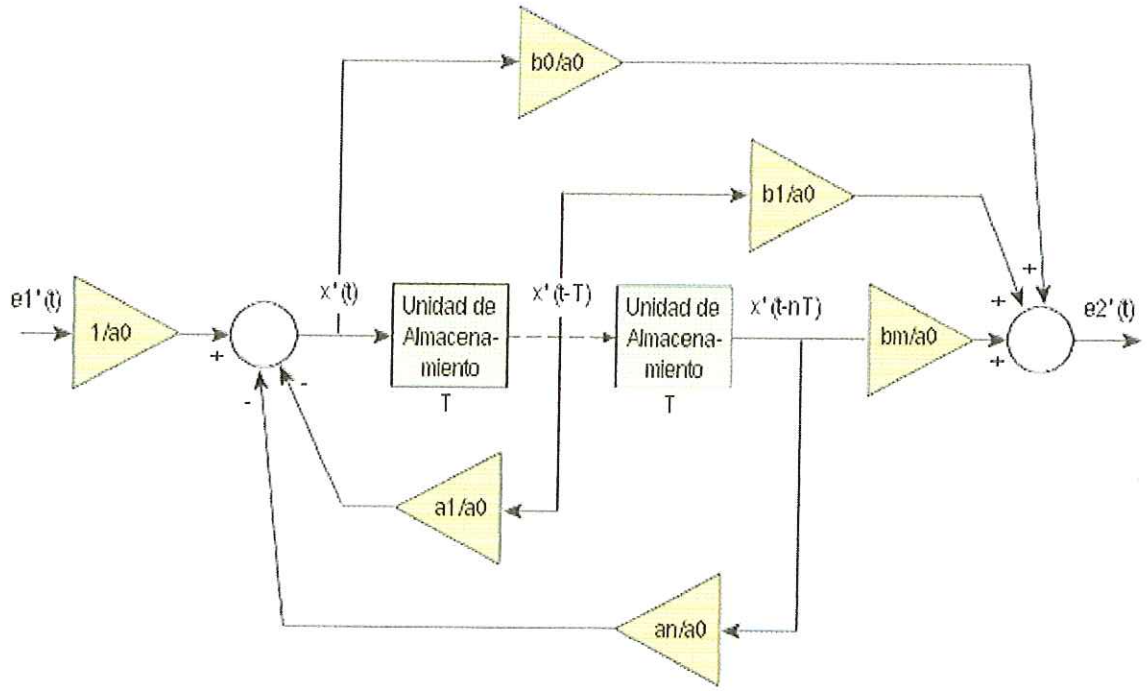


Diagrama de bloques para la programación digital directa

Donde p es mayor que n y m. La figura anterior contiene el diagrama de bloques de la representación de la programación digital en cascada de $D(z)$. En general, la función de transferencia $D_k(z)$ puede tener las siguientes formas, lo que depende de los polos y ceros de $D(z)$ y de las magnitudes relativas de m y n.

Polo y cero reales:

$$D_k(z) = K_k \frac{1 + c_k z^{-1}}{1 + d_k z^{-1}} \quad (9)$$

Dos polos complejos conjugados:

$$D_k(z) = K_k \frac{1}{1 + d_k z^{-1} + f_k z^{-2}} \quad (10)$$

Uno cero real y dos polos complejos conjugados:

$$D_k(z) = K_k \frac{1 + c_k z^{-1}}{1 + d_k z^{-1} + f_k z^{-2}} \quad (11)$$

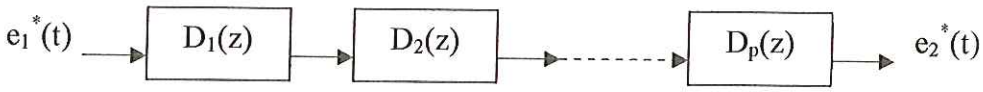


Diagrama de bloques de la programación digital en cascada de D(z)

Polos y ceros complejos conjugados:

$$D_k(z) = K_k \frac{1 + g_k z^{-1} + h_k z^{-2}}{1 + d_k z^{-1} + f_k z^{-2}} \quad (12)$$

Cero real (m > n):

$$D_k(z) = K_k (1 + c_k z^{-1}) \quad (13)$$

Ceros complejos conjugados (m > n):

$$D_k(z) = K_k (1 + g_k z^{-1} + h_k z^{-2}) \quad (14)$$

Un polo real y dos ceros complejos conjugados (m > n):

$$D_k(z) = K_k \frac{1 + g_k z^{-1} + h_k z^{-2}}{1 + d_k z^{-1}} \quad (15)$$

Todas estas funciones de transferencia son viables con el método de programación digital directa. El caso ilustrado por la ecuación (9) es quizá el más común, ya que en ella están representados los polos y ceros reales de *D(z)*. El caso de la ecuación (10) es para dos polos complejos conjugados de *D(z)*, ya que se desea evitar el manejo de números complejos en el problema digital. Los casos de las ecuaciones (11-15) son sólo algunas de las muchas configuraciones posibles en función de las magnitudes relativas de *n* y *m*.

Interacción Controlador Digital – Planta Continua

Al introducir un controlador digital para controlar una planta continua el modelado se complica un poco, dado que mientras la evolución de la planta se desarrolla en tiempo continuo, el controlador es un sistema que evoluciona bajo una base de ***tiempo discreto*** marcado por un reloj patrón. Para los fines de analizar la interacción control digital – planta continua el reloj que gobierna la evolución del controlador no es el reloj patrón de la CPU

de la computadora, sino el reloj que marca los ***instantes de muestreo*** tk de las tarjetas convertidoras A/D y D/A.

Muestreo Uniforme

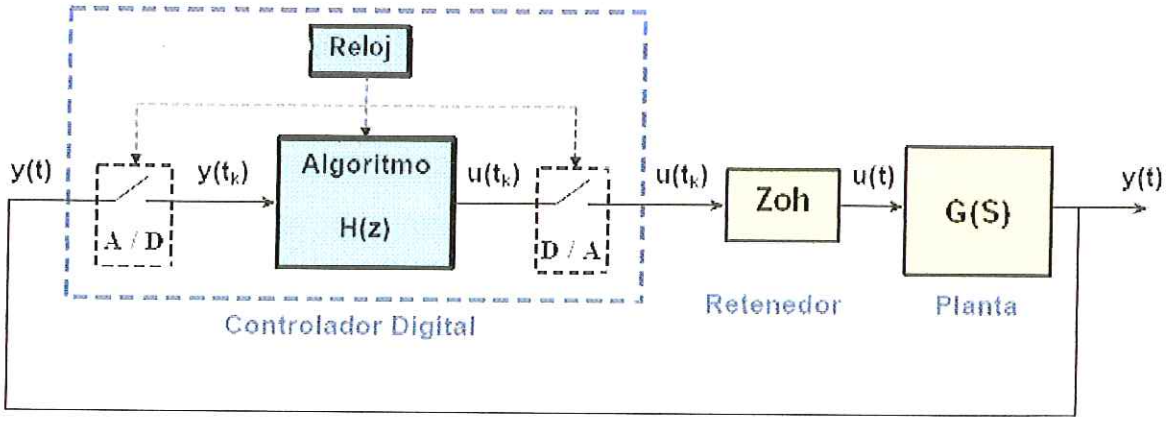
Los instantes que marcan el proceso de muestreo pueden estar espaciados de manera arbitraria, pero normalmente se tiene un control sobre ellos de manera que ocurren de manera periódica (uniformemente espaciados), por ello es conveniente definir en este caso un periodo de muestreo (**Ts**) y su correspondiente frecuencia de muestreo **fs** (en hertz) o bien, ws (en radianes), de esta manera, en el caso de muestreo uniforme se tiene que

$$\mathbf{tk = k\ Ts},\ \text{para } k=0,1,2,3,\ldots \quad (16)$$

Por ello es común "obviar" el periodo de muestreo y representar el instante de muestreo usando sólo el valor de k.

Modelado de los convertidores A/D y D/A

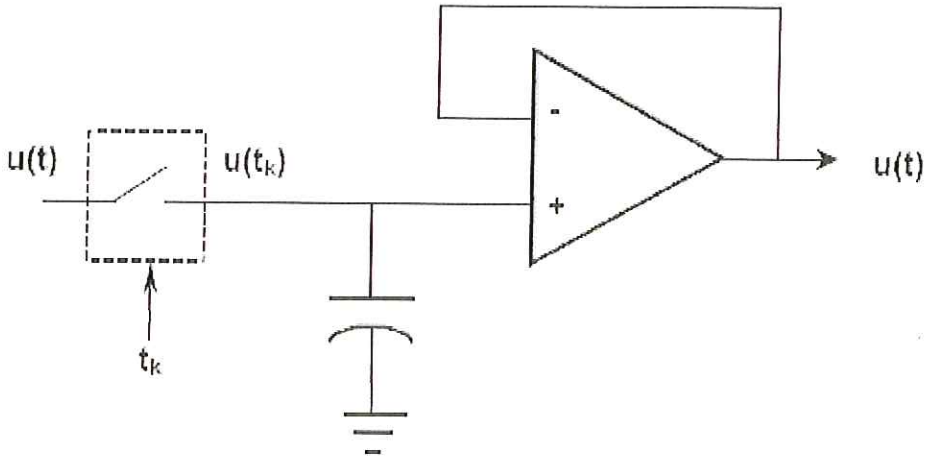
En la figura siguiente se muestra el esquema general de un control por computadora, en donde se utiliza un controlador digital para controlar una planta continua, pero ahora el convertidor A/D ha sido reemplazado por su equivalente para fines de modelado, el cual es un ***muestreador***, es decir, un switch controlado que siempre está abierto y sólo se cierra para tomar muestras en los instantes de muestreo. En forma similar, el convertidor D/A se ha reemplazado por un ***muestreador*** en serie con un ***retenedor***.
La función del retenedor es la de "sostener" el valor de la señal de entrada entre un instante de muestreo tk y el siguiente tk+1.



Modelado de los conversores A/D y D/A

El retenedor de orden cero (Zoh)

Un retenedor analógico práctico denominado retenedor de orden cero. (Zoh) consiste en un capacitor que almacena el valor muestreado u(tk) proveniente de un muestreador, con el conveniente acoplamiento de impedancia que puede ser proporcionado por un amplificador operacional en configuración de seguidor como se muestra a continuación:

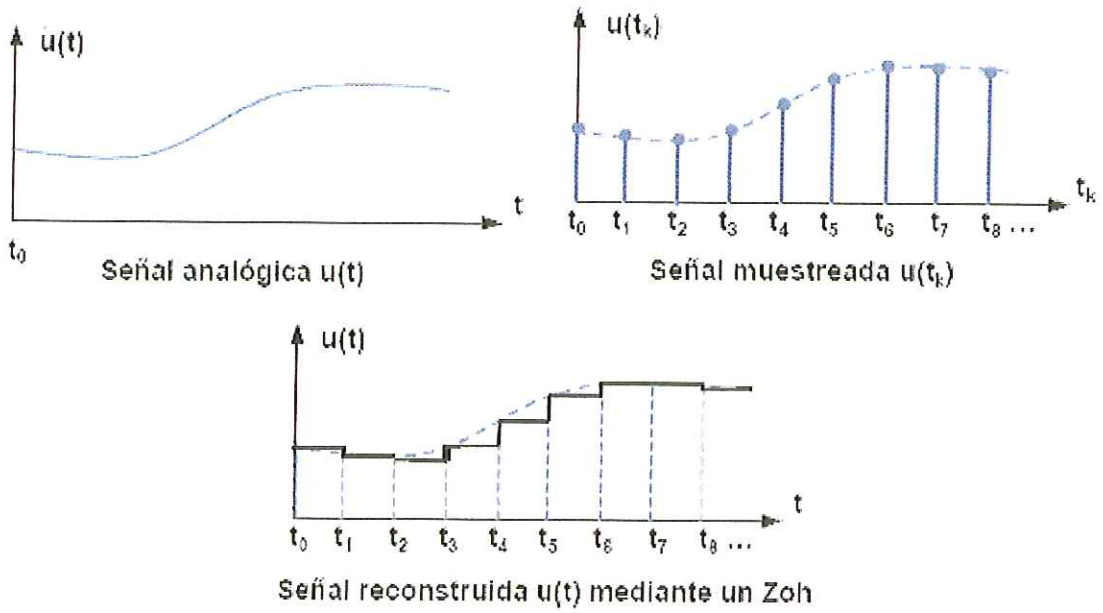


Muestreador / Retenedor de orden cero

En un retenedor ideal, el objetivo es realizar el proceso inverso al ***muestreo***, es decir, la ***reconstrucción*** de la señal u(t) previa al muestreador. Sin embargo, el proceso de reconstrucción normalmente no es perfecto, y lo que se obtiene es una aproximación a la señal u(t), en la figura siguiente se muestran señales típicas que tienen lugar en el muestreo

y la reconstrucción con un retenedor de orden cero.

Observación: El proceso de retención también puede realizarse de manera digital, mediante un registro que sostenga el dato digital u(tk) mientras el convertidor D/A realiza la conversión.



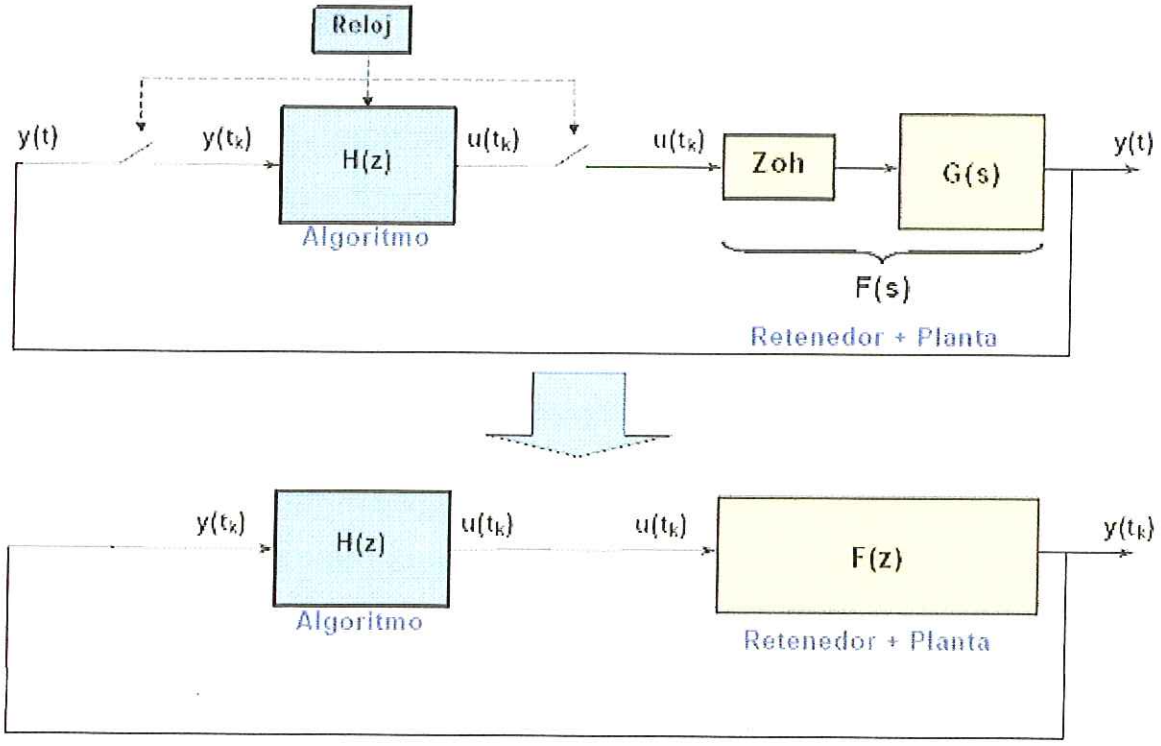
Señales típicas en el proceso de muestreo y reconstrucción

Un enfoque de modelado del sistema de control digital es adoptar ***el punto de vista de la computadora***, la cual es un sistema de tiempo discreto y por lo tanto, "ve" a la planta como si fuera otro sistema de tiempo discreto, ya que sólo puede ver de ella los valores muestreados de y(t) en cada instante de muestreo.

Este enfoque sugiere modelar todos los componentes del sistema de control mostrado en la figura *Modelado de los conversores A/D y D/A* en un dominio de tiempo discreto, es decir, como si la planta fuese discreta, como se ve en la figura siguiente, esto obliga a considerar un equivalente F(s) de la planta que incluye al retenedor Zoh como si este último fuese parte de la planta.

Un enfoque alternativo consiste en considerar todos los componentes como si fueran analógicos, esto conduce a considerar que la computadora es solamente una aproximación digital para un controlador continuo. Este enfoque es muy restrictivo, ya que no permite

explotar las capacidades de muchos algoritmos digitales que no son una aproximación de controladores continuos.



Modelado desde el punto de vista de la computadora

Por ello el enfoque más conveniente es el mostrado en la figura 4.9, en la cual se ha usado la variable **z** para indicar que el modelado es en tiempo discreto.
A continuación se presenta el equivalente natural del modelado en el dominio del tiempo continuo (ecuaciones diferenciales), que en el caso de tiempo discreto conduce a las ecuaciones de diferencias.

Ecuaciones de Diferencias

En forma similar a como una ecuación diferencial establece una relación entre una variable y(t) y sus derivadas, una ecuación de diferencias establece una relación entre una variable de tiempo discreto y(k) y sus valores pasados y(k-1), y(k-2), etc.
En esta notación k representa el instante de muestreo y se evita usar tk por simplicidad.

Si en una ecuación de diferencias aparecen involucrados los valores de y(k) en diferentes instantes anteriores, por ejemplo, y(k-1), y(k-2), ..., y(k-n), donde n es el máximo valor de la lista (y(k-n) es el valor más anterior de todos), a la ecuación se le llama ***ecuación de diferencias de orden n***.

Ejemplo: La siguiente es una ***ecuación de diferencias de primer orden***, ya que y(k) depende solamente de sus valores un instante antes de k:

$$y(k) = 1 + a\, y(k-1)$$
(valor presente = 1 + valor anterior)

la cual también puede escribirse como:

$$y(k+1) = 1 + a\, y(k)$$
(valor siguiente = 1 + valor presente)

Es fácil encontrar la solución de la ecuación anterior por simple iteración. Así, suponiendo condiciones iniciales cero, es decir, y(0)=0, tendríamos:

Para k=1: $y(1)=1+a\, y(0) = 1$

Para k=2: $y(2)=1+a\, y(1) = 1 + a$

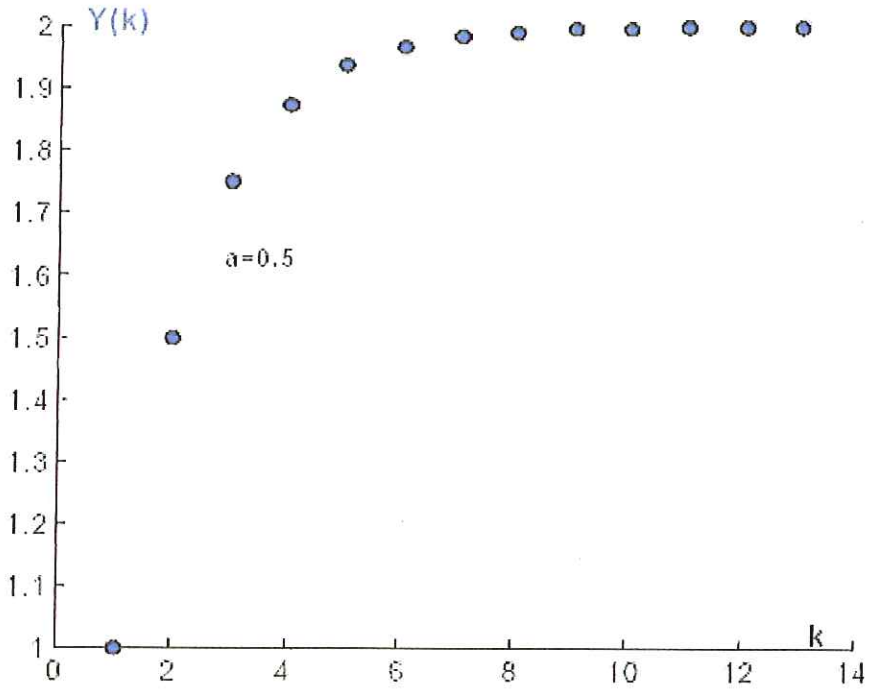
Para k=3: $y(3)=1+a\, y(2) = 1+a(1+a)= 1+a+a^2$

Para k=4: $y(4)=1+a\, y(3) = 1+a(1+a+a2) = 1+a+a_2+a_3$

...

para k=n: $y(n)= 1 + a + a^2 +...+ a^{n-1}$

Esta última expresión nos da la ***solución particular de la ecuación de diferencias*** para la condición inicial y(0)=0. Dicha solución tiene características diferentes dependiendo del valor de la constante **a**. En la figura siguiente se muestra la forma de la solución y(k) para una valor de **a = 0.5**.



Solución de la ecuación de diferencias del ejemplo para a=0.5

Ecuaciones Lineales de Diferencias

En forma similar a las ecuaciones diferenciales, las de diferencias pueden clasificarse también en Lineales y No Lineales. La forma general de una ecuación de diferencias lineal de orden n es como sigue:

$$y(k+n) + a1y(k+n-1) +...+ any(k) = b0u(k+m) + b1u(k+m-1) +...+ bmu(k) \quad (17)$$

Donde:

y es la variable dependiente (salida).

u es una variable independiente (entrada).

k es el tiempo discreto.

a1,...,an,b0,...,bm, son en general funciones del tiempo **k**.

En el caso en que los parámetros **a1,...,an,b0,...,bm** sean constantes, la ecuación (17) se dice ***Lineal, Invariante en el Tiempo***. Este es el caso más sencillo y será el único tratado en estos apuntes.

Causalidad

Se puede ver de la ecuación (17) que si m>n entonces los valores de y en cada instante dependerían de un valor que aún no ha ocurrido (de un valor futuro), por ejemplo, una ecuación con m=1, n=0 pudiera ser:

$$y(k)=u(k+1),$$

la cual para evaluarse en el instante presente k=1 requiere conocer el valor de u(2) que está en el futuro!!.

Un sistema descrito por la ecuación (17), con m>n se llama **No Causal** y no es realizable físicamente en ***tiempo real***. En forma similar, si m £ n el sistema se dice *Causal*.

Observación. Pudiera pensarse que al no ser realizables los sistemas no causales no tienen ningún interés, sin embargo, si el sistema trabaja con datos almacenados no tiene porque procesarlos en tiempo real y entonces puede realizarse aún siendo no causal. Otro caso de interés en sistemas no causales es cuando **k** no es una variable de tiempo, sino, por ejemplo, de posición, entonces los valores de **k** mayores que el actual no son "futuros", sino simplemente están "adelante".

PROCESOS TRANSITORIOS DE ARRANQUE Y DE FRENAJE EN LOS MOTORES DE CORRIENTE DIRECTA EXCITADOS SEPARADAMENTE

Los procesos transitorios de arranque y de frenaje constituyen fenómenos frecuentes en el trabajo de los accionamientos eléctricos. Se realizará este proceso con el fin de analizar el comportamiento del sistema ante el arranque y frenaje proporcionado por el Driver

Toshiba, a los cuales van conectados cada uno de los motores del manipulador y para obtener criterios de diseño.

Por *arranque* de un accionamiento eléctrico se entenderá el proceso en el cual el motor alcanza desde su estado de reposo un valor estacionario de la velocidad. Mientras que por *frenaje* se entenderá la disminución de la velocidad de la máquina, la cual podrá inclusive alcanzar su estado de reposo.

Los motores de corriente directa no admiten arranque directo debido a que los picos de corriente que se producen superan los permisibles desde el punto de vista de la conmutación. Por ello el arranque se produce mediante la conexión a la red del motor a través de resistencias insertadas en su circuito de armadura. A medida que el motor se acelera, estas resistencias se van cortocircuitando. De esta forma el arranque se realiza en varios pasos y el motor y el motor va pasando de una característica reostática a otra. Al final, el motor alcanza su velocidad de trabajo de acuerdo con la carga conectada a su eje y con su característica natural.

Para diseñar el arrancador de un motor de corriente directa con excitación separada deberá conocerse la característica natural del motor y el valor límite de la corriente o, el momento de arranque del sistema, el cual no podrá ser mayor que el correspondiente al límite de conmutación y deberá ser calculado a partir del valor requerido de aceleración del mecanismo por el motor accionado.

ϕ = Flujo de excitación.

ω = Velocidad angular.

ρ = # de pares de polos del motor.

U = Voltaje.

E = Fuerza electromotriz inducida en los devanados de armadura.

T_a = Constante de tiempo eléctrica del sistema.

$$T_a = \frac{L_{am} + L_{ag}}{R_{am} + R_{ag}}$$

ag = armadura del generador.

am = armadura del motor.

e_m = FEM del motor.

M_c = Momento de la carga.

M = Momento del motor.

e_g = Voltaje de a fuente.

El valor de la resistencia total o de arranque del motor se puede calcular a partir de la fórmula:

$$R_{a1} = \frac{U}{I_{a1}} \quad (18)$$

El resto de los pasos de resistencia se pueden hallar a partir de la expresión de la característica mecánica del motor y sobre la base del diagrama de arranque:

$$\varpi = \frac{U}{K\phi} - \frac{R_a}{(K\phi)^2} M$$

Si se le considera R_{a1} a la resistencia de la característica reostática 1 y R_{a2} a la correspondiente a la número 2 y así sucesivamente, se puede escribir que:

$$\Delta\varpi_1 = \frac{M_2 R_{a1}}{(K\phi)^2} = \Delta\varpi_2 = \frac{M_1 R_{a2}}{(K\phi)^2} \quad (19)$$

De donde se puede obtener el valor de la resistencia R_{a2} de la segunda característica y el valor del primer paso de resistencia: $R_{a2} = R_{a1}\frac{M_2}{M_1}$ y $R_{p1} = R_{a1}\left(1 - \frac{M_2}{M_1}\right)$. Un planteamiento similar se puede hacer para las características 2 y 3 de donde se obtiene que:

$$R_{a3} = R_{a2}\frac{M_2}{M_1} = R_{a1}\left(\frac{M_2}{M_1}\right)^2 \quad \text{y} \quad R_{p2} = R_{a1}\frac{M_2}{M_1}\left(1 - \frac{M_2}{M_1}\right)$$

Así para *i* pasos de resistencia se tendrá que el valor de los pasos y el de las resistencias podrán ser calculadas por:

$$R_{pi} = R_{ai} - R_{a(i+1)} = R_{a1}\left(\frac{M_2}{M_1}\right)^{i+1}\left(1-\frac{M_2}{M_1}\right) \quad (20)$$

$$\text{y} \quad R_{a(i+1)} = R_{a1}\left(\frac{M_2}{M_1}\right)^{i} \quad (21)$$

Si se supone, por ejemplo, un número de pasos m se puede escribir que:

$$R_{a(m+1)} = R_a = R_{a1}\left(\frac{M_2}{M_1}\right)^{m}$$

De donde puede ser hallado el valor de M_2:

$$M_2 = M_1 \sqrt[m]{\frac{R_a}{R_{a1}}} \quad (22)$$

Entonces cuando se dan M_1 y m se puede determinar M_2, el cual deberá ser mayor que el momento de la carga (M_c). En caso de que no cumpla esta condición la cantidad de pasos deberá ser aumentada.

Se puede, además, disponiendo de los valores de M_1 y M_2 hallar el valor del número de pasos de arranque:

$$m = \frac{\log\frac{R_a}{R_{a1}}}{\log\frac{M_2}{M_1}} \quad (23)$$

El valor de m obtenido es redondeado y, aplicando nuevamente la expresión (22) se obtiene el valor de M_2 el cual deberá cumplir la condición de ser mayor que el momento de la carga (M_c) al eje del motor.

Una vez determinados los pasos de resistencia es necesario calcular los tiempos de aceleración, para lo cual se partirá de la ecuación del accionamiento eléctrico y del concepto de constante de tiempo mecánica arriba definido, de manera que:

$$M - M_c = T_M \frac{dM}{dt} \qquad e\int dt = \int_{M_1}^{M_2} T_M \frac{dM}{M - M_c}$$

De donde:

$$t = T_M \ln \frac{M_1 - M_c}{M_2 - M_c} \quad \text{y} \quad t_a = \sum T_{Mi} \ln \frac{M_1 - M_c}{M_2 - M_c} \qquad (24)$$

Donde:

t_a – tiempo total de aceleración del motor y T_{Mi} cambia con las características macanas que poseen diferente rigidez.

Para determinar las características transitorias de arranque de este motor ω(t) e i(t) o M(t), se parte del sistema de ecuaciones:

$$M - M_c = J\frac{d\omega}{dt} \qquad M = M_{kz} - \beta\omega \qquad M_c = M_{co} - \beta_c\omega \qquad (25)$$

Donde:

M_{kz} y M_{co} – valores del momento del motor y de la carga para velocidad cero.

β y β_c – módulo de la rigidez de las características mecánicas del motor y de la carga respectivamente.

A partir de (25) se obtiene que:

$$T_M \frac{d\omega}{dt} + \omega = \omega_{est} \qquad (26)$$

Donde:

$$T_M = \frac{J}{\beta + \beta_c} \quad \text{y} \quad \omega_{est} = \frac{M_{kz} - M_{co}}{\beta + \beta_c}, \text{ valor estacionario de la velocidad.}$$

La solución de esta ecuación diferencial de primer orden posee una respuesta libre y otra forzada, siendo su valor total:

$$\omega = \omega_{est} + (\omega_{ini} - \omega_{est})e^{-1/T_M} \qquad (27)$$

Debido a que existe una relación lineal entre la velocidad y momento se tiene que:

$$M = M_{est} + (M_{ini} - M_{est})e^{-1/T_M} \qquad (28)$$

Observación: Para el caso específico de nuestro proyecto no se siguió el procedimiento anterior por que se trata de un prototipo, el cual se compone de motores pequeños, los cuales responden casi instantáneamente al arranque. De todas formas se planteó el procedimiento de diseño, especialmente para mecanismos grandes, como por ejemplo, manipuladores industriales robustos, en este caso los motores son más grandes y las características de arranque requieren de un análisis cuidadoso y consecuentemente de un

proceso de diseño.

Frenaje

Para la detención rápida de los motores con frecuencia se utiliza un frenaje eléctrico. La realización de frenaje por contracorriente de un motor de corriente directa, se lleva a cabo mediante la inversión de los terminales de armadura del motor. Al mismo tiempo en el circuito de armadura del motor se introduce una resistencia con el objetivo de limitar las corrientes durante el proceso de frenaje del motor. A diferencia del arranque, el frenaje se ejecuta con un solo paso de resistencia con el objetivo de disminuir la instalación de conmutación. Durante el proceso de frenaje el momento varía entre los valores $-M_1$ y $-M_2$. Para este sector de la característica de frenaje se puede escribir la expresión:

$$\omega = -\omega_{ini}\frac{M + M_2}{M_1 - M_2} \quad (29)$$

El momento máximo de frenaje $M_{ini} = -M_1$ que tiene lugar al inicio del frenaje se determina para las máquinas de corriente directa de ejecución normal como $M_1 \cong 2.5\ M_N$. Conocido el valor permisible de la corriente para el frenaje, se puede determinar la resistencia necesaria a insertar en el circuito de armadura a través de la expresión:

$$R_{aa} = \frac{U_c + E}{I_a} - R_a$$

Las características transitorias $\omega(t)$ y $M(t)$ durante el proceso de frenaje por contracorriente se determina a partir de las expresiones (27) y (28). De acuerdo con (29) se puede decir que:

$$\omega_{est} = -\omega_{ini}\frac{M_c + M_2}{M_1 - M_2}$$

Entonces:

$$\omega = \omega_{ini}\frac{M_1 + M_c}{M_1 - M_2}e^{-1/T_M} - \omega_{ini}\frac{M_c + M_2}{M - M_2} \quad (30)$$

$$M = -(M_1 - M_c)e^{-1/T_M} + M_c \quad (31)$$

Donde:

$$T_M = J\frac{\Delta\omega}{\Delta M} = J\frac{\omega_{ini}}{M_1 - M_2}$$

El tiempo de frenaje se puede determinar utilizando la expresión (24):

$$t_{fc} = T_M \ln\frac{M_1 + M_2}{M_2 + M_c} \quad (32)$$

El frenaje por corriente posibilita de manera rápida detener el motor, pero se encuentra asociado a pérdidas significativas de energía. Más económico resulta el frenaje dinámico. En el proceso de frenaje, la velocidad del motor disminuye de acuerdo con la característica mecánica.

La resistencia a insertar en el circuito de armadura puede ser calculada mediante el uso de la expresión:

$$R_{aa} = \frac{E_{ini}}{I_{perm}} - R_a \quad (33)$$

Donde:

E_{ini} – valor de la fem del motor al inicio del frenaje.

I_{perm} – corriente a limitar durante el proceso de frenaje.

Debido a que en este caso $\omega_{est} = -\omega_{ini}\frac{M_c}{M_1}$, entonces sustituyendo esta expresión en (27) y (28) se tiene que:

$$\omega = \omega_{ini}\frac{M_1 + M_c}{M_1}e^{-1/T_M} - \omega_{ini}\frac{M_c}{M_1} \quad (34)$$

$$\text{y} \quad M = -(M_1 + M_c)e^{-1/T_M} + M_c \quad (35)$$

El tiempo de frenaje dinámico se calcula a través de la expresión:

$$t_{fd} = T_M \ln\frac{M_1 + M_c}{M_c} \quad (36)$$

Observación: Para el caso particular del sistema de control del manipulador PUMA, se usó un Driver Toshiba, el cual tiene la capacidad adicional de aplicar una señal de freno casi instantáneo. Las pruebas se hicieron para cada una de las articulaciones, la más crítica fue

la del motor que hace girar todo el manipulador, ya que este es muy pequeño y la articulación tiene doble rodamiento con balines grandes, lo que lo hace más propenso a la inercia de la carga. Se calculó aproximadamente que este motor se detiene a los 2 segundos después de haberle quitado la energía eléctrica, pero, con el Driver utilizado, el tiempo de frenaje es prácticamente instantáneo, no hay efectos de inercia. El procedimiento de frenaje anterior se podría aplicar en el primer caso, en donde no hay Driver capaz de detener el motor, pues es un efecto natural de los motores de corriente continua, y por tanto se debe realizar un cálculo de diseño para observar su comportamiento; con el uso de este tipo de Drivers estos cálculos no pueden ser utilizados, ya que estos integrados efectúan un sistema de frenaje especial sobre los motores a controlar y aseguran un frenaje casi instantáneo sin dañar el motor.

DIAGRAMA DE BLOQUES Y ANALISIS EN MATLAB

La función de transferencia para los motores de las articulaciones son similares, y son de la siguiente forma:

$$G_p(s) = \frac{20}{0.2s + 1} \quad (37)$$

Como el control del sistema es digital, se debe hacer un análisis del sistema en tiempo discreto, por lo tanto, la ecuación (37) debe pasar de continuo a discreto, por medio de un programa en Matlab:

```
num = [0 20]
den = [0.2 1]
Ts = 0.1
[numd, dend] = c2dm(num, den, Ts, 'method')
```

La función de transferencia discreta se obtiene por medio del numerador y denominador obtenidos en numd y dend respectivamente, obteniendo lo siguiente:

$$G_p(z) = \frac{7.8694}{z - 0.6065} \quad (38)$$

La función de transferencia del microcontrolador (polos y ceros reales) va a ser la siguiente:

$$D_k(z) = K_k \frac{1 + c_k z^{-1}}{1 + d_k z^{-1}}$$

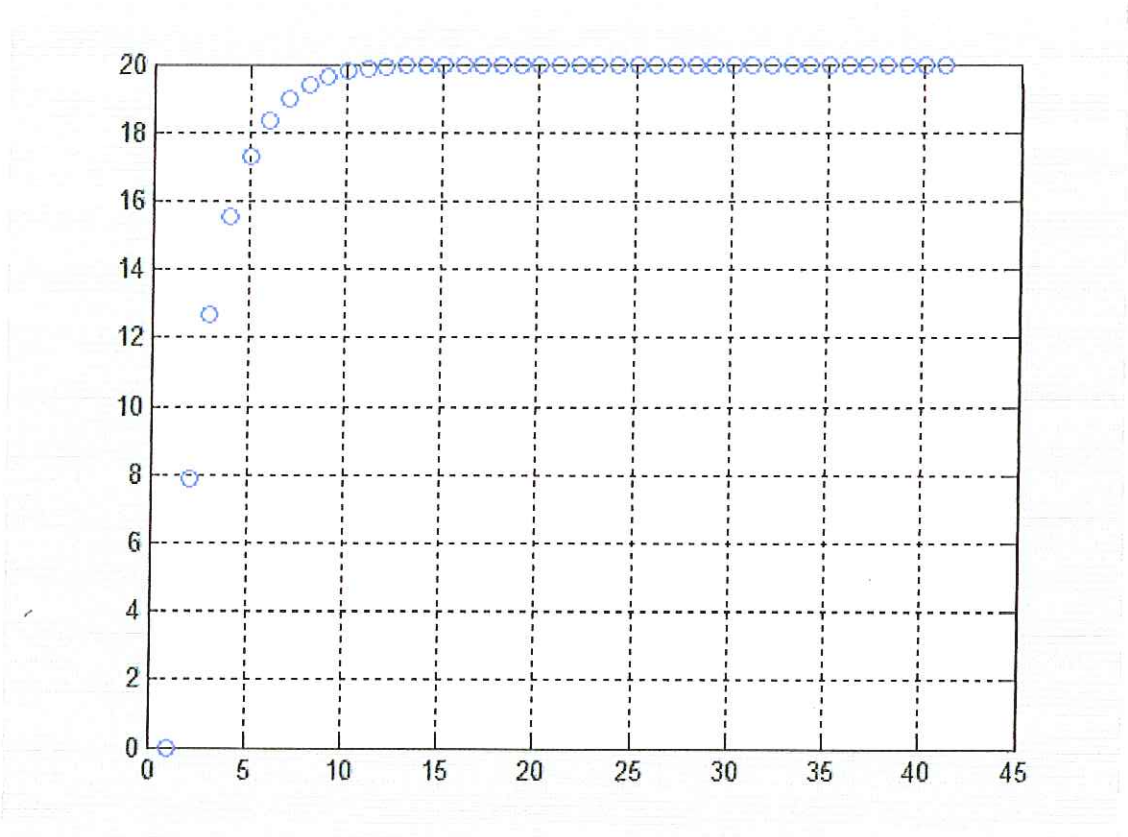
En donde $K_k = 1$, $c_k = 0.5$ y $d_k = 0.2$, entonces se obtiene:

$$D_k(z) = \frac{1 + 0.5z^{-1}}{1 + 0.2z^{-1}} = \frac{z^2 + 0.5z}{z^2 + 0.2z} \quad (39)$$

Con la función de transferencia del motor (38) se puede hacer el primer análisis en Matlab con el siguiente programa:

```
num=[0 7.8694];
den=[1 -0.6065];
u=ones(1,41);
v=[0 40 0 1.6];
axis(v);
y=filter(num,den,u);
plot(y,'o')
grid
title('Respuesta a un escalón unitario')
xlabel('k+1')
ylabel('y(k)')
```

Y se obtiene la siguiente gráfica para el motor de corriente continua:



1.10.2 Sistema de control en tiempo discreto

Los sistemas de eventos discretos son sistemas en los que le tiempo y los estados son continuos. El estado del sistema puede variar instantáneamente en instantes separados de tiempo. Un evento es un suceso instantáneo que puede cambiar el estado del sistema. En un intervalo de tiempo finito no puede haber un número infinito de cambios de estados.

Es necesario encontrar un modelo que describa el funcionamiento del sistema. El cual será un modelo funcional del sistema que describirá el funcionamiento deseado de la máquina. Pueden existir funcionamientos diferentes para la misma máquina, si las acciones a realizar son diferentes. El modelo debe relacionar las

salidas o acciones sobre el sistema con las entradas y órdenes de mando para poder ser implementado en la unidad de control o automatismo.

MODELO MEDIANTE GRAFOS DE ESTADOS

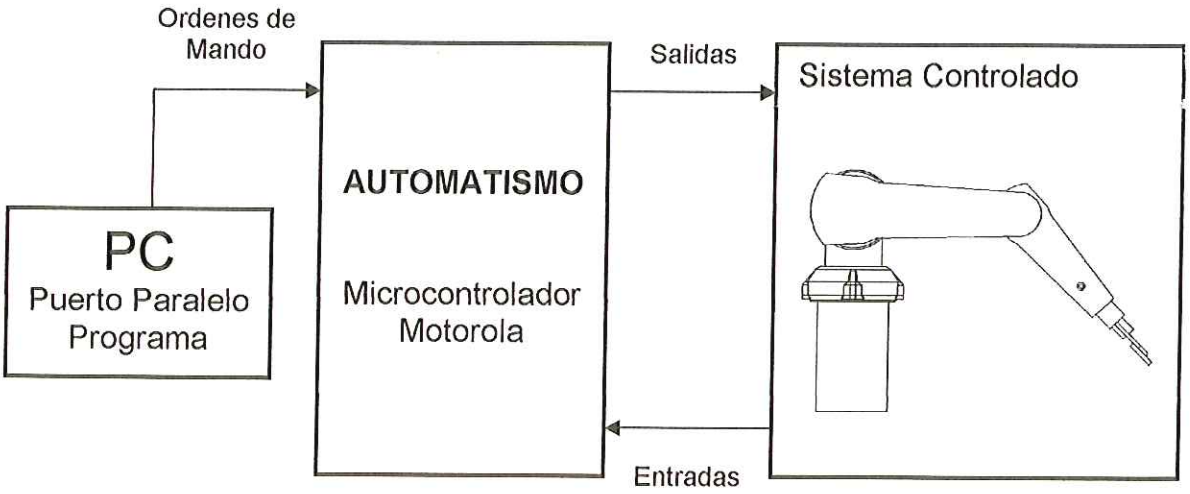


Figura 34. MODELO MEDIANTE GRAFOS DE ESTADOS 01

El anterior esquema cumple con la función del automatismo:

Salida = Función (Entradas, Órdenes de mando)

El automatismo para este sistema se encuentra principalmente en el Microcontrolador Motorota, en donde ocurren todas las entradas y salidas del sistema. El computador es el que da las ordenes de mando al automatismo. Las salidas del automatismo son una función booleana combinatoria de las entradas y las ordenes de mando. Las siguientes son las características del automatismo:

Ordenes de mando (M):

A: Ángulo deseado
Nm: # de motor

Entradas (E):

EP: Estado del potenciómetro

Salidas (S):

Sd: Sentido a la derecha del motor
Si: Sentido a la izquierda del motor
F: Freno del motor
Ms: Motor seleccionado del manipulador
i: Intervalo de tiempo para la secuencia de estados en ms

Estados (Q):

D: Sentido derecha
I: Sentido Izquierda
R: Reposo

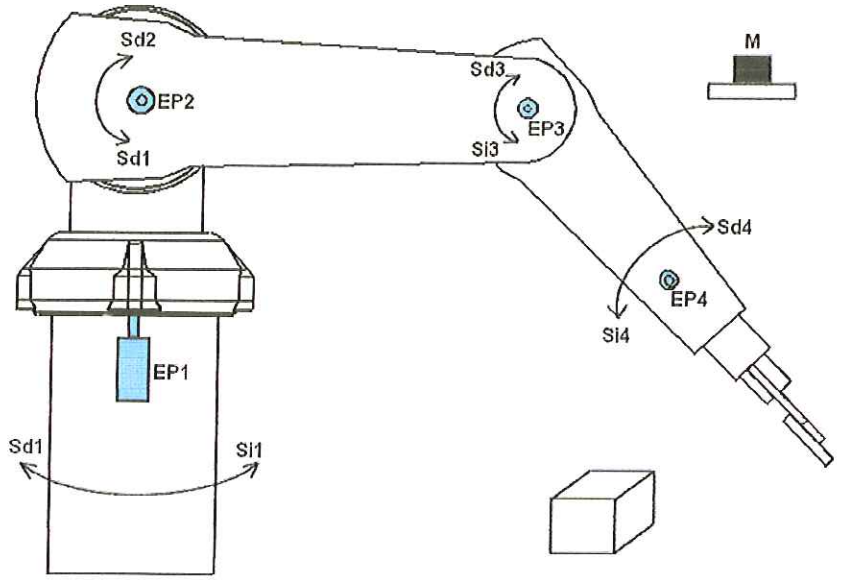


Figura 35. MODELO MEDIANTE GRAFOS DE ESTADOS 02

En la gráfica anterior se pueden observar las salidas y las entradas del automatismo. Las salidas son Sd (sentido derecha) y Si (sentido izquierda). El botón "M" simboliza las ordenes de mando que provienen del computador y que hacen cambiar de estado el sistema, en el estado de reposo, por ejemplo, el botón "M" daría la orden de arranque para el manipulador.

El grafo de estados del manipulador es:

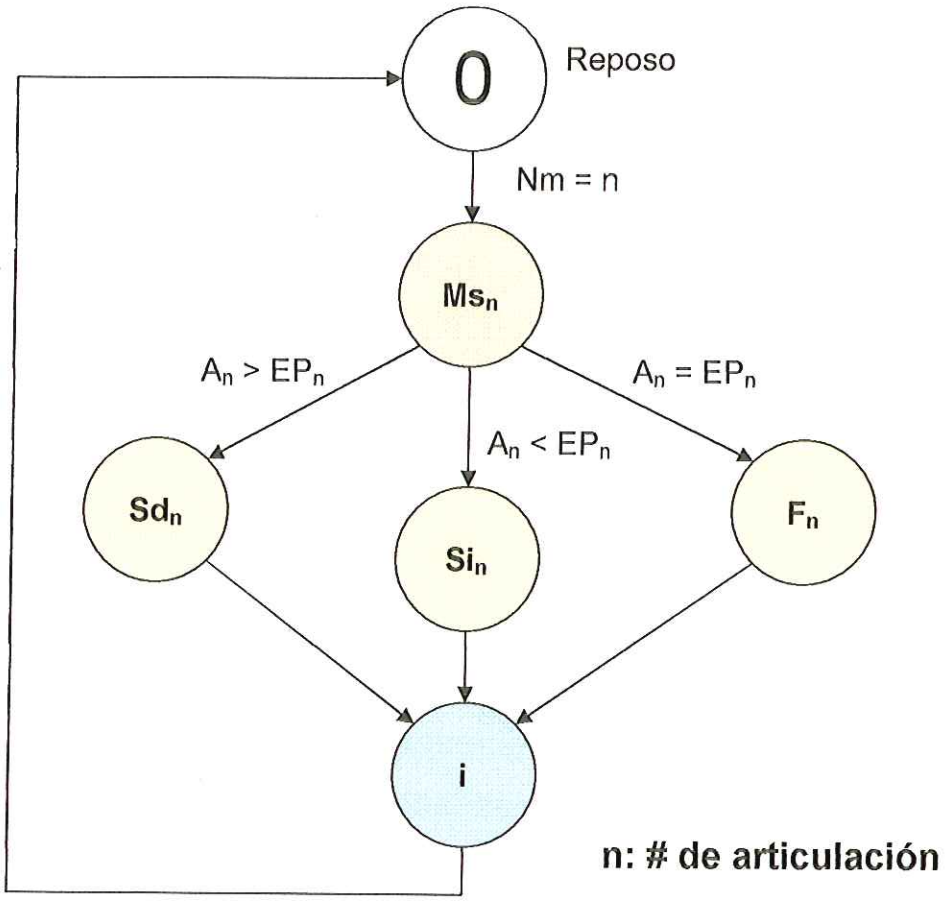


Figura 36. MODELO MEDIANTE GRAFOS DE ESTADOS 03

En la tabla anterior se explica el significado de cada una de las variables del diagrama. El modelo realiza una secuencia repetitiva. Primero se selecciona el motor a activar. Las activaciones para los motores (en amarillo) se realizan a partir de condiciones y se llega finalmente a un intervalo de tiempo, esto se repite n veces, siendo n el numero de articulaciones, las cuales son cuatro.

Diseño e implementación del control

Para el sistema de control se implementó un control en lazo cerrado. Debido a que los motores CD manejan altas velocidades, se hace difícil implementar sistemas de control de lazo abierto y posicionarlo de manera exacta y confiable, pero al implementar un sistema de retroalimentación por medio de potenciómetros

ubicados en cada una de las articulaciones, se hace posible tener un posicionamiento correcto y grandes velocidades, las cuales se pueden aplicar a trenes de engranajes robustos. Los potenciómetro en todo momento informan al sistema de control de la posición de cada articulación, a diferencia de los *encoders*, y se hace posible decidir en que sentido hacer girar el motor a partir de la posición actual del potenciómetro. Los potenciómetros implementados al manipulador son lineales a 5kΩ. La linealidad en una característica absolutamente importante, pues, la posición es directamente proporcional al nivel de voltaje que este provee (0 a 5V), es así como se asegura un posicionamiento uniforme. De esta forma el programa de control puede activar determinado motor hasta que en el potenciómetro se llegue a la posición deseada.

Los sistemas de control basados en microcontroladores y ordenador, manejan señales digitales, por tanto se aplica un control en tiempo discreto al sistema. Como las salidas de estos dispositivos producen cambios en las articulaciones del manipulador de manera secuencial, se habla de un control por estados, estos se analizarán más adelante con el sistema de grafos de estado.

1.11 ELECTRONICA

1.11.1 Componentes electrónicos

1.11.2 Electrónica digital:

1.11.3 Demultiplexores

Los demultiplexores actúan como distribuidor de datos y es un circuito lógico combinatorio con una línea de entrada (G), un cierto número de líneas de selección (N) y varias vías de salida (M), de acuerdo con el código aplicado a las líneas de selección, transfiere el dato presente de la entrada a una de las salidas.
Parte del circuito electrónico tendrá dos demultiplexores encargados de seleccionar los datos que vienen del microcontrolador y dirigirlos hacia los drive's de los motores.

Se selecciono el demultiplexor 74LS138 por ser un integrado muy popular de fácil adquisición y económico además cumple con las expectativas que se requiere para que funcione correctamente el circuito electrónico. A continuación se nombran sus principales características: (ver *Figura 34. DeMultiplexor 74LS138N)*

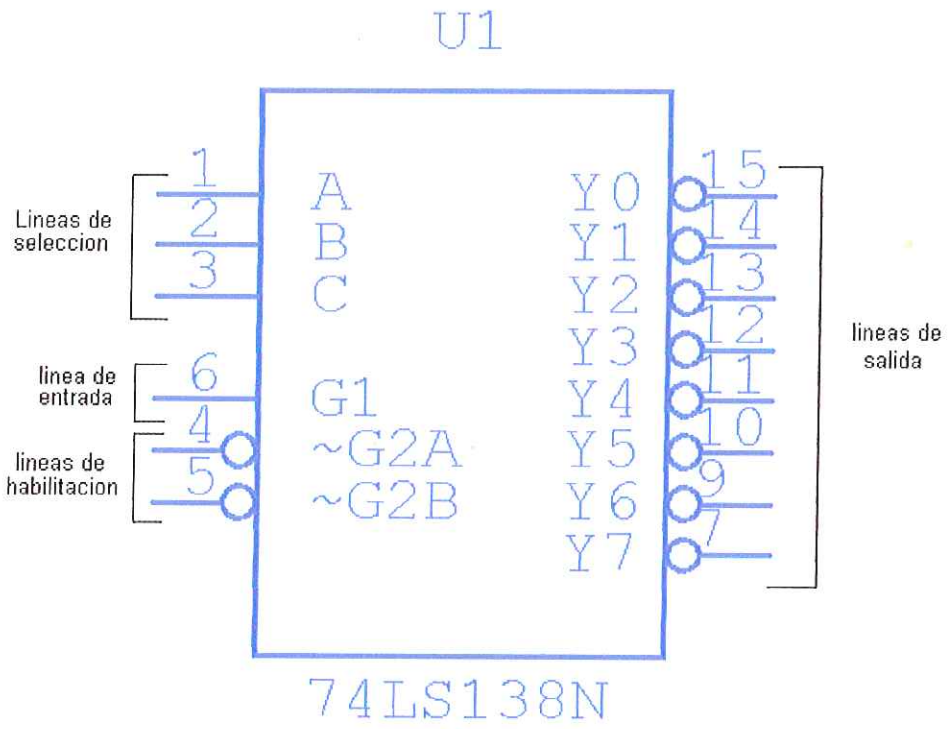


Figura 37. DeMultiplexor 74LS138N

- posee 8 líneas de salida.
- tres líneas de selección activas en bajo.
- Una línea de entrada.
-

Para mas detalles ver el datasheet, anexo A5.

1.11.4 Microcontrolador

Para este manipulador escogimos un microcontrolador de 8 bits de la familia motorota MC68HC908GP32 ya que esta familia de microcontroladores ha sido perfeccionada para aplicaciones de control en lugar de procesamiento de datos como lo son las familias de microchip entre otros, además en un microcontrolador basada su construcción en tecnología CISC (complex instruction set computer), a

diferencia de microchip que basa su construcción con tecnología RISC(Reduced instruction set computer), esto permite un conjunto de instrucciones mas extendido para así poder facilitar la programación, por ende es una de las familias mas utilizadas en el mundo.

1.11.4.1 Principales características de microcontrolador MC68HC908GP32:

- 4 puertos de E/S, de 8 bits (Puertos A,B,C y D)
- 32KB de memoria Flash
- 512B de RAM
- Unidad de comunicaciones serie asíncronas (SCI)
- Unidad de comunicaciones serie síncrona (SPI)
- Dos temporizadores de 16 bits
- 8 canales A/D de 8 bits
- Frecuencia interna del bus de 8-MHz
- Arquitectura de alto rendimiento M68HC08 optimizada para compiladores C.

Para mas información ver anexo A6

1.11.5 Electrónica de potencia:

1.11.5.1 Puente H

Para la aplicaciones de potencia que los motores necesitan se considero la estructura típica de un puente H, pues es una configuración especial para el control del sentido de giro de un motor en cualquier dirección y sin usar relevos mecánicos. Esta es una configuración electrónica que permite manejar un motor DC para invertir la polaridad automáticamente.

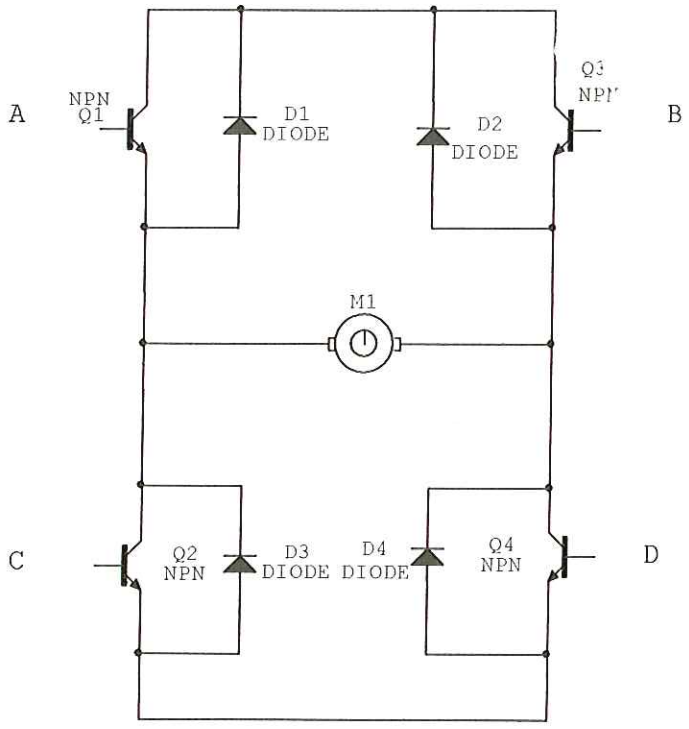


Figura 38. PUENTE H

En la figura se ilustra la configuración típica del puente, Cuando se activan las señales de control que energizan los transistores por ejemplo la señal A y D los transistores Q1 y Q4 se energizan permitiendo que fluya una corriente a través del motor y así gire en un sentido, al desactivar las señales A y D y activar B y C los transistores Q1 y Q4 se desenergizan mientras que los transistores Q2 y Q3 se energizan permitiendo que fluya corriente en sentido contrario y por tanto el motor cambie de giro. Se procuraba construir un puente H especial con especificaciones de corriente superiores a 3A ya que es la máxima corriente que va ha manejar el motor del antebrazo. Obviamente la construcción de este driver resulta de un costo muy elevado debido a que vamos manejar 5 motores para realizar los movimientos del manipulador pues se necesitaría una cierta cantidad de transistores especiales para su construcción, por tal razón se recurrió a driver's especiales en el mercado que cumplieran las exigencias que se necesitan.

Se consideraron dos driver's especiales el LMD18200 el cual se descarto por ser excesivamente costoso. Finalmente se opto por el TA7291P de TOSHIBA.

Este es un driver con configuración de puente para manejar motores DC. Este tiene una entrada analógica de voltaje de referencia *Vref*, con la cual es posible controlar o referenciar la potencia entregada al motor. Entre las principales características tenemos:

- Cuatro modos disponibles (izquierda, derecha, apagado y freno).
- Salida de corriente de 2A.
- rango de operación: - Vcc= 4.5 – 20 V
  -Vs= 0 -20v.
  -Vref= 0-20v.
- protección contra sobrecargas.
- Disponibilidad de modo de espera (STOP).
- Histéresis para todas las salidas.
- Modo de conexión especial de varios driver's para aumentar su capacidad de corriente.

El circuito integrado incorpora un estado de freno (parada) que evita los picos de corriente que normalmente se presentan al momento de cambiar el sentido del motor a plena marcha.

INPUT		OUTPUT		MODE
IN1	IN2	OUT1	OUT2	
0	0	»	«	STOP
1	0	H	L	CW / CCW
0	1	L	H	CCW / CW
1	1	L	L	BRAKE

» alta impedancia

entradas activas en alto

Tabla 7. Tabla de verdad del driver TA7291P.

La siguiente tabla describe el modo de conexión del drive TA7291P.

pin	Símbolo	Descripción
7	Vcc	Voltaje de Alimentación – Terminal lógico
8	Vs	Voltaje de Alimentación al Motor
4	Vref	Voltaje de referencia para control de Motor
1	GND	tierra
5	IN 1	Entrada 1
6	IN 2	Entrada 2
2	OUT 1	Salida 1
10	OUT 2	Salida 2

Tabla 8. Tabla de conexiones de TA7291P

Los pines 3 y 9 no se conectan.

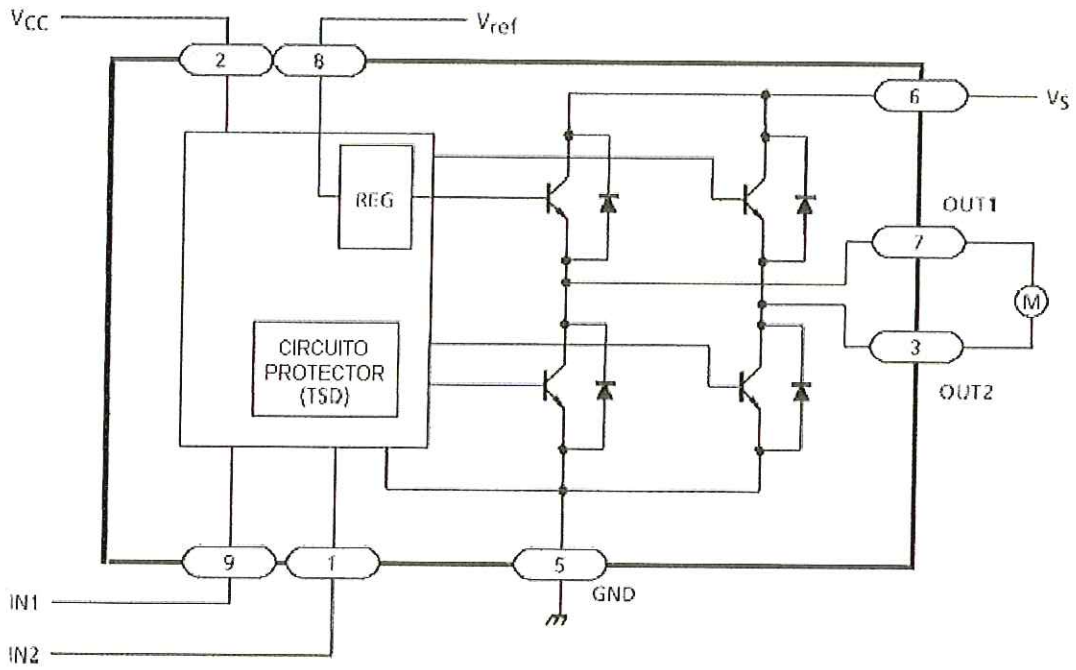


Figura 39. Diagrama interno del TA7291P

Para este diseño, voltaje de referencia será 5 voltios (provenientes de un microcontrolador).

la resolución es:

$$resolucion = \frac{5v}{2^8} = 19.5mV$$

Para más especificaciones ver anexo A4

1.12 DIAGRAMA DE BLOQUES DEL CIRCUITO ELECTRICO

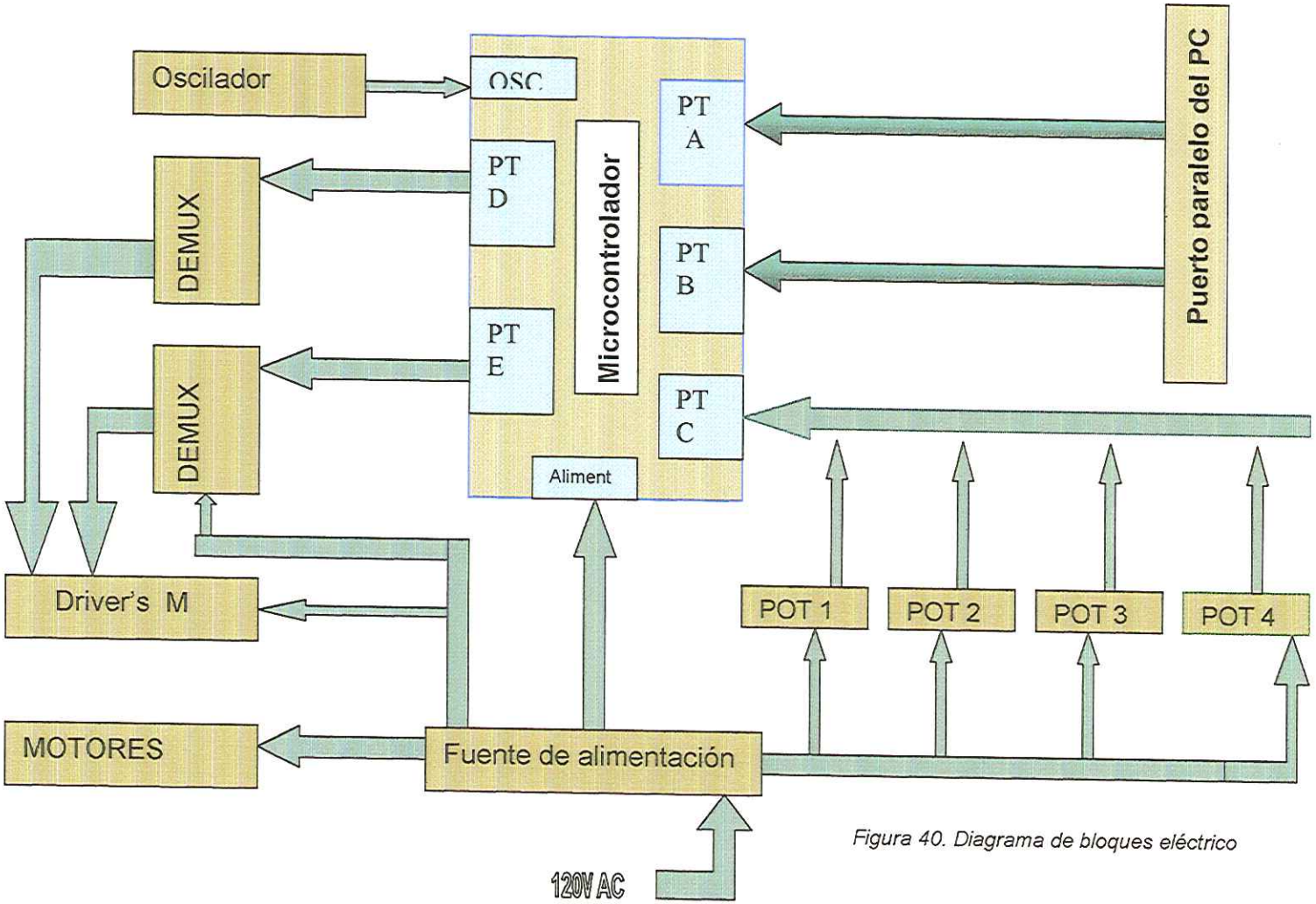


Figura 40. Diagrama de bloques eléctrico

1.13 FUENTE DE ALIMENTACION

Teniendo en cuenta las especificaciones de los motores y las practicas realizadas con ellos se obtuvo las corrientes máximas que estos consumen esto se hizo colocándole carga a los motores, se obtuvieron corrientes máximas de hasta 4 amperios para el motor que moverá el brazo.

Debido que este es el que mas consume corriente se toma como soporte para construir la fuente de potencia, teniendo en cuenta esta corriente se busco en el mercado un puente rectificador cuya capacidad de corriente sea superior a la máxima corriente que consume la carga, se opto por un factor de seguridad relativamente alto ya que cuando todo el circuito este funcionando consumirá mas corriente, el puente rectificador escogido es el FB25A el cual tiene una capacidad de corriente de 25A, se decidió por este ya que nos da un factor de seguridad elevado.

Para estos factores se tuvieron en cuenta los siguientes cálculos: voltaje que entrega el transformador (12v-0-12v)rms a 10A, voltaje pico de entrada Vm, voltaje de rizo Vr(rms), el rizo(r), voltaje DC de la Rectificación.

Para esta fuente se tiene un voltaje Vm=24v (rms) que la entrega el transformador

$$Vm = 24V(rms) = 24 * \sqrt{2} = 33.94Vp$$

El componente Ac de la señal de salida se obtiene como se indica a continuación.

$$Vr(rms) = 0.385Vm \Rightarrow Vr(rms) = 0.385 * 33.94V = 13.06V$$

El nivel de voltaje dc de la rectificación se obtiene:

$$Vdc = 0.636Vm = 0.636 * 33.94 = 21.6V$$

Finalmente se hallo el porcentaje de rizo con las siguientes cálculos:

$$r = \frac{Vr(rms)}{Vdc} * 100\% = \frac{0.308Vm}{0.636Vm} = *100\%$$

$$r = \frac{0.308 * 33.94}{0.636 * 33.94} * 100\% = 48.42\%$$

Adicionalmente se colocaron dos filtros en paralelo a la salida del puente rectificador esto con el fin de minimizar el voltaje pulsante a la salida del rectificador.

Finalmente colocaron los reguladores de voltaje específicos que nos entreguen los niveles de voltaje requeridos para nuestro circuito general los circuitos seleccionados son el LM317T y LM7805. Los valores de las resistencias se calcularon de la siguiente manera para obtener un voltaje de salida de 20v.
Según la formula:

$$Vo = Vref\left(1 + \frac{R2}{R1}\right) + Iadj * R2$$

De electrónica teoría de circuitos. Pág. 825

Vref=1.25 e Iadj=100uA.

Se selecciono una resistencia de 330ohm para la resistencia fija, despejando matemáticamente la resistencia variable y teniendo en cuenta el voltaje a la salida

20v se calculo la resistencia del potenciómetro como se muestra en los siguientes calculos:

$$20 = 1.25\left(1 + \frac{R2}{R1}\right) + 100\mu A * R2$$

$$20 = 1.25\left(1 + \frac{R2}{330\Omega}\right) + 100\mu A * R2$$

$$20 = 1.25 + \frac{1.25R2}{330\Omega} + 100\mu A * R2$$

$$18.75 = R2\left(\frac{1.25}{330\Omega} + 100\mu A\right)$$

$$\frac{18.75}{\left(\frac{1.25}{330\Omega} + 100\mu A\right)} = R2$$

$$R2 = 4.8K\Omega$$

De esta manera se obtienen los 20v requeridos los circuitos de potencia.

En las siguientes figuras se observa el circuito esquemático de la fuente de potencia y su respectivo circuito impreso construido.

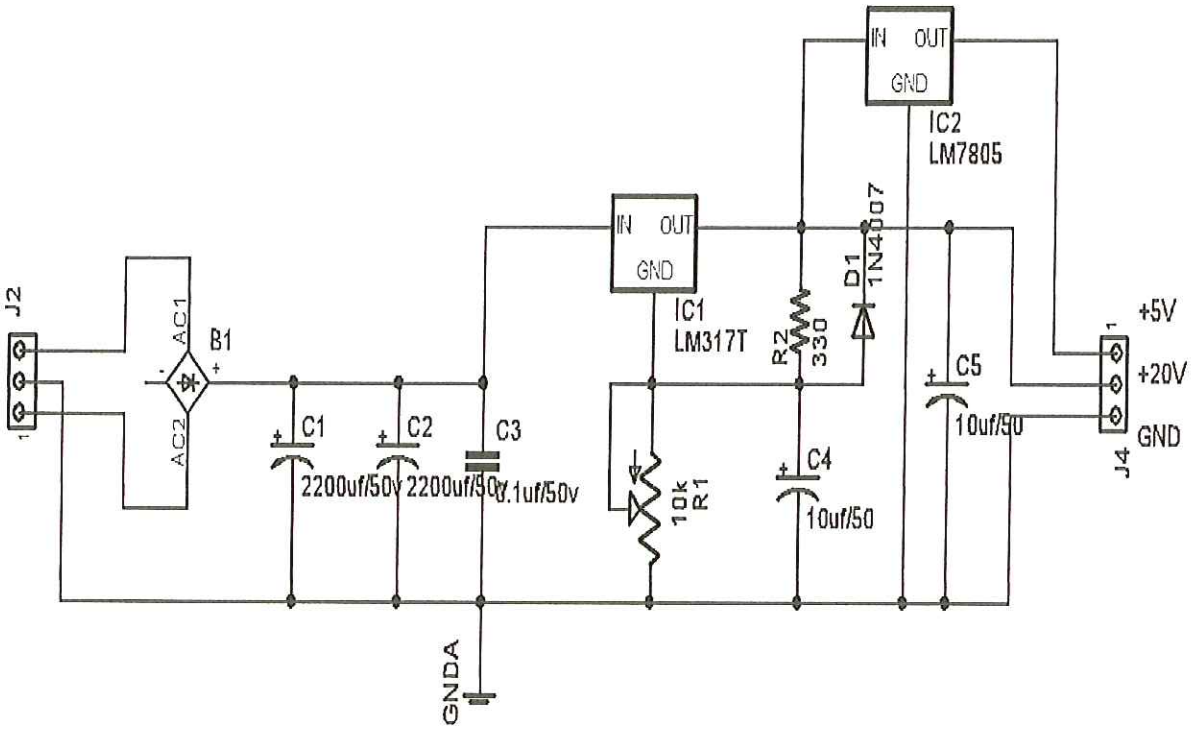


Figura 41. Esquematico fuente de alimentacion

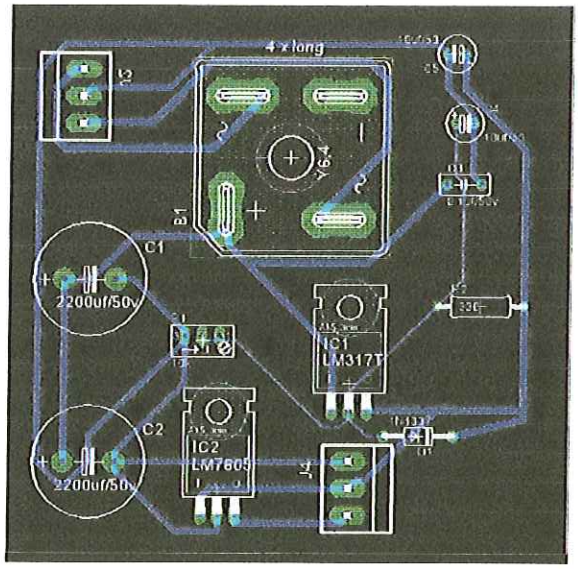


Figura 42. IMPRESO DE LA FUENTE

1.13.1 Circuitos electrónicos

Para el control de los motores se realizaron tres driver's uno que será encargado de manejar el motor que consume mayor corriente, para este se realizo una

configuración especial para aumentar la capacidad de corriente y así poder manejar este motor. se requeria manejar una corriente maxima de 4A, se opto por realizar una conexión con 5 drivers los cuales se dividen esta carga de la siguiente manera:

$$Carga\_por\_driver = \frac{Carga\_Total}{\#\_drivers}$$

$$Carga\_por\_driver = \frac{4Amp}{5} = 800mA$$

$$Vref = 20V \Rightarrow \max ima\_potencia\_al\_motor$$

$$Potencia\_driver = 800mA * 20V = 16W$$

$$Potencia\_total = 64W$$

A continuación se observa los circuitos realizados.

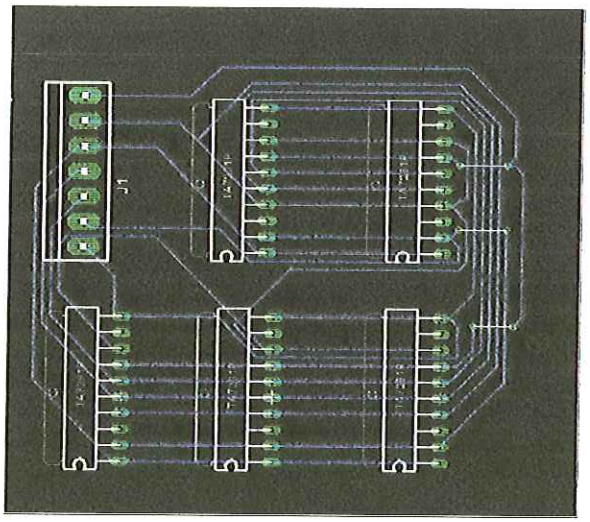
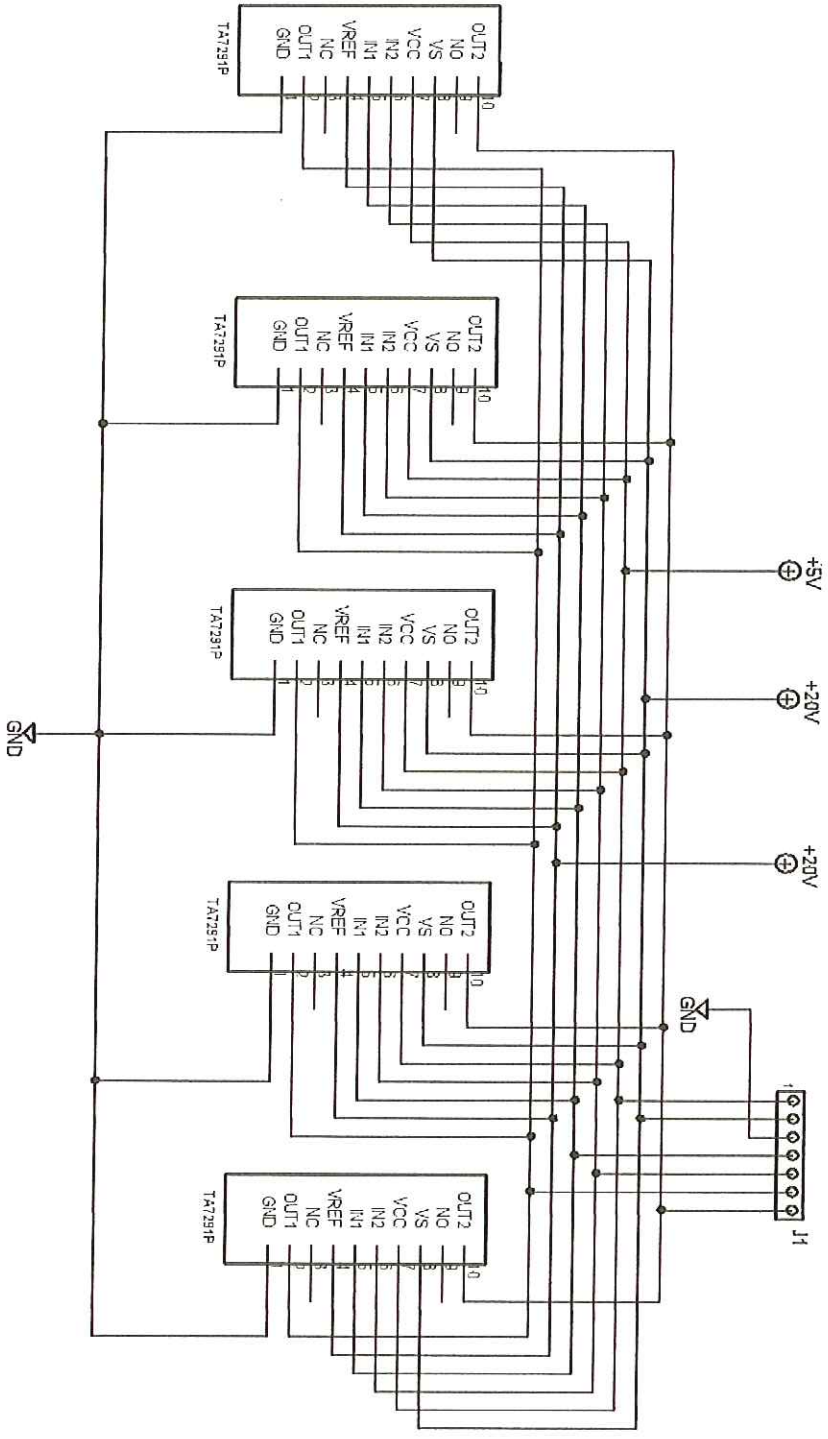


Figura 43. Circuito impreso Del motor 2

DEL AUTOR

Figura 44. CIRCUITO ESQUEMATICO DEL MOTOR 2



DEL AUTOR

Los otros dos driver's son destinados para controlar el motor de la base y el resto de motores que son el motor que mueve el antebrazo, y los otros dos destinados para realizar los movimientos del gripper. Para este se realizo los circuitos independientes debido al poco espacio que se tiene en el manipulador ya que toda la circuiteria estará montada dentro del manipulador. Los calculos obtenidos para manejar cada motor con su respectiva especificación de corriente se tienen:

Para el motor de la base.

$Corriente = 1.2Amp$

$Vref = 20V$ Para mayor torque.

$$Potencia = 1.2Amp * 20V = 24W$$

$$Potencia\_\max = 2A * 20V = 40W$$

Motor antebrazo.

$Corriente = 800mA$

$Vref = 20V$ Mayor torque.

$$Potencia = 800mAmp * 20V = 16W$$

$$Potencia\_\max = 2A * 20V = 40W$$

Motor gripper.

$Corriente = 600mA$

$Vref = 20V$ Mayor torque.

$$Potencia = 600mAmp * 20V = 12W$$

$$Potencia\_\max = 2A * 20V = 40W$$

A continuacion se observa los circuitos esquemáticos con su respectivo impreso.

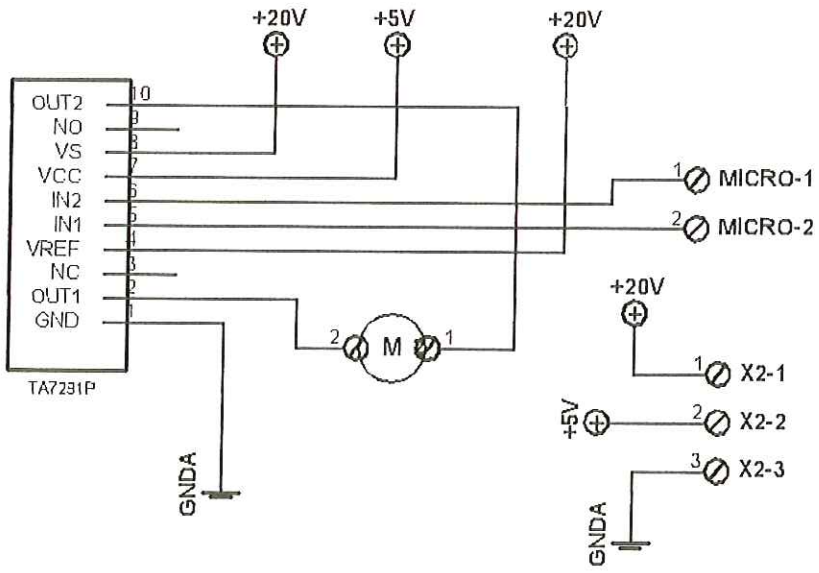


Figura 45. DIAGRAMA ESQUEMATICO PARA CONTROL DE LA BASE

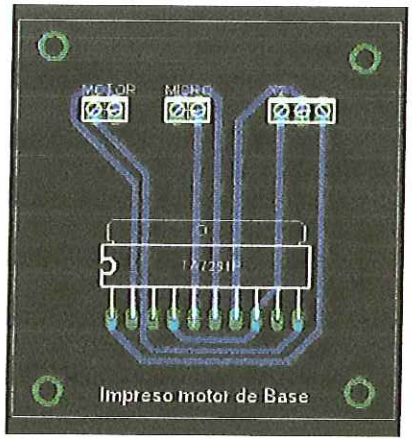


Figura 46. CIRCUITO IMPRESO DE LA BASE

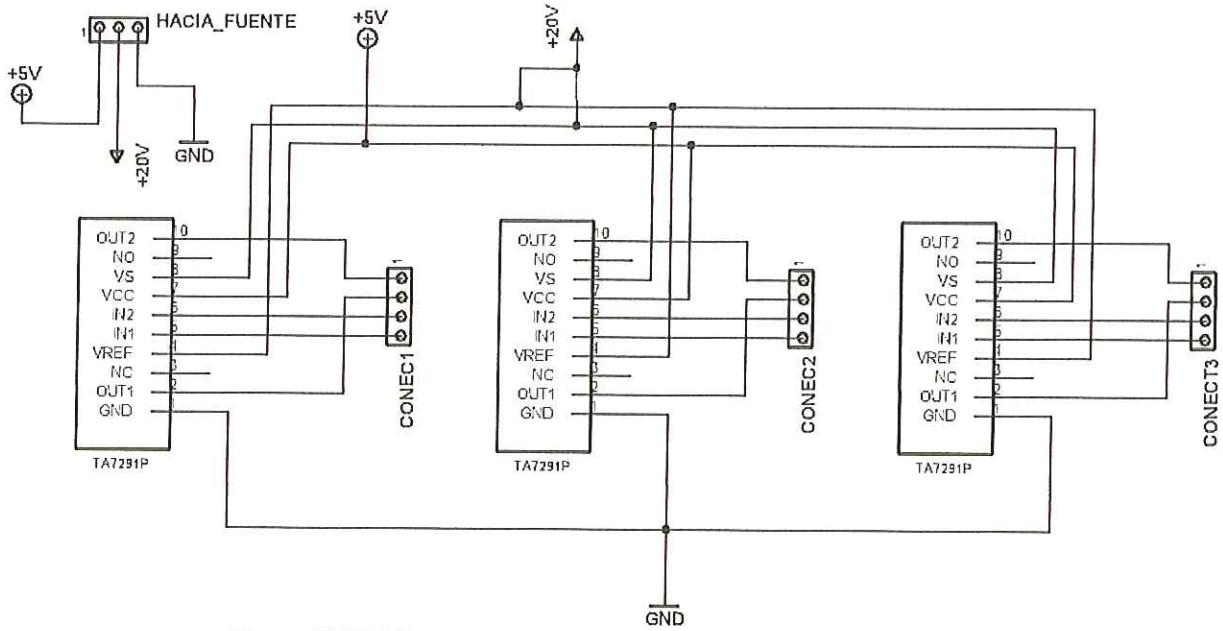


Figura 47. ESQUEMATICO MOTORES ANTEBRAZO Y GRIPPER

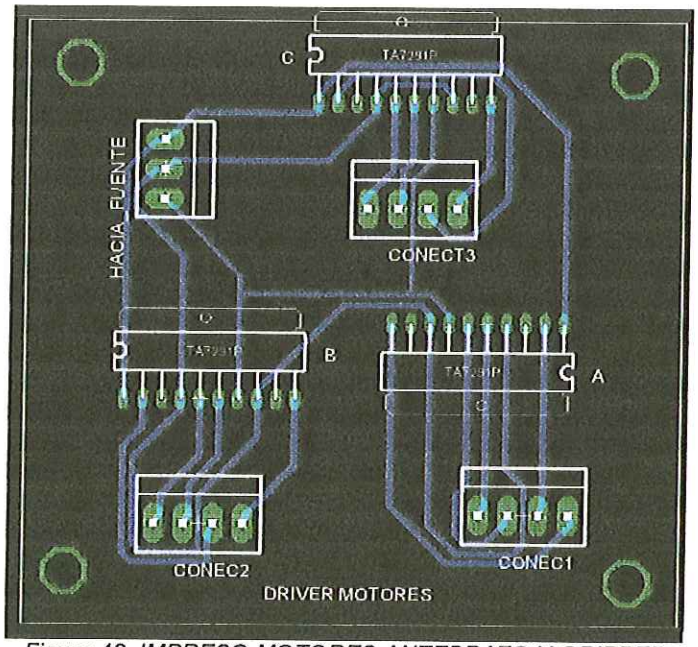


Figura 48. IMPRESO MOTORES ANTEBRAZO Y GRIPPER

Finalmente se construyo el diagrama que controlara los driver's este esta compuesto por un microcontrolador MC68HC908GP32 encargado de tomar dediciones y ejecutar la acción necesaria para que este funcione correctamente. Los demultiplexores son encargados de seleccionar el dato que es enviado desde el microcontrolador para enviar este hacia algún driver de los motores los cuales son encargados de activarlos, Para finalmente realizar el movimiento deseado.
A continuación se observa las figura el circuito controlador del manipulador.

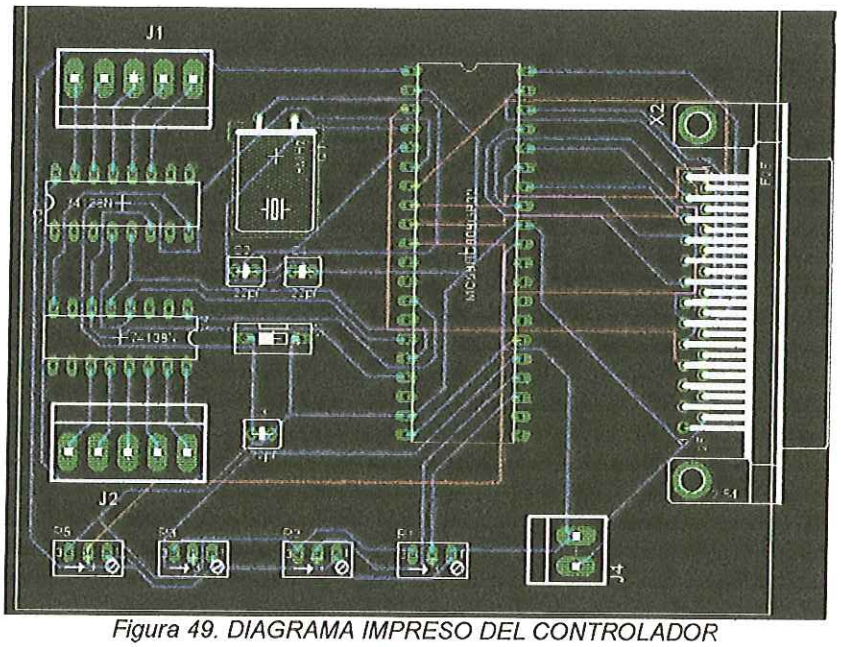
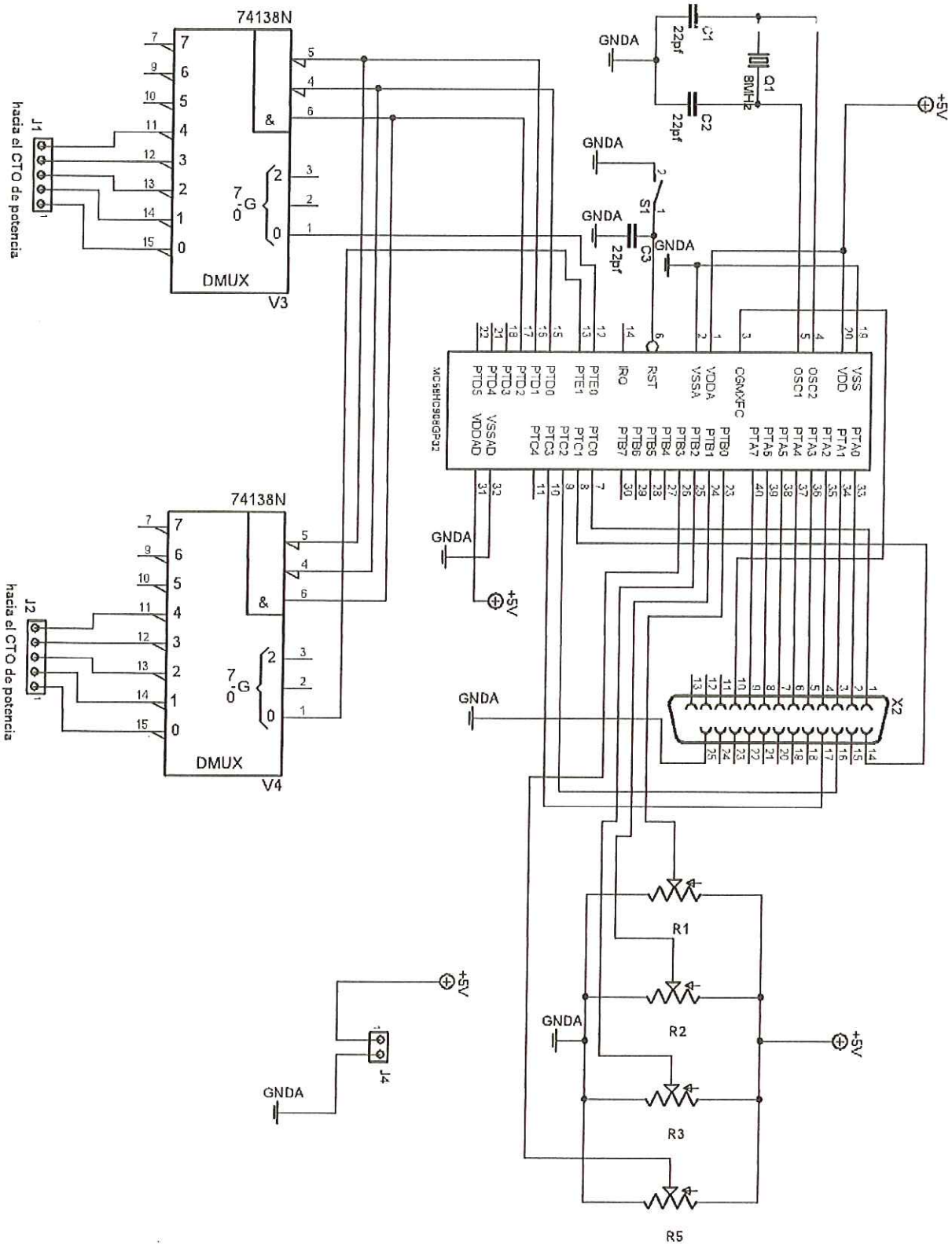


Figura 49. DIAGRAMA IMPRESO DEL CONTROLADOR

Figura 50. CIRCUITO ESQUEMATICO DEL CONTROLADOR



1.14 DIAGRAMA DE FLUJO DEL MICROCONTROLADOR

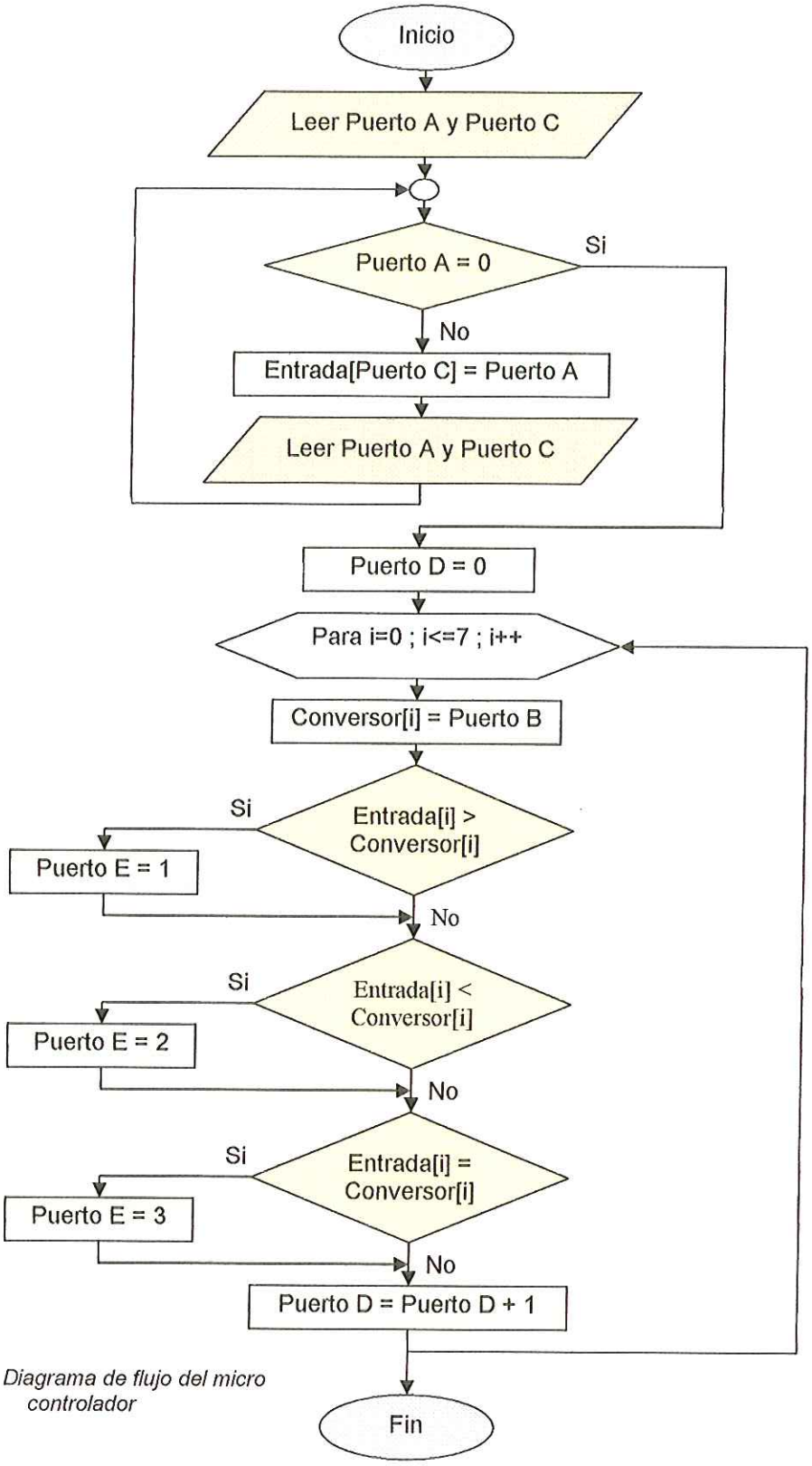


Figura 51. Diagrama de flujo del micro controlador

PUERTOS DEL MICROCONTROLADOR	
NOMBRE	**DESCRIPCIÓN**
Puerto A	Puerto de 8 entradas. Recibe las salidas enviadas por el puerto paralelo correspondientes al ángulo deseado en binario.
Puerto B	Puerto de 8 entradas. Conversor A/D. Recibe los niveles de voltaje de 0 a 5V de los 4 potenciómetros lineales de las articulaciones.
Puerto C	Puerto de 4 entradas. Recibe el código binario correspondiente a cual motor se va a seleccionar.
Puerto D	Puerto de 3 salidas. Se encarga de enviar una señal de *clock* al demultiplexor para enviar una señal de activación a los motores en forma secuencial.
Puerto E	Puerto de 2 salidas. Se encarga de proveer la señal al demultiplexor para que la aplique a cada motor. La señal produce 4 estados: mover izquierda, mover derecha, freno y nulo, los cuales se aprecian en detalle en la tabla anterior.

Tabla ¿? Puertos del microcontrolador

VARIABLES PRINCIPALES	
NOMBRE	**DESCRIPCIÓN**
Entrada[x]	Vector que acumula los datos enviados por el puerto paralelo. La posición de almacenamiento 'x' del vector corresponde al número de motor a activar, el cual es proveniente del Puerto C.
Conversor[x]	Vector que acumula el dato leído por el Puerto B.

Tabla ¿? Variables principales en el microcontrolador

1.14.1 Descripción del diagrama de flujo del microcontrolador

El programa comienza leyendo el estado de los Puertos A y C, correspondientes al ángulo deseado y el motor a activar respectivamente. Entonces se pregunta si en el Puerto A no hay nada (¿Puerto A = 0?), si no lo es, se acumulan los datos de cada motor en el vector 'entrada' y se vuelve a leer el Puerto A y C para continuar con el ciclo hasta que el estado del Puerto A sea cero.

Una vez acumulados los datos de entrada se inicia con cero el Puerto D, es decir, se resetea, para entonces entrar al bucle Para (*For*), el cual hace 8 ciclos incrementando una variable 'i' de uno en uno (*i*++), en un intervalo de 0 a 7 (*i=0 ; i<=7*). Dentro del ciclo, se inicia acumulando en el vector 'conversor' el dato existente en el puerto B (conversor análogo/digital).

Entonces se procesan 3 condiciones, cada una corresponde a un estado de activación para los motores. La primera compara si 'entrada[i]' es mayor a 'conversor[i]', si es correcto, el Puerto E manda un 1 (*10*) en binario correspondiente al estado: mover a la derecha, si no es correcto, pasa a la siguiente condición. La segunda compara si 'entrada[i]' es menor a 'conversor[i]', si es correcto, el Puerto E manda un 2 (*01*) en binario correspondiente al estado: mover a la izquierda, si no es correcto, pasa a la siguiente condición. La tercera compara si 'entrada[i]' es igual a 'conversor[i]', si es correcto, el Puerto E manda un 3 (*11*) en binario correspondiente al estado: detener o frenar.

Después de pasar por los tres ciclos, al Puerto D de le asigna su valor anterior más 1, para garantizar de que la activación se realiza de manera secuencial en el orden que tienen las articulaciones, de la base al antebrazo o del antebrazo a la base, de cualquier modo el sistema parecerá moverse en conjunto, por que las ordenes del demultiplexor se hacen en el orden de los milisegundos.

1.14.2 Programa en assembler del microcontrolador

```
$Include 'gpgtregs.inc'

RAMStart      EQU  $0040
RomStart      EQU  $8000
VectorStart   EQU  $FFDC

              org      RamStart

entrada         ds       8
conversor       ds       8

          org      RomStart

principal:
           rsp
           mov  #$70,ADCLK
           clra
           clrx
           mov    #$31,CONFIG1
           mov    #$00,DDRA
           mov    #$00,DDRB
           mov    #$10,DDRC
           mov    #$03,DDRE
           mov    #$FF,DDRD
inicio:
           ldx    PTC
           cbeqx #$00,condicion
           nop
           nop

motor:
      aix        #-1
      lda        PTA
      sta        entrada,X                    ;salta almacena la dirección
del registro para nuevo dato

condicion:
           mov        #$00,PTD
           ldx        #$00


loop:
        cpx      #$08
        beq      final
        stx      ADSCR
final_adc:
          brset 7,ADSCR,compara
          bra    final_adc

compara:
        lda      ADR
        sta      conversor,x
        lda      entrada,x
```

```
        cmp     conversor,x                     ;'valor entrada'-'valor del
conversor'

        blo     menor
        bhi     mayor
        bra     igual
mayor:
      mov       #$01,PTE
      bra       fin
menor:
      mov       #$02,PTE
      bra       fin
igual:
      mov       #$03,PTE
      bset      4,PTC
      bra       fin
fin:
    aix #1
    lda #$FF
retardo:
        dbnza   retardo
    inc PTD                                 ;sugerencia: poner retardo antes
que D cambie
    bra loop

final:
          bra   inicio

dummy_isr:
          rti
swi_isr:                        ; do nothing
        rti

     org     VectorStart
        dw      dummy_isr       ; Time Base Vector
        dw      dummy_isr       ; ADC Conversion Complete
        dw      dummy_isr       ; Keyboard Vector
        dw      dummy_isr       ; SCI Transmit Vector
        dw      dummy_isr       ; SCI Receive Vector
        dw      dummy_isr       ; SCI Error Vector
        dw      dummy_isr       ; SPI Transmit Vector
        dw      dummy_isr       ; SPI Receive Vector
        dw      dummy_isr       ; TIM2 Overflow Vector
        dw      dummy_isr       ; TIM2 Channel 1 Vector
        dw      dummy_isr       ; TIM2 Channel 0 Vector
        dw      dummy_isr       ; TIM1 Overflow Vector
        dw      dummy_isr       ; TIM1 Channel 1 Vector
        dw      dummy_isr       ; TIM1 Channel 0 Vector
        dw      dummy_isr       ; PLL Vector
        dw      dummy_isr       ; ~IRQ Vector
        dw      swi_isr         ; SWI Vector
        dw      principal           ; Reset Vector
```

2. CONSTRUCCIÓN Y MONTAJE

2.1 CONSTRUCCIÓN DE PIEZAS

2.1.1 Base

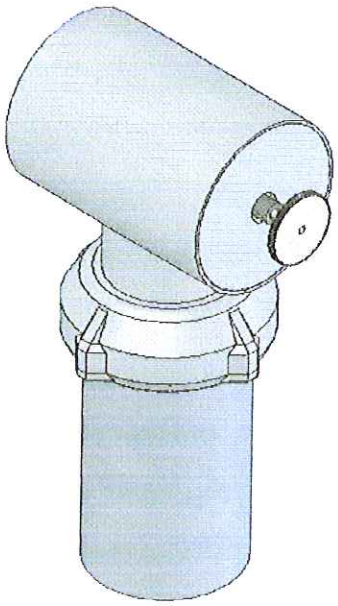


Figura 52. BASE 01

La base se compone de tres partes. La primera parte corresponde al *tubo base*. El tubo base es el que da la altura en la que va suspendido el brazo y antebrazo del manipulador, este va soldado a un disco el cual se atornilla a una mesa metálica.

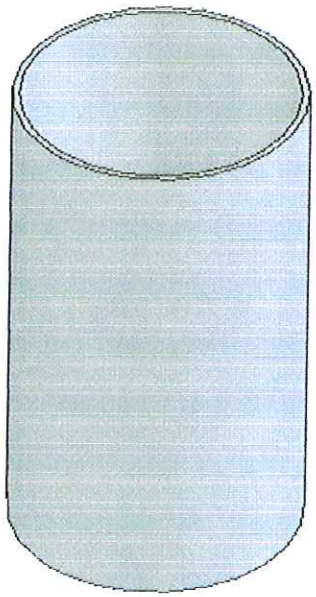


Figura 53. BASE 02

La segunda parte corresponde a la *estructura rotatoria* (implementación de mecanismo de rotación para antenas de radiofrecuencia), va atornillada a al tubo base mencionado anteriormente, esta hace posible la rotación en el eje z de todo el brazo mecánico, en el se encuentran el tren de engranajes correspondiente al motor y al potenciómetro de posicionamiento y un sistema de rodamiento por balines de 8 mm, a esta estructura se le hicieron 4 orificios (para tornillos de hierro fresado) para atornillar con la estructura superior y un orificio de 1 pulgada para pasar cableado a través de toda la estructura. Para adaptar el motor cd en la estructura rotatoria se utilizó una lámina en "L" y se ajustó aquí, después se atornilló esta pequeña estructura de tal forma que el motor hiciera contacto con el primer piñón del tren de engranajes. A este motor se le acopló un piñón de bronce SAE-65 de 12 dientes (diseñado en freza), el bronce es utilizado para piñones especialmente por su resistencia al desgaste por fricción y a su facilidad para maquinar. El potenciómetro que detecta la posición angular de este articulación de adapta al piñón final (con un prisionero roscado) de un tren de engranajes, compuesto de dos piñones de pasta, el primero de estos va engranado con el piñón principal de dientes internos de la estructura rotatoria, este tren de engranajes está calculado para que el potenciómetro de una vuelta en la misma proporción que da la vuelta toda la estructura.

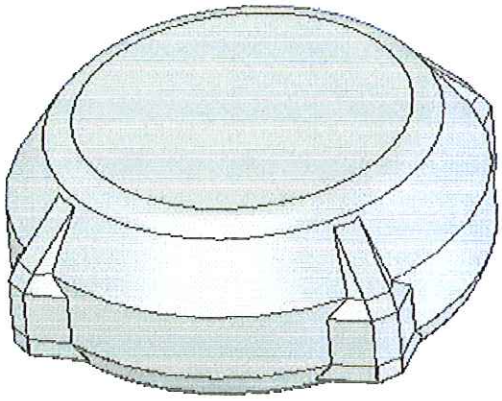


Figura 54. BASE 03

La tercera parte es el *tubo en "T"* que queda en la parte superior de toda la base va atornillada a la estructura rotatoria. Esta se compone de dos tubos de acero inoxidable los cuales se interceptan para formar una "T".

El tubo pequeño, de 10.4mm de diámetro y 2mm de espesor, el cual coincide con el diámetro superior de la estructura rotatoria va de forma vertical para ser atornillado a la base; para poder hacerlo se soldó (con soldadura de argón) una placa de 3mm de espesor al tubo e internamente se hicieron 5 orificios de la misma forma que la estructura rotatoria mencionada en el párrafo anterior, en los 4 orificios distribuidos radial y uniformemente se soldaron (con soldadura de argón) 4 tuercas coincidentes con estos para que los tornillos de hierro tipo frezado ajusten la estructura rotatoria y la placa del tupo "T".

Por un lado del tubo pequeño se soldó la placa con los 5 orificios y las 4 tuercas, por el otro lado se hace la forma del tubo que va incrustado de forma perpendicular para formar un tubo "T", este corte se realizó primero con plasma y después con *Motor-Tool* y lima redonda para obtener como resultado un corte uniforme. Finalmente se cortó 25cm de tubo de acero inoxidable de 5 pulgadas de diámetro y 2mm de espesor (tubo horizontal de la "T"), el cual encaja en el corte hecho en el tubo pequeño.

Estos dos tubos finalmente se soldaron (con soldadura de acero inoxidable) a una distancia, en uno de los extremos, de 6mm. El tubo horizontal lleva también un orificio de 1 pulgada coincidente con el de la placa el tubo pequeño de la "T" y el de la estructura rotatoria, para cableado interno.

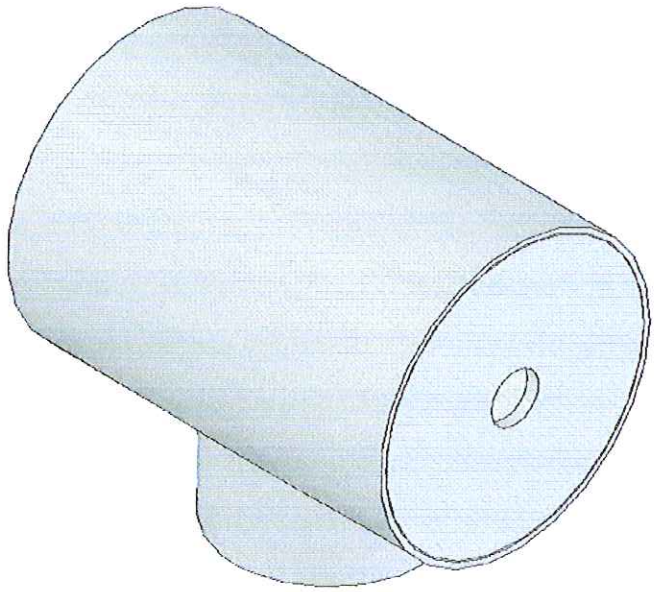


Figura 55. BASE 04

El tren de engranajes y el motor van ensamblados en una estructura compuesta por dos placas, tal estructura encaja dentro del tubo grande de la "T". La placa externa, en donde sale el eje principal de la articulación es más gruesa que la otra, pues aquí va sostenido por medio de un rodamiento grande el peso total del brazo, antebrazo y carga adicional.

El motor va atornillado a la otra placa, la cual es más delgada y queda adentro del tubo de 25cm de largo, entre las dos placas va el tren de engranajes sostenido por ejes y bujes, los cuales fueron hechos en torno a partir de una barra de bronce de ¾ de pulgada; el bronce antifricción es el más utilizado para la fabricación de bujes.

2.1.2 Brazo y Antebrazo

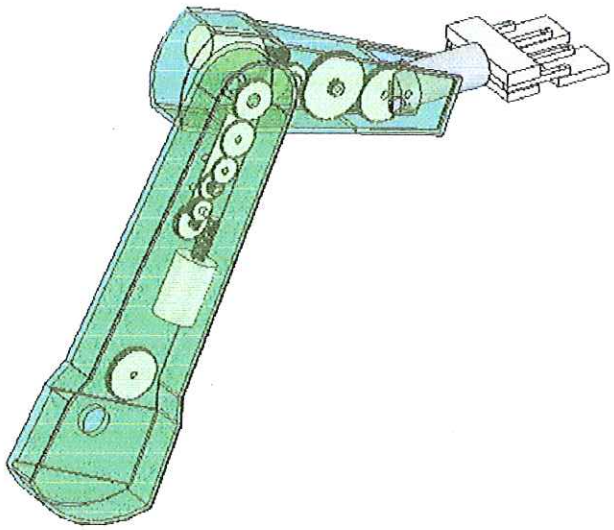


Figura 56. BRAZO Y ANTEBRAZO 01

Debido a que son piezas de difícil construcción a partir de formas básicas, se procedió a obtenerlas por medio del proceso de fundición. Para los moldes se hicieron dos modelos en madera, los cuales se basan en un diseño de cajón de espesor 5mm aproximadamente.

El material seleccionado para estas piezas fue el aluminio, por ser un metal liviano y de fácil maquinado, y el molde de fundición se hizo en arena. Un fenómeno que ocurre con este proceso es que la pieza final en aluminio se contrae y se disminuye el diámetro interno con respecto al modelo en madera. Es por esto que se realizaron los modelos en madera a partir de diseños en SolidWorks aumentados 3mm aproximadamente y a escala.

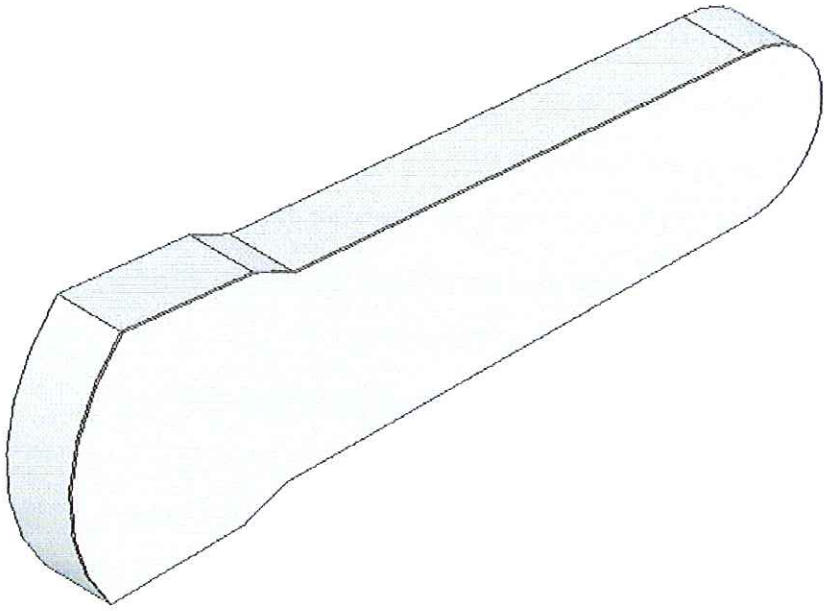


Figura 57. BRAZO Y ANTEBRAZO 02

El brazo, que es la pieza más grande de las dos, es la que en uno de sus extremos se ensambla con el eje final del tubo "T" y en el otro extremo con el antebrazo, éste último es más pequeño por que lleva menos engranajes y su largo sumado con el del *gripper* (agarrador) debe ser similar al largo del brazo.

El brazo lleva dos trenes de engranajes. El primero es un sistema de engranajes planetario de dos piñones, uno gira y otro no, el que gira va ensamblado con el eje final del tubo "T", este eje es de acero inoxidable de 1 pulgada y va por entre dos rodamientos grandes, uno que va en un extremo del tubo "T" y otro en el brazo, este aje de acero hace girar el piñón del planetario ya mencionado el cual engrana con otro de diámetro más grande, este último está anclado al brazo, es así que este empieza a girar alrededor del pequeño, llevándose consigo toda la estructura de aluminio, cada ves que haya movimiento en la articulación, como los planetas, la tierra gira alrededor del sol, en este caso la tierra sería el piñón grande y el sol sería el pequeño, el cual gira alrededor de su propio eje, el centro del universo, de ahí el concepto de *sistema planetario de engranajes*.

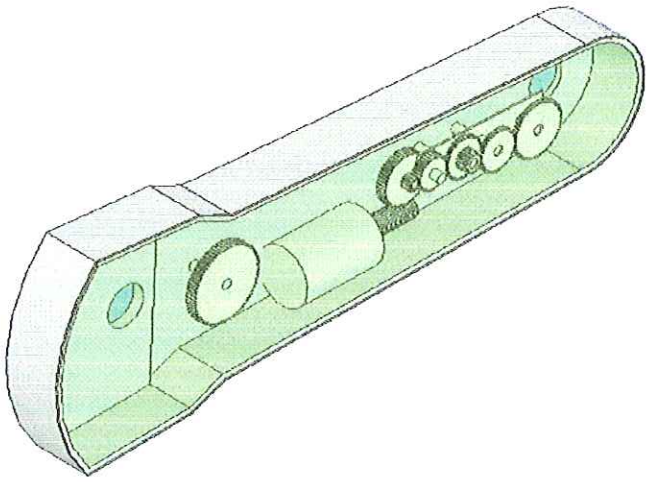


Figura 58. BRAZO Y ANTEBRAZO 03

El segundo es un tren de engranajes largo de varios piñones con un motor cc normal a 24V, este se encarga de mover y soportar el antebrazo, el gripper y la carga adicional. Todo va montado en una placa de acero inoxidable con espesor de $^1/_8$ de pulgada (ejes, bujes, piñones y motor), la cual se atornilla dentro de la estructura del brazo; el eje final lleva un rodamiento pequeño y su extremo va anclado a una pequeña placa de acero inoxidable la cual se atornilla en un extremo del antebrazo.

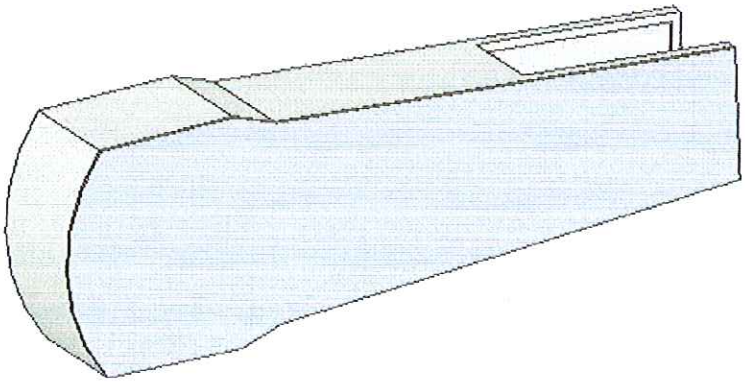


Figura 59. BRAZO Y ANTEBRAZO 04

El antebrazo es similar al diseño del brazo pero más pequeño y del mismo espesor, va ensamblado en uno de sus extremos con el brazo por medio de una

placa atornillada, como se explicaba anteriormente, y en el otro extremo va un tren de engranajes con un motor similar al del brazo, todo va, de igual forma, montado en una placa de acero; en el piñón final va atornillado el gripper, el cual se construyó a partir de una barra de aluminio de 2 pulgadas, en el extremo de este van dos pequeños engranes (cada uno lleva anclado una palanca para agarrar y soltar los objetos a manipular), en medio de estos va un tornillo sin fin conducido por un pequeño motor cd con reducción.

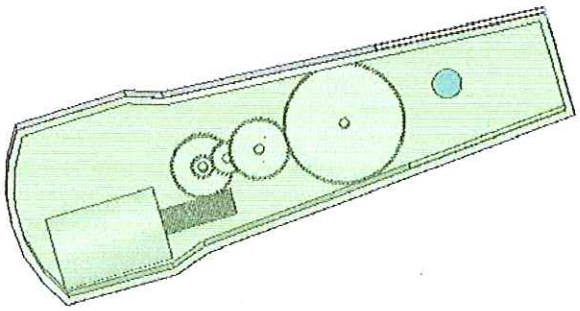


Figura 60. BRAZO Y ANTEBRAZO 05

Los dos motores cd usados tanto para el brazo como el antebrazo son similares. Ambos tienen anclado al eje un tornillo sin fin, esto tiene principalmente dos ventajas, la facilidad de montaje en la estructura tipo cajón de aluminio, y agrega mayor oposición a que el motor se deje llevar por la inercia del movimiento resultante. Ambos se manejan a 24V, pero el del brazo tiene mayor característica de fuerza con respecto a su imán interno.

El motor usado en el tubo "T" es grande y lleva internamente una reducción con engranajes de dientes inclinados, lo cual agrega fuerza a esta articulación, el motor tiene 180 RPM a 24V.

3. SOFTWARE

Se implemento un programa en el software matlab para el modelamiento matemático, control y seguimiento real del manipulador, el software utiliza algunas funciones de la librería HEMERO (Herramienta- Simulink para la Enseñanza de la Robótica), las cuales fueron desarrolladas por el autor del libro "manipuladores y robots móviles"

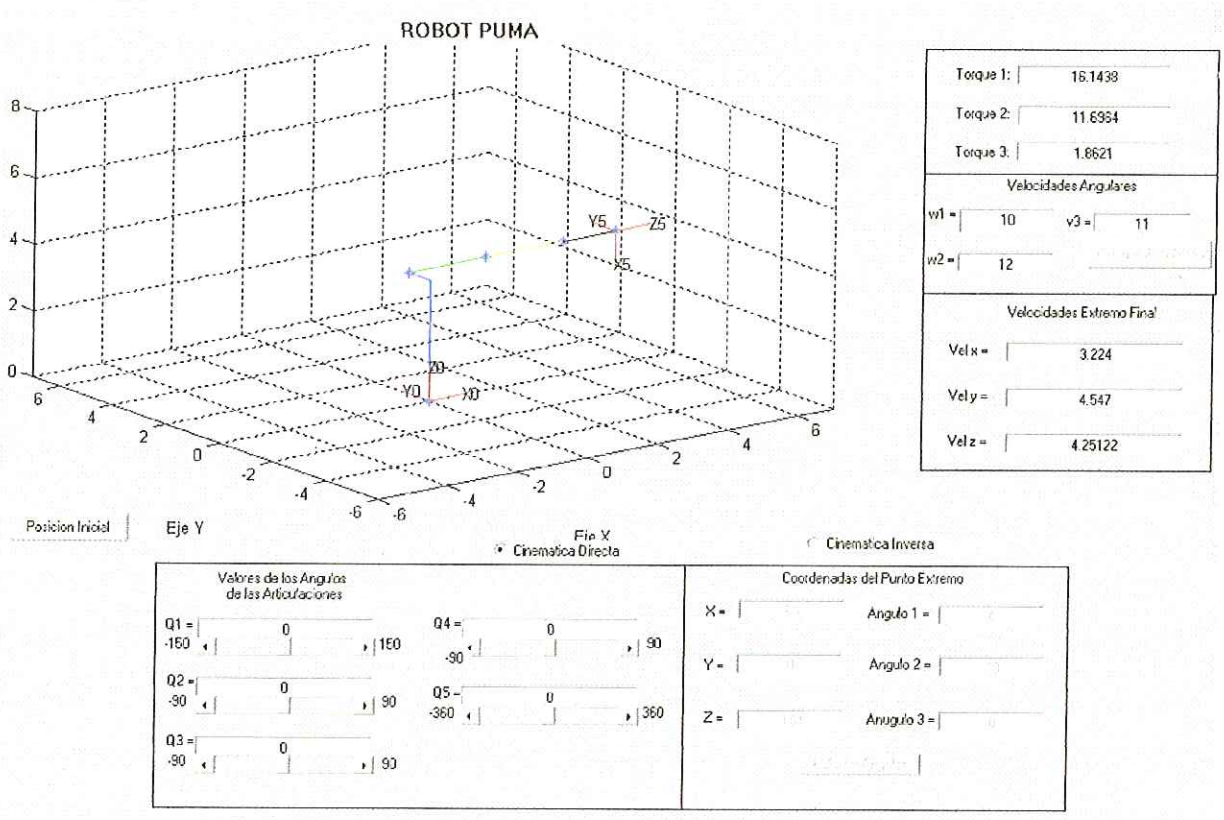


Figura 61. Visualización software 01

Teniendo en cuenta que el proyecto busca que el manipulador sea una herramienta didáctica para el usuario, este programa sirve como elemento de ayuda, logrando que los estudiantes puedan comprobar sus cálculos cinemáticos y dinámicos, con los resultados que genera el software.

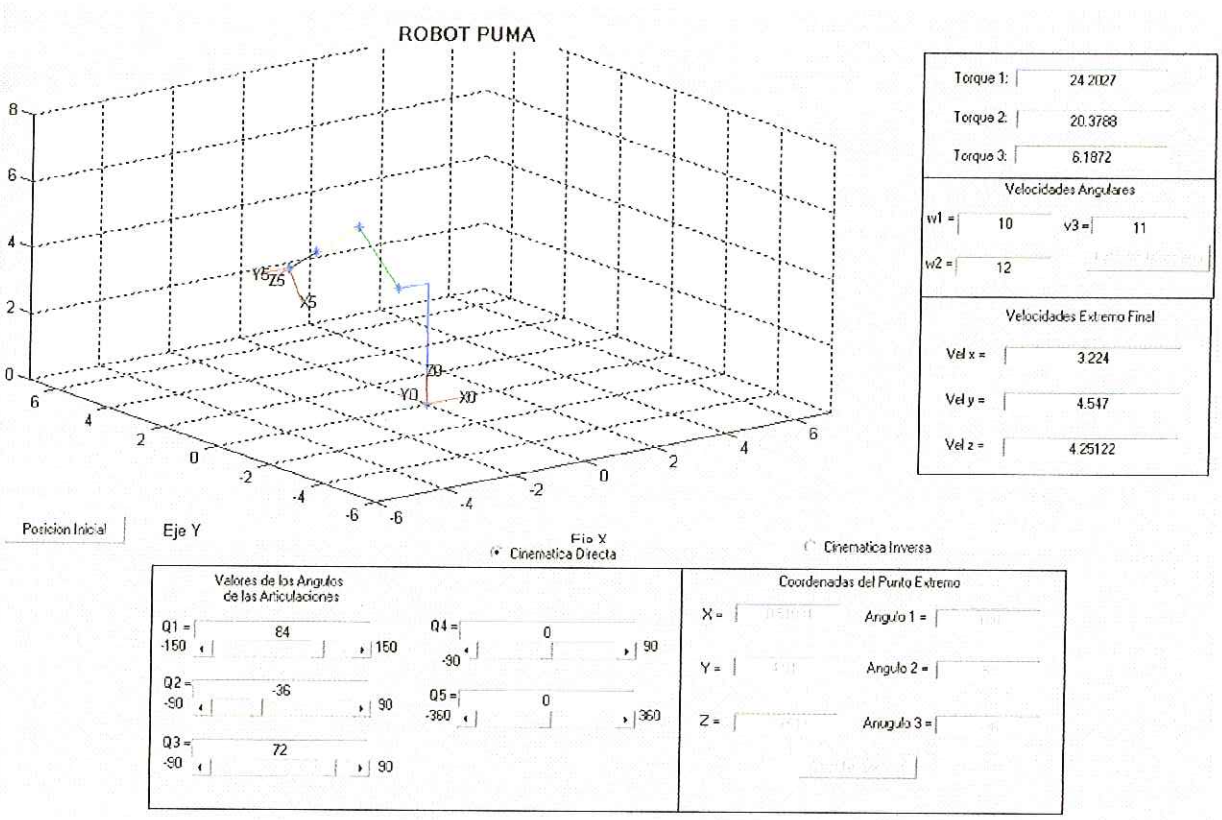


Figura 62. Visualización software 02

El usuario es quien le ingresa los ángulos de posicionamiento de cada articulación para que el manipulador logre ubicarse; o sino le ingresa un valor de coordenadas (X,Y,Z), para que él se localice directamente en un punto en el espacio. Esto se logra mediante el modelamiento Cinemático Directo e Inverso descrito en dos funciones en matlab que se incluyen dentro del programa principal.

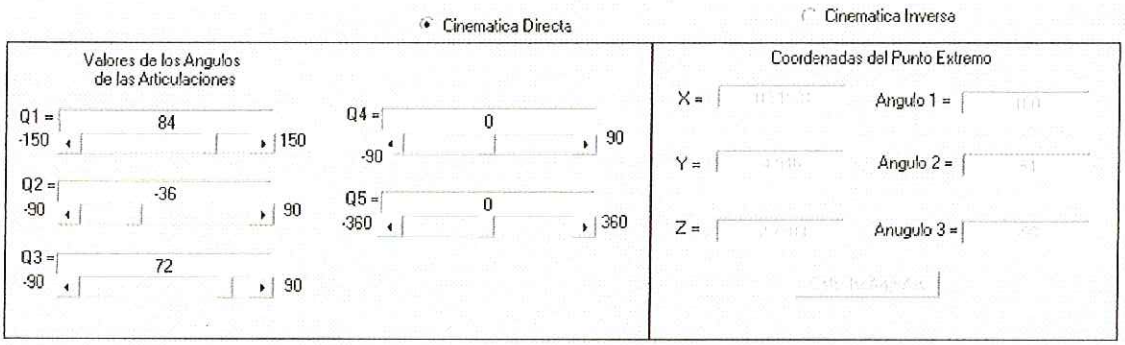


Figura 63. Visualización software cinemática

El programa también permite calcular las velocidades a las cuales el manipulador llega a su ubicación final. Se utilizó el cálculo Jacobiano descrito en una función en matlab para lograr estos datos.

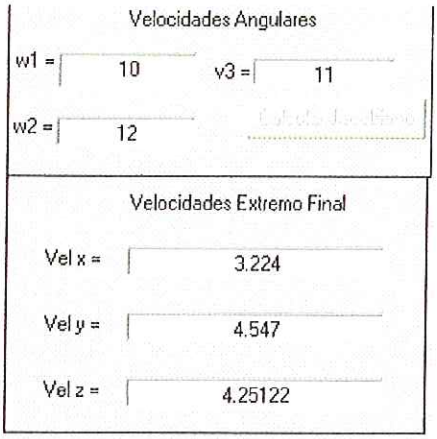


Figura 64. Visualización software velocidades

Finalmente podemos obtener el torque de cada movimiento que realiza el manipulador puma en sus tres rotaciones principales respectivamente. Esto se logró mediante las ecuaciones dinámicas que resultaron del Modelamiento Dinámico realizado previamente. Dentro del programa principal van llamando a estas ecuaciones descritas en una función para que realicen la operación.

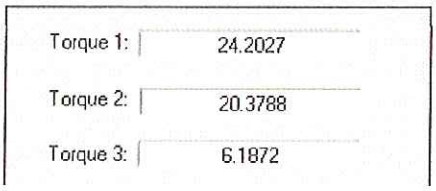


Figura 65. Visualización software torques

El programa esta configurado de tal manera que no solo nos indica los valores de la cinemática directa, inversa, torque y la velocidad del extremo final. También controla la posición real del manipulador, enviando los cambios de los ángulos por medio del puerto paralelo al microcontrolador.

CONCLUSIONES Y RECOMENDACIONES

La elaboración del manipulador nos permitió comprobar la integración que debe existir entre las diferentes áreas de estudio de la mecatronica para poder desarrollar un proyecto de este tipo.

Con el desarrollo de este proyecto de grado, se ha logrado hacer un aporte significativo al estado del arte de la robótica en esta región, al diseñar, construir y documentar un manipulador puma, que servirá de precedente para futuros trabajos e investigaciones.

Se comprobó que para realizar un buen control para estos manipuladores es necesario desarrollar cálculos matemáticos como de posicionamiento, cinéticos y cinemáticos ya que estos describen las trayectorias, los torques y velocidades que el manipulador necesita para efectuar sus movimientos de forma correcta.

La metodología de diseño Mecatrónico aplicada a este trabajo de grado, permitió la integración entre el diseño electrónico, mecánico, computacional y de control en un manipulador diseñado integralmente.

La realización de este proyecto de grado cumplió con expectativas esperadas ya que ratificó la aplicación de conocimientos adquiridos en nuestra carrera.

Es importante tener claro las etapas de diseño previas para la elaboración del proyecto ya que de ello dependen los mejores resultados, además nos permite ahorrar tiempo, reducir costos y obtener buenos beneficios.

Para la realización de este proyecto es esencial estar familiarizado con herramientas computacionales como matlab la cual nos facilito el desarrollo matemático del manipulador, un software CAD/CAE con el cual se procedió al desarrollo de su estructura y elección de materiales efectuando simulaciones de carga que este soportara, para finalmente hacer buena elección de los actuadores que mueve sin problemas al manipulador.

BIBLIOGRAFIA

- Aníbal Ollero Baturone. Robótica, Manipuladores y Robots Móviles. Editorial Alfaomega Marcombo.2001.
- Antonio Barrientos. Fundamentos de Robótica. Editorial McGraw Hill. 1997.
- G Ferrate, L Basañez. Robótica Industrial. Editorial Marcombo.
- H. Asada, J. Slotine. Robot Analysis and control. Editorial John Wiley. 1986.
- H. Wayne Beaty, James L. Kirtley, Jr. Manual del Motor Eléctrico. Editorial McGraw Hill Interamerica Editores, S.A. de C.V. 1998.
- Lung Wen Tsai. Robot Análisis. The Mechanics of Serial and Parallel Manipulators. Editorial John Wiley & Sons, Inc. 1999.
- Robert L. Mott, P.E. Diseño de Elementos de Maquinas 2ª Edición. Editorial Prentice Hall.1992.

Web sities:

- Asociación para la Práctica y Enseñanza de la Robótica,Spain. ; email: aperobot@bitmailer.net , "El robot en la historia de la Tecnología" [serie en Internet], disponible en: http://www.usuarios.bitmailer.com/aperobot/robothistoriatecno.htm
- Antonio J. Muñoz, " Introducción a la Robótica" [articulo en Internet], disponible en: http://www.isa.uma.es/personal/antonio/Robotica/Tema1%20-%20Introduccion.pdf
- Instituto Colombiano de Normas Técnicas "Icontec", disponible en: http://www.icontec.org.com
- www.omega.com
- www.jameco.com
- www.x-robotics.com

ANEXOS

A1 CODIGO FUENTE DEL PROGRAMA EN MATLAB

```
function varargout = puma02(varargin)
% PUMA02 M-file for puma02.fig
%      PUMA02, by itself, creates a new PUMA02 or raises the existing
%      singleton*.
%
%      H = PUMA02 returns the handle to a new PUMA02 or the handle to
%      the existing singleton*.
%
%      PUMA02('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PUMA02.M with the given input arguments.
%
%      PUMA02('Property','Value',...) creates a new PUMA02 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before puma02_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to puma02_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

% Edit the above text to modify the response to help puma02

% Last Modified by GUIDE v2.5 18-Apr-2005 17:40:11
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @puma02_OpeningFcn, ...
                   'gui_OutputFcn',  @puma02_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(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 puma02 is made visible.
function puma02_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 puma02 (see VARARGIN)

% Choose default command line output for puma02
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes puma02 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = puma02_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;

set(handles.radiobutton1,'value',1);
set(handles.radiobutton2,'value',0);
set(handles.radiobutton3,'value',0);


set(handles.edit6,'enable','off');
set(handles.edit7,'enable','off');
set(handles.edit8,'enable','off');
set(handles.pushbutton1,'enable','off');
set(handles.pushbutton3,'enable','off');
set(handles.pushbutton4,'enable','off');
set(handles.pushbutton5,'enable','off');
set(handles.edit9,'enable','off');
```

```
set(handles.edit10,'enable','off');
set(handles.edit11,'enable','off');

set(handles.edit12,'enable','off');
set(handles.edit13,'enable','off');
set(handles.edit14,'enable','off');
set(handles.edit15,'enable','off');
set(handles.edit16,'enable','off');
set(handles.edit17,'enable','off');
set(handles.edit18,'enable','off');
set(handles.edit19,'enable','off');
set(handles.edit20,'enable','off');
set(handles.edit21,'enable','off');
set(handles.edit22,'enable','off');
set(handles.edit23,'enable','off');
set(handles.edit24,'enable','off');
set(handles.edit25,'enable','off');
set(handles.edit26,'enable','off');



angulo1=get(handles.slider1,'value');
set(handles.edit1,'string',num2str(angulo1));
angulo2=get(handles.slider2,'value');
set(handles.edit2,'string',num2str(angulo2));
angulo3=get(handles.slider3,'value');
set(handles.edit3,'string',num2str(angulo3));
angulo4=get(handles.slider4,'value');
set(handles.edit4,'string',num2str(angulo4));
angulo5=get(handles.slider5,'value');
set(handles.edit5,'string',num2str(angulo5));
```

```
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));

[xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
set(handles.edit6,'string',num2str(xextremo));
set(handles.edit7,'string',num2str(yextremo));
set(handles.edit8,'string',num2str(zextremo));
set(handles.edit9,'string',num2str(gamma));
set(handles.edit10,'string',num2str(beta));
set(handles.edit11,'string',num2str(alpha));

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

angulo1=str2num(get(handles.edit1,'string'));
if (angulo1 >= -150) & (angulo1 <= 150)

    set(handles.slider1,'value',angulo1);
    angulo2=str2num(get(handles.edit2,'string'));
    set(handles.slider2,'value',angulo2);
    angulo3=str2num(get(handles.edit3,'string'));
    set(handles.slider3,'value',angulo3);
    angulo4=str2num(get(handles.edit4,'string'));
    set(handles.slider4,'value',angulo4);
    angulo5=str2num(get(handles.edit5,'string'));
    set(handles.slider5,'value',angulo5);
    graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

    [torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
    set(handles.edit27,'string',num2str(torque3));
    set(handles.edit28,'string',num2str(torque1));
    set(handles.edit29,'string',num2str(torque2));

   [xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
    set(handles.edit6,'string',num2str(xextremo));
    set(handles.edit7,'string',num2str(yextremo));
    set(handles.edit8,'string',num2str(zextremo));
```

```
    set(handles.edit9,'string',num2str(gamma));
    set(handles.edit10,'string',num2str(beta));
    set(handles.edit11,'string',num2str(alpha));
end

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor',[.9 .9 .9]);
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

angulo1=get(handles.slider1,'value');
set(handles.edit1,'string',num2str(angulo1));
angulo2=get(handles.slider2,'value');
set(handles.edit2,'string',num2str(angulo2));
```

```
angulo3=get(handles.slider3,'value');
set(handles.edit3,'string',num2str(angulo3));
angulo4=get(handles.slider4,'value');
set(handles.edit4,'string',num2str(angulo4));
angulo5=get(handles.slider5,'value');
set(handles.edit5,'string',num2str(angulo5));
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));

[xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
set(handles.edit6,'string',num2str(xextremo));
set(handles.edit7,'string',num2str(yextremo));
set(handles.edit8,'string',num2str(zextremo));
set(handles.edit9,'string',num2str(gamma));
set(handles.edit10,'string',num2str(beta));
set(handles.edit11,'string',num2str(alpha));


% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

angulo2=str2num(get(handles.edit2,'string'));
if (angulo2 >= -90) & (angulo2 <= 90)
    angulo1=str2num(get(handles.edit1,'string'));
    set(handles.slider1,'value',angulo1);

    set(handles.slider2,'value',angulo2);
    angulo3=str2num(get(handles.edit3,'string'));
    set(handles.slider3,'value',angulo3);
    angulo4=str2num(get(handles.edit4,'string'));
    set(handles.slider4,'value',angulo4);
    angulo5=str2num(get(handles.edit5,'string'));
```

```
    set(handles.slider5,'value',angulo5);
    graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

    [torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
    set(handles.edit27,'string',num2str(torque3));
    set(handles.edit28,'string',num2str(torque1));
    set(handles.edit29,'string',num2str(torque2));

    [xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
 angulo2, angulo3, angulo4, angulo5);
    set(handles.edit6,'string',num2str(xextremo));
    set(handles.edit7,'string',num2str(yextremo));
    set(handles.edit8,'string',num2str(zextremo));
    set(handles.edit9,'string',num2str(gamma));
    set(handles.edit10,'string',num2str(beta));
    set(handles.edit11,'string',num2str(alpha));
end


% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor',[.9 .9 .9]);
else
```

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


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


angulo1=get(handles.slider1,'value');
set(handles.edit1,'string',num2str(angulo1));
angulo2=get(handles.slider2,'value');
set(handles.edit2,'string',num2str(angulo2));
angulo3=get(handles.slider3,'value');
set(handles.edit3,'string',num2str(angulo3));
angulo4=get(handles.slider4,'value');
set(handles.edit4,'string',num2str(angulo4));
angulo5=get(handles.slider5,'value');
set(handles.edit5,'string',num2str(angulo5));
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));


[xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
```

```
set(handles.edit6,'string',num2str(xextremo));
set(handles.edit7,'string',num2str(yextremo));
set(handles.edit8,'string',num2str(zextremo));
set(handles.edit9,'string',num2str(gamma));
set(handles.edit10,'string',num2str(beta));
set(handles.edit11,'string',num2str(alpha));

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
angulo3=str2num(get(handles.edit3,'string'));
if (angulo3 >= -90) & (angulo3 <= 90)
    angulo1=str2num(get(handles.edit1,'string'));
    set(handles.slider1,'value',angulo1);
    angulo2=str2num(get(handles.edit2,'string'));
    set(handles.slider2,'value',angulo2);

    set(handles.slider3,'value',angulo3);
    angulo4=str2num(get(handles.edit4,'string'));
    set(handles.slider4,'value',angulo4);
    angulo5=str2num(get(handles.edit5,'string'));
    set(handles.slider5,'value',angulo5);
    graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

    [torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
    set(handles.edit27,'string',num2str(torque3));
    set(handles.edit28,'string',num2str(torque1));
    set(handles.edit29,'string',num2str(torque2));


   [xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
   set(handles.edit6,'string',num2str(xextremo));
   set(handles.edit7,'string',num2str(yextremo));
   set(handles.edit8,'string',num2str(zextremo));
   set(handles.edit9,'string',num2str(gamma));
   set(handles.edit10,'string',num2str(beta));
   set(handles.edit11,'string',num2str(alpha));
end
```

```
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor',[.9 .9 .9]);
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

angulo1=get(handles.slider1,'value');
set(handles.edit1,'string',num2str(angulo1));
angulo2=get(handles.slider2,'value');
set(handles.edit2,'string',num2str(angulo2));
angulo3=get(handles.slider3,'value');
```

```
set(handles.edit3,'string',num2str(angulo3));
angulo4=get(handles.slider4,'value');
set(handles.edit4,'string',num2str(angulo4));
angulo5=get(handles.slider5,'value');
set(handles.edit5,'string',num2str(angulo5));
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));

[xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
set(handles.edit6,'string',num2str(xextremo));
set(handles.edit7,'string',num2str(yextremo));
set(handles.edit8,'string',num2str(zextremo));
set(handles.edit9,'string',num2str(gamma));
set(handles.edit10,'string',num2str(beta));
set(handles.edit11,'string',num2str(alpha));


% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (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 edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


angulo4=str2num(get(handles.edit4,'string'));
if (angulo4 >= -90) & (angulo4 <= 90)
    angulo1=str2num(get(handles.edit1,'string'));
    set(handles.slider1,'value',angulo1);
    angulo2=str2num(get(handles.edit2,'string'));
    set(handles.slider2,'value',angulo2);
    angulo3=str2num(get(handles.edit3,'string'));
    set(handles.slider3,'value',angulo3);

    set(handles.slider4,'value',angulo4);
    angulo5=str2num(get(handles.edit5,'string'));
    set(handles.slider5,'value',angulo5);
    graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);
```

```
    [torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
    set(handles.edit27,'string',num2str(torque3));
    set(handles.edit28,'string',num2str(torque1));
    set(handles.edit29,'string',num2str(torque2));


    [xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
 angulo2, angulo3, angulo4, angulo5);
    set(handles.edit6,'string',num2str(xextremo));
    set(handles.edit7,'string',num2str(yextremo));
    set(handles.edit8,'string',num2str(zextremo));
    set(handles.edit9,'string',num2str(gamma));
    set(handles.edit10,'string',num2str(beta));
    set(handles.edit11,'string',num2str(alpha));
end


% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
```

```
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

angulo1=get(handles.slider1,'value');
set(handles.edit1,'string',num2str(angulo1));
angulo2=get(handles.slider2,'value');
set(handles.edit2,'string',num2str(angulo2));
angulo3=get(handles.slider3,'value');
set(handles.edit3,'string',num2str(angulo3));
angulo4=get(handles.slider4,'value');
set(handles.edit4,'string',num2str(angulo4));
angulo5=get(handles.slider5,'value');
set(handles.edit5,'string',num2str(angulo5));
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));

[xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
```

```
set(handles.edit6,'string',num2str(xextremo));
set(handles.edit7,'string',num2str(yextremo));
set(handles.edit8,'string',num2str(zextremo));
set(handles.edit9,'string',num2str(gamma));
set(handles.edit10,'string',num2str(beta));
set(handles.edit11,'string',num2str(alpha));


% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (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 edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

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

angulo5=str2num(get(handles.edit5,'string'));
if (angulo5 >= -360) & (angulo5 <= 360)
    angulo1=str2num(get(handles.edit1,'string'));
    set(handles.slider1,'value',angulo1);
    angulo2=str2num(get(handles.edit2,'string'));
    set(handles.slider2,'value',angulo2);
    angulo3=str2num(get(handles.edit3,'string'));
    set(handles.slider3,'value',angulo3);
    angulo4=str2num(get(handles.edit4,'string'));
    set(handles.slider4,'value',angulo4);

    set(handles.slider5,'value',angulo5);
    graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

    [torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
    set(handles.edit27,'string',num2str(torque3));
    set(handles.edit28,'string',num2str(torque1));
    set(handles.edit29,'string',num2str(torque2));

   [xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
    set(handles.edit6,'string',num2str(xextremo));
    set(handles.edit7,'string',num2str(yextremo));
    set(handles.edit8,'string',num2str(zextremo));
    set(handles.edit9,'string',num2str(gamma));
    set(handles.edit10,'string',num2str(beta));
    set(handles.edit11,'string',num2str(alpha));
end
```

```
% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

angulo1=get(handles.slider1,'value');
set(handles.edit1,'string',num2str(angulo1));
angulo2=get(handles.slider2,'value');
set(handles.edit2,'string',num2str(angulo2));
angulo3=get(handles.slider3,'value');
```

```
set(handles.edit3,'string',num2str(angulo3));
angulo4=get(handles.slider4,'value');
set(handles.edit4,'string',num2str(angulo4));
angulo5=get(handles.slider5,'value');
set(handles.edit5,'string',num2str(angulo5));
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));

[xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
set(handles.edit6,'string',num2str(xextremo));
set(handles.edit7,'string',num2str(yextremo));
set(handles.edit8,'string',num2str(zextremo));
set(handles.edit9,'string',num2str(gamma));
set(handles.edit10,'string',num2str(beta));
set(handles.edit11,'string',num2str(alpha));

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
set(handles.radiobutton2,'value',0);
set(handles.radiobutton3,'value',0);
set(handles.edit1,'enable','on');
set(handles.slider1,'enable','on');
set(handles.edit2,'enable','on');
set(handles.slider2,'enable','on');
set(handles.edit3,'enable','on');
set(handles.slider3,'enable','on');
set(handles.edit4,'enable','on');
set(handles.slider4,'enable','on');
set(handles.edit5,'enable','on');
set(handles.slider5,'enable','on');
set(handles.edit6,'enable','off');
set(handles.edit7,'enable','off');
set(handles.edit8,'enable','off');
set(handles.pushbutton1,'enable','off');
set(handles.pushbutton3,'enable','off');
set(handles.pushbutton4,'enable','off');
set(handles.pushbutton5,'enable','off');
set(handles.edit9,'enable','off');
set(handles.edit10,'enable','off');
set(handles.edit11,'enable','off');

set(handles.edit12,'enable','off');
set(handles.edit13,'enable','off');
set(handles.edit14,'enable','off');
set(handles.edit15,'enable','off');
set(handles.edit16,'enable','off');
set(handles.edit17,'enable','off');
set(handles.edit18,'enable','off');
set(handles.edit19,'enable','off');
```

```
set(handles.edit20,'enable','off');
set(handles.edit21,'enable','off');
set(handles.edit22,'enable','off');
set(handles.edit23,'enable','off');
set(handles.edit24,'enable','off');
set(handles.edit25,'enable','off');
set(handles.edit26,'enable','off');


% Hint: get(hObject,'Value') returns toggle state of radiobutton1


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.radiobutton1,'value',0);
set(handles.edit1,'enable','off');
set(handles.slider1,'enable','off');
set(handles.edit2,'enable','off');
set(handles.slider2,'enable','off');
set(handles.edit3,'enable','off');
set(handles.slider3,'enable','off');
set(handles.edit4,'enable','off');
set(handles.slider4,'enable','off');
set(handles.edit5,'enable','off');
set(handles.slider5,'enable','off');
set(handles.edit6,'enable','on');
set(handles.edit7,'enable','on');
```

```
set(handles.edit8,'enable','on');
set(handles.pushbutton1,'enable','on');
set(handles.pushbutton3,'enable','off');
set(handles.pushbutton4,'enable','off');
set(handles.pushbutton5,'enable','off');
set(handles.edit9,'enable','on');
set(handles.edit10,'enable','on');
set(handles.edit11,'enable','on');


set(handles.edit12,'enable','off');
set(handles.edit13,'enable','off');
set(handles.edit14,'enable','off');
set(handles.edit15,'enable','off');
set(handles.edit16,'enable','off');
set(handles.edit17,'enable','off');
set(handles.edit18,'enable','off');
set(handles.edit19,'enable','off');
set(handles.edit20,'enable','off');
set(handles.edit21,'enable','off');
set(handles.edit22,'enable','off');
set(handles.edit23,'enable','off');
set(handles.edit24,'enable','off');
set(handles.edit25,'enable','off');
set(handles.edit26,'enable','off');



% Hint: get(hObject,'Value') returns toggle state of radiobutton2
```

```
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (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 edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (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 edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double



% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (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 edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (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 edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (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 edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (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 edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (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 edit9_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit9 (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 edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (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 edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (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 edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (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 edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (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 edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
```

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

xextremo=str2num(get(handles.edit6,'string'));
yextremo=str2num(get(handles.edit7,'string'));
zextremo=str2num(get(handles.edit8,'string'));
alpha=str2num(get(handles.edit9,'string'));
phi=str2num(get(handles.edit10,'string'));
tetha=str2num(get(handles.edit11,'string'));

[angulo1, angulo2, angulo3, angulo4, angulo5] = kineinv(xextremo, yextremo,
zextremo, alpha, phi, tetha);

set(handles.edit1,'string',num2str(angulo1));
set(handles.edit2,'string',num2str(angulo2));
set(handles.edit3,'string',num2str(angulo3));
set(handles.edit4,'string',num2str(angulo4));
set(handles.edit5,'string',num2str(angulo5));
set(handles.slider1,'value',angulo1);
set(handles.slider2,'value',angulo2);
set(handles.slider3,'value',angulo3);
set(handles.slider4,'value',angulo4);
set(handles.slider5,'value',angulo5);

graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

angulo1=0;
angulo2=0;
angulo3=0;
angulo4=0;
angulo5=0;
set(handles.edit1,'string',num2str(angulo1));
set(handles.edit2,'string',num2str(angulo2));
set(handles.edit3,'string',num2str(angulo3));
set(handles.edit4,'string',num2str(angulo4));
set(handles.edit5,'string',num2str(angulo5));
set(handles.slider1,'value',angulo1);
set(handles.slider2,'value',angulo2);
set(handles.slider3,'value',angulo3);
set(handles.slider4,'value',angulo4);
set(handles.slider5,'value',angulo5);
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));
```

```
[xextremo, yextremo, zextremo, gamma, beta, alpha]=puntoextremodif(angulo1,
angulo2, angulo3, angulo4, angulo5);
set(handles.edit6,'string',num2str(xextremo));
set(handles.edit7,'string',num2str(yextremo));
set(handles.edit8,'string',num2str(zextremo));
set(handles.edit9,'string',num2str(gamma));
set(handles.edit10,'string',num2str(beta));
set(handles.edit11,'string',num2str(alpha));


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (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 edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (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 edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (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 edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (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 edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit14 (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 edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (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 edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (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 edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (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 edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a double


% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (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 edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (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 edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16 as a double


% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (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 edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (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 edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17 as a double


% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (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 edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (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 edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18 as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (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 edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (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 edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19 as a double


% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (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 edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (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 edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20 as a double


% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21 as a double


% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (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 edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (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 edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22 as a double


% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (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 edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (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 edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (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 edit24_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (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 edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24 as a double


% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (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 edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (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 edit25 as text
%        str2double(get(hObject,'String')) returns contents of edit25 as a double


% --- Executes during object creation, after setting all properties.
function edit26_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit26 (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 edit26_Callback(hObject, eventdata, handles)
% hObject    handle to edit26 (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 edit26 as text
%        str2double(get(hObject,'String')) returns contents of edit26 as a double


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

task1_1=str2num(get(handles.edit12,'string'));
task1_2=str2num(get(handles.edit13,'string'));
task1_3=str2num(get(handles.edit14,'string'));
task1_4=str2num(get(handles.edit15,'string'));
task1_5=str2num(get(handles.edit16,'string'));

task2_1=str2num(get(handles.edit17,'string'));
task2_2=str2num(get(handles.edit18,'string'));
task2_3=str2num(get(handles.edit19,'string'));
task2_4=str2num(get(handles.edit20,'string'));
task2_5=str2num(get(handles.edit21,'string'));
```

```
task3_1=str2num(get(handles.edit22,'string'));
task3_2=str2num(get(handles.edit23,'string'));
task3_3=str2num(get(handles.edit24,'string'));
task3_4=str2num(get(handles.edit25,'string'));
task3_5=str2num(get(handles.edit26,'string'));


[angulo1, angulo2, angulo3, angulo4, angulo5] =
tarea(task1_1,task1_2,task1_3,task1_4,task1_5);

set(handles.edit1,'string',num2str(angulo1));
set(handles.edit2,'string',num2str(angulo2));
set(handles.edit3,'string',num2str(angulo3));
set(handles.edit4,'string',num2str(angulo4));
set(handles.edit5,'string',num2str(angulo5));
set(handles.slider1,'value',angulo1);
set(handles.slider2,'value',angulo2);
set(handles.slider3,'value',angulo3);
set(handles.slider4,'value',angulo4);
set(handles.slider5,'value',angulo5);
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));

pause(5);
```

```
[angulo1, angulo2, angulo3, angulo4, angulo5] =
tarea(task2_1,task2_2,task2_3,task2_4,task2_5);

set(handles.edit1,'string',num2str(angulo1));
set(handles.edit2,'string',num2str(angulo2));
set(handles.edit3,'string',num2str(angulo3));
set(handles.edit4,'string',num2str(angulo4));
set(handles.edit5,'string',num2str(angulo5));
set(handles.slider1,'value',angulo1);
set(handles.slider2,'value',angulo2);
set(handles.slider3,'value',angulo3);
set(handles.slider4,'value',angulo4);
set(handles.slider5,'value',angulo5);
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));

pause(5);


[angulo1, angulo2, angulo3, angulo4, angulo5] =
tarea(task3_1,task3_2,task3_3,task3_4,task3_5);

set(handles.edit1,'string',num2str(angulo1));
set(handles.edit2,'string',num2str(angulo2));
set(handles.edit3,'string',num2str(angulo3));
set(handles.edit4,'string',num2str(angulo4));
set(handles.edit5,'string',num2str(angulo5));
```

```
set(handles.slider1,'value',angulo1);
set(handles.slider2,'value',angulo2);
set(handles.slider3,'value',angulo3);
set(handles.slider4,'value',angulo4);
set(handles.slider5,'value',angulo5);
graficapuma(angulo1,angulo2,angulo3,angulo4,angulo5);

[torque1,torque2,torque3]=torques_puma(angulo2,angulo3,24,24,100,100);
set(handles.edit27,'string',num2str(torque3));
set(handles.edit28,'string',num2str(torque1));
set(handles.edit29,'string',num2str(torque2));

pause(5);


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3


set(handles.radiobutton1,'value',0);
set(handles.radiobutton2,'value',0);
set(handles.edit1,'enable','off');
set(handles.slider1,'enable','off');
set(handles.edit2,'enable','off');
set(handles.slider2,'enable','off');
set(handles.edit3,'enable','off');
```

```
set(handles.slider3,'enable','off');
set(handles.edit4,'enable','off');
set(handles.slider4,'enable','off');
set(handles.edit5,'enable','off');
set(handles.slider5,'enable','off');
set(handles.edit6,'enable','off');
set(handles.edit7,'enable','off');
set(handles.edit8,'enable','off');
set(handles.pushbutton1,'enable','on');
set(handles.pushbutton3,'enable','on');
set(handles.pushbutton4,'enable','on');
set(handles.pushbutton5,'enable','off');
set(handles.edit9,'enable','off');
set(handles.edit10,'enable','off');
set(handles.edit11,'enable','off');


set(handles.edit12,'enable','on');
set(handles.edit13,'enable','on');
set(handles.edit14,'enable','on');
set(handles.edit15,'enable','on');
set(handles.edit16,'enable','on');
set(handles.edit17,'enable','on');
set(handles.edit18,'enable','on');
set(handles.edit19,'enable','on');
set(handles.edit20,'enable','on');
set(handles.edit21,'enable','on');
set(handles.edit22,'enable','on');
set(handles.edit23,'enable','on');
set(handles.edit24,'enable','on');
set(handles.edit25,'enable','on');
```

```
set(handles.edit26,'enable','on');



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in togglebutton2.
function togglebutton2_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton2


% --- Executes during object creation, after setting all properties.
function edit27_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit27 (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 edit27_Callback(hObject, eventdata, handles)
% hObject    handle to edit27 (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 edit27 as text
%        str2double(get(hObject,'String')) returns contents of edit27 as a double


% --- Executes during object creation, after setting all properties.
function edit28_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit28 (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 edit28_Callback(hObject, eventdata, handles)
% hObject    handle to edit28 (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 edit28 as text
%        str2double(get(hObject,'String')) returns contents of edit28 as a double


% --- Executes during object creation, after setting all properties.
function edit29_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit29 (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 edit29_Callback(hObject, eventdata, handles)
% hObject    handle to edit29 (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 edit29 as text
```

```
%       str2double(get(hObject,'String')) returns contents of edit29 as a double


% --- Executes during object creation, after setting all properties.
function edit30_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit30 (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 edit30_Callback(hObject, eventdata, handles)
% hObject    handle to edit30 (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 edit30 as text
%       str2double(get(hObject,'String')) returns contents of edit30 as a double


% --- Executes during object creation, after setting all properties.
function edit31_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit31 (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 edit31_Callback(hObject, eventdata, handles)
% hObject    handle to edit31 (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 edit31 as text
%        str2double(get(hObject,'String')) returns contents of edit31 as a double


% --- Executes during object creation, after setting all properties.
function edit32_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit32 (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 edit32_Callback(hObject, eventdata, handles)
% hObject    handle to edit32 (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 edit32 as text
%        str2double(get(hObject,'String')) returns contents of edit32 as a double


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

A 2.1 CODIGO FUENTE DEL PROGRAMA DE LA CINEMATICA DIRECTA

```
function [xextremo, yextremo, zextremo, gamma, beta, alpha] =
puntoextremodif(angulo1, angulo2,angulo3, angulo4, angulo5)

% DEFINICION DE LAS DIRECCIONES DEL PUERTO PARALELO
dio = digitalio('parallel','LPT1') % define dentro de la variable "dio" la dir del puerto
data=addline(dio,0:7,0,'out');% dio = puerto paralelo,  0:7= 8 lineas de informacion
control=addline(dio,0:3,2,'out'); % 0:3= 4 lineas de informacion


L1=1.9;
L2=1.74;
L3=0.2;
L4= 2.2;
L5= 2.2;
L6= 1.5;

an1=round(angulo1)+150;
an2=round(angulo2)+90;
an3=round(angulo3)+90;
an4=round(angulo4)+90;
an5=round(angulo5);

angulo1=angulo1*pi/180;
```

```
angulo2=angulo2*pi/180;
angulo3=angulo3*pi/180;
angulo4=angulo4*pi/180;
angulo5=angulo5*pi/180;

q=[angulo1 angulo2 angulo3 angulo4+pi/2 angulo5];
T=fkine(dh,q);
gba=tr2rpy(T);
gba=gba*180/pi;
xextremo= T(1,4);
yextremo= T(2,4);
zextremo= T(3,4);
gamma=gba(1);
beta=gba(2);
alpha=gba(3);



putvalue(data,an1);
putvalue(control,1);% BIT SELECCION Motor1
pause(0.01);

putvalue(data,an2);
putvalue(control,2);% BIT SELECCION Motor2
pause(0.01);

putvalue(data,an3);
putvalue(control,3);% BIT SELECCION Motor3
pause(0.01);

putvalue(data,an4);
```

```
putvalue(control,4);% BIT SELECCION Motor4
pause(0.01);
putvalue(data,an5);
putvalue(control,5);% BIT SELECCION Motor5
pause(0.01);
```

A 1.1 CODIGO FUENTE DEL PROGRAMA DE LA CINEMATICA INVERSA

```
function [angulo1, angulo2, angulo3, angulo4, angulo5] =
kineinv(xextremo, yextremo, zextremo, gamma, beta, alpha)


% DEFINICION DE LAS DIRECCIONES DEL PUERTO PARALELO
dio = digitalio('parallel','LPT1') % define dentro de la variable "dio" la dir
del puerto
data=addline(dio,0:7,0,'out');% dio = puerto paralelo,  0:7= 8 lineas de
informacion
control=addline(dio,0:3,2,'out'); % 0:3= 4 lineas de informacion


%PROGRAMA PARA CALCULAR LA CINEMATICA INVERSA DEL
MANIPULADOR A PARTIR DE LAS
%COORDENADAS DEL PUNTO MAS EXTREMO DEL MISMO
%SOLUCION POR METODO GEOMETRICO

%MEDIDAS DE LOS ESLABONES DEL MANIPULADOR
```

```
L1= 1.9;
L2= 1.74;
L3= 0.2;
L4= 2.2;
L5= 2.2;
L6= 1.5;

%CALCULO DE LA MATRIZ DE ORIENTACION
gamma=gamma*pi/180;
beta=beta*pi/180;
alpha=alpha*pi/180;
A05=rpy2tr([gamma beta alpha]);

%MATRIZ HOMOGENEA DE POSICION FINAL A05
A05(1,4)= xextremo;
A05(2,4)= yextremo;
A05(3,4)= zextremo;

px=A05(1,3)*L6;
py=A05(2,3)*L6;
pz=A05(3,3)*L6;

%MATRIZ HOMOGENEA HASTA LA MUÑECA A03
A03= eye(4);
A03(1,4)=A05(1,4)-px;
A03(2,4)=A05(2,4)-py;
A03(3,4)=A05(3,4)-pz;
```

```
%CALCULO DE LA CINEMATICA INVERSA DEL SCORBOT POR
METODOS GEOMETRICOS.

%PRIMERA ARTICULACION
angulo1= atan2(A03(2,4), A03(1,4));

%SEGUNDA ARTICULACION
x=((((A03(1,4))^2)+((A03(2,4))^2)+((A03(3,4))^2)+((L3)^2)+((L1+L2)^2)
)-
((2*L3*sqrt(((A03(1,4))^2)+((A03(2,4))^2)))+(2*A03(3,4)*(L1+L2))+((L4)
^2)+((L5)^2)));
x=x/(2*L4*L5);
y=sqrt(1-(x^2));
k1=sqrt(((A03(1,4))^2)+((A03(2,4))^2))-L3;
k2=A03(3,4)-L1-L2;
m=-((k1*L5*y)+(k2*L5*x)+(k2*L4));
m=m/((k1)^2+(k2)^2);
n=(k1*L5*x)-(k2*L5*y)+(k1*L4);
n=n/((k1)^2+(k2)^2);
angulo2= atan2(m,n);

%TERCERA ARTICULACION
angulo3= atan2(y,x);

%CUARTA ARTICULACION
ax=A05(1,3);
```

```
ay=A05(2,3);
az=A05(3,3);
o1=cos(angulo1)*cos(angulo2)*cos(angulo3)-
cos(angulo1)*sin(angulo2)*sin(angulo3);
o2=sin(angulo1)*cos(angulo2)*cos(angulo3)-
sin(angulo1)*sin(angulo2)*sin(angulo3);
o3=sin(angulo2)*cos(angulo3)+cos(angulo2)*sin(angulo3);
o=o1*ax+o2*ay-o3*az;
p=sqrt(1 - (o^2));
angulo4= atan2(p,o);

%QUINTA ARTICULACION
ox=A05(1,2);
oy=A05(2,2);
r=-sin(angulo1)*ox+cos(angulo1)*oy;
s=sqrt(1 - (r^2));
angulo5= atan2(s,r);



%Se entregan en grados

angulo1= angulo1*180/pi;
angulo2= angulo2*180/pi;
angulo3= angulo3*180/pi;
angulo4= angulo4*180/pi;
angulo5= angulo5*180/pi;
```

```
ang1=round(angulo1)+150
ang2=round(angulo2)+90
ang3=round(angulo3)+90
ang4=round(angulo4)+90
ang5=round(angulo5)

angulo1= angulo1
angulo2= angulo2
angulo3= angulo3
angulo4= angulo4
angulo5= angulo5

putvalue(data,ang1);
putvalue(control,1);% BIT SELECCION Motor1
pause(2);

putvalue(data,ang2);
putvalue(control,2);% BIT SELECCION Motor2
pause(2);

putvalue(data,ang3);
putvalue(control,3);% BIT SELECCION Motor3
pause(2);

putvalue(data,ang4);
putvalue(control,4);% BIT SELECCION Motor4
```

```
pause(2);
putvalue(data,ang5);
putvalue(control,5);% BIT SELECCION Motor5
pause(2);
```

A 1.2 CODIGO FUENTE DEL PROGRAMA DEL MICRO CONTROLADOR

```
$Include 'gpgtregs.inc'

RAMStart    EQU  $0040
RomStart    EQU  $8000
VectorStart EQU  $FFDC

        org    RamStart

entrada      ds     8
conversor    ds     8

      org    RomStart

principal:
       rsp
       mov  #$70,ADCLK
       clra
       clrx
       mov  #$31,CONFIG1
       mov  #$00,DDRA
       mov  #$00,DDRB
       mov  #$10,DDRC
       mov  #$03,DDRE
```

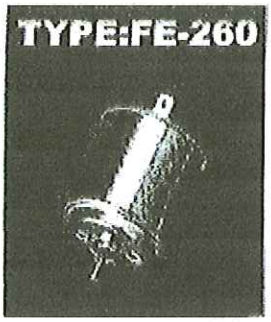
```
        mov  #$FF,DDRD
inicio:
        ldx  PTC
        cbeqx #$00,condicion
        nop
        nop

motor:
    aix     #-1
    lda     PTA
    sta     entrada,X           ;salta almacena la dirección del registro para
nuevo dato

condicion:
        mov     #$00,PTD
        ldx     #$00



loop:
    cpx   #$08
    beq    final
    stx   ADSCR
final_adc:
        brset 7,ADSCR,compara
        bra  final_adc

compara:
    lda   ADR
    sta   conversor,x
    lda   entrada,x
    cmp    conversor,x              ;'valor entrada'-'valor del conversor'
```

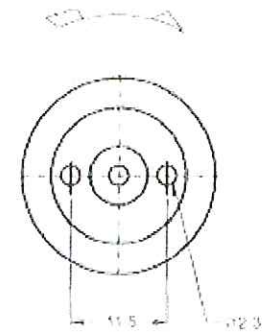
```
        blo    menor
        bhi    mayor
        bra    igual
mayor:
      mov     #$01,PTE
      bra     fin
menor:
      mov     #$02,PTE
      bra     fin
igual:
      mov     #$03,PTE
      bset    4,PTC
      bra     fin
fin:
   aix #1
   lda #$FF
retardo:
       dbnza   retardo
   inc PTD                    ;sugerencia: poner retardo antes que D cambie
   bra loop

final:
        bra   inicio

dummy_isr:
        rti
swi_isr:                ; do nothing
      rti

   org    VectorStart
```

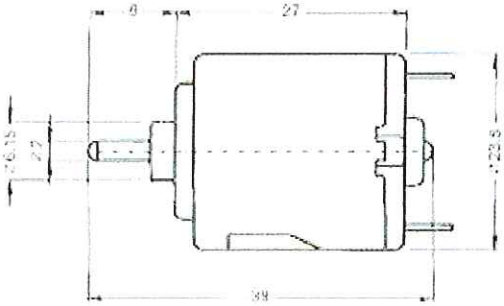
```
dw    dummy_isr     ; Time Base Vector
dw    dummy_isr     ; ADC Conversion Complete
dw    dummy_isr     ; Keyboard Vector
dw    dummy_isr     ; SCI Transmit Vector
dw    dummy_isr     ; SCI Receive Vector
dw    dummy_isr     ; SCI Error Vector
dw    dummy_isr     ; SPI Transmit Vector
dw    dummy_isr     ; SPI Receive Vector
dw    dummy_isr     ; TIM2 Overflow Vector
dw    dummy_isr     ; TIM2 Channel 1 Vector
dw    dummy_isr     ; TIM2 Channel 0 Vector
dw    dummy_isr     ; TIM1 Overflow Vector
dw    dummy_isr     ; TIM1 Channel 1 Vector
dw    dummy_isr     ; TIM1 Channel 0 Vector
dw    dummy_isr     ; PLL Vector
dw    dummy_isr     ; ~IRQ Vector
dw    swi_isr       ; SWI Vector
dw    principal     ; Reset Vector
```

A2 DATASHEET MOTORES

NICHIBO DC MOTOR



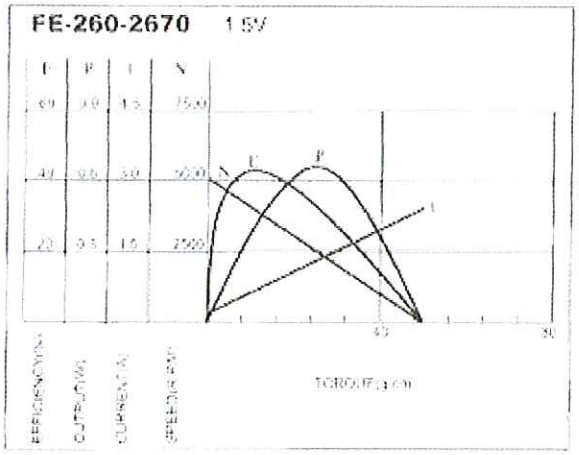


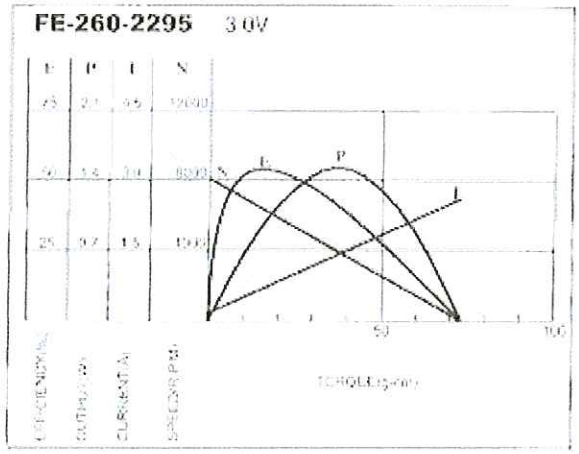


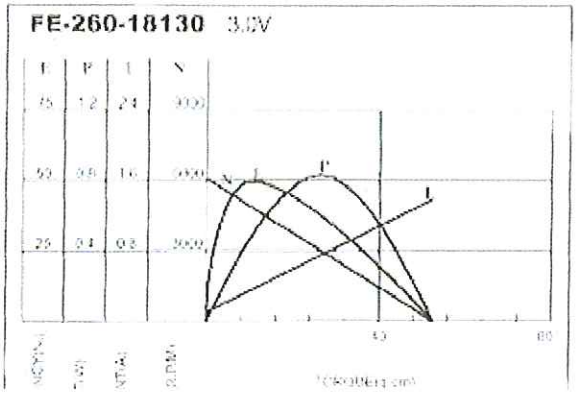
代表的用途：玩具
Typical Application : Motorized Toy

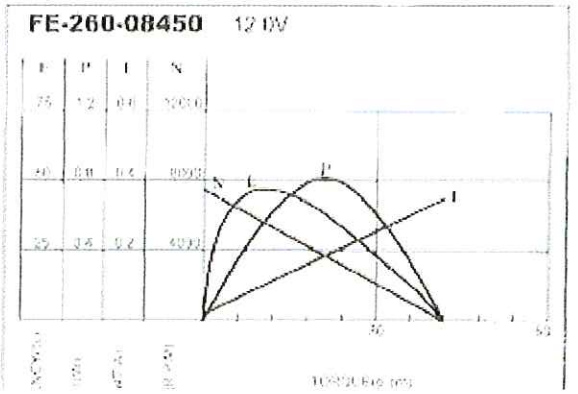
WEIGHT : 30g(APPROX)

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL	
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE	OUTPUT	EFF.	CURRENT	TORQUE
			rpm	A	rpm	A	g-cm	W	%	A	g-cm
FE-260-2670	1.5 – 3.0	1.5V CONSTANT	5100	0.180	3700	0.66	10.53	0.40	42.92	2.41	50.06
FE-260-2295	1.5 – 4.5	3.0V CONSTANT	8100	0.150	6850	0.63	13.56	0.98	53.91	2.62	73.29
FE-260-18130	1.5 – 4.5	3.0V CONSTANT	6100	0.100	4780	0.36	10.27	0.50	49.63	1.39	52.75
FE-260-08450	6.0 – 12.0	12.0V CONSTANT	7500	0.035	5550	0.11	10.32	0.60	46.63	0.35	42.00

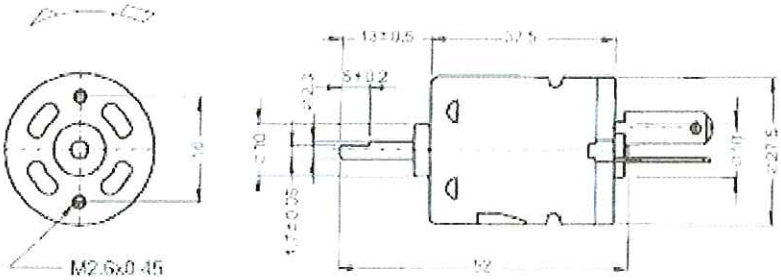








NICHIBO DC MOTOR

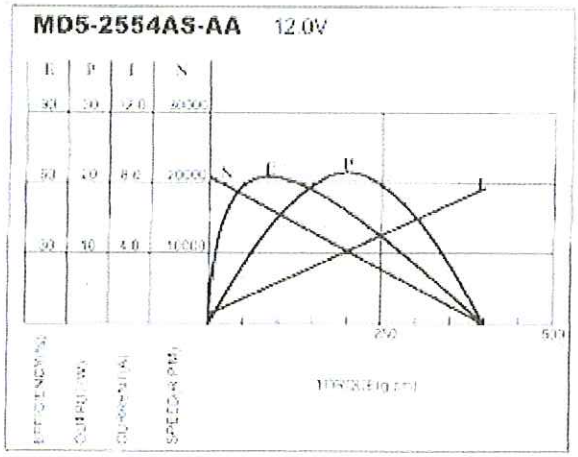


代表的用途：汽(車用)水泵
Typical Application : Automotive Water Pump

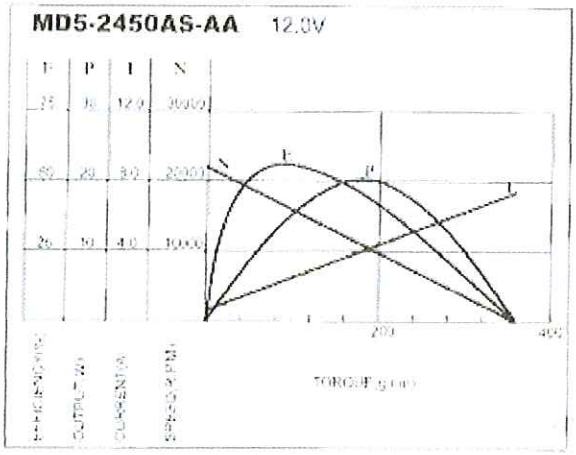
WEIGHT : 55g(APPROX)

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL	
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE	OUTPUT	EFF	CURRENT	TORQUE
	V		rpm	A	rpm	A	g-cm	W	%	A	g-cm
MD5-2554AS-AA	6.0 – 12.0	12.0V CONSTANT	20800	0.55	17000	1.70	72.92	12.72	62.38	7.70	400
MD5-2450AS-AA	6.0 – 12.0	12.0V CONSTANT	22000	0.60	17200	2.00	77.50	13.73	55.91	7.40	360
MD5-1980AS-AA	12.0 – 24.0	24.0V CONSTANT	24000	0.40	22600	1.13	88.50	20.55	60.92	5.87	470
MD5-15120AS-AA	12.0 – 24.0	24.0V CONSTANT	15200	0.24	15100	0.70	66.68	10.34	61.54	2.86	340

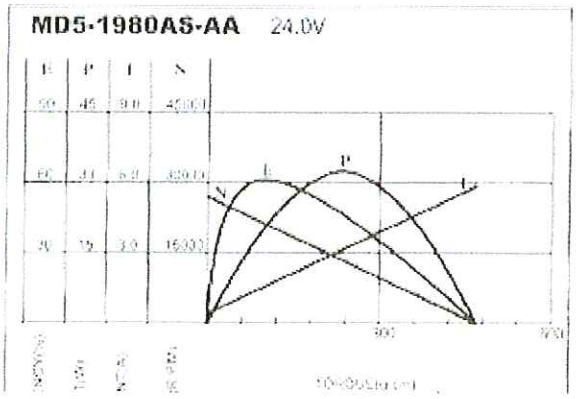
MD5-2554AS-AA 12.0V



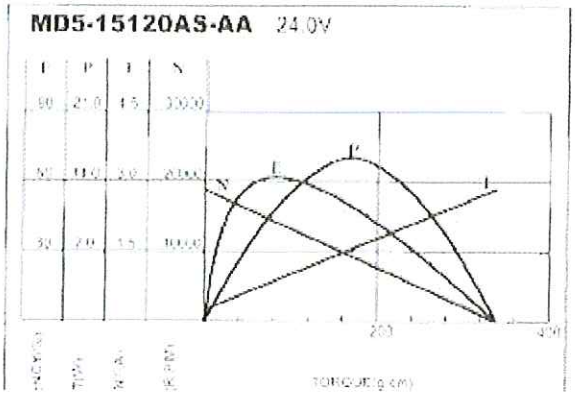
MD5-2450AS-AA 12.0V



MD5-1980AS-AA 24.0V



MD5-15120AS-AA 24.0V



DC Gear Head Motors

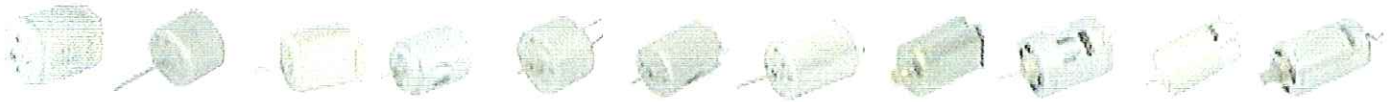
JAMECO RellaPro — ✓ Reliable & Consistent ✓ Professional Grade ✓ Awesome Prices ✓ Backed by Jameco

Color Code Legend: New | Sale | Lower$ | QtySave

1.2VDC-24VDC Motors

Quantity SAVE

MITSUMI | MABUCHI MOTOR | Nidec Motor | star motor | JOHNSON ELECTRIC



311976CH | 295849CH | 154915CH/ 154923CH | 211246CH | 267231CH | 177498CH | 174692CH | 181112CH | 206949CH | 153656CH | 267291CH

Part Number	Product Number	Manufacturer	Nominal Voltage (VDC)	Voltage Range (VDC)	@ Maximum Efficiency Current (Amps)	@ Maximum Efficiency Speed (RPM)	@ Maximum Efficiency Torque (g-cm)	Efficiency	Terminal Type	Shaft "Dia.	Shaft "Length	Size (D" x L")	Pricing 1	Pricing 10	Pricing 100
311976CH	20519-34442	Johnson Electric	1.2	—	0.363	4692	130.4	55	0.08"/solder	0.08	0.33	1.00 x 1.20	$1.25	$1.09	$.99
295849CH	RF-310T-11400	Mabuchi Motor	2.5	1-6	0.060	2190	3.3	—	0.06"/solder	0.08	0.89	0.96 x 0.71	1.49	1.34	1.21
154915CH	ST-130-12240[1]	Star Motor	3	1.5-3	0.170	3588	7.4	55	0.09"/solder	0.08	0.37	0.79 x 1.13	1.09	.98	.86
154923CH	ST-130-22770[1]	Star Motor	3	1.5-3	1.150	12500	10.3	40	0.09"/solder	0.08	0.37	0.79 x 1.13	.99	.84	.63
211246CH	RC-260RA-2670	Mabuchi Motor	4.5	3-4.5	1.430	15290	21.8	53	0.11"/solder	0.08	0.28	0.93 x 1.06	1.49	1.27	.95
267231CH	FRF-510-12620	Telco	6	3-12	0.030	2700	10.0	—	Wire w/ conn.	0.08	0.31	1.25 x 0.86	.99	.89	.79
177498CH	RC280RA-2485	Mabuchi Motor	7.2	3-14	1.180	14260	33.3	57	11"/solder	0.08	0.43	0.95 x 1.20	1.75	1.49	1.12
174692CH	21226-384221	Johnson Electric	12	9-14	0.015	3000	—	—	0.11"/solder	0.08	0.69	0.97 x 1.19	2.49	2.15	1.59
181112CH	M25N-2 ✪	Mitsumi	12	5-15	0.018	3000	18.0	—	11"/solder	0.08	0.26	0.97 x 1.30	2.29	2.05	1.85
206949CH	QJT360S14280	Nichibo	12	9-30	0.350	4930	44.5	52	0.11"/solder	0.09	0.45	1.09 x 1.30	1.69	1.45	1.09
153656CH	RS550PF	Mabuchi Motor[2]	12	6-15	1.400	6500	44.1	—	0.15"/solder	0.13	0.66	1.45 x 2.25	3.29	2.99	2.69
267291CH	SUN-555A-LMRK	Telco	24	—	0.150	4100	84	—	Wire w/ conn.	0.13	0.40	1.45 x 2.25	2.99	2.54	1.91

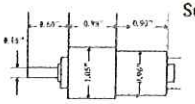
154915CH/154923CH: Use bushing P/N 162392CH, pg 256 [2] Manufacturer may vary ✪ Blue Part No. = Limited Quantity Available

12VDC and 24VDC Reversible Gear Head Motors

JAMECO RellaPro — Jameco EXCLUSIVE

- Solder-type terminal
- High torque construction
- Oil bearing design for long service life
- Insulation resistance: 10MΩ
- Dielectric strength: 300VDC

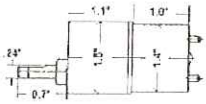
167038CH | 159417CH | 155011CH



HN27GD Series

GH30GM Series

GH35GM Series



Quantity SAVE

HN27GD Series

Part Number	Product Number	Rated Voltage (VDC)	Operating Range (VDC)	@ Maximum Efficiency Current (mA)	@ Maximum Efficiency Speed (RPM)	@ Maximum Efficiency Torque (g-cm)	Gear Ratio	Gear Case Size Dia. x Length (in.)	Motor Size Dia. x Length (in.)	Shaft Size Dia. x Length (in.)	Pricing 1	Pricing 10	Pricing 50
167038CH	GH12-0953Y	12	4.5 – 24	90	9	1200	188:1	1.0 x 0.9	1.1 x 1.0	.16 x .60	$16.49	$14.85	$13.39
167046CH	GH12-1828Y	12	4.5 – 24	195	31	1300	90:1	1.1 x 1.3	1.1 x 0.8	.16 x .60	21.95	19.79	17.79

GH30GM Series

170641CH	GH12-1828Y-10	12	6.0 – 24	105	8	3500	332:1	1.2 x 1.2	1.2 x 0.9	.16 x .70	$23.95	$20.36	$15.27
153437CH	GH12-1015Y	12	3.0 – 12	90	7	800	200:1	0.9 x 1.0	1.1 x 0.6	.16 x .70	18.95	17.09	15.35
159417CH	GH12-1921Y	12	4.5 – 12	200	35	2000	132:1	1.2 x 1.2	1.2 x 0.9	.16 x .70	21.95	19.79	17.79
162190CH	GH12-1324Y225	12	4.5 – 12	145	176	300	30:1	1.0 x 0.9	1.1 x 0.6	.16 x .70	16.95	15.25	13.75

GH35GM Series

155011CH	GH12-1926	12	4.5 – 12	250	2	6000	3000:1	1.3 x 0.9	1.5 x 1.1	.23 x .90	$21.95	$19.79	$17.79
155820CH	GH12-1830Y-P	12	4.5 – 12	220	4.5	4500	1000:1	1.3 x 0.9	1.5 x 1.0	.23 x .90	21.95	19.79	17.79
155838CH	GH12-1641T-L	12	3.0 – 12	250	15	3000	270:1	1.3 x 0.9	1.5 x 0.8	.23 x .90	23.95	21.59	19.45
152910CH	GH12-1926Y	12	4.5 – 12	172	35	2370	100:1	1.3 x 1.0	1.5 x 0.7	.24 x .90	21.95	19.75	17.75
151440CH	GH12-1632TI	12	4.5 – 12	275	57	2200	100:1	1.3 x 0.9	1.5 x 0.7	.23 x .90	21.95	18.55	16.69
155854CH	GH12-1641T-F	12	3.0 – 12	250	71	1000	60:1	1.3 x 0.9	1.4 x 0.9	.23 x .90	21.95	19.79	17.79
155862CH	GH12-1926Y-F	12	4.5 – 12	300	70	1000	60:1	1.3 x 0.9	1.4 x 0.9	.23 x .90	21.95	19.79	17.79
161373CH	GH12-1345T	12	4.5 – 12	185	116	650	30:1	1.3 x 0.9	1.5 x 0.7	.23 x .90	23.95	21.59	19.45
161381CH	GH12-1634T	12	4.5 – 12	293	145	850	30:1	1.3 x 0.9	1.4 x 0.7	.23 x .90	21.95	19.79	17.79
161785CH	GH12-1640Y	12	4.5 – 24	120	220	180	10:1	1.3 x 0.9	1.4 x 0.7	.23 x .90	23.95	20.36	15.27
176049CH	GH12-1640Y00	24	12 – 24	340	1.8	6000	3000:1	1.3 x 0.9	1.5 x 1.1	.23 x .90	21.95	18.66	15.52
176031CH	GH24-1640Y06	24	12 – 24	340	50	2000	90:1	1.3 x 0.9	1.4 x 0.7	.23 x .90	21.95	19.79	17.79
176022CH	GH24-1640Y19	24	12 – 24	340	148	1000	30:1	1.3 x 0.9	1.4 x 0.7	.23 x .90	21.95	19.79	17.79

Save Big!
Special Values!

New Lower Prices = Orange,
Sale Price Items = Yellow,
Exceptional Quantity Discounts = Green

A3 DATASHEET POTENCIOMETRO

Potentiometers – 24mm Linear and Audio Taper

1/2-Watt Linear Taper (±10%)

Temperature range: -55°C to +125°C
Non-rotational key
Non-slotted shaft end
Mounting hole: 0.3125"
Shaft diameter: 0.25"
Shaft length: 0.50"
Maximum panel thickness: 0.1875"

29081CH

Includes: Mounting hardware

Qty.	1	10	100	500
Price	$.99	$.79	$.65	$.55

Part No	Product No.	Ohms	Part No	Product No.	Ohms
9049CH	P1K	1k	29102CH	P100K	100k
9196CH	P5K	5k	29065CH	P1MEG	1M
9081CH	P10K	10k			

RV4N Style 2-Watt Linear Taper (±10%)

Shaft diameter: 0.25"
Shaft length: 0.875"
2W @ 70°C ambient
Conductive plastic element
Temp. range: -55°C to +120°C
Non-rotational key
Mounting hole: 0.375"
Maximum panel thickness: 0.25"

95417CH

Includes: Mounting hardware

Qty.	1	10	25	50
Price	$7.39	$6.54	$5.45	$4.51

Part No.	Product No.	Military No.	Ohms
41550CH	53C3-50	RV4NAYSD500A	50
41568CH	53C3-100	RV4NAYSD101A	100
41576CH	53C3-150	RV4NAYSD151A	150
41584CH	53C3-250	RV4NAYSD251A	250
41592CH	53C3-350	RV4NAYSD351A	350
43610CH	53C3-500	RV4NAYSD501A	500
5936CH	53C3-1k	RV4NAYSD102A	1k
41605CH	53C3-1.5k	RV4NAYSD152A	1.5k
41613CH	53C3-2k	RV4NAYSD202A	2k
41621CH	53C3-2.5k	RV4NAYSD252A	2.5k
41630CH	53C3-3.5k	RV4NAYSD352A	3.5k
4995CH	53C3-5k	RV4NAYSD502A	5k
4952CH	53C3-10k	RV4NAYSD103A	10k
41648CH	53C3-15k	RV4NAYSD153A	15k
41656CH	53C3-20k	RV4NAYSD203A	20k
41664CH	53C3-25k	RV4NAYSD253A	25k
4117CH	53C3-50k	RV4NAYSD503A	50k
4979CH	53C3-100k	RV4NAYSD104A	100k
41672CH	53C3-150k	RV4NAYSD154A	150k
41681CH	53C3-250k	RV4NAYSD254A	250k
41699CH	53C3-350k	RV4NAYSD354A	350k
41701CH	53C3-500k	RV4NAYSD504A	500k
41710CH	53C3-750k	RV4NAYSD754A	750k
4728CH	53C3-1Meg	RV4NAYSD105A	1M
4736CH	53C3-2Meg	RV4NAYSD205A	2M
4942CH	53C3-3Meg	RV4NAYSD305A	3M

We're Listening to Your Requests

Email us 24/7 at
NewProducts@Jameco.com
with suggestions for products you'd like to see in future catalogs.

1/2-Watt 24mm Linear Taper (±20%)

255476CH

264372CH

255610CH

- Maximum voltage: 500V
- Rotational angle: 300° (±5°)
- Withstanding voltage for 1 minute: 500VAC

0.335"L shaft; solder lugs

Qty.	1	10	100	500
Price	$1.19	$.91	$.83	$.76

Part No.	Product No.	Ohms
255476CH	RV24A-10-15R1-B500	500
255484CH	RV24A-10-15R1-B1K	1k
255492CH	RV24A-10-15R1-B2K	2k
255505CH	RV21A-10-15R1-B2.5K	2.5k
255513CH	RV24A-10-15R1-B5K	5k
255521CH	RV24A-10-15R1-B10K	10k
255530CH	RV24A-10-15R1-B25K	25k
255548CH	RV24A-10-15R1-B50K	50k
255556CH	RV24A-10-15R1-B100K	100k
255564CH	RV24A-10-15R1-B250K	250k
255572CH	RV24A-10-15R1-B500K	500k
255581CH	RV24A-10-15R1-B1M	1M
255599CH	RV24A-10-15R1-B2M	2M
255601CH	RV24A-10-15R1-B5M	5M

0.531"L knurled shaft; solder lugs

Qty.	1	10	100	500
Price	$1.19	$.91	$.83	$.76

Part No.	Product No.	Ohms
264372CH	RV24A-10-20K-B500	500
264381CH	RV24A-10-20K-B1K	1k
264399CH	RV24A-10-20K-B2K	2k
264401CH	RV24A-10-20K-B5K	5k
264410CH	RV24A-10-20K-B10K	10k
264428CH	RV24A-10-20K-B50K	50k
264436CH	RV24A-10-20K-B100K	100k
264444CH	RV24A-10-20K-B250K	250k
264452CH	RV24A-10-20K-B500K	500k
264461CH	RV24A-10-20K-B1M	1M
264479CH	RV24A-10-20K-B2M	2M

1.32"L shaft; solder lugs

Qty.	1	10	100	500
Price	$1.13	$.95	$.86	$.79

Part No.	Product No.	Ohms
255610CH	RV24A-10-40R1-B500	500
255628CH	RV24A-10-40R1-B1K	1k
255636CH	RV24A-10-40R1-B2K	2k
255644CH	RV24A-10-40R1-B2.5K	2.5k
255652CH	RV24A-10-40R1-B5K	5k
255661CH	RV24A-10-40R1-B10K	10k
255679CH	RV24A-10-40R1-B25K	25k
255687CH	RV24A-10-40R1-B50K	50K
255695CH	RV24A-10-40R1-B100K	100k
255708CH	RV24A-10-40R1-B250K	250k
255716CH	RV24A-10-40R1-B500K	500k
255724CH	RV24A-10-40R1-B1M	1M
255732CH	RV24A-10-40R1-B5M	5M

1/2-Watt 24mm Linear Taper with SPST Switch (±20%)

- Shaft length: 0.335"
- Shaft diameter: .250"
- Terminal type: solder lugs
- Maximum voltage: 500V
- Switch rating: 125VAC, 1A
- Rotational angle: 300° (±5°)

263759CH

Qty.	1	10	100	500
Price	$1.59	$1.36	$1.24	$1.13

Part No.	Product No.	Ohms
263759CH	RV24A01-10-15R1-B500	500
263767CH	RV24A01-10-15R1-B1K	1k
263775CH	RV24A01-10-15R1-B2.5K	2.5k
263783CH	RV24A01-10-15R1-B5K	5k
263791CH	RV24A01-10-15R1-B10K	10k
263804CH	RV24A01-10-15R1-B25K	25k
263812CH	RV24A01-10-15R1-B50K	50k
263821CH	RV24A01-10-15R1-B100K	100k
263839CH	RV24A01-10-15R1-B250K	250k
263847CH	RV24A01-10-15R1-B500K	500k
263855CH	RV24A01-10-15R1-B1M	1M

1/4-Watt 24mm Audio Taper (±20%)

- Shaft length: 0.335"
- Maximum voltage: 250V
- Shaft diameter: .250"
- Terminal type: solder lugs
- Insulation resistance: >100MΩ @ 500VDC
- Rotational angle: 300° (±5°)

255396CH

Qty.	1	10	100	500
Price	$1.19	$.91	$.83	$.76

Part No.	Product No.	Ohms
255396CH	RV24A-10-15R1-A1K	1k
255417CH	RV24A-10-15R1-A5K	5k
255425CH	RV24A-10-15R1-A10K	10k
255433CH	RV24A-10-15R1-A50K	50k
255441CH	RV24A-10-15R1-A100K	100k
255450CH	RV24A-10-15R1-A500K	500k
255468CH	RV24A-10-15R1-A1M	1M

1/4-Watt 24mm Audio Taper with SPST Switch (±20%)

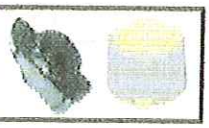
- Shaft length: 0.590"
- Voltage rating: 250V
- Shaft diameter: .250"
- Switch rating: 125VAC, 1A
- Rotational angle: 300° (±5°)

263601CH

Qty.	1	10	100	500
Price	$1.59	$1.36	$1.24	$1.13

Part No.	Product No.	Ohms
263601CH	RV24A01-10-15R1-A1K	1k
263610CH	RV24A01-10-15R1-A5K	5k
263636CH	RV24A01-10-15R1-A10K	10k
263644CH	RV24A01-10-15R1-A50K	50k
263652CH	RV24A01-10-15R1-A100K	100k
263661CH	RV24A01-10-15R1-A500K	500k
263679CH	RV24A01-10-15R1-A1M	1M

Instrumentation Knobs
See page 145



Color Code Legend

A4 DATASHEET DRIVER

TOSHIBA BIPOLAR LINEAR INTEGRATED CIRCUIT SILICON MONOLITHIC

TA7291P,TA7291S,TA7291F

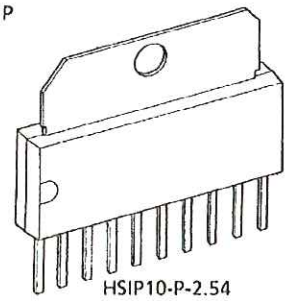
BRIDGE DRIVER

The TA7291P / S / F are Bridge Driver with output voltage control.

FEATURES

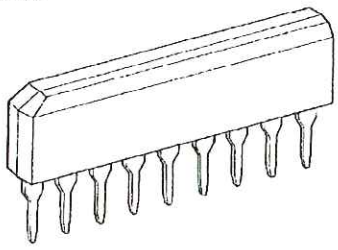
- 4 modes available (CW / CCW / STOP / BRAKE)
- Output current: P type 1.0 A (AVE.) 2.0 A (PEAK)
  S / F type 0.4 A (AVE.) 1.2 A (PEAK)
- Wide range of operating voltage: $V_{CC\ (opr.)} = 4.5 \sim 20$ V
  $V_{S\ (opr.)} = 0 \sim 20$ V
  $V_{ref\ (opr.)} = 0 \sim 20$ V
- Build in thermal shutdown, over current protector and punch = through current restriction circuit.
- Stand-by mode available (STOP MODE)
- Hysteresis for all inputs.

TA7291P



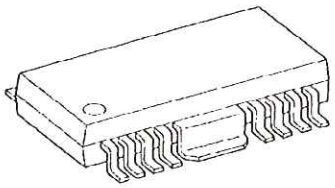
HSIP10-P-2.54

TA7291S



SIP9-P-2.54A

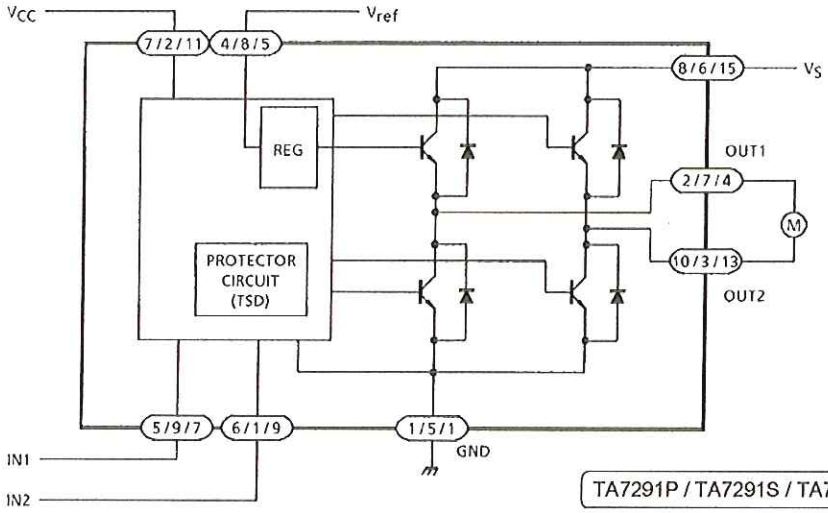
TA7291F



HSOP16-P-300-1.00

Weight
HSIP10-P-2.54 : 2.47 g (Typ.)
SIP9-P-2.54A : 0.92 g (Typ.)
HSOP16-P-300-1.00 : 0.50 g (Typ.)

BLOCK DIAGRAM



PIN FUNCTION

PIN No.			SYMBOL	FUNCTION DESCRIPTION
P	S	F		
7	2	11	V_{CC}	Supply voltage terminal for Logic
8	6	15	V_S	Supply voltage terminal for Motor driver
4	8	5	V_{ref}	Supply voltage terminal for control
1	5	1	GND	GND terminal
5	9	7	IN1	Input terminal
6	1	9	IN2	Input terminal
2	7	4	OUT1	Output terminal
10	3	13	OUT2	Output terminal

P Type: Pin (3), (9): NC
S Type: PIN (4): NC
F Type: PIN (2), (3), (6), (8), (10), (12), (14), and (16): NC
For F Type, We recommend FIN to be connected to the GND.

FUNCTION

INPUT		OUTPUT		MODE
IN1	IN2	OUT1	OUT2	
0	0	∞	∞	STOP
1	0	H	L	CW / CCW
0	1	L	H	CCW / CW
1	1	L	L	BRAKE

∞: High impedance
Note: Inputs are all high active type

MAXIMUM RATINGS (Ta = 25°C)

CHARACTERISTIC			SYMBOL	RATING	UNIT
Supply Voltage			V_{CC}	25	V
Motor Drive Voltage			V_S	25	V
Reference Voltage			V_{ref}	25	V
Output Current	PEAK	P Type	$I_{O\ (PEAK)}$	2.0	A
		S / F Type		1.2	
	AVE.	P Type	$I_{O\ (AVE.)}$	1.0	
		S / F Type		0.4	
Power Dissipation		P Type	P_D	12.5 (Note 1)	W
		S Type		0.95 (Note 2)	
		F Type		1.4 (Note 3)	
Operating Temperature			T_{opr}	−30~75	°C
Storage Temperature			T_{stg}	−55~150	°C

Note 1: Tc = 25°C (TA7291P)
Note 2: No heat sink
Note 3: PCB (60 × 30 × 1.6 mm, occupied copper area in excess of 50%) Mounting Condition.

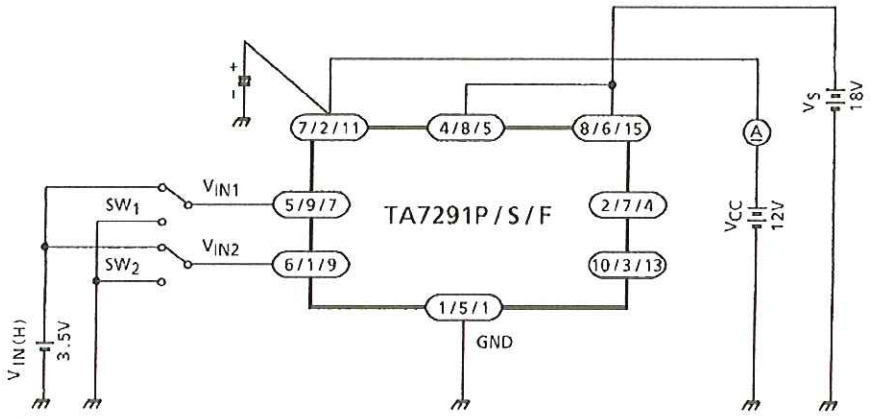
Wide range of operating voltage: $V_{CC\ (opr.)}$ = 4.5~20 V
$V_{S\ (opr.)}$ = 0~20 V
$V_{ref\ (opr.)}$ = 0~20 V
$V_{ref} \leq V_S$

ELECTRICAL CHARACTERISTICS
(Unless otherwise specified, Ta = 25°C, V_{CC} = 12 V, V_S = 18 V)

CHARACTERISTIC			SYMBOL	TEST CIR-CUIT	TEST CONDITION	MIN	TYP.	MAX	UNIT
Supply Current			I_{CC1}	1	Output OFF, CW / CCW mode	—	8.0	13.0	mA
			I_{CC2}		Output OFF, Stop mode	—	0	50	μA
			I_{CC3}		Output OFF, Brake mode	—	6.5	10.0	mA
Input Operating Voltage		1 (High)	V_{IN1}	2	T_j = 25°C	3.5	—	5.5	V
		2 (Low)	V_{IN2}			GND	—	0.8	
Input Current			I_{IN}		V_{IN} = 3.5 V, Sink mode	—	3	10	μA
Input Hysteresis Voltage			ΔV_T		—	—	0.7	—	V
Saturation Voltage	P / S / F Type	Upper Side	$V_{SAT\ U-1}$	3	V_{ref} = V_S, V_{OUT} – V_S measure I_O = 0.2 A, CW / CCW mode	—	0.9	1.2	V
		Lower Side	$V_{SAT\ L-1}$		V_{ref} = V_S, V_{OUT} – GND measure I_O = 0.2 A, CW / CCW mode	—	0.8	1.2	
	S / F Type	Upper Side	$V_{SAT\ U-2}$		V_{ref} = V_S, V_{OUT} – V_S measure I_O = 0.4 A, CW / CCW mode	—	1.0	1.35	
		Lower Side	$V_{SAT\ L-2}$		V_{ref} = V_S, V_{OUT} – GND measure I_O = 0.4 A, CW / CCW mode	—	0.9	1.35	
	P Type	Upper Side	$V_{SAT\ U-3}$		V_{ref} = V_S, V_{OUT} – V_S measure I_O = 1.0 A, CW / CCW mode	—	1.3	1.8	
		Lower Side	$V_{SAT\ L-3}$		V_{ref} = V_S, V_{OUT} – GND measure I_O = 1.0 A, CW / CCW mode	—	1.2	1.85	
Output Voltage (Upper Side)	S / F Type		$V_{SAT\ U-1'}$	3	V_{ref} = 10 V V_{OUT} – GND measure, I_O = 0.2 A, CW / CCW mode	—	11.2	—	V
			$V_{SAT\ U-2'}$		V_{ref} = 10 V V_{OUT} – GND measure, I_O = 0.4 A, CW / CCW mode	10.4	10.9	12.2	
	P Type		$V_{SAT\ U-3'}$		V_{ref} = 10 V V_{OUT} – GND measure, I_O = 0.5 A, CW / CCW mode	—	11.0	—	
			$V_{SAT\ U-4'}$		V_{ref} = 10 V V_{OUT} – GND measure, I_O = 1.0 A, CW / CCW mode	10.2	10.7	12.0	
Leakage Current		Upper Side	$I_{L\ U}$	4	V_L = 25 V	—	—	50	μA
		Lower Side	$I_{L\ L}$		V_L = 25 V	—	—	50	
Diode Forward Voltage	S / F Type	Upper Side	$V_{F\ U-1}$	5	I_F = 0.4 A	—	1.5	—	V
	P Type	Lower Side	$V_{F\ U-2}$		I_F = 1 A	—	2.5	—	
	S / F Type	Upper Side	$V_{F\ L-1}$		I_F = 0.4 A	—	0.9	—	
	P Type	Lower Side	$V_{F\ L-2}$		I_F = 1 A	—	1.2	—	
Reference Current			I_{ref}	2	V_{ref} = 10 V, Source mode	—	20	40	μA

TEST CIRCUIT 1

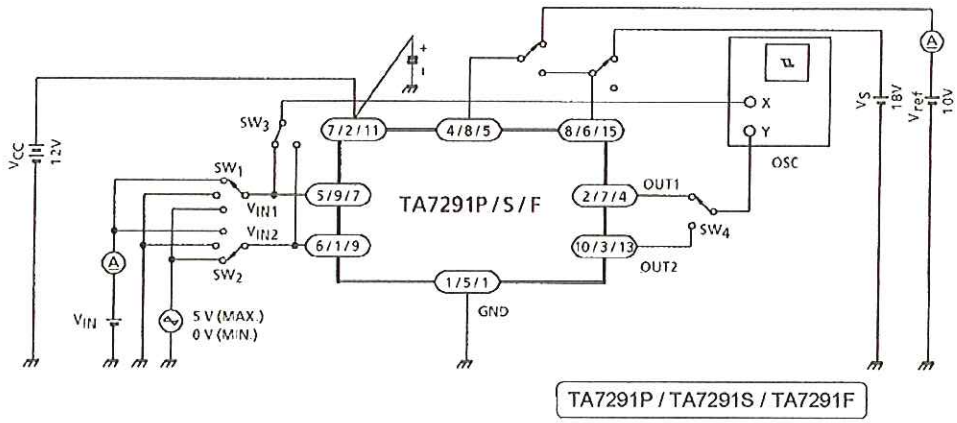
I_{CC1}, I_{CC2}, I_{CC3}



Note: HEAT FIN of TA7291F is connected to GND.

TEST CIRCUIT 2

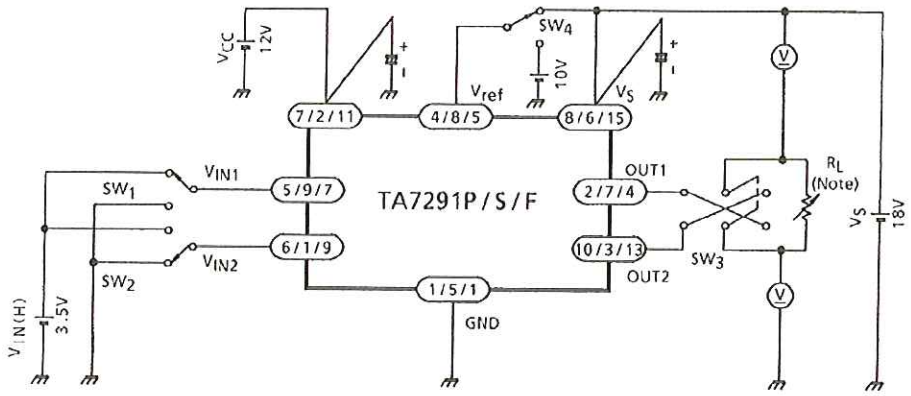
$V_{IN\,1}$, $V_{IN\,2}$, I_{IN}, ΔV_T, I_{ref}



Note: HEAT FIN of TA7291F is connected to GND.

TEST CIRCUIT 3

$V_{SAT\ U-1, 2, 3}$ $V_{SAT\ L-1, 2, 3}$ $V_{SAT\ U-1', 2', 3', 4'}$

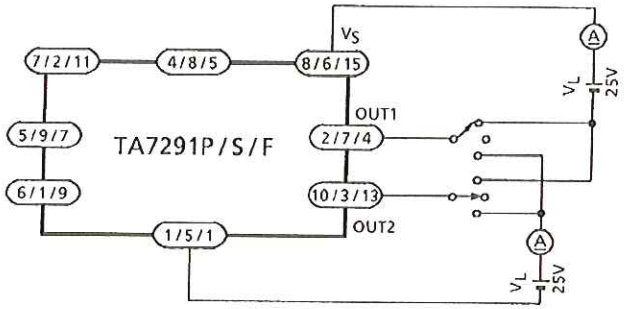


Note: I_{OUT} calibration is required to adjust specified values of test conditions by R_L.
(I_{OUT} = 0.2 A / 0.4 A / 0.5 A / 1.0 A)

Note: HEAT FIN of TA7291F is connected to GND.

TEST CIRCUIT 4

$I_{L\ U, L}$

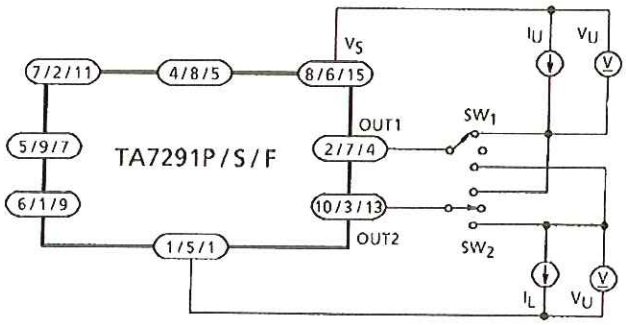


TA7291P / TA7291S / TA7291F

Note: HEAT FIN of TA7291F is connected to GND.

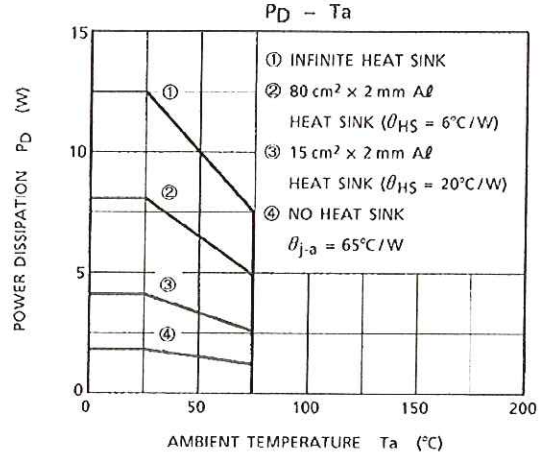
TEST CIRCUIT 5

$V_{F\ U-1, 2}$ $V_{F\ L-1, 2}$



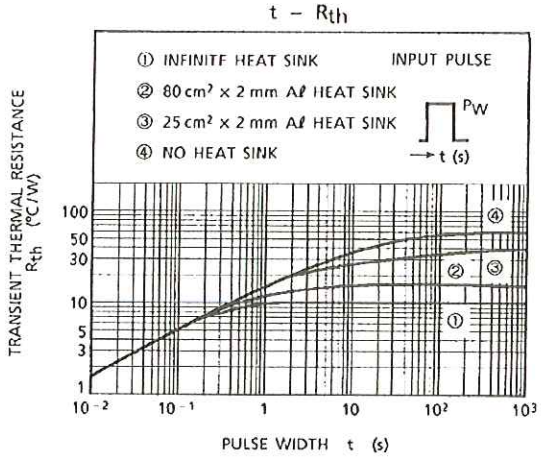
TA7291P

P_D – Ta



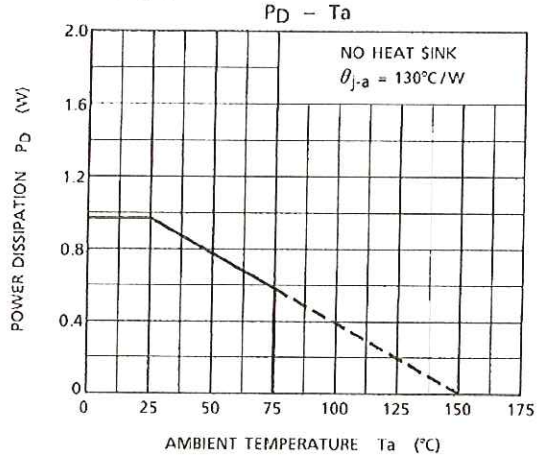
TA7291P

t – R_{th}



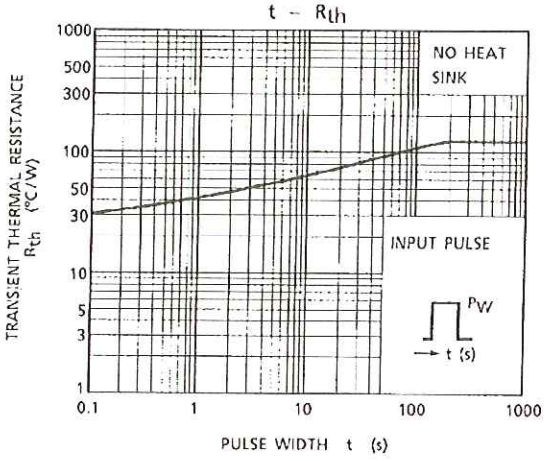
TA7291S

P_D – Ta



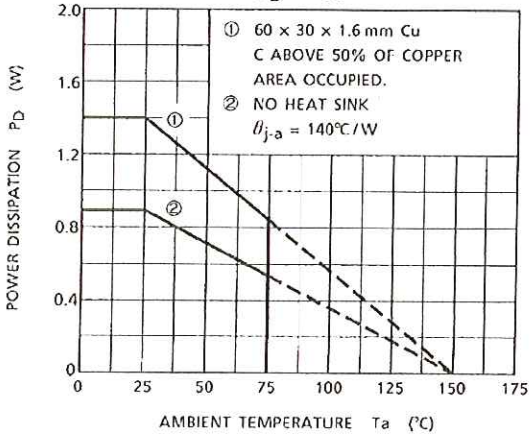
TA7291S

t – R_{th}



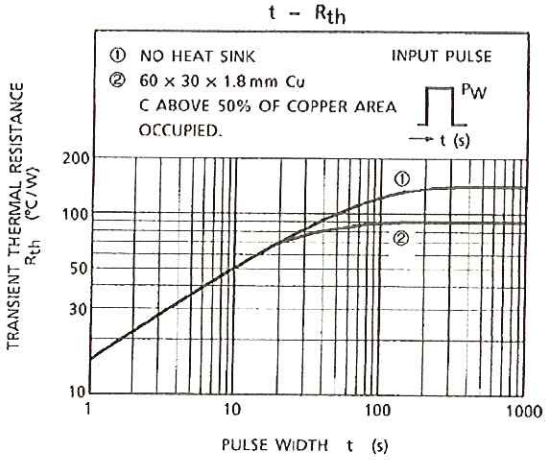
TA7291F

P_D – Ta



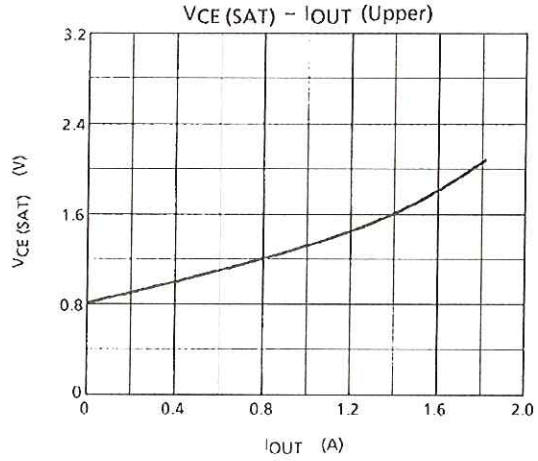
TA7291F

t – R_{th}



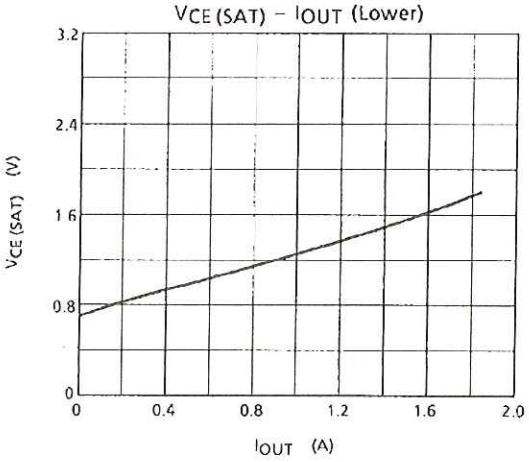
TA7291P

$V_{CE\,(SAT)}$ – I_{OUT} (Upper)

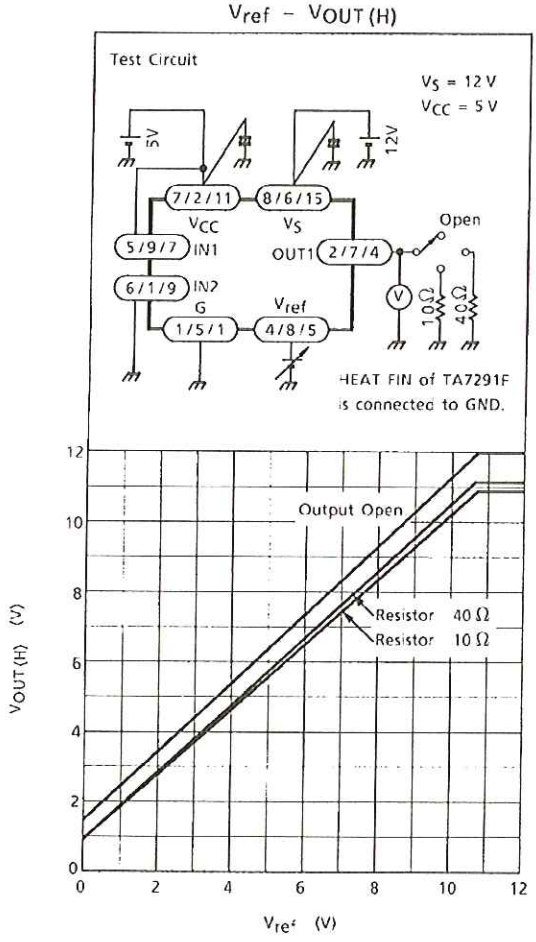


TA7291P

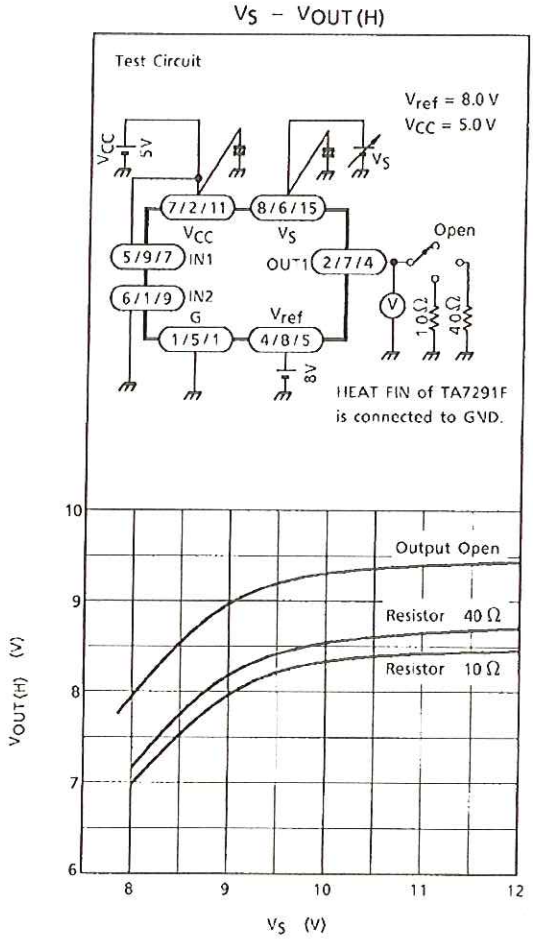
$V_{CE\,(SAT)}$ – I_{OUT} (Lower)



V_{ref} – $V_{OUT\,(H)}$



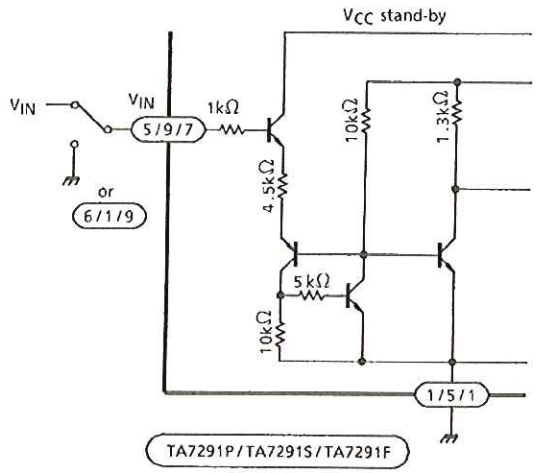
V_S – $V_{OUT\,(H)}$



NOTES

Input circuit

Input Terminals of pin (5) and (6) (TA7291P) are all high active type and have a hysteresis of 0.7 V (typ.), 3 μA (typ.) of source mode input current is required.



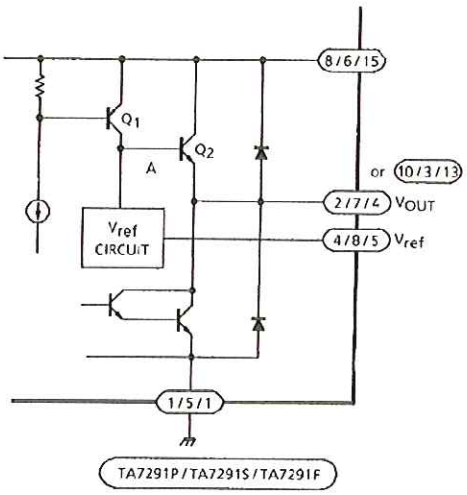
Output circuit

Output voltage is controlled by V_{ref} voltage.
Relationship between V_{OUT} and V_{ref} is

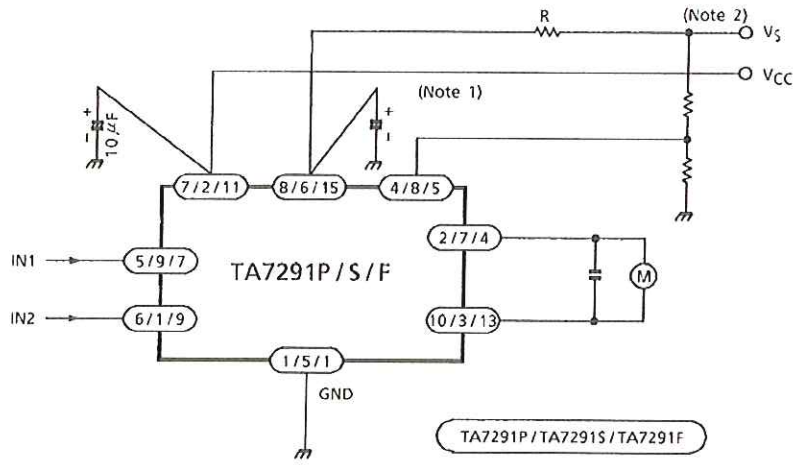
$V_{OUT} = V_{BE} (\approx 0.7) + V_{ref}$

V_{ref} terminal required to connect to V_S terminal for stable operation in case of no requirement of V_{OUT} control.

$V_{ref} \leq V_S$



APPLICATION CIRCUIT



Note 1: Experiment to find the optimum capacitor valve.
Note 2: To protect against excess current, current limitation resistor R should be inserted where necessary.

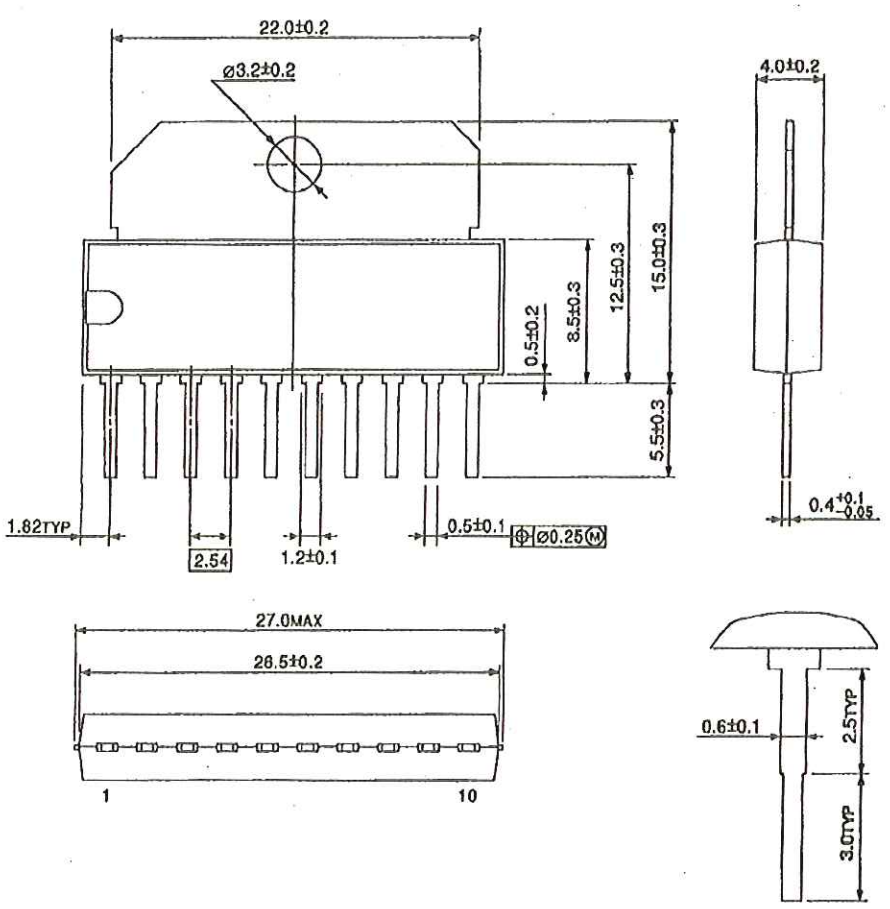
NOTES

- Be careful when switching the input because rush current may occur.
  When switching, stop mode should be entered or current limitation resister R should be inserted.
- The IC functions cannot be guaranteed when turning power on of off.
  Before using the IC for application, check that there are no problems.
- Utmost care is necessary in the design of the output line, V_S, V_{CC} and GND line since IC may be destroyed due to short−circuit between outputs, air contamination fault, or fault by improper grounding.

PACKAGE DIMENSIONS

HSIP10-P-2.54

Unit: mm

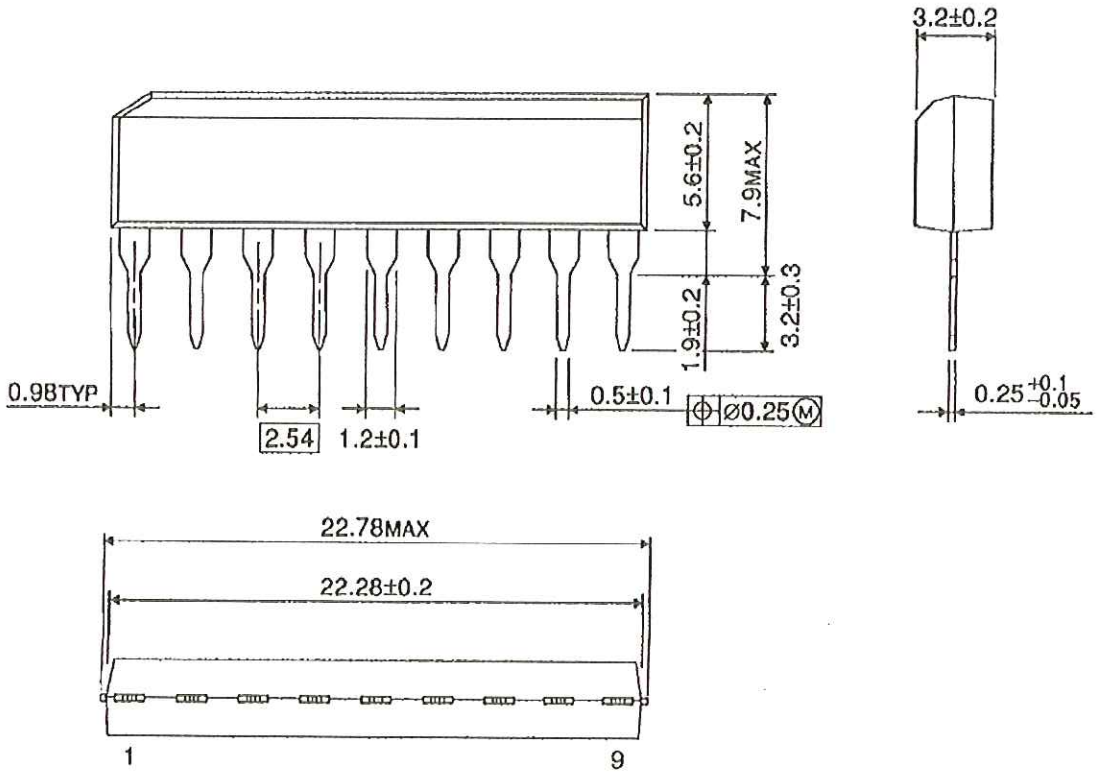


Weight: 2.47 g (Typ.)

PACKAGE DIMENSIONS

SIP9-P-2.54A

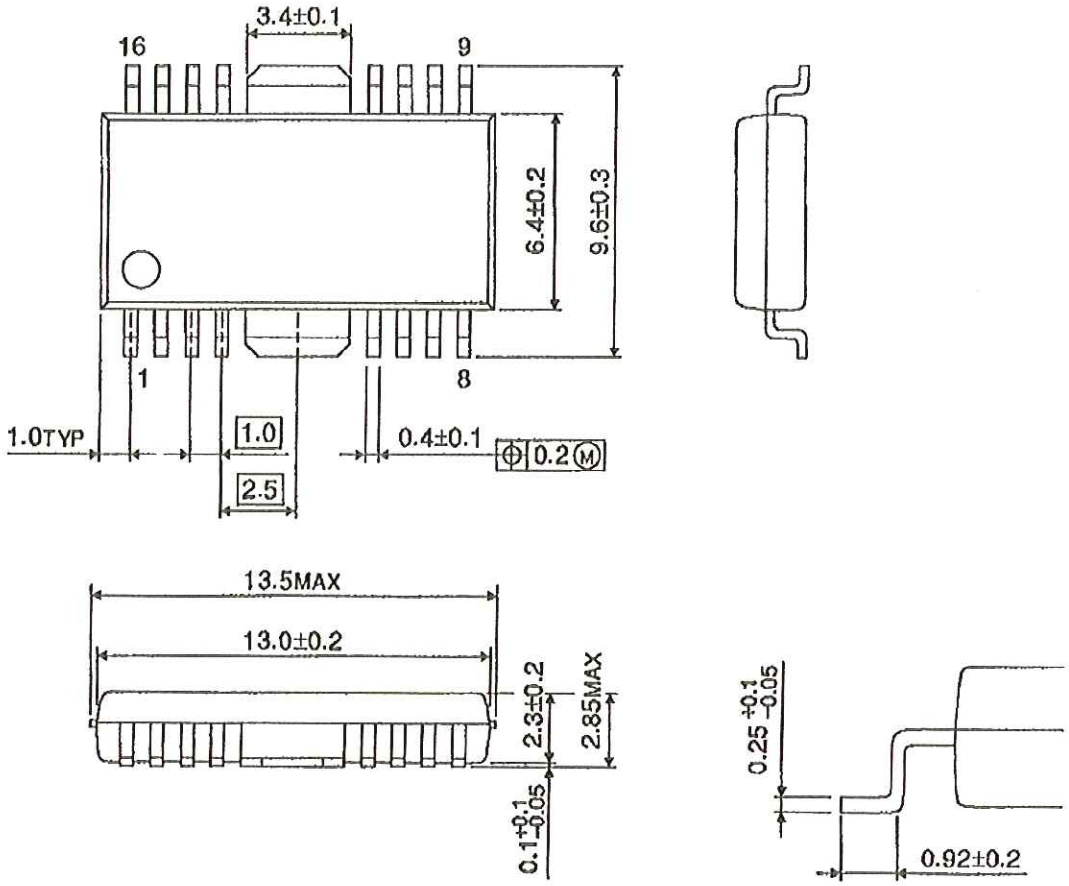
Unit: mm



Weight: 0.92 g (Typ.)

PACKAGE DIMENSIONS

HSOP16-P-300-1.00 Unit: mm



Weight: 0.50 g (Typ.)

RESTRICTIONS ON PRODUCT USE

000707EBA

- TOSHIBA is continually working to improve the quality and reliability of its products. Nevertheless, semiconductor devices in general can malfunction or fail due to their inherent electrical sensitivity and vulnerability to physical stress. It is the responsibility of the buyer, when utilizing TOSHIBA products, to comply with the standards of safety in making a safe design for the entire system, and to avoid situations in which a malfunction or failure of such TOSHIBA products could cause loss of human life, bodily injury or damage to property.
  In developing your designs, please ensure that TOSHIBA products are used within specified operating ranges as set forth in the most recent TOSHIBA products specifications. Also, please keep in mind the precautions and conditions set forth in the "Handling Guide for Semiconductor Devices," or "TOSHIBA Semiconductor Reliability Handbook" etc..
- The TOSHIBA products listed in this document are intended for usage in general electronics applications (computer, personal equipment, office equipment, measuring equipment, industrial robotics, domestic appliances, etc.). These TOSHIBA products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury ("Unintended Usage"). Unintended Usage include atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, medical instruments, all types of safety devices, etc.. Unintended Usage of TOSHIBA products listed in this document shall be made at the customer's own risk.
- The products described in this document are subject to the foreign exchange and foreign trade laws.
- The information contained herein is presented only as a guide for the applications of our products. No responsibility is assumed by TOSHIBA CORPORATION for any infringements of intellectual property or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any intellectual property or other rights of TOSHIBA CORPORATION or others.
- The information contained herein is subject to change without notice.

A5 DATASHEET 74LS138

SN74LS138

1-of-8 Decoder/ Demultiplexer

The LSTTL/MSI SN74LS138 is a high speed 1-of-8 Decoder/ Demultiplexer. This device is ideally suited for high speed bipolar memory chip select address decoding. The multiple input enables allow parallel expansion to a 1-of-24 decoder using just three LS138 devices or to a 1-of-32 decoder using four LS138s and one inverter. The LS138 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all ON Semiconductor TTL families.

- Demultiplexing Capability
- Multiple Input Enable for Easy Expansion
- Typical Power Dissipation of 32 mW
- Active Low Mutually Exclusive Outputs
- Input Clamp Diodes Limit High Speed Termination Effects

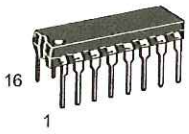
GUARANTEED OPERATING RANGES

Symbol	Parameter	Min	Typ	Max	Unit
V_{CC}	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current – High			–0.4	mA
I_{OL}	Output Current – Low			8.0	mA

ON Semiconductor

http://onsemi.com

LOW POWER SCHOTTKY



PLASTIC
N SUFFIX
CASE 648

SOIC
D SUFFIX
CASE 751B

SOEIAJ
M SUFFIX
CASE 966

ORDERING INFORMATION

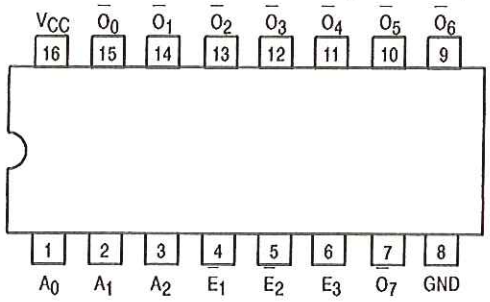
Device	Package	Shipping
SN74LS138N	16 Pin DIP	2000 Units/Box
SN74LS138D	SOIC–16	38 Units/Rail
SN74LS138DR2	SOIC–16	2500/Tape & Reel
SN74LS138M	SOEIAJ–16	See Note 1
SN74LS138MEL	SOEIAJ–16	See Note 1

1. For ordering information on the EIAJ version of the SOIC package, please contact your local ON Semiconductor representative.

October, 2001 – Rev. 7

Publication Order Number:
SN74LS138/D

CONNECTION DIAGRAM DIP (TOP VIEW)

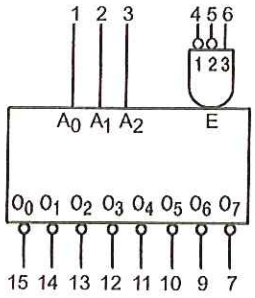


NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

PIN NAMES		LOADING (Note a) HIGH	LOW
$\overline{A_0} - \overline{A_2}$	Address Inputs	0.5 U.L.	0.25 U.L.
$E_1, \overline{E_2}$	Enable (Active LOW) Inputs	0.5 U.L.	0.25 U.L.
E_3	Enable (Active HIGH) Input	0.5 U.L.	0.25 U.L.
$\overline{O_0} - \overline{O_7}$	Active LOW Outputs	10 U.L.	5 U.L.

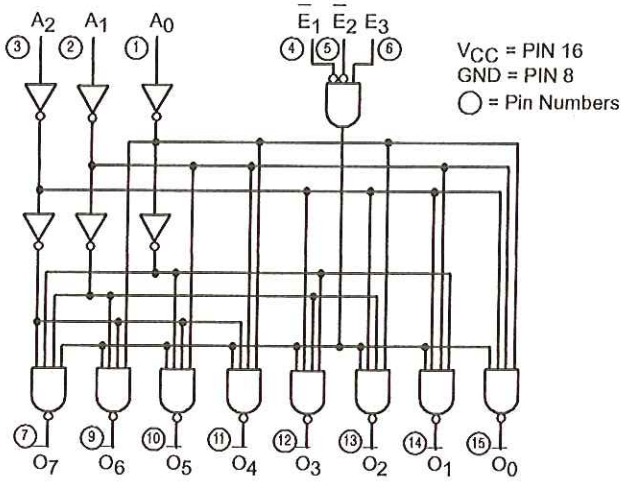
NOTES:
a) 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.

LOGIC SYMBOL



V_{CC} = PIN 16
GND = PIN 8

LOGIC DIAGRAM



FUNCTIONAL DESCRIPTION

The LS138 is a high speed 1-of-8 Decoder/Demultiplexer fabricated with the low power Schottky barrier diode process. The decoder accepts three binary weighted inputs (A_0, A_1, A_2) and when enabled provides eight mutually exclusive active LOW Outputs ($\overline{O_0}$–$\overline{O_7}$). The LS138 features three Enable inputs, two active LOW ($\overline{E_1}$, $\overline{E_2}$) and one active HIGH (E_3). All outputs will be HIGH unless $\overline{E_1}$ and $\overline{E_2}$ are LOW and E_3 is HIGH. This multiple enable function allows easy parallel expansion of the device to a 1-of-32 (5 lines to 32 lines) decoder with just four LS138s and one inverter. (See Figure a.)

The LS138 can be used as an 8-output demultiplexer by using one of the active LOW Enable inputs as the data input and the other Enable inputs as strobes. The Enable inputs which are not used must be permanently tied to their appropriate active HIGH or active LOW state.

TRUTH TABLE

INPUTS						OUTPUTS							
$\overline{E_1}$	$\overline{E_2}$	E_3	A_0	A_1	A_2	$\overline{O_0}$	$\overline{O_1}$	$\overline{O_2}$	$\overline{O_3}$	$\overline{O_4}$	$\overline{O_5}$	$\overline{O_6}$	$\overline{O_7}$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

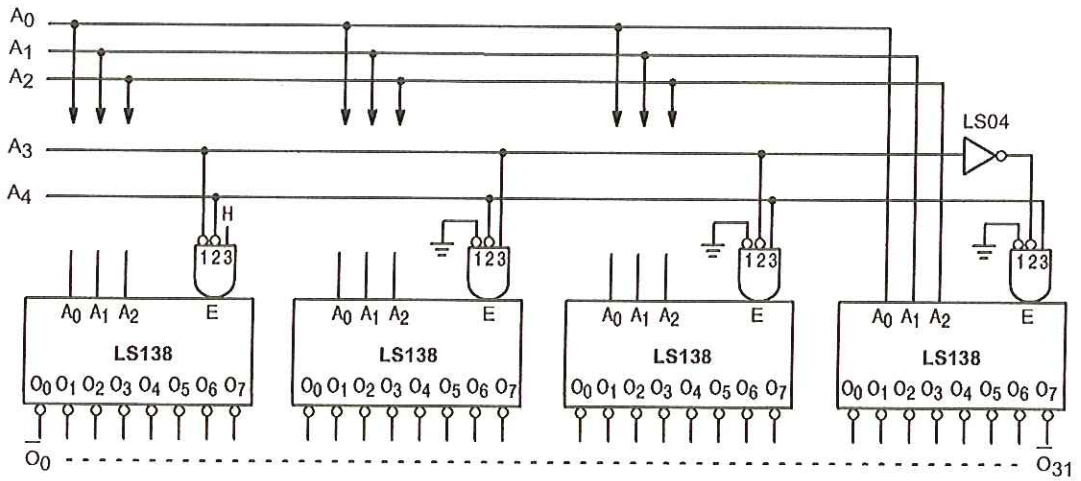


Figure a

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions	
		Min	Typ	Max			
V_{IH}	Input HIGH Voltage	2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage			0.8	V	Guaranteed Input LOW Voltage for All Inputs	
V_{IK}	Input Clamp Diode Voltage		−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	2.7	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table	
V_{OL}	Output LOW Voltage		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
			0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current			20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
				0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current			−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Short Circuit Current (Note 2)	−20		−100	mA	V_{CC} = MAX	
I_{CC}	Power Supply Current			10	mA	V_{CC} = MAX	

2. Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25°C)

Symbol	Parameter	Levels of Delay	Limits			Unit	Test Conditions
			Min	Typ	Max		
t_{PLH} t_{PHL}	Propagation Delay Address to Output	2 2		13 27	20 41	ns	V_{CC} = 5.0 V C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay Address to Output	3 3		18 26	27 39	ns	
t_{PLH} t_{PHL}	Propagation Delay E_1 or E_2 Enable to Output	2 2		12 21	18 32	ns	
t_{PLH} t_{PHL}	Propagation Delay E_3 Enable to Output	3 3		17 25	26 38	ns	

AC WAVEFORMS

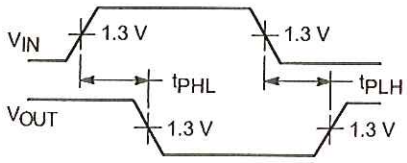


Figure 1.

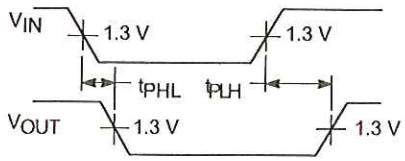
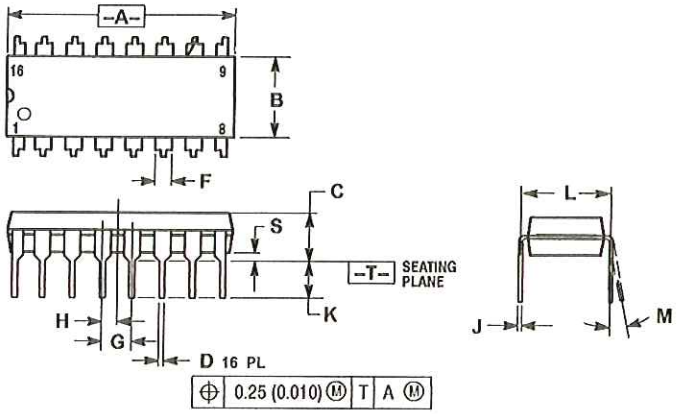


Figure 2.

PACKAGE DIMENSIONS

N SUFFIX
PLASTIC PACKAGE
CASE 648–08
ISSUE R

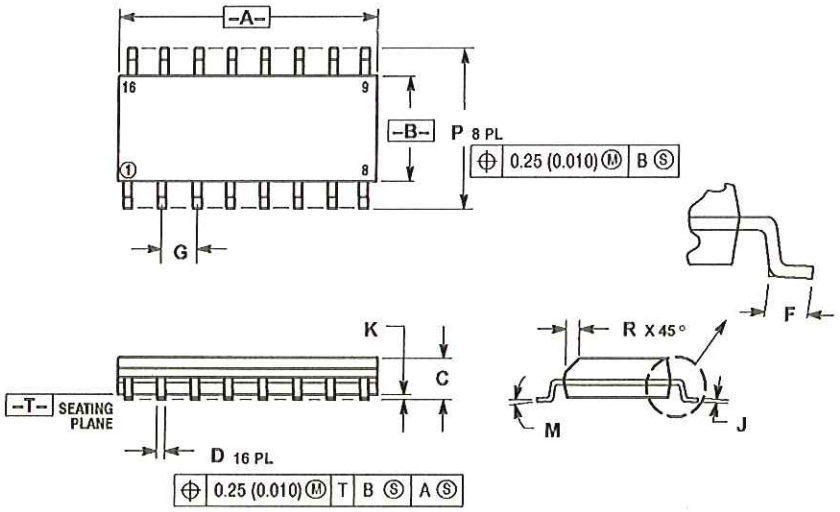


NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
4. DIMENSION B DOES NOT INCLUDE MOLD FLASH.
5. ROUNDED CORNERS OPTIONAL.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.740	0.770	18.80	19.55
B	0.250	0.270	6.35	6.85
C	0.145	0.175	3.69	4.44
D	0.015	0.021	0.39	0.53
F	0.040	0.70	1.02	1.77
G	0.100 BSC		2.54 BSC	
H	0.050 BSC		1.27 BSC	
J	0.008	0.015	0.21	0.38
K	0.110	0.130	2.80	3.30
L	0.295	0.305	7.50	7.74
M	0°	10°	0°	10°
S	0.020	0.040	0.51	1.01

PACKAGE DIMENSIONS

D SUFFIX
PLASTIC SOIC PACKAGE
CASE 751B–05
ISSUE J

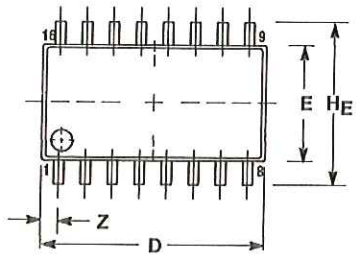


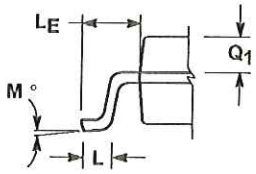
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 (0.006) PER SIDE.
5. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.127 (0.005) TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	9.80	10.00	0.386	0.393
B	3.80	4.00	0.150	0.157
C	1.35	1.75	0.054	0.068
D	0.35	0.49	0.014	0.019
F	0.40	1.25	0.016	0.049
G	1.27 BSC		0.050 BSC	
J	0.19	0.25	0.008	0.009
K	0.10	0.25	0.004	0.009
M	0°	7°	0°	7°
P	5.80	6.20	0.229	0.244
R	0.25	0.50	0.010	0.019

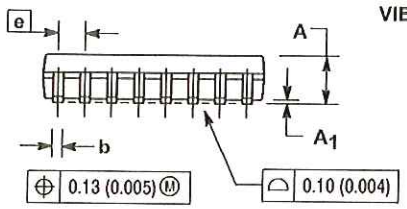
PACKAGE DIMENSIONS

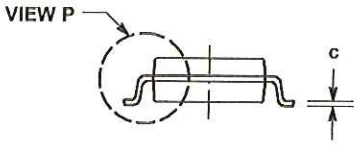
M SUFFIX
SOEIAJ PACKAGE
CASE 966–01
ISSUE O





DETAIL P





NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DIMENSIONS D AND E DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS AND ARE MEASURED AT THE PARTING LINE. MOLD FLASH OR PROTRUSIONS SHALL NOT EXCEED 0.15 (0.006) PER SIDE.
4. TERMINAL NUMBERS ARE SHOWN FOR REFERENCE ONLY.
5. THE LEAD WIDTH DIMENSION (b) DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) TOTAL IN EXCESS OF THE LEAD WIDTH DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSIONS AND ADJACENT LEAD TO BE 0.46 (0.018).

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	---	2.05	---	0.081
A_1	0.05	0.20	0.002	0.008
b	0.35	0.50	0.014	0.020
c	0.18	0.27	0.007	0.011
D	9.90	10.50	0.390	0.413
E	5.10	5.45	0.201	0.215
e	1.27 BSC		0.050 BSC	
H_E	7.40	8.20	0.291	0.323
L	0.50	0.85	0.020	0.033
L_E	1.10	1.50	0.043	0.059
M	0 °	10 °	0 °	10 °
Q_1	0.70	0.90	0.028	0.035
Z	---	0.78	---	0.031

PUBLICATION ORDERING INFORMATION

Literature Fulfillment:
Literature Distribution Center for ON Semiconductor
P.O. Box 5163, Denver, Colorado 80217 USA
Phone: 303–675–2175 or 800–344–3860 Toll Free USA/Canada
Fax: 303–675–2176 or 800–344–3867 Toll Free USA/Canada
Email: ONlit@hibbertco.com

N. American Technical Support: 800–282–9855 Toll Free USA/Canada

JAPAN: ON Semiconductor, Japan Customer Focus Center
4–32–1 Nishi–Gotanda, Shinagawa–ku, Tokyo, Japan 141–0031
Phone: 81–3–5740–2700
Email: r14525@onsemi.com

ON Semiconductor Website: http://onsemi.com

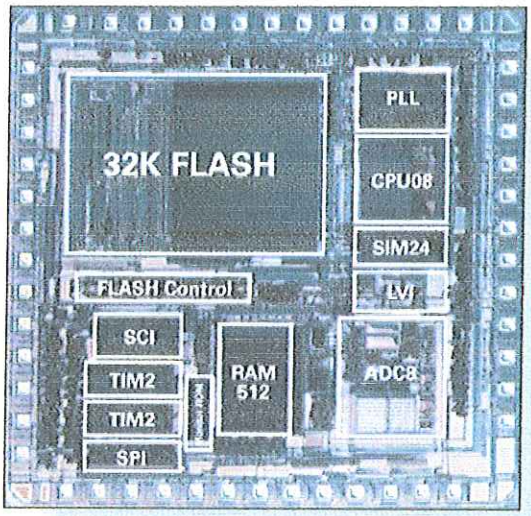
For additional information, please contact your local Sales Representative.

SN74LS138/D

A6 DATASHEET MICROCONTROLADOR

68HC908GP32

The Motorola 68HC908GP32 provides designers with a highly integrated 8-bit FLASH microcontroller (MCU) solution. The 68HC908GP32 builds on the success of the 68HC05 family by offering a code compatible migration path to higher performance FLASH MCUs.



Features

- 32,256 bytes of in-system programmable FLASH memory
- FLASH-wire technology – a single wire interface for in-circuit programming which does not require high voltage for entry
- 10,000 program/erase cycles
- FLASH programming as fast as 2 msec for a 64 byte block
- FLASH memory security features
- 512 bytes of user RAM
- High-performance 68HC08 CPU core
  - Code compatible with 68HC05
  - 8.0 MHz internal operating frequency at 5.0 V
- Peripheral modules
  - Computer Operating Properly (COP) watchdog
  - SCI asynchronous serial communications port
    - Full duplex operation
    - 32 programmable baud rates
    - Interrupt driven operation
    - 8-bit or 9-bit character length
  - SPI synchronous serial communications port
    - Full duplex operation with master and slave modes
    - Up to 4 MHz master, and 8 MHz slave mode frequencies
  - 8-channel 8-bit analog-to-digital-converter
  - Dual 16-bit two-channel timers with input capture, output compare, and PWM modes
  - Timebase module with eight user selectable periodic real-time interrupts
    - Auto wake-up out of stop capability
- Memory-mapped I/O registers
- 33 bi-directional input/output (I/O) lines, including:
  - 10 mA sink/source capability on all I/O pins
  - 15 mA sink capability on five I/O pins
  - Software programmable pullups on all I/O pins
  - Keyboard scan with selectable interrupts on eight I/O pins
- Internal pullups to V_{DD} on RESET and IRQ pins for reduced system cost
- Vectored interrupts
  - Selectable sensitivity on external interrupt (edge- and level-sensitive or edge-sensitive only)
  - External interrupt mask bit and acknowledge bit
- Illegal address reset

Digital DNA™ from Motorola

68HC908GP32

- Illegal opcode reset
- Low Voltage Inhibit with selectable trip points
- Clock options
  - 32 KHz crystal compatible oscillator and on-chip PLL
  - External clock
- Bi-directional RESET pin
- Power-saving Stop and Wait modes
- 40-pin DIP, 42-pin SDIP, and 44-pin QFP packages
- Pin compatible with the 68HC908GP20
- V_{DD}/V_{SS} pins adjacent for easy bypass capacitor connection
- Hyper-text linked on-line databook:
  - MC68HC908GP32/H
- Cost effective, full-featured development tools that support programming, in-circuit debug, simulation, and in-circuit emulation

Application Notes

- AN-HK-32/H In-circuit Programming of FLASH Memory in the 68HC908GP32
- AN-HK-31/H Using the MC68HC908GP32 in place of MC68HC908GP20
- AN1222/D Arithmetic Waveform Synthesis with 68HC05/68HC08 MCUs
- AN1221/D Hamming Error Control Coding Techniques with the HC08 MCU
- AN1219/D M68HC08 Integer Math Routines
- AN1218/D HC05 to HC08 Optimization
- More MCU application notes available on our website

Comprehensive Development Support

Broad third party software and hardware support – see our web site at http://www.mcu.motsps.com

EASY TO ORDER KITS		RESALE*
M68ICS08GP	GPxx programmer/in-circuit debug kit	$295
KITMMEVS08GP	Cost effective real-time in-circuit emulator kit	$1450
KITMMDS08GP	High performance real-time in-circuit emulator kit	$3950

INDIVIDUAL DEVELOPMENT TOOL COMPONENTS		RESALE*
M68MMDS0508	High performance emulator	$2950
M68MMPFB0508	MMEVS Platform Board	$395
M68EML08GP32	Emulation module daughter board	$495
M68CBL05B	Low noise flex-cable	$120
M68CBL05C	Low noise flex-cable	$120
M68TB08GP32P40	40-pin DIP target head adapter	$175
M68TB08GP32B42	42-pin SDIP target head adapter	$175
M68TC08GP32FB44	44-pin QFP target head adapter	$175
M68TQS044SAG1	44-pin TQ socket with guides	$50
M68TQP044SAM01	44-pin TQPACK	$70

*All prices are manufacturer's suggested resale for North America.

This product incorporates SuperFlash® technology licensed from SST.

Digital DNA™ from Motorola

Appendix A. MC68HC08GP32

A.1 Contents

A.2 Introduction

This section introduces the MC68HC08GP32, the ROM part equivalent to the MC68HC908GP32. The entire data book apply to this ROM device, with exceptions outlined in this appendix.

Table A-1. Summary of MC68HC08GP32 and MC68HC908GP32 differences

	MC68HC08GP32	MC68HC908GP32
Memory ($8000–$FDFF)	32,256 bytes ROM	32,256 bytes FLASH
User vectors ($FFDC–$FFFF)	36 bytes ROM	36 bytes FLASH
Registers at $001E and $001F	Mask option registers; defined by mask; read only. $001E — MOR2 $001F — MOR1	Configuration registers; write once after reset. $001E — CONFIG2 $001F — CONFIG1
Registers at $FE08 and $FF7E	Not used; locations are reserved	FLASH related registers. $FE08 — FLCR $FF7E — FLBPR
Bit 2 at $FE01	Not used; bit is reserved	MODRST: monitor mode entry by blank reset vector bit.
Monitor ROM ($FE20–$FF52)	Used for testing purposes only.	Used for testing and FLASH programming/erasing.
Available Packages	42-pin SDIP 44-pin QFP	40-pin PDIP 42-pin SDIP 44-pin QFP

A.3 MCU Block Diagram

Figure A-1 shows the block diagram of the MC68HC08GP32.

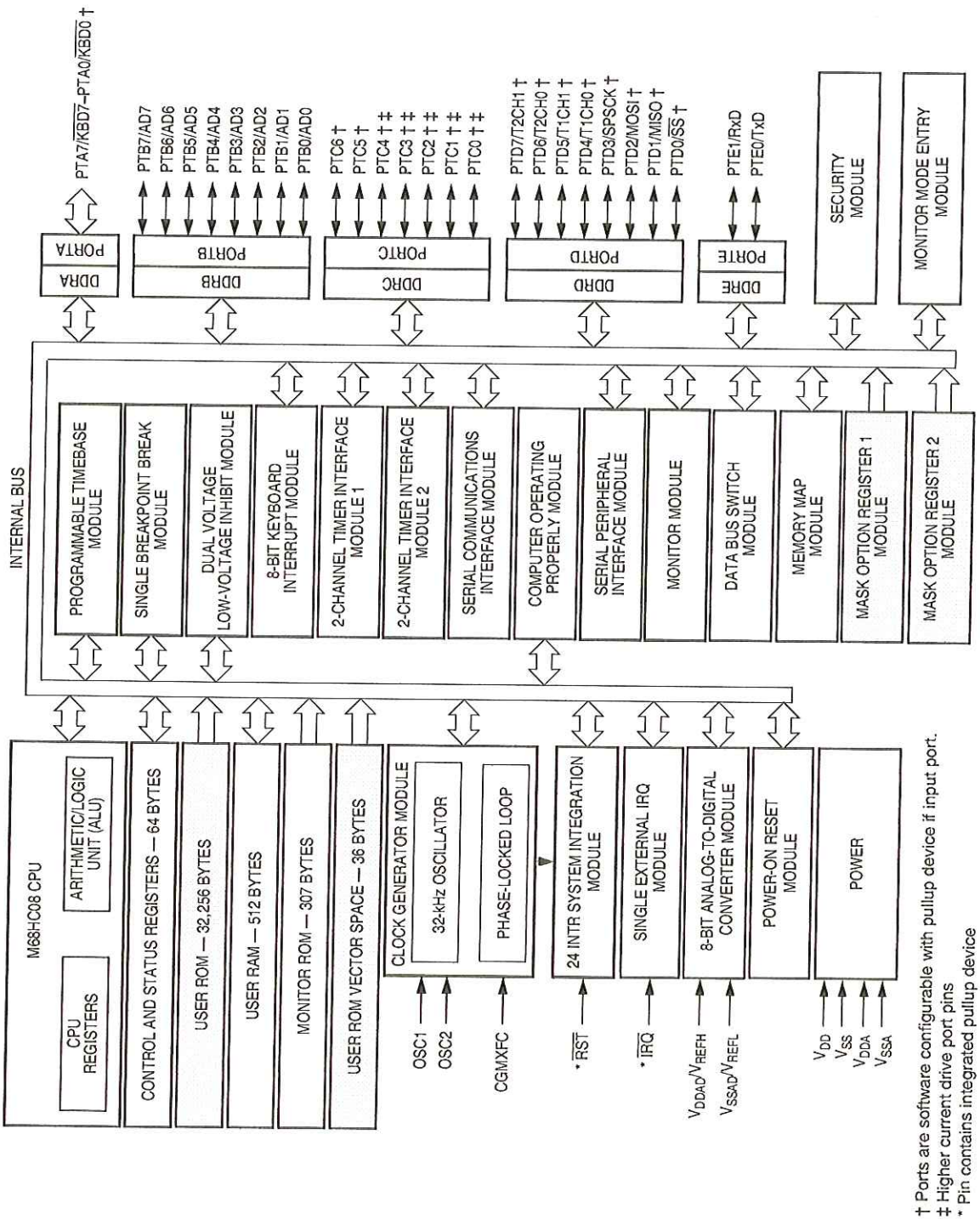


Figure A-1. MC68HC08GP32 Block Diagram

A.4 Memory Map

The MC68HC08GP32 has 32,256 bytes of user ROM from $8000 to $FDFF, and 36 bytes of user ROM vectors from $FFDC to $FFFF. On the MC68HC908GP32, these memory locations are FLASH memory.

Figure A-2 shows the memory map of the MC68HC08GP32.

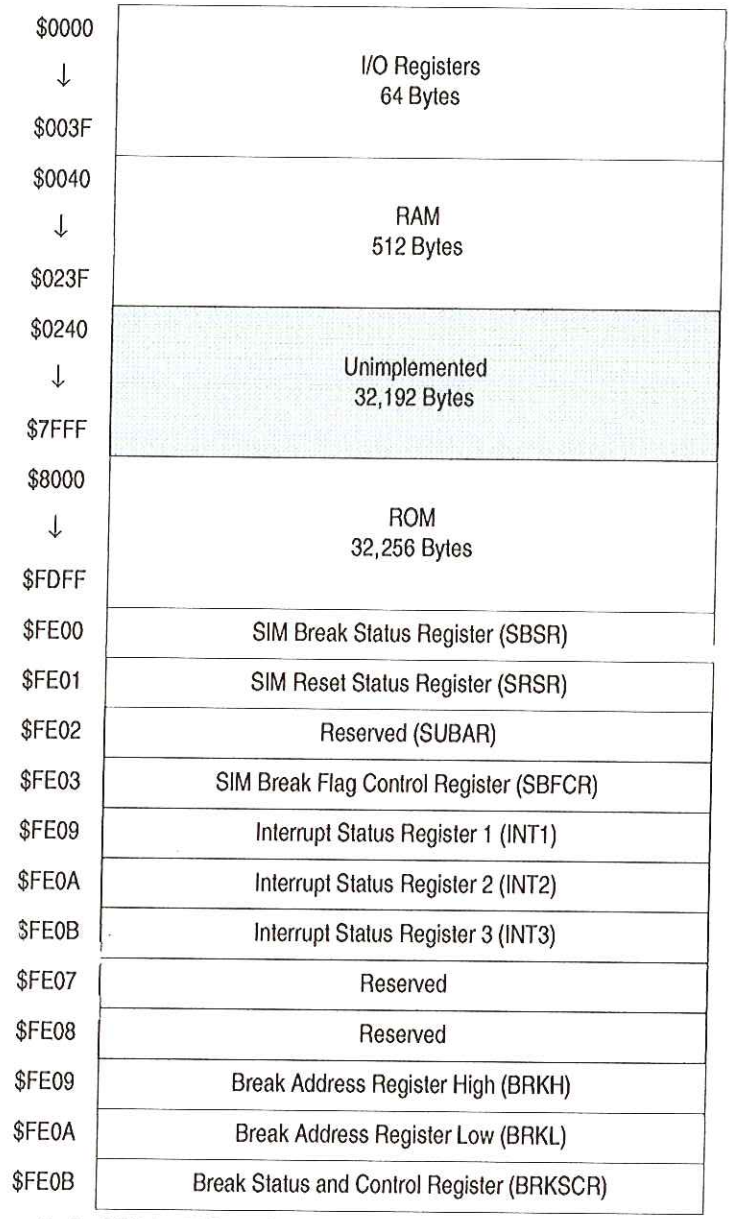


Figure A-2. MC68HC08GP32 Memory Map

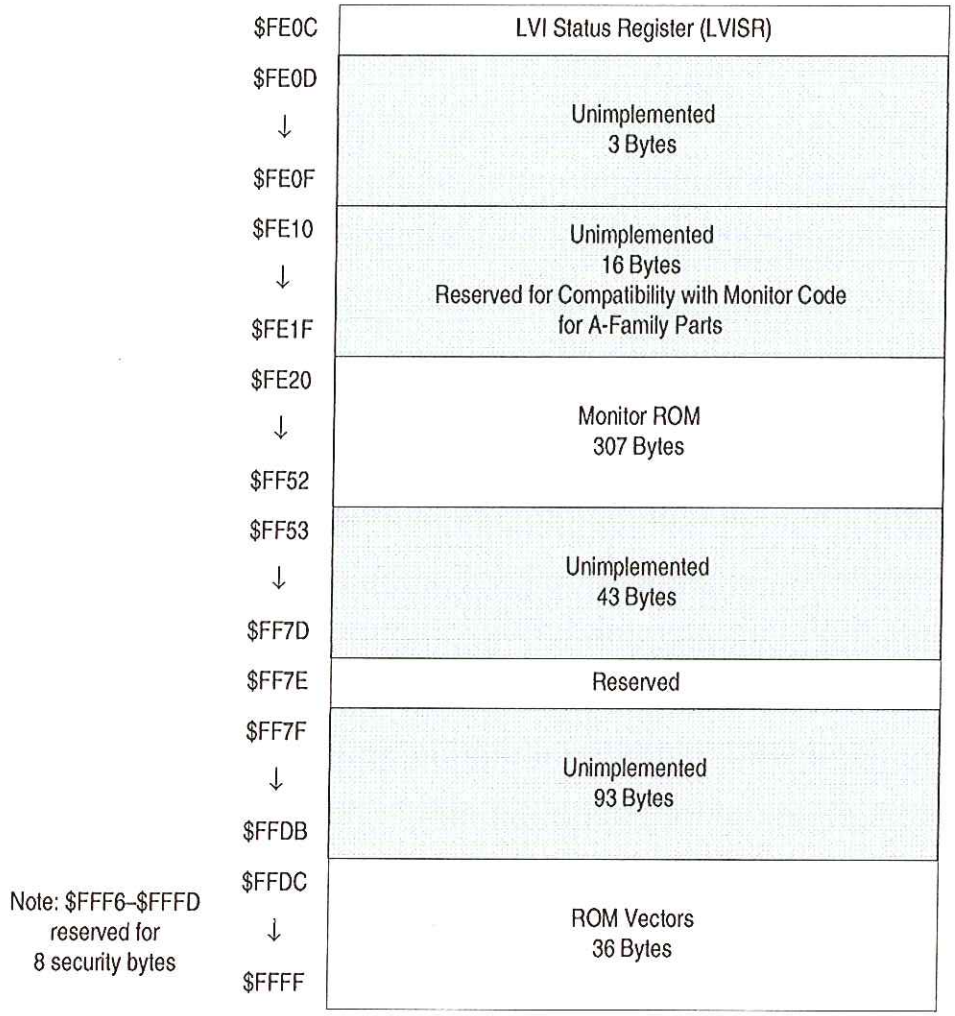


Figure A-2. MC68HC08GP32 Memory Map (Continued)

A.5 Mask Option Registers

The two mask option registers at $001E and $001F (see Figure A-3 and Figure A-4) are read-only registers. They are defined by mask options (hard-wired connections) specified at the same time as the ROM code submission.

On the MC68HC908GP32, these two registers are called configuration registers (CONFIG2 and CONFIG1).

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	OSC-STOPENB	SCIBD-SRC
Write:								
Reset:	Mask defined							

Figure A-3. Mask Option Register 2 (MOR2)

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVI5OR3	SSREC	STOP	COPD
Write:								
Reset:	Mask defined							

Figure A-4. Mask Option Register 1 (MOR1)

The bit functions for these two registers are the same as the configuration registers in MC68HC908GP32 (see Section 8. Configuration Register (CONFIG)).

A.6 Reserved Registers

The two registers at $FE08 and $FF7E are reserved locations on the MC68HC08GP32.

On the MC68HC908GP32, these two locations are the FLASH control register and the FLASH block protect register respectively.

A.7 Monitor ROM

The monitor program (monitor ROM, $FE20–$FF52) on the MC68HC08GP32 is for device testing only.

The monitor mode entry by blank reset vector bit, MODRST bit (bit 2 at $FE01), is not used in the ROM device — the reset vector will always contain data in the MC68HC08GP32.

A.8 Electrical Specifications

Electrical specifications for the MC68HC908GP32 apply to the MC68HC08GP32, except for the parameters indicated below.

A.8.1 Functional Operating Range

Characteristic	Symbol	Value			Unit
Operating temperature range	T_A	C	V	M	°C
		−40 to +85	−40 to +105	−40 to +125	
Operating voltage range	V_{DD}	3V ± 10% 5V ± 10%	3V ± 10% 5V ± 10%	— 5V ± 10%	V

A.8.2 5.0-V DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
V_{DD} supply current					
Run[3]		—	15	20	mA
Wait[4]		—	4	8	mA
Stop[5]					
25 °C	I_{DD}	—	2	—	µA
25 °C with TBM enabled[6]		—	20	—	µA
25 °C with LVI and TBM enabled[6]		—	300	—	µA
–40 °C to 125 °C		—	—	35	µA
–40 °C to 85 °C with TBM enabled[6]		—	50	—	µA
–40 °C to 85 °C with LVI and TBM enabled[6]		—	500	—	µA
Low-voltage inhibit, trip falling voltage	V_{TRIPF}	3.90	4.25	4.50	V
Low-voltage inhibit, trip rising voltage	V_{TRIPR}	4.00	4.35	4.60	V
Low-voltage inhibit reset/recover hysteresis ($V_{TRIPF} + V_{HYS} = V_{TRIPR}$)	V_{HYS}	—	100	—	mV

Notes:

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted
2. Typical values reflect average measurements at midpoint of voltage range, 25 °C only.
3. Run (operating) I_{DD} measured using external square wave clock source (f_{OSC} = 32.8 MHz). All inputs 0.2V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I_{DD}. Measured with all modules enabled.
4. Wait I_{DD} measured using external square wave clock source (f_{OSC} = 32.8 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait I_{DD}. Measured with PLL and LVI enabled.
5. Stop I_{DD} is measured with OSC1 = V_{SS}.
6. Stop I_{DD} with TBM enabled is measured using an external square wave clock source (f_{OSC} = 32.8 MHz). All inputs 0.2V from rail. No dc loads. Less than 100 pF on all outputs. All inputs configured as inputs.

A.8.3 3.0-V DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
V_{DD} supply current					
Run[3]	I_{DD}	—	5.2	8	mA
Wait[4]		—	1.65	4	mA
Stop[5]					
25 °C		—	1	—	µA
25 °C with TBM enabled[6]		—	12	—	µA
25 °C with TBM enabled[7]		—	25	—	µA
25 °C with LVI and TBM enabled[6]		—	200	—	µA
–40 °C to 85 °C		—	—	5	µA
–40 °C to 85 °C with TBM enabled[7]		—	—	50	µA
–40 °C to 85 °C with LVI and TBM enabled[6]		—	300	—	µA
Low-voltage inhibit, trip falling voltage	V_{TRIPF}	2.45	2.60	2.70	V
Low-voltage inhibit, trip rising voltage	V_{TRIPR}	2.50	2.66	2.80	V
Low-voltage inhibit reset/recover hysteresis ($V_{TRIPF} + V_{HYS} = V_{TRIPR}$)	V_{HYS}	—	60	—	mV

Notes:

1. V_{DD} = 3.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted
2. Typical values reflect average measurements at midpoint of voltage range, 25 °C only.
3. Run (operating) I_{DD} measured using external square wave clock source (f_{OSC} = 16.4 MHz). All inputs 0.2V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I_{DD}. Measured with all modules enabled.
4. Wait I_{DD} measured using external square wave clock source (f_{OSC} = 16.4 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait I_{DD}. Measured with PLL and LVI enabled.
5. Stop I_{DD} is measured with OSC1 = V_{SS}.
6. Stop I_{DD} with TBM enabled is measured using an external square wave clock source (f_{OSC} = 16.4 MHz). All inputs 0.2V from rail. No dc loads. Less than 100 pF on all outputs. All inputs configured as inputs.
7. Measured with TBM enabled using 32kHz crystal.

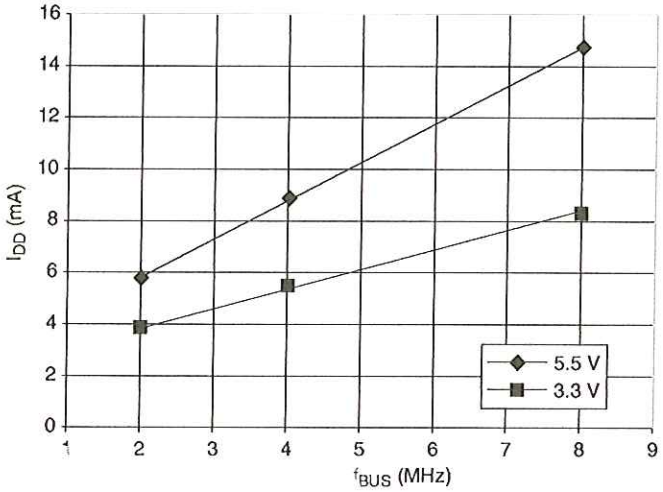


Figure A-5. Typical Operating I_{DD}

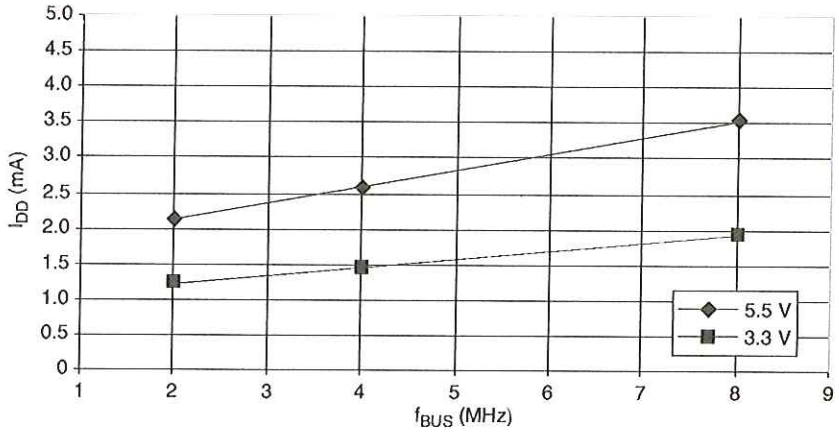


Figure A-6. Typical Wait Mode I_{DD}

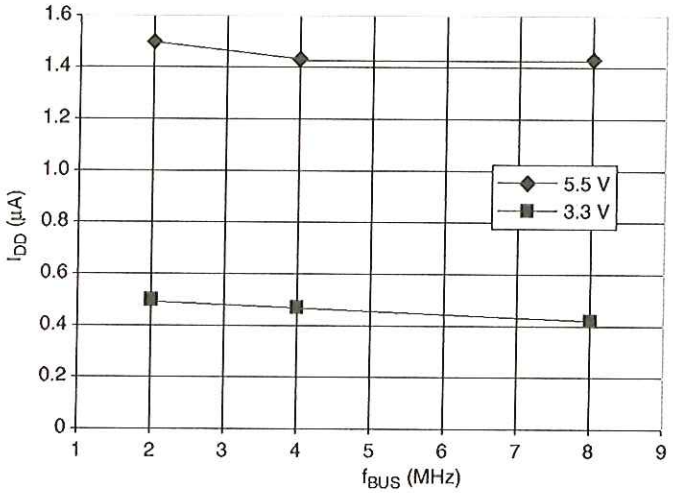


Figure A-7. Typical Stop Mode I_{DD}

A.8.4 Memory Characteristics

Characteristic	Symbol	Min	Max	Unit
RAM data retention voltage	V_{RDR}	1.3	—	V

Notes:
Since MC68HC08GP32 is a ROM device, FLASH memory electrical characteristics do not apply.

A.9 ROM MC Order Numbers

These part numbers are generic numbers only. To place an order, ROM code must be submitted to the ROM Processing Center (RPC).

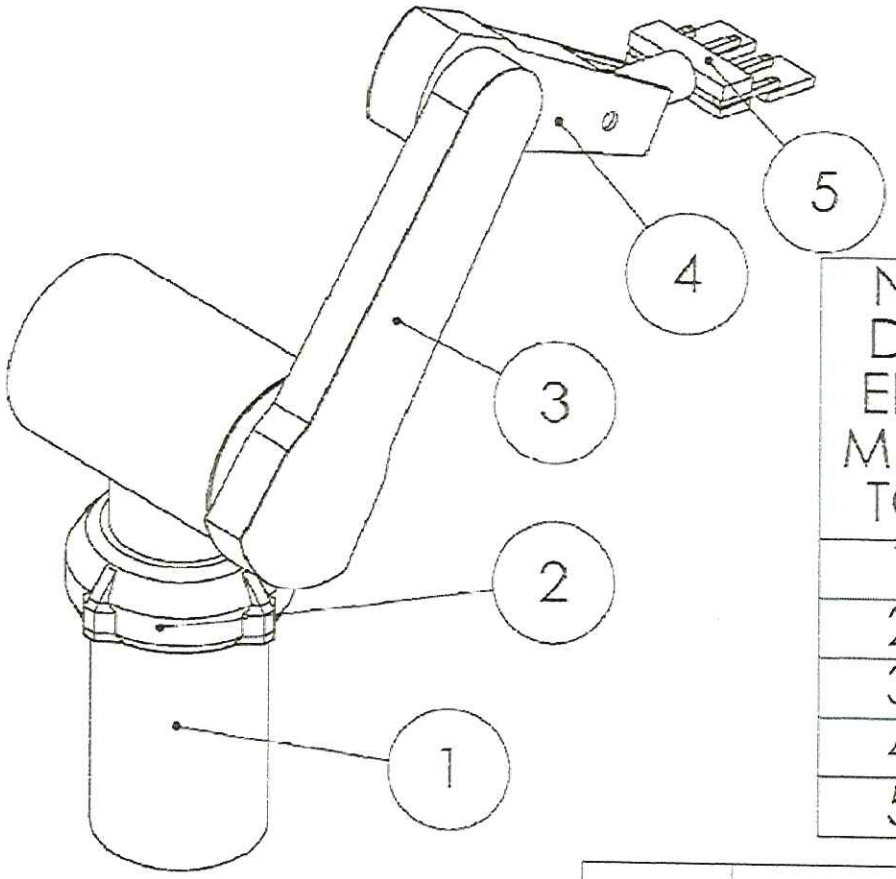
Table A-2. ROM MC Order Numbers

MC order number	Operating temperature range	Package
MC68HC08GP32CB	–40 °C to +85 °C	42-pin SDIP
MC68HC08GP32VB	–40 °C to +105 °C	
MC68HC08GP32MB(1)	–40 °C to +125 °C	
MC68HC08GP32CFB	–40 °C to +85 °C	44-pin QFP
MC68HC08GP32VFB	–40 °C to +105 °C	
MC68HC08GP32MFB(1)	–40 °C to +125 °C	

Notes:

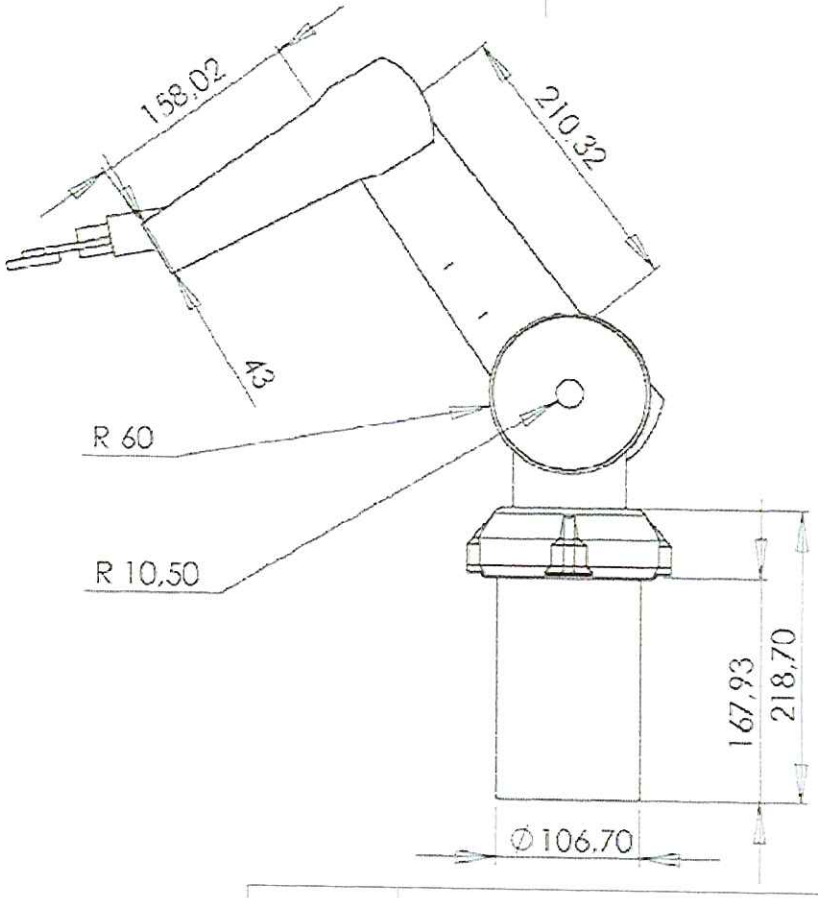
1. Temperature grade "M" is available for 5V operating voltage only.

A5 PLANOS

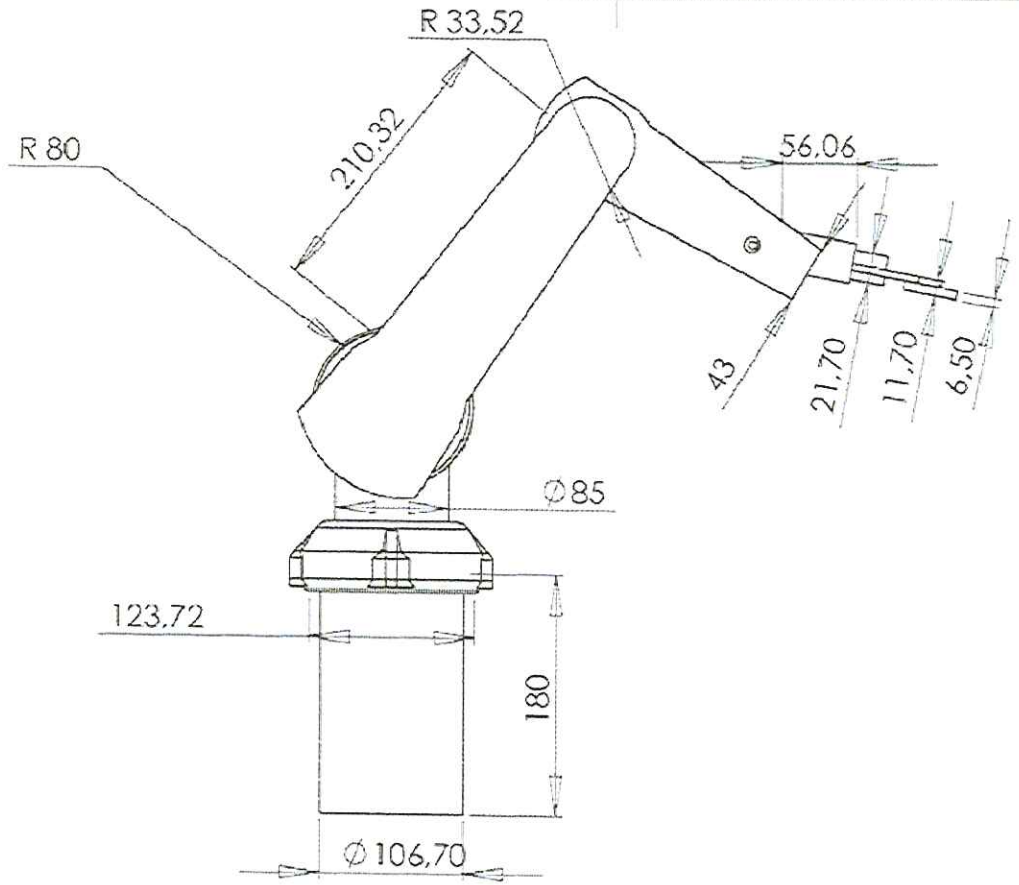


N° DE ELEMENTO	NÚMERO DE PIEZA	CANTIDAD.
1	base motor 1	1
2	base	1
3	brazo	1
4	Antebrazo	1
5	gripper	1

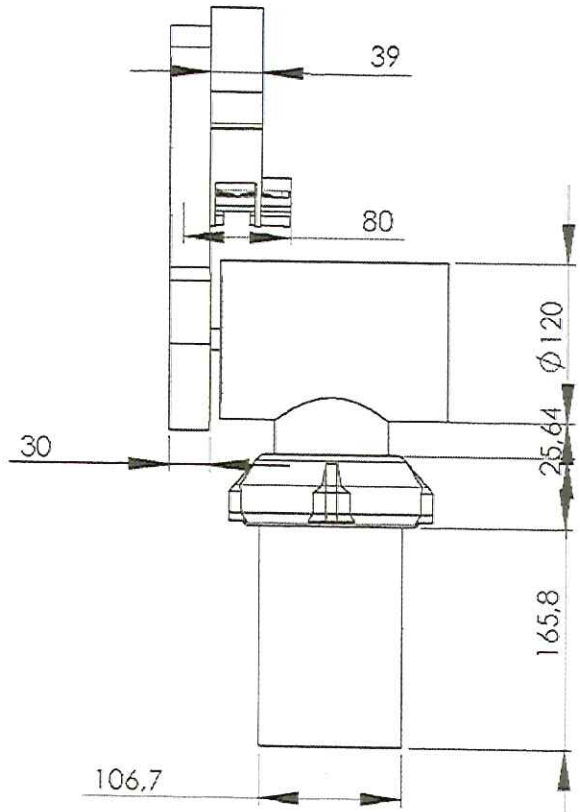
ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	1810003, 19199034, 18100080		
ESCALA 1:5	ISOMETRICA DEL MANIPULADOR		PROYECTO DE GRADO

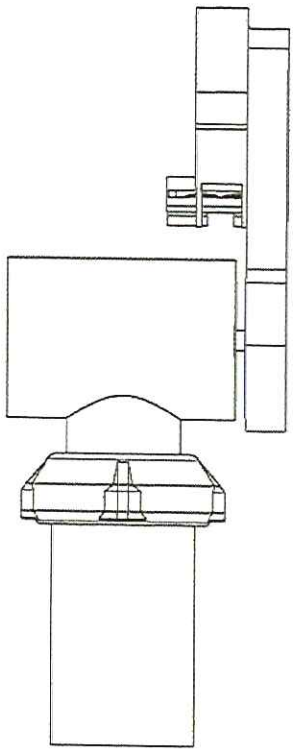


ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 18199034, 18100080		
ESCALA 1:1	VISTA AUXILIAR IZQUIERDA		PROYECTO DE GRADO

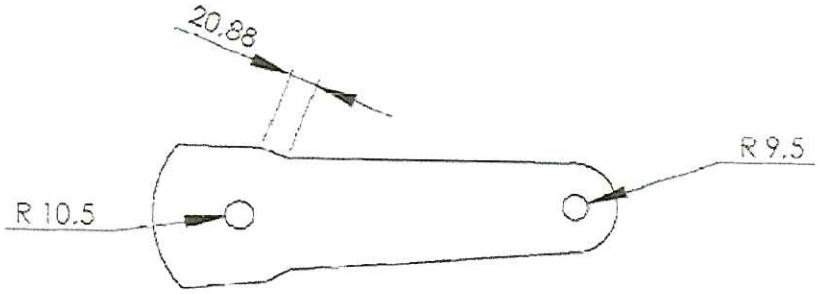


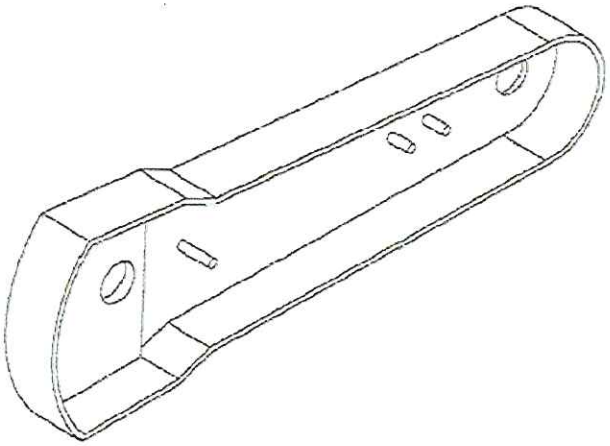
ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 18199034, 18100080		
ESCALA 1:5	VISTA AUXILIAR DERECHA		PROYECTO DE GRADO

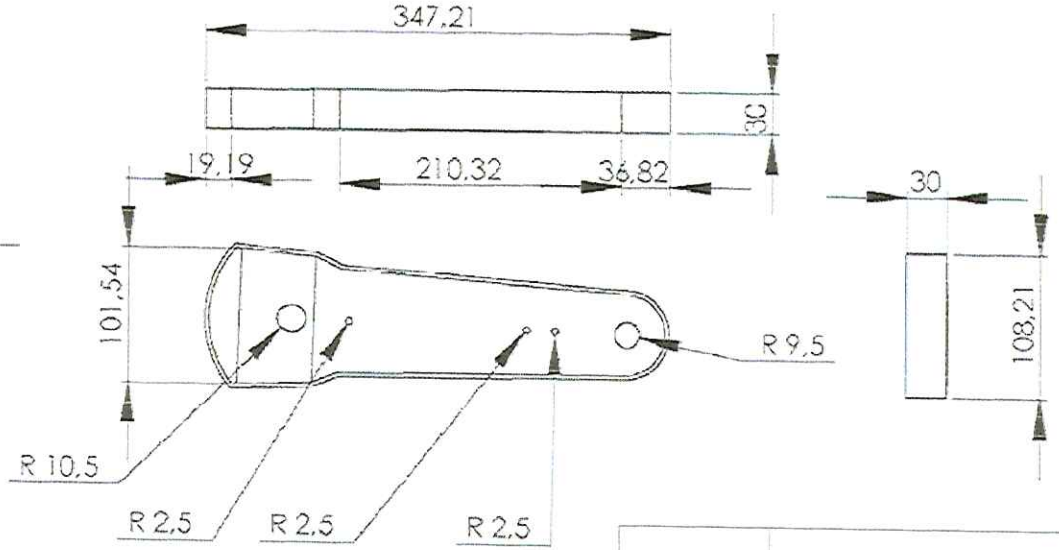




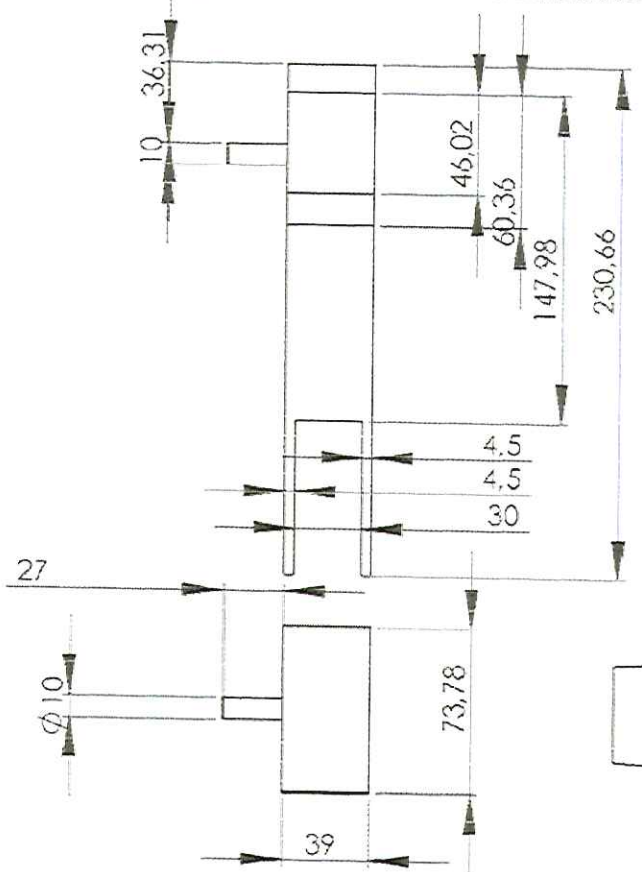
ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA	VISTAS FRONTAL Y POSTERIOR		PROYECTO DE GRADO

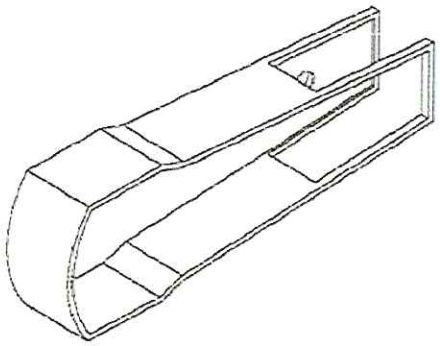




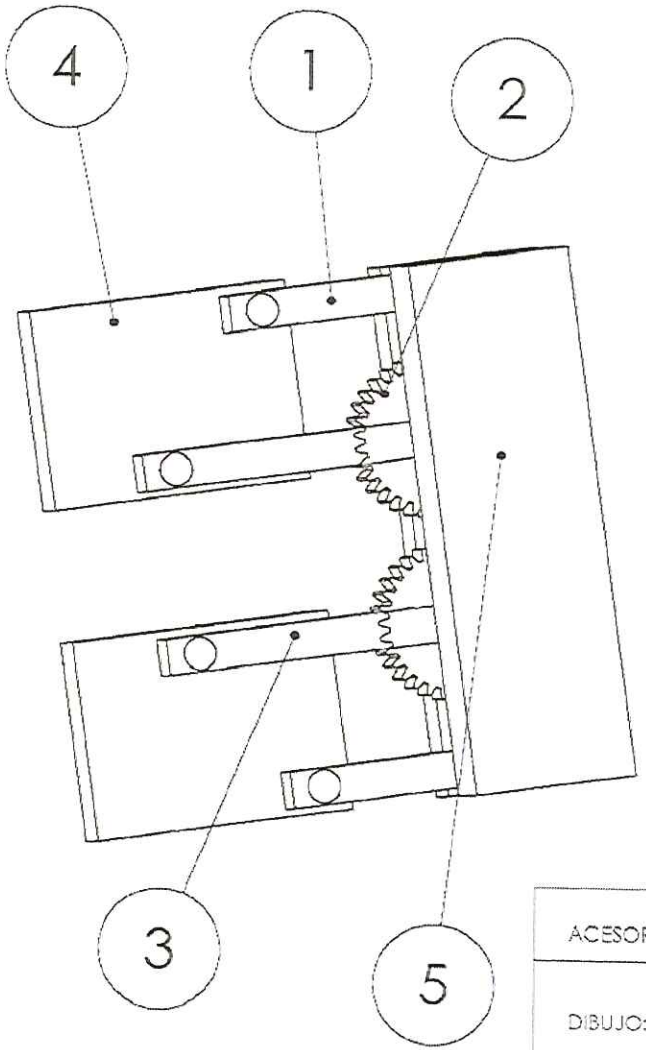


ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA 1:5	BRAZO		PROYECTO DE GRADO



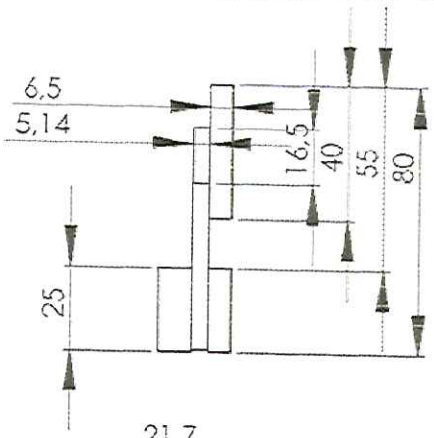


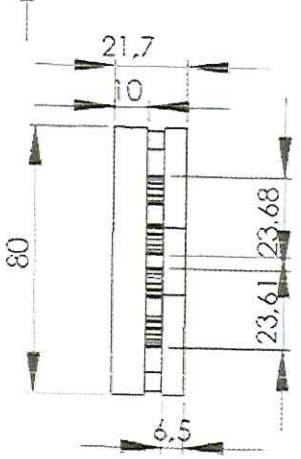
ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA 1:5	ANTEBRAZO		PROYECTO DE GRADO

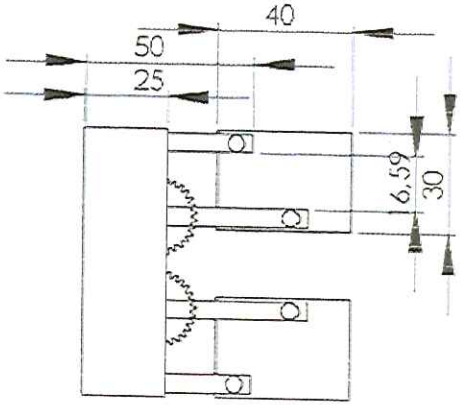


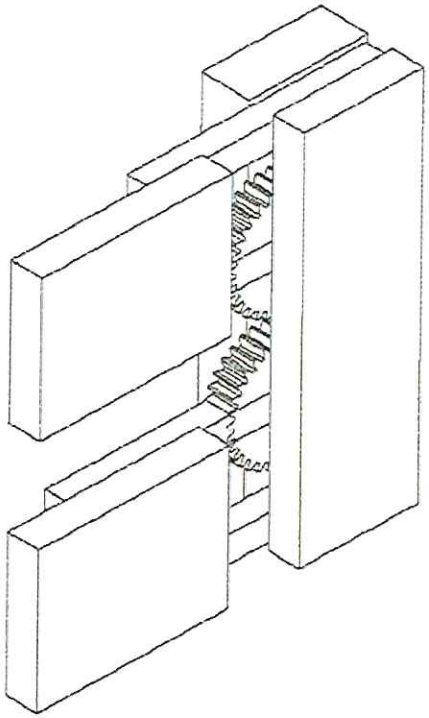
N° Pieza	descripcion	CANTIDAD.
1	eslabon1	2
2	gear 0.7M 32T	2
3	eslabon 2	2
4	tenaza	2
5	soporte	2

ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA	TENAZA		PROYECTO DE GRADO
1:1			

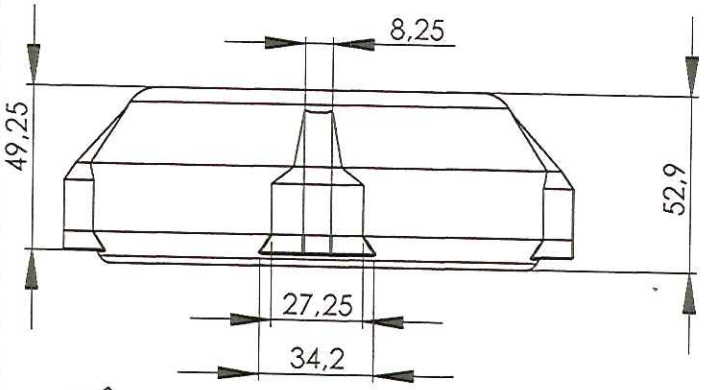


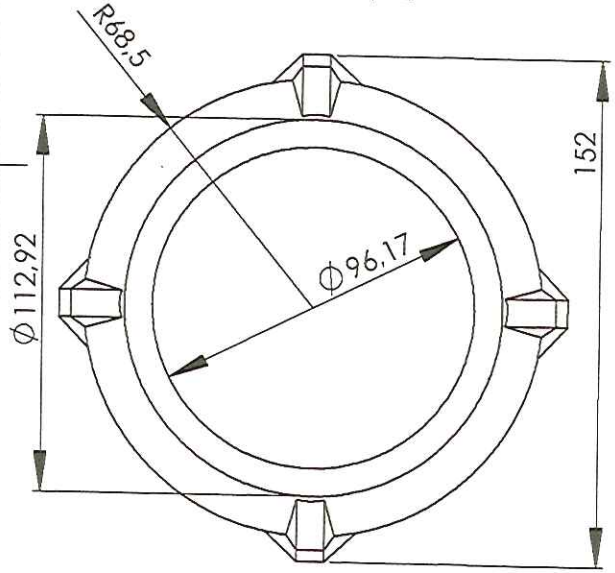


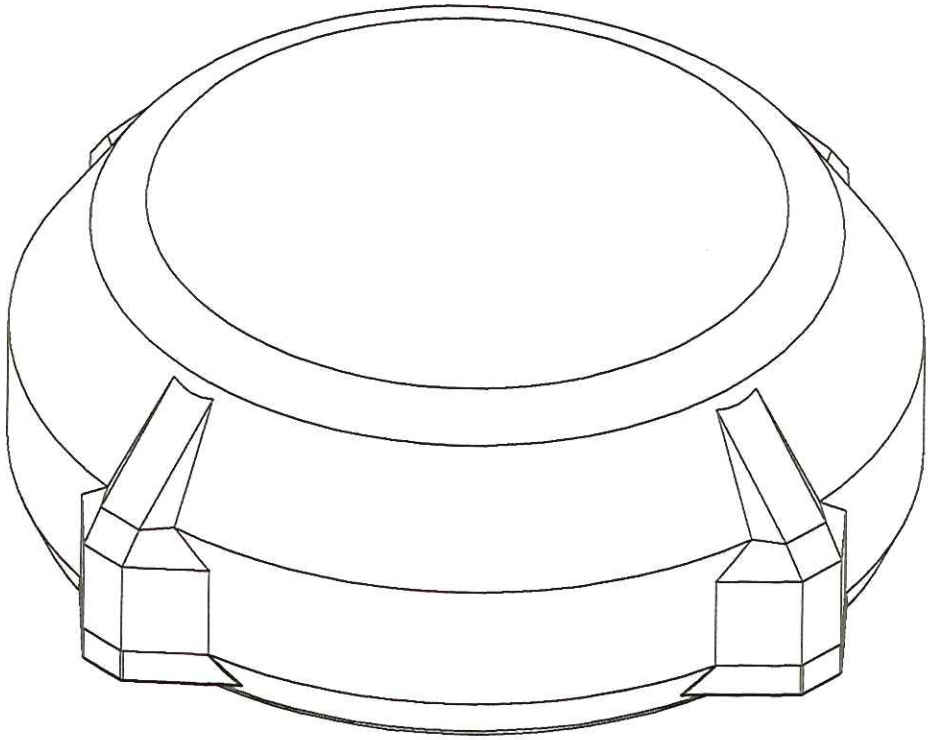




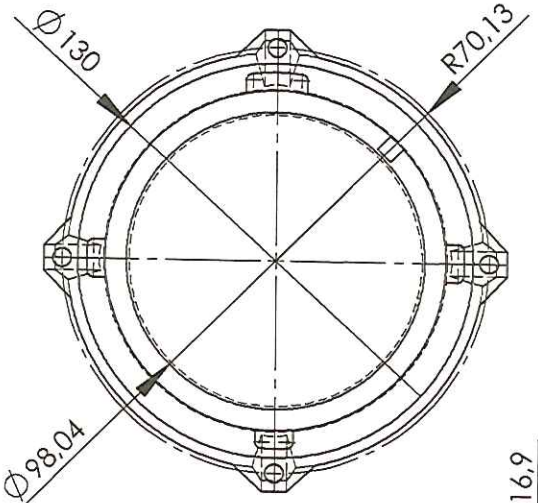
ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA 1:2	TENAZA		PROYECTO DE GRADO

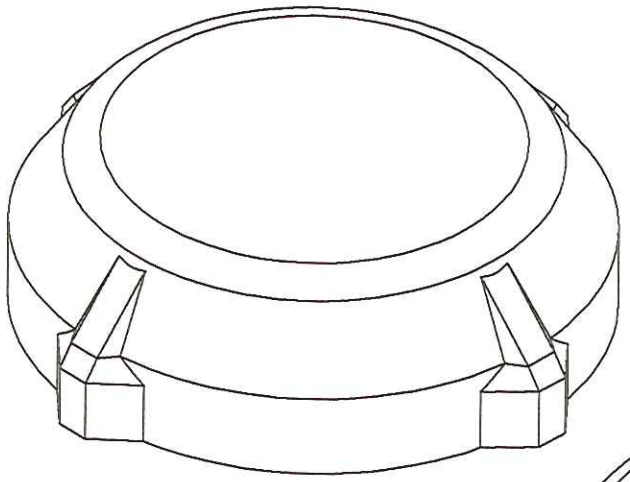


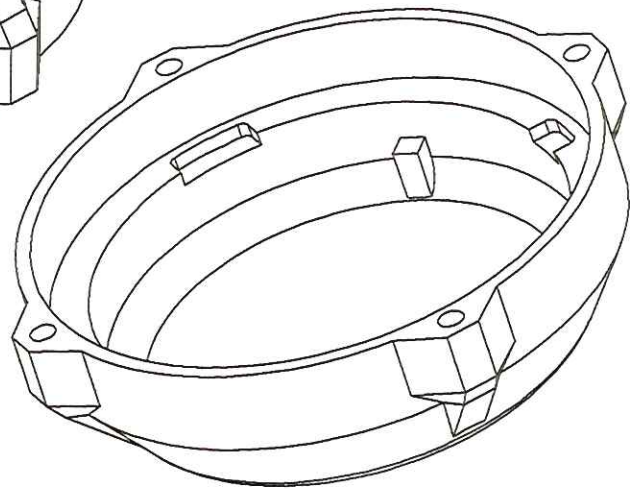


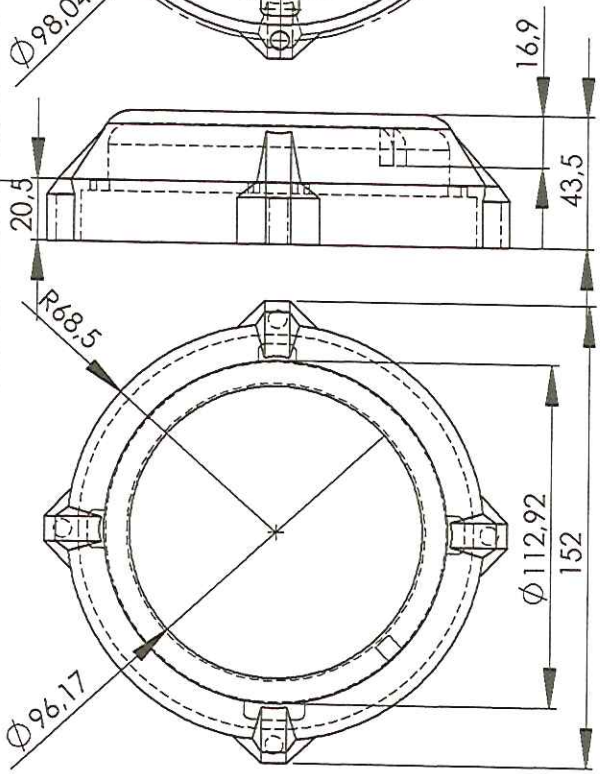


ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA 1:2	BASE ROTATORIA		PROYECTO DE GRADO

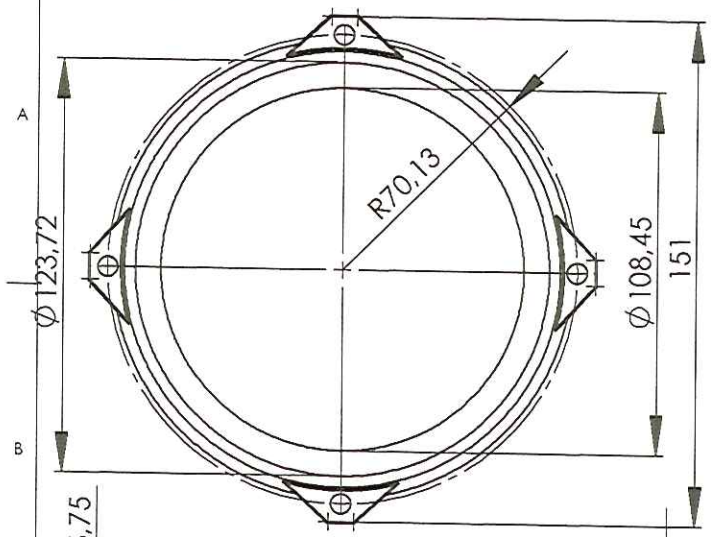


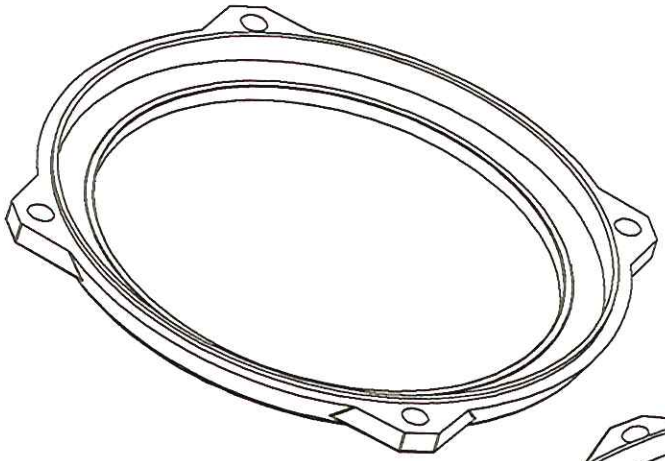


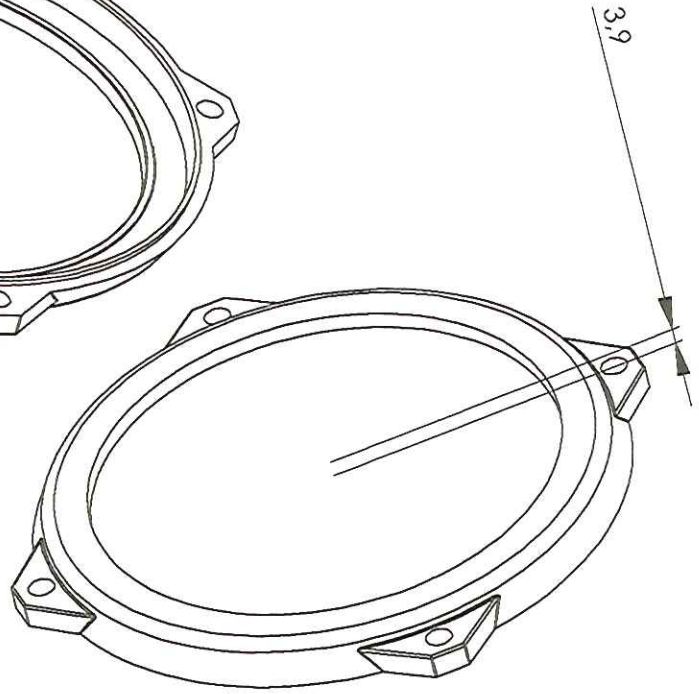


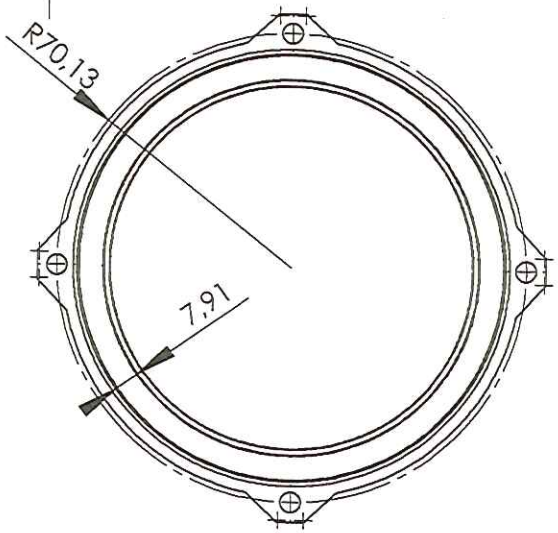


ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	1810003, 19199034, 18100080		
ESCALA 1:2.2	CARCAZA SUPERIOR - BASE ROTATORIA		PROYECTO DE GRADO

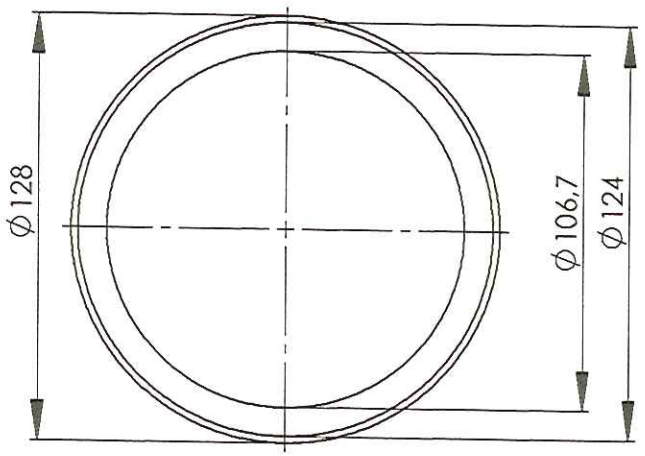


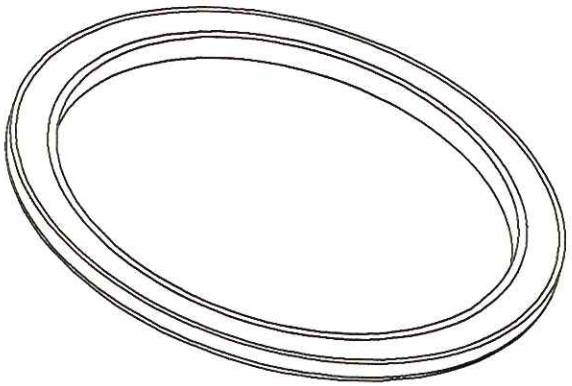


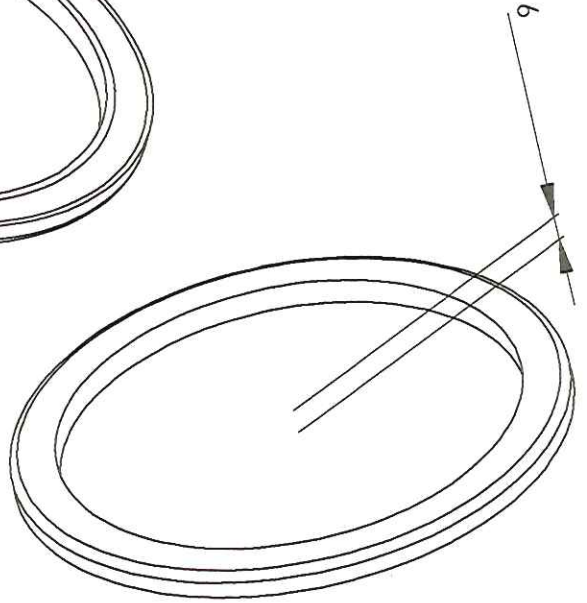


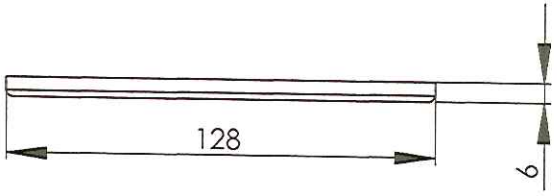


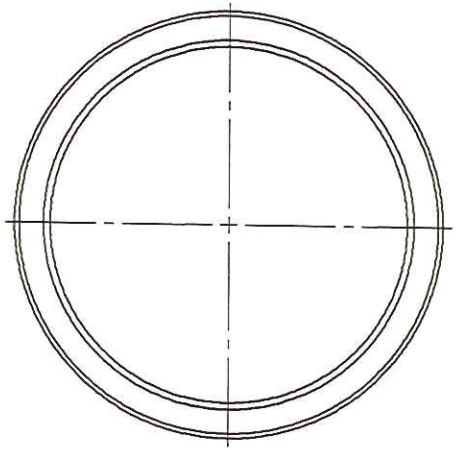
ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA 1:2	CARCAZA INFERIOR - BASE GIRATORIA		PROYECTO DE GRADO



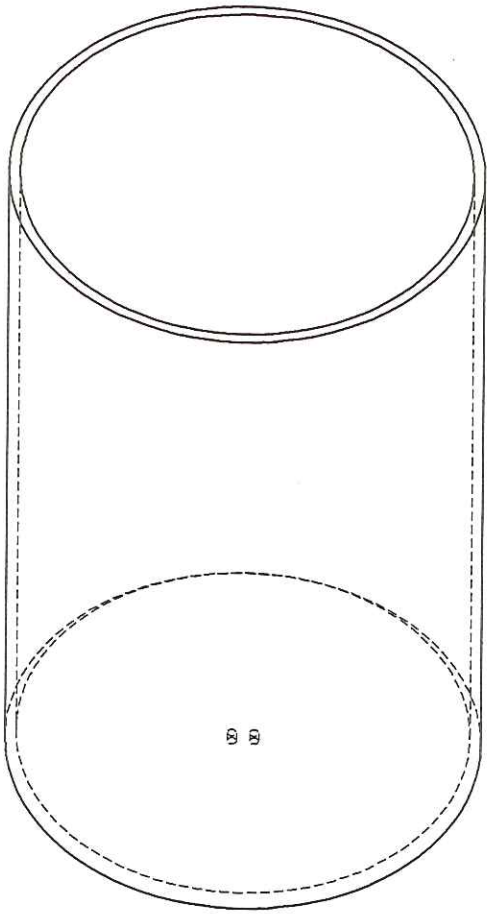


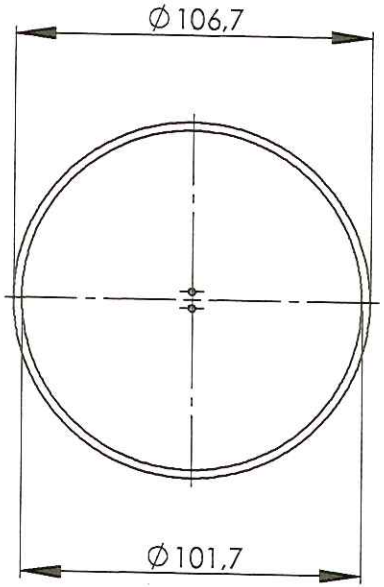


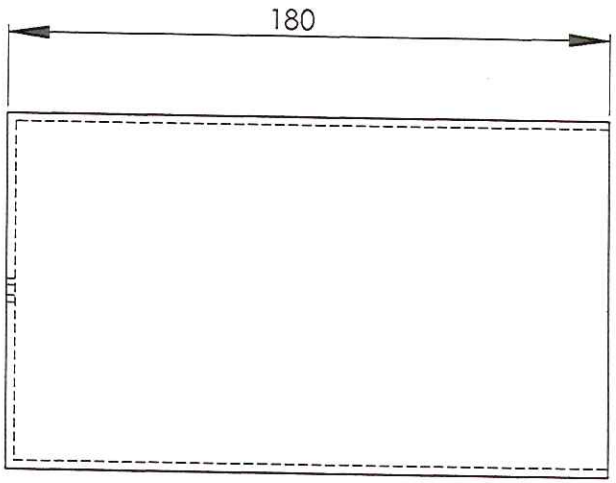




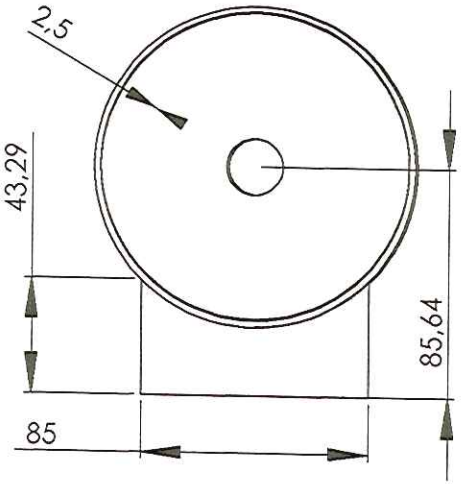
ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA 1:2	SOPORTE BALINERAS - BASE ROTATORIA		PROYECTO DE GRADO

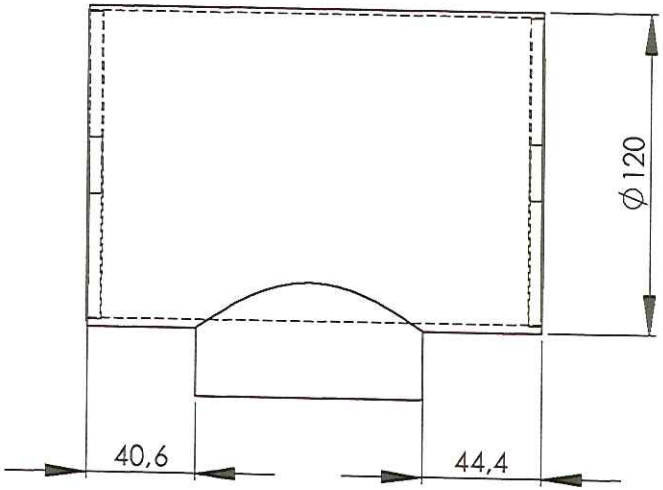


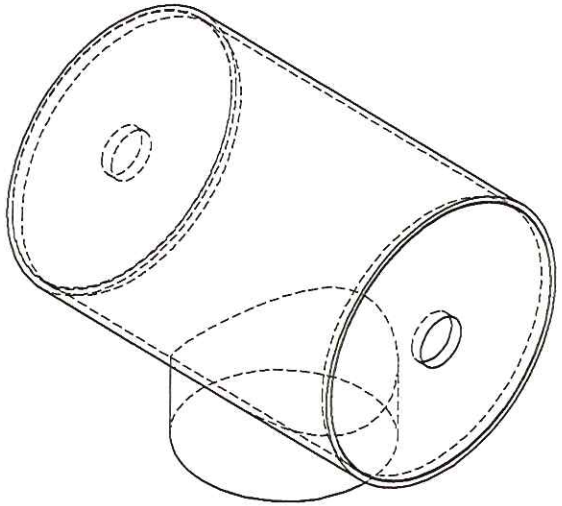


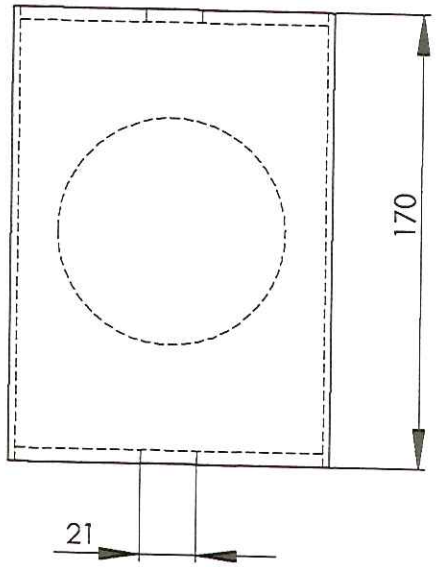


ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA 1:2	BASE SOPORTE		PROYECTO DE GRADO









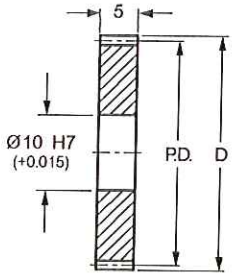
ACESOR:	JOHN FABER ARCHILA	UNAB Universidad Autonoma de Bucaramanga	FACULTAD DE INGENIERIA MECATRONICA
DIBUJO:	John W Moreno Vergel Raul Munive Herrera Joel Rivera Peña		
CODIGO:	18100038, 19199034, 18100080		
ESCALA 1:2.5	BASE "T"		PROYECTO DE GRADO

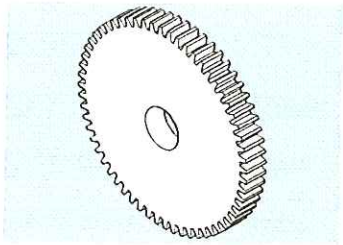
Metric Metric Metric Metric Metric Metric Metric

SDP/SI Hubless Spur Gears - Module 0.8

Stock Drive Products/Sterling Instrument ■ Phone: 516-328-3300 ■ Fax: 516-326-8827

■ ISO CLASS 7 ■ 20° PRESSURE ANGLE ■ 5 mm FACE ■ 10 mm BORE





Precision Component

Catalog Number		No. of Teeth	P.D.	D
303 Stainless Steel	2024 Aluminum* Anodized			
S12N08M020S0510	S12N08M020A0510	20	16	17.6
S12N08M024S0510	S12N08M024A0510	24	19.2	20.8
S12N08M025S0510	S12N08M025A0510	25	20	21.6
S12N08M028S0510	S12N08M028A0510	28	22.4	24
S12N08M032S0510	S12N08M032A0510	32	25.6	27.2
S12N08M036S0510	S12N08M036A0510	36	28.8	30.4
S12N08M040S0510	S12N08M040A0510	40	32	33.6
S12N08M048S0510	S12N08M048A0510	48	38.4	40
S12N08M050S0510	S12N08M050A0510	50	40	41.6
S12N08M054S0510	S12N08M054A0510	54	43.2	44.8
S12N08M056S0510	S12N08M056A0510	56	44.8	46.4
S12N08M060S0510	S12N08M060A0510	60	48	49.6
S12N08M064S0510	S12N08M064A0510	64	51.2	52.8
S12N08M065S0510	S12N08M065A0510	65	52	53.6
S12N08M070S0510	S12N08M070A0510	70	56	57.6
S12N08M072S0510	S12N08M072A0510	72	57.6	59.2
S12N08M075S0510	S12N08M075A0510	75	60	61.6
S12N08M080S0510	S12N08M080A0510	80	64	65.6
S12N08M084S0510	S12N08M084A0510	84	67.2	68.8
S12N08M085S0510	S12N08M085A0510	85	68	69.6
S12N08M090S0510	S12N08M090A0510	90	72	73.6
S12N08M096S0510	S12N08M096A0510	96	76.8	78.4
S12N08M100S0510	S12N08M100A0510	100	80	81.6
S12N08M108S0510	S12N08M108A0510	108	86.4	88
S12N08M110S0510	S12N08M110A0510	110	88	89.6
S12N08M115S0510	S12N08M115A0510	115	92	93.6
S12N08M120S0510	S12N08M120A0510	120	96	97.6
S12N08M126S0510	S12N08M126A0510	126	100.8	102.4
S12N08M128S0510	S12N08M128A0510	128	102.4	104
S12N08M132S0510	S12N08M132A0510	132	105.6	107.2

* T4 or T351 Aluminum Alloy, anodized before cutting.

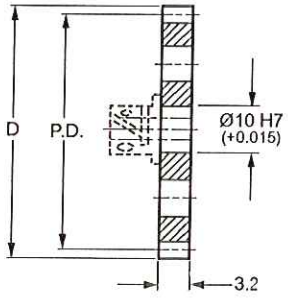
Available as special order: Number of teeth not listed, different bore size and/or material, passivation for stainless steel.

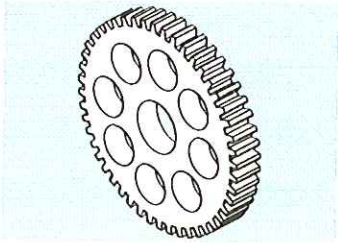
Metric Metric Metric Metric Metric Metric Metric

Hubless Spur Gears - Module 0.8

SDP/SI Stock Drive Products/Sterling Instrument ■ Phone: 516-328-3300 ■ Fax: 516-326-8827

■ ISO CLASS 7 ■ 20° PRESSURE ANGLE ■ 3.2 mm FACE ■ 10 mm BORE





Precision Component

- All gears over 54 mm diameter supplied with lightening holes.
- See index for hubs. Assembled upon request at nominal charge.

Catalog Number		No. of Teeth	P.D.	D
303 Stainless Steel	2024 Aluminum* Anodized			
S12S08M020N0310	S12A08M020N0310	20	16	17.6
S12S08M024N0310	S12A08M024N0310	24	19.2	20.8
S12S08M028N0310	S12A08M028N0310	28	22.4	24
S12S08M032N0310	S12A08M032N0310	32	25.6	27.2
S12S08M042N0310	S12A08M042N0310	42	33.6	35.2
S12S08M048N0310	S12A08M048N0310	48	38.4	40
S12S08M056N0310	S12A08M056N0310	56	44.8	46.4
S12S08M064N0310	S12A08M064N0310	64	51.2	52.8
S12S08M066N0310	S12A08M066N0310	66	52.8	54.4
S12S08M070N0310	S12A08M070N0310	70	56	57.6
S12S08M072N0310	S12A08M072N0310	72	57.6	59.2
S12S08M080N0310	S12A08M080N0310	80	64	65.6
S12S08M096N0310	S12A08M096N0310	96	76.8	78.4
S12S08M120N0310	S12A08M120N0310	120	96	97.6
S12S08M128N0310	S12A08M128N0310	128	102.4	104

*T4 or T351 Aluminum Alloy, anodized before cutting.

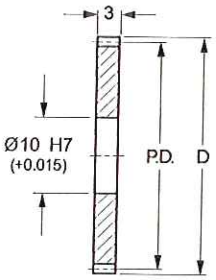
Available as special order: 14-1/2° P.A., teeth not listed, different bore size and/or material, passivation for Stainless Steel.

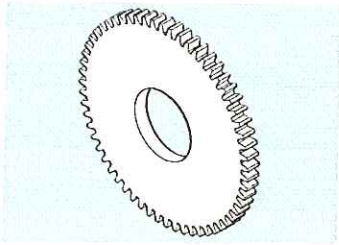
Metric Metric Metric Metric Metric Metric Metric

Hubless Spur Gears - Module 0.8

Stock Drive Products/Sterling Instrument ■ Phone: 516-328-3300 ■ Fax: 516-326-8827

■ ISO CLASS 7 ■ 20° PRESSURE ANGLE ■ 3 mm FACE ■ 10 mm BORE





Precision Component

Catalog Number		No. of Teeth	P.D.	D
303 Stainless Steel	2024 Aluminum* Anodized			
S12N08M020S0310	S12N08M020A0310	20	16	17.6
S12N08M024S0310	S12N08M024A0310	24	19.2	20.8
S12N08M025S0310	S12N08M025A0310	25	20	21.6
S12N08M028S0310	S12N08M028A0310	28	22.4	24
S12N08M032S0310	S12N08M032A0310	32	25.6	27.2
S12N08M036S0310	S12N08M036A0310	36	28.8	30.4
S12N08M040S0310	S12N08M040A0310	40	32	33.6
S12N08M048S0310	S12N08M048A0310	48	38.4	40
S12N08M050S0310	S12N08M050A0310	50	40	41.6
S12N08M054S0310	S12N08M054A0310	54	43.2	44.8
S12N08M056S0310	S12N08M056A0310	56	44.8	46.4
S12N08M060S0310	S12N08M060A0310	60	48	49.6
S12N08M064S0310	S12N08M064A0310	64	51.2	52.8
S12N08M065S0310	S12N08M065A0310	65	52	53.6
S12N08M070S0310	S12N08M070A0310	70	56	57.6
S12N08M072S0310	S12N08M072A0310	72	57.6	59.2
S12N08M075S0310	S12N08M075A0310	75	60	61.6
S12N08M080S0310	S12N08M080A0310	80	64	65.6
S12N08M084S0310	S12N08M084A0310	84	67.2	68.8
S12N08M085S0310	S12N08M085A0310	85	68	69.6
S12N08M090S0310	S12N08M090A0310	90	72	73.6
S12N08M096S0310	S12N08M096A0310	96	76.8	78.4
S12N08M100S0310	S12N08M100A0310	100	80	81.6
S12N08M108S0310	S12N08M108A0310	108	86.4	88
S12N08M110S0310	S12N08M110A0310	110	88	89.6
S12N08M115S0310	S12N08M115A0310	115	92	93.6
S12N08M120S0310	S12N08M120A0310	120	96	97.6
S12N08M126S0310	S12N08M126A0310	126	100.8	102.4
S12N08M128S0310	S12N08M128A0310	128	102.4	104
S12N08M132S0310	S12N08M132A0310	132	105.6	107.2

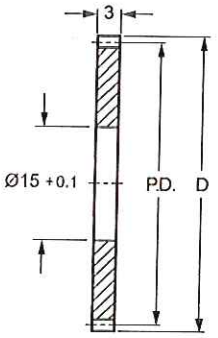
* T4 or T351 Aluminum Alloy, anodized before cutting.

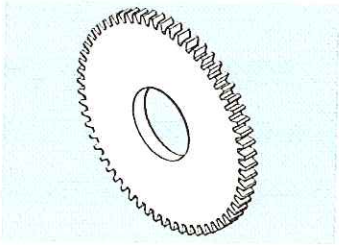
Available as special order: Number of teeth not listed, different bore size and/or material, passivation for stainless steel.

Hubless Spur Gears - Module 0.75

SDP/SI Stock Drive Products/Sterling Instrument ■ Phone: 516-328-3300 ■ Fax: 516-326-8827

■ 20° PRESSURE ANGLE ■ 3 mm FACE WIDTH





MATERIAL: Brass

Catalog Number	No. of Teeth	P.D.	D
A 1B 1MYKH7050	50	37.5	39
A 1B 1MYKH7056	56	42	43.5
A 1B 1MYKH7060	60	45	46.5
A 1B 1MYKH7064	64	48	49.5
A 1B 1MYKH7065	65	48.8	50.3
A 1B 1MYKH7066	66	49.5	51
A 1B 1MYKH7070	70	52.5	54
A 1B 1MYKH7072	72	54	55.5
A 1B 1MYKH7075	75	56.3	57.8
A 1B 1MYKH7080	80	60	61.5
A 1B 1MYKH7090	90	67.5	69
A 1B 1MYKH7100	100	75	76.5
A 1B 1MYKH7120	120	90	91.5

Did You Know?

...You can obtain additional literature by viewing our "home page" on the INTERNET?

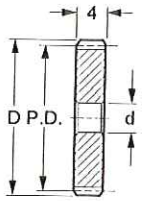
http://www.sdp-si.com

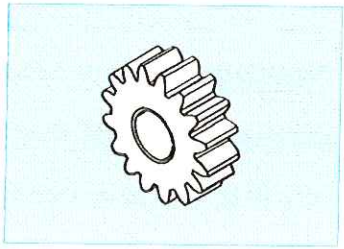
SDP/SI Metric Metric Metric Metric Metric Metric Metric

Pinions - Module 0.8

Stock Drive Products/Sterling Instrument ■ Phone: 516-328-3300 ■ Fax: 516-326-8827

■ 20° PRESSURE ANGLE ■ HUBLESS ■ 4 mm FACE WIDTH





MATERIAL: Brass

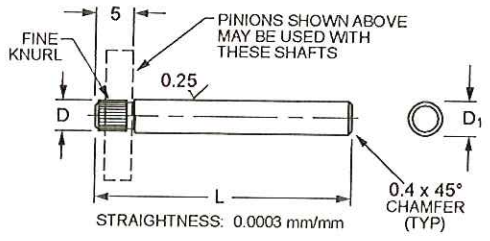
Catalog Number	No. of Teeth	P.D.	D	d Bore +0.025
A 1B 1MY08006	6	4.8	6.4	2
A 1B 1MY08008	8	6.4	8	3
A 1B 1MY08010	10	8	9.6	
A 1B 1MY08012	12	9.6	11.2	
A 1B 1MY08014	14	11.2	12.8	
A 1B 1MY08016	16	12.8	14.4	
A 1B 1MY08018	18	14.4	16	

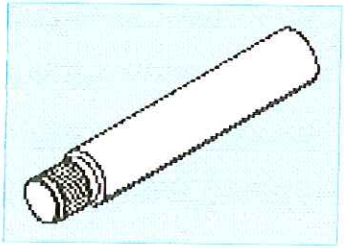
SDP/SI Metric Metric Metric Metric Metric Metric Metric

Pinion Shafts

Stock Drive Products/Sterling Instrument ■ Phone: 516-328-3300 ■ Fax: 516-326-8827

■ UNDERSIZED DIAMETER





MATERIAL: 303 Stainless Stee

Catalog Number	D Dia. +0.025	D_1 Dia. -0.004 -0.012	L Length ± 0.25
A 7X 1MP0550B	2	5	50
A 7X 1MP0675	3	6	75

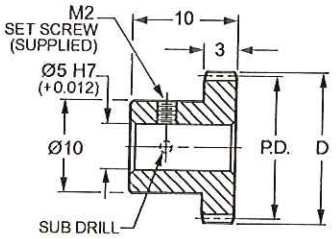
Discounts available for large quantity orders.

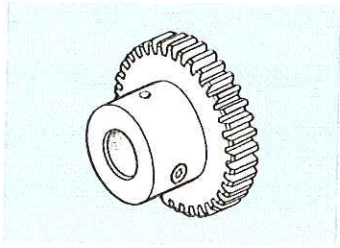
Metric Metric Metric Metric Metric Metric Metric

Spur Gears - Module 0.5

SDP/SI Stock Drive Products/Sterling Instrument ■ Phone: 516-328-3300 ■ Fax: 516-326-8827

■ ISO CLASS 7 ■ 20° PRESSURE ANGLE ■ 3 mm FACE ■ 5 mm BORE





Precision Component

Catalog Number		No. of Teeth	P.D.	D
303 Stainless Steel	2024 Aluminum* Anodized			
S10T05M020S0305	S10T05M020A0305	20	10	11
S10T05M021S0305	S10T05M021A0305	21	10.5	11.5
S10T05M022S0305	S10T05M022A0305	22	11	12
S10T05M024S0305	S10T05M024A0305	24	12	13
S10T05M025S0305	S10T05M025A0305	25	12.5	13.5
S10T05M028S0305	S10T05M028A0305	28	14	15
S10T05M030S0305	S10T05M030A0305	30	15	16
S10T05M032S0305	S10T05M032A0305	32	16	17
S10T05M034S0305	S10T05M034A0305	34	17	18
S10T05M036S0305	S10T05M036A0305	36	18	19
S10T05M040S0305	S10T05M040A0305	40	20	21
S10T05M042S0305	S10T05M042A0305	42	21	22
S10T05M046S0305	S10T05M046A0305	46	23	24
S10T05M048S0305	S10T05M048A0305	48	24	25
S10T05M050S0305	S10T05M050A0305	50	25	26
S10T05M060S0305	S10T05M060A0305	60	30	31
S10T05M064S0305	S10T05M064A0305	64	32	33
S10T05M070S0305	S10T05M070A0305	70	35	36
S10T05M072S0305	S10T05M072A0305	72	36	37
S10T05M080S0305	S10T05M080A0305	80	40	41
S10T05M084S0305	S10T05M084A0305	84	42	43
S10T05M090S0305	S10T05M090A0305	90	45	46
S10T05M096S0305	S10T05M096A0305	96	48	49
S10T05M105S0305	S10T05M105A0305	105	52.5	53.5
S10T05M120S0305	S10T05M120A0305	120	60	61
S10T05M132S0305	S10T05M132A0305	132	66	67
S10T05M144S0305	S10T05M144A0305	144	72	73
S10T05M156S0305	S10T05M156A0305	156	78	79
S10T05M168S0305	S10T05M168A0305	168	84	85
S10T05M192S0305	S10T05M192A0305	192	96	97

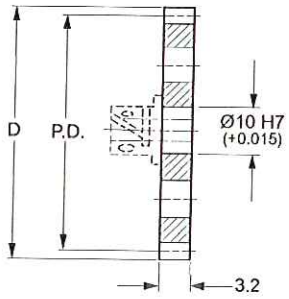
* T4 or T351 Aluminum Alloy, anodized before cutting.

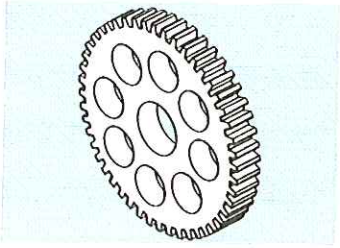
Available as special order: Number of teeth not listed, different bore size and/or material, passivation for stainless steel.

Hubless Spur Gears - Module 0.5

Stock Drive Products/Sterling Instrument ■ Phone: 516-328-3300 ■ Fax: 516-326-8827

■ ISO CLASS 7 ■ 20° PRESSURE ANGLE ■ 3.2 mm FACE ■ 10 mm BORE





Precision Component

- All gears over 54 mm diameter supplied with lightening holes.
- See index for hubs. Assembled upon request at nominal charge.

Catalog Number		No. of Teeth	P.D.	D
303 Stainless Steel	2024 Aluminum* Anodized			
S12S05M030N0310	S12A05M030N0310	30	15	16
S12S05M032N0310	S12A05M032N0310	32	16	17
S12S05M048N0310	S12A05M048N0310	48	24	25
S12S05M054N0310	S12A05M054N0310	54	27	28
S12S05M060N0310	S12A05M060N0310	60	30	31
S12S05M065N0310	S12A05M065N0310	65	32.5	33.5
S12S05M070N0310	S12A05M070N0310	70	35	36
S12S05M080N0310	S12A05M080N0310	80	40	41
S12S05M096N0310	S12A05M096N0310	96	48	49
S12S05M105N0310	S12A05M105N0310	105	52.5	53.5
S12S05M120N0310	S12A05M120N0310	120	60	61
S12S05M130N0310	S12A05M130N0310	130	65	66
S12S05M150N0310	S12A05M150N0310	150	75	76
S12S05M180N0310	S12A05M180N0310	180	90	91
S12S05M192N0310	S12A05M192N0310	192	96	97
S12S05M198N0310	S12A05M198N0310	198	99	100

*T4 or T351 Aluminum Alloy, anodized before cutting.

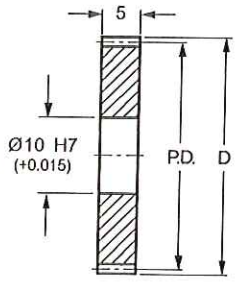
Available as special order: 14-1/2° P.A., teeth not listed, different bore size and/or material, passivation for Stainless Steel.

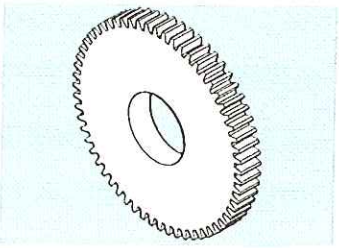
Metric Metric Metric Metric Metric Metric Metric

SDP/SI Hubless Spur Gears - Module 0.5

Stock Drive Products/Sterling Instrument ■ Phone: 516-326-3300 ■ Fax: 516-326-8827

■ ISO CLASS 7 ■ 20° PRESSURE ANGLE ■ 5 mm FACE ■ 10 mm BORE





Precision Component

Catalog Number		No. of Teeth	P.D.	D
303 Stainless Steel	2024 Aluminum* Anodized			
S12N05M030S0510	S12N05M030A0510	30	15	16
S12N05M032S0510	S12N05M032A0510	32	16	17
S12N05M036S0510	S12N05M036A0510	36	18	19
S12N05M040S0510	S12N05M040A0510	40	20	21
S12N05M042S0510	S12N05M042A0510	42	21	22
S12N05M045S0510	S12N05M045A0510	45	22.5	23.5
S12N05M048S0510	S12N05M048A0510	48	24	25
S12N05M050S0510	S12N05M050A0510	50	25	26
S12N05M056S0510	S12N05M056A0510	56	28	29
S12N05M060S0510	S12N05M060A0510	60	30	31
S12N05M064S0510	S12N05M064A0510	64	32	33
S12N05M070S0510	S12N05M070A0510	70	35	36
S12N05M072S0510	S12N05M072A0510	72	36	37
S12N05M080S0510	S12N05M080A0510	80	40	41
S12N05M084S0510	S12N05M084A0510	84	42	43
S12N05M090S0510	S12N05M090A0510	90	45	46
S12N05M096S0510	S12N05M096A0510	96	48	49
S12N05M100S0510	S12N05M100A0510	100	50	51
S12N05M108S0510	S12N05M108A0510	108	54	55
S12N05M110S0510	S12N05M110A0510	110	55	56
S12N05M120S0510	S12N05M120A0510	120	60	61
S12N05M130S0510	S12N05M130A0510	130	65	66
S12N05M132S0510	S12N05M132A0510	132	66	67
S12N05M138S0510	S12N05M138A0510	138	69	70
S12N05M140S0510	S12N05M140A0510	140	70	71
S12N05M150S0510	S12N05M150A0510	150	75	76
S12N05M160S0510	S12N05M160A0510	160	80	81
S12N05M180S0510	S12N05M180A0510	180	90	91
S12N05M186S0510	S12N05M186A0510	186	93	94
S12N05M192S0510	S12N05M192A0510	192	96	97

* T4 or T351 Aluminum Alloy, anodized before cutting.

Available as special order: Number of teeth not listed, different bore size and/or material, passivation for stainless steel.