UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
(CAMPO DE CONOCIMIENTO: MECÁNICA)
(CAMPO DISCIPLINARIO: MECATRÓNICA)

MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE: COORDINACIÓN DE MOVIMIENTOS EN AMBIENTES INTELIGENTES

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
ING. IGNACIO CARLOS CRUZ LÓPEZ

TUTOR PRINCIPAL
DR. VÍCTOR JAVIER GONZÁLEZ VILLELA, FACULTAD DE INGENIERÍA
COMITÉ TUTOR
DR. FRANCISCO CUENCA JIMÉNEZ, FACULTAD DE INGENIERÍA
DR. OCTAVIO DÍAZ HERNÁNDEZ, FACULTAD DE INGENIERÍA
DR. PATRICIO MARTÍNEZ ZAMUDIO, FACULTAD DE INGENIERÍA
DRA. MARÍA DEL PILAR CORONA LIRA, FACULTAD DE INGENIERÍA

MÉXICO, D.F., AGOSTO 2015

JURADO ASIGNADO:

Presidente: Dr. Francisco Cuenca Jiménez

Secretario: Dra. María del Pilar Corona Lira

1er. Vocal: Dr. Víctor Javier González Villela

2do. Vocal: Dr. Patricio Martínez Zamudio

3er. Vocal: Dr. Octavio Díaz Hernández

Lugar o lugares donde se realizó la tesis: Facultad de Ingeniería, UNAM

TUTOR DE TESIS:

DR. VÍCTOR JAVIER GONZÁLEZ VILLELA

_____

FIRMA

AGRADECIMIENTOS

Agradezco en lo que corresponde a la DGAPA, por el apoyo brindado para la realización de este trabajo, a través del proyecto UNAM-DGAPA-PAPIIT IN117614: “Robótica intuitiva, adaptable, reactiva, híbrida y móvil aplicada al servicio, el rescate y la medicina”

Así también, se extiende un agradecimiento al programa de becas para realizar estudios de maestría de CONACYT por el apoyo económico que permitió dirigir la atención directamente a la realización de este proyecto

Índice

RESUMEN

Este trabajo desarrolló el estudio de un manipulador móvil constituido por un brazo manipulador serial de 5GDL unido en serio con un robot móvil omnidireccional redundante. El estudio comprendió el análisis cinemático, se utilizó un control de movimientos por campos potenciales para la plataforma móvil, se elaboró un algoritmo de coordinación basado en los principios de la intuición artificial, se construyó un prototipo funcional, se implementó un ambiente inteligente y dentro de éste, se realizó la tarea de llevar un objeto de un lugar a otro; siendo estos temas de gran importancia para la robótica móvil.

Las trayectorias descritas por el robot móvil en las pruebas fueron calculadas activamente por el control por campos potenciales, es decir que nunca fueron planeadas previamente. El movimiento del manipulador serial fue definido por un espacio geométrico y fue coordinado siguiendo los principios de la intuición artificial.

Los resultados de las pruebas realizadas presentaron un movimiento continuo por parte de ambos robots durante la tarea. A su vez, la trayectoria descrita por el robot móvil presentó un movimiento similar a una línea recta entre su posición y el objeto. El brazo manipulador logró tomar el objeto, llevarlo y soltarlo en la posición objetivo final. El uso de un ambiente inteligente disminuyó el procesamiento para la identificación del espacio de trabajo en el que se encontró inmerso el manipulador móvil omnidireccional redundante.

ABSTRACT

This work developed the study of a mobile manipulator consisting of a serial 5DOF manipulator arm attached serially with a redundant omnidirectional mobile robot. The study included kinematic analysis, control movements by potential fields for the mobile platform was used, an algorithm of coordination based on the principles of artificial intuition was developed, a working prototype was built, an intelligent environment was implemented and within this, the task of taking an object from one place to another; being these issues of great importance for mobile robotics.

The trajectories described by the mobile robot in the tests were actively calculated by the control potential fields, ergo they were never previously planned. Serial manipulator movement was defined by a geometric space and was coordinate by following the principles of artificial intuition.

The results of tests showed a continuous movement by both robots during task. In turn, the trajectory described by the mobile robot provided a similar straight line movement between its position and the target position. The manipulator arm managed to take the object, and drop it into the final objective position. Using intelligent environment processing declined to identify the workspace in which the redundant omnidirectional mobile manipulator found itself.

CAPÍTULO 1 INTRODUCCIÓN

1.1 MANIPULADORES SERIALES

La robótica es la técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales (*definición robótica*, *Real Academia Española*) y un robot es una maquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas (*definición robot, Real Academia Española*).

La robótica ha desempeñado un papel muy importante en el desarrollo industrial y tecnológico porque ha reemplazado la intervención humana en tareas repetitivas, tareas difíciles y/o peligrosas para el ser humano. En la industria esto se ve reflejado en un incremento en la eficiencia de la producción.

Los robots utilizados en la industria son los robots industriales y se definen como un manipulador programable en tres o más ejes multipropósito, controlado automáticamente y reprogramable (ISO Standard 8373: 1994, Manipulating Industrial Robots). Los robots manipuladores son usados para diversas tareas como: operaciones de manufactura, pick and place, posicionamiento de objetos, etc.

Teniendo en cuenta su estructura, los robots manipuladores se pueden clasificar en: robots manipuladores tipo serie, robots manipuladores paralelos o robots manipuladores híbridos (González Villela y Martínez-Zamudio, 2009).

Mecánicamente un robot se forma por los eslabones y las articulaciones. Las articulaciones son los elementos mecánicos que permiten el movimiento relativo entre sus eslabones.

Las articulaciones delimitan en gran medida el espacio de trabajo del robot. Las articulaciones más comunes en los robots son las articulaciones rotacionales, y las articulaciones prismáticas. En la figura tres encontramos las articulaciones los diagramas de distintos tipos de articulaciones y los grados de libertad que permiten cada una de estas.

Figura 1.1 Robot Manipulador Antropomórfico

Los robots manipuladores tipo serie están formados por una cadena cinemática abierta. Son diseñados como una serie de eslabones conectados por eslabones actuados que se extienden de una base a un efector final. En la Figura 1.1 se muestra un robot manipulador antropomórfico.

Los tipos básicos de robots seriales, de acuerdo a la configuración de sus ejes (Introductory robotics, Selig), está determinada por el tipo de las 3 primeras articulaciones y son:

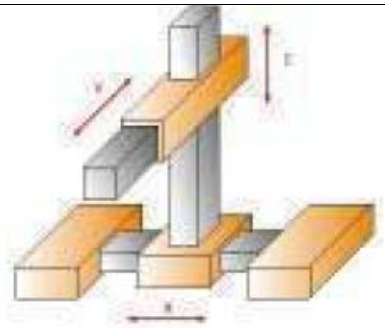
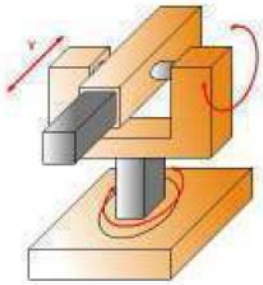
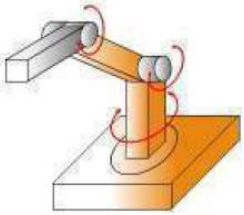
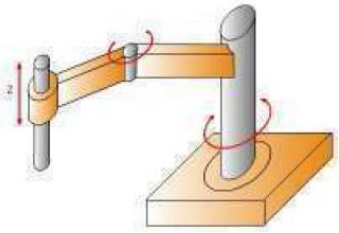
Robot cartesiano Sus tres articulaciones principales son prismáticas, los ejes son ortogonales ente si y los desplazamientos sobre ellos dan las coordenadas cartesianas X, Y, Z de los puntos de trabajo.	
Robot esférico o polar La primera y segunda articulación son de ejes de rotación perpendiculares entre sí, la tercera es prismática; de esta manera se tienen dos giros y un desplazamiento que permiten posicionar un punto en el espacio mediante coordenadas polares.	
Robot antropomórfico Tiene sus 3 principales articulaciones de tipo rotacional, con lo cual emplea coordenadas angulares para determinar las posiciones de su elemento terminal.	
Robot SCARA Es un robot con dos articulaciones rotacionales y una prismática, con las dos rotaciones se controla la posición respecto al plano X-Y y con la prismática la coordenada Z.	

Tabla1.1 Manipuladores Seriales.

1.2 MOVILES OMNIDIRECCIONALES

Los robots móviles son robots con sistemas de locomoción, por lo general ruedas, orugas o patas que permiten el desplazamiento libremente de un lugar a otro en un área de trabajo. Este tipo de robots cuentan con una gran capacidad de desplazamiento de esta forma su espacio de trabajo aumenta considerablemente. Existen múltiples tipos de robots móviles, en el presente trabajo solo nos enfocaremos en robots móviles que emplean ruedas; definiéndolos como un vehículo capaz de moverse de manera autónoma sobre una superficie, mediante la acción de las ruedas montadas sobre el robot.

Para los robots móviles con ruedas se asumen las siguientes hipótesis:

- Se mueve sobre una superficie plana horizontal, es decir, la energía potencial es constante.
- No existen elementos flexibles en la estructura del robot, incluyendo a la rueda.
- El contacto entre cada rueda y el suelo se reduce a un punto.
- No existe deslizamiento.

La forma común de desplazamiento de robot móvil utiliza el método diferencial impulsado, donde dos ruedas motorizadas independientes impulsan el robot en movimiento. Cuando ambas ruedas giran a la misma velocidad, el robot se mueve en un movimiento lineal, ya sea hacia adelante o hacia atrás. Los robots cambian de dirección al girar cada rueda a velocidades diferentes. Este tipo de dirección tiene varias desventajas. En primer lugar, las ruedas pueden girar en una sola dirección en un momento dado limitar sus capacidades de navegación. En segundo lugar, el robot debe cambiar su orientación se requiere movimiento giratorio. En general, esto aumenta la complejidad en la planificación de rutas durante la navegación.

Los vehículos robóticos están comúnmente diseñados para un movimiento plano, es decir, en un espacio de dos dimensiones (2D), un cuerpo de tres grados de libertad (3 GDL). Se pueden trasladar sobre los ejes “x” y “y” y pueden rotar sobre su centro de gravedad, el eje “theta” (ver figura 1.2).

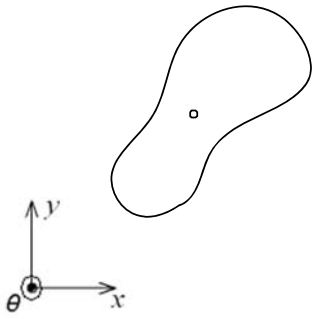


Figura 1.2 3GDL: x, y, theta

La mayoría de los vehículos no son capaces de controlar estos tres grados de libertad de forma independiente, debido a que los robots móviles constituyen una clase de sistemas mecánicos caracterizados por restricciones cinemáticas no-holonómicas, entre las velocidades en el punto de contacto y las velocidades angulares, las cuales no son integrables en forma de relaciones algebraicas entre las variables de desplazamiento traslacional y rotacional, por lo tanto no pueden ser eliminadas de las ecuaciones del modelo.

En contraste, un vehículo que no es obstaculizado por estas restricciones es capaz de una movilidad omnidireccional. Los robots omnidireccionales o también conocidos como robots holonómicos, pueden moverse en cualquier dirección permisible por el medio en cualquier orientación. Un robot se dice que es holonómico si puede modificar su dirección instantáneamente y sin la necesidad de haber rotado previamente, esta consideración no toma en cuenta a la masa del robot.

En muchos casos donde los robots móviles son puestos en acción, especialmente en espacios confinados o congestionados, un movimiento omnidireccional es altamente ventajoso, hay un decremento a la complejidad del control del sistema y permite un movimiento más rápido y eficaz.

Lograr que un robot sea omnidireccional no es algo trivial, dado que el movimiento de avance de una rueda convencional está restringido a un solo sentido. Por ello, algunos investigadores proponen la fabricación de ruedas especiales que permiten el movimiento del vehículo en cualquier dirección, como la rueda sueca o la rueda universal mencionada por Francisco Cuellar (Cuellar).

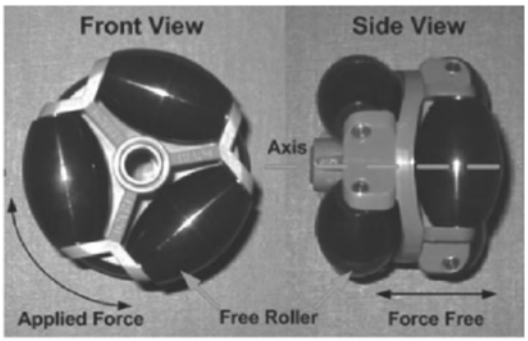
a)  b) 

Figura 1.3 a) Rueda sueca, b) Rueda universal

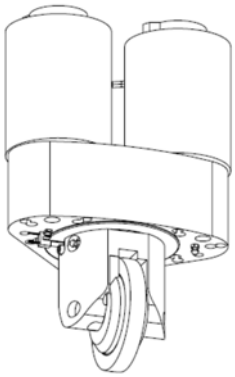
La implementación de tres de éstas últimas, por ejemplo, en un arreglo triangular permite dicho movimiento, además de simplificar considerablemente el análisis cinemático del robot. Algunos robots triangulares desarrollados por (Baede, 2006) y (T. W. Kang, 2011) muestran una navegación exitosa en cualquier dirección, manteniendo la omnidireccionalidad deseada.

No obstante, estas ruedas son de manufactura complicada, no están pensadas para avanzar en superficies irregulares, además de que presentan inestabilidad dado que en ciertos puntos perimetrales se pierde el contacto con el suelo. Esto da lugar a que no se dejen de lado los sistemas de tracción convencionales.

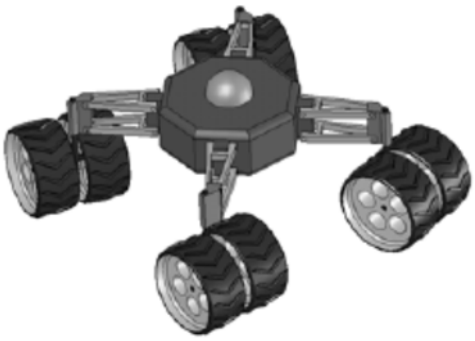
A partir de analizar la cinemática de un vehículo que tenga una configuración de ruedas cualquiera que le permita girar, se puede comprobar que cualquier punto que no esté situado en el eje transversal de las ruedas es capaz de seguir cualquier trayectoria continua. Bajo este principio funcionan las *ruedas castoras*, que son las que encontramos en los carritos del supermercado o en las sillas de oficina.

(Holmberg, 1999) y (Wada) proponen para sus respectivas aplicaciones modelos de castora automática que permita desplazar al robot de manera que éste sea holonómico, esto motorizando la rueda en sus rotaciones paralela y perpendicular al suelo. (Udengaard, 2008) y (Yamada, 2001) utilizan el principio de la castora con dos ruedas unidas entre ellas y con tracción independiente, que a su vez tienen una unión descentrada y libre de giro con respecto al robot. El giro en el punto de apoyo del robot se logra mediante la diferencia de velocidades en las ruedas.

a)



b)



c)

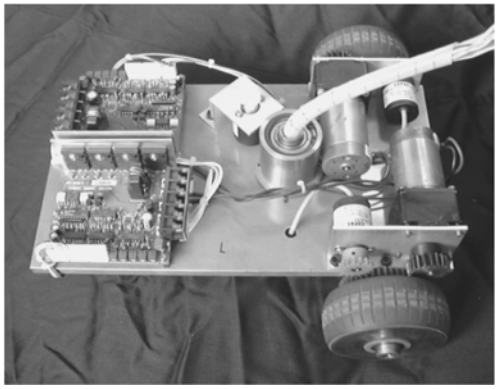


Figura 1.4 a) Castora automática (Holmberg) b) Robot omnidireccional todo terreno (Udengaard) c) Robot de tracción diferencial para robot omnidireccional (Yamada)

Todas estas alternativas son más factibles para aplicarse en vehículos fuera de laboratorio, particularmente la de (Udengaard, 2008) que está pensado expresamente para ello. No obstante, en los últimos dos trabajos mencionados, el análisis matemático que comprueba la holonomicidad no queda enteramente claro y sólo aplica para ese caso en particular.

Por otro lado, (González-Villela, A Unifying Theory on Conventional Wheeled Mobile Robots. Research on semiautonomous mobile robots for loosely structured environments focused on transporting mail trolleys, 2006) propone una teoría unificadora para el análisis cinemático y dinámico de robots con ruedas convencionales. Los trabajos propuestos por (Udengaard, 2008) y (Yamada, 2001) utilizan configuraciones no convencionales, pero parten de las mismas bases que el análisis para uno convencional.

De igual forma, en (González-Villela, 2013) se propone el empleo de un robot móvil omnidireccional no convencional, el cual se forma por la unión de dos robots móviles diferenciales conectados a una plataforma triangular por medio de una junta rotatoria. En el trabajo se verifica la teoría unificadora de (González-Villela, A Unifying Theory on Conventional Wheeled Mobile Robots. Research on semiautonomous mobile robots for loosely structured environments focused on transporting mail trolleys, 2006) para robots no convencionales.

Figura 1.5 Plataforma móvil omnidireccional

En este proyecto se aborda el empleo de este robot móvil omnidireccional llevándolo a la redundancia, es decir, se formará mediante cuatro robots móviles diferenciales, lo cual genera redundancia en la plataforma móvil. El estudio de la redundancia en robots móviles omnidireccionales presenta un amplio campo de oportunidad para nuevas investigaciones, debido a que las investigaciones se centran en el uso y desarrollo de ruedas y mecanismos especiales para la generación de omnidireccionalidad.

1.3 COORDINACIÓN DE MOVIMIENTOS

Teniendo los robots manipuladores y los robots móviles presentes, al combinar más de un tipo de robot obtenemos un robot híbrido, por ejemplo el unir un manipulador serial con un móvil. Al realizar esta fusión, se obtiene como resultado un manipulador móvil, el cual suma las problemáticas de ambas partes haciendo más complejo su control, pero a su vez, aumenta indefinidamente el área de trabajo del manipulador serial; permitiendo desempeñar simultáneamente las tareas de locomoción y manipulación. Aun cuando se tienen las dos partes solucionadas, surge una nueva serie de problemas donde la coordinación de ambas partes es una de las problemáticas que ha atraído la atención de los investigadores en la actualidad.

Para la coordinación de movimientos entre el brazo manipulador y el robot móvil existen muchos trabajos que modelan su control por separado. En (O. Khatib, 1996) se desarrolla un control a nivel dinámico para seguir una trayectoria en el espacio de tarea, pero usa la arquitectura de dos controladores en paralelo. Por otro lado en (Weiming Ge, 2008) el control propuesto se basa en dos esquemas diferentes de modos deslizantes para la base móvil y del manipulador.

También existen controles que consideran al manipulador móvil como un sistema integrado. En (M. Boukattaya, 2009) se usa un control en el espacio de la tarea usando par calculado, pero se cierra el lazo de control por medio de un controlador difuso. En (Tang Hong, 2009) modela la incertidumbre del manipulador móvil por medio de una red neuronal y el control se realiza por medio de técnicas de modos deslizantes.

La Coordinación de la planificación de movimiento del punto final del manipulador y la base móvil es una tarea crucial para un control de robot móvil. En (Y. Yamamoto, 1994) y (S. Bhowmik, 2007) se presenta un algoritmo de control para la plataforma cuando el movimiento del punto extremo del manipulador es desconocido a priori, en donde la planificación de la trayectoria tiene que ser hecha localmente y en tiempo real, tomando el desplazamiento medido de la articulación del manipulador como la entrada para la planificación de movimiento y controla la plataforma para llevar el manipulador en una región de operación preferida medida por su manipulabilidad.

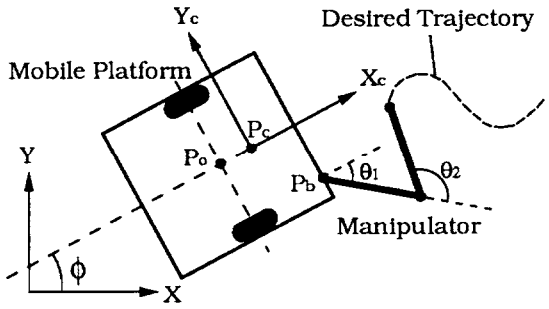


Figura 1.6 Configuración del móvil utilizada por (Y. Yamamoto, 1994) *y* (S. Bhowmik, 2007)

En (H. G. Tanner, 2000) se presenta un planificador de movimiento de un manipulador móvil que se mueve en el ambiente al aire libre entre los árboles. El enfoque se basa en una ley de realimentación discontinua bajo la influencia de un campo potencial especialmente diseñado para coordinar el movimiento de la base móvil y el manipulador a bordo que cuenta con 3 juntas rotacionales y evitar la colisión de obstáculos.

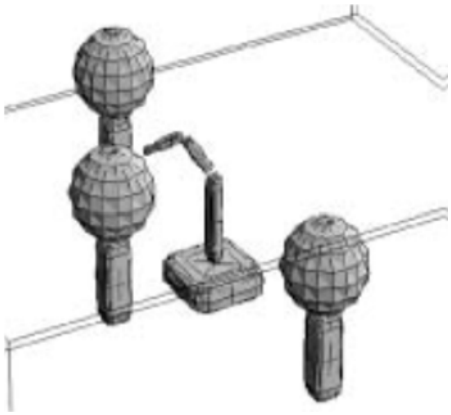


Figura 1.7 Simulación de (H. G. Tanner, 2000)

En (D. H. Shin, 2003) se ha propuesto un método de planificación de movimiento que se basa en el paradigma de que los movimientos plataforma móvil, establece una posición y se detiene, momento en el que el manipulador mueve su efector final para el seguimiento de una parte de la ruta, seguido por la plataforma móvil reposicionarse para el manipulador para rastrear la siguiente parte de la trayectoria. Esto permite maximizar la función de rendimiento a lo largo de la ruta.

De igual manera, en (Escobedo-Castillo, 2012) se busca realizar una coordinación entre un robot móvil con ruedas y un manipulador serial de 5GDL acoplado. La finalidad de la coordinación es cumplir una tarea de manera sencilla, con un algoritmo de coordinación heurístico de forma que no se necesite un análisis dinámico del sistema para que se pueda llevar a cabo.

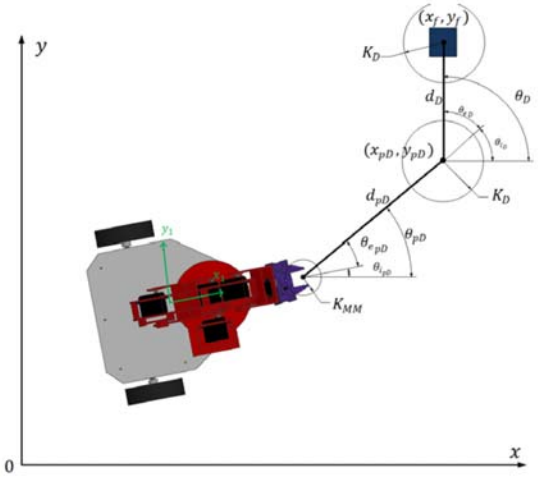


Figura 1.8 Planeación de trayectoria de (Escobedo-Castillo, 2012)

1.4 DEFINICIÓN DEL PROBLEMA

La robótica híbrida es la combinación de dos o más estructuras robóticas que toman ventaja de las propiedades de cada estructura robótica. Dentro de la robótica móvil, los manipuladores móviles (compuestos por un robot móvil y un manipulador) encuentran que la coordinación de movimientos de los robots que lo componen es compleja.

Este trabajo presenta el estudio de un manipulador móvil que lo constituye la unión en serie de un brazo manipulador serial y un robot móvil holonómico. El estudio comprende el análisis cinemático, elaboración de un algoritmo de coordinación, y construcción de un prototipo de pruebas para la mejor comprensión de su comportamiento físico.

1.4.1 OBJETIVO

Desarrollar, construir y coordinar un prototipo funcional de un robot manipulador móvil omnidireccional redundante, así como el estudio de sus movimientos y generación de trayectorias para el transporte de un objeto de un lugar a otro dentro de un ambiente inteligente.

1.4.2 HIPÓTESIS

El análisis cinemático de la nueva configuración propuesta, el control de movimiento por campos potenciales y el algoritmo de coordinación del manipulador serial y la base móvil usando el principio de intuición artificial dentro de un ambiente inteligente, brindará herramientas de estudio de los robots manipuladores móviles.

1.4.3 ALCANCES

Probar el algoritmo de coordinación de movimientos y observar el comportamiento mecánico sobre un prototipo del robot manipulador móvil omnidireccional redundante propuesto dentro de un ambiente inteligente.

1.4.4 JUSTIFICACIÓN

El manipulador móvil estudiado está compuesto por un manipulador serial de 5GDL unido en serie con un robot móvil omnidireccional redundante. Este manipulador móvil está dentro de la clasificación de robots híbridos y cuenta con redundancia; tanto en la parte móvil como en la de manipulación.

La configuración propuesta incrementa el espacio de trabajo diestro de un manipulador serial mediante la adición en serie de un robot móvil. La coordinación de estos tipos de robots es un tema que se investiga en la actualidad y sigue mostrando ser un campo de oportunidad para nuevas investigaciones.

Las ecuaciones de control por campos potenciales para el robo móvil, el principio de intuición artificial utilizado para la movilidad del manipulador, la nueva configuración del robot móvil omnidireccional redundante y la coordinación de dos robots redundantes para la generación de movimientos; son los temas de gran importancia para la robótica móvil que abarca este trabajo.

1.4.5 METODOLOGÍA

Para este trabajo la metodología que sigue se define en 6 capítulos:

Capítulo 1 Introducción
Primero se describe el campo de la robótica móvil, haciendo un enfoque en los manipuladores móviles y una revisión del estado del arte sobre la coordinación que se plantea para llevar a cabo una correcta manipulación del sistema que presenta redundancia.

Capítulo 2 Análisis cinemático
Después se realiza el análisis cinemático para la propagación de las velocidades del robot móvil a las ruedas. Esto es, que las velocidades deseadas del punto de análisis puedan ser traducidas en velocidades de giro de las ruedas, a través de la cadena cinemática formada por el eslabonamiento de los cuerpos que integran todo el robot. Se determina el tipo de vehículo que es, para comprobar la omnidireccionalidad del mismo. El análisis cinemático del manipulador serial es efectuado mediante el uso de matrices de transformación la cual nos da la cinemática directa del brazo manipulador. Se definen los ángulos que se encargarán del posicionamiento y los ángulos que orientarán al efector final.

Capítulo 3 Diseño y construcción del prototipo
Posteriormente se diseña y construye un modelo de funcionalidad donde se programarán las ecuaciones cinemáticas y de control para poder observar el comportamiento de la plataforma y verificar la respuesta que se genera por el algoritmo de coordinación.

Capítulo 4 Generación de movimiento
Ya con el modelo cinemático obtenido, se propone una ley para controlar a la plataforma, con la intención de que sea totalmente automatizada. Se realiza de igual manera la definición del lugar geométrico del manipulador serial y se define el algoritmo de coordinación; basándose en el principio de intuición artificial.

Capítulo 5 Descripción del ambiente inteligente y del sistema de pruebas
Sólo queda la descripción del sistema de pruebas, el cual es definido por un ambiente inteligente. Se describe la arquitectura del ambiente inteligente y del sistema de pruebas, listando los componentes y definiéndolos.

Capítulo 6 Pruebas y resultados
Por último, se recopila toda la información que sea generada por la tarea que se define , se analiza y se obtienen las conclusiones de los experimentos.

CAPÍTULO 2 ANÁLISIS CINEMÁTICO

2.1 INTRODUCCIÓN

Este capítulo comprende el análisis cinemático del robot móvil omnidireccional redundante. El análisis se separa en el análisis cinemático del robot móvil diferencial y en el análisis de la plataforma omnidireccional. Posteriormente se encontrará la solución de la cinemática inversa del robot móvil omnidireccional redundante mediante una superposición.

2.1.1 PROPAGACIÓN DE VELOCIDADES

Para poder llevar a cabo el análisis, es necesario usar el método de propagación de velocidades, descrito por (Craig, 2003). Este método estudia la propagación de las velocidades por las articulaciones de un robot. Los eslabones entre dos articulaciones de cuerpos rígidos con movimiento que es descrito por vectores de velocidades lineales y angulares.

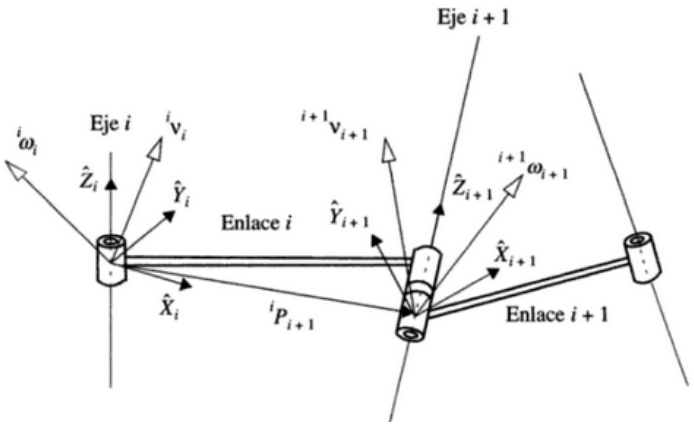


Figura 2.1 Vectores de velocidad de los eslabones colindantes.

La velocidad del eslabón $\{i+1\}$, es la del eslabón i mas cualquier otra componente de velocidad que agregue la junta $i+1$. La velocidad lineal del origen del sistema $\{i\}$ se representa por v_i y la velocidad angular de este sistema por ω_i. Las velocidades de rotación se agregan cuando los vectores ω_i se trazan en el mismo marco coincidiendo con el eje Z. Por lo que la velocidad angular del eslabón $i+1$, es la misma que la del eslabón i, y en la que agrega un nuevo término producido por la velocidad de rotación de la junta $i+1$. Y así queda expresado en el marco de referencia $\{i\}$:

$$ {}^{i}\omega_{i+1} = {}^{i}\omega_i + {}^{i}_{i+1}R\,\dot{\theta}_{i+1}\,{}^{i+1}Z_{i+1} \tag{2.1} $$

Siendo $\dot{\theta}_{i+1}$ la derivada con respecto al tiempo de la variable articular, en forma vectorial, se representa como:

$$ \dot{\theta}_{i+1}\,{}^{i+1}Z_{i+1} = {}^{i+1}\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix} \tag{2.2} $$

Por otro lado ${}^{i}_{i+1}R$, la matriz que rota el eje de $\{i+1\}$ a su descripción en $\{i\}$ para poder sumar las dos componentes de velocidades angulares.

La velocidad angular del eslabón $i+1$ expresado en $\{i+1\}$ queda como:

$$ {}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\ {}^{i}\omega_i + \dot{\theta}_{i+1}\ {}^{i+1}Z_{i+1} \tag{2.3} $$

Por otro lado, en la misma articulación, la velocidad lineal del origen del marco $\{i+1\}$ es la misma que la del origen del marco $\{i\}$ más una componente que se origina por la velocidad angular del eslabón i, teniendo:

$$ {}^{i}v_{i+1} = {}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1} \tag{2.4} $$

Siendo ${}^{i}P_{i+1}$ el vector constante que expresa las coordenadas de posición del origen del marco $\{i+1\}$ en el sistema $\{i\}$.

La expresión resultante de la velocidad sobre el eslabón $i+1$ con respecto al sistema $\{i+1\}$ es:

$$ {}^{i+1}v_{i+1} = {}^{i+1}_{i}R\left({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}\right) \tag{2.5} $$

Las ecuaciones resultantes (2.3) y (2.5) son válidas cuando la junta de unión es rotacional. De esta manera, se pueden obtener las velocidades lineales y angulares de cada eslabón del sistema, en su mismo marco de referencia.

2.2 ARQUITECTURA DEL MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

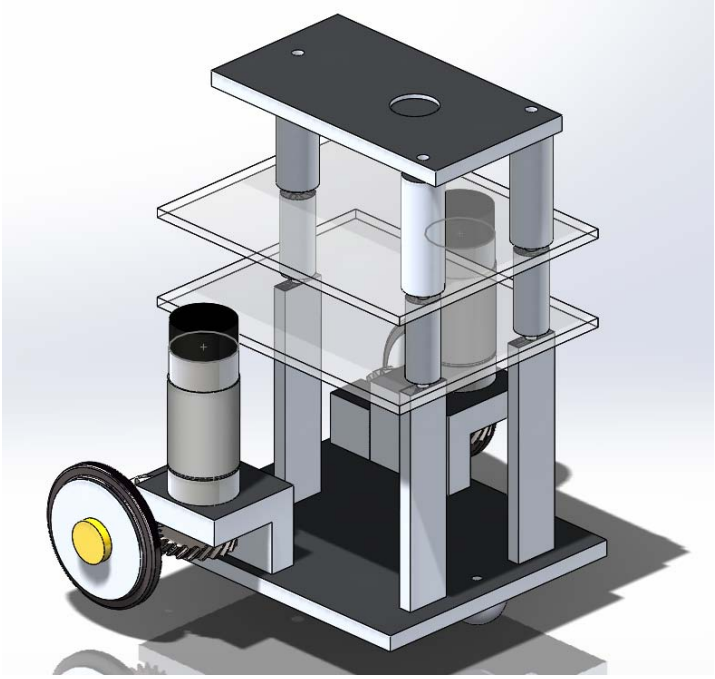


Figura 2.2 Robot Móvil Diferencial Tipo (2,0)

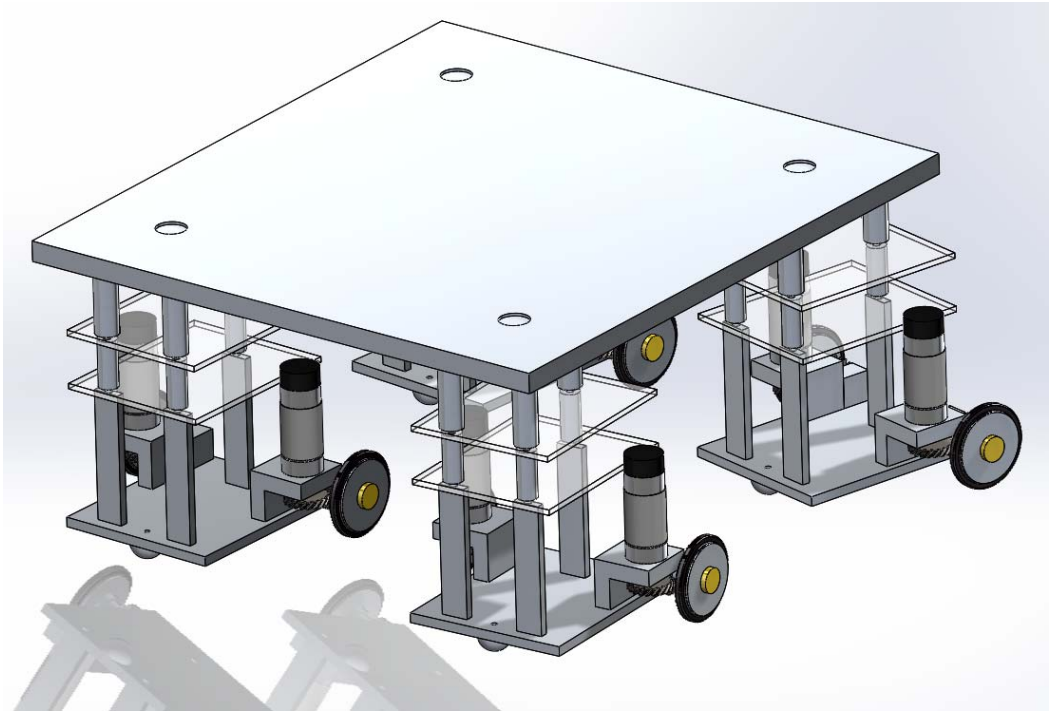


Figura 2.3 Robot Móvil Omnidireccional Redundante

Los robots móviles diferenciales, están constituidos por una configuración tipo (2,0), la cual cuenta con dos llantas traseras motorizadas fijas y una rueda frontal pasiva.

A la plataforma móvil, de forma rectangular, la constituyen 4 robots móviles diferenciales posicionados en cada una de sus esquinas, los cuales se encuentran fijos con una unión rotacional.

Cada robot móvil diferencial, por la naturaleza de su configuración, no son omnidireccionales por si mismos pero con tener al menos 2 robots móviles diferenciales unidos a una plataforma, se obtiene la característica de omnidireccionalidad. En este caso, al poseer 4 robots móviles diferenciales, tenemos dicha omnidireccionalidad y a su vez el sistema se vuelve redundante.

Finalmente, se puede decir que la plataforma y los 4 robots móviles diferenciales constituyen a lo que es el robot móvil omnidireccional redundante. Cuenta con 3 grados de libertad, permitiéndole desplazarse sobre el plano “XY” y a su vez tener una rotación respecto al eje “Z”.

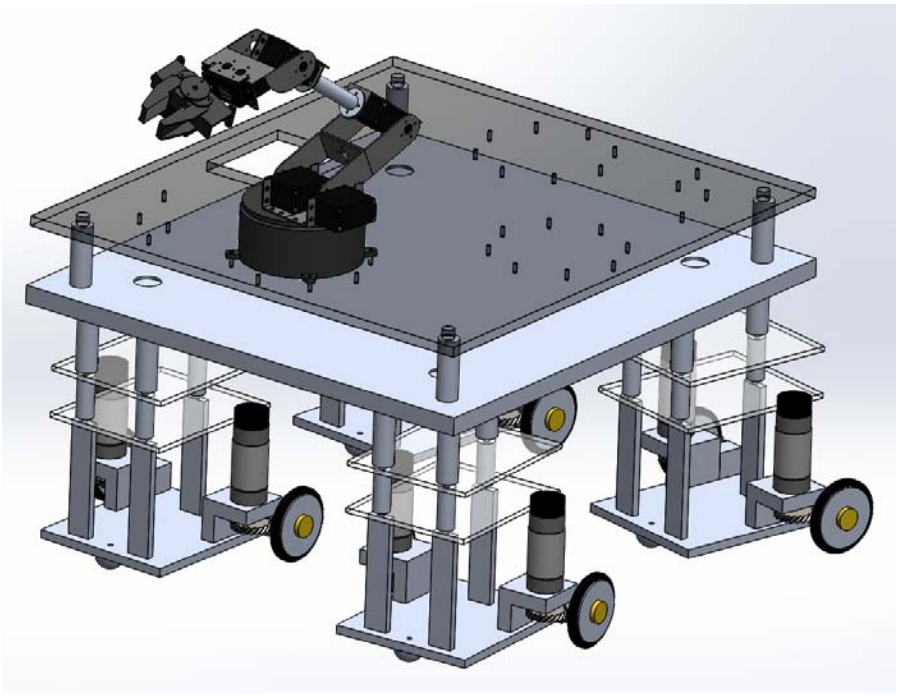


Figura 2.4 Manipulador Móvil Omnidireccional Redundante

Sobre el robot móvil omnidireccional redundante se colocó una placa de acrílico a un nivel diferente para poder colocar un manipulador serial de 5GDL (grados de libertad) el cual se encargará de la acción de "*Pick and Place*". El manipulador serial de 5GDL se encuentra ubicado en lo que se definió como la parte frontal del robot móvil omnidireccional redundante, cuyo origen se encuentra ubicado sobre uno de los ejes cartesianos del plano de referencia que se encuentra en el centro de la plataforma.

2.3 MODELO DEL ROBOT MÓVIL OMNIDIRECCIONAL REDUNDANTE

2.3.1 ANÁLISIS DE LA POSICIÓN

El robot móvil omnidireccional redundante, está compuesto por cuatro robots móviles diferenciales fijados a cada una de las esquinas de una plataforma rectangular por una junta rotacional. Debido al posicionamiento de cada uno de los robots móviles diferenciales, se realizó el análisis cinemático de uno de ellos y se extrapoló para los tres restantes.

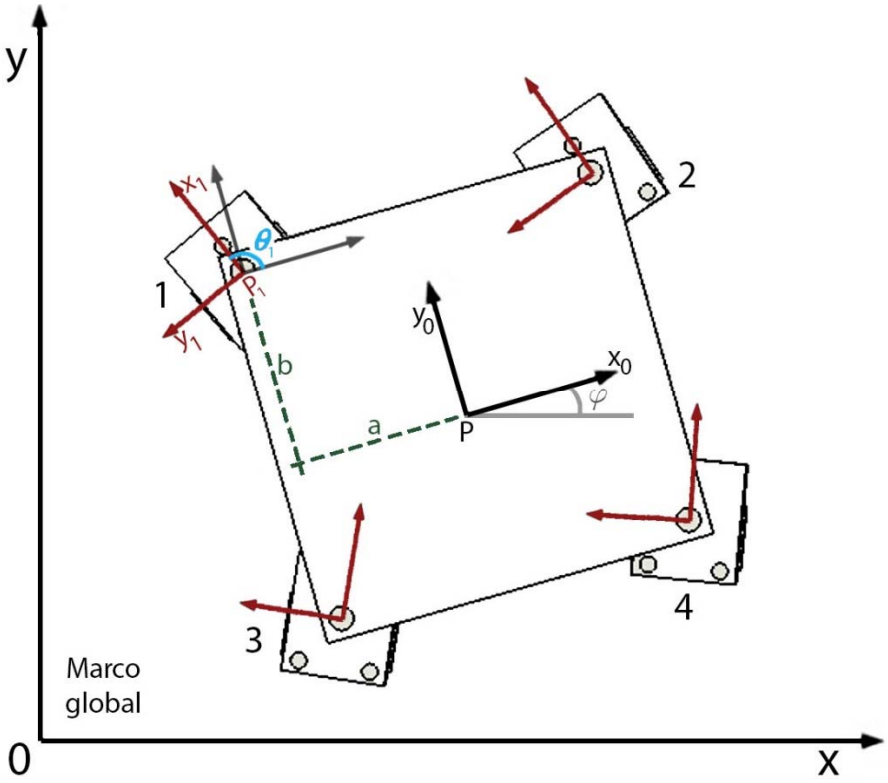


Figura 2.5 Postura del robot móvil omnidireccional redundante

Para el análisis cinemático se tiene que tener en cuenta en todo momento la orientación de todos los robots móviles diferenciales. En este caso, se definió un frente de la plataforma y se enumeraron arbitrariamente cada uno de los robots móviles diferenciales.

En la figura 2.5 se muestra la posición de los componentes del robot móvil omnidireccional redundante. También se muestra el marco de referencia global y a su vez, el marco de referencia local que se encuentra ubicado al centro de la plataforma rectangular al cual se encuentran referidos cada uno de los robots móviles diferenciales.

Cada uno de los robots móviles constituidos por una configuración tipo (2,0), teniendo dos llantas traseras motorizadas fijas y una rueda frontal pasiva, la cual fue ignorada para el análisis. En la figura 2.5 se muestra la configuración del robot móvil.

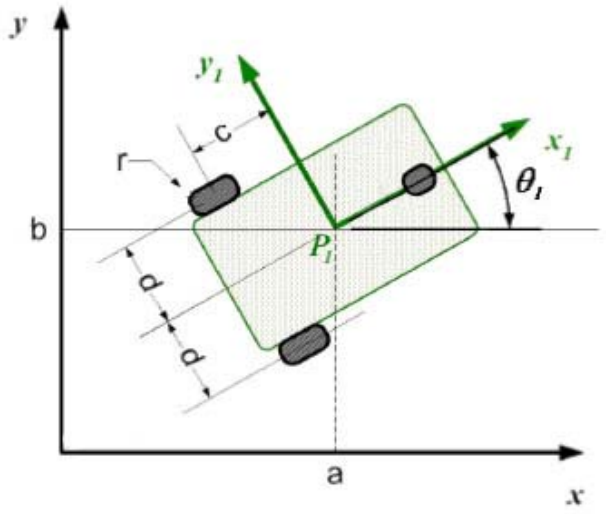


Figura 2.6 Postura del robot tipo (2,0)

Como se puede observar, el centro de referencia local de cada robot móvil diferencial se encuentra a coordenadas constantes (a, b) del centro de la plataforma. De esta forma, se necesita un vector de posición para llevar el plano de orientación de los robots diferenciales, quedando los vectores de la siguiente manera.

$$P_{P_1} = \begin{pmatrix} -a \\ b \\ 0 \end{pmatrix} \tag{2.6}$$

$$P_{P_2} = \begin{pmatrix} a \\ b \\ 0 \end{pmatrix} \tag{2.7}$$

$$P_{P_3} = \begin{pmatrix} -a \\ -b \\ 0 \end{pmatrix} \tag{2.8}$$

$$P_{P_4} = \begin{pmatrix} a \\ -b \\ 0 \end{pmatrix} \tag{2.9}$$

Matriz de Rotación desde el centro de la plataforma (punto P) a cada una de las esquinas donde se encuentran cada uno de los robots diferenciales, para poder tener así un plano cartesiano con la orientación de cada robot.

$$R_{iP} = \begin{bmatrix} cos(\varphi + \theta_i) & sin(\varphi + \theta_i) & 0 \\ -sin(\varphi + \theta_i) & cos(\varphi + \theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

$$donde\ i = 1,2,3,4$$

$$donde\ \varphi = \text{orientacion de la plataforma}$$

$$donde\ \theta_i = \text{orientacion del robot diferencial i con respecto a la plataforma}$$

2.3.2 PROPAGACIÓN DE VELOCIDADES

Tomando en cuenta lo descrito en la introducción del capítulo, se pueden obtener las ecuaciones cinemáticas que definen al sistema. Para ello se tiene que considerar que el vector de velocidad que define al sistema es el siguiente:

$$\dot{\xi}_1 = \begin{bmatrix} v_{x_P} & v_{y_P} & \dot{\theta_P} \end{bmatrix}^T \tag{2.11}$$

Este vector representa la postura en velocidad del marco de la plataforma móvil, respecto a ella misma, donde las velocidades lineales del punto P están representadas por v_{x_P} y v_{y_P}, y la velocidad angular por $\dot{\theta_P}$. Para que se exprese en el marco global se tiene que realizar una transformación, definiendo así a las velocidades generalizadas.

$$\dot{\xi} = R^T(\theta_P)\,\dot{\xi}_1 \tag{2.12}$$

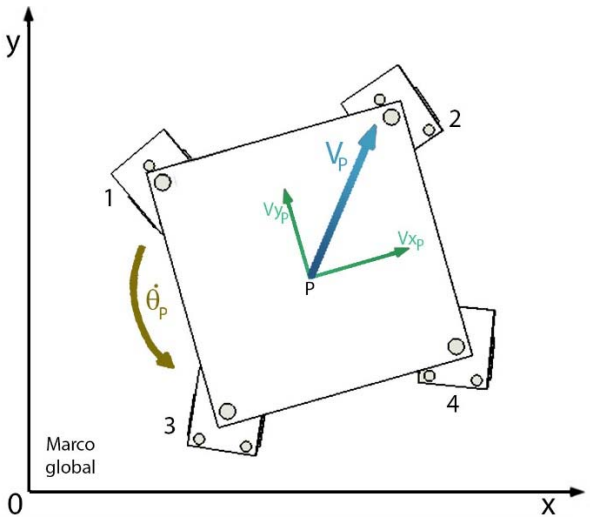


Figura 2.7 Velocidad del Robot Móvil Omnidireccional Redundante

De esta forma se tienen que la velocidad lineal y la velocidad angular del punto P; el centro de la plataforma, que se expresarán de la siguiente manera:

$$\omega_{PP} = \begin{pmatrix} 0 \\ 0 \\ \theta'_P \end{pmatrix} \tag{2.13}$$

$$v_{PP} = \begin{pmatrix} v_{x_P} \\ v_{y_P} \\ 0 \end{pmatrix} \tag{2.14}$$

Para poder trasladar la velocidad del punto P a cada una de las cuatro esquinas se utiliza un vector de transformación para poder llevar a cabo las operaciones correspondientes.

$$z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.15}$$

Para el caso de la *velocidad angular* se calcula como sigue

$$\omega_{i_i} = R_{iP} \cdot \omega_{PP} + \theta'_i \cdot z \tag{2.16}$$

$$donde\ i = 1,2,3,4$$

Esto nos dice que, la velocidad angular del punto i visto desde el plano cartesiano que se ubica en el punto i, se define como la velocidad angular existente en el punto i adicionado a la velocidad angular del punto P trasladada.

$$\omega_{i_i} = \begin{pmatrix} 0 \\ 0 \\ \theta'_i + \theta'_P \end{pmatrix} \tag{2.17}$$

Para el caso de la *velocidad lineal* en cada una de las esquinas se calcula de la siguiente manera.

$$v_{i_i} = R_{iP} \cdot (v_{PP} + \omega_{PP} \times P_{Pi}) \tag{2.18}$$

$$donde\ i = 1,2,3,4$$

De esta forma, la velocidad lineal de cada una de las esquinas es definida como el traslado de la velocidad lineal y la velocidad angular del punto P reflejada.

$$v_{i_i} = \begin{pmatrix} v_{P_x}cos(\theta_i) + v_{P_y}sin(\theta_i) \pm a\theta'_P sin(\theta_i) \pm b\theta'_P cos(\theta_i) \\ -v_{P_x}sin(\theta_i) + v_{P_y}cos(\theta_i) \pm a\theta'_P cos(\theta_i) \pm b\theta'_P sin(\theta_i) \\ 0 \end{pmatrix} \quad (2.19)$$

$$a\theta'_P sin(\theta_i), a\theta'_P cos(\theta_i)\ cuando\ i = 2{,}4$$

$$-a\theta'_P sin(\theta_i), -a\theta'_P cos(\theta_i)\ cuando\ i = 1{,}3$$

$$b\theta'_P sin(\theta_i), -b\theta'_P cos(\theta_i)\ cuando\ i = 1{,}2$$

$$-b\theta'_P sin(\theta_i), b\theta'_P cos(\theta_i)\ cuando\ i = 3{,}4$$

Ahora, se analiza cada uno de los robots diferenciales a partir de su punto de referencia P_i.

En primera instancia, tenemos que la forma en que se llevara a cabo el análisis de cada robot diferencial, se muestra en la figura siguiente.

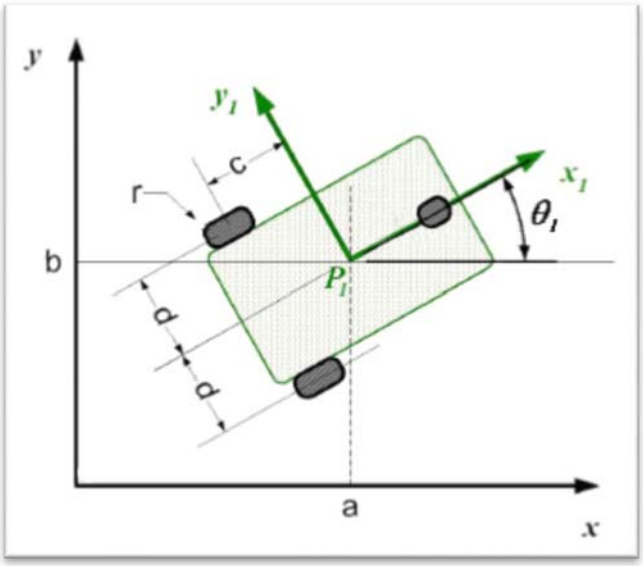


Figura 2.8 Postura del robot tipo (2,0)

De manera análoga se utilizan vectores para analizar el robot móvil diferencial y poder de esta manera utilizar el método de propagación de velocidades. Para ello se utiliza una matriz de rotación y se definen los vectores correspondientes.

$$R_{ij\,i} = \begin{bmatrix} cos(\theta_{ij}) & sin(\theta_{ij}) & 0 \\ -sin(\theta_{ij}) & cos(\theta_{ij}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.20}$$

$donde\ \theta_{ij} =$ orientación de la llanta j con respecto al robot diferencial i.

$donde\ R_{ij\,i} =$ matriz de rotación de i hacia la llanta j con respecto al robot diferencial i.

$$P_{i_{ij}} = \begin{pmatrix} \pm d \\ -c \\ 0 \end{pmatrix} \tag{2.21}$$

d cuando j $= 1, -d$ cuando j $= 2$

$donde\ P_{i_{ij}} =$ vector de traslación de i hacia la llanta j a partir del punto P_i.

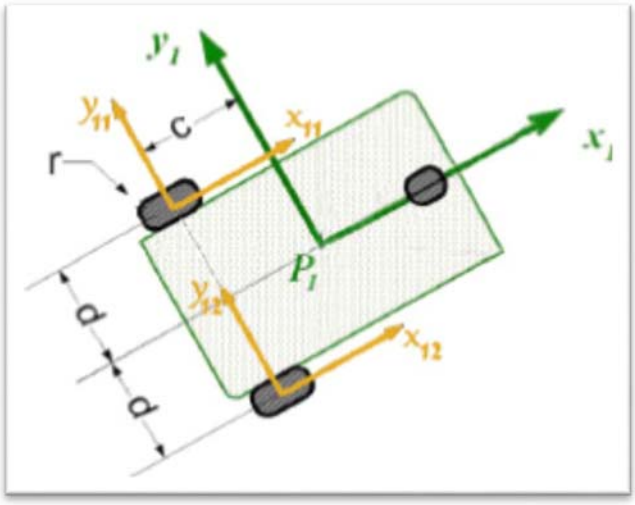


Figura 2.9 Marcos de referencia para las llantas

Para el método de propagación de velocidades hacia cada una de las llantas del robot diferencial, se utiliza la ecuación 2.1 y se redefinen los subíndices para este caso, quedando de la siguiente manera.

$$\omega_{ij_{ij}} = R_{ij\,i} \cdot \omega_{i_i} + \theta'_{ij} \cdot z \tag{2.22}$$

$$donde\ i = 1,2,3,4\ \ j = 1,2$$

Esto nos dice que, la velocidad angular del punto ij visto desde el plano cartesiano que se ubica en el punto ij, se define como la velocidad angular existente en el punto ij adicionado a la velocidad angular del punto i trasladado.

$$\omega_{ij_{ij}} = \begin{pmatrix} 0 \\ 0 \\ \theta'_{ij} + \theta'_{i} + \theta'_{P} \end{pmatrix} \tag{2.23}$$

Para el caso de la velocidad lineal en cada una de las llantas del robot diferencial, se reutiliza de forma análoga la ecuación planteada anteriormente 2.17 y se calcula de la siguiente manera.

$$v_{ij_{ij}} = R_{ij\,i} \cdot \left(v_{i_i} + \omega_{i_i} \times P_{i_{ij}}\right) \tag{2.24}$$

$$donde\ i = 1{,}2{,}3{,}4\ \ j = 1{,}2$$

De esta forma, la velocidad lineal de cada una de las llantas es definida como el traslado de la velocidad lineal y la velocidad angular del punto i reflejada.

$$v_{ij_{ij}} = \begin{pmatrix} v_{P_x}cos(\theta_i) + v_{P_y}sin(\theta_i) \pm a\theta'_P sin(\theta_i) \pm b\theta'_P cos(\theta_i) + c(\theta'_P + \theta'_i) \\ -v_{P_x}sin(\theta_i) + v_{P_y}cos(\theta_i) \pm a\theta'_P cos(\theta_i) \pm b\theta'_P sin(\theta_i) \pm d(\theta'_P + \theta'_i) \\ 0 \end{pmatrix} \tag{2.25}$$

$$a\theta'_P sin(\theta_i), a\theta'_P cos(\theta_i)\ cuando\ i = 2{,}4$$

$$-a\theta'_P sin(\theta_i), -a\theta'_P cos(\theta_i)\ cuando\ i = 1{,}3$$

$$b\theta'_P sin(\theta_i), -b\theta'_P cos(\theta_i)\ cuando\ i = 1{,}2$$

$$-b\theta'_P sin(\theta_i), b\theta'_P cos(\theta_i)\ cuando\ i = 3{,}4$$

$$d(\theta'_P + \theta'_i)\ cuando\ j = 1$$

$$-d(\theta'_P + \theta'_i)\ cuando\ j = 2$$

Con lo anterior, podemos definir una matriz de velocidad que se encuentre constituida a partir de las velocidades lineales definidas para cada una de las llantas. Donde dicha matriz será de dimensión 16x1, siendo conformada de la siguiente forma.

$$v = \begin{pmatrix} v_{11_{11}1} \\ v_{12_{12}1} \\ v_{21_{21}1} \\ \vdots \\ v_{ij_{ij}1} \\ v_{11_{11}0} \\ v_{12_{12}0} \\ v_{21_{21}0} \\ \vdots \\ v_{ij_{ij}0} \end{pmatrix} para \begin{matrix} i = 1 \dots 4 \\ j = 1{,}2 \end{matrix} \tag{2.26}$$

Donde los subíndices ¨0¨ y ¨1¨ se refieren a los renglones del vector $v_{ij_{ij}}$, en donde se tiene que ¨0¨ está referido al primer renglón el cual contiene la velocidad reflejada sobre el eje "x" y ¨1¨ al segundo renglón del vector describiendo la velocidad reflejada en el eje "y".

Por ejemplo, $v_{11_{11_0}}$se refiere al primer renglón que constituye al vector $v_{11_{11}}$, siendo la ecuación que constituye al renglón de la matriz v y cuyo efecto se ve reflejado en el eje x_1 del diferencial:

$$v_{P_x}cos(\theta_1) + v_{P_y}sin(\theta_1) - a\theta'_P sin(\theta_1) - b\theta'_P cos(\theta_1) + c(\theta'_P + \theta'_1) \quad (2.27)$$

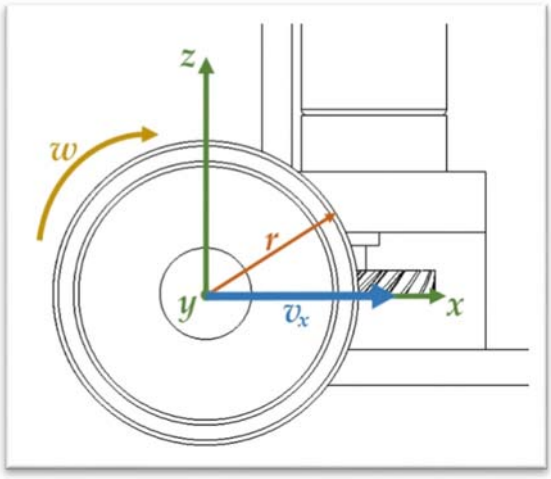


Figura 2.10 Relación de velocidad en las llantas.

En la figura 2.9 se observa que la velocidad angular de las llantas únicamente se refleja con respecto al eje "x" del marco de referencia local. Tomando esto en consideración, se define una matriz de transformación dimensión 16x1, la cual relaciona la velocidad lineal de las llantas con su velocidad angular. En los primeros 8 renglones de la matriz, los elementos serán nulos ya que no afectaran de manera alguna al eje "y", mientras que en el resto de los renglones de la matriz se coloca la relación existente.

$$r_\phi = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ -r \cdot \phi_{11} \\ -r \cdot \phi_{12} \\ -r \cdot \phi_{21} \\ \vdots \\ -r \cdot \phi_{ij} \end{pmatrix} para \begin{matrix} i = 1 \dots 4 \\ j = 1{,}2 \end{matrix} \quad (2.28)$$

$donde\ \phi_{ij} =$ posición angular de la llanta j del robot diferencial i.

$donde\ r =$ radio de las llantas.

De esta manera, se construye el vector de velocidad que define a nuestro sistema como:

$$V = v + r_{\phi} \tag{2.29}$$

Existen principalmente tres tipos de llantas: las fijas (N_f), centradas direccionales (N_c) y descentradas direccionales (N_{oc}) (Escobedo-Castillo, 2012), un robo móvil diferencial tiene un número N_T de llantas convencionales el cual se define como:

$$N_T = N_f + N_c + N_{oc} \tag{2.30}$$

Esto nos permitirá saber el total de coordenadas de configuración (n_T) que son necesarias para definir a cada uno de los robots móviles diferenciales.

$$n_T = 3 + N_T + N_c + N_{oc} \tag{2.31}$$

Sustituyendo la ecuación (2.29) en la (2.30) y simplificando:

$$n_T = 3 + N_f + 2N_c + 2N_{oc} \tag{2.32}$$

En la configuración utilizada; definida anteriormente, se tienen presentes 2 llantas fijas (N_f) y una llanta pasiva la cual no se involucra con el análisis ya que únicamente nos apoya con el equilibrio del robot diferencial y no modifica la cinemática de éste. Por consiguiente no tenemos llantas centradas ($N_c = 0$) ni llantas descentradas ($N_{oc} = 0$). Entonces, se tiene que para un robot diferencial son necesarias 5 coordenadas de configuración.

$$q_{d_i} = \begin{pmatrix} v_{d_{i_x}} & v_{d_{i_y}} & \theta'_i & \dot{\phi_{\imath 1}} & \dot{\phi_{\imath 2}} \end{pmatrix}^T \; donde \; i = 1 \ldots 4 \tag{2.33}$$

Al tener 4 robots móviles diferenciales, el total de coordenadas de configuración serían 20, las cuales se ven reducidas en gran medida debido a las restricciones que se presentan al encontrarse unidos con una junta rotacional a una plataforma. Todas aquellas coordenadas relacionadas con la velocidad lineal de cada robot móvil diferencial se encuentran relacionadas con la velocidad lineal de la plataforma, como se observa en la ecuación (2.25).

Finalmente, podemos definir que para el sistema, se tiene que las coordenadas de configuración o variables de interés, estarán definidas como un vector q_p, el cual se estructura de la siguiente manera:

$$q_p = \begin{pmatrix} v_{P_x} & v_{P_y} & \theta'_P & \theta'_1 & \theta'_2 & \theta'_3 & \theta'_4 & \dot{\phi}_{11} & \dot{\phi}_{12} & \dot{\phi}_{21} & \dot{\phi}_{22} & \dot{\phi}_{31} & \dot{\phi}_{32} & \dot{\phi}_{41} & \dot{\phi}_{42} \end{pmatrix}^T \tag{2.34}$$

Una vez que se tiene el vector de las coordenadas de configuración, se tiene que encontrar la expresión de la cinemática en la forma de variables de estado o espacio de estados. De esta forma se tiene que en el espacio de estados la ecuación tiene que estar de la forma:

$$q_p = S_q(q) * v(t) \tag{2.35}$$

Donde $v(t)$ es el vector de velocidades de entrada del modelo cinemático y $S(q)$ es la matriz que contiene todas las restricciones cinemáticas del sistema y las relaciona con las velocidades de entrada.

Las velocidades de entrada necesarias para controlar el sistema son:

$$v(t) = \begin{bmatrix} v_{P_x} \\ v_{P_y} \\ \theta'_P \end{bmatrix} \tag{2.36}$$

Para el cálculo de la matriz $S(q)$ que se encargará de relacionar las 15 variables de interés que componen al vector q_p se tiene que encontrar en primera instancia una matriz A tal que:

$$A * q_p = 0 \tag{2.37}$$

Donde A es la matriz asociada a las restricciones cinemáticas la cual se encuentra parametrizada por las variables generalizadas definiendo así la dirección de las fuerzas de reacción producidas por las velocidades generalizadas ($\dot{\xi}$) (González-Villela, 2006).

Para la construcción de la matriz A se obtiene al ir evaluando la matriz de velocidad V de forma que se tome en cuenta el efecto de cada una de las variables de interés en el sistema, siendo cada resultado una columna de la matriz que se está generando. Para el caso de la primera columna de A (a_1), se evalúa a V de forma que la primera variable de interés sea unitaria y las restantes sean nulas, como se puede observar a continuación:

$$a_1 = V_{\substack{v_{P_x}=1 & \theta'_1=0 & \theta'_4=0 & \phi_{21}=0 & \phi_{32}=0 \\ v_{P_y}=0 & \theta'_2=0 & \phi_{11}=0 & \phi_{22}=0 & \phi_{41}=0 \\ \theta'_P=0 & \theta'_3=0 & \phi_{12}=0 & \phi_{31}=0 & \phi_{42}=0}} \tag{2.38}$$

De esta manera, se sigue evaluando a V de forma que todas las variables de interés se relacionen con cada una de las columnas de la matriz A y así se construye una matriz de forma 16x15 donde con columnas linealmente independientes.

$$A_{16x15} = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8 \quad a_9 \quad a_{10} \quad a_{11} \quad a_{12} \quad a_{13} \quad a_{14} \quad a_{15}] \tag{2.39}$$

Una vez que se tiene a la matriz A se calcula:

$$s = A \cdot q_p \tag{2.40}$$

De esta manera se obtiene una matriz s de forma 16x1, la cual es un sistema de ecuaciones linealmente independientes con el que se pueden obtener cada una de las variables de interés. Se resuelve el sistema (Sol) y se construye una matriz de forma 15x3 donde se involucre solamente una de las velocidades de entrada ($v(t)$) en cada columna que las componga.

$$S_{q1} = \begin{bmatrix} Sol_{v_{P_x}=1} & Sol_{v_{P_x}=0} & Sol_{v_{P_x}=0} \\ v_{P_y}=0 & v_{P_y}=1 & v_{P_y}=0 \\ \theta'_P=0 & \theta'_P=0 & \theta'_P=1 \end{bmatrix} \tag{2.41}$$

Se define una matriz de rotación R_{0P} la cual referencie a S_{q1} con respecto al marco global de referencia con ayuda de una matriz identidad (I_{15x15}).

$$R_{0P} = \begin{bmatrix} \cos(\theta_P) & sin(\theta_P) & 0 \\ -sin(\theta_P) & cos(\theta_P) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.42}$$

$$S_q = R_{0P} * I_{15x15} * S_{q1} \tag{2.43}$$

De esta manera, obtenemos a la matriz S_q. De forma que retomando la ecuación (2.35) se tiene que el sistema de ecuaciones en el espacio de estados que representa al robot móvil omnidireccional redundante.

$$\begin{pmatrix} {}^{v}Px \\ {}^{v}Py \\ \theta'_{p} \\ \theta'_{1} \\ \theta'_{2} \\ \theta'_{3} \\ \theta'_{4} \\ \phi'_{11} \\ \phi'_{12} \\ \phi'_{21} \\ \phi'_{22} \\ \phi'_{31} \\ \phi'_{32} \\ \phi'_{41} \\ \phi'_{42} \end{pmatrix} = \begin{pmatrix} \cos(\theta_p) & \sin(\theta_p) & 0 \\ -\sin(\theta_p) & \cos(\theta_p) & 0 \\ 0 & 0 & 1 \\ \frac{\sin(\theta_1)}{c} & \frac{\cos(\theta_1)}{c} & -\frac{c + a\cdot\cos(\theta_1) + b\cdot\sin(\theta_1)}{c} \\ \frac{\sin(\theta_2)}{c} & \frac{\cos(\theta_2)}{c} & -\frac{c - a\cdot\cos(\theta_2) + b\cdot\sin(\theta_2)}{c} \\ \frac{\sin(\theta_3)}{c} & \frac{\cos(\theta_3)}{c} & -\frac{c + a\cdot\cos(\theta_3) - b\cdot\sin(\theta_3)}{c} \\ \frac{\sin(\theta_4)}{c} & \frac{\cos(\theta_4)}{c} & \frac{a\cdot\cos(\theta_4) - c + b\cdot\sin(\theta_4)}{c} \\ \frac{c\cdot\cos(\theta_1) - d\cdot\sin(\theta_1)}{c\cdot r} & \frac{c\cdot\sin(\theta_1) + d\cdot\cos(\theta_1)}{c\cdot r} & \frac{a\cdot c\cdot\sin(\theta_1) + a\cdot d\cdot\cos(\theta_1) - b\cdot c\cdot\cos(\theta_1) + b\cdot d\cdot\sin(\theta_1)}{c\cdot r} \\ \frac{c\cdot\cos(\theta_1) + d\cdot\sin(\theta_1)}{c\cdot r} & \frac{c\cdot\sin(\theta_1) - d\cdot\cos(\theta_1)}{c\cdot r} & -\frac{a\cdot d\cdot\cos(\theta_1) - a\cdot c\cdot\sin(\theta_1) + b\cdot c\cdot\cos(\theta_1) + b\cdot d\cdot\sin(\theta_1)}{c\cdot r} \\ \frac{c\cdot\cos(\theta_2) - d\cdot\sin(\theta_2)}{c\cdot r} & \frac{c\cdot\sin(\theta_2) + d\cdot\cos(\theta_2)}{c\cdot r} & \frac{a\cdot c\cdot\sin(\theta_2) + a\cdot d\cdot\cos(\theta_2) + b\cdot c\cdot\cos(\theta_2) - b\cdot d\cdot\sin(\theta_2)}{c\cdot r} \\ \frac{c\cdot\cos(\theta_2) + d\cdot\sin(\theta_2)}{c\cdot r} & \frac{c\cdot\sin(\theta_2) - d\cdot\cos(\theta_2)}{c\cdot r} & -\frac{a\cdot c\cdot\sin(\theta_2) - a\cdot d\cdot\cos(\theta_2) + b\cdot c\cdot\cos(\theta_2) + b\cdot d\cdot\sin(\theta_2)}{c\cdot r} \\ \frac{c\cdot\cos(\theta_3) - d\cdot\sin(\theta_3)}{c\cdot r} & \frac{c\cdot\sin(\theta_3) + d\cdot\cos(\theta_3)}{c\cdot r} & \frac{a\cdot c\cdot\sin(\theta_3) + a\cdot d\cdot\cos(\theta_3) + b\cdot c\cdot\cos(\theta_3) - b\cdot d\cdot\sin(\theta_3)}{c\cdot r} \\ \frac{c\cdot\cos(\theta_3) + d\cdot\sin(\theta_3)}{c\cdot r} & \frac{c\cdot\sin(\theta_3) - d\cdot\cos(\theta_3)}{c\cdot r} & -\frac{a\cdot c\cdot\sin(\theta_3) - a\cdot d\cdot\cos(\theta_3) + b\cdot c\cdot\cos(\theta_3) + b\cdot d\cdot\sin(\theta_3)}{c\cdot r} \\ \frac{c\cdot\cos(\theta_4) - d\cdot\sin(\theta_4)}{c\cdot r} & \frac{c\cdot\sin(\theta_4) + d\cdot\cos(\theta_4)}{c\cdot r} & -\frac{a\cdot c\cdot\sin(\theta_4) + a\cdot d\cdot\cos(\theta_4) - b\cdot c\cdot\cos(\theta_4) + b\cdot d\cdot\sin(\theta_4)}{c\cdot r} \\ \frac{c\cdot\cos(\theta_4) + d\cdot\sin(\theta_4)}{c\cdot r} & \frac{c\cdot\sin(\theta_4) - d\cdot\cos(\theta_4)}{c\cdot r} & \frac{a\cdot d\cdot\cos(\theta_4) - a\cdot c\cdot\sin(\theta_4) + b\cdot c\cdot\cos(\theta_4) + b\cdot d\cdot\sin(\theta_4)}{c\cdot r} \end{pmatrix} \begin{pmatrix} {}^{v}Px \\ {}^{v}Py \\ \theta'_{p} \end{pmatrix}$$

(2.44)

2.4 MODELO DEL MANIPULADOR SERIAL DE 5GDL

2.4.1 ARQUITECTURA DEL MANIPULADOR SERIAL DE 5GDL

El manipulador serial de 5GDL utilizado para el presente trabajo es un robot armado a partir de componentes comerciales distribuidos por la tienda "*SandoRobotics*". Al ser ensamblado, los servomotores utilizados tuvieron una posición particular, la cual se tiene que considerar ya que estos son únicamente de 180 grados de giro.

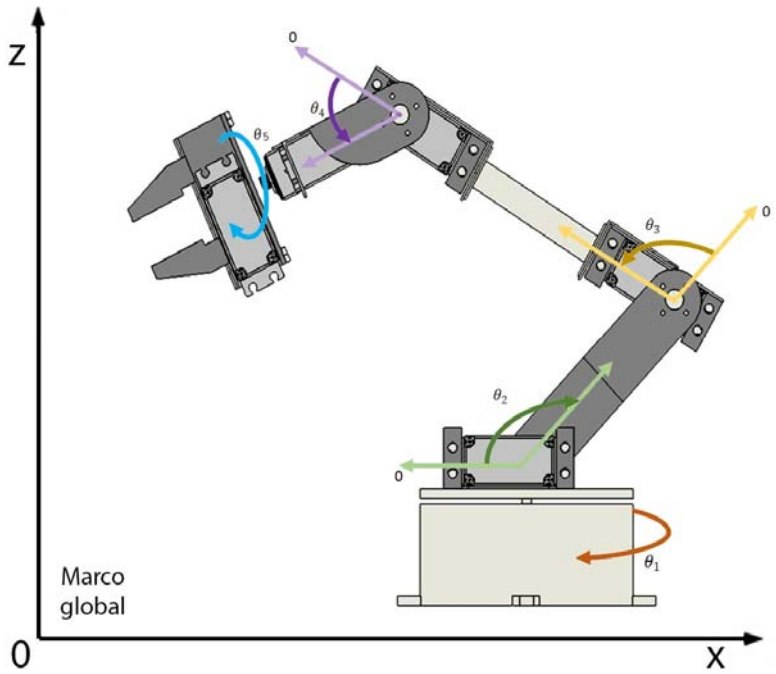


Figura 2.11 Relación de ángulos de servomotores.

El análisis cinemático que se llevará a cabo consistirá en usar vectores que definan la cadena cinemática del robot manipulador serial de 5GDL. La forma en la que se transformarán los vectores desde un sistema de coordenadas hacia otro es empleando matrices de transformación homogéneas de 4x4. Las componentes que conforman una matriz de transformación homogénea son las siguientes:

$$T\_H = \begin{bmatrix} R_{3x3} & P_{3x1} \\ f_{1x3} & E_{1x1} \end{bmatrix} \tag{2.45}$$

La matriz R_{3x3} representa a la matriz de rotación, la matriz P_{3x1} representa al vector de posición del origen del sistema rotado con respecto al sistema de referencia, la matriz f_{1x3} representa la transformación de la perspectiva y la matriz E_{1x1} representa el escalado.

Para la construcción de la matriz de transformación homogénea son necesarias las matrices de rotación básicas en los ejes coordenados "x", "y", "z" y las matrices de traslación básicas sobre los mismos ejes coordenados.

$$Q_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta_x & -S\theta_x & 0 \\ 0 & S\theta_x & C\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.46}$$

$$Q_y = \begin{bmatrix} C\theta_y & 0 & S\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta_y & 0 & C\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.47}$$

$$Q_z = \begin{bmatrix} C\theta_z & -S\theta_z & 0 & 0 \\ S\theta_z & C\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.48}$$

$$T_x = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.49}$$

$$T_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.50}$$

$$T_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.51}$$

2.4.2 ANÁLISIS CINEMÁTICO

El robot manipulador seria de 5GDL consta de una cadena cinemática con la que se puede llegar a las coordenadas del punto del efector final. La ecuación de lazo que utiliza para el análisis del manipulador serial de 5GDL se define de la siguiente manera:

$$R_{05} = R_{01} + R_{12} + R_{23} + R_{34} + R_{45} \tag{2.52}$$

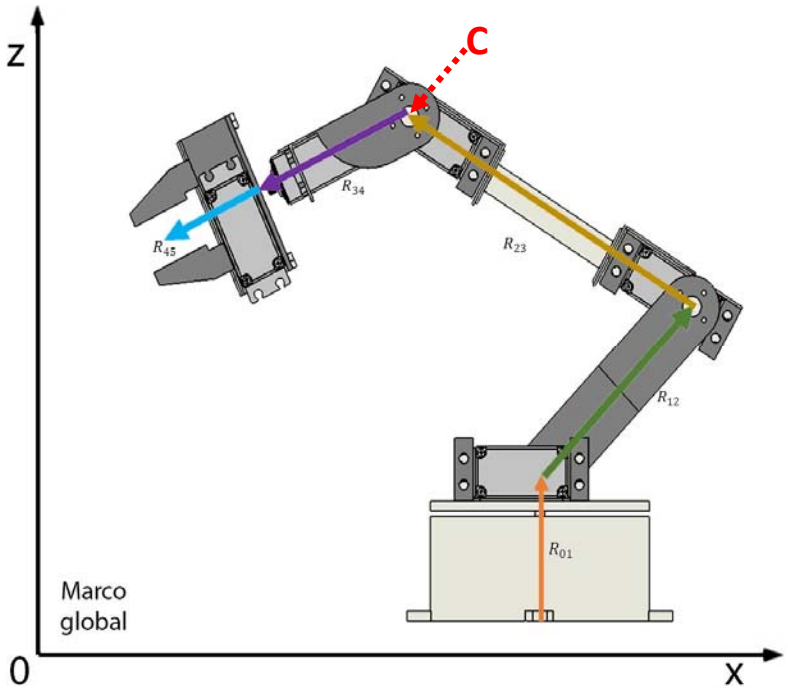


Figura 2.12 Vectores de la cadena cinemática de manipulador serial de 5GDL.

Se expresan los vectores de la ecuación de lazo con las matrices de transformación correspondientes para alcanzar el sistema de referencia local. Dadas las posiciones, referencias y sentidos de giro de los servomotores, a los vectores se les definen unas constantes tales que repliquen el movimiento de los actuadores.

$$R_{01} = Q_z(\pi)Q_z(-\theta_1)T_z(r_{01}) \tag{2.53}$$

$$R_{12} = Q_z(\pi)Q_z(-\theta_1)Q_y(-\theta_2)T_x(r_{12}) \tag{2.54}$$

$$R_{23} = Q_z(\pi)Q_z(-\theta_1)Q_y(-\theta_2)Q_y(\theta_3)T_x(r_{23}) \tag{2.55}$$

$$R_{34} = Q_z(\pi)Q_z(-\theta_1)Q_y(-\theta_2)Q_y(\theta_3)Q_y(\theta_4)T_x(r_{34}) \tag{2.56}$$

$$R_{45} = Q_z(\pi)Q_z(-\theta_1)Q_y(-\theta_2)Q_y(\theta_3)Q_y(\theta_4)Q_x(\theta_5)T_x(r_{45}) \tag{2.57}$$

Sustituyendo las ecuaciones (2.53), (2.54), (2.55), (2.56) y (2.57) en la ecuación vectorial (2.52), considerando únicamente al vector de posición (P_{3x1}):

$$X_p = -C(\theta_1)[r_{12}C(\theta_2) + r_{23}C(\theta_2 - \theta_3) + (r_{34} + r_{45})C(\theta_2 - \theta_3 - \theta_4)] \quad (2.58)$$

$$Y_p = S(\theta_1)[r_{12}C(\theta_2) + r_{23}C(\theta_2 - \theta_3) + (r_{34} + r_{45})C(\theta_2 - \theta_3 - \theta_4)] \quad (2.59)$$

$$Z_p = r_{01} + r_{12}S(\theta_2) + +r_{23}S(\theta_2 - \theta_3) + (r_{34} + r_{45})S(\theta_2 - \theta_3 - \theta_4) \quad (2.60)$$

Dado el sistema de ecuaciones anterior, se tiene que el sistema no tiene solución ya que se cuentan con tres ecuaciones y cuatro incógnitas $(\theta_1, \theta_2, \theta_3, \theta_4)$. A su vez, este sistema no contempla la orientación del ángulo θ_5 que actúa sobre el efector final (gripper) el cual es muy importante para objetos con formas y aristas bien definidas.

Para tener en cuenta la orientación del efector final, se sustituyen las ecuaciones (2.53), (2.54), (2.55), (2.56) y (2.57) en la ecuación vectorial (2.52), considerando ahora al vector de orientación (R_{3x3}):

$$R_{3x3} = \begin{bmatrix} RR_{11} & RR_{12} & RR_{13} \\ RR_{21} & RR_{22} & RR_{23} \\ RR_{31} & RR_{32} & RR_{33} \end{bmatrix} \quad (2.61)$$

$$donde$$

$$RR_{11} = -C(\theta_1)[1 + C(\theta_2) + C(\theta_2 - \theta_3) + 2C(\theta_2 - \theta_3 - \theta_4)] \quad (2.62)$$

$$RR_{21} = S(\theta_1)[1 + C(\theta_2) + C(\theta_2 - \theta_3) + 2C(\theta_2 - \theta_3 - \theta_4)] \quad (2.63)$$

$$RR_{31} = S(\theta_2) + S(\theta_2 - \theta_3) + 2S(\theta_2 - \theta_3 - \theta_4) \quad (2.64)$$

$$RR_{12} = S(\theta_5)C(\theta_1)S(\theta_2 - \theta_3 - \theta_4) - S(\theta_1)[4 + C(\theta_5)] \quad (2.65)$$

$$RR_{22} = -S(\theta_5)S(\theta_1)S(\theta_2 - \theta_3 - \theta_4) - C(\theta_1)[4 + C(\theta_5)] \quad (2.66)$$

$$RR_{32} = S(\theta_5)C(\theta_2 - \theta_3 - \theta_4) \quad (2.67)$$

$$RR_{13} = S(\theta_5)S(\theta_1) + C(\theta_1)[S(\theta_2) + S(\theta_2 - \theta_3) + [1 + C(\theta_5)]S(\theta_2 - \theta_3 - \theta_4)] \quad (2.68)$$

$$RR_{23} = S(\theta_5)C(\theta_1) - S(\theta_1)[S(\theta_2) + S(\theta_2 - \theta_3) + [1 + C(\theta_5)]S(\theta_2 - \theta_3 - \theta_4)] \quad (2.69)$$

$$RR_{33} = 1 + C(\theta_2) + C(\theta_2 - \theta_3) + [1 + C(\theta_5)]C(\theta_2 - \theta_3 - \theta_4) \quad (2.70)$$

En (Escobedo-Castillo, 2012), se utiliza un manipulador serial de 5GDL similar al que se utilizará en esta ocasión. La solución de la cinemática inversa, que se encuentra perfectamente explicada en la sección IV.1.2, muestra que con la posición y orientación del efector final del manipulador como variables a controlar se puede calcular el valor del ángulo de las juntas restantes.

Para la solución de la cinemática inversa se plantea realizar un análisis hasta el punto o junta a la que llega el vector R_{23} (*figura 2.12*), llamémoslo punto C cuyas coordenadas serán (x_c, y_c, z_c), resolviendo aquí el problema de posicionamiento del manipulador. Esta solución es correcta dado que dicha junta se ubica en un mismo plano que el efector final en todo momento, dado que el diseño del manipulador serial de 5GDL no cuenta con una tercera junta de orientación que cambie el plano del efector final con respecto a esta junta.

De esta forma, el problema se divide en dos partes, de las cuales la solución de la problemática de la posición para el manipulador serial de 5GDL, propuesta en (Escobedo-Castillo, 2012) es:

$$\theta_1 = tan^{-1}\left(\frac{x_c}{y_c}\right) \tag{2.71}$$

$$\theta_2 = tan2^{-1}\left(\frac{z_c - r_{01}}{R}C(\beta) + \frac{x_c}{R}\sqrt{1 - C^2(\beta)}\ , -\left(\frac{z_c}{R}C(\beta) - \frac{z_c - r_{01}}{R}\sqrt{1 - C^2(\beta)}\right)\right)$$

$$donde\ R = \sqrt{{x_c}^2 + (z_c - r_{01})^2}\ y\ C(\beta) = \frac{R^2 + {r_{12}}^2 - {r_{23}}^2}{2r_{12}R} \tag{2.72}$$

$$\theta_3 = tan2^{-1}\left(\sqrt{1 - C^2(\gamma)}\ , -\frac{{r_{12}}^2 + {r_{23}}^2 - R^2}{2r_{12}r_{23}}\right)$$

$$donde\ R^2 = {r_{12}}^2 + {r_{23}}^2 - 2r_{12}r_{23} \quad y \quad \theta_3 = \pi - \gamma \tag{2.73}$$

El problema de la orientación, que está dirigido a la resolución del ángulo de las últimas dos juntas, las cuales dependerán del objeto a manipular y las coordenadas donde se encuentre. La solución para el problema de orientación consta en relacionar las coordenadas del efector final con las del punto C.

$$(x_c, y_c, z_c) = \left(\sqrt{{x_f}^2 + {y_f}^2} - r_{34}C(\theta_4), y_f - r_{34}S(\theta_1), z_f - r_{34}S(\theta_4)\right) \tag{2.74}$$

Para la resolución de los ángulos involucrados para la orientación θ_4, θ_5, estos ángulos son parámetros necesarios correspondientes a la orientación del objeto a manipular. Para poder definirlos, es necesario aclarar algunas cuestiones sobre la ubicación del objeto a manipular, la ubicación del manipulador móvil omnidireccional redundante en el espacio y la forma en que se agarrará el objeto. Todo lo anterior se trata en capítulos posteriores.

CAPITULO 3 DISEÑO Y CONSTRUCCIÓN DEL PROTOTIPO

3.1 INTRODUCCION

Este capítulo abarca la fase de diseño y construcción del prototipo funcional para pruebas.

El diseño de Ingeniería puede describirse como el proceso de aplicar diversas técnicas y principios científicos, con el objeto de definir un dispositivo, un proceso o un sistema con suficiente detalle para permitir su realización. Es importante destacar que es un proceso que debe ir encaminado a cubrir cierta necesidad (Ertas, A. y Jones, J. ,1996).

En el proceso de diseño que se siguió se generaron conceptos a partir de los requerimientos del robot móvil omnidireccional redundante. Posteriormente los conceptos generados fueron evaluados y filtrados mediante el uso de tablas de selección de conceptos.

La matriz de selección de conceptos es usada en la etapa denominada *evaluación de conceptos.* La matriz de selección de conceptos es una herramienta de apoyo para seleccionar el concepto o los conceptos que tengan más probabilidad de llevar el producto al éxito (Ulrich y Eppinger, 1995).

Para construir la matriz de selección de concepto se escribe en la primera fila los diferentes conceptos a evaluar, en la primera columna los criterios de selección que serán calificados con el siguiente código:

- 1 si es “mejor que”,
- 0 si “igual a”
- -1 si es “peor que”

Posteriormente se suman todos los valores asignados para cada concepto. Se elige el concepto con la mejor evaluación neta.

3.2 REQUERIMIENTOS Y ESPECIFICACIONES

- ✓ Se requiere construir un prototipo piloto de un robot móvil omnidireccional redundante, tomando como base el prototipo ya existente de Arturo M.C. y Víctor J.G.V.
- ✓ Se necesita que el manipulador serial cuente con 5GDL, utilice servomotores para su generación de movimiento y sea a partir de componentes comerciales.
- ✓ Se utilizará una configuración similar, tendrá cuatro robots móviles diferenciales de tipo (2,0) en vez de dos, que se encontrarán unidos con una junta rotacional a una plataforma rectangular.

Ahora, para establecer las especificaciones se usa de apoyo la teoría de dominios, la cual afirma que la síntesis de máquinas consiste del establecimiento sucesivo de cuatro sistemas.

1. Sistema de Proceso
2. Sistema Funcional
3. Sistema de Órganos
4. Sistema de Partes

Entonces, se tienen que ir definiendo y especificando cada uno de los sistemas que componen a nuestra máquina. Pero antes de la definición del Sistema de Proceso se hace preciso definir la tarea para la cual se requiere diseñar el sistema.

Tarea: Consiste en desplazar a un objeto de un lugar a otro con movimientos coordinados con la capacidad de evadir obstáculos.

El sistema de proceso es una estructura de procesos, la cual transforma energía materia e información. Un proceso es una serie de actividades que se realizan bajo ciertas circunstancias con un fin determinado. En este paso se definirán las actividades, las transformaciones, que la tarea requiere.

Procesos:

1. Se le indicará al manipulador móvil la posición a la cual desea llegar.
2. Se determinará la presencia de obstáculos.
3. Se obtendrá la orientación actual de los robos diferenciales con respecto a la plataforma, y los errores de posición y velocidad.
4. Se obtendrán las distancias iniciales entre el manipulador móvil y los objetivos.
5. Se trazará la ruta óptima por la que se desplazará el manipulador móvil.
6. Se calculará la velocidad y orientación necesaria para llegar a la posición deseada de la base del manipulador móvil.
7. Se realizará el cálculo y envío de las velocidades y orientaciones individuales para cada robot diferencial, así como las velocidades de cada una de las ruedas del mismo.
8. Se definirá la ruta que siga el manipulador serial.
9. Se calculará y enviará las posiciones de los servomotores de cada una de las articulaciones del manipulador serial.
10. Habrá la recepción de velocidades y orientaciones por cada robot diferencial.

El sistema funcional es una estructura de funciones propósito o efectos necesarios en las máquina para crear las transformaciones deseadas. Esto es mejor entendido como la transformación de entradas a salidas. Dado que el proceso son las transformaciones deseadas, en este apartado se enlistaran los efectos necesarios para cumplir con dicha transformación. Solo se muestran las funciones esenciales para cumplir el proceso.

Funciones:

1. Se le indicará a la plataforma la posición a la cual desea llegar.
    a. Función de solicitar una posición meta.
    b. Función de obtención de la posición meta.
2. Se determinará la presencia de obstáculos.
    a. Función de sensado.
    b. Función de determinación de obstáculo.
3. Se obtendrá la orientación actual de los robots diferenciales con respecto a la plataforma, y los errores de posición y velocidad.
    a. Función de recepción de datos.
    b. Función de sensado de orientación.
    c. Función de cálculo de parámetros.
4. Se obtendrán las distancias iniciales entre el manipulador móvil y los objetivos.
    a. Función de sensado.
5. Se trazará la ruta óptima por la que se desplazará el manipulador móvil.
    a. Función de navegación y control.
6. Se calculará la velocidad y orientación necesaria para llegar a la posición deseada de la base del manipulador móvil.
    a. Función de navegación y control.
7. Se realizará el cálculo y envío de las velocidades y orientaciones individuales para cada robot diferencial, así como las velocidades de cada una de las ruedas del mismo.
    a. Función de cálculo de parámetros individuales.
    b. Función de cálculo de velocidad.
    c. Función de envío de datos.
8. Se definirá la ruta que siga el manipulador serial.
    a. Función de navegación y control.
9. Se calculará y enviará las posiciones de los servomotores de cada una de las articulaciones del manipulador serial.
    a. Función de cálculo de posición.

b. Función de envío de datos.

10. Habrá la recepción de velocidades y orientaciones por cada robot diferencial.
    a. Función de recepción de datos

El Sistema Orgánico es una estructura de órganos, cada uno de los cuales realizan una o más funciones a través de efectos físicos. A continuación se muestran los órganos que definen a nuestro sistema.

Órganos:

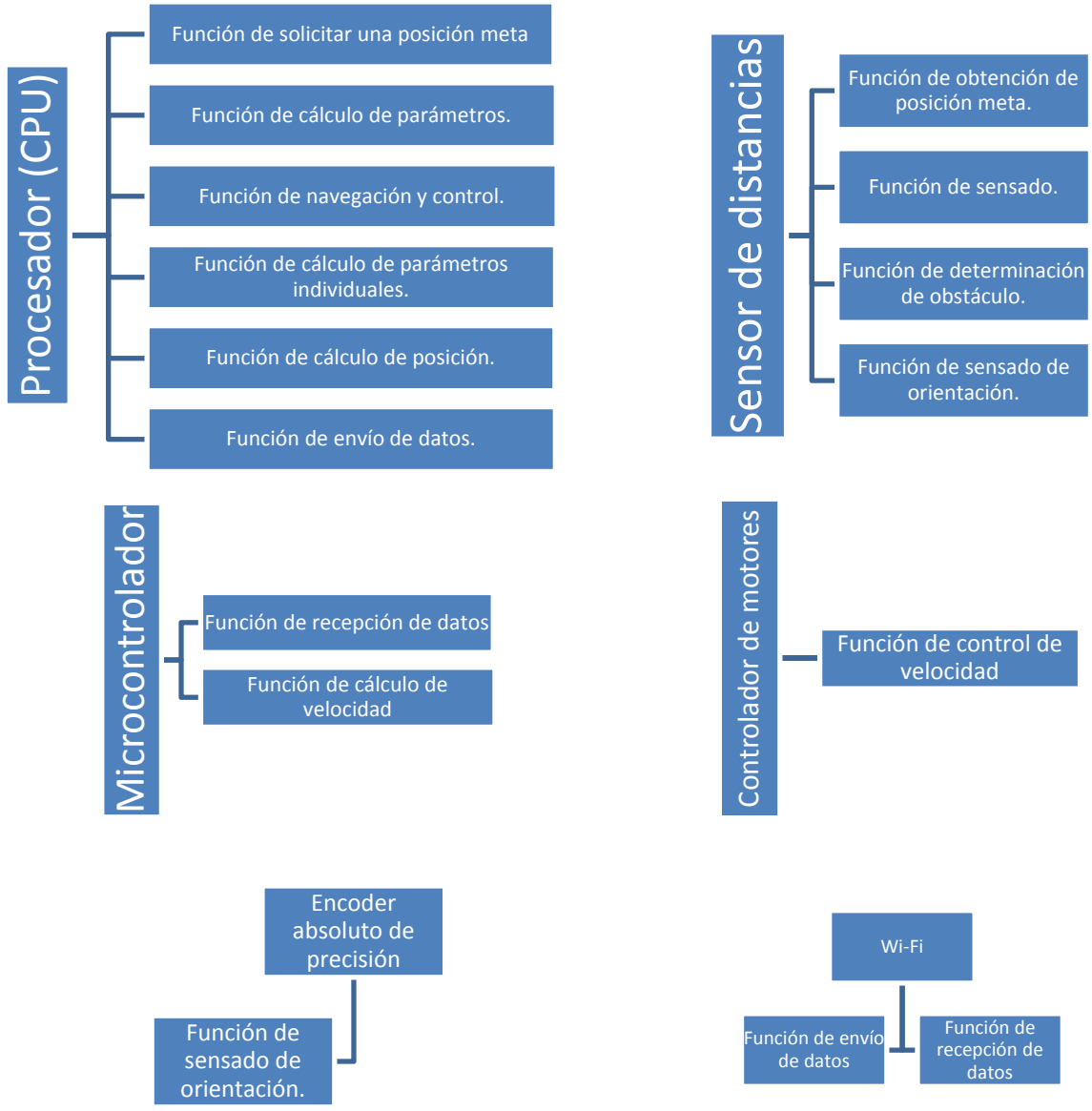


Figura 3.1 Órganos del Sistema

El Sistema de Partes es una estructura de partes mecánicas simples que forma el conjunto de la máquina.

Partes:

1. Llantas
2. Motores.
3. Servomotores.
4. Cables.
5. Conectores.
6. Baterías.
7. Bases para los robots diferenciales.
8. Elementos del brazo robótico.
9. Soportes.
10. Espaciadores.
11. Base principal del robot móvil omnidireccional.
12. Base principal del manipulador serial.
13. Acoplamientos.

Finalmente, tomando en cuenta todos los sistemas previamente definidos, se procede a realizarse una lista de especificaciones la cual se tiene que cumplir para el diseño y la construcción del modelo del robot móvil omnidireccional redundante.

Especificaciones:

1. Deberá ser capaz de desplazarse por superficies lisas.
2. La comunicación del manipulador móvil omnidireccional redundante con computadoras externas deberá de ser de forma inalámbrica.
3. La alimentación del robot deberá estar integrada en él.
4. Debe haber libre acceso a las baterías.
5. Al menos el 80% de los componentes deben venderse o manufacturarse en México.
6. Cada robot móvil diferencial deberá de tener su propio microcontrolador y controlador de motores.
7. El sistema deberá contar con un sensor de distancia que cubra todo el sistema de pruebas.
8. La obtención de la posición deseada será recibida por una computadora.

3.3 DISEÑO CONCEPTUAL

Como ya se comentó en capítulos anteriores, la plataforma base del manipulador móvil omnidireccional redundante está constituida por cuatro robots diferenciales tipo (2,0) unidos por una junta rotacional a una plataforma rectangular rígida.

La plataforma base del manipulador móvil omnidireccional redundante debe cumplir con la función de desplazamiento, debe llevar al manipulador serial, el cual tendrá un objeto a la posición que se asigna. Por lo tanto, tiene que trazar el camino, que lleve a la plataforma móvil a la postura que permita cumplir con la tarea.

Además de proporcionar movimiento, dada la configuración de la base del manipulador móvil omnidireccional redundante, cada uno de los robots móviles diferenciales de la plataforma debe de contar con espacio para la batería que le proporcionará el voltaje de alimentación, también necesita el espacio para colocar todos los componentes que lo controlan. El manipulador serial a su vez, debe contar con su propio voltaje de alimentación y espacio para los componentes que lo controlan.

Por las razones anteriores, se tiene que diseñar un manipulador móvil omnidireccional redundante de arquitectura modular. Esto, con la finalidad de que el sistema pueda ser ensamblado con facilidad y la sustitución de componentes ante fallo sea rápida.

Tomando en cuenta el trabajo (Martínez-Carrillo, 2011), donde se presentan robots móviles con estas características, se decide tomar como base el diseño planteado realizando algunas modificaciones en los robots móviles diferenciales para poder adquirir una arquitectura más rígida.

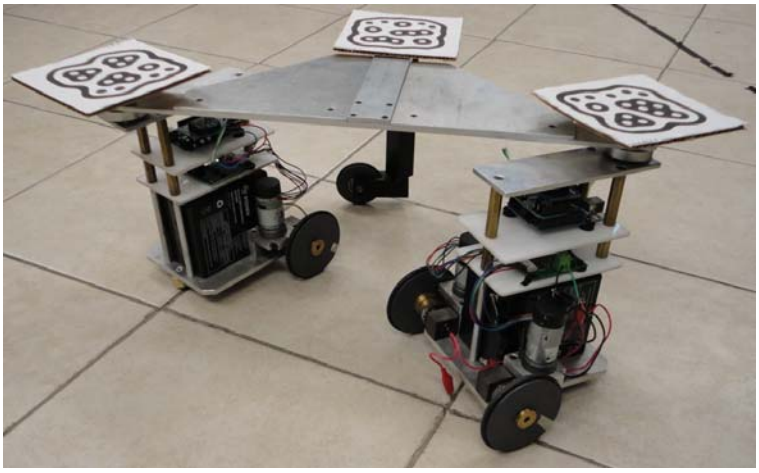


Figura 3.2 Plataforma Omnidireccional de (Martínez-Carrillo, 2011)

3.4 DISEÑO MECÁNICO DEL ROBOT MÓVIL OMNIDIRECCIONAL REDUNDANTE

3.4.1 SELECCIÓN DE MATERIALES

Para mejorar la rigidez de la plataforma móvil omnidireccional previa, se han propuesto diferentes acciones que pueden ser tomadas. En primera instancia, se busca que los diferentes niveles donde se colocan los dispositivos electrónicos sean lo más rígido posibles para evitar que aparezcan momentos cortantes, a su vez se tiene que considerar que el peso del robot puede verse afectado. Se propusieron dos posibles soluciones para resolver este problema y se realizó una matriz de selección de concepto para facilitar la elección a tomar.

Criterio / Acción	Material rígido en toda la estructura	Mayor cantidad de puntos de sujeción
Manufactura	1	0
Materiales	0	1
Rigidez	1	1
Peso	0	1
Evaluación neta	2	3

Tabla 3.1 Matriz de selección de concepto de modificación del robot diferencial

La mejor opción es la de modificar la cantidad de los puntos de sujeción del robot móvil diferencial en cada uno de los niveles que presenta, ya que causa un movimiento no deseado que se presenta cuando el robot gira, esto es debido a que hay una fuerza normal por parte de la plataforma que se aplica sobre el móvil exigiéndole mayor rigidez interna al querer desplazarse.

Para la plataforma se decide tomar el mismo material empleado en la plataforma anterior ya que el Aluminio presenta una gran rigidez al ser un metal sólido y al mismo tiempo es uno de los más ligeros entre ellos.

Las dimensiones del robot móvil diferencial tipo (2,0) son las mismas que el prototipo previo hecho por Manuel M.C. y Víctor J.G.V., mientras que la plataforma está diseñada a escala dado que tiene una forma geométrica diferente.

3.4.2 SELECCIÓN DE ACTUADORES

Los actuadores se encargarán de dar movimiento al robot móvil omnidireccional redundante. Existen tres grandes grupos en los que son clasificados los actuadores, según el tipo de energía que lleguen a utilizar:

- Neumáticos
- Hidráulicos
- Eléctricos

Utilizando matrices de selección de concepto se tiene lo siguiente.

Criterio / Actuador	Neumáticos	Hidráulicos	Eléctricos
Tamaño	-1	-1	1
Costo	-1	-1	1
Precisión	0	0	1
Control	0	0	1
Evaluación neta	-2	-2	4

Tabla 3.2 Matriz de selección de actuadores

Entre éstos, se elige utilizar actuadores eléctricos ya que para el caso de un robot móvil, en comparación con los actuadores neumáticos e hidráulicos, ya que son de menor tamaño, de menor costo, tienen una mayor precisión, su control es más sencillo no requieren de tanto mantenimiento y a su vez la facilidad de implementación que nos brindan.

Dentro de los actuadores eléctricos, los más viables que pueden ser utilizados para esta aplicación son los motores de corriente continua, los motores a pasos y los servomotores.

Criterio / Actuador	Servomotores	Motor CC	Motor a pasos
Control	1	0	-1
Alimentación	1	1	-1
Torque	1	1	0
Velocidad	0	1	0
Evaluación neta	3	3	-2

Tabla 3.3 Matriz de selección de actuadores eléctricos

La velocidad de un motor de corriente continua es mucho mayor, en el caso de un robot móvil la velocidad de sus motores junto con el par diferencial que estos tengan es lo que más se tiene que considerar. A su vez, con la ayuda de un sensor externo que retroalimente al sistema se puede verificar la velocidad angular con la que se mueve el actuador.

En la tabla 3.2, se observa el mismo valor de la "Evaluación neta", pero al tener claros cuales son los criterios más necesarios para nuestro robot móvil, podemos definir que un motor de corriente continua es más apto para el sistema.

El modelo anterior estaba constituido con motores de corriente continua, dado el análisis mostrado se puede rectificar que los motores seleccionados nos proporcionan una gran ventaja, por ello se elige de la misma manera el motorreductor MG30.

Figura 3.3 Motorreductor EMG30

Modelo	**EMG30**
Voltaje	12 [V]
Velocidad	170 [rpm]
Par diferencial	1.5 [kg/cm]
Peso	43 [g]
Corriente	530 [mA]

Tabla 3.4 Detalles del servomotor de la figura 3.1

3.4.3 SELECCIÓN DE SENSORES

Se quiere que el sistema tenga una retroalimentación para saber la posición real de cada uno de los robots móviles diferenciales tipo (2,0) en todo momento, para ello es necesaria la implementación de sensores que nos lo indiquen.

Criterio / Sensor	Encoder	Potenciómetro	Cámara
Precisión	1	1	0
Acoplamiento	0	-1	1
Lectura	1	1	0
Calibración	1	0	-1
Evaluación neta	3	1	0

Tabla 3.5 Matriz de selección de sensores

Los encoders son los sensores que nos brindan una gran precisión y facilidad en la calibración y en la lectura de la posición actual sin la necesidad de realizar una programación extensa como sería el caso para la cámara.

Criterio / Encoder	Absolutos	Incrementales	Magnéticos Absolutos
Precisión	1	1	1
Acoplamiento	0	0	1
Lectura	0	-1	0
Calibración	1	0	1
Evaluación neta	2	0	3

Tabla 3.6 Matriz de selección de encoders

La utilización de encoders magnéticos absolutos resulta ser la más atractiva, debido a que su acoplamiento es mucho más sencillo al eje de unión que se presenta entre el robot móvil diferencial y la plataforma, además lo que estará en contacto directo con el eje solamente será un imán que rotará a la par del eje del robot móvil diferencial. El sensor magnético no entrará en contacto con el eje, evitando problemas mecánicos en el movimiento de los robots móviles por generación de alguna fricción.

Se realizó una búsqueda en la web de los diferentes encoders que existen en el mercado, para saber los tipos de sensores y sus características. Entre los existentes, los que tenían una buena accesibilidad de adquisición fueron los siguientes:

Característica / Encoder	AEAT-6010/6012	AEAT – 6600 – T16	MA3	MAE3
Compañía	Avago Technologies	Avago Technologies	US Digital	US Digital
Resolución	10 or 12 bits	10 to 16 bits	10 or 12 bits	10 or 12 bits
Salida de datos	Synchronous Serial Interface (SSI)	SSI, ABI, UVW, PWM	Analog or PWM	Analog or PWM
Voltaje de Alimentación	5 Volts	3.3 or 5 Volts	5 Volts	5 Volts
Sampling Rate	10.42 kHz	12 kHz	2.61 kHz	2.61 kHz
Precio (cantidad mínima)	27 (1), 17.4 (280)	4.58 (970)	77.11 (1), 52.13 (5+)	84.14 (1), 58.98 (5+)

Tabla 3.7 Tabla de algunas características de encoders magnéticos

De la tabla (3.7) se seleccionó al encoder magnético AEAT-6010/6012 como el sensor que retroalimentará la posición angular de los robots móviles diferenciales tipo (2,0) debido a que cumple satisfactoriamente con todos los criterios de selección previamente establecidos y, además, es el encoder más accesible en cuanto a cantidad y precio.

3.5 CONSTRUCCIÓN DEL MANIPULADOR SERIAL

3.5.1 SELECCIÓN DE MATERIALES

Se busca obtener un manipulador serial de fácil ensamble, cuyos componentes sean comerciales y de fácil obtención. Para ello se realizó una búsqueda en el mercado local; que se encuentren dentro del Distrito Federal, para poder obtener rápidamente una cotización de un manipulador serial de 5GDL y a su vez, se puedan adquirir componentes individualmente si llega a fallar alguno de ellos.

El resultado de la búsqueda nos dio como proveedor para los componentes del manipulador serial de 5GDL a la compañía SANDOROBOTICS, la cual se pidieron los siguientes componentes para la construcción del brazo robótico que se ubicará sobre el robot móvil omnidireccional redundante.

Cantidad	Código	Descripción
1	LG-NS	lynxmotion little grip kit (no servos)
1	ASB-26	aluminum mini-servo rotate bracket single pack
1	ASB-04	aluminum multi-purpose servo bracket two pack
1	ASB-06	aluminum "I" connector bracket two pack
2	HUB-08	aluminum tubing connector hub (pair)
1	AT-05	aluminum tubing - 2.250"
1	AT-02	aluminum tubing - 3.0"
1	ASB-10	aluminum long "c" servo bracket with ball bearings two pack
1	ASB-13	aluminum dual servo bracket (single)
1	BR-NS	Base Rotate Kit (no servos)
1	SDR-BARM	Base Brazo Robótico
1	SDR-CARM	Juego de extensiones para servo brazo robótico (3Pzas)
3	2143	Power HD High-Speed Mini Servo HD-1705MG
4	HS-422	hitec hs-422 servo motor

Tabla 3.8 Lista de componentes del manipulador serial de 5GDL

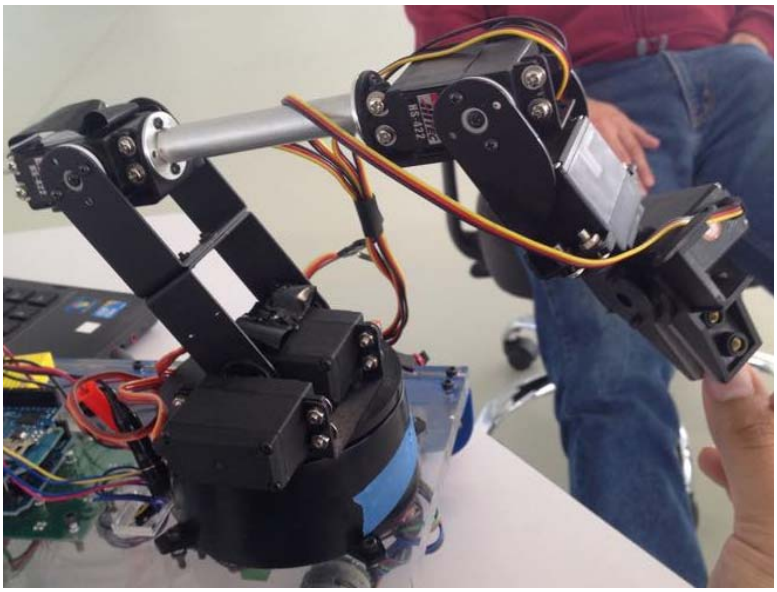


Figura 3.4 Manipulador Serial de 5GDL, SANDOROBOTICS

3.6 DISEÑO ELECTRÓNICO Y PROGRAMACIÓN

3.6.1 SELECCIÓN DE UN MICRO-CONTROLADOR

Un micro-controlador se define como un dispositivo programable capaz de realizar diferentes actividades que requieran del procesamiento de datos digitales, del control y comunicación digital de diferentes dispositivos. Los micro-controladores poseen memoria interna donde guardan dos tipos de datos; la instrucciones, que corresponden al programa que se va a ejecutar, y los registros, que son los datos que el usuario maneja así como los registros especiales para el control de las diferentes funciones del micro-controlador (I. C. Cruz-López, 2013).

Para este trabajo es necesario es uso de un micro-controlador, debido a la gran cantidad y dificultad que presentan las instrucciones que necesitan ser ejecutadas para un correcto desempeño del manipulador móvil omnidireccional redundante. Es utilizado el micro-controlador denominado ATMEGA328. Dicho micro-controlador es fabricado por la familia Atmel.

Este modelo de micro-controlador lo podemos encontrar en la plataforma electrónica de ARDUINO, el cual se basa en hardware y software fácil de usar (Arduino, 2015). Las características más importantes presentes en esta plataforma con dicho micro-controlador que hacen versátil, eficiente y práctico para ser empleado en esta aplicación son las siguientes:

- Es un producto de carácter modular.
- Cuenta con la viabilidad de implementar comunicación inalámbrica.
- Memoria interna de gran capacidad, además de la capacidad de ampliarla.
- Tiene memoria tipo FLASH, es decir, que puede reprogramarse electrónicamente.
- Posee su propio ambiente de desarrollo y lenguaje de programación.
- Cuenta con librerías de diferentes protocolos de comunicación.

El micro-controlador seleccionado se encargará de establecer una comunicación entre una PC y la manipulación del manipulador móvil omnidireccional redundante. Es decir, los datos serán enviados desde una PC al micro-controlador.

El micro-controlador que se encuentre ubicado en cada uno de los robots móviles tendrá conectado a sus pines entradas de control de la tarjeta MD25; que se encargarán del movimiento de los motores con la velocidad deseada.

El micro-controlador que este sobre la plataforma, se encargará de conectar a los encoders magnéticos absolutos, quienes son los encargados de leer la posición angular real de todos los robots móviles diferenciales, mediante el uso de un SSI.

El manipulador serial de 5GDL, será controlado por uno de estos micro-controladores ya que cuentan con salidas digitales que se encargarán de enviar cadenas de pulsos a los servomotores del manipulador serial de 5GDL y así poder posicionarlo como es debido.

Para hacer óptimo al sistema, es necesario el empleo de varios micro-controladores, lo cual da paso a que éstos se encuentren comunicados entre sí, para ello es utilizado el protocolo de comunicación Wifi.

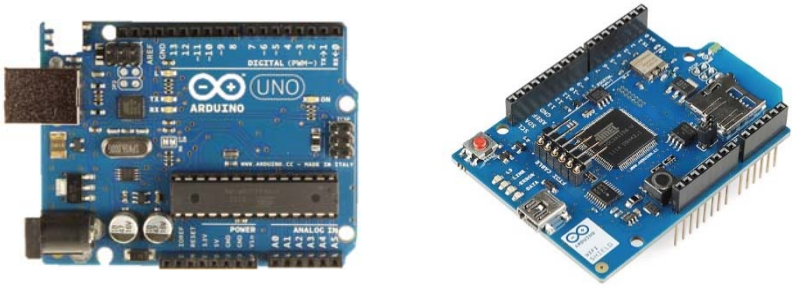


Figura 3.5 Micro-controlador ARDUINO y módulo de comunicación Wifi

3.6.2 ACTUADORES

El motor utilizado, el EMG30 (encoder, motor, gearbox 30:1), es un motor que se encuentra equipado con encoders incrementales y un “gearbox” o caja de engranes de reducción de 30:1. Es ideal para pequeñas o medianas aplicaciones robóticas, proporcionando costo efectivo de accionamiento y retroalimentación para el usuario. El motor cuenta con condensadores internos de filtro que ayudan a minimizar el ruido y los parásitos generados por el motor al girar.

EMG30 es un motor de corriente continua de 12V y 170 revoluciones que se caracteriza por incluir un encoder o codificador de cuadrante que manda un tren de impulsos cuando gira el eje del motor, permitiendo así que un circuito externo pueda saber la velocidad real a la que está girando el eje y cuantas vueltas da. El encoder está formado por dos sensores de efecto hall que proporcionan un total de 360 pulsos por cada vuelta completa del rotor.

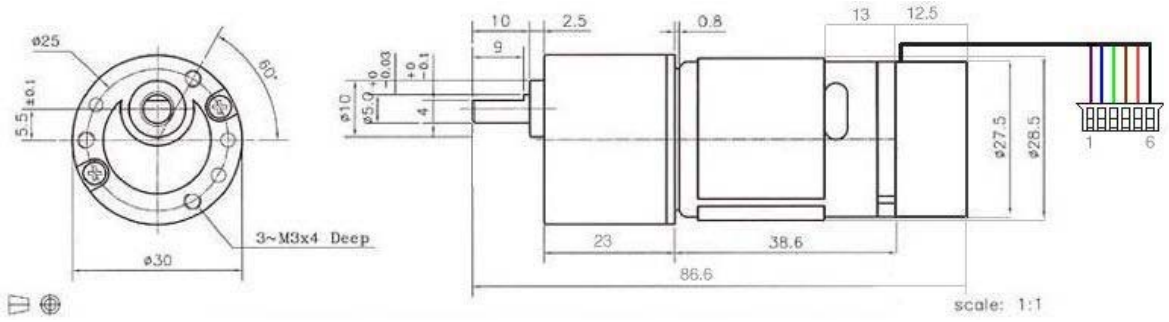


Figura 3.6 Diagrama de conexión del EMG30

El EMG30 cuenta con un conector de 6 vías JST (PHR-6) al final de un cable estándar de aproximadamente 90 [mm] de longitud. Las conexiones que presenta son:

COLOR	Conexión
Morado (1)	Vout del Sensor Hall B
Azul (2)	Vout del Sensor Hall A
Verde (3)	Tierra del Sensor Hall
Café (4)	Vcc del Sensor Hall
Rojo (5)	(+) Motor
Negro (6)	(-) Motor

Tabla 3.9 Código de colores del cable de EMG30

Los sensores de efector Hall acepta voltajes entre 3.5 – 10 volts. Las salidas son un colector abierto que requiere resistencias en "pull-up" para cualquier nivel de señal que sea requerido. En la tarjeta MD25 están alimentadas de 12 volts y "pull-up" a 5 volts para las señales.

La tarjeta Devantech MD25 (MD25, 2015) es un controlador de doble puente H desarrollado para el control de dos motores de corriente continua, en particular para el modelo EMG30. Sus características principales son:

- Lectura de encoders proveyendo el desplazamiento de los motores y sentido de giro.
- Maneja dos motores con control independiente o combinado.
- Lectura de la corriente que consume el motor.
- Se alimenta con un voltaje único de 12 volts de corriente directa.

- Regulador de +5Vcc a 300mA para alimentar la circuitería externa.
- Suministra hasta 2.8A de corriente para cada motor.
- Interfaz serial o I^2C con posibilidad de conectar hasta 8 controladores MD25 en el mismo bus.
- Control de potencia y aceleración.
- La resolución del encoder muestra 360 pulsos por vuelta

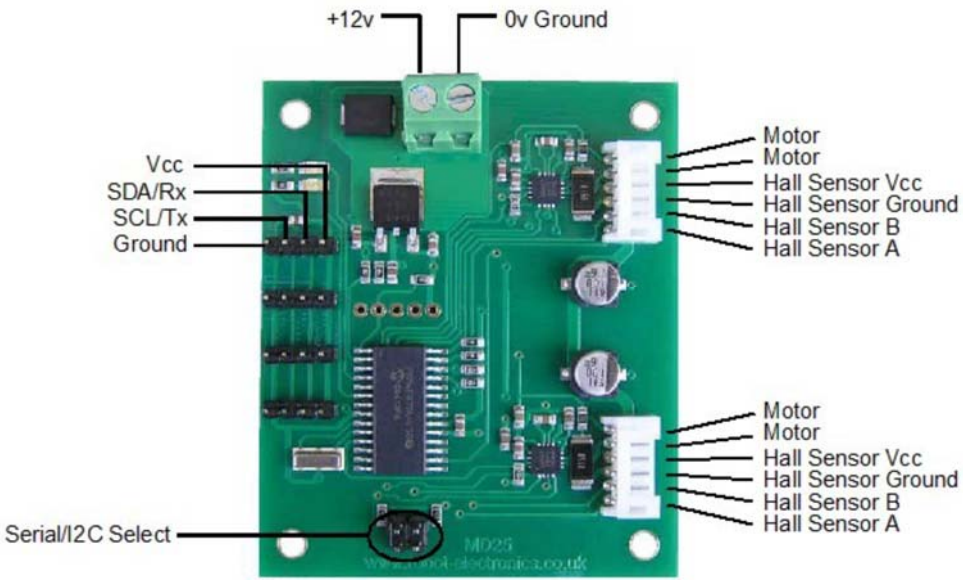


Figura 3.7 Tarjeta MD25

Esta tarjeta facilita el control de los motores, gracias a los comandos preestablecidos que sólo requieren la activación de distintos registros. Se decide utilizar la interfaz I^2C para el control de la tarjeta, debido a que el microcontrolador a utilizar sólo cuenta con una terminal para comunicación serial, misma que se utiliza para la comunicación inalámbrica.

Para el control de los motores, existen distintas opciones configurables en la tarjeta, dependiendo de la aplicación y de la practicidad para el manejo de datos. Para este proyecto, los motores se manejan en modo independiente con datos de entrada de velocidad que van de 0 a 255, siendo 0 la velocidad máxima en un sentido y 255 la velocidad máxima en sentido contrario.

3.6.3 LECTURA DE SENSORES

El sensor que ha sido seleccionado AEAT-6010/6012 de la familia Avago Technologies es un encoder magnético absoluto que nos provee una solución integral para leer la posición angular real de los eslabones del robot manipulador paralelo híbrido. El sensor está basado en tecnología magnética, por lo que no se encuentra en contacto con el sistema mecánico y asegura operación confiable.

El sensor que se adquirió para este proyecto es el AEA-6012, el cual tiene la característica especial de que tiene una resolución de 0.0879 grados (12 bits), lo equivalente a 4096 posiciones que puede tener en 360 grados. Los datos son enviados al micro-controlador de manera serial mediante una cadena de bits.

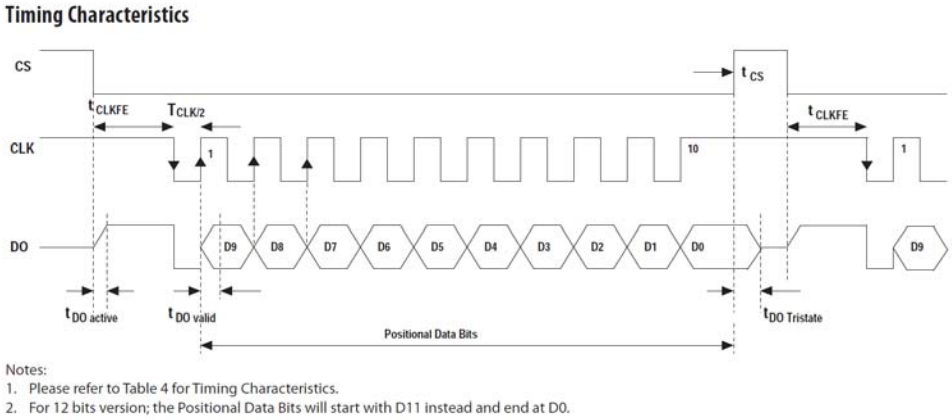


Figura 3.8 Diagrama de Tiempos para el Encoder Magnético de 10 bits

Con ayuda de la hoja de especificaciones (datasheet) con la que cuenta el encoder magnético, es posible programar al micro-controlador para el recibimiento de datos. Para el robot manipulador paralelo plano se utilizan tres entradas digitales por las que se recibirá la cadena de bits con la posición angular actual de cada uno de los actuadores (DO), una salida digital para enviar el tren de pulsos necesario para establecer la correcta transferencia de datos (CLK), y una salida digital con la que enviaremos la señal de inicio y final de la transferencia de datos (CS). Con estas conexiones, es posible efectuar la lectura de los tres sensores de manera simultánea.

Mientras que para el robot manipulador paralelo plano se utilizan cuatro entradas digitales para recibir la cadena de bits con la posición angular actual de los actuadores (DO), una salida digital para enviar el tren de pulsos (CLK), y una salida digital con la que enviaremos la señal de inicio y final de la transferencia de datos (CS).

3.6.4 INTERFAZ GRÁFICA

Matlab es un lenguaje de alto nivel con un ambiente interactivo para computación numérica, visualización y programación. Usando MATLAB, es posible analizar, desarrollar algoritmos y crear modelos y aplicaciones. El lenguaje, herramientas y funciones matemáticas permiten explorar diferentes aproximaciones y llegar a la solución más rápidamente que con hojas de cálculo o lenguajes de programación tradicionales como C/C++ o Java.

Matlab cuenta con un módulo llamado Real Time Windows Target que provee de un motor de tiempo real para ejecutar modelos de Simulink en una computadora corriendo Microsoft Windows y bloques que conectan a una gran variedad de tarjetas I/O. permite crear y controlar un Sistema en tiempo real para prototipos rápidos o simulación de hardware en ciclo. Real Time Windows Target soporta dos modos de simulación: modo normal, para una ejecución simple de tiempo real con acceso a dispositivos I/O, y modo externo, para tiempo real de alto rendimiento.

La programación de la cinemática del manipulador omnidireccional redundante se llevó a cabo en un archivo con extensión ".c" pues de esta manera se ejecuta el código de manera más rápida y puede ser implementado en un solo bloque de Simulink. De la misma manera, se programó el código necesario para hacer los cambios necesarios en la información de la velocidad de los motores antes de ser enviada a los robots móviles diferenciales.

En Simulink, también se puede simplificar el código de bloques anidando varios bloques en uno más simple para una mejor apreciación del código. Tal es el caso del bloque que modifica la información que recibe el ordenador de la posición de los encoders.

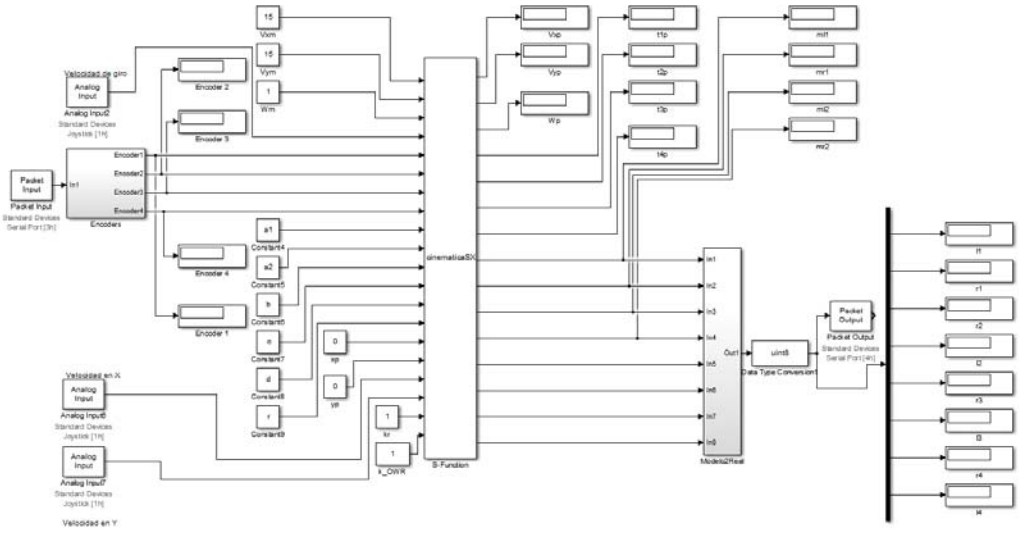


Figura 3.9 Interfaz de Matlab

3.6.5 PROTOCOLO DE COMUNICACIÓN WIFI

El Arduino Wifi Shield es la placa de adaptación del módulo Wifi WIZ610wi de WIZnet. Ésta placa (shield) da conectividad inalámbrica a internet siendo compatible con las plataformas Duemilanove, Mega y Uno. Con éste shield se hace sencillo conectarse a internet utilizando la infraestructura de comunicaciones ampliamente usada como Wifi. El módulo Wifi WIZ610wi posee el stack TCP/IP por hardware lo que lo hace ser una de las plataformas más estables del mercado, sin necesidad de ocupar recursos del procesador o microcontrolador en tareas de comunicación.

El Arduino Wifi Shield viene con una completa biblioteca de funciones para transmisión de datos y configuración del Shield desarrollada por MCI electronics. Arduino Wifi Shield es ideal para monitoreo y control de equipos usando Arduino y comunicándose vía internet.

La configuración del módulo se realiza vía web, aunque pueden configurarse algunos parámetros usando el modo de configuración vía serial.

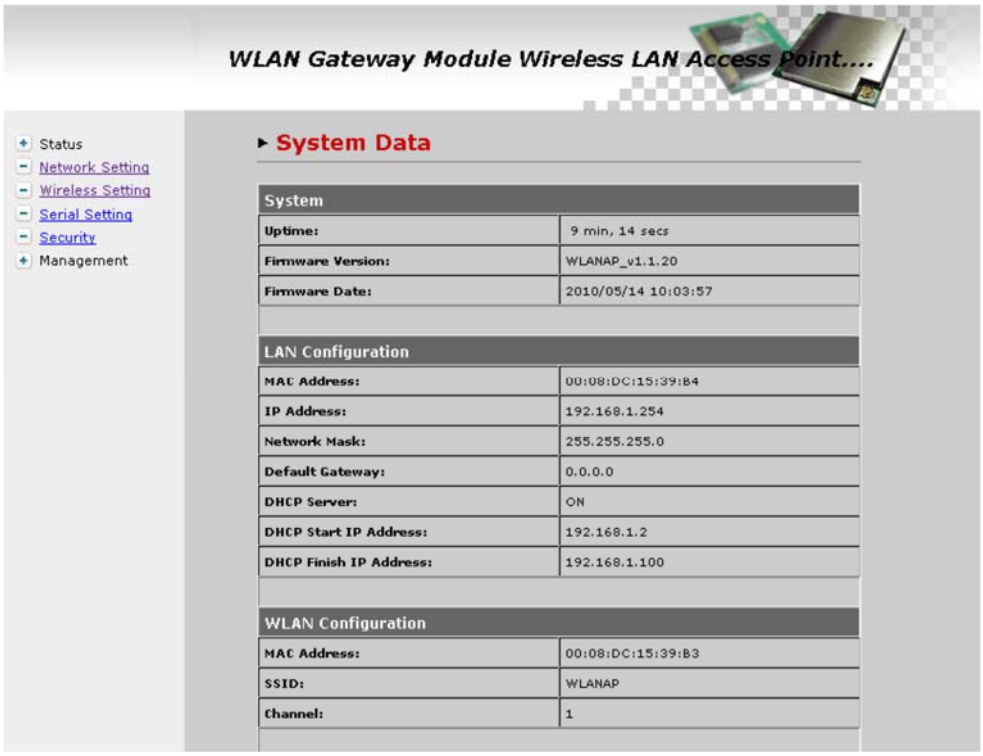


Figura 3.10 Página de configuración del módulo WIZ610

3.6.6 PROTOCOLO DE COMUNICACIÓN I^2C

Sus siglas vienen de *Inter-Integrated Circuit* y es un bus de comunicaciones en serie. Utilizado principalmente para la comunicación de circuitos integrados entre sí que normalmente se encuentran en un mismo circuito impreso. Utiliza tres líneas para la transmisión de la información; Datos (SDA), Reloj (SCL) y Tierra (GND).

Los dispositivos conectados al bus de I^2C tienen una dirección única para cada uno. Estos dispositivos pueden ejercer el rol de *maestros* o *esclavos*. El dispositivo maestro es el encargado de iniciar la transferencia de datos así como de generar la señal de reloj. Para realizar la comunicación inter-circuito solo requiere de tres cables: SDA, SCL y GND; uno donde se intercambiaran datos, el otro una señal de reloj y la tierra, respectivamente.

Este protocolo de comunicación es utilizado por la tarjeta MD25 para que éste envíe y reciba los comandos de movimiento por parte del micro-controlador ARDUINO. En la figura 3.8 se muestra el diagrama de conexión que tomará lugar para cada robot móvil diferencial.

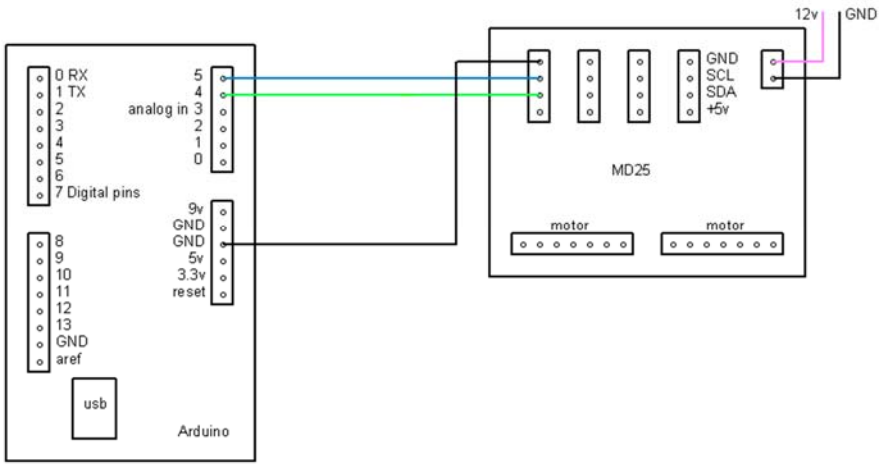


Figura 3.11 Conexión entre ARDUINO y la tarjeta MD25

3.7 IMPLEMENTACIÓN ELECTRÓNICA

Cada robot móvil diferencial tiene que ser independiente de los demás, para ello es necesario que cuente con su propio sistema de alimentación, procesamiento, transmisión de información y locomoción. La selección de los componentes que integran los robots diferenciales se llevó a cabo principalmente por su capacidad y facilidad de implementación. Los componentes seleccionados para cada robot diferencial son los siguientes:

Componente	# de Componentes
Módulo Wifi para ARDUINO	1
ARDUINO UNO Rev3	1
Motorreductor con Encoder 30:1. (EMG30)	2
cables M-F jumper (PRT-09140)	8
Controlador dual 12V 2.8A (MD25)	1
Batería 12V 4A	1

Tabla 3.10 Componentes de robot diferencial

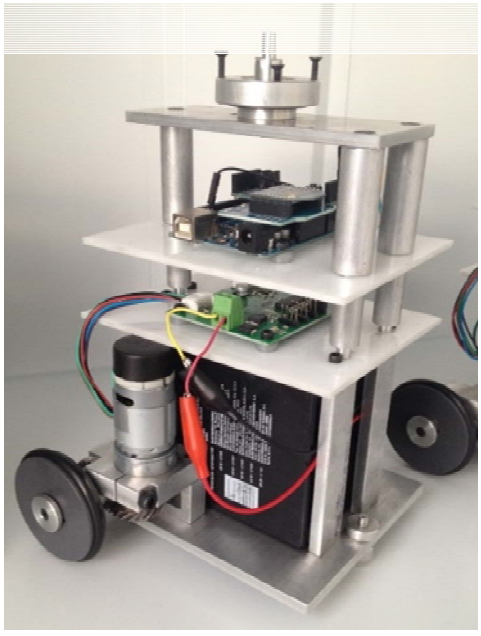


Figura 3.12 Robot Móvil Diferencial

Finalmente, para la sección superior de nuestro manipulador móvil omnidireccional redundante se encuentran situados los siguientes componentes:

Componente	# de Componentes
Módulo Wifi para ARDUINO	2
ARDUINO UNO Rev3	1
ARDUINO MEGA	1
cables M-F jumper (PRT-09140)	11
Reductor de voltaje 12V – 5V	1
Batería 12V 4ª	1
Encoder Magnético 10bits	4
Cable conexión de encoder	4

Tabla 3.11 Componentes de la parte superior del sistema

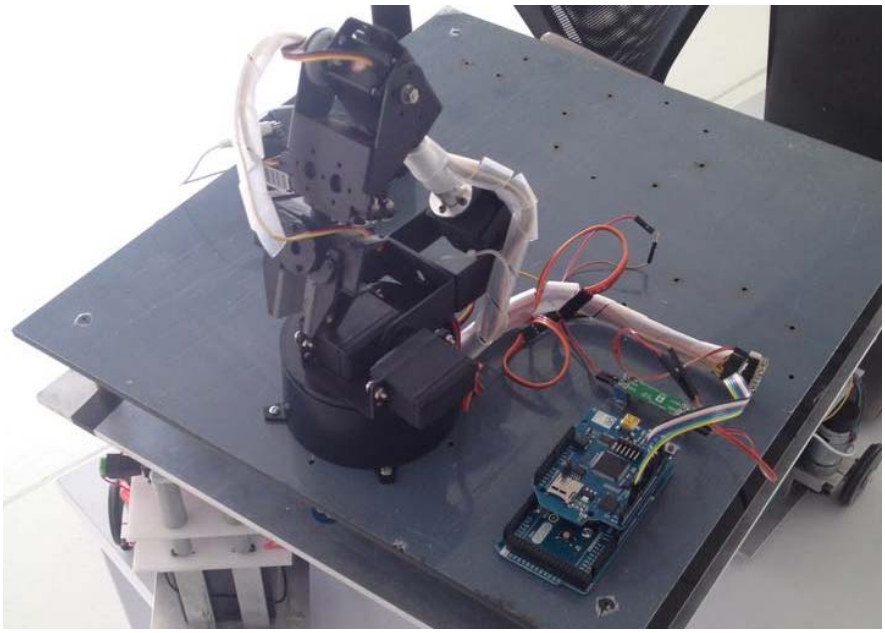


Figura 3.13 Manipulador Serial Instrumentado

CAPITULO 4 GENERACIÓN DE MOVIMIENTO

4.1 INTRODUCCIÓN

CONTROL POR CAMPOS POTENCIALES

Los campos potenciales artificiales son campos de fuerzas generados por obstáculos y metas. Esta teoría surge de una analogía con la electrostática, en donde cargas de signos contrarios se atraen y cargas de igual signo se repelen; en este caso los robots móviles y los obstáculos presentan una carga de igual signo y de diferente signo con respecto a las metas.

El control por campos potenciales toma a los campos potenciales artificiales como vectores de fuerza que representan una distancia repulsiva (al obstáculo) y una distancia atractiva (a la meta) ejercidos sobre el robot móvil. Dicha fuerza ejercida sobre el robot móvil, le es calculada el gradiente con la finalidad de obtener la dirección hacia la cual va cambiando la fuerza con lo que se obtiene un vector de dirección que va a guiar al robot móvil al punto deseado.

El vector de fuerza es utilizado como un vector de velocidad, el cual nos permite calcular comandos de velocidad. Para evitar vectores de velocidad infinitos o excesivamente grandes, se define una velocidad máxima y un área de evasión alrededor del objetivo a partir del cual la velocidad comienza a disminuir, esto es porque el vector de velocidad es directamente proporcional a la distancia existente entre ambos objetos.

Este tipo de control permite una planeación reactiva, es decir es una planeación hecha en el momento y sin intervención humana con la finalidad de llevar al robot móvil a su destino. A su vez, el control por campos potenciales pueden ser aplicados con el sólo conocimiento del modelo cinemático del robot.

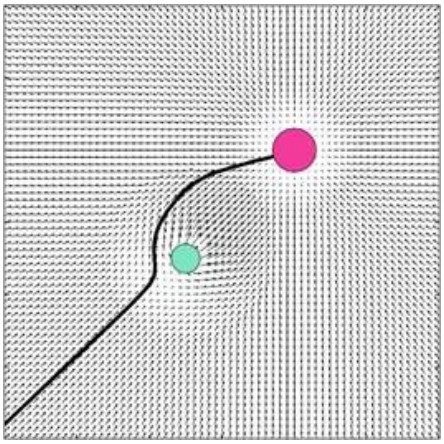


Figura 4.1 Control por Campos Potenciales

4.2 PLANIFICACIÓN DE LA TRAYECTORIA DE LA PLATAFORMA MÓVIL

El control de robots móviles por campos potenciales presentado por (González-Villela, 2004), es el acercamiento que será implementado para el control de la plataforma de nuestro manipulador móvil omnidireccional redundante. En la Figura 4.2, se muestra el vector de atracción hacia la meta d_g, el ángulo de orientación de este vector θ_g, el ángulo de orientación de la plataforma móvil θ_p y el ángulo de error de orientación θ_e. El radio de evasión k_r y el radio de seguridad del móvil k_{owr} son definidos por el usuario para poder evitar colisiones.

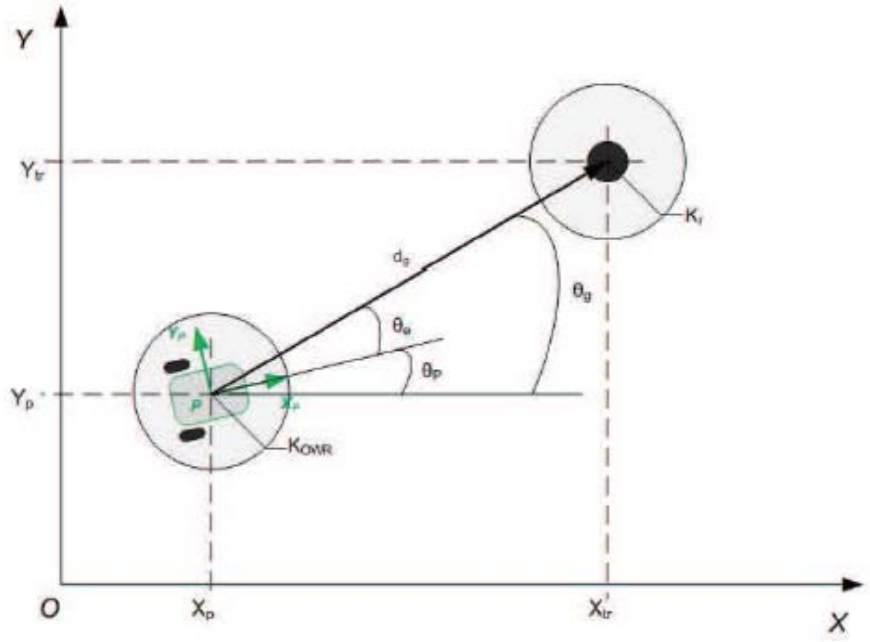


Figura 4.2 Definición de la ley de control por campos potenciales (Martínez-Carrillo, 2011)

Teniendo definidas las velocidades lineales como:

$$v_{xp} = \begin{cases} v_{xmax}\cos\theta_e\,, & si\ |d_g| > k_r + k_{owr} \\ \frac{v_{xmax} d_g \cos\theta_e}{k_r}, & si\ |d_g| < k_r + k_{owr} \\ 0, & si\ |d_g| \le k_{owr} \end{cases} \tag{4.1}$$

$$v_{yp} = \begin{cases} v_{ymax}\cos\theta_e\,, & si\ |d_g| > k_r + k_{owr} \\ \frac{v_{ymax} d_g \cos\theta_e}{k_r}, & si\ |d_g| < k_r + k_{owr} \\ 0, & si\ |d_g| \le k_{owr} \end{cases} \tag{4.2}$$

Las ecuaciones (4.1) y (4.2) previamente mostradas, definen la ley de control para la entrada de las velocidades lineales v_{xp} y v_{yp} para la tarea de llegar al objetivo. Además, la orientación de la plataforma puede ser controlada definiendo la siguiente ley de control.

$$\omega_p = \omega_{max} \sin \theta_E \tag{4.3}$$

Donde θ_E es el error entre la orientación actual θ_p y la orientación deseada θ_d del robot móvil omnidireccional redundante. ω_p es la velocidad angular de la plataforma móvil y ω_{max} es la velocidad angular máxima definida.

4.3 OPERACIÓN PICK AND PLACE

El movimiento que realice un brazo manipulador debe ser, por regla, tan suave como sea posible, esto quiere decir que los cambios de posición, velocidad y aceleración no deben de ser abruptos, ya que dichos cambios requieren de grandes cantidades de energía para ser efectuados. Para esto se utiliza la técnica para planear trayectorias de "pick and place".

En la operación "pick and place", se requiere que el brazo manipulador lleve a un objeto de una posición y orientación inicial a una posición y orientación final. Para la planeación de la trayectoria en el espacio cartesiano o global, se requerirán conocer los puntos; inicial y final del órgano terminal, mientras que en el espacio de juntas se encontrará el movimiento del manipulador serial de 5GDL.

4.3.1 DEFINICIÓN DE TRAYECTORIA

La planeación de la trayectoria de un brazo manipulador está compuesta por un lugar geométrico y un perfil de trayectoria. En este trabajo, no se define un perfil de trayectoria, ya que el movimiento del brazo no se encontrará relacionado con un tiempo específico, sino más bien se utilizará un algoritmo de intuición artificial el cual controlará el movimiento del manipulador serial de 5GDL.

4.3.1.1 INTUICIÓN ARTIFICIAL

La intuición artificial es una rama de la inteligencia artificial que se define como una representación limitada de las capacidades intuitivas del ser humano. Se dedica a emular el reconocimiento automático de patrones de información para generar respuestas rápidas y confiables. Creando algoritmos sintetizados que representan mecanismos inconscientes compuestos por conocimiento pre-adquirido, nos generan una solución única para un problema en particular disminuyendo el tiempo de búsqueda y cantidad de procesamiento (Díaz-Hernández, 2014).

El uso de intuición artificial da como resultado movimientos más naturales, ya que emula algunas características de la intuición humana. Esto permite mejorar el desempeño del robot, siendo la tarea de manipulación de objetos un área potencial para ser usada. Cabe señalar que la intuición artificial no posee mecanismos de aprendizaje, ya que éste necesita un alto poder de procesamiento.

El acercamiento a la intuición artificial es para realizar una coordinación de movimiento más natural para la operación de "pick and place" por parte del manipulador serial de 5GDL y el movimiento de la base móvil omnidireccional redundante, ya que al estar juntos se presenta la ventaja de incrementar el área de trabajo indefinidamente debido a la ausencia de alguna restricción que limite el movimiento.

4.3.1.2 DETERMINACIÓN DEL LUGAR GEOMÉTRICO

Para la determinación del lugar geométrico, se consideraron los puntos inicial y final en los que se tiene que encontrar nuestro efector final. Durante el análisis cinemático en el Capítulo 2, se mencionó que la trayectoria del brazo manipulador será descrita por los primeros 3GDL y los 2GDL restantes se encargarán de describir la orientación del efector final al momento de tomar y colocar el objeto.

Para el cálculo del espacio geométrico se utilizó el software *Wolfram Mathematica*, en el cual se programaron los vectores de posición que describen la cinemática del manipulador serial y se resolvió el sistema con un análisis numérico de Newton-Raphson, con el cual se obtiene el espacio de puntos por los que pasa el sistema, así como el posicionamiento angular de cada articulación del manipulador serial para los primeros 3GDL. Se hizo una obtención de 200 posiciones del manipulador serial durante cada etapa del estado inicial hacia el estado final.

El lugar geométrico que describirá el movimiento completo del brazo durante la tarea, se dividió en 4 etapas en las cuales se tienen que describir las ecuaciones que muestren el camino.

- Pick
- Pick-Back
- Place
- Return

❖ Etapa 1: Pick

Este movimiento es el que el brazo tiene que ejecutar para poder sostener el objeto a manipular.

Las ecuaciones que describen el movimiento del brazo manipulador, al realizar una regresión polinómica en *Excel* sobre los datos obtenidos por *Wolfram Mathematica*, son:

$$\theta_1 = 0.0000010298x^3 - 0.0008045680x^2 + 0.2882862394x + 89.9926123357 \tag{4.4}$$

$$\theta_2 = -0.0000001708x^4 + 0.0000789183x^3 - 0.0115734416x^2 - 0.2337581307x + 153.5629845608 \tag{4.5}$$

$$\theta_3 = -0.0000000739x^4 + 0.0000346464x^3 - 0.0073229767x^2 + 0.2413659558x + 112.4207944097 \tag{4.6}$$

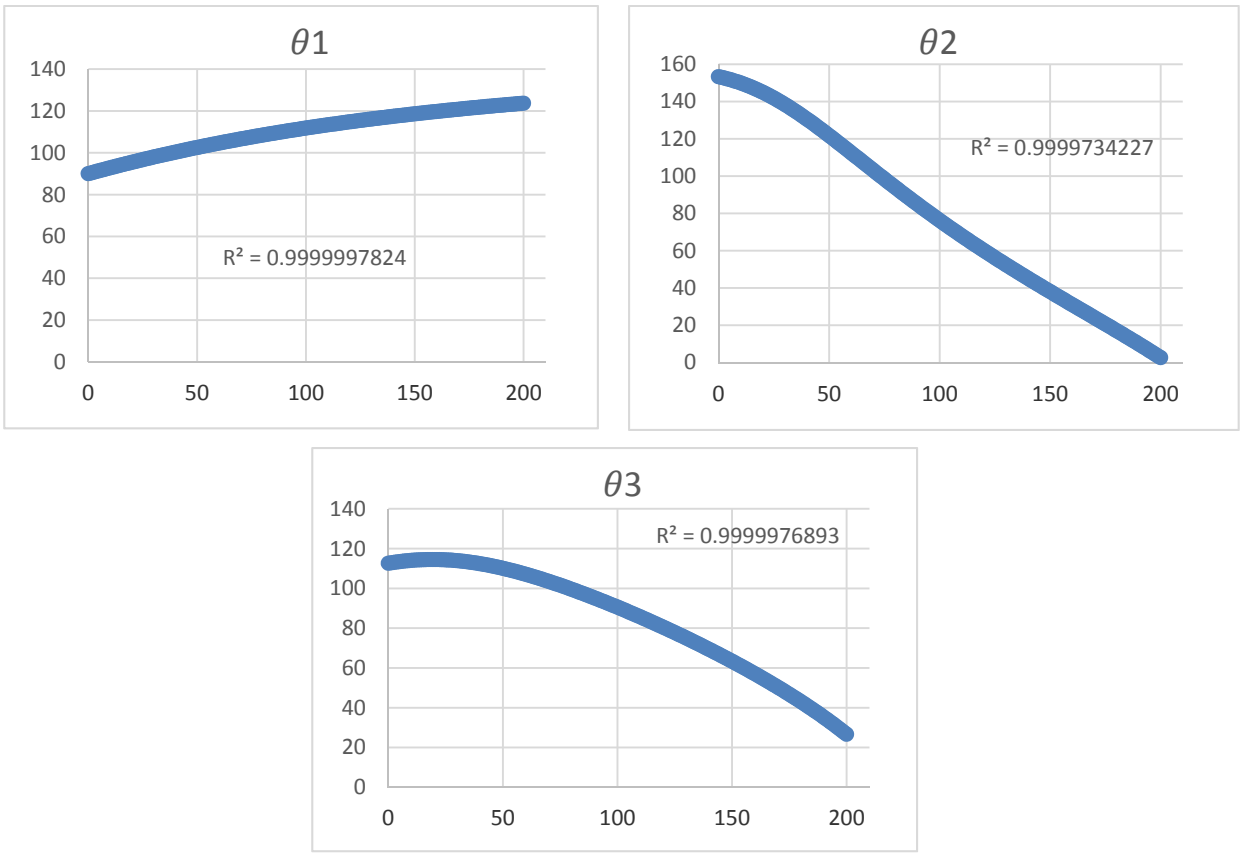


Figura 4.3 Gráficas de posición de la etapa "Pick"

❖ Etapa 2: Pick-Back

Este movimiento es el que se realiza una vez tomado el objeto, el brazo tiene que seguir el mismo espacio geométrico que en la etapa "pick" solamente que en sentido contrario; mientras se mantiene el objeto sujetado.

Las ecuaciones que describen el movimiento del brazo manipulador, al realizar una regresión polinómica en *Excel* sobre los datos obtenidos por *Wolfram Mathematica*, son:

$$\theta_1 = -0.0000010298x^3 - 0.0001866796x^2 - 0.0900367277x + 123.7056531859 \tag{4.7}$$

$$\theta_2 = -0.0000001708x^4 + 0.0000577554x^3 - 0.0052245667x^2 + 0.8598857647x + 1.8727625735 \tag{4.8}$$

$$\theta_3 = -0.0000000739x^4 + 0.0000244590x^3 - 0.0042667802x^2 + 0.8944770131x + 26.7350455199 \tag{4.9}$$

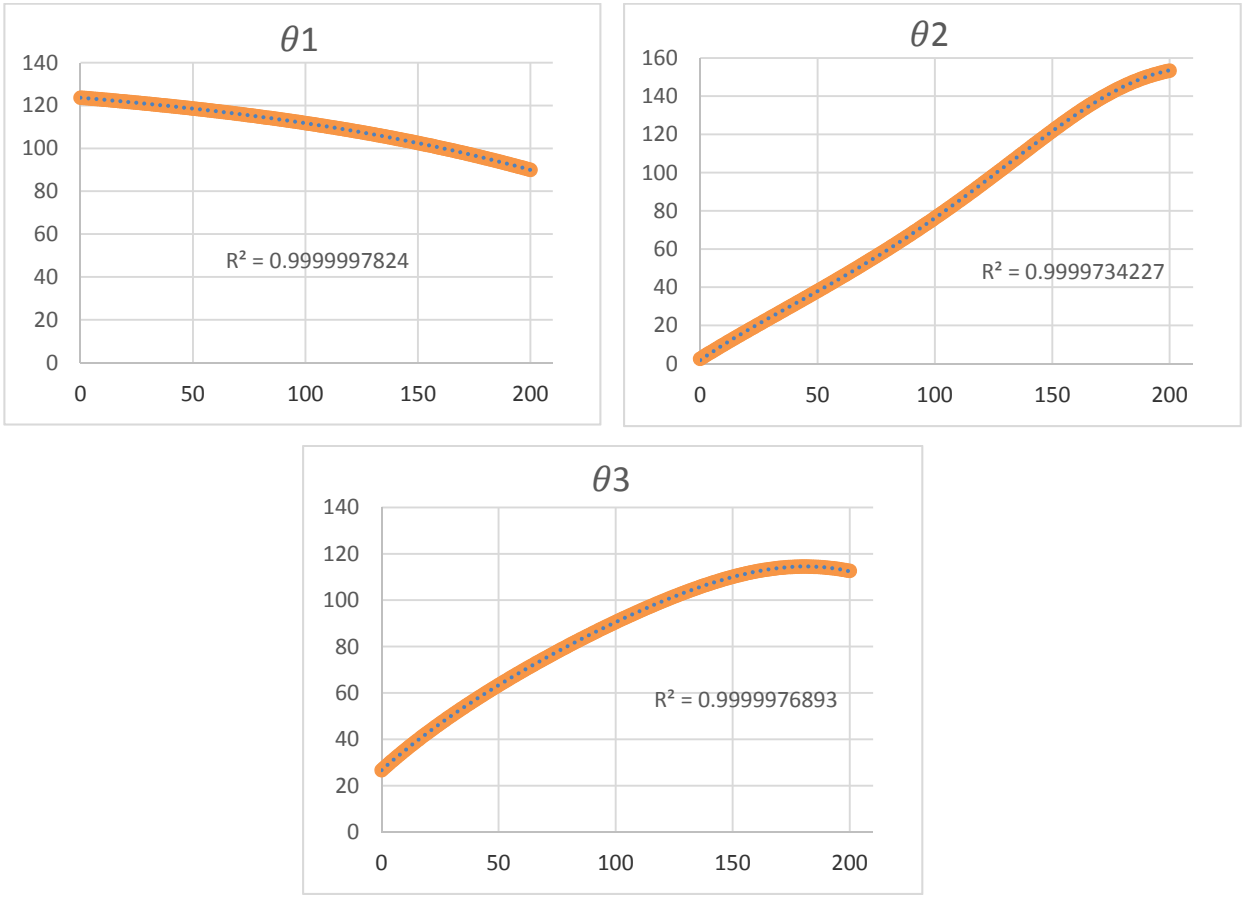


Figura 4.4 Gráficas de posición de la etapa "Pick-Back"

❖ Etapa 3: Place

Este movimiento es el que se realiza una vez tomado el objeto, después de haber ejecutado la etapa "pick-back". El brazo manipulador tiene que ejecutar el movimiento para situar el objeto en su destino, mientras éste se mantiene sujetado y soltarlo.

Las ecuaciones que describen el movimiento del brazo manipulador, al realizar una regresión polinómica en *Excel* sobre los datos obtenidos por *Wolfram Mathematica*, son:

$$\theta_1 = -0.0000010298x^3 + 0.0008045680x^2 - 0.2882862394x + 90.0073876643 \tag{4.10}$$

$$\theta_2 = -0.0000001708x^4 + 0.0000789183x^3 - 0.0115734416x^2 - 0.2337581307x + 153.5629845608 \tag{4.11}$$

$$\theta_3 = -0.0000000739x^4 + 0.0000346464x^3 - 0.0073229767x^2 + 0.2413659558x + 112.4207944097 \tag{4.12}$$

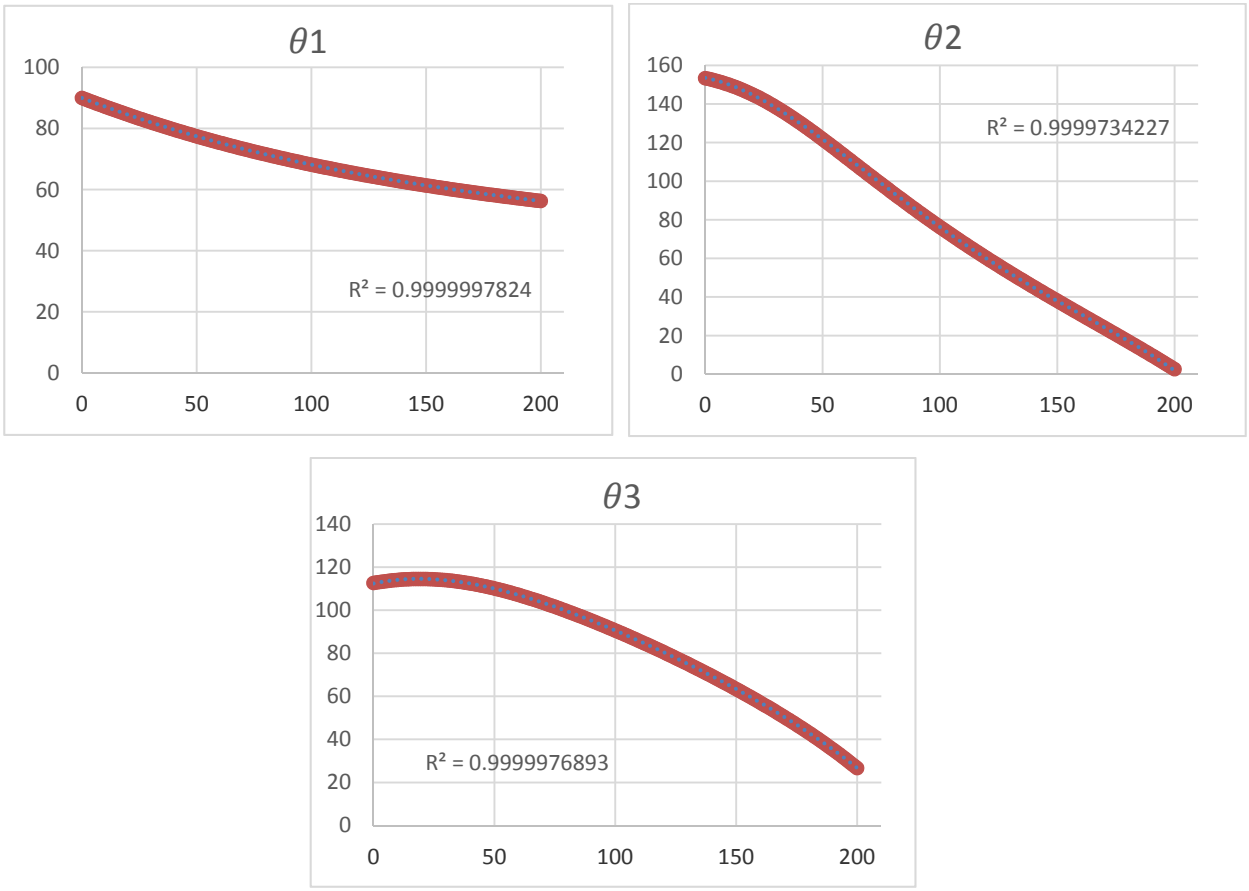


Figura 4.5 Gráficas de posición de la etapa "Place"

- Etapa 4: Return

Durante esta última etapa, el manipulador serial de 5GDL habrá soltado el objeto y tiene que colocarse en la posición inicial de la operación. Para ello, de manera similar a la etapa de "Pick-Back" se reutilizará el espacio geométrico recorrido anteriormente, en este caso el espacio geométrico de la etapa "Place".

Por ser la última etapa de la tarea a realizar, la programación nos facilitará llevar a cabo el recorrido de regreso utilizando las mismas ecuaciones de la etapa de "Place" sin tener que realizar una nueva regresión polinómica a los datos.

En el caso de la etapa de "Pick-Back", debido a que esta se encuentra en medio de la tarea, la programación dificulta su regreso sin el apoyo de nuevas ecuaciones. Esto se aclara en el código de programación localizado en los apéndices.

4.3.1.3 ORIENTACIÓN DEL EFECTOR FINAL

Ahora, se definirá el espacio geométrico para el efector final. La definición de los movimientos efectuados se basa en que la orientación del efector final se dé ya que el móvil está cerca del objetivo, ya sea para tomar o dejar el objeto.

Dicho lo anterior, se establece que la mayor parte de la tarea el efector final permanecerá sin moverse. Se requiere que los movimientos del efector final estén descritos en 200 posiciones al igual que los datos que fueron generados para la posición, esto es para facilitar la coordinación de movimientos. Así mismo, se tendrá dividida la tarea en 4 etapas en las cuales se definirán los movimientos a realizar.

Debido a que el movimiento solo se requiere al final de cada etapa, de las 200 posiciones que definen la etapa serán modificadas las últimas 30 posiciones con la finalidad de orientar el efector final y agarrar el objeto con el "*gripper*". Para realizar una orientación simple, uno de los ángulos involucrados se mantendrá fijo.

- Etapa 1: Pick

Este movimiento es el que el efector final tiene que ejecutar para poder agarrar el objeto a manipular. Los ángulos definidos para los primeros 170 movimientos fijos son:

$$\theta_4 = 120 \tag{4.13}$$

$$\theta_5 = 90 \tag{4.14}$$

$$\theta_6 = 15 \tag{4.15}$$

Donde θ_6 es el ángulo que se le envía al servomotor que se encuentra en el gripper para abrir y cerrar la pinza.

Las ecuaciones que describen el movimiento del efector final del manipulador, al realizar una regresión polinómica en *Excel* sobre los datos definidos, son:

$$\theta_4 = -0.0000436215x^4 + 0.0347848868x^3 - 10.3385682795x^2 + 1{,}357.3638320434x - 66{,}309.8035615282 \tag{4.16}$$

$$\theta_5 = 90 \tag{4.17}$$

$$\theta_6 = -0.0006139017x^4 + 0.4521241747x^3 - 124.6505794176x^2 + 15{,}248.7607203834x - 698{,}423.5903937180 \tag{4.18}$$

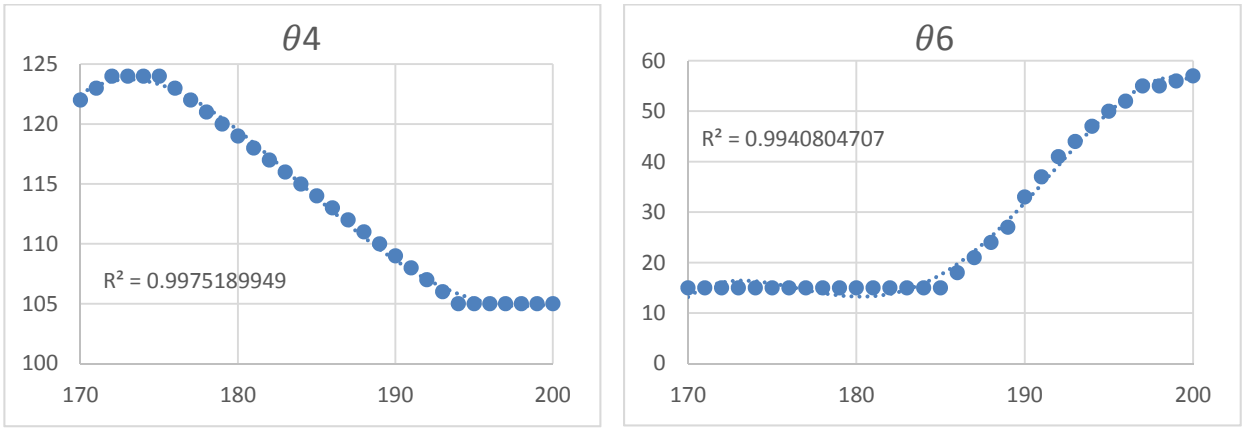


Figura 4.6 Gráficas de orientación de la etapa "Pick"

- Etapa 2: Pick-Back

Este movimiento es el que se realiza una vez tomado el objeto, el efector final regresará a la posición inicial definida del espacio geométrico que en la etapa "pick" de manera inmediata; mientras se mantiene el objeto sujetado. Esto es definido así ya que no colocará el objeto en ninguna parte durante esta etapa de la tarea.

$$\theta_4 = 120 \tag{4.19}$$

$$\theta_5 = 90 \tag{4.20}$$

$$\theta_6 = 60 \tag{4.21}$$

❖ Etapa 3: Place

Este movimiento es el que se realiza después de haber ejecutado la etapa “pick-back”. El efector final del manipulador tiene que ejecutar el movimiento para situar el objeto en su destino, mientras éste se mantiene sujetado y soltarlo. En este caso se mantendrá una orientación fija y sólo se soltará el objeto.

Las ecuaciones que describen el movimiento del brazo manipulador, al realizar una regresión polinómica en *Excel* sobre los datos definidos, son:

$$\theta_4 = 18 \tag{4.22}$$

$$\theta_5 = 90 \tag{4.23}$$

$$\theta_6 = 0.0005706934x^4 - 0.4255357425x^3 + 118.7507632308x^2 - 14{,}700.8841651713x + 681{,}334.3318258550 \tag{4.24}$$

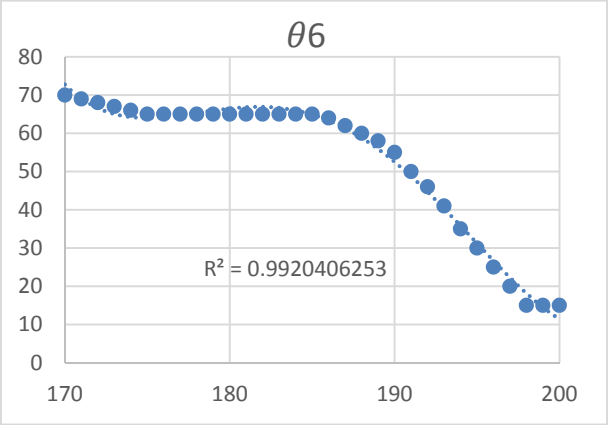


Figura 4.7 Gráfica de orientación de la etapa “Place”

❖ Etapa 4: Return

Durante esta última etapa, el manipulador serial de 5GDL habrá soltado el objeto y tiene que colocarse en la posición inicial de la operación. Para ello, de manera similar a los movimientos de posición del manipulador, se regresará por el mismo espacio geométrico de la etapa “Place”.

De manera análoga, al ser la última etapa de la tarea a realizar, la programación nos facilitará llevar a cabo el recorrido de regreso utilizando las mismas ecuaciones de la etapa de “Place” sin tener que realizar una nueva regresión polinómica a los datos.

4.4 COORDINACIÓN DE MOVIMIENTOS

Como se menciona en el Capítulo 1, el problema principal que se tiene en los manipuladores móviles ha sido típicamente el de coordinar la locomoción y la manipulación. Esto se debe particularmente a que la solución de los parámetros de control es redundante ya que el manipulador serial cuenta con más grados de libertad de control necesarios para dar una solución única (D. H. Shin, 2003).

En el caso de nuestro manipulador móvil omnidireccional redundante es muy importante, ya que la base móvil es redundante dificultando el problema al unirlo con el manipulador serial. Al usar un robot móvil que es omnidireccional y redundante, la limitación de movimientos se ve reducida significativamente mejorando la coordinación.

4.4.1 CRITERIOS DE MOVIMIENTO

Una vez definida la ley de control que seguirá la plataforma móvil omnidireccional redundante y el espacio geométrico del manipulador serial de 5GDL, se tiene que definir la coordinación de estos dos sistemas. En este apartado se muestra la forma para coordinar los movimientos, tomando como base el principio de la intuición artificial, lo que permitirá movimientos más naturales.

Para resolver el problema, se tienen que tener en cuenta varios factores involucrados a lo largo de la tarea, la cual de forma general consiste en ir a una meta a tomar el objeto y trasladarlo a la siguiente meta para colocarlo. La distancia, resulta ser el factor crítico a considerar en todo momento para realizar los movimientos de manera coordinada sin depender del tiempo. Para ello se tiene que calcular la distancia total al inicio de la tarea y, a su vez, la distancia entre el robot y el objetivo.

$$Dist_{Meta_i} = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{4.25}$$

$$Dist_{Movil_{Meta_i}} = \sqrt{(x_i - x_{movil})^2 + (y_i - y_{movil})^2} \tag{4.26}$$

Durante el movimiento de una meta a otra, el brazo manipulador tiene que moverse a lo largo de su espacio de trabajo. Para ello, se tiene que relacionar la distancia total y la distancia actual del robot móvil omnidireccional redundante con la posición actual del manipulador serial. En la sección anterior se mencionó que se generaron 200 posiciones, las fórmulas generadas para la relación del manipulador serial con la base móvil se describen a continuación.

$$E_i = 1 - \frac{Dist_{Movil_{Meta_i}}}{Dist_{Meta_i}} \tag{4.27}$$

Donde E_i es el factor de escala que permitirá conocer en qué movimiento necesita estar el manipulador serial dada la posición actual del robot móvil omnidireccional redundante. Para las distintas etapas de la tarea, es necesario definir diferentes ecuaciones que permitan definir de manera clara la posición del manipulador serial de 5GDL.

$$Pos_{Etapa1} = 200 * E_i \tag{4.28}$$

$$Pos_{Etapa2} = 200 + 200 * (2 * E_i) \tag{4.29}$$

$$Pos_{Etapa3} = 400 + 200 * \big(2 * (E_i - 0.5)\big) \tag{4.30}$$

$$Pos_{Etapa4} = 600 - 200 * E_i \tag{4.31}$$

4.4.2 ALGORITMO DE COORDINACIÓN

En las secciones anteriores se describe la estrategia para tratar ambas partes del sistema como uno solo y realizar el movimiento del sistema en su totalidad. Finalmente, para poder coordinar ambos sistemas, se establece el algoritmo coordinante de ambos sistemas, el cual se explica en los siguientes párrafos.

- Se detecta la posición inicial del manipulador móvil omnidireccional redundante, del objeto a manipular y de las metas a las que se quiere llegar.
- La distancia total entre las metas se calcula con (4.25).
- Se inicia con la primera etapa de la tarea; "*Pick*". Durante esta tarea, ejecutar en todo momento:
  - Se calcula la distancia entre el robot móvil y la meta 1 con (4.26).
  - Se calcula el factor de escala a utilizar con (4.27).
  - Se posiciona el manipulador serial de 5GDL en la posición indicada por (4.28).
- Se inicia con la segunda etapa de la tarea; "*Pick-Back*". Durante esta tarea, ejecutar en todo momento:
  - Se calcula la distancia entre el robot móvil y la meta 2 con (4.26).
  - Se calcula el factor de escala a utilizar con (4.27).
  - Se posiciona el manipulador serial de 5GDL en la posición indicada por (4.29).
  - Pasar a la siguiente tarea cuando el factor de escala (4.27) sea mayor a 0.5.
- Se inicia con la tercera etapa de la tarea; "*Place*". Durante esta tarea, ejecutar en todo momento:
  - Se utiliza la distancia entre el robot móvil y la meta 2 calculada en "*Pick-Back*".
  - Se calcula el factor de escala a utilizar con (4.27).
  - Se posiciona el manipulador serial de 5GDL en la posición indicada por (4.30).
- Se inicia con la última etapa de la tarea; "*Return*". Durante esta tarea, ejecutar en todo momento:
  - Se calcula la distancia entre el robot móvil y la meta 3 (el inicio) con (4.26).
  - Se calcula el factor de escala a utilizar con (4.27).
  - Se posiciona el manipulador serial de 5GDL en la posición indicada por (4.31).
- Se coloca al manipulador serial de 5GDL en su configuración inicial.

4.5 IMPLEMENTACIÓN FÍSICA

Durante éste capítulo se trataron las leyes de control y coordinación de movimientos que regirán el comportamiento del manipulador móvil omnidireccional redundante.

Se le programará el control por campos potenciales a la plataforma móvil omnidireccional redundante, el cual calculará en todo momento la ruta óptima que tiene que seguir el robot.

El movimiento que presenta el manipulador serial de 5DGL se controlará con el algoritmo generado para la coordinación de movimientos, el cual seguirá la trayectoria descrita para la operación de “pick and place” del objeto deseado.

El objetivo a conseguir es trasladar un objeto teniendo un movimiento suave en el sistema, de forma que parezca más natural, tomando en cuenta lo que plantea la intuición artificial.

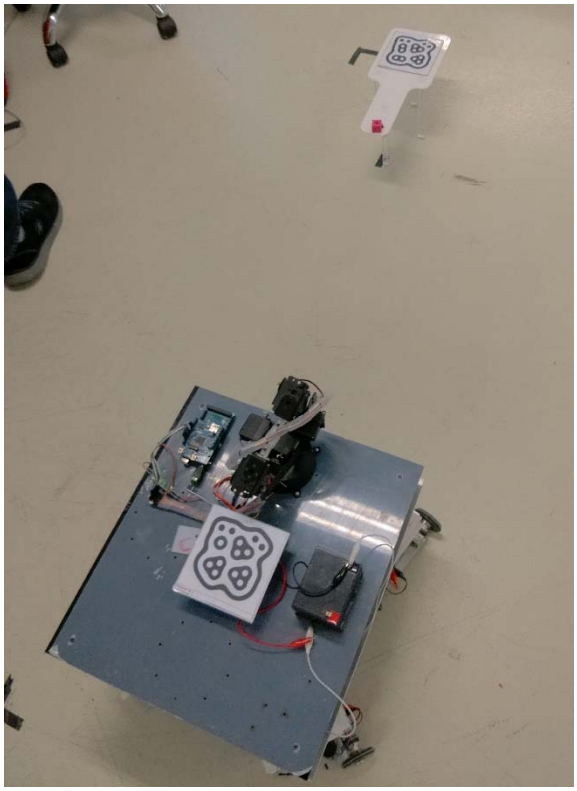


Figura 4.8 Implementación

CAPITULO 5 DESCRIPCIÓN DEL AMBIENTE INTELIGENTE Y DEL SISTEMA DE PRUEBAS

5.1 INTRODUCCIÓN

El concepto de espacio inteligente ha sido propuesto por diversos autores y ha ido evolucionando separándose en distintas implementaciones, incluyendo aquellas aplicaciones destinadas a la localización y navegación de robots.

Los espacios inteligentes son habitaciones o áreas capaces de percibir y entender lo que ocurre en ellas. Están equipados con diferentes tipos de sensores, actuadores y dispositivos de comunicación. A través de los sensores, los espacios inteligentes captan lo que ocurre en ellos, analizan las diferentes situaciones, reaccionan ante ellas y se comunican con los usuarios a través de los actuadores (Kazuyuki Morioka, 2008).

Lo que hace que un espacio sea denominado inteligente es cuando toma decisiones de manera autónoma, debe ser capaz de "estar consciente" de lo que pasa dentro de él, y así poder actuar de acuerdo a lo deseado dentro del espacio. El objetivo es definir espacios donde exista una interacción no intrusiva entre usuarios y el propio entorno, teniendo ésta como objetivo el de ayudar al usuario en algún tipo de tarea.

Para entender la estructura de los "Espacios Inteligentes" es necesario definir una serie de conceptos, los cuales (Pé, 2007) los presenta de la siguiente manera:

Entorno: se considera una entidad fija en el espacio. En dicho entorno se producirán una serie de eventos que deberán ser observados, analizados y comprendidos por el entorno. A su vez, el entorno se divide en varias capas funcionales:

- *Capa Pasiva de Percepción:* Está compuesta por un número de sensores que permiten captar información del espacio de trabajo que será utilizada por el entorno. Los sensores que componen esta capa pueden ser de distintos tipos, estar fijos en el espacio o incluso ir unidos a la *Capa de Interacción*.
- *Capa de Interacción:* Esta capa se compone de un conjunto de *Agentes Controlables* que pueden interactuar físicamente con el mundo. Esta capa tiene doble funcionalidad: por un lado, se utiliza para cumplir con los objetivos del entorno para el que fue diseñado y por otro lado, puede utilizarse como parte de la *Capa de Percepción Activa*. Debido a que estos agentes interaccionan con el mundo, tendrán que estar dotados a su vez de un sistemas sensorial, por lo que la información que captan puede utilizarse para complementar la información captada por los propios sensores del entorno.
- *Capa de Comunicaciones:* Los *Agentes Controlables* y eventualmente los *Agentes Autónomos* se encuentran conectados al entorno a través de una red de comunicaciones que representa el sistema nervioso central del "Espacio Inteligente". Esta red conecta la *Capas de Percepción* y la *Capa de Interacción* con la *Capa de Inteligencia*.
- *Capa de Inteligencia*: Se necesita un sistema de procesado que sea capaz de gestionar toda la información captada por los sensores, que genere bases de datos con toda esta información y

que permita el intercambio de información con los distintos agentes del entorno. No se compone necesariamente de una gran unidad de proceso central y se estructura en una jerarquía de capas.

- *Objetivos del entorno:* En general, existen dos fuentes encargadas de fijar los objetivos del entorno. Por un lado están los *Agentes Autónomos* (por ejemplo, una persona) que se encuentran en el espacio y con los que el entorno tiene que establecer una comunicación a través de las *Capas de Percepción*. Por otro lado, puede haber agentes que se encuentren conectados a la *Capa de Comunicaciones*.

La interacción del entorno con el resto del mundo se realiza mediante la interpretación realizada en la *Capa de Percepción* y los *Agentes Controlables*.

Mundo de Percepción: La Capa de Percepción y la Capa de Acción están relacionadas con los eventos que tienen lugar en el entorno. Dichos eventos pueden catalogarse como:

- *Agentes Autónomo*: Desarrolla un tipo de actividad en el entorno sin que éste tenga control sobre él. Puede generar objetivos o simplemente puede ser un obstáculo no estático que los *Agentes Controlables* deben evitar.
- *Objetos Estáticos*: Los objetos estáticos son parte del conocimiento que el entorno debe de adquirir con el fin de desarrollar satisfactoriamente su capacidad de interacción y el cumplimiento de los objetivos.
- *Agentes Controlables*: Forman parte de la *Capa de Acción*, y el entorno sabe acerca de su existencia gracias a la *Capa de Comunicaciones*. Están presentes en el mundo de percepción del entorno, por lo que poseen una firma o apariencia desde el punto de vista de la *Capa de Percepción*.

En resumen, tenemos una serie de elementos en el entorno de distinta naturaleza. Independientemente del tipo de elemento y de su funcionalidad, el entorno debe ser capaz de detectarlos y posicionarlos espacialmente utilizando la información que aportan los distintos sensores del “Espacio Inteligente”.

La incorporación de robots móviles al concepto de “Espacio Inteligente” es una disciplina aún emergente, en la que no se han explotado todavía las posibilidades que ofrece. Las técnicas de visión artificial aplicadas hasta la fecha, en general, requieren de balizamiento y no aprovechan toda la información de la que se dispone desde el conjunto de cámaras.

La idea de un espacio inteligente ha ido en auge en los últimos años debido en gran medida a un decremento en el coste de los sensores y equipos de cómputo, llevado de la mano de una mayor capacidad de procesamiento. El desarrollo de espacios inteligentes en todo tipo de áreas permitirá *ambientes inteligentes*, es decir un conjunto de espacios inteligentes.

Entonces, se tiene que los ambientes inteligentes describen y manejan entornos físicos, es decir, son espacios que usan la tecnología de sistemas embebidos así como otras tecnologías de la información y la comunicación, para crear ambientes interactivos que acerquen la computación al mundo físico y a los problemas cotidianos, para introducir mejoras imperceptibles o superficiales en las actividades comunes. Estos sistemas pasan mayoritariamente desapercibidos para los usuarios, puesto que se hallan discretamente integrados a objetos físicos, a infraestructuras, y al entorno cotidiano en el cual vivimos, viajamos, y trabajamos.

5.2 ARQUITECTURA

En el caso del presente trabajo se presenta un ambiente inteligente que será capaz de “ser consciente” de las cosas que ocurren dentro del espacio de trabajo, contando con las cuatro capas del entorno, donde habrán agentes controlables, objetos estáticos y en base a ello tomar decisiones sobre la tarea a desarrollar. Este ambiente inteligente define lo que es el sistema de pruebas.

El sistema de pruebas está constituido por tres sub sistemas. El subsistema del entorno, el subsistema de visión artificial y el subsistema de cómputo.

El subsistema del entorno está compuesto por el manipulador móvil omnidireccional redundante, dos mesas objetivo y un objeto a manipular.

El subsistema de visión artificial lo compone una cámara web y figuras especiales que marcan la posición del objetivo (fiducials).

El subsistema de cómputo es un ordenador central el cual se encarga de procesar la información que se recibe del subsistema de visión y del subsistema del entorno y enviar los movimientos que deben ejecutarse. El intercambio de información entre los subsistemas se realiza mediante el protocolo de comunicación inalámbrico WiFi-UDP.

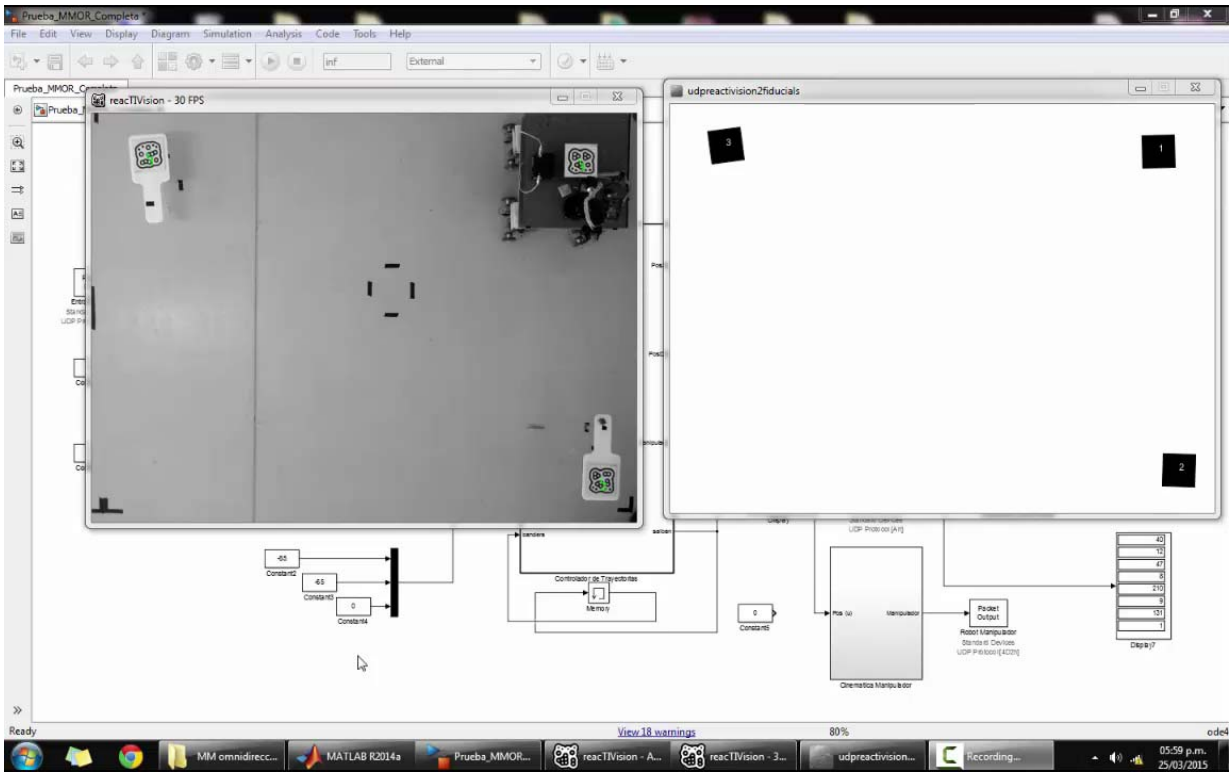


Figura 5.1 Sistema de Pruebas

A continuación, se describen más a detalle los componentes del sistema de pruebas.

5.3 VISIÓN ARTIFICIAL

La visión artificial es la capacidad de la máquina para ver el mundo que le rodea, más precisamente para reducir la estructura y las propiedades del mundo tridimensional a partir de una o más imágenes bidimensionales (Martinsanz, 2008).

El proceso fundamental de un sistema de visión puede definirse como sigue:

1) **Adquisición de imágenes**: Por medio de una cámara la imagen es adquirida.
2) **Procesamiento de la imagen**: La imagen pasa por una serie de fases que transforman y filtran la información para facilitar la extracción de parámetros de importancia.
3) **Extracción de información**: Se obtienen los parámetros importantes para la realización de una determinada tarea, como ejemplo está la posición y orientación de un robot manipulador con respecto a un objeto a transportar.

5.3.1 REACTIVISION

ReacTIVision es un código abierto multiplataforma, de visión por computador para el seguimiento rápido y robusto de los marcadores de referencia adjuntos a los objetos físicos. Fue diseñado principalmente como un conjunto de herramientas para el rápido desarrollo de interfaces basados en tablas tangibles de usuario (TUI) y superficies interactivas multi-touch (reacTIVision, 2015).

ReacTIVision es una aplicación independiente, que envía mensajes TUIO a través del puerto UDP 3333 para cualquier aplicación cliente TUIO activo.

La aplicación ReacTIVision se ejecuta actualmente en los siguientes sistemas operativos: Windows, Mac OS X y Linux. En Windows, es compatible con cualquier cámara con un controlador WDM adecuado, como la mayoría de las cámaras USB, FireWire y DV.

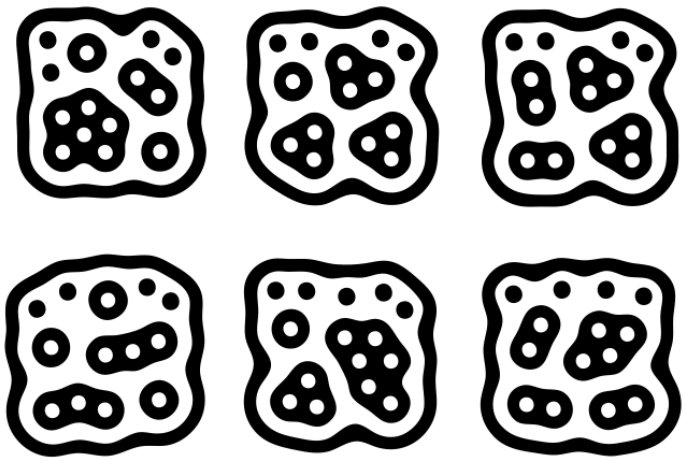


Figura 5.2 Marcadores Fiducial

ReacTIVision localiza los marcadores especialmente diseñados llamados “fiducial” en una secuencia de video en tiempo real. Esto lo hace convirtiendo la imagen de origen en blanco y negro con un algoritmo de umbral adaptativo, segmenta la imagen en un árbol alternando regiones negras y blancas, después se busca una secuencia de profundidad única la cual ha sido decodificada en el símbolo del fiducial (Figura 5.2). El diseño del fiducial permite el cálculo eficiente del centro del marcador, así como su orientación. Esta información, junto con un número ID único, es transmitida al cliente con mensajes OSC que implementan el código TUIO.

Para nuestro sistema en particular, se necesitarán tres marcadores para los componentes físicos que estarán presentes durante la tarea; el manipulador móvil omnidireccional redundante (ID 1), el objetivo 1 (ID 2) y el objetivo 2 (ID 3).

Figura 5.3 Fiducials del sistema

5.3.2 SOFTWARE DE ENLACE

El software de enlace es un programa desarrollado en el grupo de trabajo MRG, por el M.I. Pedro Gálvez. Tiene la función de establecer la comunicación entre el software de visión (ReacTIVision) y el software de procesamiento (Matlab) en tiempo real. El software está desarrollado en Processing.

El software primeramente obtiene la información que recibe ReacTIVision de la cámara web y la re-direcciona a un puerto UDP del ordenador para que pueda ser posteriormente leído por el software de procesamiento. Para que esto suceda, el software de procesamiento tiene que contener programados los parámetros necesarios para decodificar la información enviada.

La posición de los fiducials que es manejada se encuentra en una escala entre 0 y 100, donde 0 es la posición del origen del marco global de la imagen y 100 es la posición máxima que alcanza la imagen. Para poder representar esta escala en una métrica, se requiere hacer una calibración de la cámara la cual se encuentra orientada paralelamente al suelo y se miden las distancias que abarca la imagen y se aplica una ecuación de transformación.

El software de enlace nos permite de esta manera saber dónde se encuentra el origen del sistema y en qué coordenadas de posición y orientación en las que se encuentran nuestros componentes del entorno (x, y, θ). Permitiendo que nuestro sistema de pruebas quede referenciado de la siguiente manera:

Figura 5.4 Sistema de Visión Artificial

5.4 CÓMPUTO

El ordenador central se encarga de realizar el procesamiento de toda la información que proviene del sistema de visión y del manipulador móvil omnidireccional redundante, además de ejecutar en todo momento las coordinaciones intuitivas y el control por campos potenciales; necesarios para la generación de movimiento.

El procesamiento de la información se lleva a cabo con el uso de un software de visión (ReacTIVision), un software de enlace y un software de procesamiento (Matlab). Los primeros dos ya fueron previamente presentados, por lo que ahora corresponde mostrar el software de procesamiento.

5.4.1 MATLAB

Matlab es un lenguaje de alto nivel con un ambiente interactivo para computación numérica, visualización y programación. Usando MATLAB, puedes analizar, desarrollar algoritmos y crear modelos y aplicaciones. El lenguaje, herramientas y funciones matemáticas permiten que explores diferentes aproximaciones y llegar a la solución más rápidamente que con hojas de cálculo o lenguajes de programación tradicionales como C/C++ o Java (MATLAB Documentation, 2015).

De esta manera, la programación de la cinemática del manipulador móvil omnidireccional redundante se llevó a cabo en archivos con extensión ".c", pues de esta manera se ejecuta el código de manera más rápida y puede ser implementado en un solo bloque de Simulink. De la misma manera, se programó el código necesario para hacer los cambios necesarios en la información de la velocidad de los motores antes de ser enviada a los robots móviles diferenciales que constituyen a la plataforma móvil.

5.4.2 SIMULINK

Simulink® es un entorno de diagramas de bloque para la simulación multi-dominio y el diseño basado en modelos. Admite el diseño y la simulación a nivel de sistema, la generación automática de código y la prueba y verificación continuas de los sistemas embebidos. Simulink ofrece un editor gráfico, bibliotecas de bloques personalizables y solvers para modelar y simular sistemas dinámicos. Se integra con MATLAB®, lo que permite incorporar algoritmos de MATLAB en los modelos y exportar los resultados de la simulación a MATLAB para llevar a cabo más análisis (Simulink, 2015).

En Simulink, se realizó la programación y la integración del sistema de pruebas. En él se puede simplificar el código de bloques anidando varios bloques en uno más simple para una mejor apreciación del código. De esta manera, a continuación se muestran los bloques que componen al sistema y una descripción de su funcionamiento.

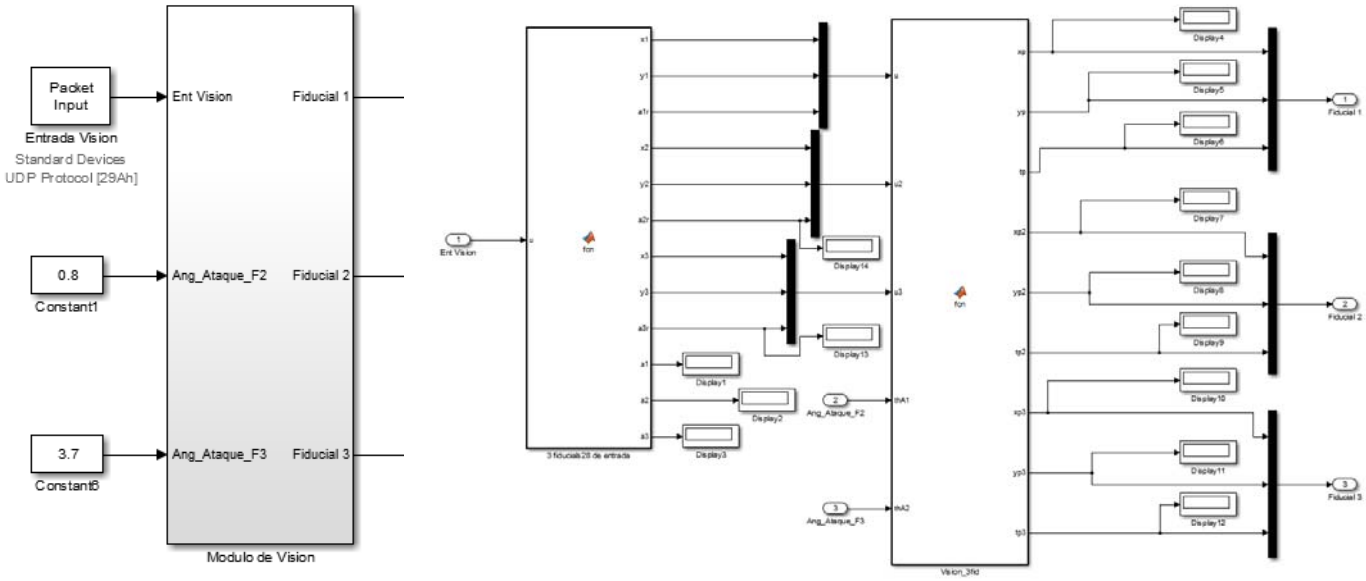


Figura 5.5 Bloque de Visión

Bloque de visión.

El bloque llamado “Modulo de Visión”, tiene como entradas un “*Packet input*” y dos constantes que corresponden al ángulo de ataque con el que llegará el manipulador móvil omnidireccional redundante a los objetivos, mientras que como salida tiene las coordenadas de posición y orientación (x, y, θ) de los tres fiducials que son identificados.

Dentro del bloque, se encuentran dos sub-bloques, uno es un bloque de función de Matlab que procesa la información recibida y la transmite al segundo bloque de función que realiza el ajuste de la escala para referenciar al sistema con el marco coordenado real, para así obtener a la salida las coordenadas de posición y orientación en centímetros y radianes.

Bloque de coordinación.

Es un bloque de función llamado “Coordinación de Trayectorias”, recibe como entradas la información proveniente del bloque de visión, las coordenadas del punto de partida del manipulador móvil omnidireccional redundante y una bandera de control que es retroalimentada de una de sus salidas. A las salidas tiene las coordenadas actuales del manipulador móvil omnidireccional redundante y del objetivo actual, la salida de control del manipulador serial de 5GDL y la bandera de control a retroalimentar.

En él se ejecuta el algoritmo de la coordinación intuitiva de movimientos con la que se le indicará al manipulador serial de 5GDL las posiciones que debe alcanzar y de la administración de objetivos a los que el manipulador móvil omnidireccional redundante tiene que dirigirse.

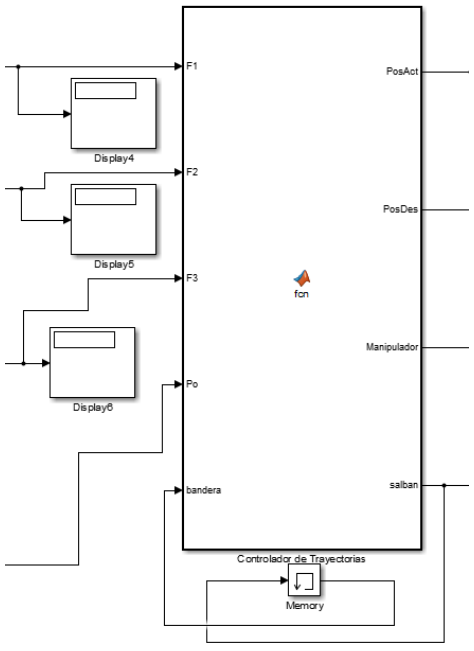


Figura 5.6 Bloque de Coordinación

Bloque cinemático del móvil

Es un bloque llamado “Cinemática RMOmniRed”, recibe como entradas la información proveniente del bloque de coordinación; las coordenadas actuales del manipulador móvil omnidireccional redundante y del objetivo actual, y un “*Packet input*” el cual contiene la orientación de los robots móviles diferenciales que es capturada por los encoders magnéticos absolutos. A las salidas se tienen las velocidades que se tienen que mandar a través de un “*Packet output*” de ambas llantas que tiene cada robot móvil diferencial.

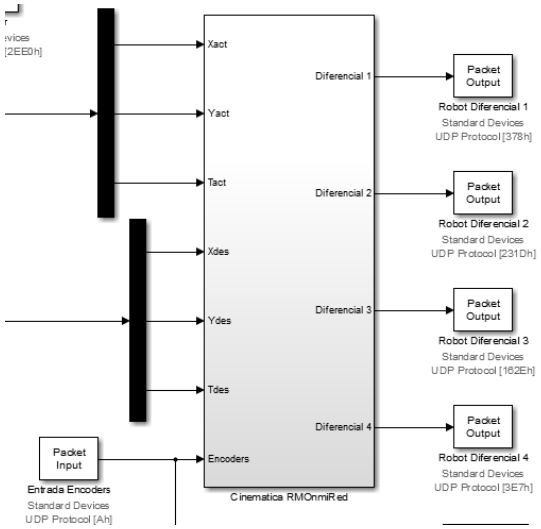


Figura 5.7 Bloque “Cinemática RMOmniRed”

Dentro de éste bloque se anidan varios sub-bloques que se encargan de todo el movimiento de la plataforma móvil. Los bloques que se encuentran dentro de “Cinemática RMOmniRed” son:

- controlador: Es un bloque S-Function en donde se ejecuta el control por campos potenciales con el que se determina las velocidades lineales y angulares que el móvil debe tener para alcanzar la referencia.
- Encoders: El un bloque que almacena sub-bloques para el procesamiento de la información recibida en su entrada (“*Packet input*”), transformándola en radianes teniendo a la salida los valores de cada uno de los encoders.
- cinematicaSX: Es un bloque S-Function el cual tiene como entrada la velocidad con la que tiene que moverse la plataforma y las orientaciones de cada robot móvil diferencial con respecto a la misma, provenientes de los bloques “controlador” y “Encoders”. En él se calcula la cinemática del robot móvil omnidireccional redundante para obtener las velocidades de las llantas de cada robot móvil diferencial.
- Modelo2Real: Es un bloque que almacena un sub-bloque S-Function (“Speed2MD25”), en el cual se procesa las entradas de velocidades de las llantas que salen del bloque “cinematicaSX” y ordena la información para poder ser interpretada fácilmente, teniendo a la salida las velocidades de las llantas de cada robot móvil diferencial ordenadas.
- uint8: Es un bloque especial de Simulink el cual convierte las unidades de la información a enviar. En nuestro caso, se requiere como unidades que sean enteros de 8 bits.

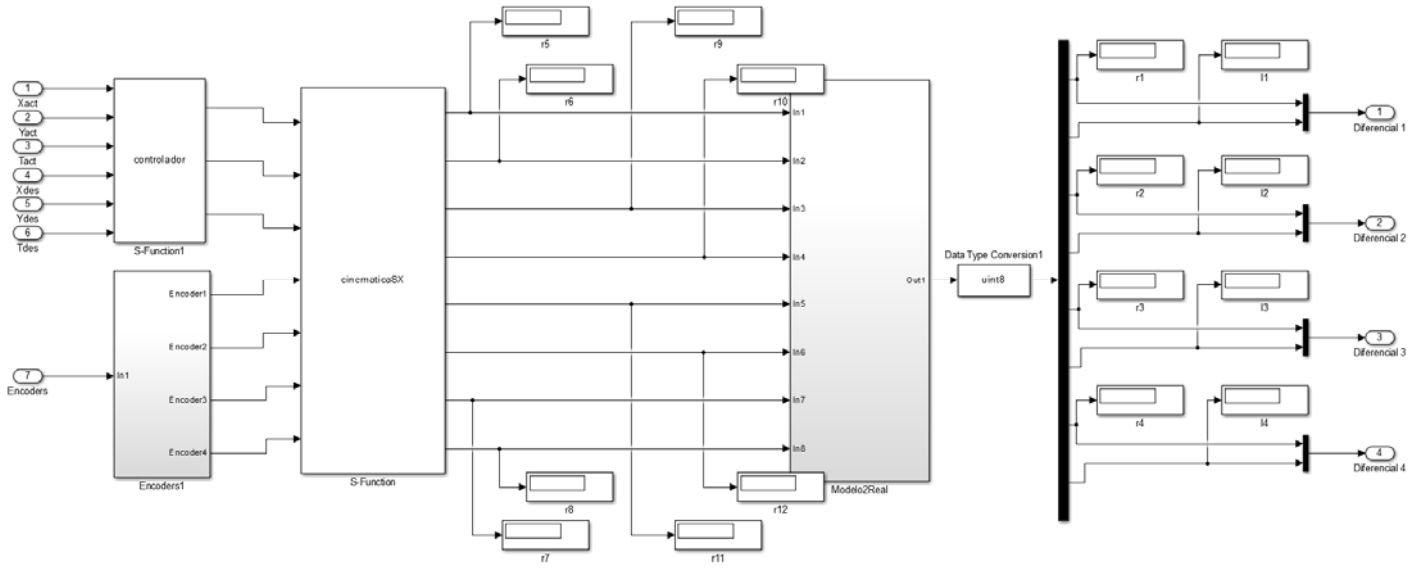


Figura 5.8 Contenido del bloque "Cinemática RMOmniRed"

Bloque cinemático del manipulador

Es un bloque llamado "Cinemática Manipulador", recibe como entrada la información proveniente del bloque de coordinación; el control de posicionamiento del manipulador serial de 5GDL. A la salida se tienen las posiciones angulares de cada servomotor que se tienen que mandar a través de un "*Packet output*" para poder posicionar al manipulador correctamente.

En su interior se encuentra un bloque de función "Cinemática Man" el cual, como su nombre lo indica, está contenida y es calculada la cinemática del manipulador serial dando a la salida el posicionamiento de los servomotores. Dicha información pasa por un bloque "uint8" para convertir las unidades de la información y así poder ser enviada a través del "*Packet output*".

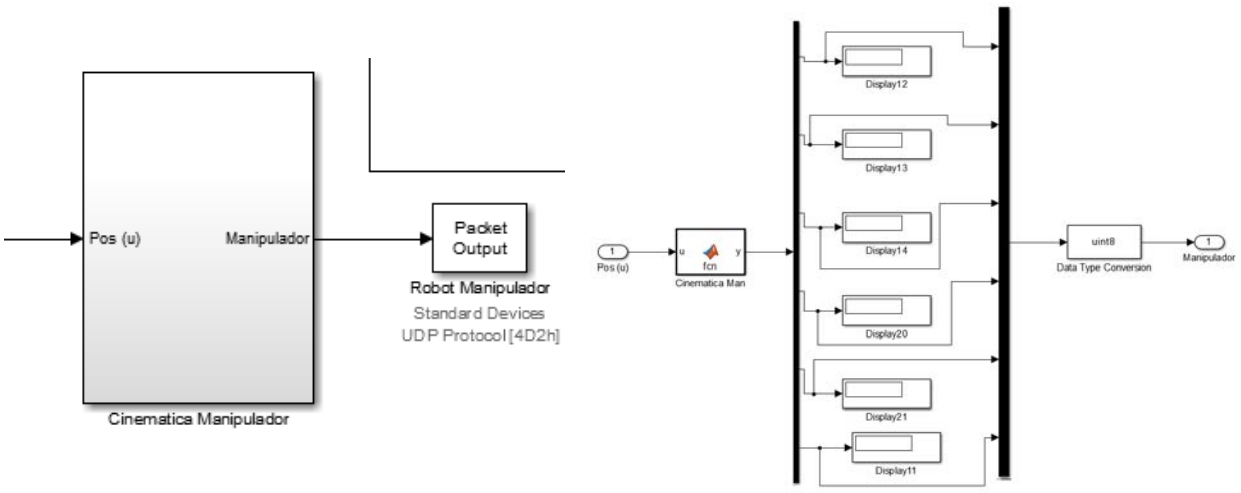


Figura 5.9 Bloque cinemático del manipulador

Al juntar e interconectar todos los bloques del sistema descritos, se obtiene finalmente el modelo del sistema de pruebas (Figura 5.10) con el que se estará trabajando para la realización de la tarea.

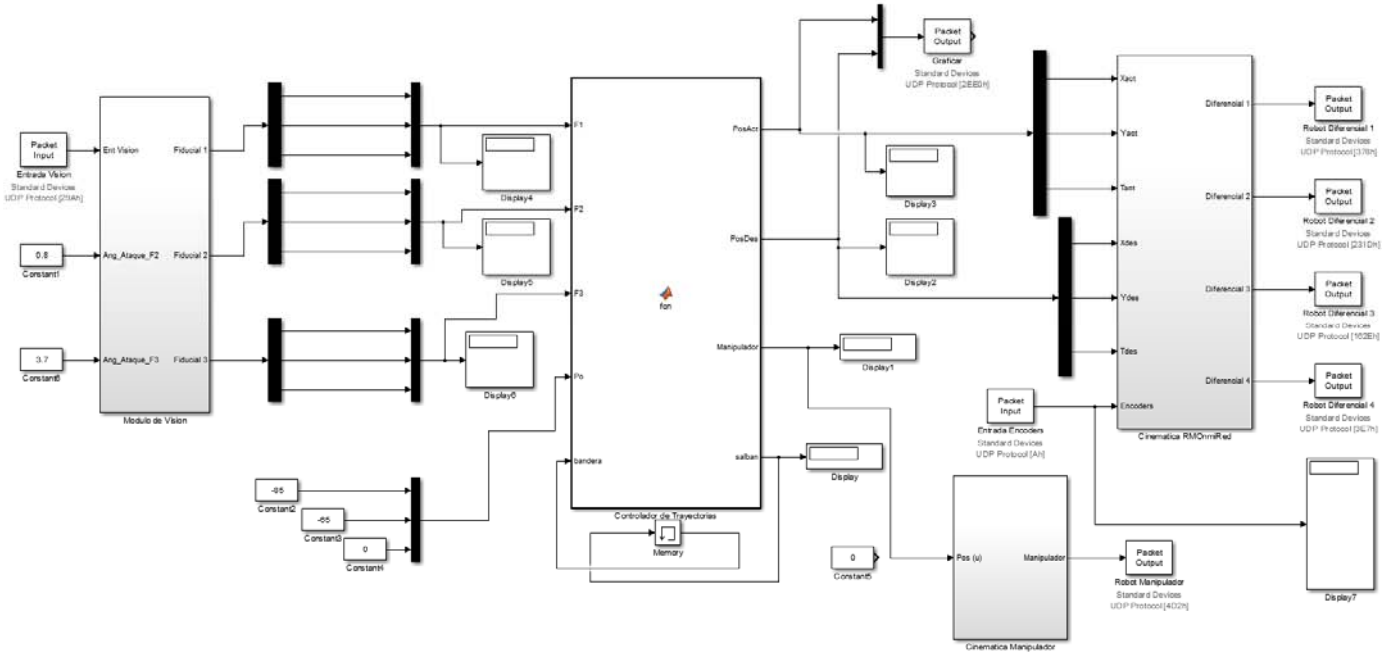


Figura 5.10 Modelo del Sistema de Pruebas en Simulink

5.4.3 MATLAB EN TIEMPO REAL

Un sistema se puede definir como un mapeo de un conjunto de elementos de entrada a un conjunto de elementos de salida. Al tiempo transcurrido entre que el sistema recibe la información de entrada, la transforma y pone a disposición el conjunto completo de salidas, se le denomina tiempo de respuesta del sistema. Por lo tanto un sistema en tiempo real es aquel que debe satisfacer una restricción de tiempo explícita, tiene un tiempo de respuesta limitado; o de lo contrario el sistema podría comportarse de forma errática e incluso producir fallos (Laplante, 2004).

Matlab trae incluido un módulo llamado Real Time Windows Target que provee de un motor de tiempo real que se ejecuta en modo kernel para establecer la conexión entre bloques de modelos de Simulink en una computadora corriendo Microsoft Windows y una gran variedad de tarjetas I/O, para poder interactuar con sensores, actuadores y otros dispositivos a desarrollar. Esto permite crear y controlar un Sistema en tiempo real para prototipos rápidos o simulación de hardware en ciclo.

Real Time Windows Target sincroniza los datos entre el motor en tiempo real y Simulink. Soporta dos modos de simulación:

- Modo normal, para una ejecución simple de tiempo real con acceso a dispositivos I/O ejecutándose en paralelo con un modelo de Simulink, alcanzando un rendimiento superior a 500 Hz.
- Modo externo, para tiempo real de alto rendimiento en donde el motor en tiempo real carga el archivo binario, resultante de la compilación, y los controladores de dispositivos de I/O, y establece una conexión con Simulink, alcanzando un rendimiento cercano a 20 kHz.

Durante la ejecución en tiempo real del modelo, el kernel interviene para dar al modelo prioridad para utilizar la CPU con la finalidad de ejecutar cada actualización del modelo en los tiempos de muestreo prescritos. Para garantizar un periodo de muestreo preciso el kernel reprograma el reloj de la computadora a una frecuencia mayor. Durante la ejecución de la aplicación en tiempo real almacena los datos en buffers, y posteriormente el contenido de los buffers es recuperado por Simulink para imprimirlas en pantalla.

Una aplicación en tiempo real tiene las siguientes características:

- Código compilado.
- Relación con el modelo de Simulink.
- Relación con el kernel.
- Cheksum.

Para poder configurar el programa de Simulink, en la barra de herramientas se tiene que entrar a los parámetros de configuración del modelo como se muestra a continuación:

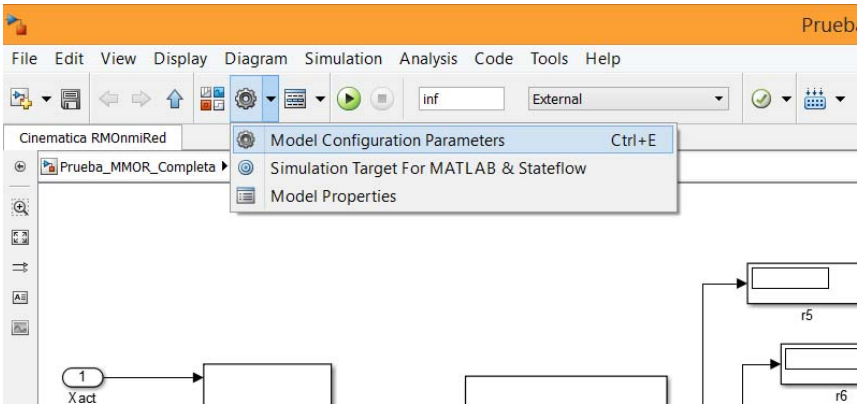


Figura 5.11 Parámetros de Configuración del Modelo

Esto nos abre una ventana nueva en la cual nos dirigimos a la sección de Generación de Código, y posteriormente seleccionamos el Real-Time Windows Target (rtwin.tlc) como nuestro archivo de destino del sistema. Finalmente, regresamos a la sección del Solucionador en donde se modificará la restricción del tiempo de muestreo con que se tiene que cumplir.

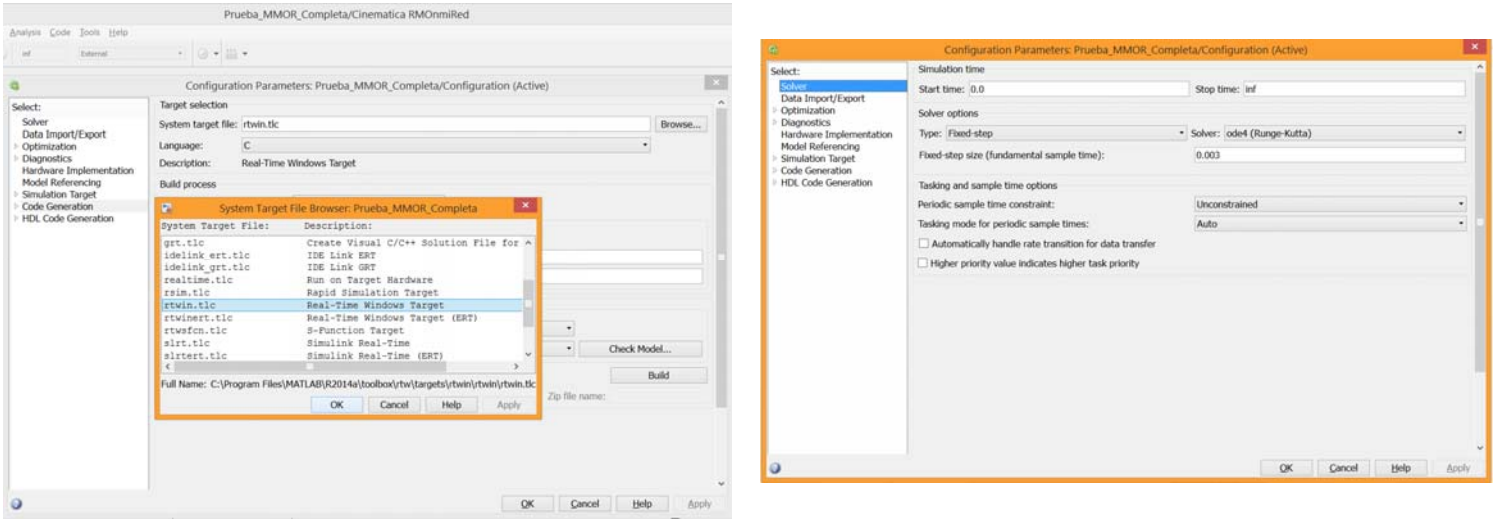


Figura 5.12 Configuración de Simulink en Tiempo Real

Nuestro sistema cuenta con una cámara web normal, cuya resolución de muestreo es normalmente de 30fps. Al relacionar la cámara con ReacTIVision se mostró que esta resolución de muestreo variaba entre 20 y 30fps, haciendo que el sistema de visión no cumpla con alguna restricción de tiempo haciendo que esta parte del sistema no funcione como un sistema en tiempo real.

De la misma manera, el microcontrolador Arduino no garantiza tampoco el cumplimiento de restricciones de tiempo ya que su método de programación, cuenta con muchas capas de código para la programación del microcontrolador en la cual se desconoce el tiempo mínimo requerido para ejecutarse.

Al presentar estas particularidades nuestro sistema, se garantiza que el programa que está corriendo en Simulink es en tiempo real, pero por otro lado, el resto de sistema no está en tiempo real.

5.4.4 COMUNICACIÓN WIFI – UDP

El protocolo de comunicación denominado UDP (por sus siglas en inglés, User Datagram Protocol) está basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red WiFi sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción.

Su uso principal es para protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos. Resulta más importante transmitir con velocidad que garantizar el hecho de que lleguen absolutamente todos los bytes.

UDP utiliza puertos para permitir la comunicación entre aplicaciones. El campo de puerto tiene una longitud de 16 bits, por lo que el rango de valores válidos va de 0 a 65.535. El puerto 0 está reservado, pero es un valor permitido como puerto origen si el proceso emisor no espera recibir mensajes como respuesta.

El WiFi *shield* de Arduino permite conectar tarjetas Arduino por medio de la comunicación UDP debido a que es compatible con el software de Matlab y poder así llevar a cabo una transmisión de datos en tiempo real.

Visión Artificial: ✓ Puerto local: 666 ✓ Dirección remota: 127.0.0.1 ✓ Puerto UDP remoto: 1000	Robot móvil diferencial 4: ✓ Puerto local: 999 ✓ Dirección remota: 192.168.0.104 ✓ Puerto UDP remoto: 15000
Robot móvil diferencial 1: ✓ Puerto local: 888 ✓ Dirección remota: 192.168.0.101 ✓ Puerto UDP remoto: 6780	Lectura de encoders: ✓ Puerto local: 10 ✓ Dirección remota: 192.168.0.105 ✓ Puerto UDP remoto: 56000
Robot móvil diferencial: ✓ Puerto local: 8989 ✓ Dirección remota: 192.168.0.102 ✓ Puerto UDP remoto: 555	Manipulador serial: ✓ Puerto local: 1234 ✓ Dirección remota: 192.168.0.107 ✓ Puerto UDP remoto: 13000
Robot móvil diferencial 3: ✓ Puerto local: 5678 ✓ Dirección remota: 192.168.0.103 ✓ Puerto UDP remoto: 12000	

Tabla 5.1 Puertos UDP del sistema de pruebas.

5.5 ENTORNO

El entorno físico en el que se realizará la tarea tiene como componentes al manipulador móvil omnidireccional redundante, las mesas objetivo y el objeto a manipular.

Tanto el manipulador móvil omnidireccional redundante como las mesas objetivo cuentan con marcadores especiales (*fiducials*) para poder ser reconocidos por el sistema de visión artificial para que éste pueda saber en dónde se encuentran éstos objetos y con qué orientación.

5.5.1 MANIPULADOR MOVIL OMNIDIRECCIONAL REDUNDANTE

El manipulador móvil omnidireccional redundante tiene como tarea, llevar a cabo el traslado desde una mesa objetivo a otra al objeto a manipular. Para ello, recibe los valores de velocidad de las llantas de cada robot móvil diferencial enviados por el sistema de cómputo y de la posición de cada uno de los servomotores del manipulador serial de 5GDL, a la vez envía los datos de los encoders magnéticos absolutos que miden la posición angular de cada robot móvil diferencial con respecto a la plataforma móvil.

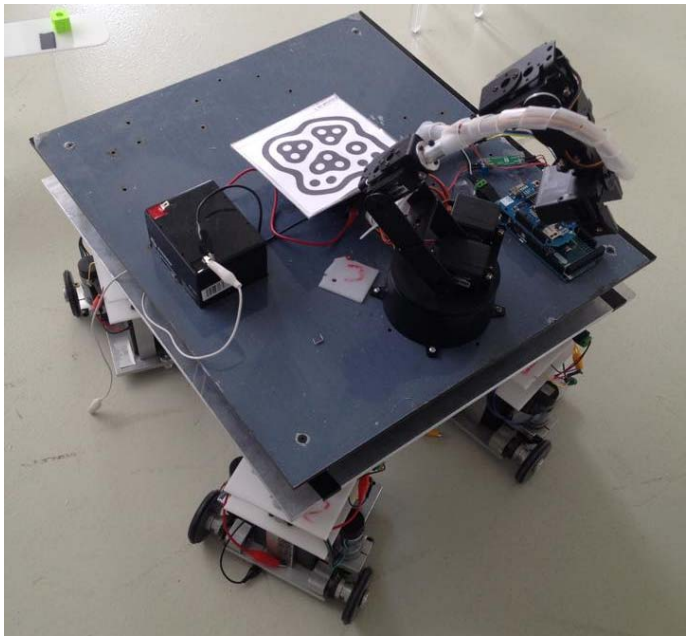


Figura 5.13 Manipulador Móvil Omnidireccional Redundante

La plataforma móvil es rectangular de 28 x 32.5 cm de lado, teniendo en cada esquina a un robot móvil diferencial acoplado, también se encuentran los encoders magnéticos absolutos que cuentan con un Arduino uno y de un WiFi *shield*, sobre la plataforma móvil se encuentra una placa de acrílico de las mismas medidas en donde se fija el manipulador móvil de 5GDL que cuenta con un Arduino Mega y un WiFi *shield*; finalmente para alimentar la electrónica se empela una batería de 12V. El marcador especial del manipulador móvil omnidireccional redundante tiene el ID 1.

5.5.2 MESAS OBJETIVO

Debido a la configuración del manipulador móvil omnidireccional redundante, el manipulador serial de 5GDL no es capaz de llevar a su actuador final (gripper) hasta el suelo para poder recoger un objeto. Para poder llevar a cabo satisfactoriamente la tarea del manipulador, se construyeron dos mesas objetivo (Figura 5.14). Tienen una altura de 22cm, y la superficie de la mesa se encuentra definida dentro de un área rectangular de 15.5 x 36cm.

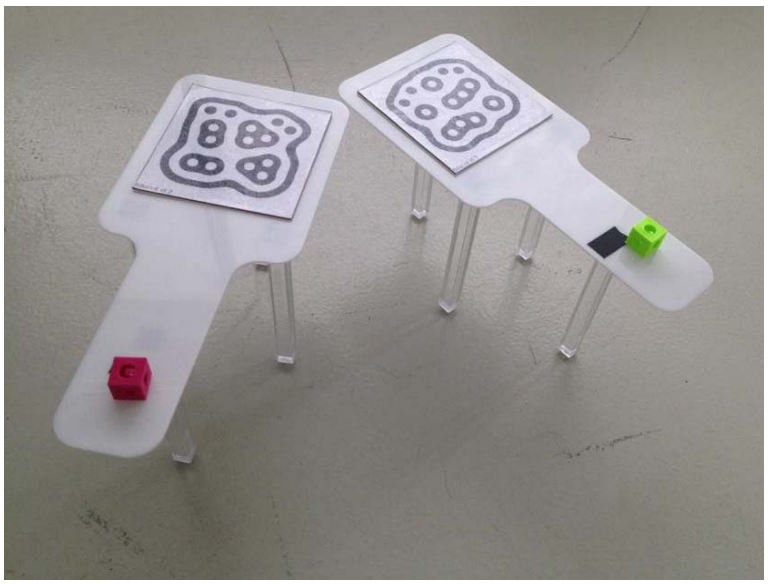


Figura 5.14 Mesas Objetivo

El objetivo de éstas, es poder colocar el objeto a manipular a una altura que se encuentre dentro del espacio de trabajo del manipulador y, a su vez, indicarle al sistema la localización del objeto a manipular en base a la localización y orientación de las mesas aplicando una transformación que muestren las coordenadas del objeto.

El objeto será colocado en el extremo contrario en el que se coloque el marcador especial a una distancia definida en base a las dimensiones del manipulador móvil omnidireccional redundante de tal forma que éste no colisione con las mesas objetivo.

Para el sistema de pruebas, la mesa que tenga el marcador especial con el ID 2 tendrá sobre ella el objeto a manipular; al inicio de la tarea, mientras que la mesa con el fiducial con ID 3 será el objetivo donde se tiene que colocar el objeto.

5.5.3 OBJETO A MANIPULAR

Se fabricó un objeto de plástico con el uso de una máquina de impresión 3D, tal que cumpliera con las siguientes características:

- Sea una forma geométrica fácil de reconocer.
- El tamaño de las aristas tiene que ser menor a la apertura máxima del gripper del manipulador serial que lo tomará.
- El manipulador que lo tomará, necesitará una orientación específica para agarrarlo correctamente.
- Sea de un color que pueda ser distinguido por la cámara del sistema de visión artificial.
- Sea hueco al centro para el ahorro de material.

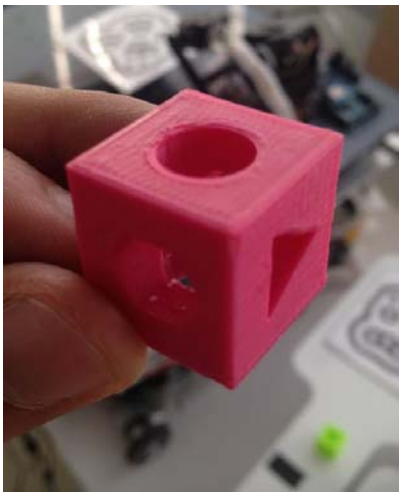


Figura 5.15 Objeto a Manipular

Como resultado, se decidió fabricar una forma cúbica de 2cm de lado. La forma elegida cumple con los criterios arriba mencionados además de que facilita al operador del sistema de pruebas a colocarlo nuevamente en su posición de origen de manera rápida y sin muchas complicaciones que otros prismas puedan tener. Se hicieron dos impresiones, una en color verde y otra en magenta, siendo la de color magenta el objeto a utilizar debido que el cubo de color verde no se apreció por parte del sistema de visión artificial.

CAPITULO 6 PRUEBAS Y RESULTADOS

6.1 PRUEBAS FÍSICAS

En este capítulo se muestran las pruebas que se realizaron en el sistema de pruebas, así como los resultados que se obtuvieron de ellas. Para ello se utilizó un control por campos potenciales para la base móvil y una coordinación basada en intuición artificial de los movimientos del brazo manipulador, dentro de un ambiente inteligente compuesto por dos objetos estáticos (mesas objetivo) y un agente controlable (manipulador móvil omnidireccional redundante); debidamente descritos en capítulos anteriores.

6.1.1 DESCRIPCIÓN DE LA TAREA

La tarea comenzará como se muestra en la Figura 6.1, colocando al manipulador móvil omnidireccional redundante en donde se encuentra el círculo naranja (coordenadas (-85,-65,0)), a la mesa objetivo uno en el círculo verde (coordenadas (-85,16,0.8)) y a la mesa objetivo dos en el rojo (coordenadas (64,-13,3.68)). Siendo el área total de trabajo de 220 x 170cm.

El manipulador móvil omnidireccional redundante se dirigirá a la mesa objetivo uno en búsqueda del objeto, lo tomará y cambiará su desplazamiento rumbo a la mesa objetivo dos donde colocará el objeto a manipular. Una vez colocado el objeto en la mesa objetivo dos, el manipulador móvil omnidireccional redundante se dirigirá a la posición de inicio finalizando así la tarea.

El llevar de nuevo al manipulador móvil omnidireccional redundante al mismo punto de partida, tiene como objetivo el poder reiniciar la tarea de manera rápida para efectuar más iteraciones sin tener que realizar muchos preparativos.

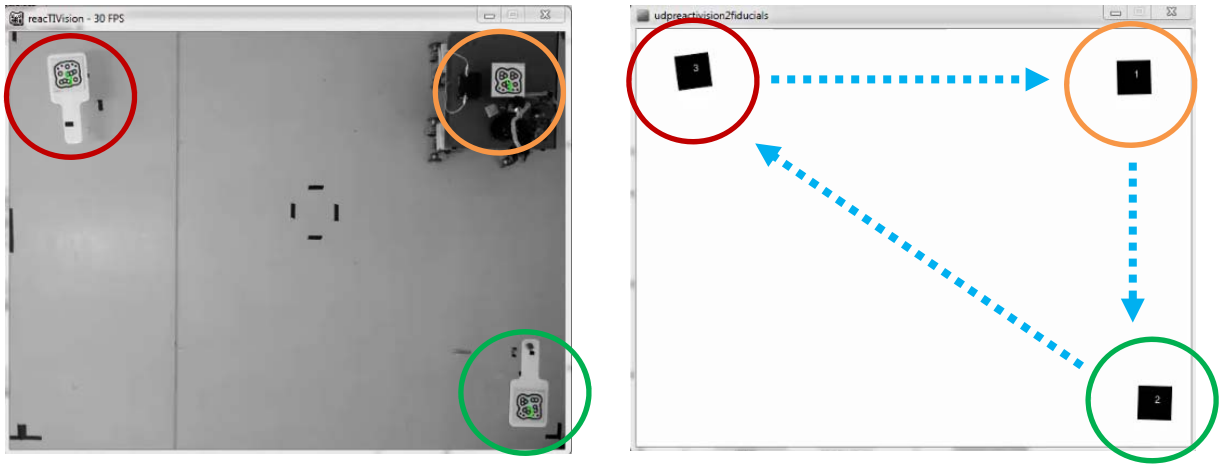


Figura 6.1 Estado inicial del sistema

De la tarea descrita, se realizaron varias iteraciones de las cuales se toman dos de ellas para su análisis. Las iteraciones seleccionadas fueron las que realizaron la tarea donde el tiempo y ruta fueron óptimos y la efectuaron sin errores.

6.1.2 GRÁFICAS DE DATOS OBTENIDOS

Se obtuvieron los datos a través de un bloque de Simulink de tipo "*Packet output*" el cual capturó en todo momento de la tarea, la posición de la base móvil omnidireccional redundante, la posición de las mesas objetivo y la coordinación del movimientos que tiene que realizarse por parte del manipulador serial.

Como ya se describió en el capítulo 4, para la coordinación de movimientos se discretizó el espacio geométrico del brazo manipulador y el algoritmo intuitivo de coordinación mandaría la orden para posicionar los servomotores del manipulador serial de 5GDL. Al no tener una retroalimentación en el brazo manipulador, se tomaron los datos capturados por la coordinación de movimientos y se calcularon las posiciones de los servomotores con ayuda de la cinemática para poder así graficar el movimiento efectuado durante la tarea.

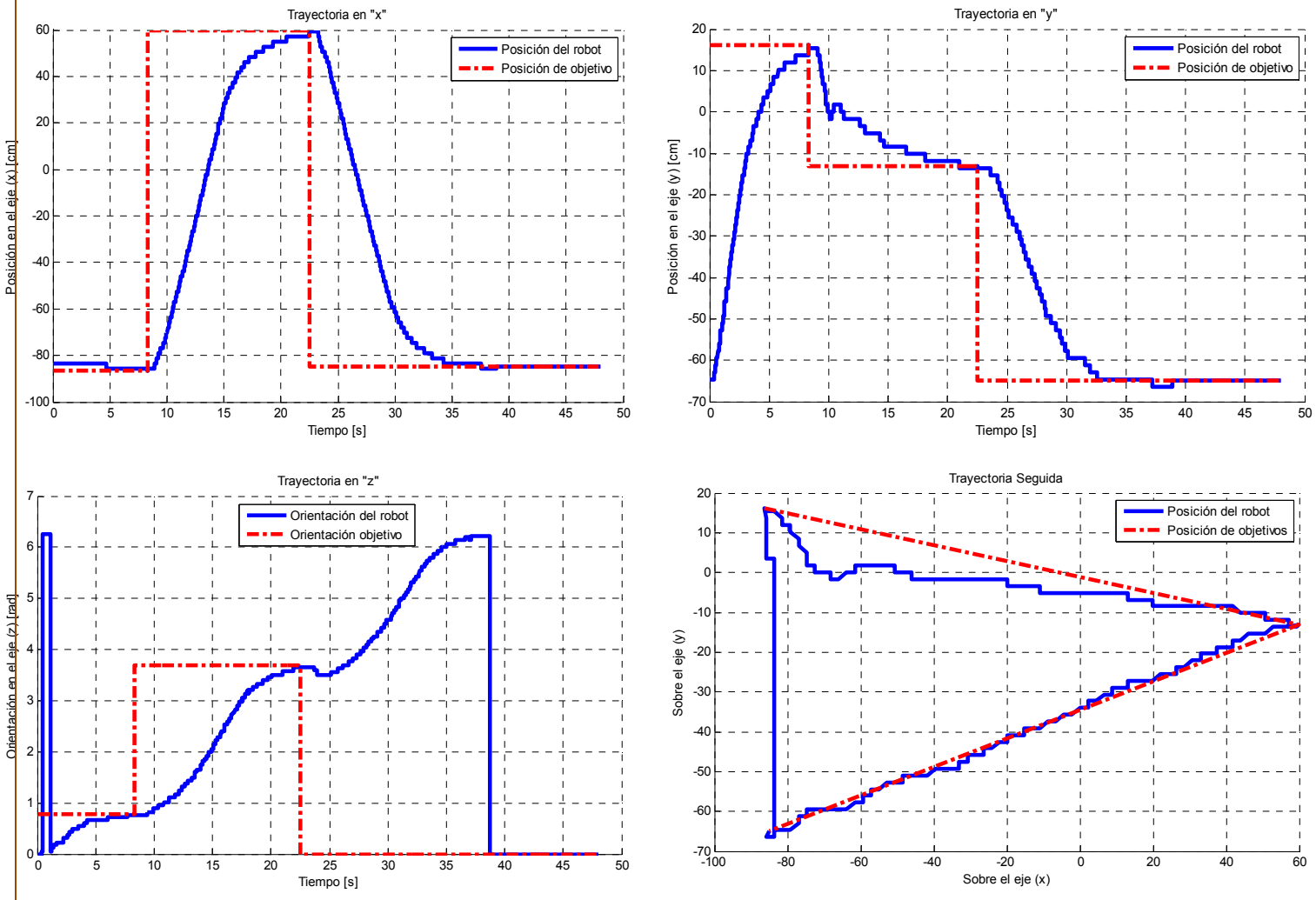


Figura 6.2 Resultados de posición de la prueba "A"

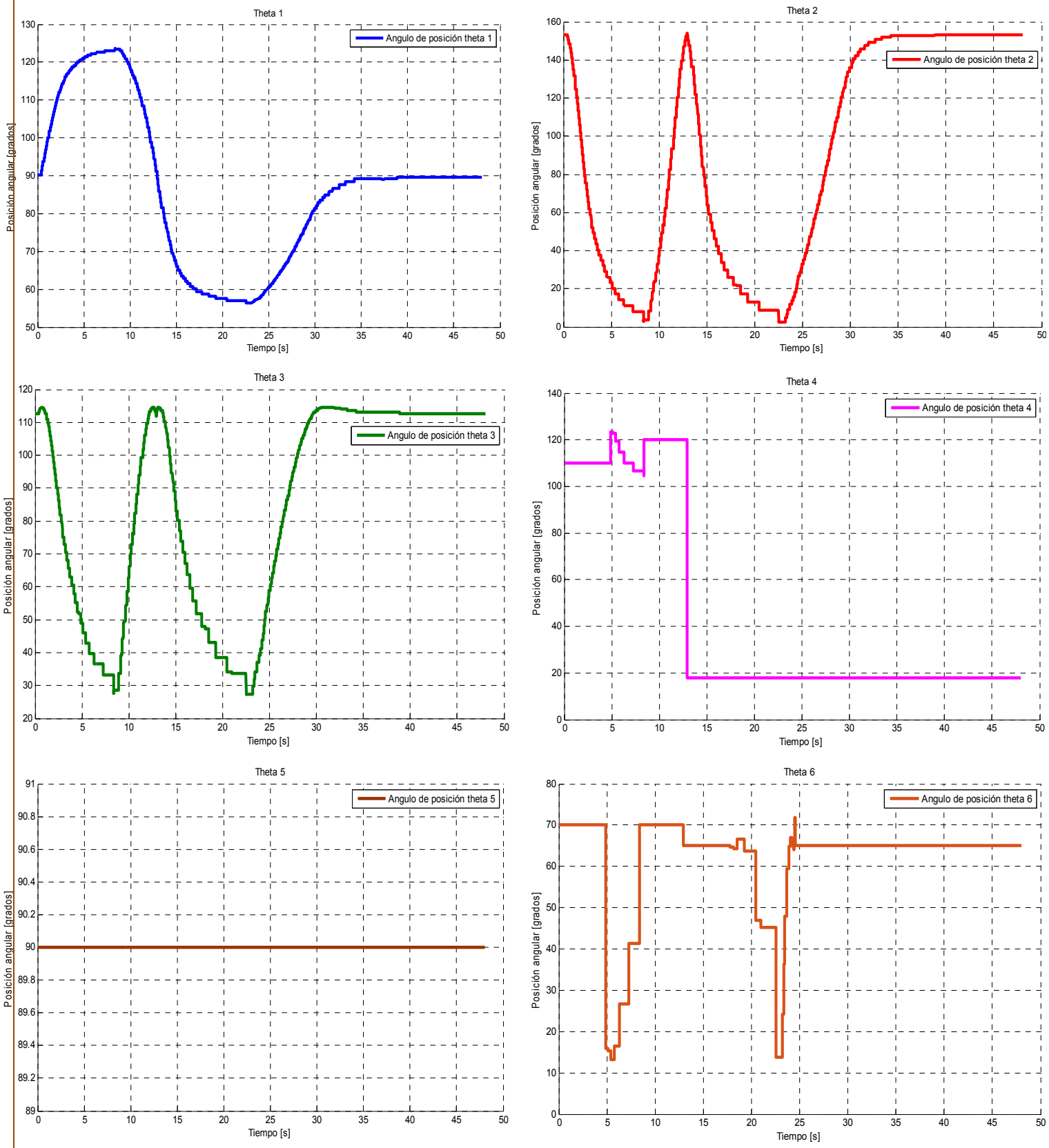


Figura 6.3 Resultados de manipulación de la prueba "A"

No está de más recordar que en el caso de la posición del manipulador serial de 5GDL no hay alguna referencia que nos diga cual debe de ser su posición ideal en cada momento en específico, ya que ésta se va calculando dependiendo del movimiento de la plataforma móvil.

A su vez, la plataforma móvil omnidireccional redundante calcula su ruta utilizando el control de campos potenciales, siendo también una ruta dinámica.

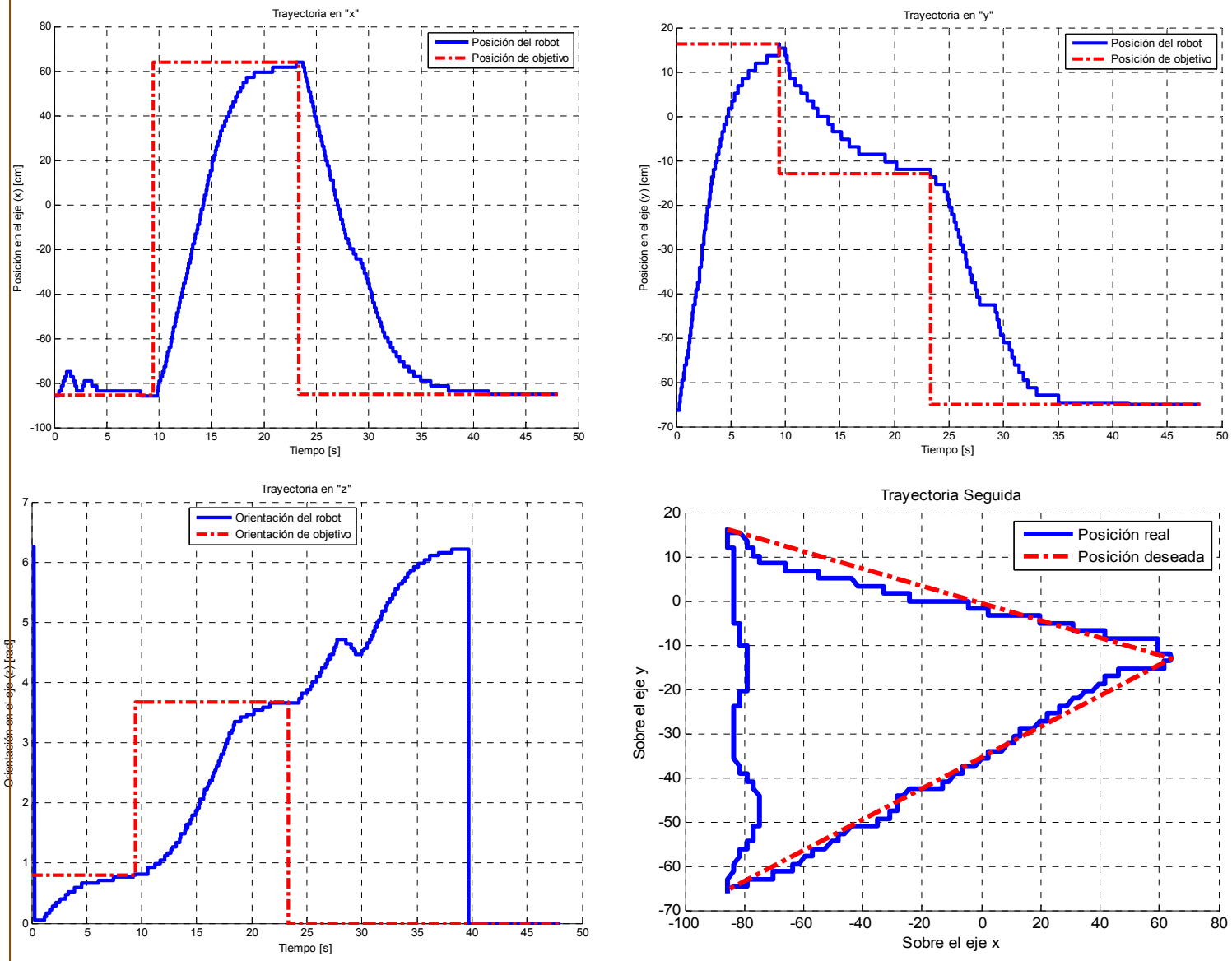


Figura 6.4 Resultados de la prueba "B"

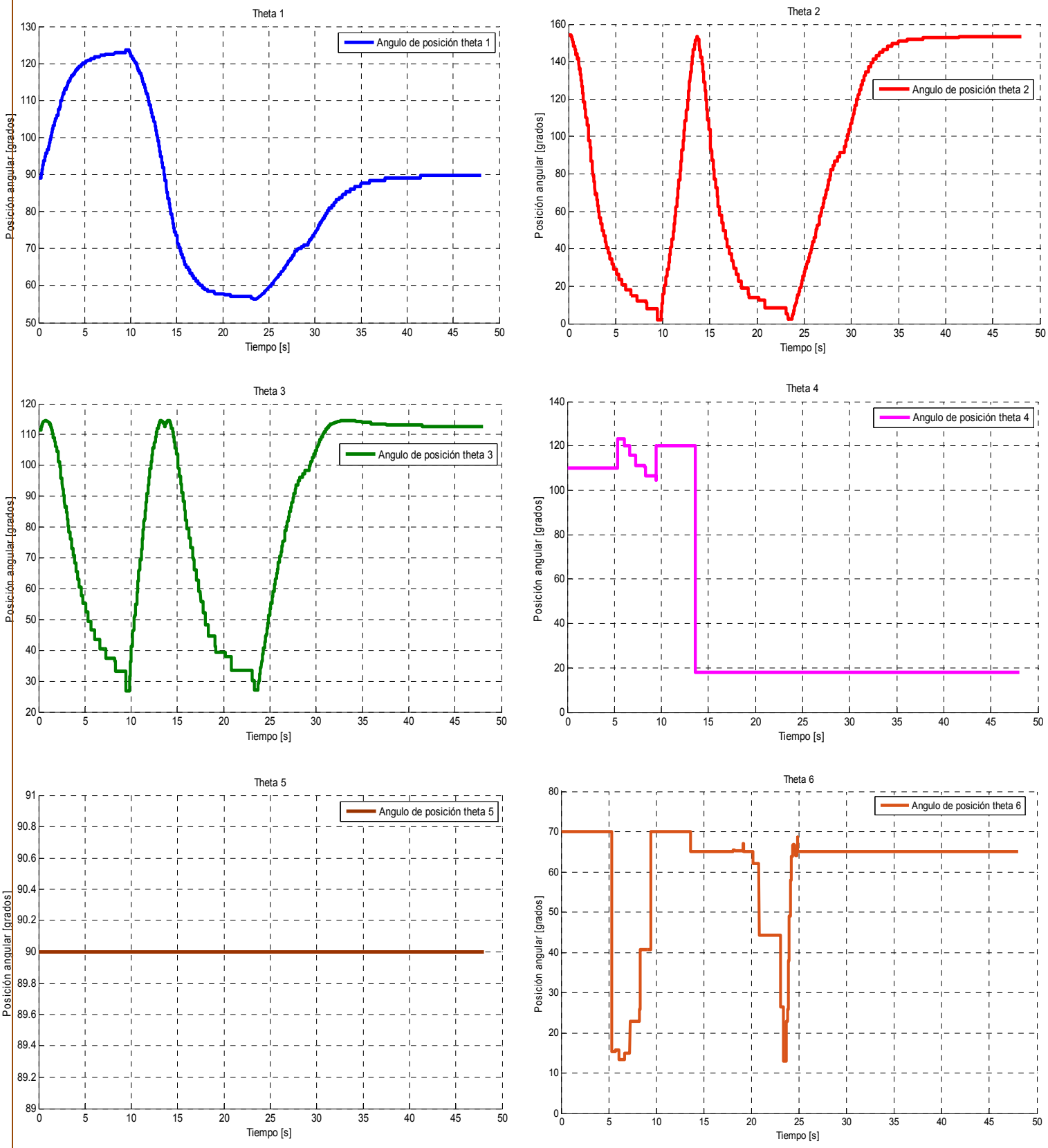


Figura 6.5 Resultados de manipulación de la prueba "B"

6.1.3 COMPARACIÓN DE RESULTADOS

Teniendo ambas pruebas, se procede a ser realizadas nuevas gráficas donde se pueda apreciar más a detalle las diferencias que éstas presentan. A continuación se muestran las gráficas:

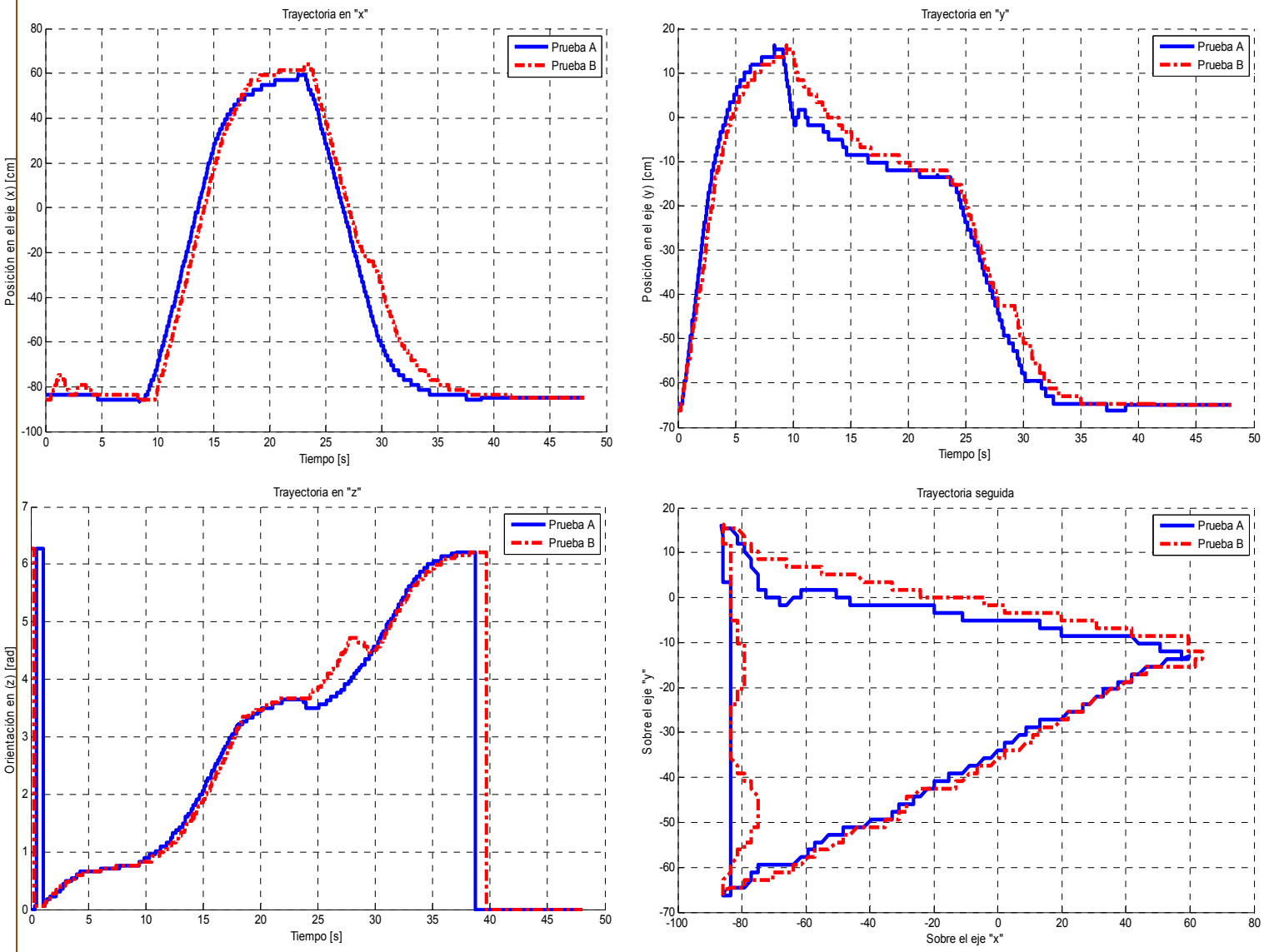


Figura 6.6 Comparación de coordenadas

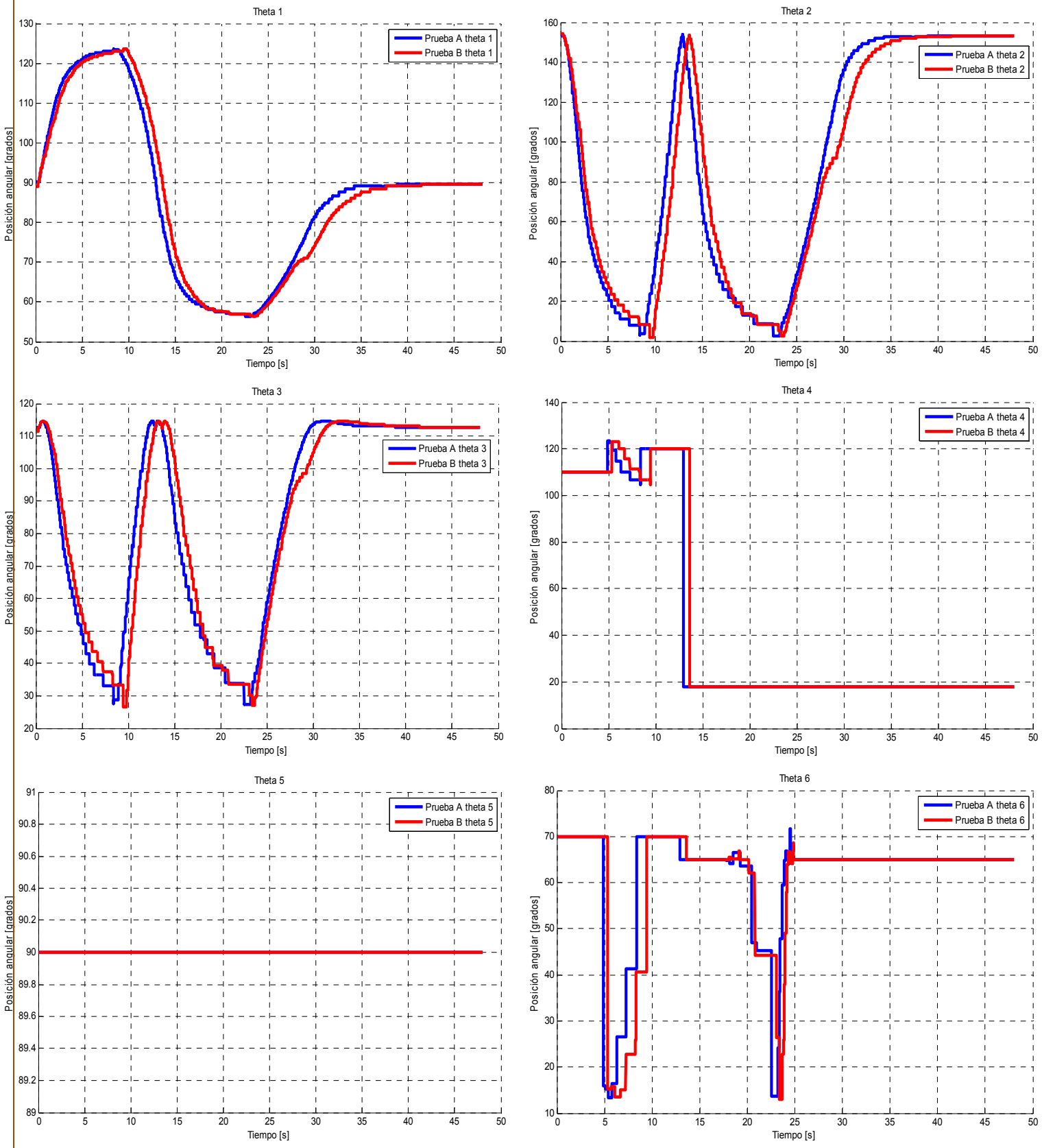


Figura 6.7 Comparación de posición angular

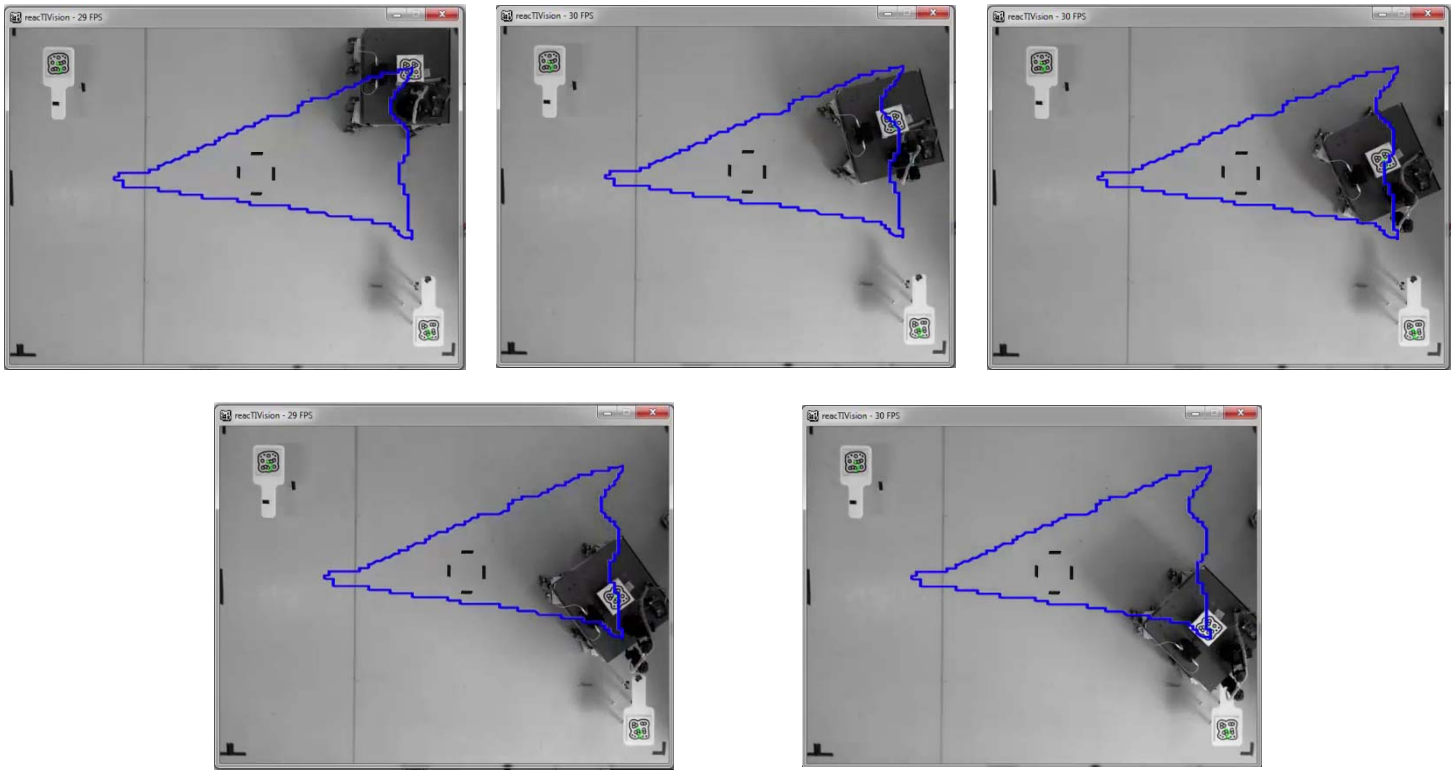


Figura 6.8 Prueba B, Etapa "Pick"

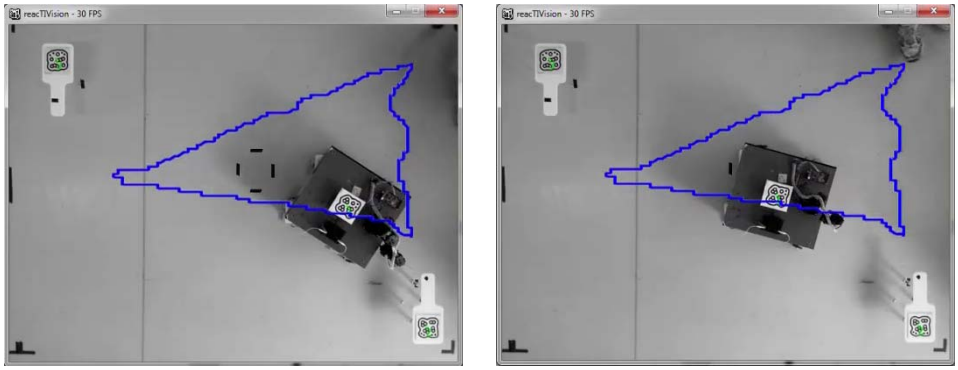


Figura 6.9 Prueba B, Etapa "Pick-Back"

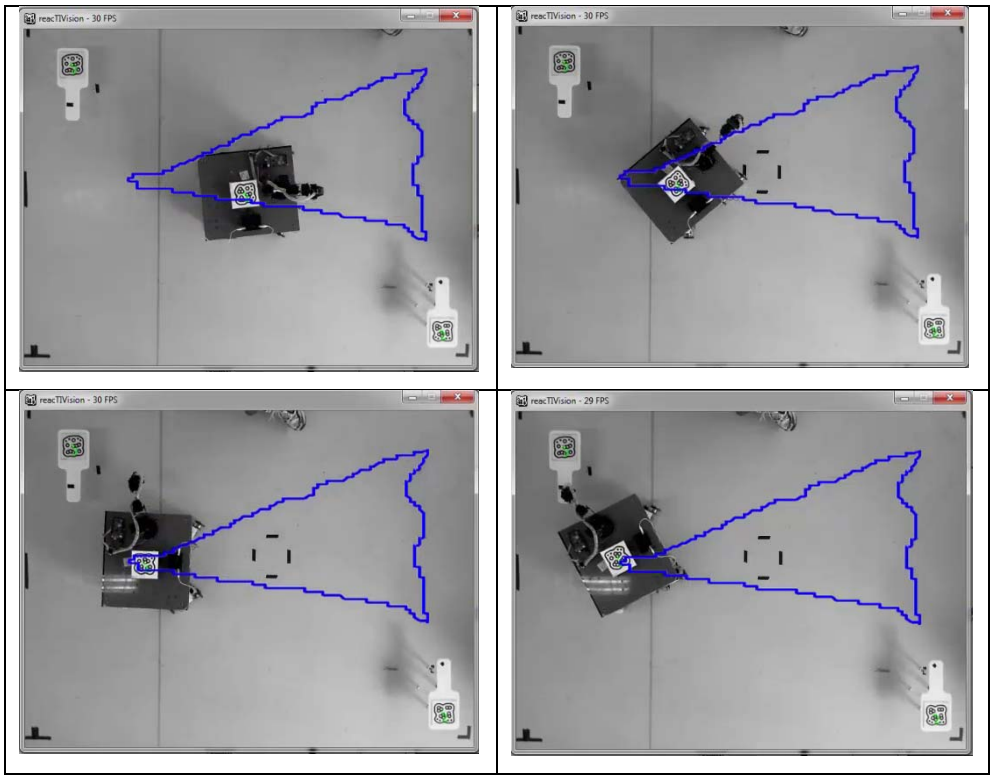


Figura 6.10 Prueba B, Etapa "Place"

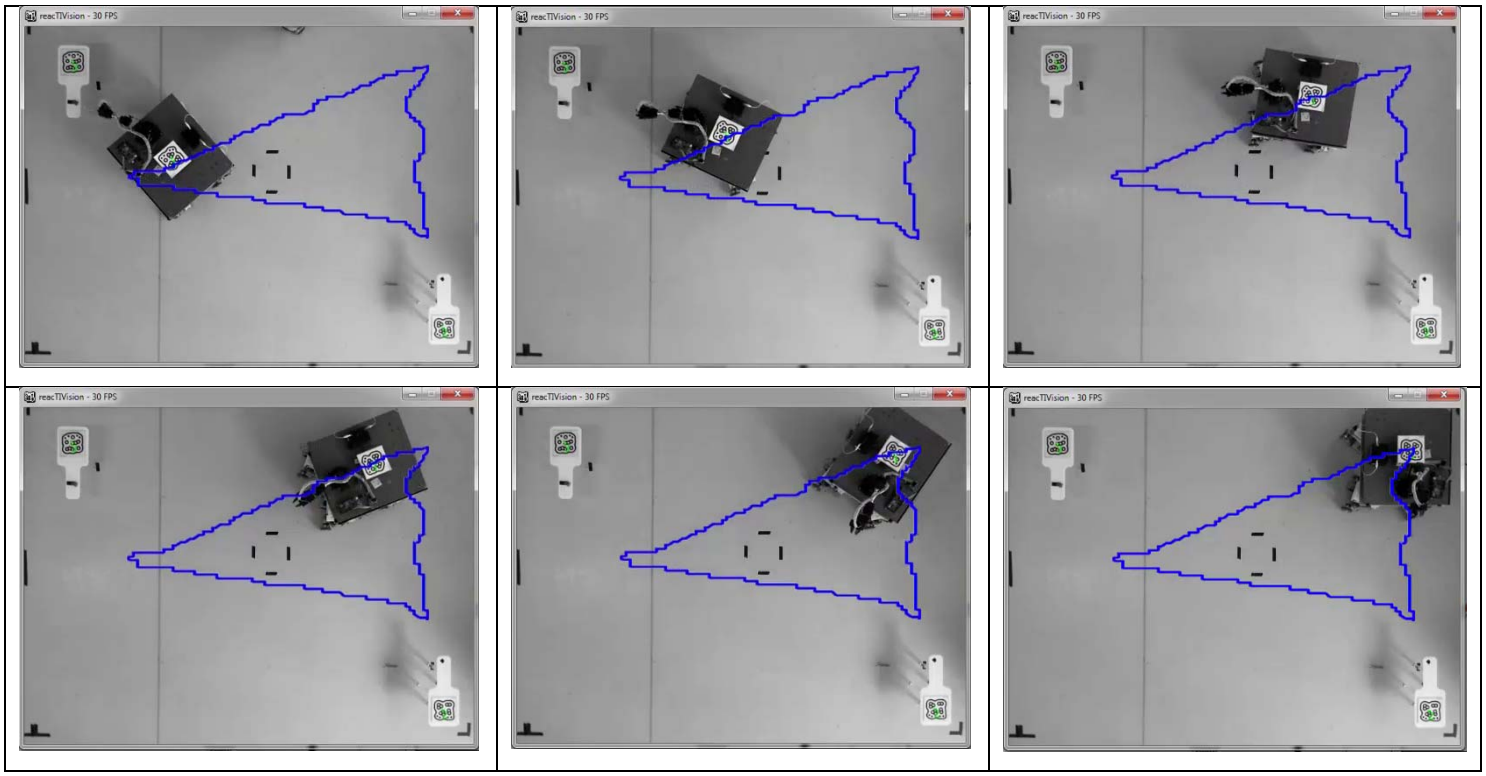


Figura 6.11 Prueba B, Etapa "Return"

CONCLUSIONES

Se llevó a cabo la construcción del prototipo piloto de forma que al ser unidos en serie un robot manipulador serial de 5GDL y un robot móvil omnidireccional redundante, se obtuvo un espacio de trabajo más amplio, proporcionando nuevos conocimientos en el área de los manipuladores móviles en el análisis cinemático de esta nueva configuración.

El robot móvil omnidireccional, al ser redundante y estar constituido por cuatro robots diferenciales en su base, le permite mantener una tracción más completa y sin deslizamientos en comparación a los robots móviles omnidireccionales con llantas especiales. Por ser redundante, se tienen múltiples soluciones para ir de un punto a otro, para ello se utiliza el control por campos potenciales el cual restringe el conjunto de soluciones calculando el camino más eficiente. Dado que el análisis cinemático realizado sobre la plataforma es suficiente para coordinar los robots diferenciales, el control por campos potenciales es ideal, ya que solamente con dicho análisis e introduciendo la posición y orientación deseada es suficiente para efectuar el movimiento.

Para el estudio de los movimientos del robot manipulador móvil omnidireccional redundante fue definida una tarea de llevar un objeto de un lugar a otro. Para la base móvil se definieron tres puntos objetivo en donde las trayectorias descritas en las pruebas fueron completamente calculadas por el manipulador móvil omnidireccional redundante, es decir que nunca fueron planeadas previamente. Para el brazo manipulador se definió el espacio geométrico por el que tiene que moverse; definiendo una secuencia de movimientos para la posición y una para la orientación del efector final.

La coordinación de movimientos basada en los principios de la intuición artificial, nos dio como resultado un movimiento continuo para la tarea, pudiendo observarse una ejecución de movimientos continuos y más cercanos a un movimiento natural. A su vez, el ambiente inteligente permitió obtener la información del sistema de pruebas; como son las posiciones y orientaciones tanto de las mesas objetivo como del manipulador móvil omnidireccional redundante, disminuyendo el procesamiento del robot móvil para identificar el espacio de trabajo en el que se encuentra inmerso.

La toma y traslado del objeto no fue siempre exitosa, debido a que no se cuenta con la identificación de la posición real del efector final del manipulador volviendo esta parte del sistema de lazo abierto. Al no tener retroalimentación del efector final, no se pueden realizar los ajustes necesarios en el mismo para compensar errores de posicionamiento.

Trabajo a futuro:

La tecnología implementada en el ambiente inteligente, no permite identificar la posición del efector final del manipulador; teniéndola se puede realizar un control a lazo cerrado disminuyendo significativamente los errores al momento de tomar y dejar el cubo en las mesas objetivo.

La definición de un espacio geométrico para el brazo manipulador que se asemeje más al de una persona para la toma y traslado de un objeto, realizando pruebas y capturando los movimientos que éste ente realizar para llevar a cabo la tarea.

REFERENCIAS

Arduino. (2015). Retrieved from http://www.arduino.cc/

Baede, T. A. (2006). *Motion control of an omnidirectional mobile robot.* Singapour: Eindhoven University of Technology.

Craig, J. J. (2003). *Robótica.* México: Pearson, Prentice Hall, 3a edición.

Cuellar, F. (n.d.). *Analysis and Design of a Wheeled Holonomic Omnidirectional Robot.* Perú.

D. H. Shin, B. H. (2003). *Motion Planning for a Mobile Manipulator with Imprecise Locomotion.* IROS, 2003; pp. 847-853. .

Díaz-Hernández, O. (2014). *Intuición Artificial Aplicada a la Teleoperación.* México: UNAM, Facultad de Ingeniería.

Escobedo-Castillo, P. M. (2012). *Manipulador Móvil: Estudio sobre la coordinación de movimientos de un manipulador serial acoplado.* México: UNAM, Facultad de Ingeniería.

González-Villela, V. J. (2004). *A wheeled mobile robot with obstacle avoidance capability.* Ingeniería Mecánica. Tecnología y Desarrollo, vol. 1, núm. 5, pp 159-166.

González-Villela, V. J. (2006). *A Unifying Theory on Conventional Wheeled Mobile Robots. Research on semiautonomous mobile robots for loosely structured environments focused on transporting mail trolleys.* Reino Unido.

González-Villela, V. J. (2013). *Omnidirectional mobile platform driven by two differetially driven wheeled mobile robots for moving heavy loads.* MEMORIAS DEL XIX CONGRESO INTERNACIONAL ANUAL DE LA SOMIM pp1064-1069.

H. G. Tanner, K. J. (2000). *Coordinating the Motion of a Manipulator and a Mobile Base in Tree Environments.* Osaka, Japón: Proceedings of the 2nd IFAC/CIGR International Workshop on Bio-Robotics, Information Technology and Intelligent Control for Bioproduction Systems.

Holmberg, R. (1999). *Development of a Holonomic Mobile Robot for Mobile Manipulation Tasks.* U.S.

I. C. Cruz-López, J. D.-M. (2013). *Robot Manipulador Paralelo Híbrido de 7GDL: Cinemática, Dinámica y Seguimiento de una Trayectoria.* México: UNAM, Facultad de Ingeniería.

Kazuyuki Morioka, J.-H. L. (2008). *Intelligent Space for human Centered Robotics.* Japan: Meiji University, Ritsumeikan University, University of Tokyo.

Laplante, P. A. (2004). *Real-Time Systems Design and Analysis.* Wiley.

M. Boukattaya, M. J. (2009). *Dynamic redudancy resolution for mobile manipulators using position fuzzy controller.* 6th International Multi-Conference on Systems, Signals and Devices, 2009, SSD ’09, pp 01-06.

Martínez-Carrillo, A. (2011). *Plataforma Móvil Omnidireccional a partir de dos robots móviles diferenciales.* México: UNAM, Facultad de Ingeniería.

Martinsanz, G. P. (2008). *Visión por computador imágenes digitales y aplicaciones.* México: Alfaomega.

MATLAB Documentation. (2015, Junio 1). Retrieved from MathWorks: http://es.mathworks.com/help/matlab/

MD25. (2015, Junio 1). Retrieved from Roboshop: http://www.robotshop.com/en/devantech-md25-dual-h-bridge-dc-motor-driver.html

O. Khatib, K. Y. (1996). *Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation.* pp546-553.

Pé, A. S. (2007). *Sistema de detección de objetos mediante cámaras. Aplicación en Espacios Inteligentes.* España: Universidad de Alcalá.

reacTIVision. (2015, Junio 1). Retrieved from http://reactivision.sourceforge.net/

S. Bhowmik, B. S. (2007). *Coordinated motion control of a mobile manipulator.* International Conference on Emerging Trends in Electrical Engineering, Kolkata, India, pp 124-131.

Simulink. (2015, Junio 1). Retrieved from MathWorks: http://es.mathworks.com/products/simulink/

T. W. Kang, Y. C. (2011). *Omnidirectional mobile robot using NIsbRIO9632xt.* Malasia: Universiti Teknologi Malaysia.

Tang Hong, Y. Q.-x. (2009). *Sliding mode control design of cleaning robot's mobile manipulator used in large condenser based on neuronal networks.* Proceedings of the 2009 IITA International Conference on Control, Automation and Systems Engineering, pp 446-449.

Udengaard, M. (2008). *Design of an Omnidirectional Mobile Robot for Rough Terrain.* U.S.

Wada, M. (n.d.). *Omnidirectional Holonomic Mobile Robot Using Nonholonomic Wheels.* Japón.

Weiming Ge, D. Y. (2008). *Sliding mode control for trajectory tracking on mobile manipulators.* IEEE Asia Pacific Conference on Circuits and Systems, 2008. APCCAS 2008, pp 1834-1837.

Y. Yamamoto, X. Y. (1994). *Coordinating locomotion and manipulation of a mobile manipulatior.* IEEE Trans. Automatic Contro, Vol 39, No. 6, pp.1326-1332.

Yamada, T. (2001). *Dynamic Model and Control for a Holonomic Omnidirectional Mobile Robot.* Japón.

APENDICES

PROGRAMAS DE MATLAB

Módulo de Visión

3fiducials28 de entrada

```
function [x1,y1,a1r,x2,y2,a2r,x3,y3,a3r,a1,a2,a3]= fcn(u)

aux=0;
e=zeros(28);
for i=1:1:28
    y=u(i);
    if y==77
        aux=i;
    else
    end
end

r=28-aux;

for j=1:1:r
    e(j)=u(aux+j);
end

for k=1:1:aux
    e(r+k)=u(k);
end

x1=(e(1)-48)*100+(e(2)-48)*10+(e(3)-48);
y1=(e(4)-48)*100+(e(5)-48)*10+(e(6)-48);
a1=(e(7)-48)*100+(e(8)-48)*10+(e(9)-48);
a1r=a1*3.1416/180;

x2=(e(10)-48)*100+(e(11)-48)*10+(e(12)-48);
y2=(e(13)-48)*100+(e(14)-48)*10+(e(15)-48);
a2=(e(16)-48)*100+(e(17)-48)*10+(e(18)-48);
a2r=a2*3.1416/180;

x3=(e(19)-48)*100+(e(20)-48)*10+(e(21)-48);
y3=(e(22)-48)*100+(e(23)-48)*10+(e(24)-48);
a3=(e(25)-48)*100+(e(26)-48)*10+(e(27)-48);
a3r=a3*3.1416/180;
```

Vision_3fid

```
function [xp,yp,tp,xp2,yp2,tp2,xp3,yp3,tp3]= fcn(u,u2,u3,thA1,thA2)

x     = u(1);
y     = u(2);
th    = u(3);
x2    = u2(1);
y2    = u2(2);
th2   = u2(3);
x3    = u3(1);
y3    = u3(2);
th3   = u3(3);

%Coordenadas del robot y del objetivo
xs   = (x/100)*(-220)+110;
ys   = (y/100)*(+170)-85;
xs2  = (x2/100)*(-220)+110;
ys2  = (y2/100)*(+170)-85;
xs3  = (x3/100)*(-220)+110;
ys3  = (y3/100)*(+170)-85;

%Ajuste de parámetros de visión (Transformación)
thT=0.178093;

xp = xs;
yp = ys;
tp = th;

xp2 = xs2+50.80354*sin(th2+thT);
yp2 = ys2-50.80354*cos(th2+thT);
tp2a = th2+thA1;

if tp2a>6.283
    if  thA1>3.1415
        tp2=tp2a-3.141592;
    else
        tp2=tp2a-(2*3.141592);
    end
else
    tp2=tp2a;
end

xp3 = xs3-24;
yp3 = ys3+55;
tp3a = th3+thA2;

if tp3a>6.283
    if  thA2>3.1415
        tp3=tp3a-3.141592;
    else
        tp3=tp3a-(2*3.141592);
    end
else
    tp3=tp3a;
end
```

Controlador de Trayectorias

```
function [PosAct,PosDes,Manipulador,salban] = fcn(F1,F2,F3,Po,bandera)
%#codegen

salban=bandera;
PosAct=F1;
PosDes=F2;
Manipulador = 0;


%Distancias Totales entre metas
L1=sqrt(power((F2(1)-Po(1)),2)+power((F2(2)-Po(2)),2)); %distancia de Po a F2
L2=sqrt(power((F3(1)-F2(1)),2)+power((F3(2)-F2(2)),2)); %distancia de F2 a F3
L3=sqrt(power((Po(1)-F3(1)),2)+power((Po(2)-F3(2)),2)); %distancia de F3 a Po


%%distancias del robot(F1) a las metas
L1d=sqrt(power((F2(1)-F1(1)),2)+power((F2(2)-F1(2)),2)); %meta F2
L2d=sqrt(power((F3(1)-F1(1)),2)+power((F3(2)-F1(2)),2)); %meta F3
L3d=sqrt(power((Po(1)-F1(1)),2)+power((Po(2)-F1(2)),2)); %meta Po


%Discretización (de 1 a 0)
Lt1=L1d/L1;
Lt2=L2d/L2;
Lt3=L3d/L3;


%Calculo de errores (de 0 a 1)
ue1=1-Lt1;
ue2=1-Lt2;
ue3=1-Lt3;


%Caluclo de U (necesaria para trayectoria deseada en manipulador)
U1=ue1*199;
U2=200+((ue2*2)*199);
U3=400+(((ue2-0.5)*2)*199);
U4=600-(ue3*199);


%Angulo deseado de plataforma
eAng1=sqrt(power((F2(3)-F1(3)),2));
eAng2=sqrt(power((F3(3)-F1(3)),2));
eAng3=sqrt(power((Po(3)-F1(3)),2));
```

```
if bandera==0
    PosDes=F2;
    Manipulador = U1;
    if L1d < 1.1 && eAng1 < 0.1
        PosAct=F2;
        salban=1;
    end
end



if bandera==1
    PosDes=F3;
    if ue2 < 0.5
        Manipulador = U2;
    else
        Manipulador = U3;
    End

    if L2d < 1 && eAng2 < 0.1
        PosAct=F3;
        salban=2;
    end
end



if bandera==2
    PosDes=Po;
    Manipulador = U4;
    if L3d < 0.9 && eAng3 < 0.1
        PosAct=Po;
        salban=3;
    end
end


if bandera==3
    PosAct=Po;
    PosDes=Po;
    Manipulador = 0;
    %salban=0;
end



end
```

Cinemática Manipulador

Cinematica Man

```
function y = fcn(u)

if u<200
    y1 = 0.0000010298*power(u,3)-
0.0008045680*power(u,2)+0.2882862394*u+89.9926123357;
    y2 = -0.0000001708*power(u,4)+0.0000789183*power(u,3)-
0.0115734416*power(u,2)-0.2337581307*u+153.5629845608;
    y3 = -0.0000000739*power(u,4)+0.0000346464*power(u,3)-
0.0073229767*power(u,2)+0.2413659558*u+112.4207944097;
    y5 = 90;
    if u<170
        y4 = 110;
        y6 = 15;
    else
        y4 = -0.0000436215*power(u,4)+0.0347848868*power(u,3)-
10.3385682795*power(u,2)+1357.3638320434*u-66309.8035615282;
        y6 = -0.0006139017*power(u,4)+0.4521241747*power(u,3)-
124.6505794176*power(u,2)+15248.7607203834*u-698423.5903937180;
    end

elseif u>200 && u<=400
    u2 = u - 200;
    y1 = -0.0000010298*power(u2,3)-0.0001866796*power(u2,2)-
0.0900367277*u2+123.7056531859;
    y2 = -0.0000001708*power(u2,4)+0.0000577554*power(u2,3)-
0.0052245667*power(u2,2)+0.8598857647*u2+1.8727625735;
    y3 = -0.0000000739*power(u2,4)+0.0000244590*power(u2,3)-
0.0042667802*power(u2,2)+0.8944770131*u2+26.7350455199;
    y4 = 120;
    y5 = 90;
    y6 = 70;

else
    u1 = u - 400;
    y1 = -0.0000010298*power(u1,3)+0.0008045680*power(u1,2)-
0.2882862394*u1+89.9926123357;
    y2 = -0.0000001708*power(u1,4)+0.0000789183*power(u1,3)-
0.0115734416*power(u1,2)-0.2337581307*u1+153.5629845608;
    y3 = -0.0000000739*power(u1,4)+0.0000346464*power(u1,3)-
0.0073229767*power(u1,2)+0.2413659558*u1+112.4207944097;
    y4 = 18;
    y5 = 90;
    if u1<170
       y6 = 65;
    else
       y6 = 0.0005706934*power(u1,4)-
0.4255357425*power(u1,3)+118.7507632308*power(u1,2)-
14700.8841651713*u1+681334.3318258550;
    end
end

y=[y1,y2,y3,y4,y5,y6];
```

Cinematica RMOmniRed

controlador.c

```
#define S_FUNCTION_NAME  controlador
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "math.h"
/*================*
 * Build checking *
 *================*/


/* Function: mdlInitializeSizes
=============================================
 * Abstract:
 *   Setup sizes of the various vectors. */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; } /* Parameter mismatch will be reported by Simulink *

    if (!ssSetNumInputPorts(S, 6)) return;
    ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 1, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 2, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 3, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 4, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 5, DYNAMICALLY_SIZED);

    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S,3)) return;
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 1, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 2, DYNAMICALLY_SIZED);

    ssSetNumSampleTimes(S, 1);

    /* specify the sim state compliance to be same as a built-in block */
    ssSetSimStateCompliance(S, USE_DEFAULT_SIM_STATE);

    ssSetOptions(S,
                 SS_OPTION_WORKS_WITH_CODE_REUSE |
                 SS_OPTION_EXCEPTION_FREE_CODE |
                 SS_OPTION_USE_TLC_WITH_ACCELERATOR);
}

/* Function: mdlInitializeSampleTimes
=======================================
 * Abstract:
 *    Specifiy that we inherit our sample time from the driving block. */
```

```
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

/* Function: mdlOutputs
=======================================================
 * Abstract:
 *    */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    float            i,dg,drep,dres,tg,trep,te;
    float            rho01,rhoc1,rhor1,n1;
    float            x01,y01;
    float            vxpp,vypp,wpp;
    float            vxmax,vymax,wmax;
    float            xp,yp,tP;
    float            a,b,c,d,r;
    float            xd,yd,td,ep,kr,k_OWR;

    //variables de entrada
    //Posición y orientación del robot móvil diferencial
    InputRealPtrsType  uPtrs0 = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType  uPtrs1 = ssGetInputPortRealSignalPtrs(S,1);
    InputRealPtrsType  uPtrs2 = ssGetInputPortRealSignalPtrs(S,2);
    //Posición deseada para el robot móvil diferencial
    InputRealPtrsType  uPtrs3 = ssGetInputPortRealSignalPtrs(S,3);
    InputRealPtrsType  uPtrs4 = ssGetInputPortRealSignalPtrs(S,4);
    InputRealPtrsType  uPtrs5 = ssGetInputPortRealSignalPtrs(S,5);
    //Variables de salida
    real_T            *vxp    = ssGetOutputPortRealSignal(S,0);
    real_T            *vyp    = ssGetOutputPortRealSignal(S,1);
    real_T            *wp     = ssGetOutputPortRealSignal(S,2);
    //Velocidad máxima permisible cm/s
    vxmax = 20;
    vymax = 20;
    wmax  = 0.4;
    //Posición y orientación del robot móvil diferencial
    xp   = *uPtrs0[0];
    yp   = *uPtrs1[0];
    tP   = *uPtrs2[0];
    //Posición deseada
    xd   = *uPtrs3[0];
    yd   = *uPtrs4[0];
    td   = *uPtrs5[0];
    //Obstáculos
    x01= 5000;
    y01= 5000;
    rho01=20;
    rhor1=4;
    n1=100;
    //Variables de control cm
    kr    = 50;
    k_OWR = 1;
```

```
    //Cálculo de las velocidades de entrada
    dg    = sqrt(pow((xd-xp),2) + pow((yd-yp),2));
    rhoc1 = sqrt(pow((x01-xp),2) + pow((y01-yp),2));

    if (yd==yp && xd==xp){
        tg = tP;  }
    else {
        tg = atan2((yd-yp),(xd-xp));
    }

    if (rhoc1 > rho01+rhor1){
        drep=0;
        trep=2*tg;  }
    else {
        drep= (n1/2)*pow(( (1/(rhoc1-rhor1)) - (1/rho01) ),2)  ;
        trep= atan2((y01-yp),(x01-xp));  }

    dres=drep+dg;
    te = (trep-tg) - tP;

    if (dres <= k_OWR){
        vxpp = 0;
        vypp = 0;   }
    else if (dres <= k_OWR + kr) {
        vxpp = vxmax*dres*cos(te)/(kr + k_OWR);
        vypp = vymax*dres*sin(te)/(kr + k_OWR);   }
    else {
        vxpp = vxmax*cos(te);
        vypp = vymax*sin(te);
    }

    ep = td - tP;

    wpp=wmax*sin(ep);

   *vxp =vxpp;
   *vyp =vypp;
   *wp  =wpp;
}

/* Function: mdlTerminate
=============================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.  */
static void mdlTerminate(SimStruct *S){
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

cinematicaSX.c

```
#define S_FUNCTION_NAME  cinematicaSX
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "math.h"
/*================*
 * Build checking *
 *================*/


/* Function: mdlInitializeSizes
=============================================
 * Abstract:
 *   Setup sizes of the various vectors.
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    if (!ssSetNumInputPorts(S, 7)) return;
    ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 1, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 2, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 3, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 4, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 5, DYNAMICALLY_SIZED);
    ssSetInputPortWidth(S, 6, DYNAMICALLY_SIZED);

    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S,8)) return;
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 1, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 2, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 3, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 4, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 5, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 6, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 7, DYNAMICALLY_SIZED);

    ssSetNumSampleTimes(S, 1);

    /* specify the sim state compliance to be same as a built-in block */
    ssSetSimStateCompliance(S, USE_DEFAULT_SIM_STATE);

    ssSetOptions(S,
                 SS_OPTION_WORKS_WITH_CODE_REUSE |
                 SS_OPTION_EXCEPTION_FREE_CODE |
                 SS_OPTION_USE_TLC_WITH_ACCELERATOR);
}
```

```
/* Function: mdlInitializeSampleTimes
=====================================
 * Abstract:
 *    Specifiy that we inherit our sample time from the driving block.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

/* Function: mdlOutputs
=====================================================
 * Abstract:
 *    y = 2*u
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{

    float            i;
    float            vxpp,vypp,wpp;
    float            vxmax,vymax,wmax;
    float            tP,t1,t2,t3,t4;
    float            a1,a2,b,c,d,r;
    //Variables de velocidad de entrada
    InputRealPtrsType  uPtrs0 = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType  uPtrs1 = ssGetInputPortRealSignalPtrs(S,1);
    InputRealPtrsType  uPtrs2 = ssGetInputPortRealSignalPtrs(S,2);
    //Variables de retroalimentación angular
    InputRealPtrsType  uPtrs3 = ssGetInputPortRealSignalPtrs(S,3);
    InputRealPtrsType  uPtrs4 = ssGetInputPortRealSignalPtrs(S,4);
    InputRealPtrsType  uPtrs5 = ssGetInputPortRealSignalPtrs(S,5);
    InputRealPtrsType  uPtrs6 = ssGetInputPortRealSignalPtrs(S,6);


    //Variables de salida
    real_T            *phi11  = ssGetOutputPortRealSignal(S,0);
    real_T            *phi12  = ssGetOutputPortRealSignal(S,1);
    real_T            *phi21  = ssGetOutputPortRealSignal(S,2);
    real_T            *phi22  = ssGetOutputPortRealSignal(S,3);
    real_T            *phi31  = ssGetOutputPortRealSignal(S,4);
    real_T            *phi32  = ssGetOutputPortRealSignal(S,5);
    real_T            *phi41  = ssGetOutputPortRealSignal(S,6);
    real_T            *phi42  = ssGetOutputPortRealSignal(S,7);

    //Variables de velocidad de entrada
    vxpp = *uPtrs0[0];
    vypp = *uPtrs1[0];
    wpp  = *uPtrs2[0];
    //Variables de retroalimentación angular
    t1  = *uPtrs3[0]*3.1415/180;
    t2  = *uPtrs4[0]*3.1415/180;
    t3  = *uPtrs5[0]*3.1415/180;
    t4  = *uPtrs6[0]*3.1415/180;
```

```
    //Dimensiones de la plataforma
    a1  = 32.5/2;
    a2  = 32.5/2;
    b   = 14;
    c   = 5;
    d   = 9;
    r   = 3;
    //_____________________Cálculo de velocidades_____________________


   *phi11 = vxpp*(-c*sin(t1)+d*cos(t1))/(c*r) +
vypp*(c*cos(t1)+d*sin(t1))/(c*r) + wpp*(b*c*sin(t1) -a1*d*sin(t1) -
a1*c*cos(t1) -b*d*cos(t1))/(c*r);
   *phi12 = vxpp*(-c*sin(t1)-d*cos(t1))/(c*r) + vypp*(c*cos(t1)-
d*sin(t1))/(c*r) + wpp*(b*c*sin(t1) +a1*d*sin(t1) -a1*c*cos(t1)
+b*d*cos(t1))/(c*r);
   *phi21 = vxpp*(-c*sin(t2)+d*cos(t2))/(c*r) +
vypp*(c*cos(t2)+d*sin(t2))/(c*r) + wpp*(b*c*sin(t2) +a2*d*sin(t2)
+a2*c*cos(t2) -b*d*cos(t2))/(c*r);
   *phi22 = vxpp*(-c*sin(t2)-d*cos(t2))/(c*r) + vypp*(c*cos(t2)-
d*sin(t2))/(c*r) + wpp*(b*c*sin(t2) -a2*d*sin(t2) +a2*c*cos(t2)
+b*d*cos(t2))/(c*r);

   *phi31 = vxpp*(-c*sin(t3)+d*cos(t3))/(c*r) +
vypp*(c*cos(t3)+d*sin(t3))/(c*r) + wpp*( -b*c*sin(t3) -a2*d*sin(t3) -
a2*c*cos(t3) +b*d*cos(t3))/(c*r);
   *phi32 = vxpp*(-c*sin(t3)-d*cos(t3))/(c*r) + vypp*(c*cos(t3)-
d*sin(t3))/(c*r) + wpp*( -b*c*sin(t3) +a2*d*sin(t3) -a2*c*cos(t3) -
b*d*cos(t3))/(c*r);
   *phi41 = vxpp*(-c*sin(t4)+d*cos(t4))/(c*r) +
vypp*(c*cos(t4)+d*sin(t4))/(c*r) + wpp*( -b*c*sin(t4) +a1*d*sin(t4)
+a1*c*cos(t4) +b*d*cos(t4))/(c*r);
   *phi42 = vxpp*(-c*sin(t4)-d*cos(t4))/(c*r) + vypp*(c*cos(t4)-
d*sin(t4))/(c*r) + wpp*( -b*c*sin(t4) -a1*d*sin(t4) +a1*c*cos(t4) -
b*d*cos(t4))/(c*r);

}


/* Function: mdlTerminate
=================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Speed2MD25

```
#define S_FUNCTION_NAME  Speed2MD25
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

/*================*
 * Build checking *
 *================*/


/* Function: mdlInitializeSizes
=============================================
 * Abstract:
 *   Setup sizes of the various vectors.
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 8);


    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, 8);


    ssSetNumSampleTimes(S, 1);

    /* specify the sim state compliance to be same as a built-in block */
    ssSetSimStateCompliance(S, USE_DEFAULT_SIM_STATE);

    ssSetOptions(S,
                 SS_OPTION_WORKS_WITH_CODE_REUSE |
                 SS_OPTION_EXCEPTION_FREE_CODE |
                 SS_OPTION_USE_TLC_WITH_ACCELERATOR);
}


/* Function: mdlInitializeSampleTimes
=======================================
 * Abstract:
 *    Specifiy that we inherit our sample time from the driving block.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
```

```
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

/* Function: mdlOutputs
=======================================================
 * Abstract:
 *    */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T             i;
    InputRealPtrsType uPtrs0 = ssGetInputPortRealSignalPtrs(S,0);
    real_T            *y0    = ssGetOutputPortRealSignal(S,0);
    int_T             width = ssGetInputPortWidth(S,0);
    int_T             j;

    for (i=0; i<width; i++) {
        /*
         * This example does not implement complex signal handling.
         * To find out see an example about how to handle complex signal in
         * S-function, see sdotproduct.c for details.
         */
         //Modifica la velocidad de entrada para acoplarla a los parámetros
de a tarjeta Md25
         // el valor 0 es la máxima velocidad de un lado y el valor 255 del
otro siendo 128 el valor medio
         j= 6.1127*(*uPtrs0[i]*(pow(-1,i)))+128;
         //Si el valor sobre pasa 255 (8bits y valor límite que se manda a la
tarjeta) o es menor a 0, los coloca en 255 y 0 respectivamente.
         if (j<0){
             *(y0+i) =0;
         }
         else if (j>255){
             *(y0+i) =255;
         }
         else *(y0+i) =j;

    }
}


/* Function: mdlTerminate
=====================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S)
{
}


#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

PROGRAMAS DE ARDUINO

udpR1.ino

```
#include <SPI.h>
#include <Wire.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#define i2cAddress 0x58
#define speed1 0x00                    // speed to first motor
#define speed2 0x01

/*   the IP address for the shield:   IPAddress ip(192, 168, 0, 110);   */
char ssid[] = "RobotMovil";     //  your network SSID (name)
char pass[] = "holamundo";    // your network password (use for WPA, or use as key for WEP)

int keyIndex = 0;           // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
int a,b;
int localPort =6780;      // local port to listen on
char packetBuffer[255]; //buffer to hold incoming packet

WiFiUDP Udp;
void setup() {
  Wire.begin();
  //Initialize serial and wait for port to open:
  //Serial.begin(9600);

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    //Serial.println("WiFi shield not present");
    // don't continue:
    while(true);
  }
//  WiFi.config(ip); // ip forzada
  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    //Serial.print("Attempting to connect to SSID: ");
    //Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  // start the server:
  // you're connected now, so print out the status:
  //printWifiStatus();

  Udp.begin(localPort);
 }
```

```
void loop() {
  //Serial.print("Entre al loop ");
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if(packetSize)
  {
    int len = Udp.read(packetBuffer,255);
    if (len >0) {
      packetBuffer[len]=0;
      a=converir(packetBuffer[0]);
      b=converir(packetBuffer[1]);
      digitalWrite(13,HIGH);
      Wire.beginTransmission(i2cAddress);          // Drive motor 1 at speed value stored in x
      Wire.write(speed1);
      Wire.write(a);
      Wire.endTransmission();
      Wire.beginTransmission(i2cAddress);          // Drive motor 2 at speed value stored in x2
      Wire.write(speed2);
      Wire.write(b);
      Wire.endTransmission();
    }
    else{
    WiFi.disconnect();
    WiFi.begin(ssid);
     }
   }
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
 // Serial.print("SSID: ");
  //Serial.println(WiFi.SSID());
  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  //Serial.print("IP Address: ");
  //Serial.println(ip);
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  //Serial.print("signal strength (RSSI):");
  //Serial.print(rssi);
  //Serial.println(" dBm");
}

int converir(int dato)
{
    if(dato<0)
   {
     int negdato=0;
     negdato=dato*(-1);
     dato=negdato+((dato+128)*2);
   }
   return dato;
}
```

udpR2.ino

```
#include <SPI.h>
#include <Wire.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#define i2cAddress 0x58
#define speed1 0x00                                  // speed to first motor
#define speed2 0x01
/*     // the IP address for the shield:    IPAddress ip(192, 168, 0, 110);    */
char ssid[] = "RobotMovil"; //  your network SSID (name)
char pass[] = "holamundo";    // your network password (use for WPA, or use as key for WEP)

int keyIndex = 0;           // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
int a,b;
int localPort =555;      // local port to listen on
char packetBuffer[255]; //buffer to hold incoming packet

WiFiUDP Udp;
void setup() {
  Wire.begin();
  //Initialize serial and wait for port to open:
  //Serial.begin(9600);
  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    //Serial.println("WiFi shield not present");
    // don't continue:
    while(true);
  }
//  WiFi.config(ip); // ip forzada
  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    //Serial.print("Attempting to connect to SSID: ");
    //Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);
    // wait 10 seconds for connection:
    delay(10000);
  }

  Udp.begin(localPort);
 }


void loop() {
  //Serial.print("Entre al loop ");
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if(packetSize)
  {
    int len = Udp.read(packetBuffer,255);
```

```
    if (len >0) {
      packetBuffer[len]=0;
      a=converir(packetBuffer[0]);
      b=converir(packetBuffer[1]);
      digitalWrite(13,HIGH);
      Wire.beginTransmission(i2cAddress);          // Drive motor 1 at speed value stored in x
      Wire.write(speed1);
      Wire.write(a);
      Wire.endTransmission();
      //Serial.println(a);

      Wire.beginTransmission(i2cAddress);          // Drive motor 2 at speed value stored in x2
      Wire.write(speed2);
      Wire.write(b);
      Wire.endTransmission();
      //Serial.println(b);
    }
    else{
    WiFi.disconnect();
    WiFi.begin(ssid);
     }


   }
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
 // Serial.print("SSID: ");
  //Serial.println(WiFi.SSID());
  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  //Serial.print("IP Address: ");
  //Serial.println(ip);
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  //Serial.print("signal strength (RSSI):");
  //Serial.print(rssi);
  //Serial.println(" dBm");
}

int converir(int dato)
{
    if(dato<0)
   {
     int negdato=0;
     negdato=dato*(-1);
     dato=negdato+((dato+128)*2);
   }

   return dato;
}
```

udpR3.ino

```
#include <SPI.h>
#include <Wire.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#define i2cAddress 0x58
#define speed1 0x00                                  // speed to first motor
#define speed2 0x01
/*    // the IP address for the shield:    IPAddress ip(192, 168, 0, 110);     */
char ssid[] = "RobotMovil"; //  your network SSID (name)
char pass[] = "holamundo";    // your network password (use for WPA, or use as key for WEP)

int keyIndex = 0;           // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
int a,b;
int localPort =12000;      // local port to listen on
char packetBuffer[255]; //buffer to hold incoming packet


WiFiUDP Udp;
void setup() {
  Wire.begin();
  //Initialize serial and wait for port to open:
  //Serial.begin(9600);

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    //Serial.println("WiFi shield not present");
    // don't continue:
    while(true);
  }
//  WiFi.config(ip); // ip forzada
  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    //Serial.print("Attempting to connect to SSID: ");
    //Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }

  Udp.begin(localPort);
 }

void loop() {
  //Serial.print("Entre al loop ");
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if(packetSize)
```

```
 {
   int len = Udp.read(packetBuffer,255);
   if (len >0) {
     packetBuffer[len]=0;
     a=converir(packetBuffer[0]);
     b=converir(packetBuffer[1]);
     digitalWrite(13,HIGH);
     Wire.beginTransmission(i2cAddress);          // Drive motor 1 at speed value stored in x
     Wire.write(speed1);
     Wire.write(a);
     Wire.endTransmission();
     Wire.beginTransmission(i2cAddress);          // Drive motor 2 at speed value stored in x2
     Wire.write(speed2);
     Wire.write(b);
     Wire.endTransmission();
   }
   else{
   WiFi.disconnect();
   WiFi.begin(ssid);
    }

  }
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
 // Serial.print("SSID: ");
  //Serial.println(WiFi.SSID());
  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  //Serial.print("IP Address: ");
  //Serial.println(ip);
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  //Serial.print("signal strength (RSSI):");
  //Serial.print(rssi);
  //Serial.println(" dBm");
}

int converir(int dato)
{
   if(dato<0)
  {
    int negdato=0;
    negdato=dato*(-1);
    dato=negdato+((dato+128)*2);
  }
  return dato;
}
```

udpR4.ino

```
#include <SPI.h>
#include <Wire.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#define i2cAddress 0x58
#define speed1 0x00                                  // speed to first motor
#define speed2 0x01
/*
// the IP address for the shield:
IPAddress ip(192, 168, 0, 110);
*/
char ssid[] = "RobotMovil"; //  your network SSID (name)
char pass[] = "holamundo";    // your network password (use for WPA, or use as key for WEP)

int keyIndex = 0;           // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;
int a,b;
unsigned int localPort =15000;      // local port to listen on

char packetBuffer[255]; //buffer to hold incoming packet


WiFiUDP Udp;
void setup() {
  Wire.begin();
  //Initialize serial and wait for port to open:
  //Serial.begin(9600);

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    //Serial.println("WiFi shield not present");
    // don't continue:
    while(true);
  }
//  WiFi.config(ip); // ip forzada
  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    //Serial.print("Attempting to connect to SSID: ");
    //Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  Udp.begin(localPort);
 }
```

```
void loop() {
  //Serial.print("Entre al loop ");
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if(packetSize)
  {
    int len = Udp.read(packetBuffer,255);
    if (len >0) {
      packetBuffer[len]=0;
      a=converir(packetBuffer[0]);
      b=converir(packetBuffer[1]);
      digitalWrite(13,HIGH);
      Wire.beginTransmission(i2cAddress);          // Drive motor 1 at speed value stored in x
      Wire.write(speed1);
      Wire.write(a);
      Wire.endTransmission();
      Wire.beginTransmission(i2cAddress);          // Drive motor 2 at speed value stored in x2
      Wire.write(speed2);
      Wire.write(b);
      Wire.endTransmission();
     }
    else{
    WiFi.disconnect();
    WiFi.begin(ssid);
     }
   }
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
 // Serial.print("SSID: ");
  //Serial.println(WiFi.SSID());
  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  //Serial.print("IP Address: ");
  //Serial.println(ip);
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  //Serial.print("signal strength (RSSI):");
  //Serial.print(rssi);
  //Serial.println(" dBm");
}

int converir(int dato)
{
    if(dato<0)
   {
     int negdato=0;
     negdato=dato*(-1);
     dato=negdato+((dato+128)*2);
   }
   return dato;
}
```

UDPENCODERS.ino

```
#include <WiFi.h>
#include <WiFiUdp.h>
#define i2cAddress 0x58
#define speed1 0x00                                  // speed to first motor
#define speed2 0x01
/*    // the IP address for the shield:    IPAddress ip(192, 168, 0, 110);    */
char ssid[] = "RobotMovil"; //  your network SSID (name)
char pass[] = "holamundo";    // your network password (use for WPA, or use as key for WEP)

int keyIndex = 0;           // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
int a,b;
unsigned int localPort =56000;      // local port to listen on
unsigned int remotelPort =10;
byte packetBuffer[8]; //buffer to hold incoming packet

WiFiUDP Udp;

unsigned int sensor1, sensor2, sensor3, sensor4;
int x = 0;
int xx = 0;
const int sen1 = A0;
const int sen2 = A1;
const int sen3 = A2;
const int sen4 = A3;
int clock1 = A4;
int cs1 = A5;
byte sensores[8] = {0,0,0,0,0,0,0,0};

void setup()
{
  if (WiFi.status() == WL_NO_SHIELD) {
    //Serial.println("WiFi shield not present");
    // don't continue:
    while(true);
  }
  while ( status != WL_CONNECTED) {
    status = WiFi.begin(ssid, pass);

    delay(10000);
  }
  Udp.begin(localPort);
  pinMode(cs1,OUTPUT);
  pinMode(clock1,OUTPUT);
  pinMode(sen1,INPUT);//Entrada del sensor 1
  pinMode(sen2,INPUT);
  pinMode(sen3,INPUT);
  pinMode(sen4,INPUT);
}
```

```
void lectura()    {
  x = 0;
  sensor1 = 0;
  sensor2 = 0;
  sensor3 = 0;
  sensor4 = 0;
  digitalWrite(cs1,HIGH);
  digitalWrite(clock1,HIGH);
  delayMicroseconds(5);
  digitalWrite(cs1,LOW);
  delayMicroseconds(10);

  while(x < 12)      {
    digitalWrite(clock1,LOW);
    delayMicroseconds(15);
    digitalWrite(clock1,HIGH);
    __asm__("nop\n\t"); //62.5 ns
    __asm__("nop\n\t"); //62.5 ns
    __asm__("nop\n\t"); //62.5 ns
    __asm__("nop\n\t"); //62.5 ns
    __asm__("nop\n\t"); //62.5 ns
    __asm__("nop\n\t"); //62.5 ns
    if (digitalRead(sen1))
      bitSet(sensor1, 11-x);
    if (digitalRead(sen2))
      bitSet(sensor2, 11-x);
    if (digitalRead(sen3))
      bitSet(sensor3, 11-x);
    if (digitalRead(sen4))
      bitSet(sensor4, 11-x);

    x++;
  }
  packetBuffer[0] = lowByte(sensor1);
  packetBuffer[1] = highByte(sensor1);
  packetBuffer[2] = lowByte(sensor2);
  packetBuffer[3] = highByte(sensor2);
  packetBuffer[4] = lowByte(sensor3);
  packetBuffer[5] = highByte(sensor3);
  packetBuffer[6] = lowByte(sensor4);
  packetBuffer[7] = highByte(sensor4);
}

void loop()
{
  lectura();
  char  ReplyBuffer[] = "12345678";
  IPAddress apc(192,168,0,100);
  Udp.beginPacket(apc, 10);
  Udp.write(packetBuffer,8);
  Udp.endPacket();
  delay(2); //delay
}
```

Robot_Serial_Wifi.ino

```
#include <SPI.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#include <Servo.h>
/*     // the IP address for the shield:     IPAddress ip(192, 168, 0, 107);    */
char ssid[] = "RobotMovil"; //  your network SSID (name)
char pass[] = "holamundo";    // your network password (use for WPA, or use as key for WEP)

int keyIndex = 0;           // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
unsigned int localPort =13000;      // local port to listen on
char packetBuffer[255]; //buffer to hold incoming packet

WiFiUDP Udp;

Servo rotZ, brazo1;  // create servo object to control a servo
Servo brazo2, brazo3, girogrip, grip;  // create servo object to control a servo
byte servo1, servo2, servo3, servo4, servo5, servo6;    // variable to read the value from Wifi

void setup()
{
 // check for the presence of the shield:
 if (WiFi.status() == WL_NO_SHIELD)
 {
  //Serial.println("WiFi shield not present");
  // don't continue:
  while(true);
 }
 //  WiFi.config(ip); // ip forzada
 // attempt to connect to Wifi network:
 while ( status != WL_CONNECTED)
 {
  //Serial.print("Attempting to connect to SSID: ");
  //Serial.println(ssid);
  // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
  status = WiFi.begin(ssid, pass);
  // wait 10 seconds for connection:
  delay(10000);
 }
 Udp.begin(localPort);

 //Wifi uses ports 10, 7, 4. Also on UNO 11,12,13. in MEGA 50,51,52.

 rotZ.attach(13);  // attaches the servo on pin 9 to the servo object
 brazo1.attach(12);  // attaches the servo on pin 9 to the servo object
 brazo2.attach(11);  // attaches the servo on pin 9 to the servo object
 brazo3.attach(9);  // attaches the servo on pin 9 to the servo object
 girogrip.attach(6);  // attaches the servo on pin 9 to the servo object
 grip.attach(5);  // attaches the servo on pin 9 to the servo object
}
```

```
void loop()
{
 // if there's data available, read a packet
 int packetSize = Udp.parsePacket();
 if(packetSize)
 {
  int len = Udp.read(packetBuffer,255);
  if (len >0)
  {
   packetBuffer[len]=0;
   servo1=packetBuffer[0];
   servo2=packetBuffer[1];
   servo3=packetBuffer[2];
   servo4=packetBuffer[3];
   servo5=packetBuffer[4];
   servo6=packetBuffer[5];

   rotZ.write(servo1);              // sets the servo position according to the scaled value
   brazo1.write(servo2);              // sets the servo position according to the scaled value
   brazo2.write(servo3);              // sets the servo position according to the scaled value
   brazo3.write(servo4);              // sets the servo position according to the scaled value
   girogrip.write(servo5);            // sets the servo position according to the scaled value
   grip.write(servo6);              // sets the servo position according to the scaled value
   delay(15);                     // waits for the servo to get there
  }

  else
  {
   WiFi.disconnect();
   WiFi.begin(ssid);
  }
 }
}

void printWifiStatus()
{
 // print the SSID of the network you're attached to:
 // Serial.print("SSID: ");
 //Serial.println(WiFi.SSID());
 // print your WiFi shield's IP address:
 IPAddress ip = WiFi.localIP();
 //Serial.print("IP Address: ");
 //Serial.println(ip);

 // print the received signal strength:
 long rssi = WiFi.RSSI();
 //Serial.print("signal strength (RSSI):");
 //Serial.print(rssi);
 //Serial.println(" dBm");
}
```

PROGRAMA EN MATHEMATICA

Análisis Cinemático (Paralelo Delta)

Ecuaciones Cinemáticas

In[139]:=
```
Clear[θ1, θ2, θ3, θ4, θ5, drs, r01, r12, r23, r34, r45]
(*Matrices de Transformación Homogéneas*)
Tx[x_] := {{1, 0, 0, x}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}};
Ty[y_] := {{1, 0, 0, 0}, {0, 1, 0, y}, {0, 0, 1, 0}, {0, 0, 0, 1}};
Tz[z_] := {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, z}, {0, 0, 0, 1}};
Qx[θx_] :=
  {{1, 0, 0, 0}, {0, Cos[θx], -Sin[θx], 0}, {0, Sin[θx], Cos[θx], 0}, {0, 0, 0, 1}};
Qy[θy_] := {{Cos[θy], 0, Sin[θy], 0}, {0, 1, 0, 0},
   {-Sin[θy], 0, Cos[θy], 0}, {0, 0, 0, 1}};
Qz[θz_] := {{Cos[θz], -Sin[θz], 0, 0}, {Sin[θz], Cos[θz], 0, 0},
   {0, 0, 1, 0}, {0, 0, 0, 1}};
coord = {0, 0, 0, 1};

Roo = {0, 0, 0, 1};
Ro0 = Ty[drs];

(*Robot Serial de 5GDL*)
(*Cadena*)
R01 = Qz[Pi].Qz[-θ1].Tz[r01];
R12 = Qy[-θ2].Tx[r12];
R23 = Qy[θ3].Tx[r23];
R34 = Qy[θ4].Tx[r34];
R45 = Qx[θ5].Tx[r45].Qx[-θ5].Qy[-θ4].Qy[-θ3].Qy[θ2].Qz[θ1].Qz[-Pi];
R05 = R01.R12.R23.R34.R45;

R05Ver = R05 // MatrixForm // FullSimplify

(*Cadena*)
R01v = Qz[Pi].Qz[-θ1].Tz[r01];
R12v = Qz[Pi].Qz[-θ1].Qy[-θ2].Tx[r12];
R23v = Qz[Pi].Qz[-θ1].Qy[-θ2].Qy[θ3].Tx[r23];
R34v = Qz[Pi].Qz[-θ1].Qy[-θ2].Qy[θ3].Qy[θ4].Tx[r34];
R45v = Qz[Pi].Qz[-θ1].Qy[-θ2].Qy[θ3].Qy[θ4].Qx[θ5].Tx[r45];
R05v = R01v + R12v + R23v + R34v + R45v;

R05vVer = R05v // MatrixForm // FullSimplify
```

Out[155]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & -\text{Cos}[\theta 1]\,(r12\,\text{Cos}[\theta 2] + r23\,\text{Cos}[\theta 2-\theta 3] + (r34+r45)\,\text{Cos}[\theta 2-\theta 3-\theta 4]) \\ 0 & 1 & 0 & (r12\,\text{Cos}[\theta 2] + r23\,\text{Cos}[\theta 2-\theta 3] + (r34+r45)\,\text{Cos}[\theta 2-\theta 3-\theta 4])\,\text{Sin}[\theta 1] \\ 0 & 0 & 1 & r01 + r12\,\text{Sin}[\theta 2] + r23\,\text{Sin}[\theta 2-\theta 3] + (r34+r45)\,\text{Sin}[\theta 2-\theta 3-\theta 4] \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Out[162]//MatrixForm=

$$\begin{pmatrix} -\text{Cos}[\theta 1]\,(1+\text{Cos}[\theta 2]+\text{Cos}[\theta 2-\theta 3]+2\,\text{Cos}[\theta 2-\theta 3-\theta 4]) & -(4+\text{Cos}[\theta 5])\,\text{Sin}[\theta 1]+\text{Cos}[\theta 1]\,\text{S} \\ (1+\text{Cos}[\theta 2]+\text{Cos}[\theta 2-\theta 3]+2\,\text{Cos}[\theta 2-\theta 3-\theta 4])\,\text{Sin}[\theta 1] & -\text{Cos}[\theta 1]\,(4+\text{Cos}[\theta 5])-\text{Sin}[\theta 1]\,\text{S} \\ \text{Sin}[\theta 2]+\text{Sin}[\theta 2-\theta 3]+2\,\text{Sin}[\theta 2-\theta 3-\theta 4] & \text{Cos}[\theta 2-\theta 3-\theta 4]\,\text{S} \\ 0 & 0 \end{pmatrix}$$

DATOS

In[163]:=
```
(*Constantes del Robot Serial de 5GDL*)
Clear[θ1, θ2, θ3, θ4, θ5, drs, r01, r12, r23, r34, r45]
r01 = 60;
r12 = 95;
r23 = 145;
r34 = 65;
r45 = 50;
drs = 0;
```

Simulación Tridimensional

Definición de Puntos y Líneas

In[170]:=
```
(*Matriz de conversion entre coordenadas homogeneas y cartesianas*)
H = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}};
(*Convirtiendo los puntos a coordenadas cartesianas*)
P0 = H.Roo;
(*Robot Serial*)
P13 = H.Ro0.Roo;

(*Cadena*)
P14 = H.Ro0.R01.Roo;
P15 = H.Ro0.R01.R12.Roo;
P16 = H.Ro0.R01.R12.R23.Roo;
P17 = H.Ro0.R01.R12.R23.R34.Roo;
P18 = H.Ro0.R01.R12.R23.R34.R45.Roo;
P19 = H.Ro0.R05.Roo;

(*Brazos de Cadena*)
Brazo1 = Cylinder[{P14, P15}, 10];
Brazo2 = Cylinder[{P15, P16}, 10];
Brazo3 = Cylinder[{P16, P17}, 10];
Brazo4 = Cylinder[{P17, P18}, 10];
Efector = Cylinder[{P18 + {0, 30, 0}, P18 + {0, -30, 0}}, 6];

(*Base*)
Base = Cylinder[{P13, P14}, 90];
```

Análisis para la Simulación (N - R)

In[185]:=
```
Increm = 1 / 200;
x1 = 0; y1 = 100; z1 = 110; (*Punto Inicial*)
x2 = 100; y2 = 150; z2 = -100;(*Punto 2*)

θ4 = 90 * Degree;
θ5 = 45 * Degree;

(*Variables Aproximadas*)
θi = 91 * Degree;
θ2i = 143 * Degree;
θ3i = 122 * Degree;

(*Matriz para acondicinamiento y resolucion del análisis*)
n = {0, 0, 0, 1};
(*Acondicionando Coordenadas de Matrices*)
NvaR05 = R05.n;

For[j = 0, j ≤ 200, j += 1,
 Parametro[j] = j * Increm;
 x[j] = x1 + (x2 - x1) * Parametro[j];
 y[j] = y1 + (y2 - y1) * Parametro[j];
 z[j] = z1 + (z2 - z1) * Parametro[j];

 Resuelve[j] = FindRoot[
   {NvaR05[[1]] == x[j], NvaR05[[2]] == y[j], NvaR05[[3]] == z[j]},
   {θ1, θi},
   {θ2, θ2i},
   {θ3, θ3i},
   MaxIterations → 100];

 θi = θ1 /. Resuelve[j];
 θ2i = θ2 /. Resuelve[j];
 θ3i = θ3 /. Resuelve[j]
]
```

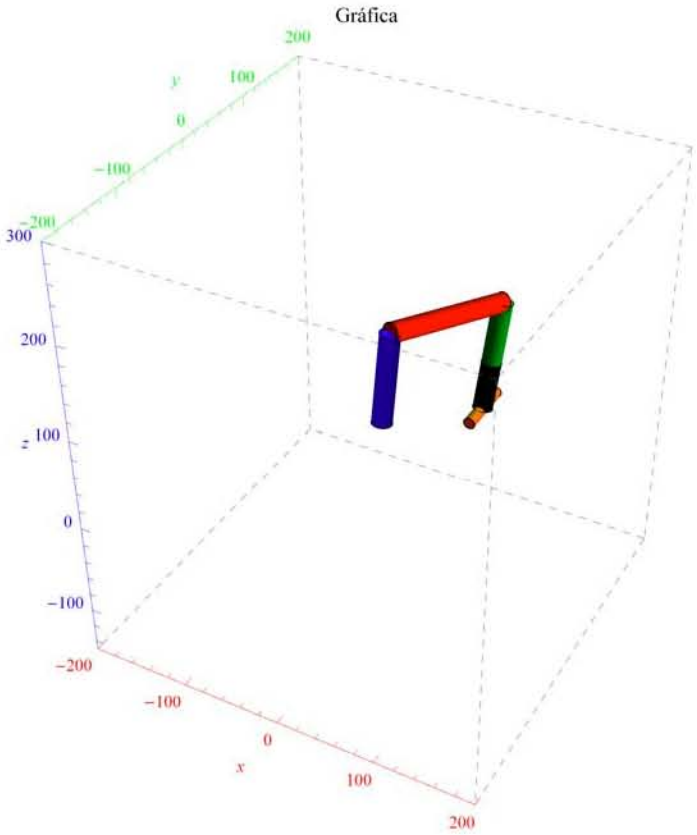
Primer Dibujo (Preliminar)

Animando

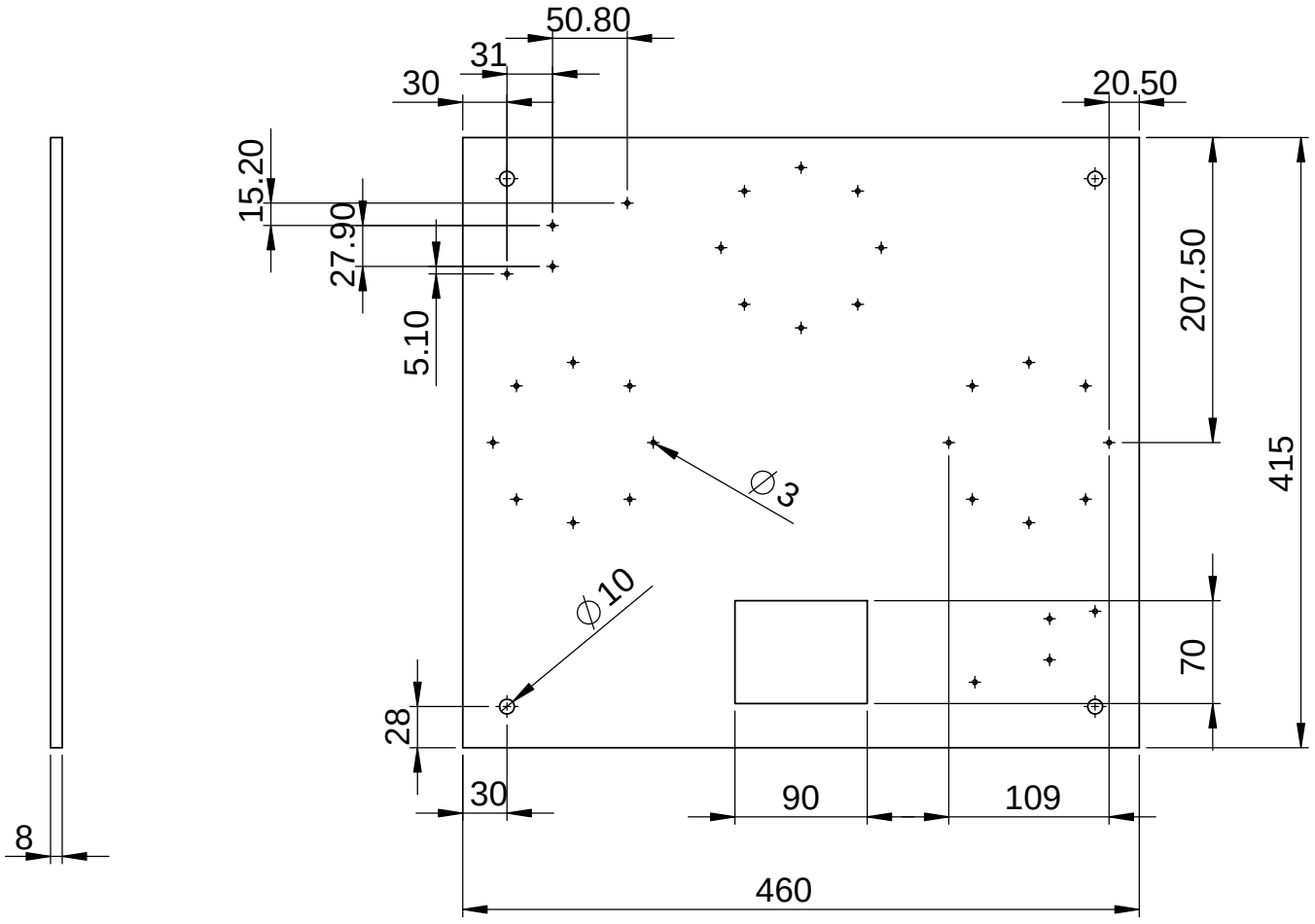
In[197]:=
```
Animate[
 Graphics3D[
  {Blue, Thick, Brazo1 /. Resuelve[j],
   Red, Brazo2 /. Resuelve[j],
   Green,  Brazo3 /. Resuelve[j],
   Black, Brazo4 /. Resuelve[j],
   Orange, Efector /. Resuelve[j]
  },
  PlotRange → {{-200, 200}, {-200, 200}, {-150, 300}},
  BoxStyle → Directive[Dashed], PlotLabel → Gráfica,
  Axes → True, AxesLabel → {x, y, z}, AxesStyle → {Red, Green, Blue}
  (*ViewPoint→{0,0,Infinity}*)],
 {j, 0, 200, 1}]

Averlo = ((θ1 /. Resuelve[1])) / Degree;
Averlo2 = ((θ1 /. Resuelve[100])) / Degree;
Averlo3 = ((θ1 /. Resuelve[200])) / Degree;

Averlo6 = ((θ2 /. Resuelve[1])) / Degree;
Averlo7 = ((θ2 /. Resuelve[100])) / Degree;
Averlo8 = ((θ2 /. Resuelve[200])) / Degree;

Averlo11 = ((θ3 /. Resuelve[1])) / Degree;
Averlo12 = ((θ3 /. Resuelve[100])) / Degree;
Averlo13 = ((θ3 /. Resuelve[200])) / Degree;


Export["manipuladorserial.xls",
  Table[{k, (θ1 /. Resuelve[k]) / Degree,
    (θ2 /. Resuelve[k]) / Degree, (θ3 /. Resuelve[k]) / Degree,
    (θ4 /. Resuelve[k]) / Degree, (θ5 /. Resuelve[k]) / Degree}, {k, 0, 200, 1}]];
```

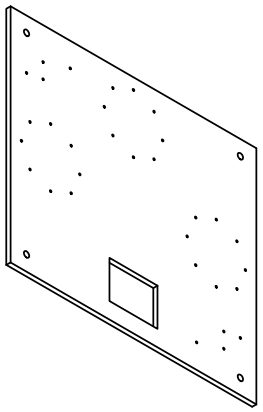
j

Out[197]=



PLANOS





PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

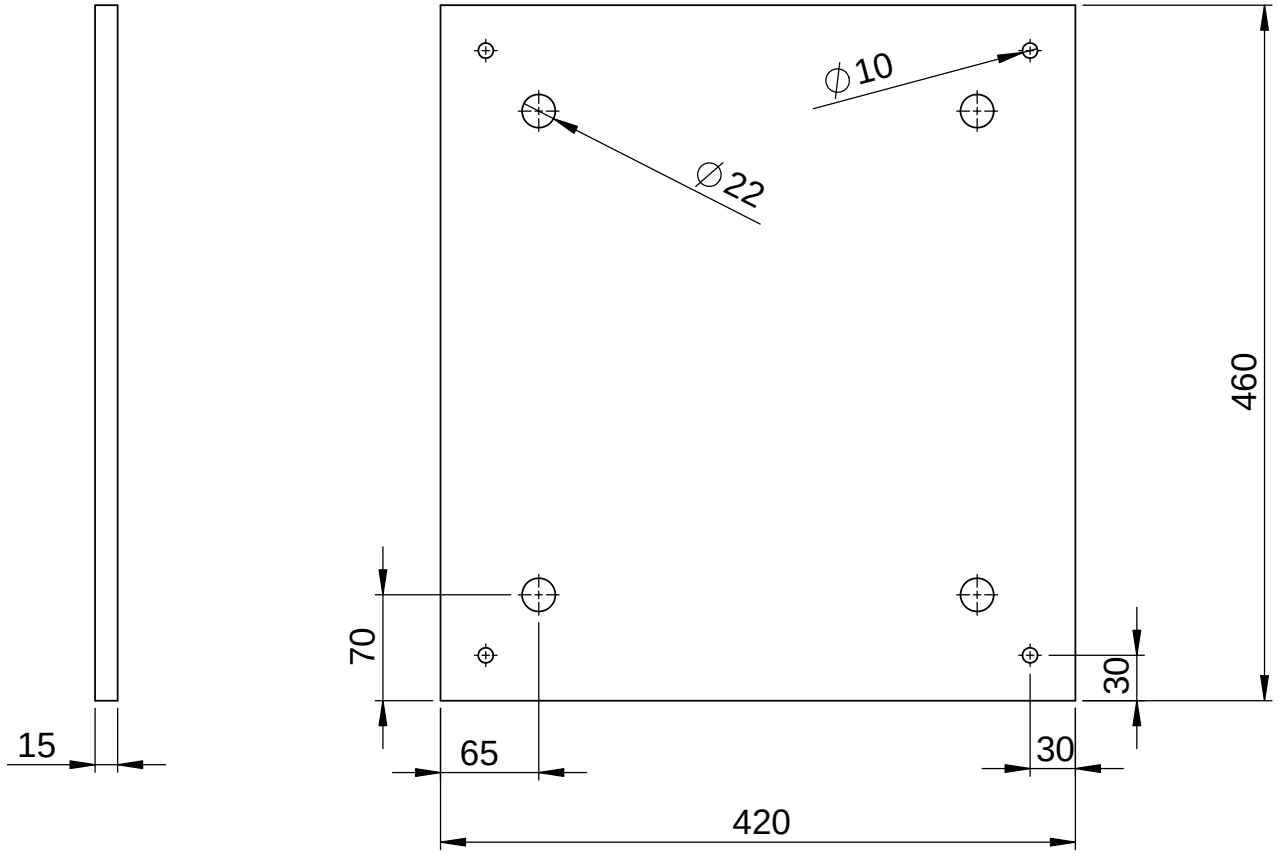
Acotación: [mm]

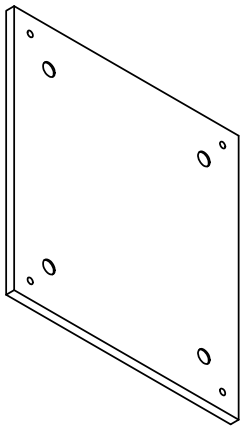
UNAM POSGRADO Ingeniería **MRG**

TÍTULO:

Base superior del móvil

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Acrílico	*1 de 25*	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 1	ESCALA: 1:5	TOL: +/- 0.1 [mm] Landscape





PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

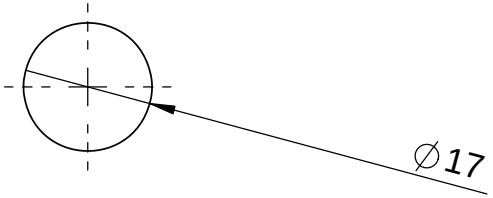
Acotación: [mm]

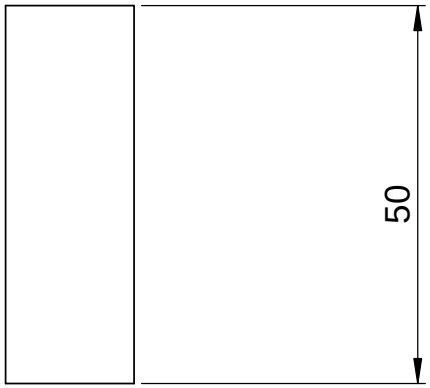
UNAM POSGRADO Ingeniería **MRG**

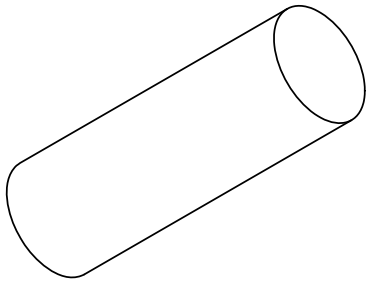
TÍTULO:

Base principal del móvil

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Placa de Aluminio 6061 T6	2 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 1	ESCALA: 1:5 TOL: +/- 0.1 [mm]	Landscape







PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Columna de 5cm

	NOMBRE	FECHA
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15
APROBÓ	Dr. Víctor J. González Villela	05/06/15

MATERIAL: Aluminio AISI 304

NÚMERO DE PIEZAS: 16

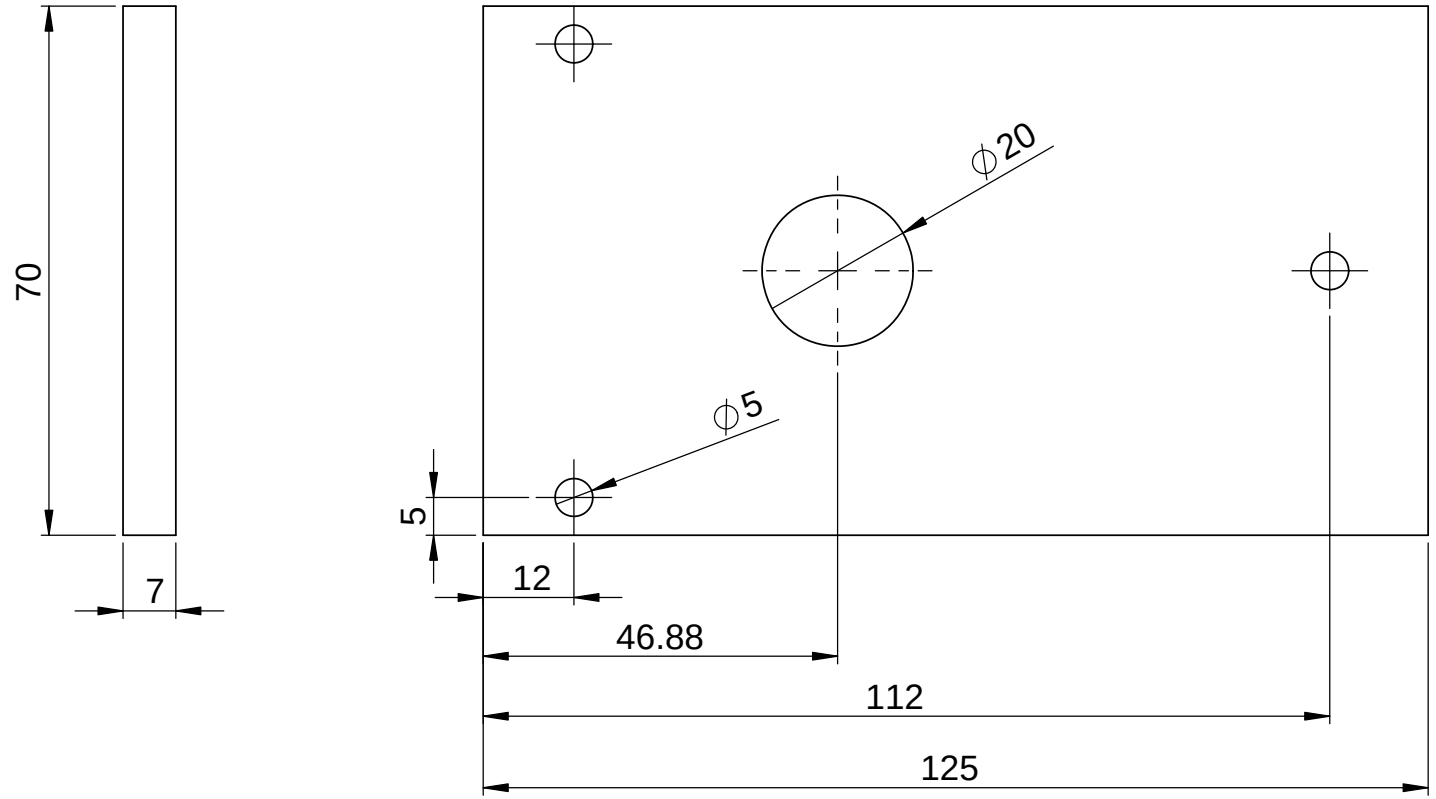
DIBUJO NÚMERO: 3 de 25

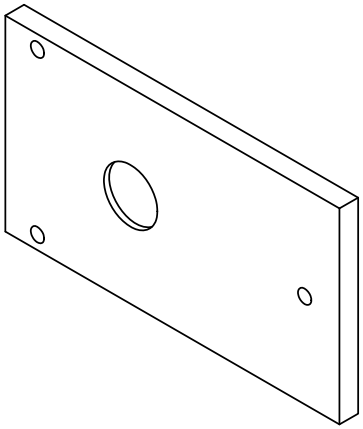
ESCALA: 1:1

TOL: +/- 0.1 [mm]

Formato: A (ANSI)

Landscape





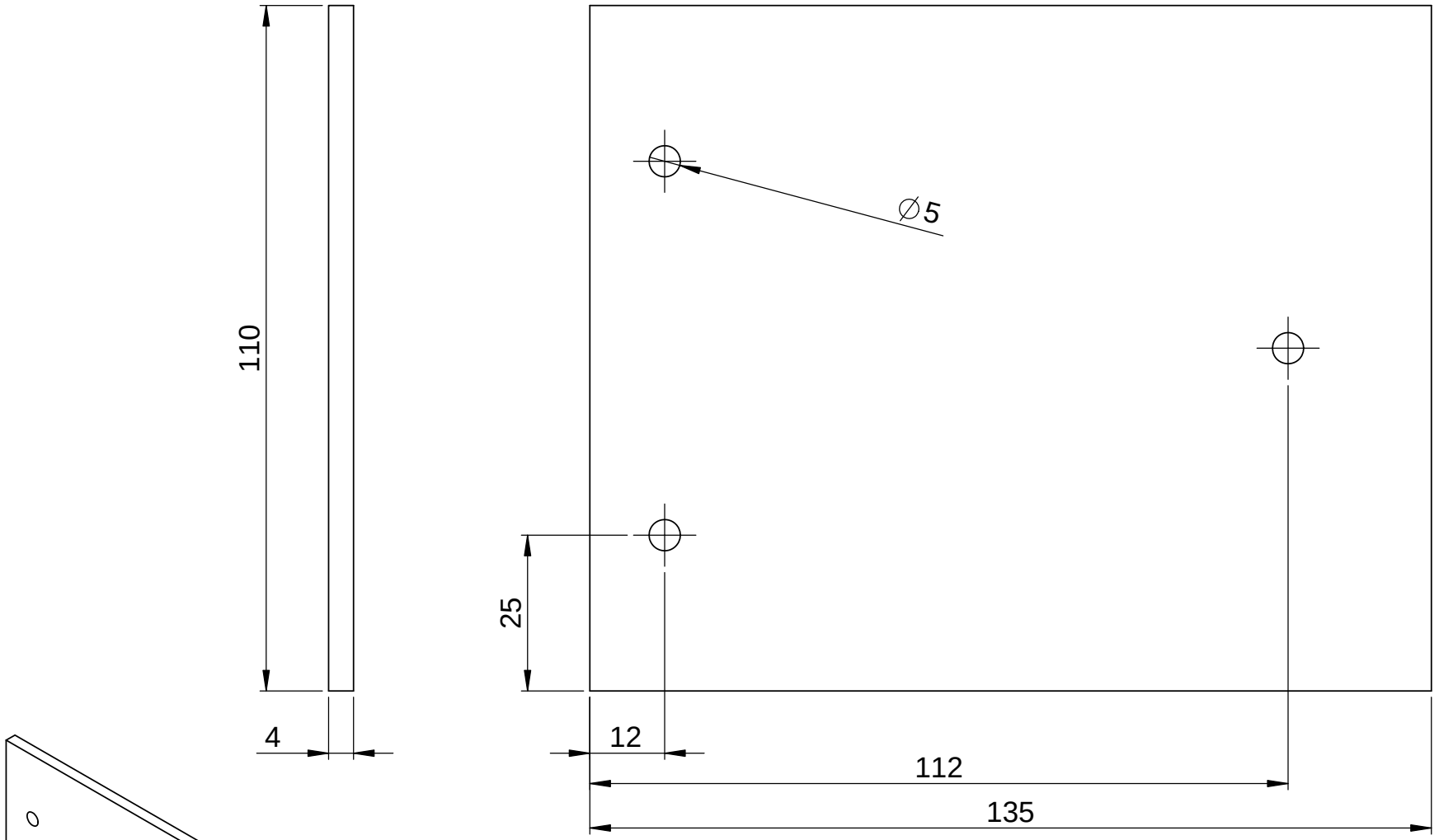
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

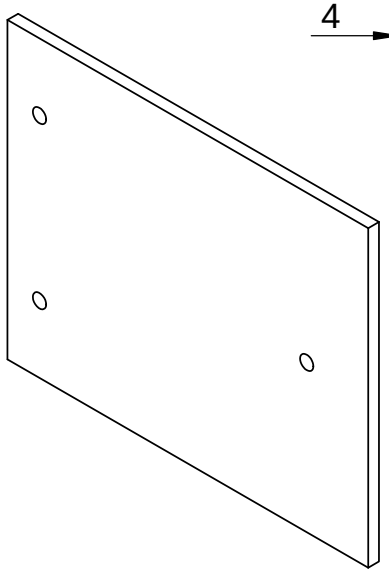
Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Base superior robot dif

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	4 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 4	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape





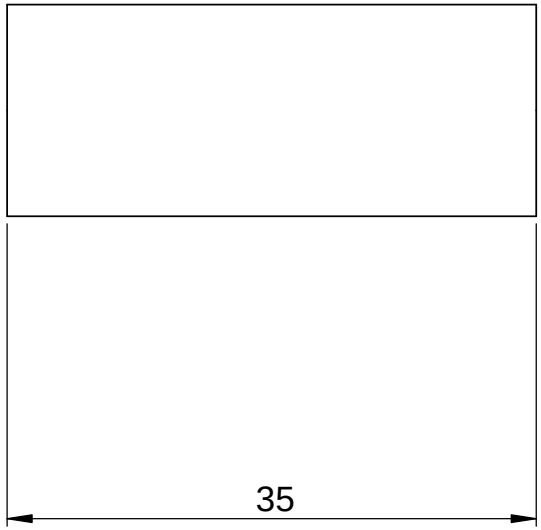
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

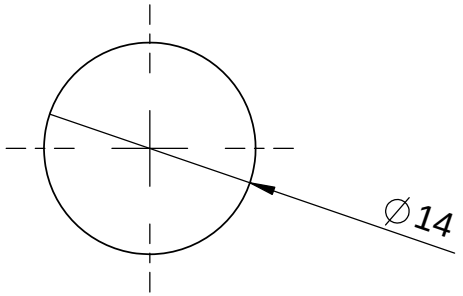
Acotación: [mm]

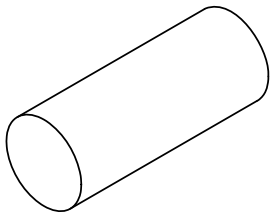
UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Base media robot dif

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Acrílico	5 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 8	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape

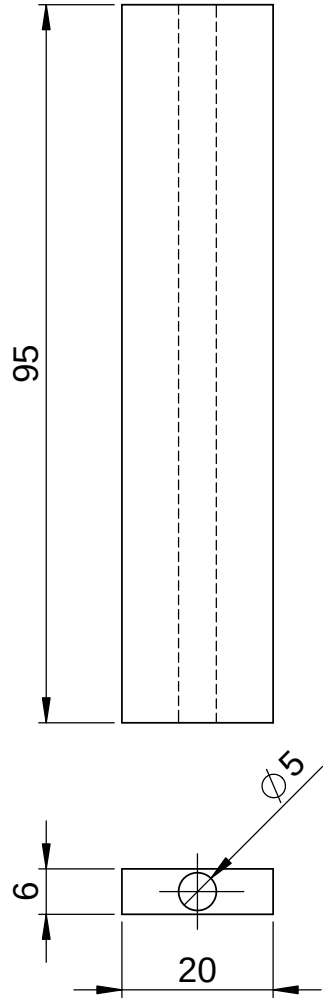


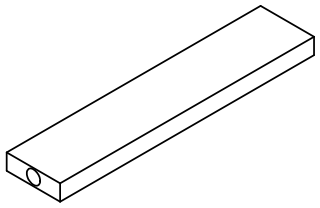




PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

Acotación: [mm]	UNAM POSGRADO Ingeniería MRG		TÍTULO: Columna de 3.5cm		
	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	Formato: A (ANSI)
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	6 de 25	
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 12	ESCALA: 2:1	TOL: +/- 0.1 [mm] Landscape





PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

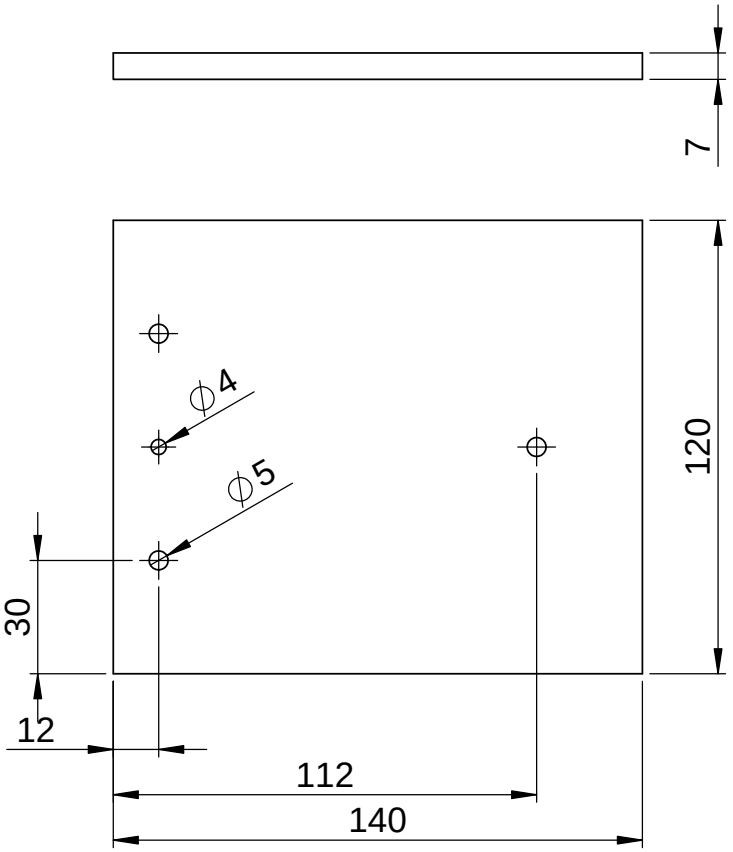
Acotación: [mm]

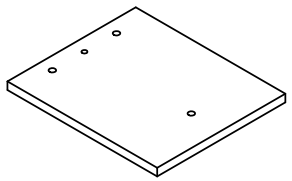
UNAM POSGRADO Ingeniería **MRG**

TÍTULO:

Columna prismática

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	7 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 12	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape





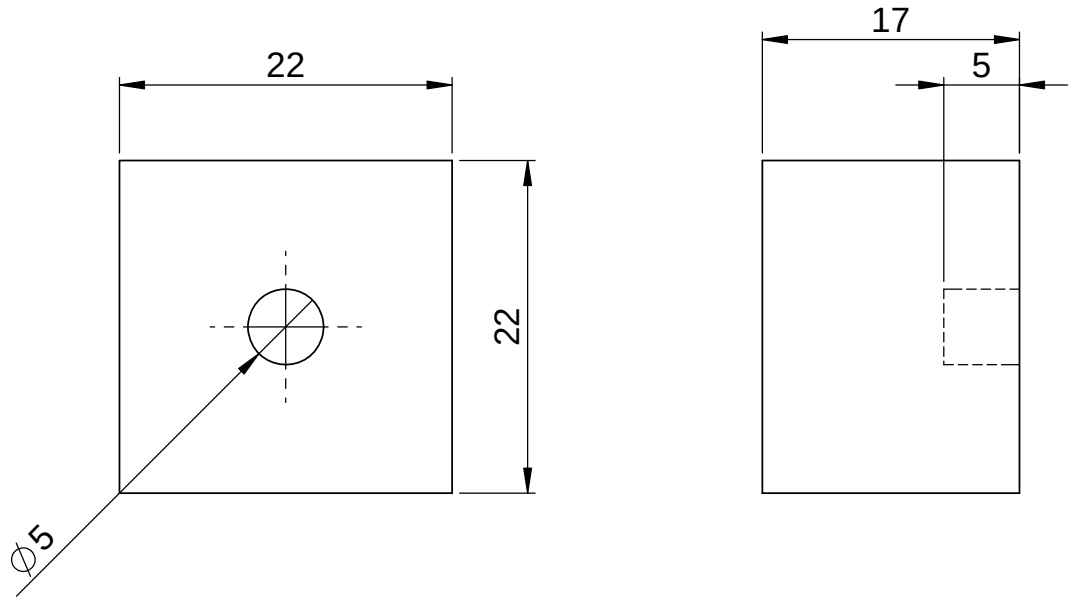
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

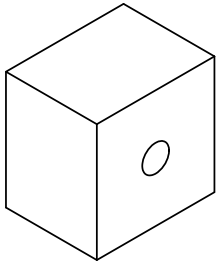
Acotación: [mm]

UNAM POSGRADO Ingeniería — MRG

TÍTULO: Base inferior robot dif

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	8 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 4	ESCALA: 1:2	TOL: +/- 0.1 [mm] Landscape





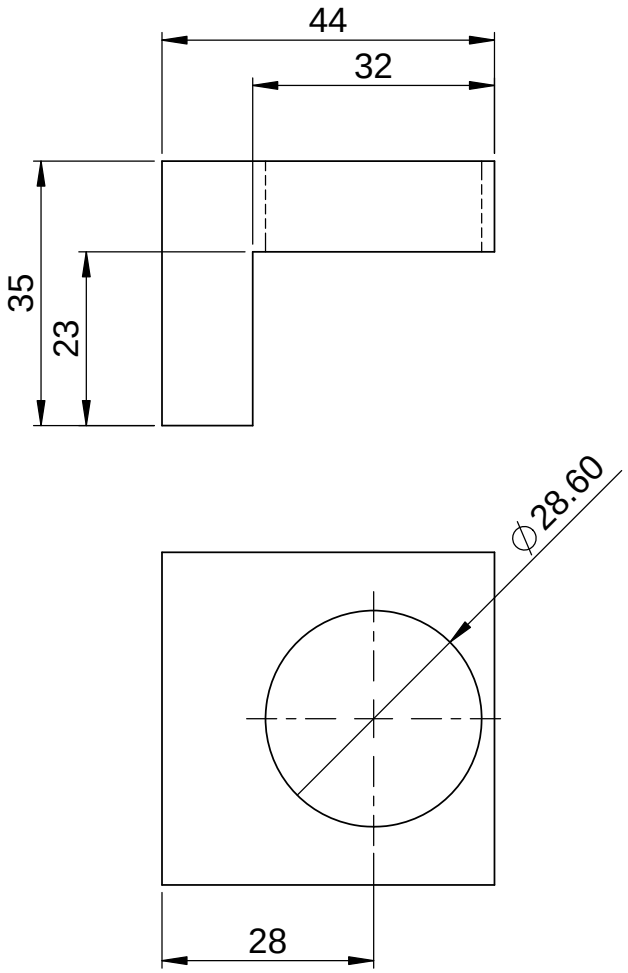
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

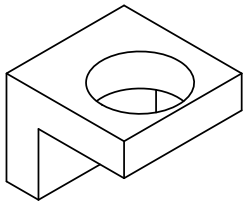
Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Soporte de eje

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	9 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 8	ESCALA: 2:1 TOL: +/- 0.1 [mm]	Landscape





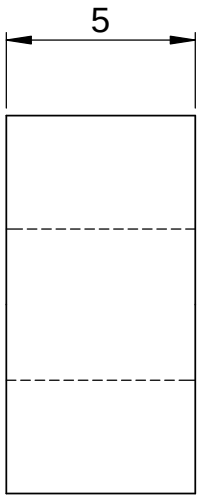
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

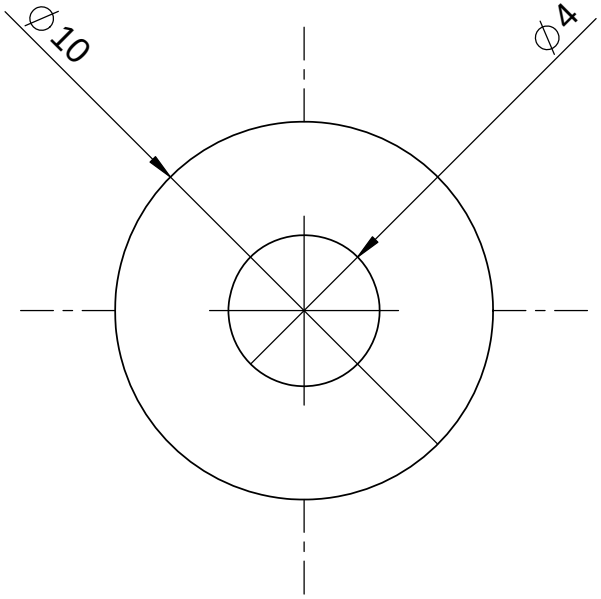
Acotación: [mm]

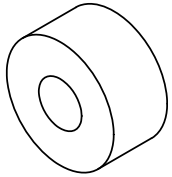
UNAM POSGRADO Ingeniería MRG

TÍTULO: Soporte de motor

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	10 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 8	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape

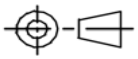






PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

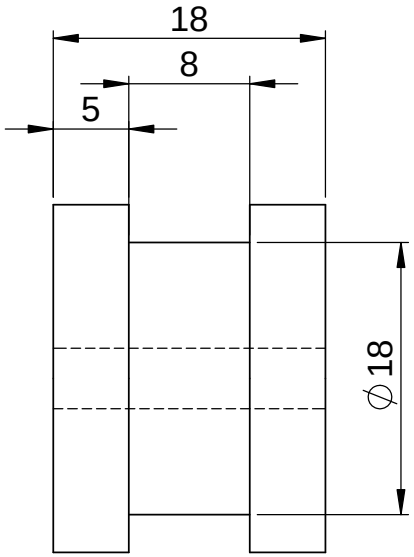
Acotación: [mm]

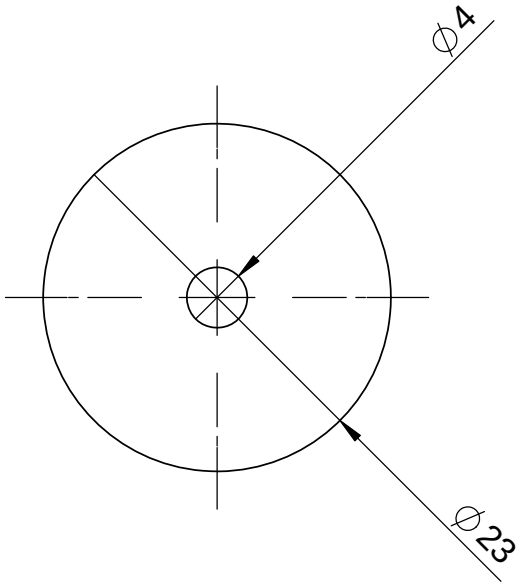


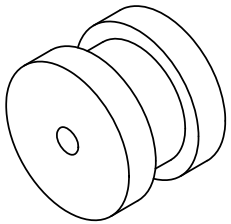
UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Inicio de eje

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Cobre	11 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 8	ESCALA: 5:1	TOL: +/- 0.1 [mm] Landscape

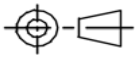






PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

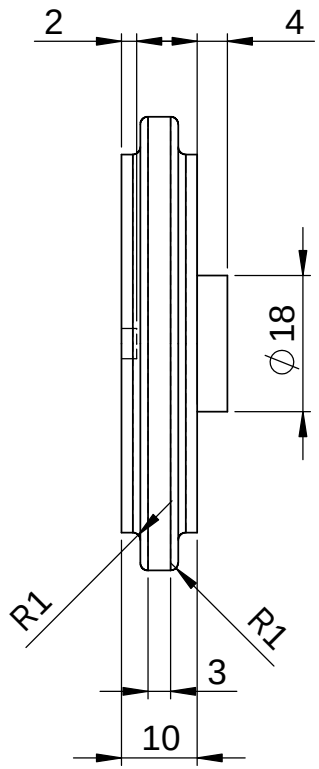
Acotación: [mm]

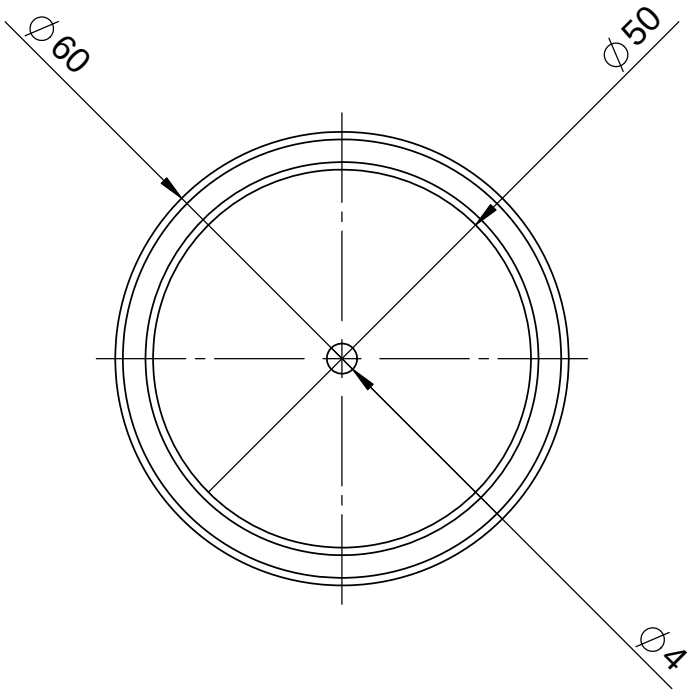


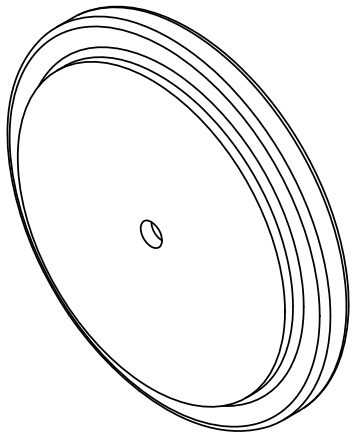
UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Eje de llanta

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	12 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 8	ESCALA: 2:1	TOL: +/- 0.1 [mm] Landscape

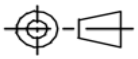






PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

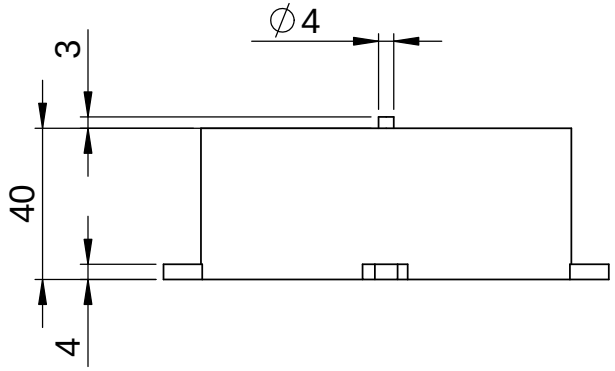
Acotación: [mm]

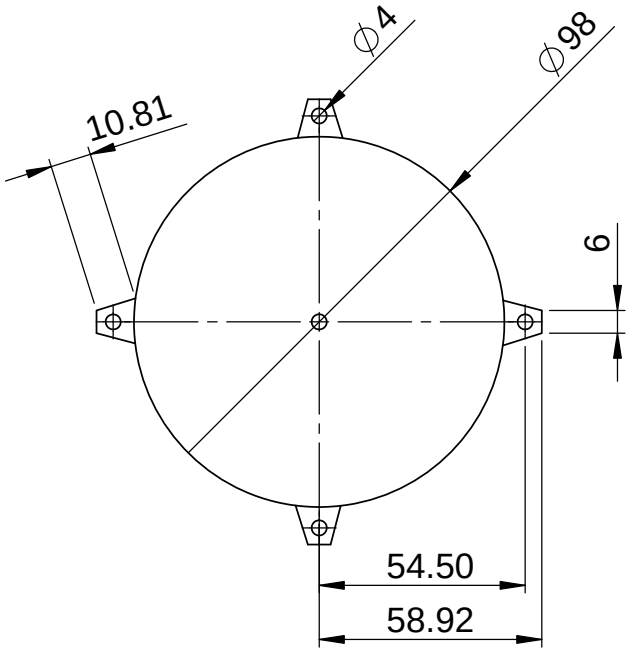


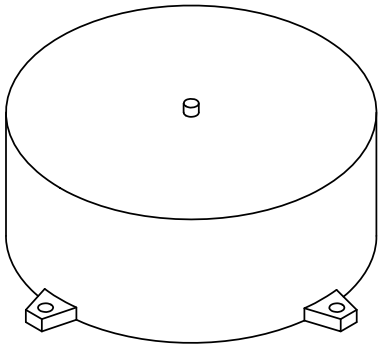
UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Llanta

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Plástico y Hule	*13 de 25*	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 8	ESCALA: 1:1 TOL: +/- 0.1 [mm]	Landscape







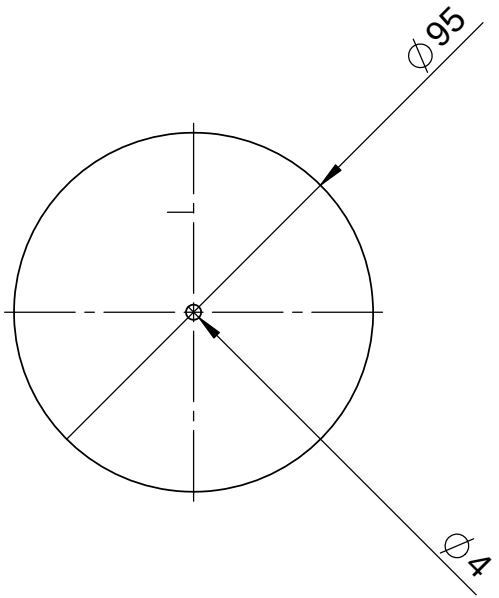
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

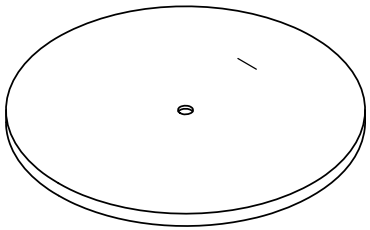
Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Base fija manipulador

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Plástico	14 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 1	ESCALA: 1:2	TOL: +/- 0.1 [mm] Landscape





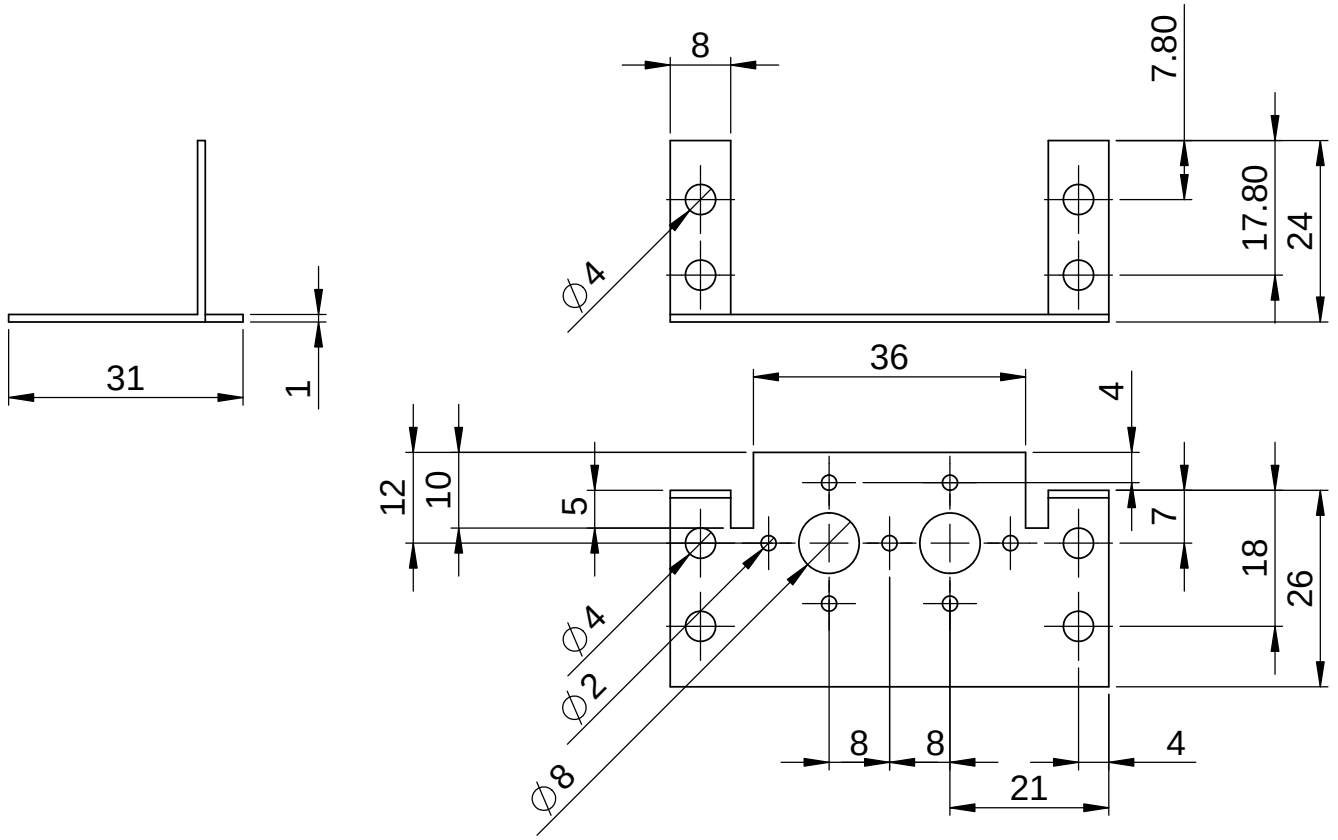
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

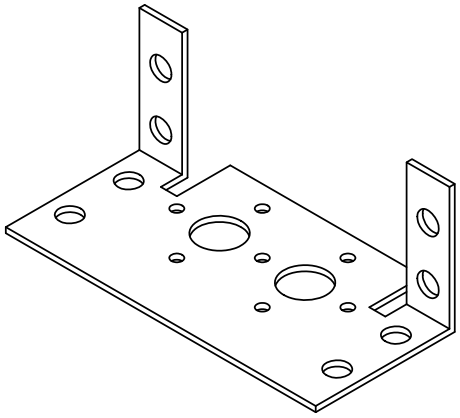
Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Base móvil manipulador

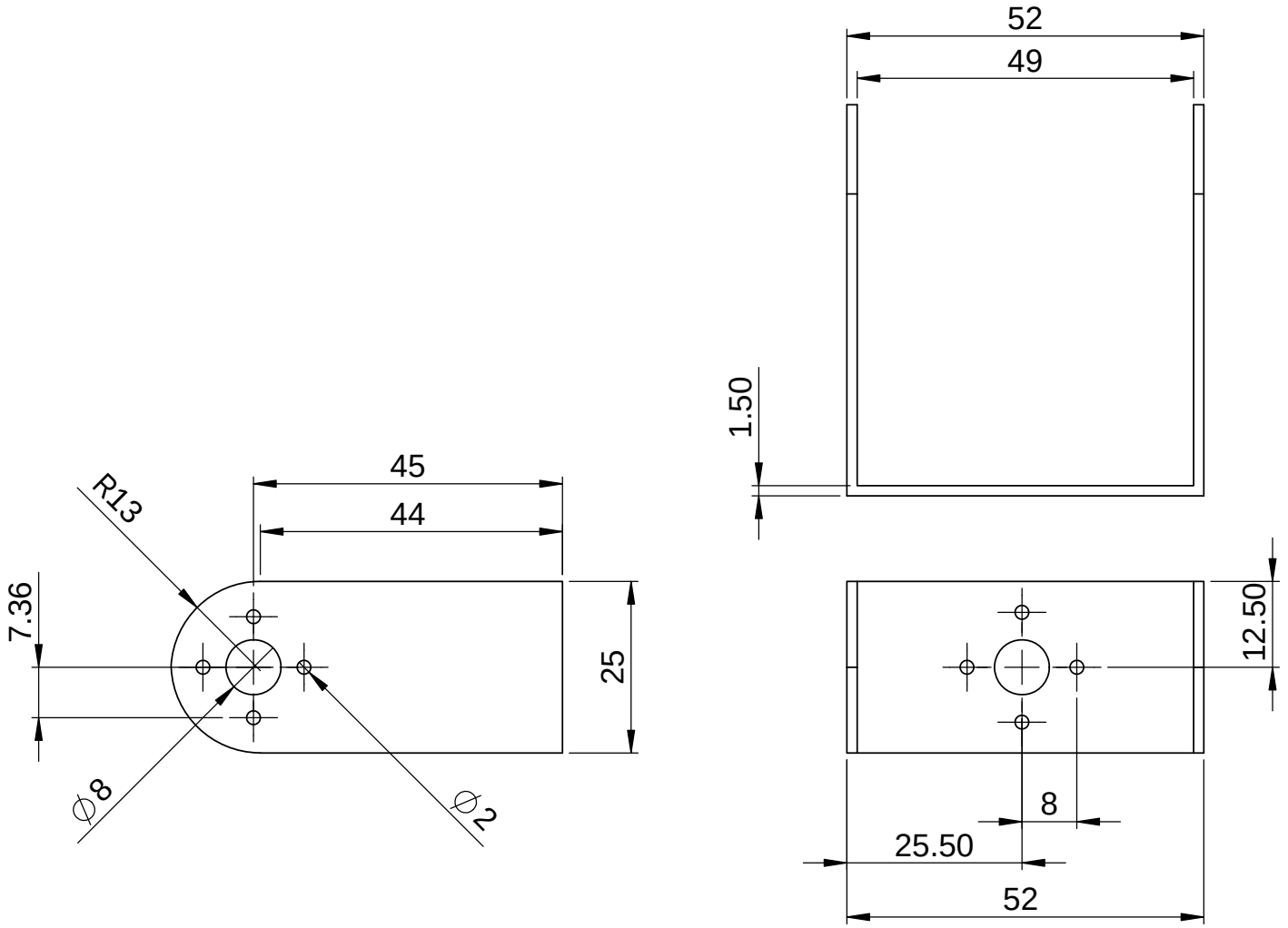
	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Plástico	15 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 1	ESCALA: 1:2	TOL: +/- 0.1 [mm] Landscape

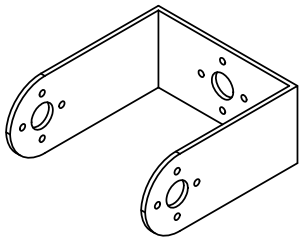




PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

Acotación: [mm]	UNAM POSGRADO Ingeniería MRG		TÍTULO: Sujeción Servos		
	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	Formato: A (ANSI)
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	16 de 25	
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 4	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape





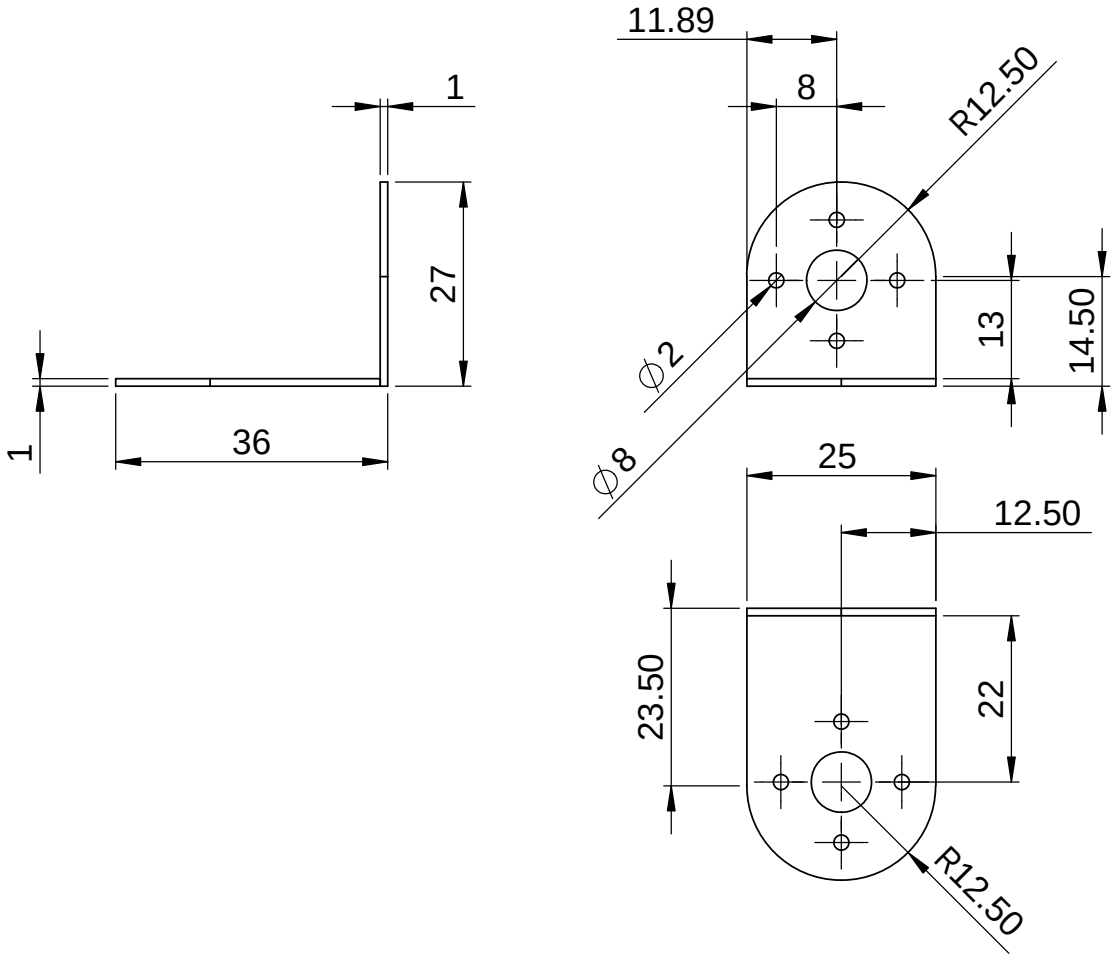
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

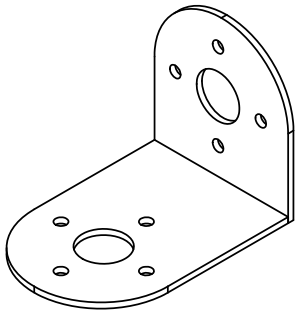
Acotación: [mm]

UNAM POSGRADO Ingeniería MRG

TÍTULO: Brazo U

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	17 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 2	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape





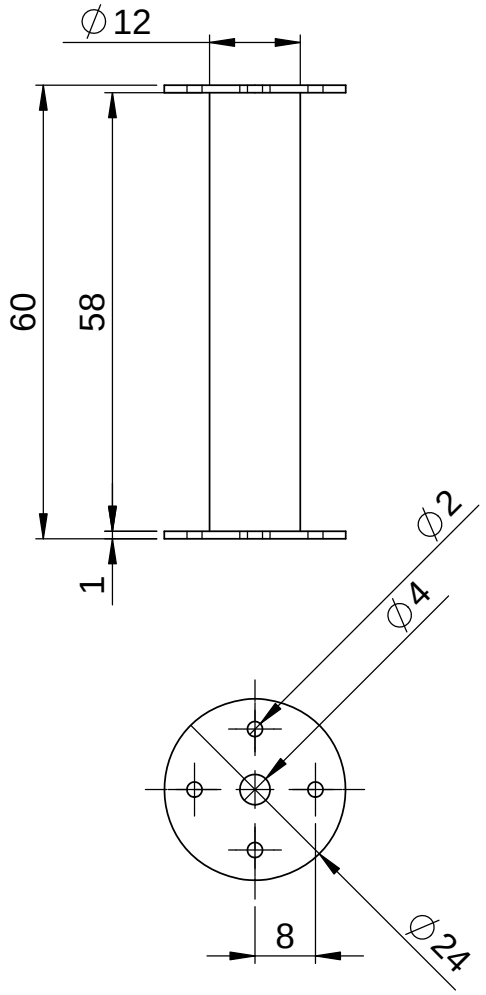
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

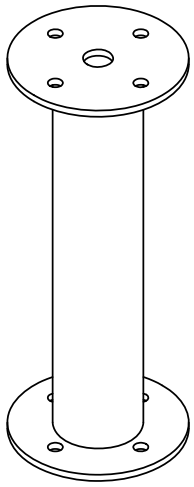
Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Sujeción L

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	*18 de 25*	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 2	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape





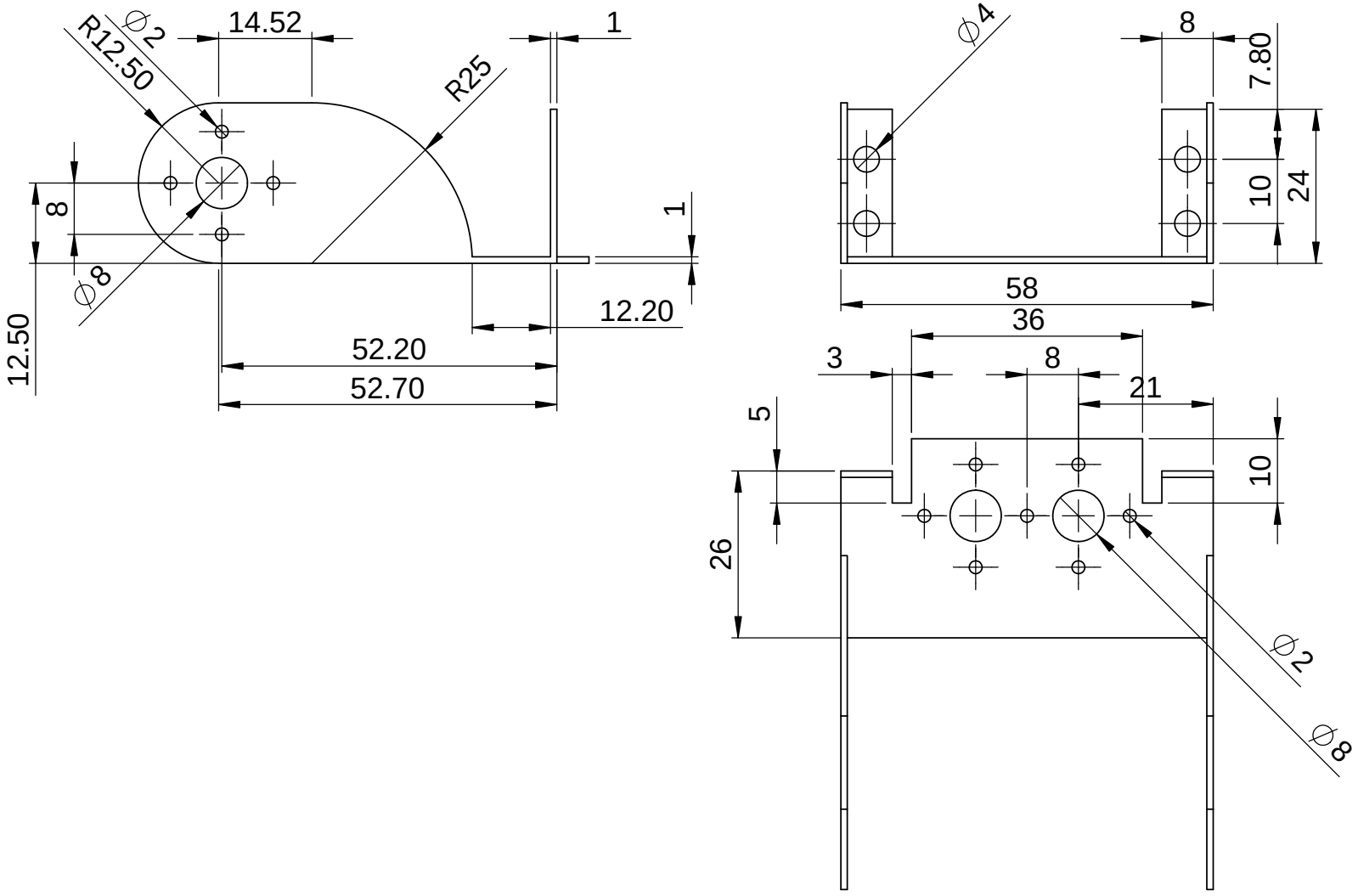
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

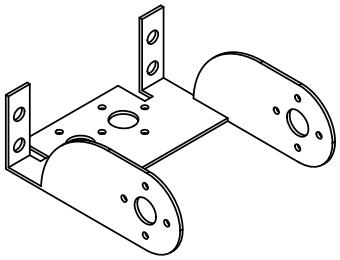
Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Brazo Tubular

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Aluminio 6061 T6	19 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 1	ESCALA: 1:1 TOL: +/- 0.1 [mm]	Landscape





PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

Acotación: [mm]

UNAM POSGRADO Ingeniería MRG

TÍTULO: Sujecion final

	NOMBRE	FECHA
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15
APROBÓ	Dr. Víctor J. González Villela	05/06/15

MATERIAL: Aluminio 6061 T6

NÚMERO DE PIEZAS: 1

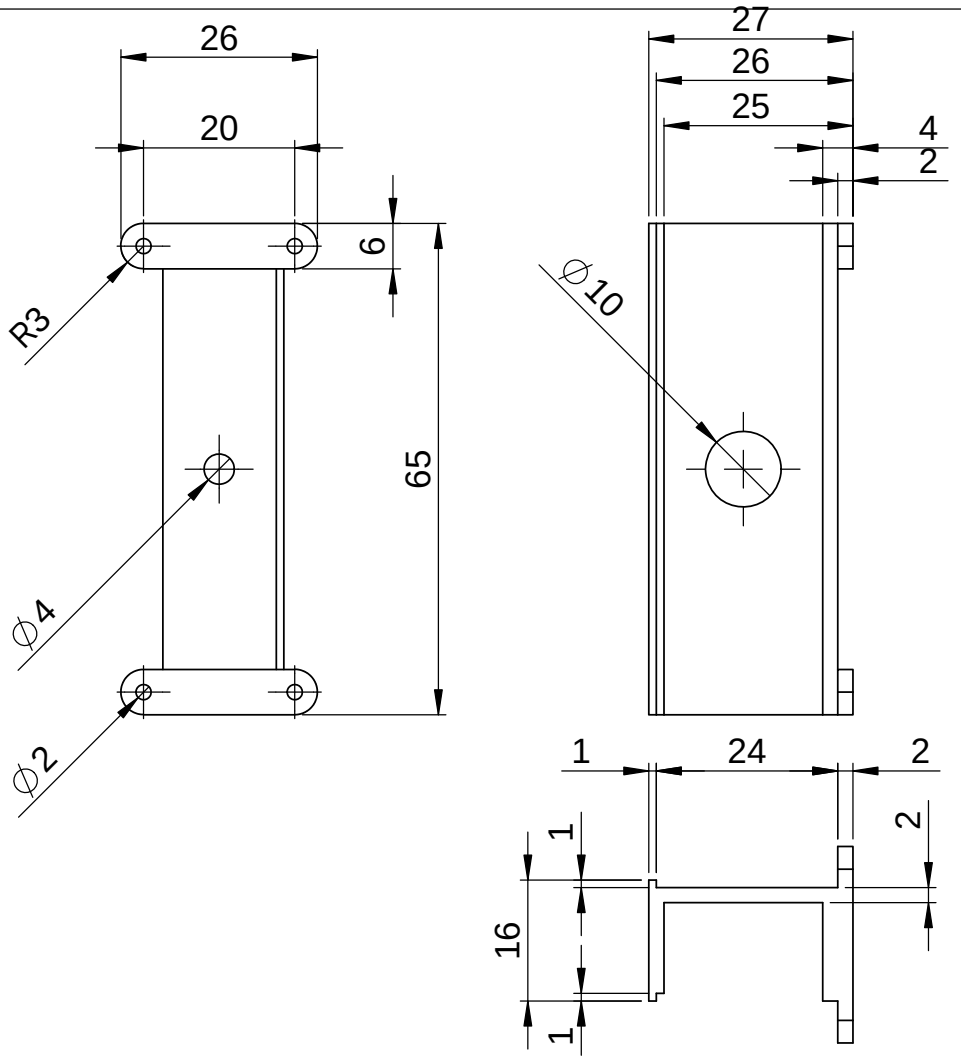
DIBUJO NÚMERO: 20 de 25

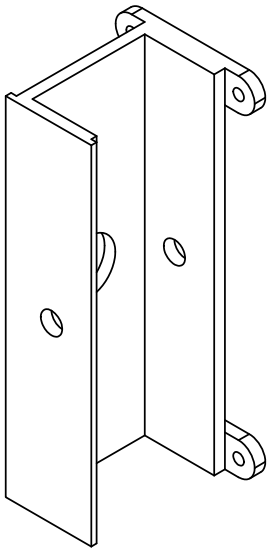
ESCALA: 1:1

TOL: +/- 0.1 [mm]

Formato: A (ANSI)

Landscape





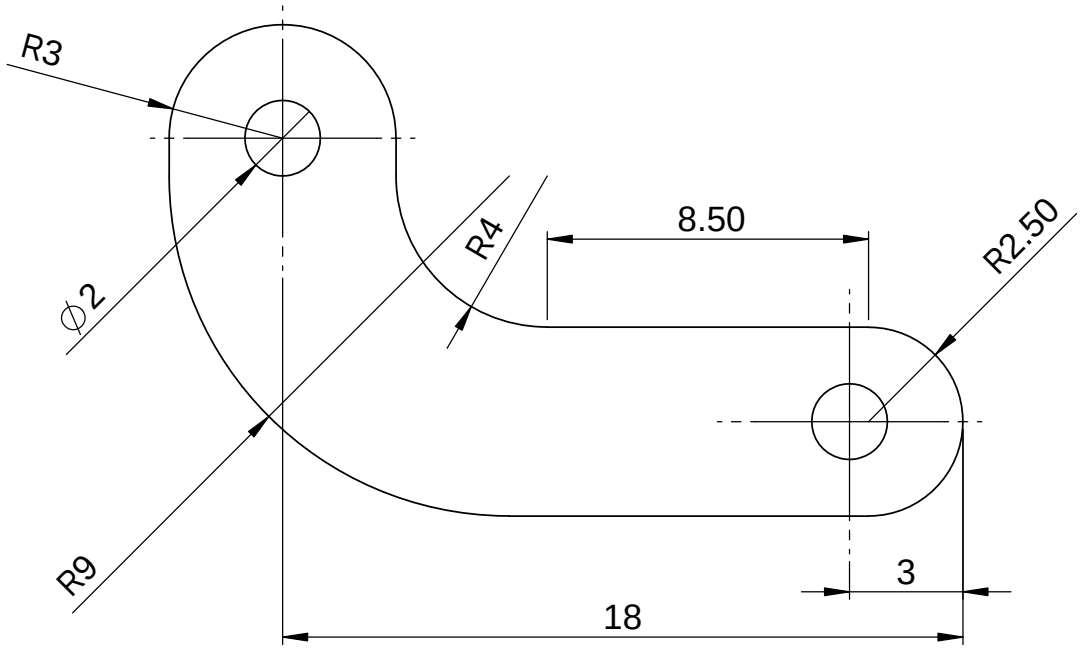
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

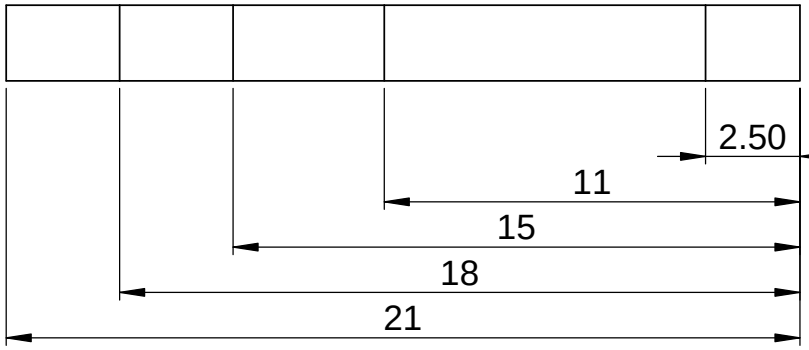
Acotación: [mm]

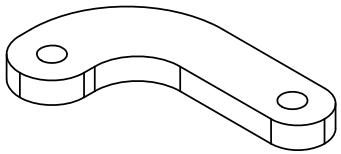
UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Base gripper

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Plástico	21 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 1	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape







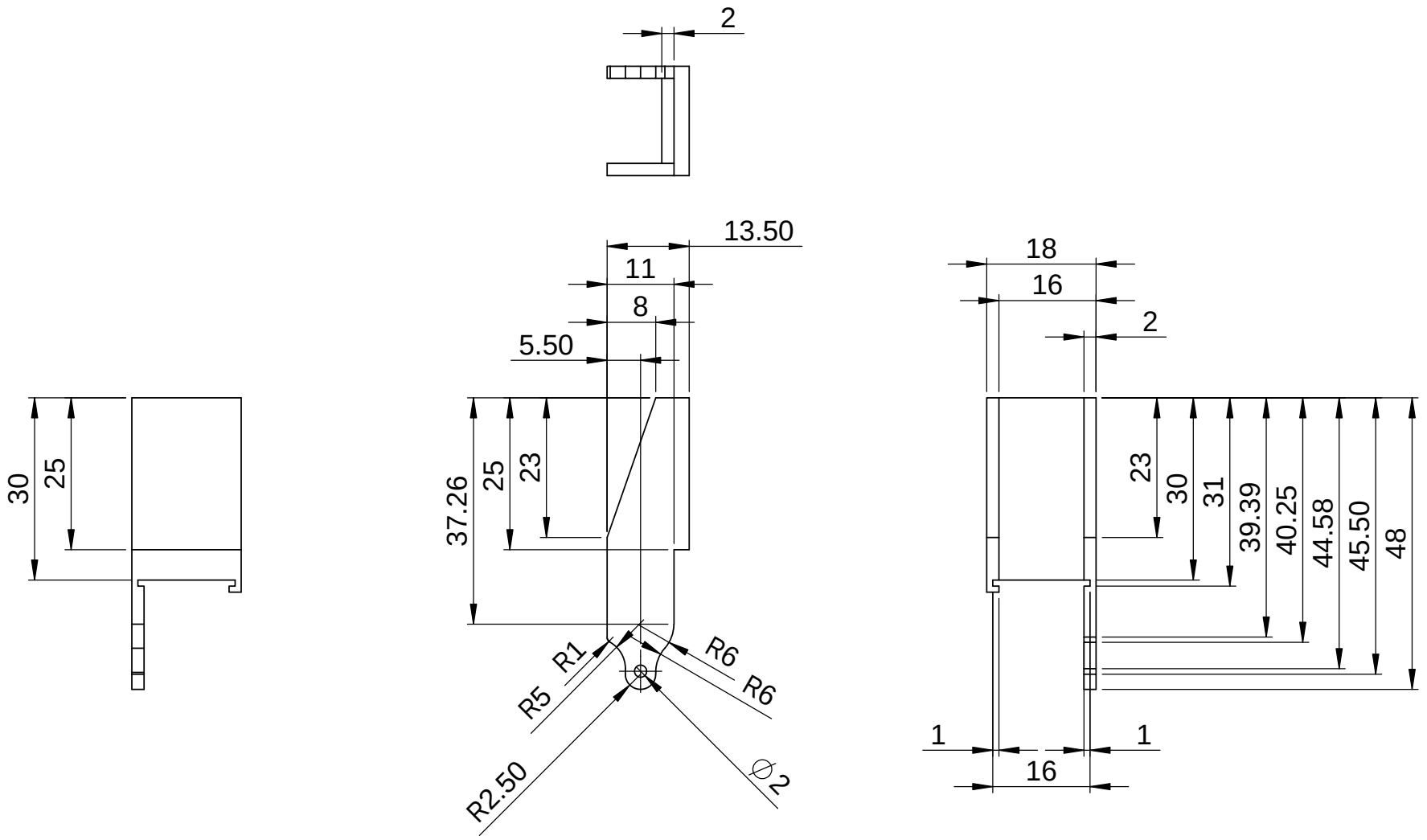
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

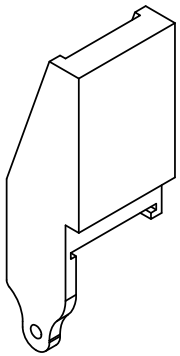
Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Sujecion gripper

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Plástico	22 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 2	ESCALA: 5:1	TOL: +/- 0.1 [mm] Landscape





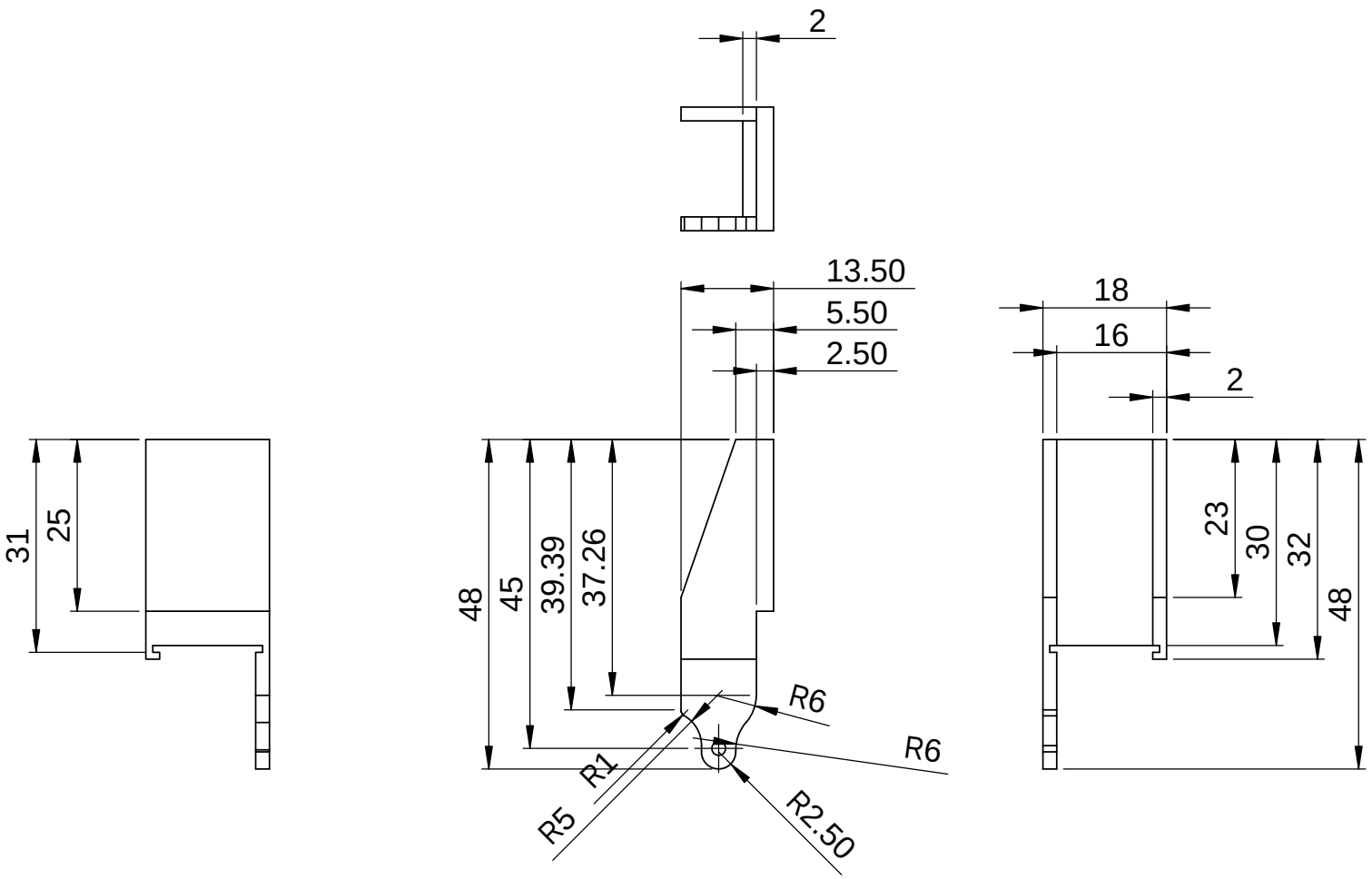
PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

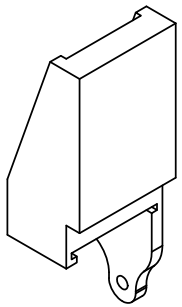
Acotación: [mm]

UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Gripper izquierdo

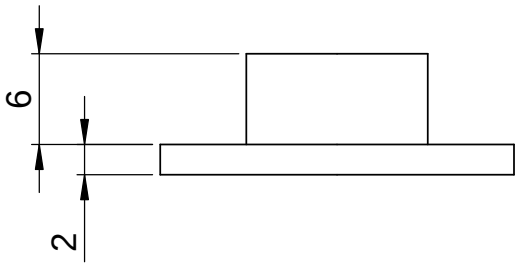
	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Plástico	23 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 1	ESCALA: 1:1 TOL: +/- 0.1 [mm]	Landscape

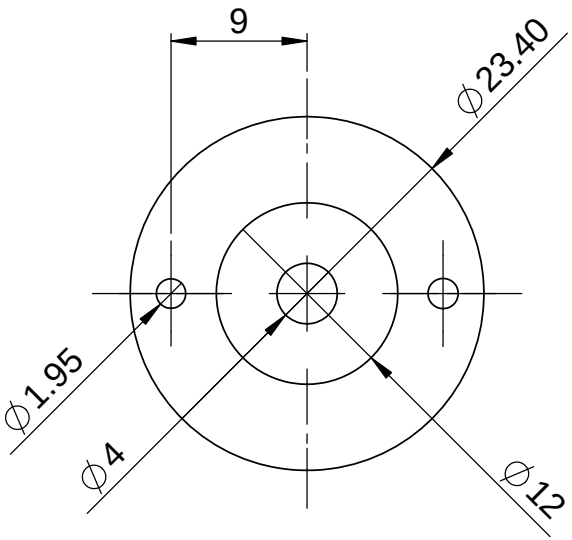


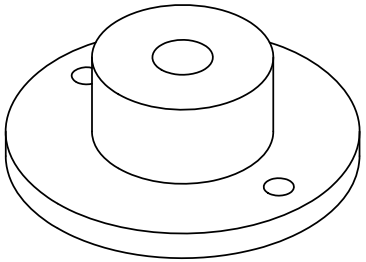


PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

Acotación: [mm]	UNAM POSGRADO Ingeniería MRG			TÍTULO: Gripper derecho	
	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	Formato: A (ANSI)
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Plastico	24 de 25	
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 1	ESCALA: 1:1	TOL: +/- 0.1 [mm] Landscape

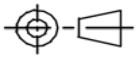






PROTOTIPO MANIPULADOR MÓVIL OMNIDIRECCIONAL REDUNDANTE

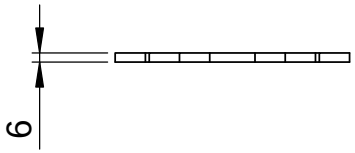
Acotación: [mm]

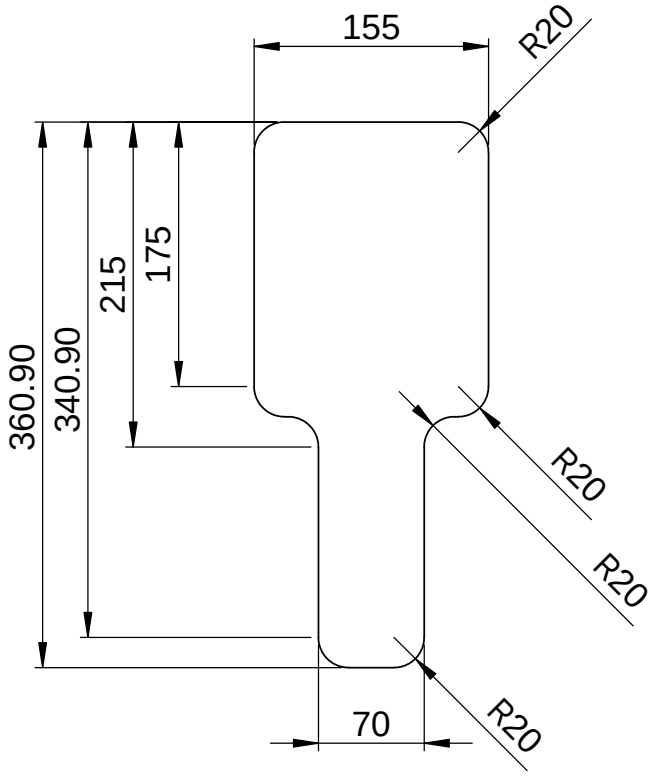


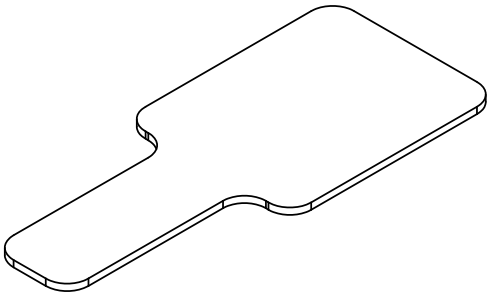
UNAM POSGRADO Ingeniería **MRG**

TÍTULO: Sujeción eje servo

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Plástico	25 de 25	Formato: A (ANSI)
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 6	ESCALA: 2:1 TOL: +/- 0.1 [mm]	Landscape

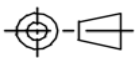






AMBIENTE INTELIGENTE

Acotación: [mm]



UNAM POSGRADO Ingeniería **MRG**

TÍTULO:

Mesa objetivo

	NOMBRE	FECHA	MATERIAL:	DIBUJO NÚMERO:	Formato: A (ANSI)
DIBUJÓ	Ing. Ignacio C. Cruz López	03/06/15	Acrílico	*1 de 1*	
VERIFICÓ	Ing. Ignacio C. Cruz López	03/06/15			
APROBÓ	Dr. Víctor J. González Villela	05/06/15	NÚMERO DE PIEZAS: 2	ESCALA: 1:5 TOL: +/- 0.1 [mm]	Landscape