UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Manipulador móvil omnidireccional, su coordinación de movimientos en ambientes inteligentes

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Irene Hernández Calderón

DIRECTOR DE TESIS

Dr. Víctor Javier González Villela

Ciudad Universitaria, CD.MX, año 2016

DEDICATORIAS

A mis padres por brindarme todo su amor, comprensión y apoyo, no sólo en mi formación profesional sino a lo largo de mi vida. Gracias por permitirme llegar hasta donde estoy.

A mis hermanas y hermano por su cariño, compañía y consejos.

A mi abuelita y mi tío. Los extraño.

A todos mis amigos, especialmente a Caro, por los momentos divertidos que pasamos a lo largo de la carrera.

AGRADECIMIENTOS

Quiero agradecer especialmente a mi profesor y director de tesis Dr. Víctor Javier González Villela por permitirme trabajar en el mejor grupo de investigación que he conocido, por guiarme, motivarme y apoyarme en todo momento para poder concluir el presente trabajo.

También al M.I. Noé Alfredo Martínez y al M.I. Erik Peña Medina por ser tan amables conmigo y brindarme su ayuda siempre que la necesité.

A la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería por la excelente formación profesional que me ha dado.

A mi amigo Pablo Ávila por motivarme y ayudarme en todo momento.

Agradezco en lo que le corresponde a la UNAM-DGAPA-PAPIIT por el apoyo brindado para la realización de este trabajo, a través del proyecto IN117617: “Robótica intuitiva, adaptable, reactiva, híbrida y móvil aplicada al servicio, el rescate y la medicina”.

RESUMEN

Este trabajo presenta el diseño, construcción, implementación y coordinación de un manipulador móvil omnidireccional enfocado a transportar materiales en un ambiente inteligente.

Dicho ambiente ha sido diseñado de tal modo que sea “consciente” del contexto en su interior, para lo cual utiliza una red de interconexiones entre los elementos que lo conforman, incluyendo al manipulador móvil omnidireccional.

Dentro del espacio, el robot funciona a modo de actuador y es el encargado de interactuar con los demás elementos del ambiente, con el fin de llevar a cabo la tarea.

Se realizó la coordinación del brazo y de la base con el fin de obtener un movimiento más natural a la hora del cumplimiento de la tarea. Lo anterior fue posible haciendo que la trayectoria del manipulador dependiera de la distancia entre la base y la meta.

ABSTRACT

This paper presents the design, construction, implementation and coordination of an omnidirectional mobile manipulator focused on transport materials in an intelligent environment.

This environment has been designed so that it is "aware" of the context inside, for which uses a network of interconnections between the elements that comprise it, including omnidirectional mobile manipulator.

Within the space, the robot functions as actuator and is responsible for interacting with the other elements of the environment, in order to carry out the task.

It is coordinating the arm and base in order to obtain a more natural time fulfilling the task performed movement. This was possible by the dependence of the trajectory of the manipulator to the distance between the base and the goal.

CONTENIDO

1 INTRODUCCIÓN

La competitividad global, la tendencia a disminuir costos y aumentar la eficiencia de producción han impulsado la creación de nuevos procesos de manufactura y fabricación, procesos que no pueden ser llevados a cabo por robots estacionarios, y que además demandan la realización de más de una tarea a la vez.

Este problema no sólo concierne a la industria, la reciente creación de ambientes “conscientes” del contexto desarrollado en su interior, ambientes actuados que sirven a sus habitantes, demanda el uso de actuadores robóticos de amplia movilidad, precisos, capaces de trasportar materiales y comunicarse con el ambiente y los elementos contenidos en él (incluyendo humanos y otros robots).

Los manipuladores móviles omnidireccionales son una respuesta a dichas demandas, puesto que suman las ventajas de los dos robots que lo conforman: la plataforma móvil omnidireccional y el brazo robótico. De este modo, la plataforma móvil omnidireccional extiende el espacio de trabajo del manipulador, permitiendo que opere en espacios a los que no podría acceder por sí mismo, mientras que la plataforma adquiere nuevas funciones de manipulación, como la manipulación de objetos, uso de herramientas y transporte de cargas. La base al ser de tipo omnidireccional, dota al robot de la capacidad

de hacer desplazamientos complicados, laterales, diagonales, giros de 360°, además de los típicos movimientos de delante y atrás

Con la creación de estos entes, surge un nuevo problema o campo de investigación: la coordinación entre base móvil y brazo manipulador. Varios autores han explorado este campo y propuesto soluciones a dicho problema. En el presente trabajo se desarrolla y ofrece otra solución, encaminada no sólo a la coordinación entre la base y el brazo, sino en procurar que el movimiento resultante de dicha coordinación sea lo más suave posible.

1.1 OBJETIVO

Diseñar, construir y coordinar un manipulador móvil omnidireccional para el trasporte de materiales en ambientes inteligentes.

1.2 HIPÓTESIS

Es posible diseñar, construir y programar un prototipo robótico sobre el cual pueda llevarse a cabo una coordinación entre su base y su manipulador para que dicho prototipo pueda transportar materiales en un ambiente inteligente.

2 ANTECEDENTES

2.1 ROBOTS MÓVILES

El desarrollo de robots móviles responde a la necesidad de extender el campo de aplicación de la Robótica, restringido inicialmente al alcance de una estructura mecánica anclada a uno de sus extremos. De los diferentes tipos de robots que existen, pueden ser identificadas cuatro grandes categorías [1]**.**

2.1.1 Robots terrestres

Los robots terrestres emplean diferentes tipos de locomoción mediante ruedas o patas, las cuales les confieren características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas útiles y maniobrabilidad. [2]

Los robots móviles pueden tener un gran número de posibles configuraciones y diseños cinemáticos. Cada tipo tiene sus pros y contras [3] dependiendo en qué aplicación sean empleados. Estas aplicaciones van desde el traslado de objetos, evasión de obstáculos, labores domésticas, coexistencia en ambientes cooperativos, hasta la de compañía y

diversión como lo es el robot tipo mascota AIBO de Sony (Figura 1) diseñado para ser versátil en sus movimientos.

Figura 1 AIBO, robot mascota de Sony.

2.1.2 Robots submarinos

Estos son operados en aguas superficiales o profundas. La mayoría de ellos emplean hélices o propulsores. Existen dos estructuras comunes: aquellas que utilizan un único propulsor que provee movimientos hacia adelante y atrás, pero tienen la desventaja de ser poco maniobrables, otras utilizan una colección de estos propulsores con control de dirección independiente, este arreglo ofrece mayor maniobrabilidad del robot, permitiéndole el cambio de orientación y posición [1].

2.1.3 Robots aéreos

Robots voladores como helicópteros robóticos, aviones de ala fija, drones y dirigibles. Estos vehículos son normalmente el resultado de la evolución de vehículos completamente teleoperados o autónomos, utilizados en recolección de datos o mantenimiento de instalaciones en entornos naturales a los que el acceso del hombre resulta muy difícil, o incluso imposible [4].

2.1.4 Robots espaciales

Estos robots son diseñados y construidos para operar en las condiciones de microgravedad presentes en el espacio exterior. Típicamente son empleados en estaciones espaciales para desempeñar tareas de reparación, construcción y mantenimiento.

2.2 MANIPULADORES

Un manipulador convencional o serial puede ser entendido como una cadena cinemática abierta, formada por un conjunto de eslabones o elementos de la cadena conectados o interrelacionados mediante articulaciones [2]. Existen además robots que no presentan esta cadena cinemática abierta, ya que tienen configuración de paralelogramo u otras estructuras cinemáticas cerradas; estos son conocidos como robots paralelos.

Generalmente los manipuladores poseen cinco o seis articulaciones, ya que el número de éstas aporta mayor maniobrabilidad, pero dificulta el problema de control, obteniéndose menores precisiones por acumulación de errores. Sin embargo, se puede emplear la multiarticulación en robots redundantes diseñados para la realización de tareas en áreas de difícil acceso, de entre estos cabe destacar los robots tipo serpiente [4].

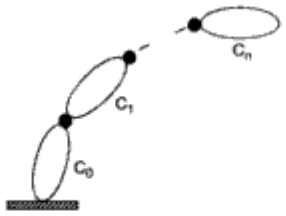


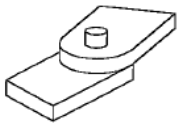
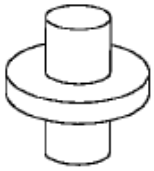
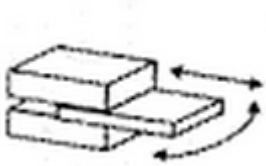
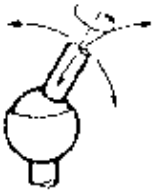
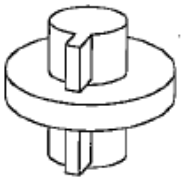
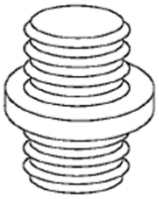
Figura 2 Cadena cinemática abierta [4].

2.2.1 Tipos de articulaciones

Las articulaciones, como ya se mencionaba, realizan la conexión entre los eslabones de los manipuladores y en general entre dos superficies, cuando el movimiento relativo entre estas es caracterizado por el deslizamiento de una sobre otra.

De entre las más utilizadas en la robótica destacan las mostradas en la Tabla 1.

Tabla 1 Tipos de articulaciones [4].

Tipo de articulación	Esquema	Grados de libertad	Descripción
Rotación		1	Rotación alrededor del eje de la articulación
Cilíndrica		2	Una rotación y una traslación
Planar		2	Caracterizada por el movimiento de desplazamiento en un plano
esférica		3	Combina tres giros en tres direcciones perpendiculares en el espacio
Prismática		1	El grado de libertad consiste en una traslación a lo largo del eje de la articulación
Tornillo		1	El movimiento se realiza a lo largo del eje de la articulación [5].

2.2.2 Espacio de trabajo de un manipulador

El espacio de trabajo de un manipulador está definido como el conjunto de todos los puntos que pueden ser alcanzados por su efector final [6] tras moverse hacia adelante, atrás, arriba y abajo. El alcance de estos puntos está determinado por la longitud del brazo del robot y el diseño de sus ejes. Cada eje aporta su propio rango de movimiento [7].

Si un punto en el espacio puede ser alcanzado sólo con una orientación específica del manipulador, se dice que la manipulabilidad del efector final es muy pobre y no es posible realizar ningún trabajo práctico satisfactoriamente. El espacio de trabajo en el cual el efector final puede alcanzar cualquier punto en cualquier orientación es llamado espacio de trabajo diestro (DWS, por sus siglas en ingles). Cabe mencionar que el espacio diestro puede ser más pequeño o el mismo que el espacio alcanzable [3].

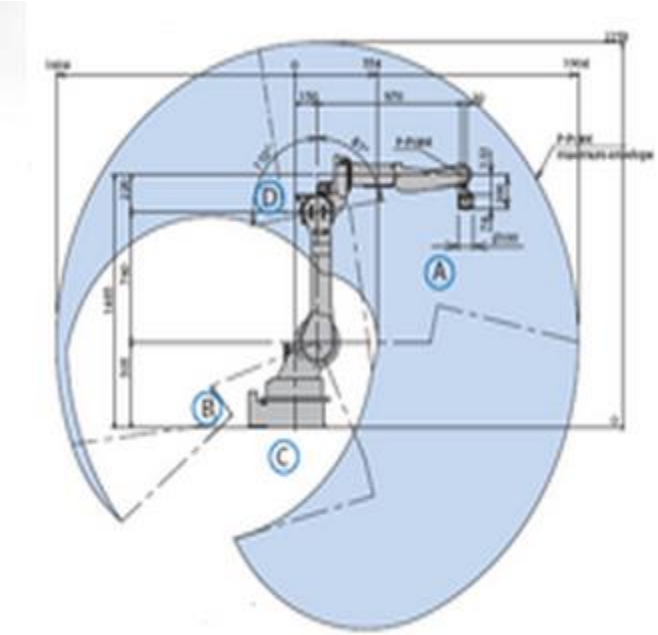


Figura 3 Espacio de trabajo de un manipulador serial [8].

2.2.3 Tipos de configuraciones

Cuando se habla de la configuración de un robot, se refiere a la forma física que se le ha dado al brazo del robot. El brazo del manipulador puede presentar hasta 36 diferentes configuraciones, de las cuales sólo doce son funcionalmente diferentes. En el mercado se encuentran cinco configuraciones clásicas: la antropomórfica, cartesiana, cilíndrica, polar y angular, las cuales serán explicadas a continuación [3].

Robot antropomórfico

Esta configuración se caracteriza por tener todas sus articulaciones de tipo rotacional. Por similitud con el brazo humano, la segunda articulación se conoce como hombro y la tercera como codo (Figura 4). El espacio de trabajo de la configuración antropomórfica corresponde a una esfera hueca cuyo radio es igual a la suma de longitudes de sus eslabones [8].

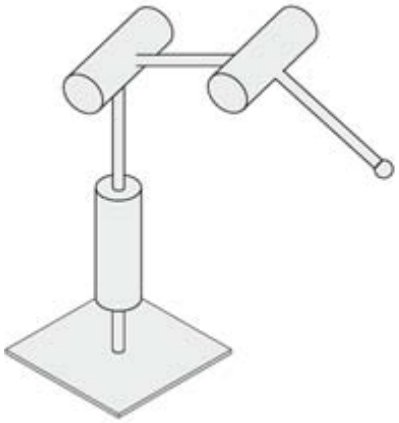


Figura 4 Configuración antropomórfica [7].

Configuración cartesiana

Tiene tres articulaciones prismáticas, y por tanto puede realizar tres movimientos lineales. Los ejes asociados a cada articulación son mutuamente perpendiculares entre sí, tal y como se observa en la Figura 5. El espacio de trabajo de este manipulador es un paralelepípedo rectangular como el que se muestra en la Figura 6 y su estructura mecánica presenta baja destreza debido al tipo de articulaciones que lo conforman.

Entre las aplicaciones de los robots manipuladores en esta configuración se encuentran aquellas que procesan cavidades horizontales y trasporte de objetos. [2].

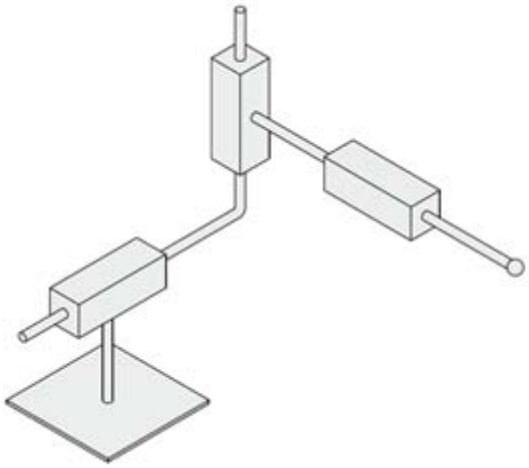


Figura 5 Configuración cartesiana [7].

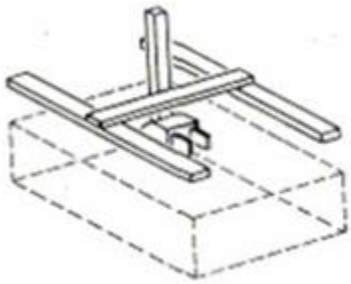


Figura 6 Espacio de trabajo de un robot cartesiano.

Configuración cilíndrica

La configuración cilíndrica, Figura 7, tiene una articulación del tipo rotacional en la base, mientras que la segunda y tercera articulaciones son prismáticas. Su estructura mecánica es compleja y su espacio de trabajo es la porción de un cilindro hueco (Figura 8).

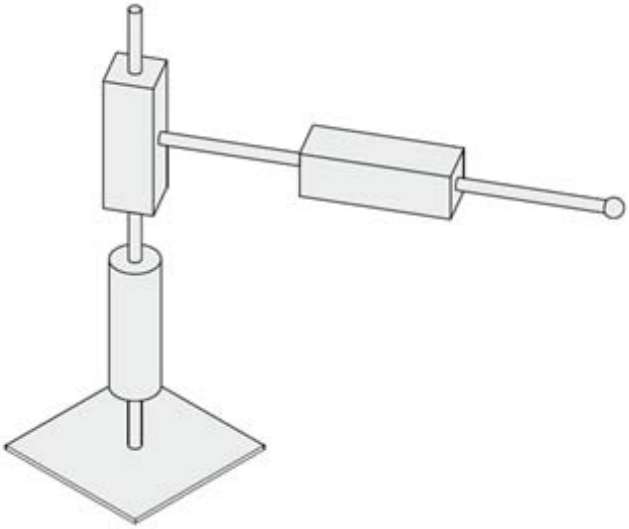


Figura 7 Configuración cilíndrica [7].

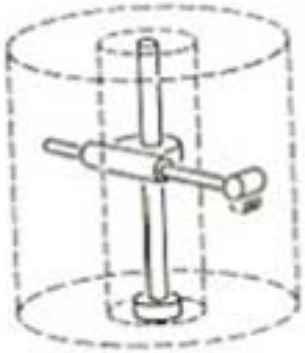


Figura 8 Espacio de trabajo de un robot cilíndrico.

Configuración esférica

Esta configuración se caracteriza por tener dos articulaciones de rotación y una prismática.

Ejemplos de esta configuración de robots manipuladores es el robot Stanford, cuya principal aplicación se encuentra en el mecanizado de piezas automotrices y la manipulación de objetos en el piso. El espacio de trabajo de esta configuración es una esfera hueca, como se aprecia en la (Figura 9).

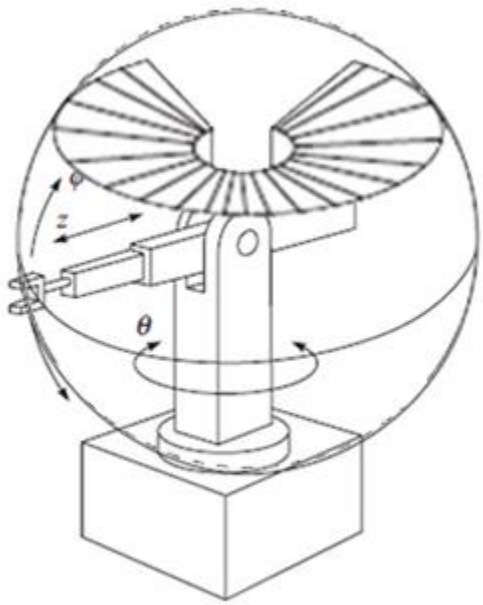


Figura 9. Configuración esférica y espacio de trabajo.

Configuración SCARA

Una geometría especial de los robots industriales es el tipo SCARA (Selective Compliant Assembly Robot Arm) mostrado en la Figura 10. Consiste es un brazo planar antropomórfico de dos articulaciones rotacionales para la base y hombro, respectivamente, que se mueve en un plano horizontal; la tercera articulación es prismática. Para este tipo de configuración todos los ejes de movimiento z_1, z_2 y z_3 son paralelos entre sí. La estructura mecánica es de alta rigidez para soportar cargas en forma vertical y para control de fuerza en el plano horizontal, por lo que la configuración SCARA es adecuada para tareas de ensamble con pequeños objetos. Su espacio de trabajo se describe en la Figura 11, [6].

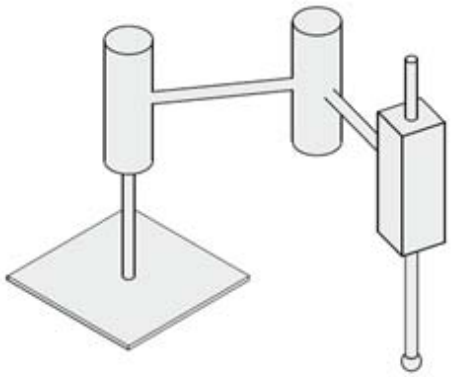


Figura 10 Configuración SCARA.

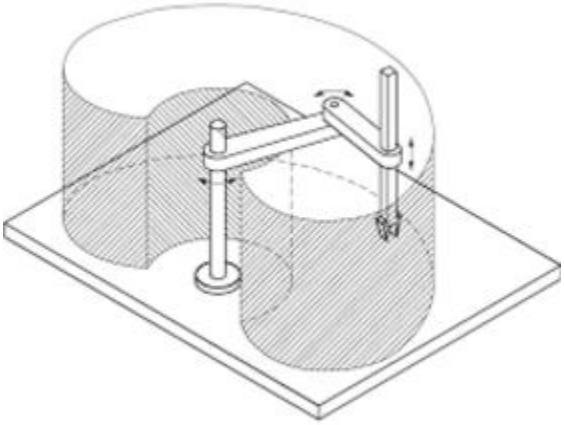


Figura 11 Espacio de trabajo, brazo SCARA.

2.3 LA ROBÓTICA EN AMBIENTES INTELIGENTES

Un ambiente inteligente es aquel espacio capaz de reconocer los elementos que interactúan en él y el contexto en que se desarrolla esta interacción, lo logra mediante una serie de capas, descritas en la Tabla 2, que le proporcionan y le comunican información no sólo a él, sino a y los elementos que lo conforman para que se lleven a cabo de manera inteligente tareas bien definidas.

Tabla 2 Capas del espacio inteligente.

Capas del espacio inteligente
Capa pasiva de percepción En esta capa se encuentran los dispositivos destinados a recabar información sobre lo que sucede en el ambiente. Algunos ejemplos son: sensores, cámaras, micrófonos, etc.
Capa de interacción La componen los elementos que pueden controlarse y utilizarse a modo de actuadores, con el fin de interactuar físicamente con el medio para modificarlo.
Capa de comunicaciones Integrada por la red de interconexiones que comunican entre sí a los distintos componentes del ambiente inteligente.

Capa de inteligencia

Es la parte encargada de procesar la información lograda por la capa de percepción para controlar a los elementos que integran la capa de interacción, mediante la red de intercomunicaciones, con el fin de realizar tareas específicas [9].

La incorporación de robots móviles y espacios inteligentes a la industria y la vida diaria aún es una disciplina emergente, en la que no se han explotado en su totalidad las posibilidades que ofrece. Anteriormente, los robots sólo eran capaces de desarrollar un comportamiento reactivo ante el ambiente y los eventos sucedidos en él, lo cual limitaba claramente su utilidad. Para extender la funcionalidad de estos es necesario hacerlos conscientes de su posición en el ambiente, así como también de la posición e intención de otros agentes [10].

A pesar de que el tema es relativamente nuevo, son numerosas las aplicaciones de los ambientes inteligentes. Abarcan todos los ámbitos de la robótica de servicio, tareas de seguridad y vigilancia automática, ayuda a personas con discapacidad y por supuesto servicios domóticos. El concepto de ambiente inteligente ha sido explorado por distintos autores, y es considerado en su mayoría como espacio habitado por una variedad de agentes, de entre los cuales destacan robots, sistemas embebidos, aplicaciones móviles y humanos, quienes en conjunto conviven, coexisten y colaboran para formar dicho ambiente. En la Figura 12 se muestran algunos de los elementos antes mencionados.

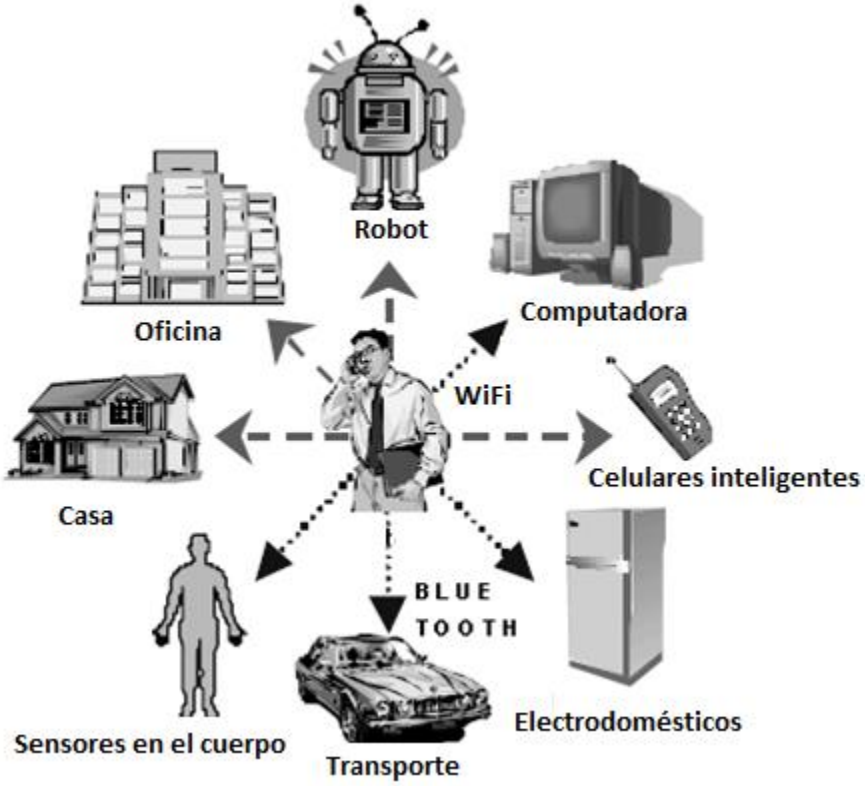


Figura 12 Elementos del ambiente inteligente.

En [10], se reporta del desarrollo de un robot doméstico capaz de interactuar con humanos mediante el habla y realimentación emocional, lo consigue por medio de una cabeza robótica, actuada por 17 servomotores, capaz de articular tres de las seis expresiones faciales emocionales básicas, Figura 13. El robot no sólo es consciente de su posición en el ambiente, sino también de la posición e intenciones de los usuarios.

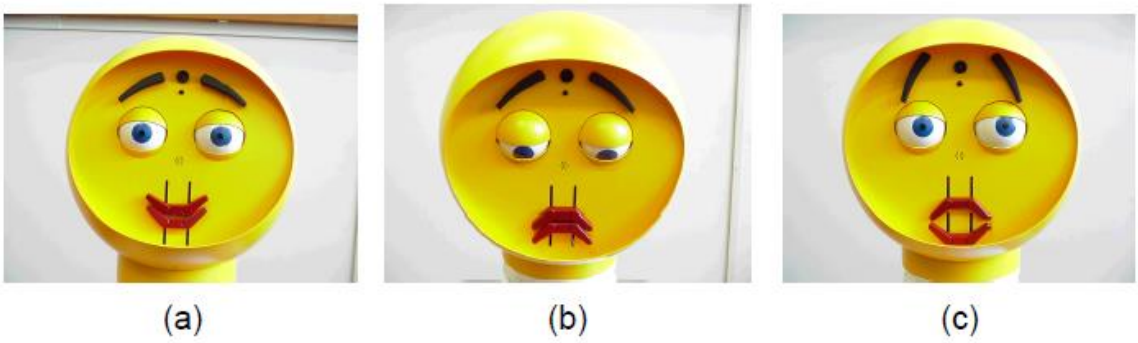


Figura 13 Cabeza robótica expresando: a) felicidad, b) tristeza y c) sorpresa.

En [11] se desarrolla el sistema AuRA (Autonomous Robot Architecture) enfocado a la creación de robots cosmopolitas dotados de herramientas que les permitan interactuar tanto en interiores como en exteriores, de ahí que se les califique de este modo. Para la planificación de la trayectoria global, AuRA utiliza conocimiento cartográfico alojado a modo de memoria a largo plazo en su sistema, además tiene programado conocimiento a priori para guiar las expectativas de los sensores. Como memoria a corto plazo emplea mapas de espacios cercanos que representan el mundo percibido en el momento actual.

Figura 14 Robots para experimentación dotados del sistema AuRA.

En [12] se describe un algoritmo que permite fusionar los datos proporcionados por un arreglo de cámaras de vídeo para crear un mapa de ocupación espacial-temporal. Se pretende que la disposición de cámaras se asemeje a una red de video de seguridad. El mapa de ocupación es una imagen de trama bidimensional, distribuida uniformemente en el suelo plano. Cada píxel del mapa contiene un valor binario, lo que significa que la superficie útil designada está vacía u ocupada.

El algoritmo emplea una tabla de consulta para determinar el efecto de cada píxel en el mapa. Esta brevedad de operaciones permite que el mapa de ocupación sea actualizado en tiempo real demostrando que es capaz de operar en escenarios dinámicos.

El trabajo de tesis [13] diseña y desarrolla un espacio inteligente entendido como un área física dotada de sensores controlados e inspeccionados por un sistema supervisor con capacidad de análisis y toma de decisiones.

La inteligencia del entorno es lograda a partir de varios nodos interconectados que permiten la captura y el análisis de la información extraída de los sensores. El entorno dispone de controladores de entre los cuales destacan robots móviles, como los mostrados

en la Figura 15, que son empleados para llevar a cabo tareas concretas como el seguimiento de trayectorias.

Figura 15 Robots para experimentación.

2.4 MANIPULADORES MÓVILES OMNIDIRECCIONALES

El manipulador móvil omnidireccional es, como su nombre lo indica, un manipulador montado sobre una plataforma móvil de tipo omnidireccional.

Los manipuladores móviles omnidireccionales suman las ventajas de las partes que lo conforman, presentando así mejoras sobre estas partes si se analizaran por separado, si se analizaran por separado. La base móvil, por ejemplo, incrementa el espacio de trabajo del manipulador que está montado sobre ella, y éste a su vez proporciona manipulabilidad a la base [14]. La redundancia que el sistema base-manipulador presenta permite emplear los grados de libertad redundantes en la realización de tareas secundarias [15]. La base al ser del tipo omnidireccional, dota al robot de la capacidad de hacer desplazamientos complicados, laterales, diagonales, giros de 360°, y los típicos movimientos de adelante y atrás [8].

Los Robots omnidireccionales que son operados en un espacio cartesiano de tres dimensiones deben tener tres grados de libertad, los cuales son dos grados de movimiento traslacional, uno a lo largo del eje *x* y otro sobre el eje *y* y un grado de movimiento rotacional [16].

2.4.1 Trabajos previos

Los manipuladores móviles omnidireccionales han llamado mucho la atención en las últimas décadas, debido a su gran escala de movilidad y su diestra manipulabilidad.

Mientras que este tipo de manipuladores móviles ofrecen muchas ventajas, su modelado y control representa una amplia gama de retos, debido a su complejidad [17]. Estos han sido enfrentados por distintos autores de varias formas. A continuación se reporta la revisión de algunos de estos trabajos.

En [18] se desarrolla un método basado en el concepto de campos potenciales para ser empleado en robots con el fin de que estos evadan obstáculos. La utilización de dicho concepto es extendida también hacia manipuladores y manipuladores móviles en los cuales su ejecución toma en cuenta los campos de atracción y repulsión representados por puntos meta y obstáculos (Figura 16), respectivamente. Fuera del campo de repulsión de los obstáculos (que son móviles) el efector final será orientado en línea recta y el robot se moverá a su máxima velocidad.

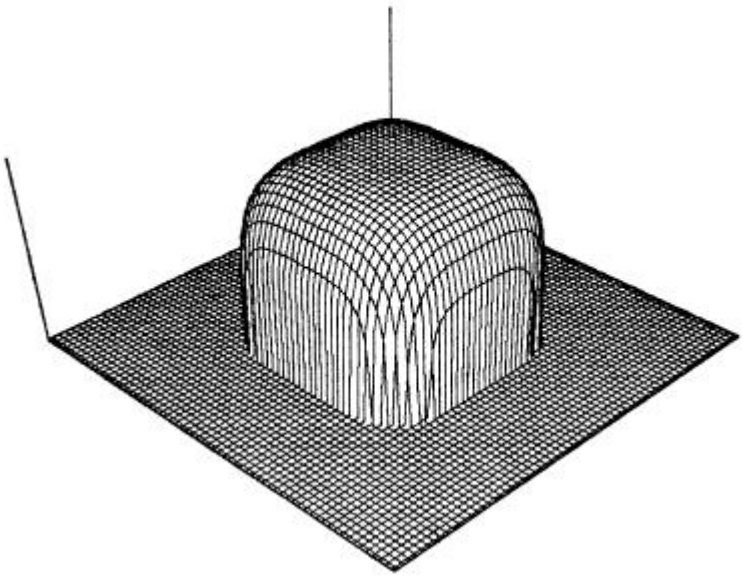


Figura 16 Espacio de atracción o repulsión de obstáculos y puntos meta.

En [14] se considera un manipulador móvil constituido por una plataforma de tres ruedas ensambladas de forma lateral y un manipulador de tres articulaciones rotacionales montado sobre el centro de gravedad de la base.

En primer lugar, se presenta el modelo cinemático para el manipulador móvil y posteriormente se determina el modelo dinámico, en el cual se toman simultáneamente las características tanto del manipulador como de la base, finalmente se realiza un control de par por computadora. Las pruebas reportadas y las conclusiones dan muestra de que el modelo empleado es capaz de mantener cualquier posición y orientación del efector final por tiempo indefinido.

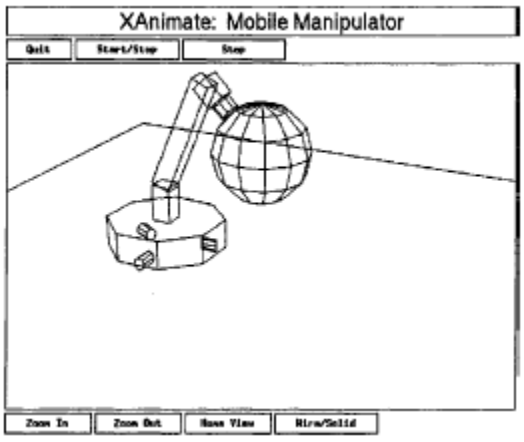


Figura 17 Manipulador móvil con ruedas ensambladas de forma lateral.

El manipulador móvil omnidireccional redundante mostrado en [19] se puede dividir en dos partes para su descripción. La primera consiste en una plataforma móvil compuesta por cuatro robots móviles diferenciales idénticos entre sí y la segunda por un brazo manipulador serial de cinco grados de libertad.

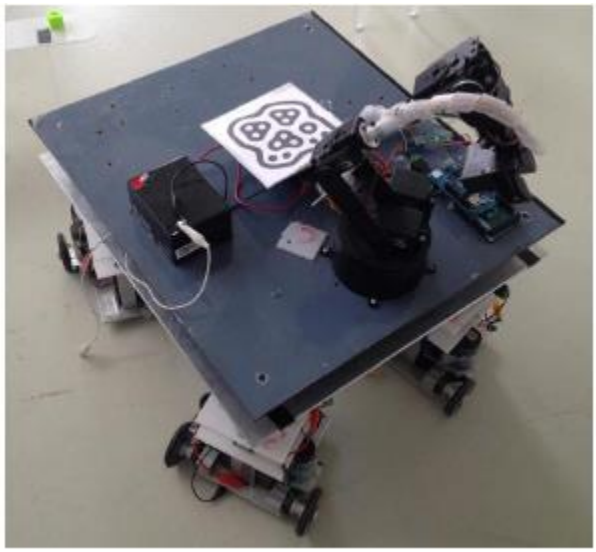


Figura 18 Robot móvil omnidireccional redundante.

2.4.2 Clasificación de los manipuladores móviles omnidireccionales

De la gama de manipuladores móviles, este trabajo se centra en aquellos que poseen plataformas omnidireccionales, por ser el tema de estudio.

Los manipuladores móviles omnidireccionales se pueden clasificar de acuerdo al tipo de manipulador y configuración de la plataforma móvil. El manipulador puede presentar alguna de las configuraciones mostradas en el capítulo 2.2.3.

El carácter omnidireccional de los manipuladores móviles, como ya se mencionó, está dado por la plataforma, las ruedas y la configuración que en ambas se utilicen.

Actualmente no existe en la literatura una clasificación como tal para los manipuladores móviles omnidireccionales. Pero de los trabajos previos a esta tesis, se pueden identificar algunas similitudes entre ellos, las cuales se reportan a continuación.

Por tipo de rueda

Las ruedas estándares, estriadas, lisas o de castor, no permiten por sí solas el movimiento omnidireccional, estas deben de ser complementadas con alguna configuración plataforma-ruedas que asegure este tipo de movimiento.

Existen dos tipos de ruedas que sí lo hacen, estas son las ruedas omnidireccionales y las esféricas. Este tipo de configuración de ruedas no impone restricciones cinemáticas en el chasis del robot, es decir, que el robot puede cambiar su dirección libremente en cualquier instante [39].

Las ruedas omnidireccionales están constituidas por una rueda estándar dotada de una corona de rodillos (Figura 19), cuyos ejes de giro son perpendiculares a la dirección normal de avance, por sí mismas tienen los tres grados de libertad que hay en el plano. Una variante de esta son las ruedas Mecanum o suecas en las cuales sus rodillos presentan un ángulo de 45° si son del tipo B, y de 135° si son del tipo A.

Figura 19 Rueda omnidireccional (izquierda) y rueda Mecanum (derecha).

Un ejemplo de aplicación de estas ruedas es el mostrado en la siguiente figura, cuya configuración omnidireccional está basada en tres ruedas omnidireccionales cada una actuada por un motor.

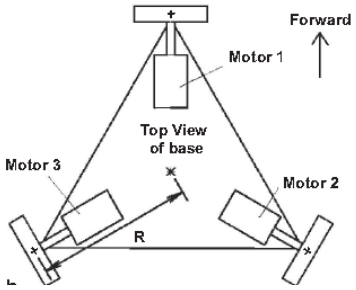


Figura 20 Plataforma con tres ruedas omnidireccionales.

Por su parte, las ruedas esféricas, Figura 21, no ponen ninguna restricción cinemática a la plataforma ya que no poseen un eje principal de rotación. La rueda esférica omnidireccional puede tener cualquier dirección arbitraria de movimiento. Sin embargo, este tipo de ruedas es estrictamente usado para aplicaciones en espacios interiores con pisos o superficies de deslizamiento regulares u homogéneas [49].

Figura 21 Rueda esférica.

En las plataformas que no utilizan ruedas omnidireccionales destacan otros tipos como son las ruedas con estrías y lisas (Figura 22) y las de tipo oruga, en cuyos casos debe existir una configuración especial entre ellas, la plataforma y otros elementos como engranes para que el robot adquiera la cualidad de omnidireccional.

Figura 22 Ruedas lisas (izquierda) y estriadas (derecha).

Por tipo de base

Se observó durante la revisión bibliográfica y hemerográfica, que los robots que no emplean ruedas omnidireccionales tienden a poseer una base de complejidad mayor ante aquellos que si las usan, es por esto que existen bases de distintas tipologías formadas a partir de mecanismos y de varias disposiciones de ruedas que conllevan a que la base sea del tipo omnidireccional. Configuraciones como [1] y [4] utilizan un ensamble de tres ruedas colocadas a una distancia igual desde el centro de gravedad de la plataforma y con 2/3π radianes de distancia entre los ensambles.

Esta configuración posee dos desventajas importantes, la primera es que el control de las ecuaciones cinemáticas es complejo debido a que los movimientos rotacionales y de traslación manejados por tres actuadores no son desacoplados, lo cual necesita trasformaciones complejas para controlar cada uno de los grados de libertad en el plano cartesiano. Y la segunda es que este tipo de robots hacen contacto con el suelo sólo en tres puntos, lo cual podría afectar su estabilidad si la posición de su centro de gravedad cambia.

Un ejemplo de manipulador móvil omnidireccional que no presenta inestabilidad en su base es el mostrado en [19], ya que posee una plataforma rectangular central unida en sus esquinas a cuatro robots móviles diferenciales.

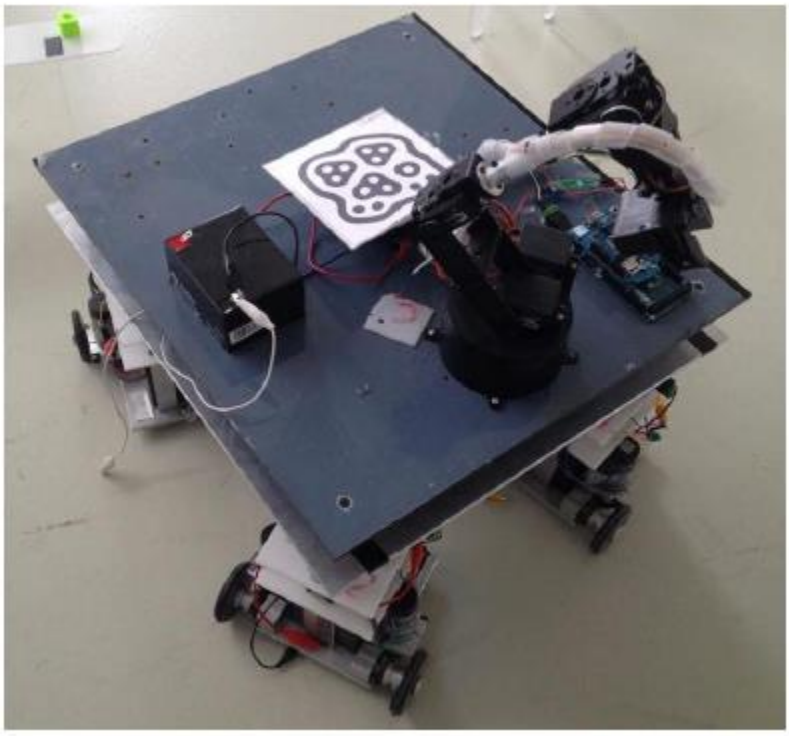


Figura 23 Manipulador móvil omnidireccional de plataforma formada por robots diferenciales.

3 TAREA A REALIZAR

Se pretende que el manipulador móvil omnidireccional (MMO) realice la tarea de trasportar materiales en un ambiente inteligente, esta debe ser llevada a cabo entre la base y el brazo de manera coordinada de modo que se obtenga un movimiento suave y sincronizado por ambas partes sin la necesidad de que la capa de inteligencia del ambiente realice cálculos exhaustivos. Para cumplir lo último se utilizaron patrones de identificación visual Fiduciales, una cámara y un método de coordinación que próximamente se explicará.

El ambiente inteligente fue diseñado construido y acondicionado en el departamento del Mechatronics Research Group ubicado en el tercer piso del Centro de Ingeniería Avanzada en la Facultad de Ingeniería de la UNAM, está constituido por una laptop que cuenta con el software MATLAB R2015a®, el manipulador móvil omnidireccional, mesas objetivo, y una cámara LifeCam Studio colocada a 2.5 m de piso cuyo alcance de visión proyectado sobre el suelo genera un rectángulo de 2.47 m por 1.85 m (Figura 24).

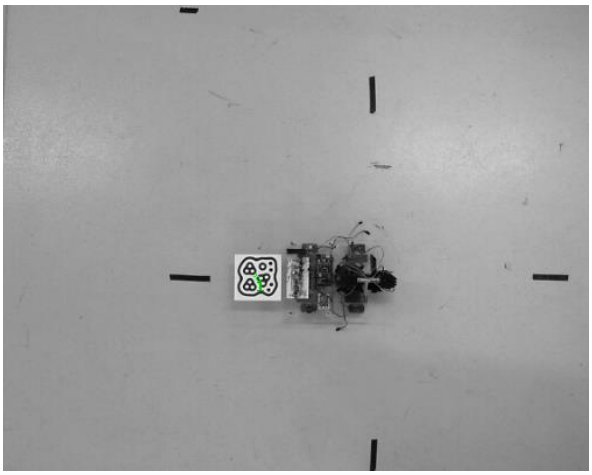


Figura 24 Espacio de trabajo.

Para realizar la tarea se plantea colocar un objeto en una mesa provista de una marca patrón que la identifique como tal, de modo que se conozca su posición y orientación en todo momento. Puede ser colocada en cualquier lugar del espacio inteligente y en cualquier posición, lo mismo para el MMO.

El MMO partirá de cualquier posición y orientación hacia la mesa objeto posicionando desde el principio el manipulador con el fin de que el MMO llegue a la meta siguiendo la trayectoria determinada, cuya obtención se explica en el capítulo 4.1.1.

La distribución descrita puede ser apreciada en la Figura 25, donde se observan el espacio de trabajo, su sistema de referencia, la mesa objetivo, el MMO y un elemento pasivo que simula un obstáculo o elemento que podría distraer al sistema; proyectado a un ambiente real, este elemento podría representar algún mueble o columna y en general cualquier objeto inmóvil presente en el ambiente.

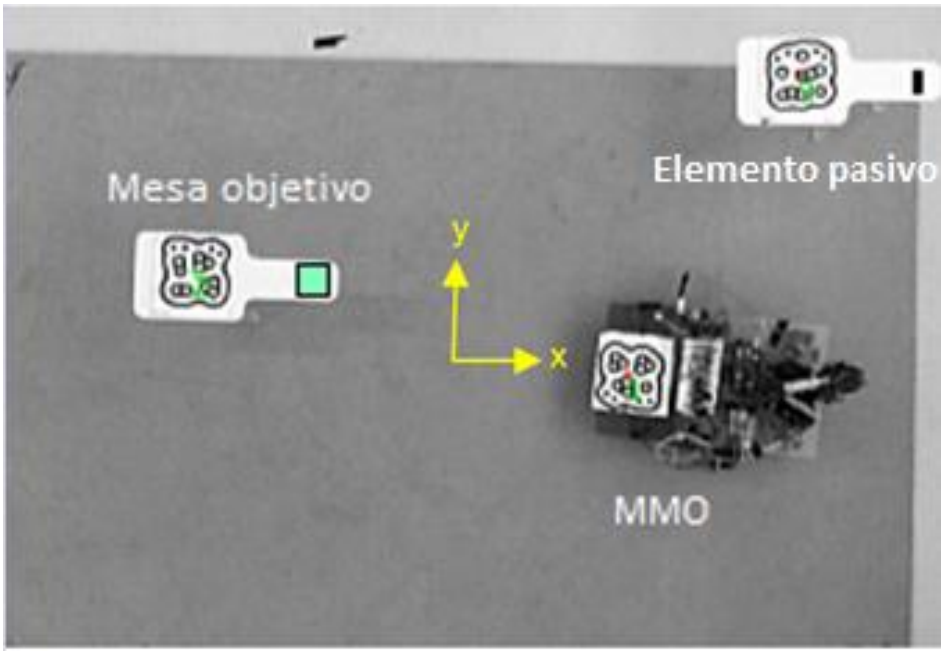


Figura 25 Elementos en el espacio inteligente.

4 GENERACIÓN DE MOVIMIENTO

A continuación, se muestra el análisis cinemático realizado tanto para el brazo manipulador como para la base omnidireccional con el fin de obtener las ecuaciones que describan ambos sistemas en la generación de movimiento. Estas ecuaciones serán implementadas en el capítulo 6.

4.1 MODELO CINEMÁTICO DEL MANIPULADOR

4.1.1 Cinemática directa

En el análisis cinemático del brazo manipulador, se debe realizar un estudio analítico del movimiento del robot con respecto a un sistema de referencia cartesiano fijo $\{x_0, y_0, z_0\}$, en el cual se busca encontrar la función vectorial $f_R(\boldsymbol{l_i}, \boldsymbol{q})$ que relacione la dependencia existente entre las coordenadas articulares y los parámetros geométricos $\boldsymbol{l_i}$ (longitudes de los eslabones), con las coordenadas cartesianas $[x, y, z]^T$ y la orientación del efector final del robot $[\theta, \phi, \psi]^T$.

$$\begin{bmatrix} x \\ y \\ z \\ \theta \\ \phi \\ \psi \end{bmatrix} = f_R(\boldsymbol{l_i}, \boldsymbol{q})$$

El manipulador robótico a analizar es una cadena cinemática abierta formada por cuatro cuerpos rígidos más el elemento terminal, estos elementos están acoplados por juntas de tipo rotacional y cuyo campo de acción es el plano XYZ [20].

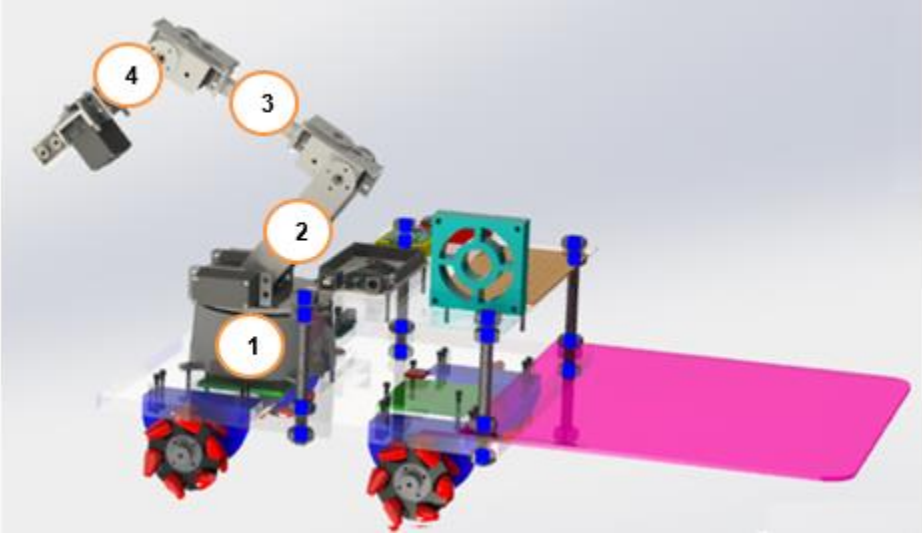


Figura 26 Cuerpos rígidos que forman al manipulador.

Para resolver el problema de la cinemática directa se utilizará el Método general, el cual emplea matrices de transformación que relacionan sistemas de referencia. El manipulador puede describirse definiendo cuatro magnitudes asociadas a cada articulación, la articulación es una variable y las tres restantes son parámetros fijos, utilizando la representación de Denavit-Hartemberg (D-H), se define lo siguiente [21]:

1. Z_i es el eje del $i - ésimo$ par, se elige como:

- El eje de rotación, si el par asociado es rotacional(R).
- La dirección de traslación si el par asociado es prismático (P).

2. X_i se define como la perpendicular común a Z_{i-1} y Z_i, dirigida de la primera a la segunda. Notando que si estos dos ejes se intersectan, la dirección positiva de X_i este indefinida y, por lo tanto, puede ser libremente asignado.

3. La distancia entre Z_i y Z_{i+1} se define como a_i.
4. La coordenada en Z_i de la intersección de Z_i con X_{i+1} se denota por b_i . También puede definirse como la distancia entre X_i y X_{i+1}.
5. El ángulo entre Z_i y Z_{i+1} se define como α_i y es medida respecto a la dirección positiva de X_{i+1}. Este parámetro es conocido como, el ángulo de giro entre los ejes pares sucesivos.
6. El ángulo entre X_i y X_{i+1} se define como θ_i y es medido respecto a la dirección positiva de Z_i.

Dichos parámetros se pueden observar en la Figura 27 y en la Tabla 3:

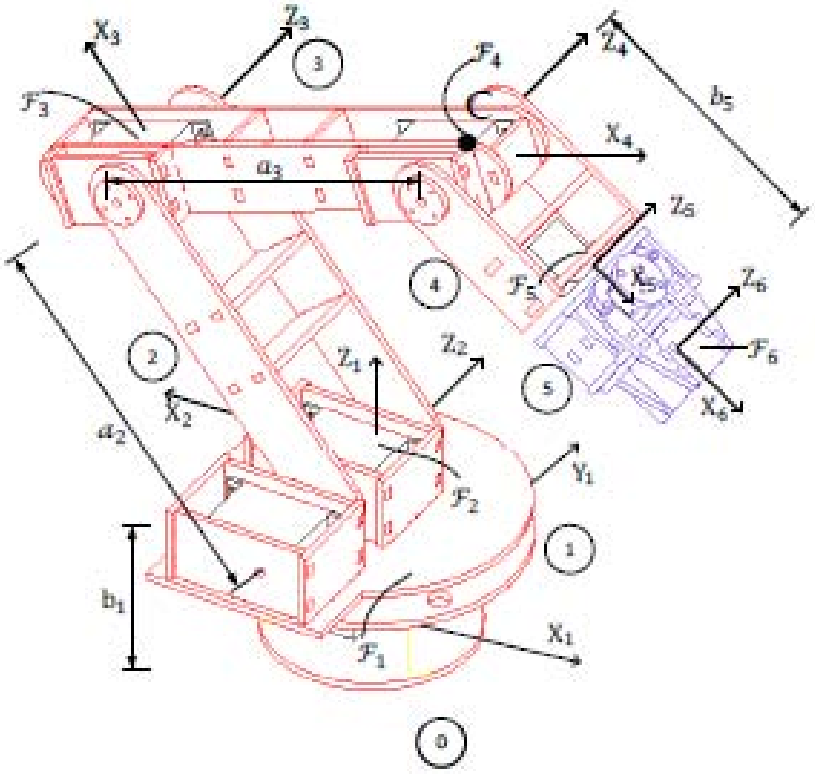


Figura 27. Parámetros D-H.

Tabla 3. Parámetros D-H

i	α_{i-1}[rad]	a_{i-1}[cm]	b_i[cm]	θ_i[rad]
1	$\frac{\pi}{2}$	0	13	θ_1
2	0	9.2	0	θ_2
3	0	14.3	0	θ_3
4	$\frac{\pi}{2}$	0	0	θ_4
5	0	0	12	θ_5

Matrices de trasformación homogénea

En la robótica es necesario poder describir de forma conveniente las posiciones y orientaciones de los objetos en el espacio, en el manipulador se logrará mediante la introducción de las transformaciones homogéneas, así como de los cambios de sistemas de referencia.

Las transformaciones consisten en una sucesión de rotaciones y traslaciones que permitan relacionar el sistema de referencia del elemento i con el sistema de referencia del elemento i-1.

Matriz de trasformación homogénea.

$$T = \begin{bmatrix} Matriz\ de\ rotación & matriz\ de\ traslación \\ 0 & 1 \end{bmatrix}$$

Ecuación 1

Las trasformaciones en cuestión son las siguientes:

Giro del ángulo alfa (α) alrededor del eje x

$$R_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\text{sen}\alpha & 0 \\ 0 & \text{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 2

Traslación de una distancia a lo largo del eje x

$$t_x = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3.

Giro del ángulo theta (θ) alrededor del eje z.

$$R_{z,\theta} = \begin{bmatrix} \cos\theta & -\text{sen}\theta & 0 & 0 \\ \text{sen}\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 4

Traslación de una distancia b a lo largo del eje z.

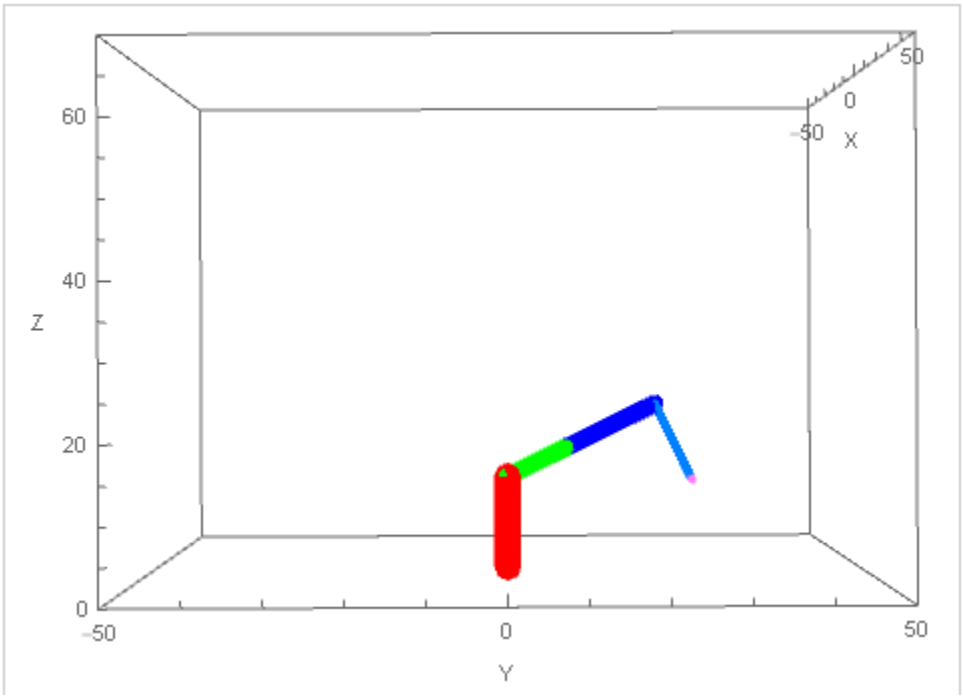
$$t_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 5

$$T = \begin{bmatrix} \text{Cos}\theta & -\text{Cos}\alpha\text{Sin}\theta & \text{Sin}\alpha\text{Sin}\theta & a\text{Cos}\theta \\ \text{Sin}\theta & \text{Cos}\alpha\text{Cos}\theta & -\text{Cos}\theta\text{Sin}\alpha & a\text{Sin}\theta \\ 0 & \text{Sin}\alpha & \text{Cos}\alpha & b \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 6

Dichas matrices fueron programadas en el software Mathematica con el fin de describir el manipulador en su totalidad obteniéndose así el vector (que parte del primer sistema coordenado, definido en la base del brazo, hasta el punto de interés para la prueba, punto en donde el manipulador debe tomar el material a trasportar).



$$\begin{bmatrix} 9.2\text{Cos}[\theta 1](\text{Cos}[\theta 2] + 1.5543478260869565\text{Cos}[\theta 2 + \theta 3] + 1.3043478260869565\text{Sin}[\theta 2 + \theta 3 + \theta 4] \\ 9.2\text{Sin}[\theta 1](\text{Cos}[\theta 2] + 1.5543478260869565\text{Cos}[\theta 2 + \theta 3] + 1.3043478260869565\text{Sin}[\theta 2 + \theta 3 + \theta 4] \\ 13 + 12\text{Cos}[\theta 2 + \theta 3 + \theta 4] - 9.2\text{Sin}[\theta 2] - 14.3\text{Sin}[\theta 2 + \theta 3] \end{bmatrix}$$

Ecuación 7.

Se realizó una simulación del brazo utilizando las ecuaciones anteriores, Figura 28, sin embargo la forma en que se movían algunos de los eslabones del brazo simulado, no coincidían del todo con los movimientos reales del brazo físico por lo cual se tuvieron que hacer ajustes sobre los parámetros D-H en la simulación con el fin de ser lo más precisos y lograr que la forma en la que el brazo simulado se movía, fuera igual a la del brazo físico. Estos ajustes se reportan en la Tabla 4.

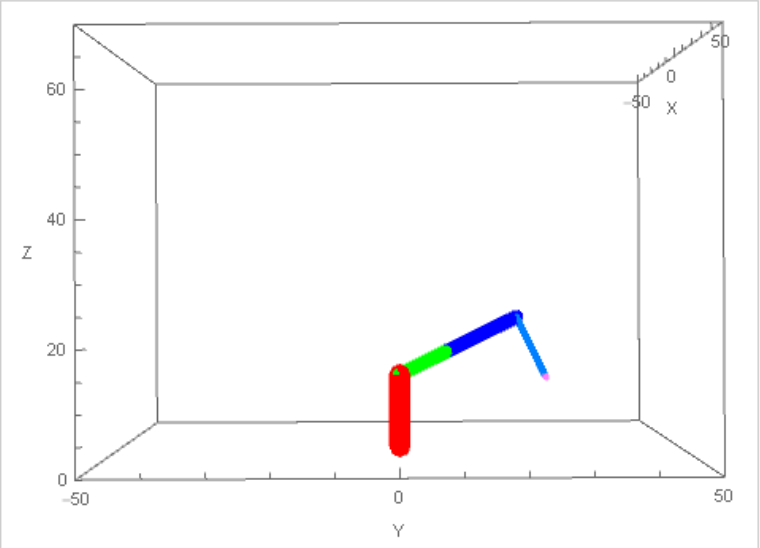


Figura 28 Brazo simulado

Tabla 4 Ajustes realizados.

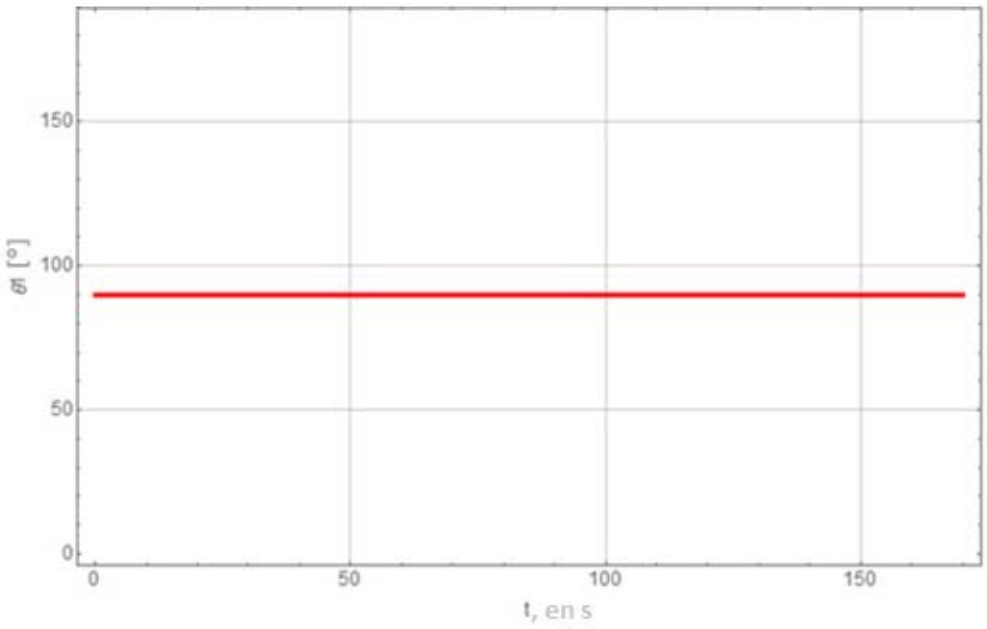
i	α_{i-1}[rad]	a_{i-1}[cm]	b_i[cm]	θ_i[rad]
1	$\frac{\pi}{2}$	0	13	$\frac{\pi}{2}+\theta_1$
2	0	9.2	0	θ_2
3	0	14.3	0	$38*\frac{\pi}{51}-\theta_3$
4	$\frac{\pi}{2}$	0	0	$\frac{10*\pi}{9}-\theta_4$
5	0	0	12	θ_5

Cabe mencionar que el vector de la base al efector final cambió y puede ser consultado en el anexo A.

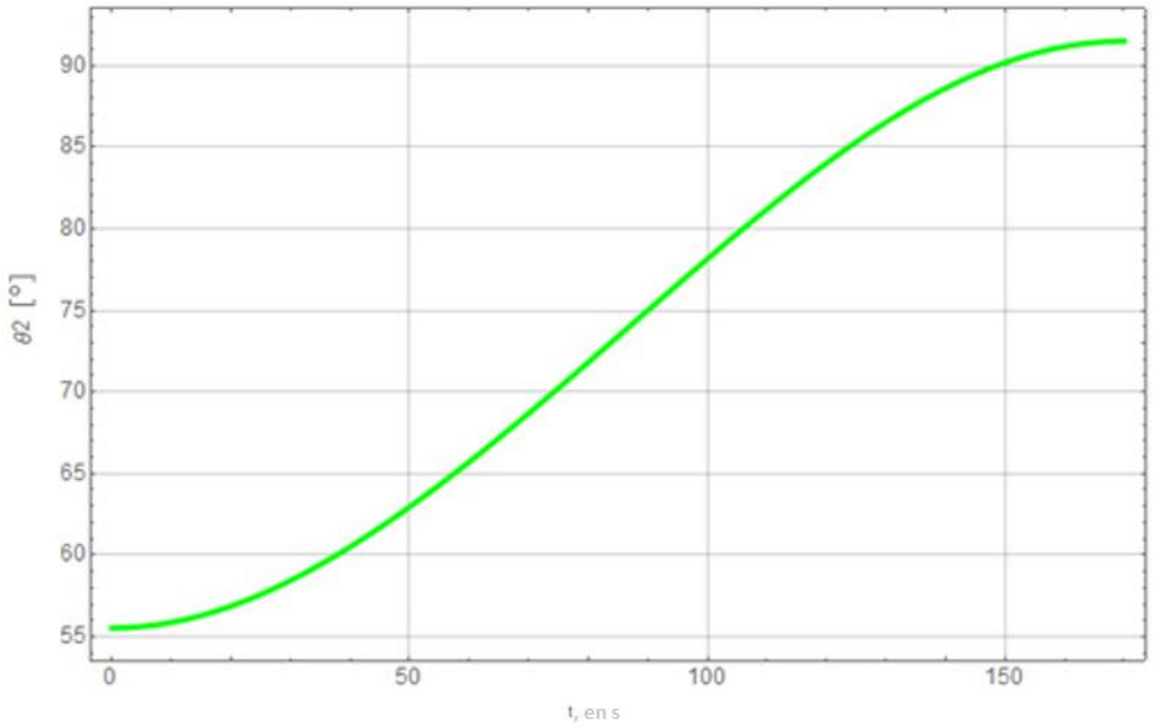
4.1.2 Cinemática inversa

En el apartado anterior se hablaba de calcular la posición y orientación del efector final en función de sus ángulos, en este capítulo se pretende lo inverso: dada la posición y orientación del efector calcular el conjunto de ángulos que logren este resultado. La cinemática inversa fue calculada numéricamente mediante el método Newton-Rapson y utilizando un perfil de trayectoria de línea recta.

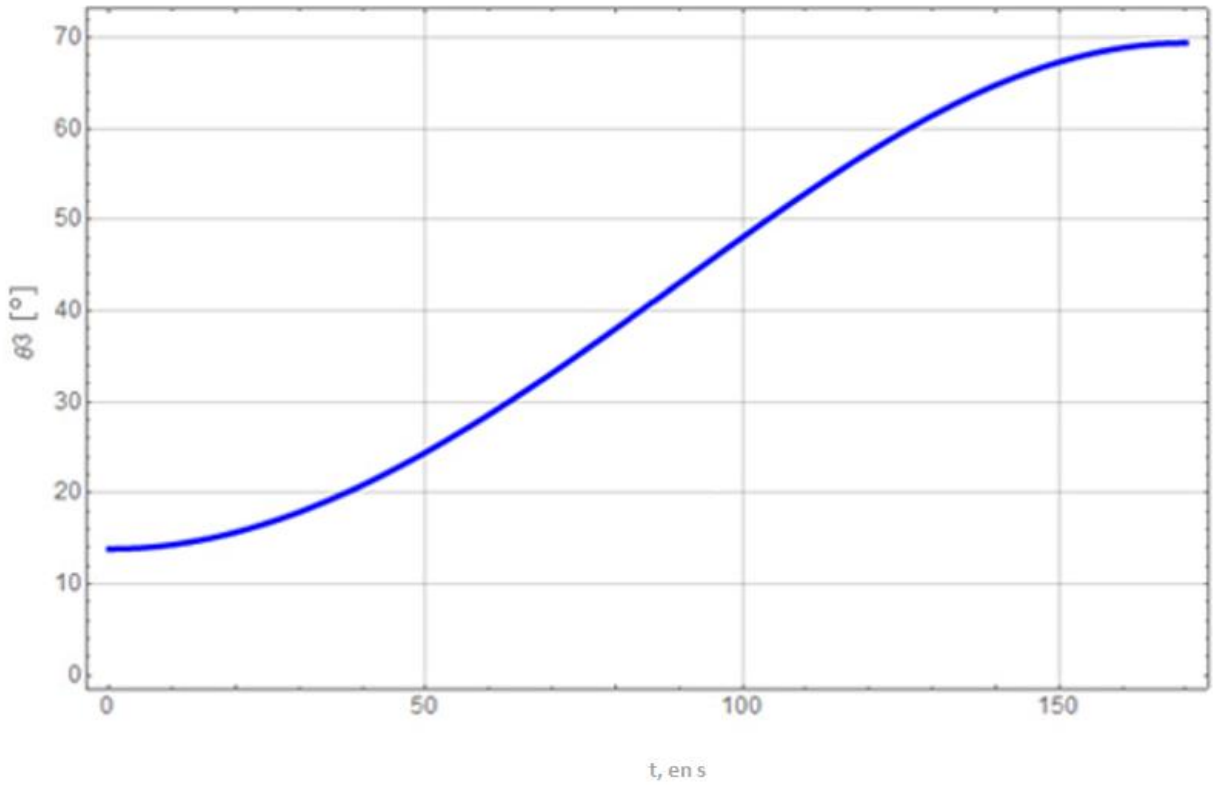
La trayectoria del manipulador se discretizó en 170 partes, únicamente por ser 170 cm la distancia máxima posible entre el MMO y la meta, de modo que por cada centímetro de distancia entre estos, hubiera también una partición de la trayectoria del manipulador. Las partes de la discretización representan la variación de ángulos que los eslabones deben experimentar para lograr configurar el brazo desde la posición inicial hasta la final, como se muestra en las siguientes gráficas.



Gráfica 1 Variación en el ángulo θ_1.



Gráfica 2 Variación en el ángulo θ_2.



Gráfica 3 Variación en el ángulo θ_3.

Se controlaron únicamente tres grados de libertad, correspondientes a θ_1, θ_2 y θ_3, por ser suficientes para resolver la posición final del efector, por lo cual se fijó θ_4 en 30° y, al no importar en la ejecución de la tarea la orientación que tome el efector final al trasladar objetos, el giro correspondiente θ_5 no se tomó en cuenta. Por su parte, el efector final sólo fue programado para abrirse y cerrarse al momento de tomar o dejar el objeto a transportar.

Como postura inicial del manipulador se optó por una que permitiera apreciar el movimiento de los eslabones a lo largo de la tarea. En dicha posición, Figura 29, se tomaron las coordenadas del efector y los ángulos correspondientes de los eslabones, resultando:

$P_1 = (12{,}0{,}10)$, en cm

$\theta_1 = 90°$

$\theta_2 = 54°$

$\theta_3 = 12°$

$\theta_4 = 30°$

θ_5 no se consideró, ya que para posicionar el brazo, este ángulo no es relevante.

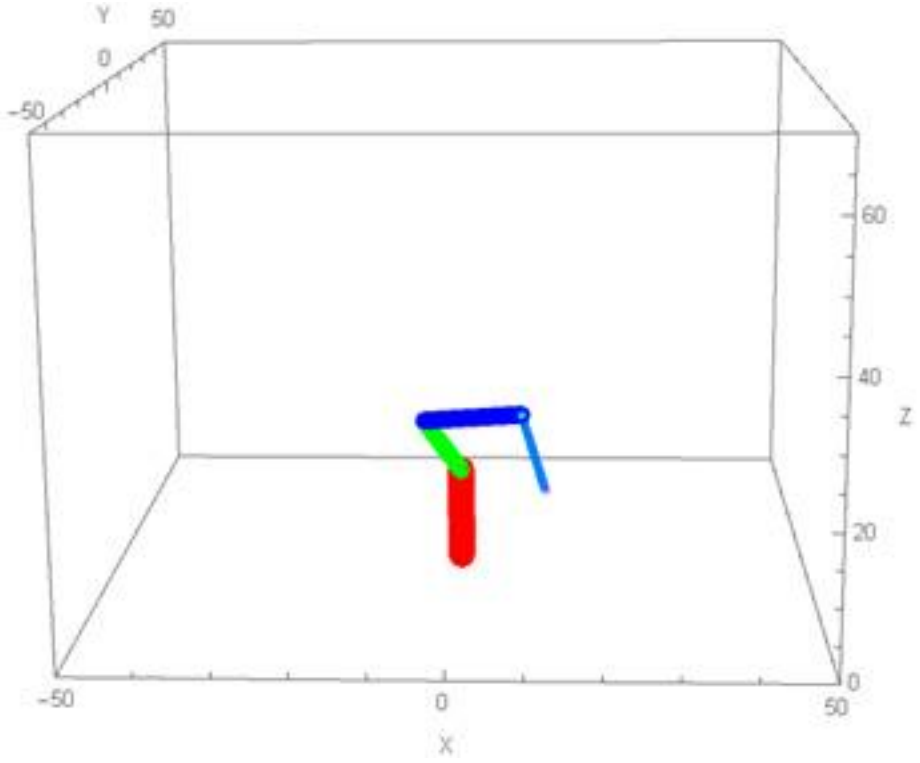


Figura 29 Postura inicial.

La postura final del manipulador, Figura 30, fue determinada tomando en cuenta la altura de la mesa meta, en dicha posición el punto que describe al efector y los ángulos de la configuración del brazo fueron:

$P_2 = (20,0,18)$, en cm

$\theta_1 = 90°$

$\theta_2 = 90°$

$\theta_3 = 70°$

$\theta_4 = 30°$

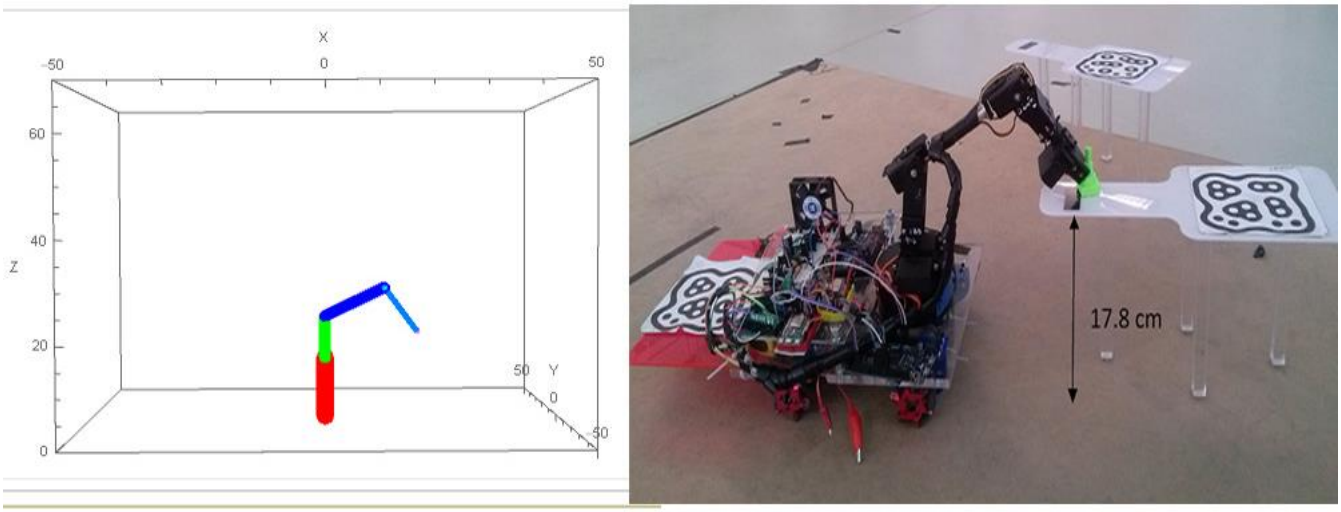


Figura 30 Posición final, simulación (izquierda) y prototipo (derecha).

Finalmete, tras discretizar la trayectoria que va del punto P_1 a P_2 en 170 partes utilizando el perfil de trayectoría en línea recta y el perfíl de velocidad cosinusoidal, se obtuvieron tres tablas (que se explicarán en el capítulo 4.3) que corresponden a los diferentes ángulos que θ_1, θ_2 y θ_3 deben tomar para así llegar a dicho punto.

4.2 MODELO CINEMÁTICO DE LA PLATAFORMA OMNIDIRECCIONAL

Como se mencionaba con anterioridad, las ruedas Mecanum tienen por sí solas restricciones mecánicas que no las hacen del todo omnidireccionales. Al colocar las ruedas en el chasis, la suma de estas restricciones genera restricciones globales sobre el movimiento del robot, pero al mismo tiempo contribuyen al movimiento de este dotando a la plataforma de un movimiento omnidireccional.

Dichas restricciones se deben expresar mediante el movimiento de cada rueda referido a un marco global ubicado en el centro del robot. Esto se logra propagando las velocidades de los distintos sistemas de referencia colocados estratégicamente sobre el robot, las ruedas y los rodillos de estas (Figura 31). El método utilizado para dicho fin es el de propagación de velocidades y fue consultado en [21].

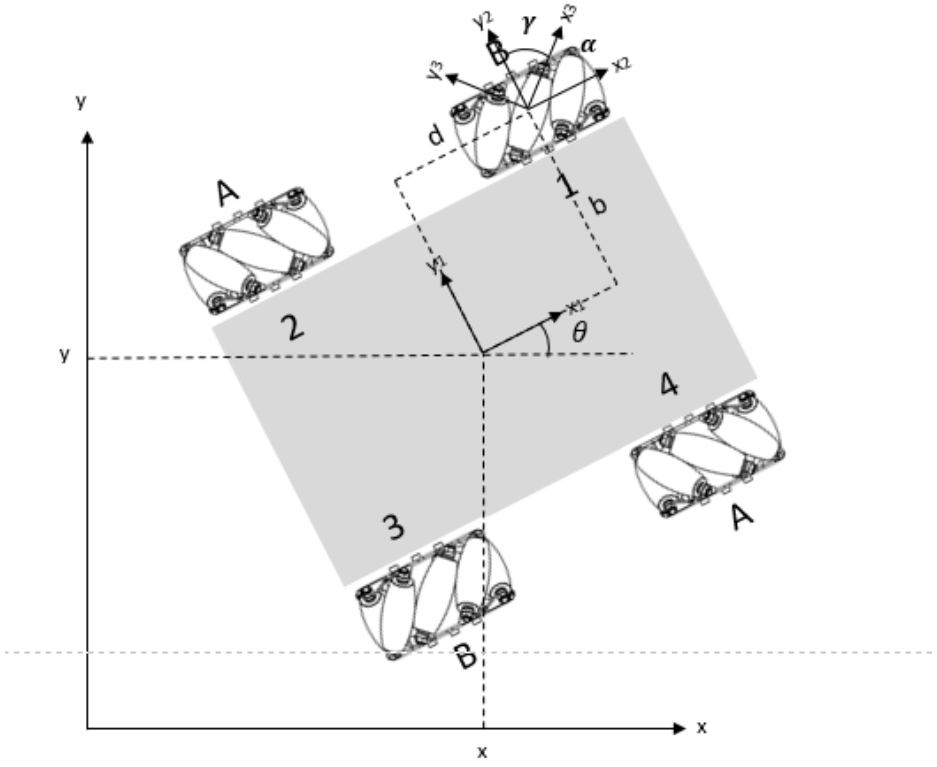


Figura 31 Configuración AB de las ruedas Mecanum y numeración.

Los parámetros mostrados en la figura anterior corresponden a:

d: coordenada x_1 de la posición de la rueda
b: coordenada y_1 de la posición de la rueda
$\boldsymbol{\alpha}$**:** ángulo de la ruedas respecto al sistema de referencia $\{x_1, y_1\}$
$\boldsymbol{\gamma}$**:** ángulo del vector velocidad de salida de cada rodillo de las ruedas respecto al sistema de referencia $\{x_2, y_2\}$
$\mathbf{v_R}$**:** velocidad lineal de la rueda
$\mathbf{v_r}$**:** velocidad lineal del rodillo de la rueda
$\boldsymbol{\phi_r}$**:** velocidad angular del rodillo
$\boldsymbol{\phi_R}$**:** velocidad angular de la rueda
$\boldsymbol{r_r}$**:** radio del rodillo
$\boldsymbol{r_R}$**: r**adio de la rueda

La cinemática de la plataforma omnidireccional que a continuación se muestra, fue desarrollada por el Dr. Víctor J. González Villela y el M.I. Noé Alfredo Martínez que forman parte del grupo de investigación Mechatronics Research Group.

Antes de comenzar con el análisis, es importante hacer mención de que la plataforma es considerada como un cuerpo rígido, que las ruedas no tienen deslizamientos y se desplazan a través de un plano horizontal.

Primeramente, se deben definir los sistemas coordenados, antes mencionados, que permitan describir el movimiento de la plataforma. Como marco inercial se define $\{x, y\}$ que se encuentra fijo al punto 0, Figura 32, y que contiene todo el ambiente del robot, el cual corresponde al espacio inteligente (delimitado por el alcance de la cámara).

La postura del robot está descrita por el vector

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{y} & \boldsymbol{\theta} \end{bmatrix}^{\boldsymbol{T}}$$

Ecuación 8.

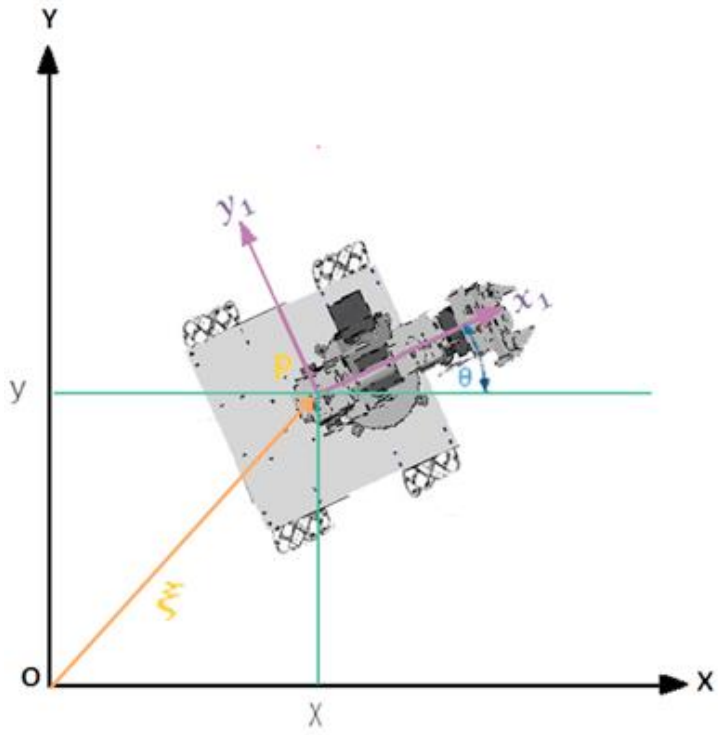


Figura 32 Sistemas coordenados propuestos.

Donde $\boldsymbol{\xi}$ describe la posición del punto P=(x,y), localizado en el centro de la plataforma omnidireccional, y la orientación $\boldsymbol{\theta}$ del marco coordenado del robot $\{x_1, y_1\}$ relativo al marco inercial $\{x, y\}$. La orientación $\boldsymbol{\theta}$ es medida desde el eje x al eje x_1 y estará descrita mediante una matriz de rotación referenciada al sistema $\{x, y\}$.

$$R(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 9.

Se utilizó el método de propagación de velocidades, consultado en [21], para resolver el problema de calcular las velocidades lineal y angular de los vínculos del robot, el cual está constituido por una cadena de cuerpos, donde cada uno es capaz de moverse en relación con sus cuerpos adyacentes. Debido a esta estructura, se puede calcular la velocidad de cada vínculo en orden (Figura 33), de tal modo que la velocidad del vínculo *i+1* será la del vínculo *i* más cualquier nuevo componente de velocidad que se haya agregado por la articulación *i+1*. En la Ecuación 10 se muestra la descripción de la velocidad lineal del vínculo i+1 visto desde la trama {i+1}.

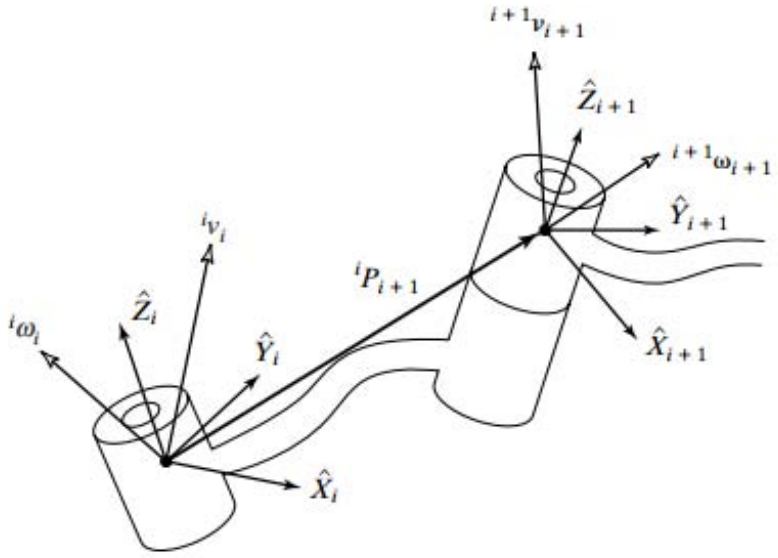


Figura 33 Velocidades lineales y angulares en los vínculos de un robot.

$$^{i+1}\boldsymbol{v}_{i+1} = {}^{i+1}_{i}\boldsymbol{R}(^{i}\boldsymbol{v}_{i} + {}^{i}\boldsymbol{\omega}_{i} \times {}^{i}\boldsymbol{P}_{i+1})$$

Ecuación 10.

En donde $^{i}\boldsymbol{P}_{i+1}$ representa un vector de magnitud constante en la trama *i*, que va del origen de la trama i al origen de la trama *i+1,* como se observa en la Figura 33.

Del mismo modo pueden agregarse velocidades de rotación cuando ambos vectores ω se refieren a una misma trama. Por lo tanto, la velocidad angular del vínculo i+1 es la misma que la del vínculo i más un nuevo componente producido por la velocidad de rotación en la articulación i+1. La descripción de la velocidad angular del vínculo i+1 respecto a la trama {i+1} queda:

$$^{i+1}\boldsymbol{\omega}_{i+1} = {}^{i+1}_{i}\boldsymbol{R}\,{}^{i}\boldsymbol{\omega}_{i} + \dot{\boldsymbol{\theta}}_{i+1}\,{}^{i+1}\hat{\boldsymbol{Z}}_{i+1}$$

Ecuación 11

En donde:

$$\dot{\boldsymbol{\theta}}_{i+1}\;\;{}^{i+1}\hat{\boldsymbol{Z}}_{i+1} = {}^{i+1}\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{0}\\ \dot{\boldsymbol{\theta}}_{i+1}\end{bmatrix}$$

Ecuación 12.

Cálculo de las restricciones cinemáticas de la plataforma omnidireccional

Con base en lo anterior, es posible calcular las restricciones cinemáticas que posee la plataforma omnidireccional calculadas por propagación de velocidades en el marco del robot $\{x_1, y_1\}$ al marco de las ruedas.

En la Figura 31 se muestra una posición aleatoria de la base omnidireccional respecto a la trama $\{x, y\}$, y los parámetros necesarios para modelar las ruedas Mecanum, además de dos sistemas de referencia adicionales, $\{x_2, y_2\}$ para describir el ángulo que presentan las ruedas respecto al sistema de la base $\{x_1, y_1\}$ y $\{x_3, y_3\}$ para referir el ángulo que los rodillos de estas al sistema $\{x_2, y_2\}$.

Los parámetros mostrados en la Figura 31 deben ser particularizados para cada rueda atendiendo los ejes coordenados dispuestos sobre el robot; dicha particularización puede ser expresada a modo de tabla como se muestra a continuación:

Tabla 5 Particularización de parámetros.

Rueda	${}^1Pn_2$	$\dot{\alpha}_n$	a_n	$\dot{\gamma}_n$	γ_n
1	${}^1P1_2$=(d,b,0)	$\dot{\alpha}_1$=0	a_1=0	$\dot{\gamma}_1$=0	$\gamma_1=\frac{\pi}{4}$
2	${}^1P2_2$=(-d,b,0)	$\dot{\alpha}_2$=0	a_2=0	$\dot{\gamma}_2$=0	$\gamma_2=\frac{3\pi}{4}$
3	${}^1P3_2$=(-d,-b,0)	$\dot{\alpha}_3$=0	a_3=0	$\dot{\gamma}_3$=0	$\gamma_3=\frac{\pi}{4}$
4	${}^1P4_2$=(d,-b,0)	$\dot{\alpha}_4$=0	a_4=0	$\dot{\gamma}_4$=0	$\gamma_4=\frac{3\pi}{4}$

Una vez descritos estos parámetros, y tras haber aplicado el método de propagación de velocidades, se obtienen las restricciones cinemáticas de cada rueda respecto al sistema de referencia del robot.

$$\begin{pmatrix} \cos(\gamma+\alpha+\theta) & \sin(\gamma+\alpha+\theta) & d\cdot\sin(\gamma+\alpha)-b\cdot\cos(\gamma+\alpha) & -r_R\cdot\cos(\gamma) & -r_r \\ -\sin(\gamma+\alpha+\theta) & \cos(\gamma+\alpha+\theta) & b\cdot\sin(\gamma+\alpha)+d\cdot\cos(\gamma+\alpha) & r_R\cdot\sin(\gamma) & 0 \end{pmatrix}$$

Ecuación 13.

Para describir el sistema general que representa el robot, no es necesario hacer uso de todas las variables que en él intervienen, es por esto que se eligieron como coordenadas generalizadas a la componente en x e y de velocidad lineal de la base, la velocidad angular de la plataforma y las velocidades angulares de cada una de las ruedas, Ecuación 14.

$$\dot{q_m} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4 \end{bmatrix}$$

Ecuación 14

Finalmente, utilizando la Ecuación 15 se pueden obtener las velocidades angulares que cada rueda debe tener para que sobre la base se logre una velocidad lineal en x (v_x) y una en y (v_y), además de la velocidad angular (ω) que se desee según sea la trayectoria que el MMO deba realizar.

$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{r_R} & -\frac{1}{r_R} & -\frac{b+d}{r_R} \\ \frac{1}{r_R} & \frac{1}{r_R} & -\frac{b+d}{r_R} \\ \frac{1}{r_R} & -\frac{1}{r_R} & \frac{b+d}{r_R} \\ \frac{1}{r_R} & \frac{1}{r_R} & \frac{b+d}{r_R} \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Ecuación 15.

4.3 COORDINACIÓN BRAZO MANIPULADOR-BASE OMNIDIRECCIONAL

El control de los manipuladores móviles principalmente contempla dos aspectos: la planeación de movimientos, tanto para la plataforma móvil como para el manipulador, y la otra, el control coordinado de ambos [22]. El tipo de coordinación dependerá de la tarea a realizar, es por esto que existen distintos tipos de coordinación además de que éste es un tema que continúa en estudio.

En esta tesis se entiende como coordinación de movimientos entre la base y el brazo al hecho de sumar las ventajas que cada uno de los robots posee para llevar a cabo la tarea propuesta, dicha coordinación debe ser expresada como un movimiento sincronizado entre base y brazo, de tal forma que el brazo parta de una configuración inicial y vaya posicionándose hasta llegar a una final, esto a medida en que la plataforma se vaya acercando a la meta.

El problema de la coordinación base-manipulador ha sido explorado y desarrollado por varios autores, algunas de las soluciones obtenidas pueden ser consultadas en el subcapítulo 2.4 donde se mencionan los antecedentes y la evolución del tema.

En el presente trabajo, se dio solución a dicha coordinación haciendo que la trayectoria del manipulador dependiera de la distancia entre la base del robot y la meta. Se tomó el valor máximo posible de esta distancia y se dividió en 170 puntos (Figura 34), estos puntos fueron dispuestos a modo de primer columna en la Tabla 6, en la segunda columna, de la misma tabla, se colocaron, para el caso de θ_1, los grados que la variable de articulación debe moverse para llegar de la posición inicial fijada, capítulo 4.1.2, a la final. Similar al caso de θ_1, se construyeron otras dos tablas, una para θ_2 y otra para θ_3;

.

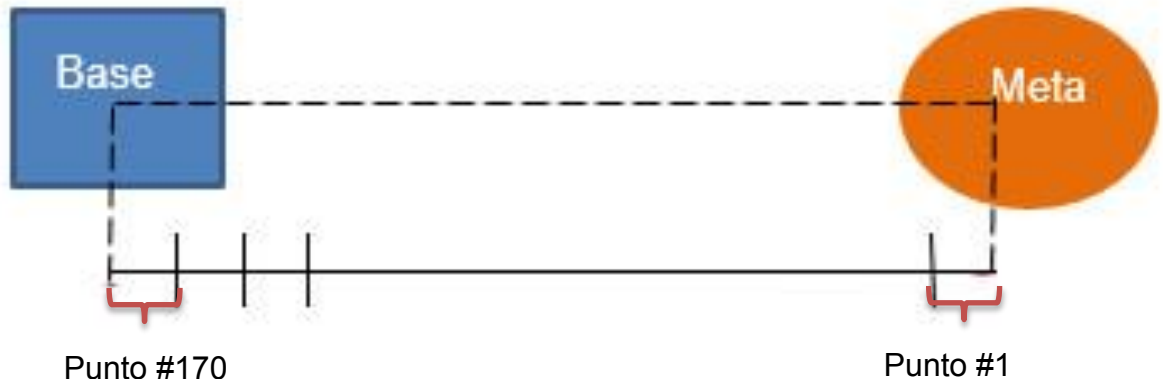
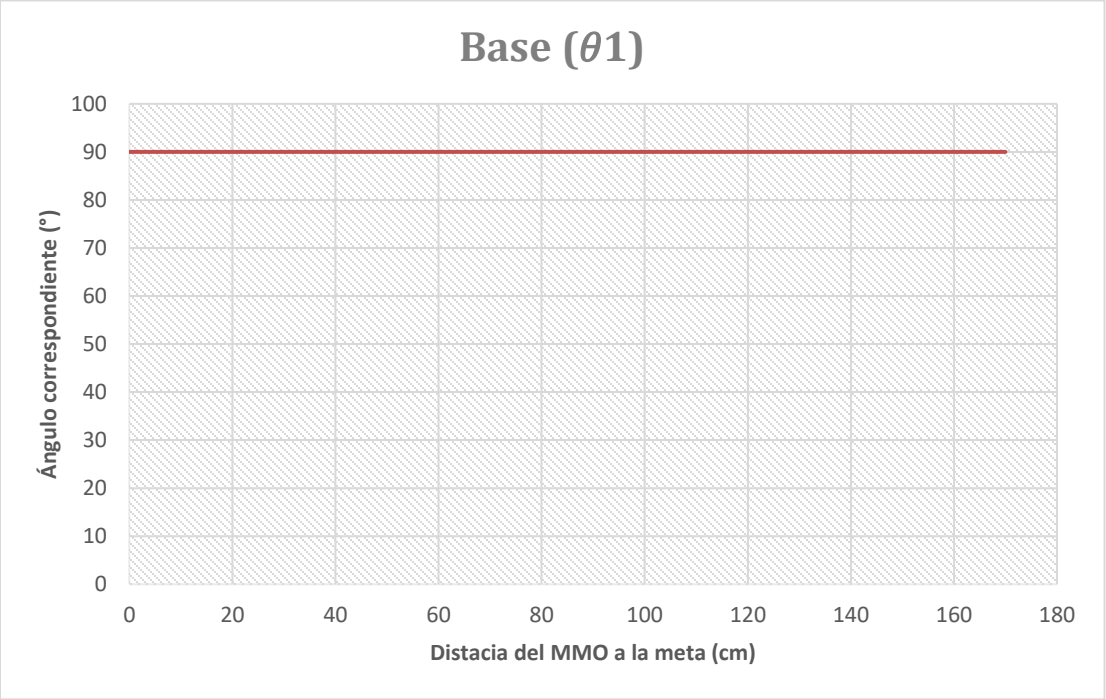


Figura 34 Obtención de intervalos.

Tabla 6 Ejemplo de la disposición de intervalos y ángulos.

Número de intervalo	Número de ángulo
170	1
169	2
168	3
.	.
.	.
.	.
1	170

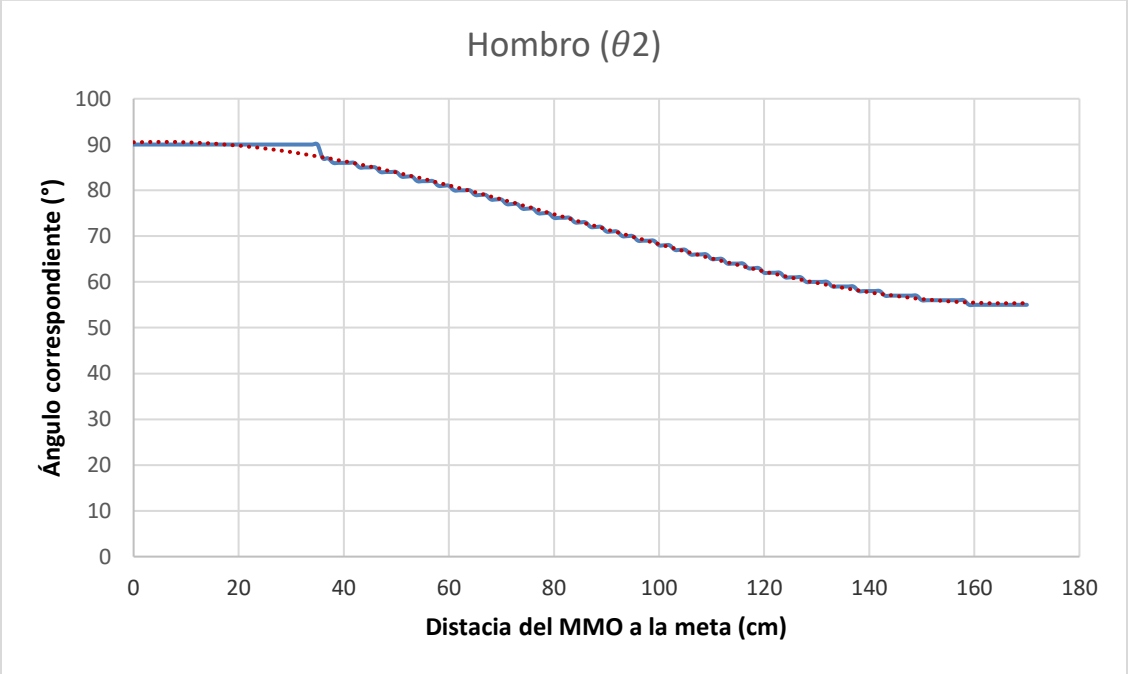
Finalmente se dibujaron las gráficas de las tres tablas mencionadas y se obtuvieron tres ecuaciones que describen el comportamiento de cada una de las curvas que se muestran a continuación.



Gráfica 4. Ángulo θ_1 dependiente de la distancia.

$$y = 90$$

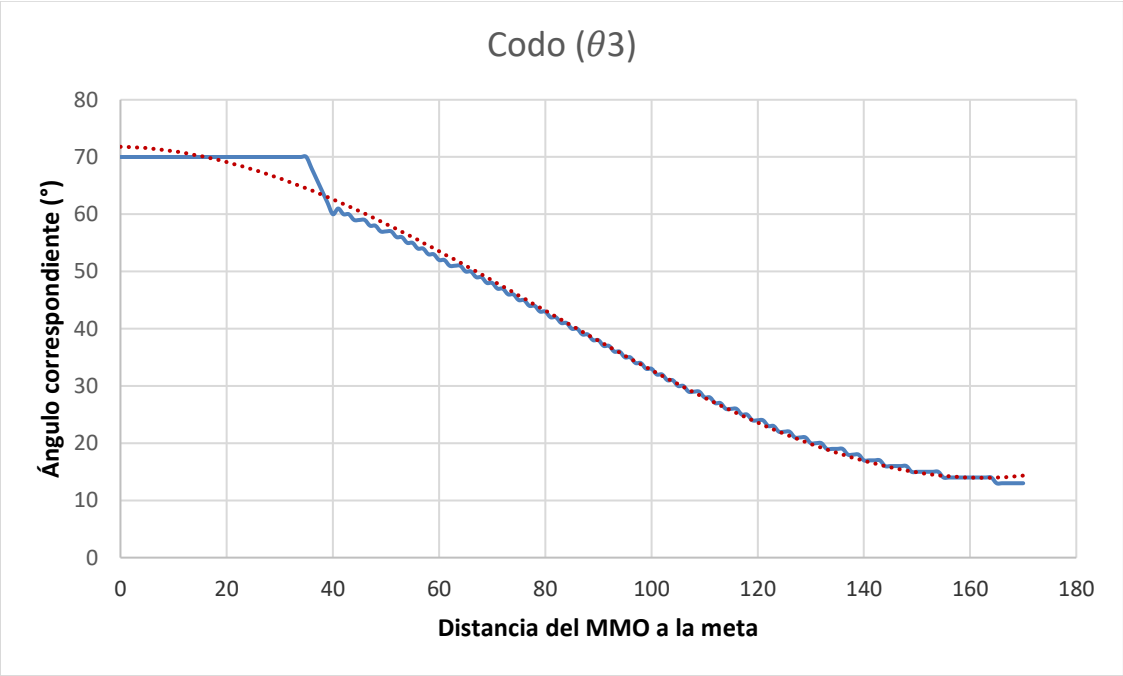
Ecuación 16.



Gráfica 5. Ángulo θ_2 dependiente de la distancia.

$$v = 2\text{x}10^{-5}x3 - 0.0044x2 + 0.0457x + 90.466$$

Ecuación 17.



Gráfica 6. Ángulo θ_2 dependiente de la distancia.

$$y = 3\text{x}10^{-5}x3 - 0.0064x2 - 0.0168x + 71.782$$

Ecuación 18.

Es mediante estas ecuaciones y el cálculo de la distancia del MMO a la meta que se realiza la coordinación entre la base y manipulador.

4.4 MÉTODO DE CAMPOS POTENCIALES

El método de campos potenciales permite al manipulador móvil omnidireccional partir desde su punto inicial y llegar a su punto meta evadiendo obstáculos, esto conociendo únicamente su modelo cinemático.

Este método toma en cuenta un espacio en el que interactúan fuerzas, representadas por vectores, tanto de atracción (puntos meta) como de repulsión (obstáculos). Cabe mencionar que, para la realización de pruebas, no se utilizaron obstáculos, ya que no es importante para el objetivo de esta tesis.

Para aplicar el método de campos potenciales desde el punto de vista cinemático, es necesario cambiar los vectores de fuerza por vectores de velocidad, asimismo, la meta está representada por un vector que parte desde el MMO hasta aquélla.

Con el fin de evitar colisiones y velocidades demasiado grandes, se define un área de evasión alrededor del objeto meta y el MMO, y una velocidad máxima, de modo que la velocidad del MMO va disminuyendo a medida que se acerca a dicha área.

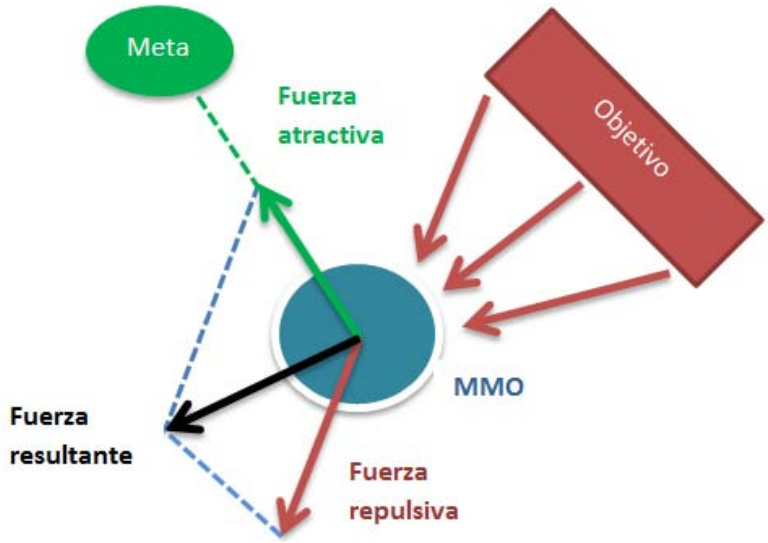


Figura 35 Modelado de fuerzas.

En la siguiente figura se pueden observar los elementos tomados en cuenta para el método de campos potenciales:

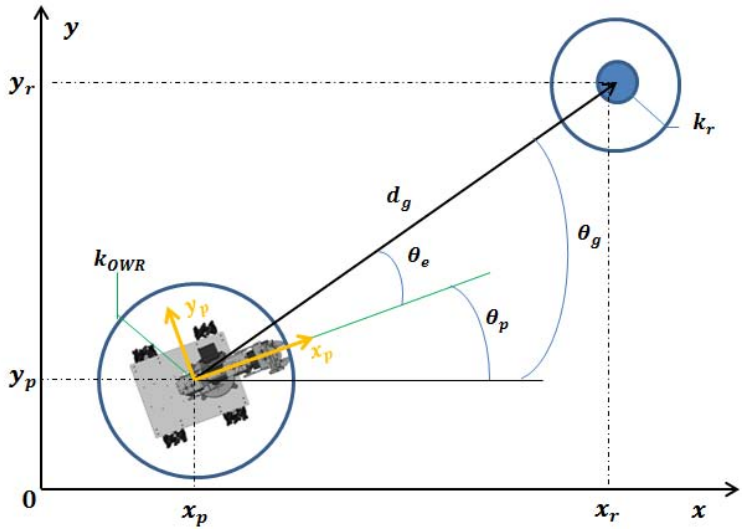


Figura 36 Parámetros del método de campos potenciales.

Donde:

$v_{máx}$ = velocidad máxima con la que puede desplazarse el MMO
k_r = radio del área de evasión
d_g = vector de atracción del robot a la meta
θ_g = ángulo de d_g
θ_i = ángulo de inclinación del robot
θ_e = ángulo de error entre θ_g y θ_i
k_{OWR} =área de seguridad del robot

La solución para llegar al punto deseado o meta, se reduce a determinar la dirección y velocidad en que el manipulador móvil podrá llegar a este punto. Considerando que el punto inicial sea $P_i = (x_i, y_i)$ y el punto final $P_f = (x_f, y_f)$. La distancia entre estos puntos puede ser calculada mediante:

$$d_g = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$$

Ecuación 19. Distancia entre punto inicial y meta

El ángulo a la meta queda definido por:

$$\theta_g = tan^{-1}\left(\frac{y_f - y_i}{x_f - x_i}\right)$$

Ecuación 20. Ángulo a la meta.

Y el ángulo de error:

$$\theta_e = \theta_g - \theta_p$$

Ecuación 21. Ángulo de error.

Para poder posicionar el manipulador, con el fin de que tome el objeto, a partir de la base y procurando que el servomotor de la base del manipulador gire lo menos posible, es necesario calcular la velocidad angular con la que la base debe moverse, para esto se emplea la siguiente ecuación:

$$\omega = \omega_{máx} * \sin(\theta_e)$$

Ecuación 22. Velocidad angular del MMO.

Donde $\omega_{máx}$ representa la velocidad angular máxima alcanzada por este, la cual se logra cuando el ángulo de error θ_e es $\pm 90°$ y la velocidad mínima cuando $\theta_e = 0°$.

Otras ecuaciones que se toman en consideración son las siguientes

$$\omega = \omega_{máx} * \sin(-\theta_p)$$

Ecuación 23. Velocidad angular

$$\omega = \omega_{máx} * \sin(\theta_e - \frac{\pi}{2})$$

Ecuación 24.Velocidad angular

La velocidad con la que el MMO se desplaza, tanto en el eje *x* como en el *y*, está dada por las siguientes ecuaciones:

$$v = \begin{cases} v_{max} * \cos(\theta_e) & si \quad |d_g| > k_r + K_{OWR} \\ \frac{v_{max}}{k_r} * d_g \cos(\theta_e) & si \quad |d_g| < k_r + K_{OWR} \\ 0 & si \quad |d_g| \leq K_{OWR} \end{cases}$$

Ecuación 25. Velocidad sobre el eje X.

$$v_{yp} = \begin{cases} v_{xmax} * \sin(\theta_e) & si \quad |d_g| > k_p + K_{OWR} \\ \frac{v_{xmax}}{k_r} * d_g \, \mathrm{s}\, in(\theta_e) & si \quad |d_g| < k_p + K_{OWR} \\ 0 & si \quad |d_g| \leq K_{OWR} \end{cases}$$

Ecuación 26. Velocidad sobre el eje Y.

5 DISEÑO DEL MANIPULADOR MÓVIL OMNIDIRECCIONAL

Anterior a este trabajo se encuentra el de Balpuesta [23], en el cual se propone la utilización de campos potenciales como control de una plataforma omnidireccional de ruedas suecas en configuración AB. Se considera ésta ya que algunos de los modelos matemáticos y físicos planteados en ella se tomaron como base para el presente trabajo. Cabe mencionar que se hicieron cambios importantes tanto en la parte física como en la programación de ella, con el fin de acoplar el manipulador serial y llevar a cabo la coordinación base-brazo necesaria en el cumplimiento de la tarea que esta tesis reporta.

A continuación se describen los elementos que componen al manipulador móvil omnidireccional, tanto aquéllos que fueron diseñados y manufacturados como los que se seleccionaron y compraron. Los planos de dichos componentes y el ensamble general entre estos, pueden ser consultados en el Anexo F. En las siguientes figuras se puede apreciar el diseño final del MMO.

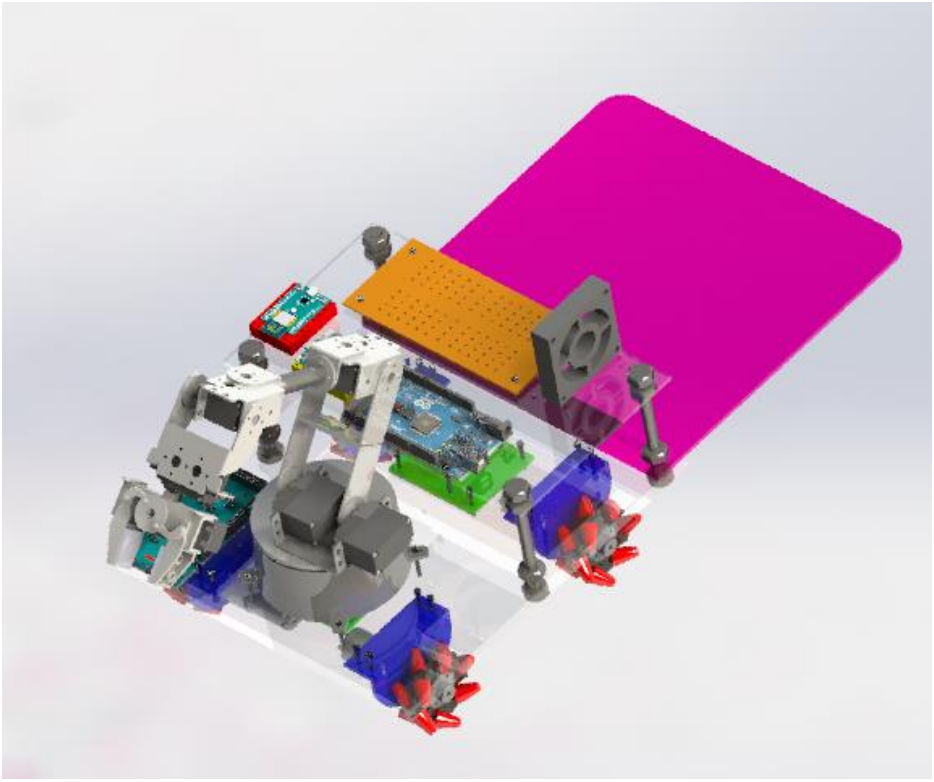


Figura 37. Diseño final del MMO.

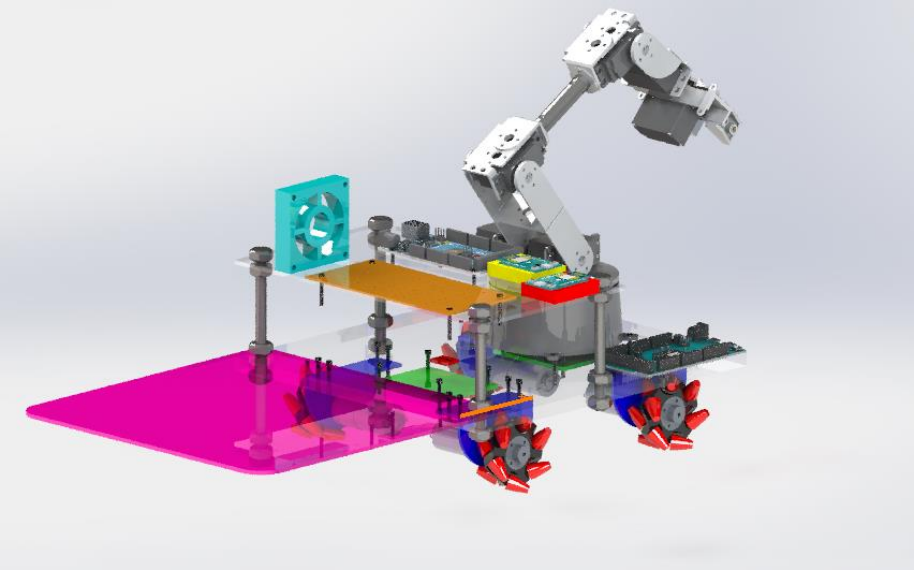


Figura 38. Modelado en CAD del MMO.

5.1 CHASIS

El chasis del MMO está constituido por tres placas de acrílico (Figura 37), la primera que sirve como soporte para las ruedas suecas, los motores que actúan sobre estas, la batería que alimenta al sistema y el brazo manipulador, la segunda diseñada para montaje de los elementos electrónicos como la tarjeta Arduino MEGA, el regulador de voltaje, el módulo de comunicación ESPino y el convertidor de niveles lógicos, y la última para alojar el marcador fiducial. Los planos de dichas placas, además de los planos de ensamble, pueden ser consultados en el Anexo F.

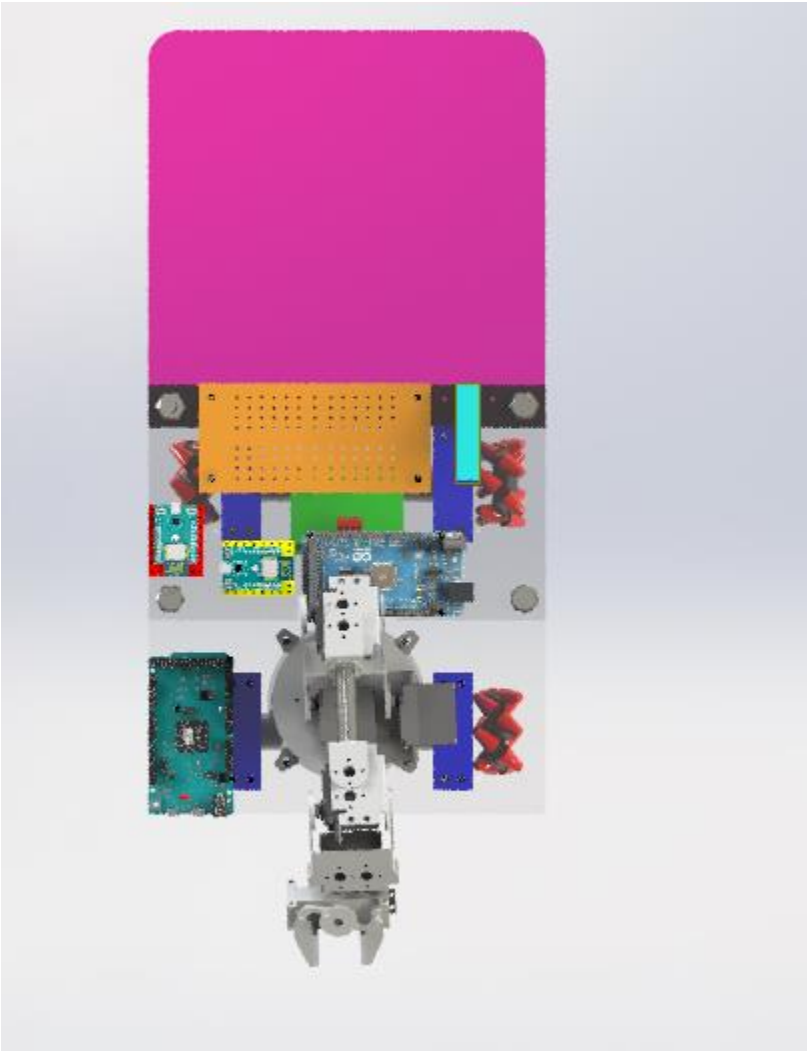


Figura 39. Chasis del MMO.

5.2 MOTORES

En la Figura 40 se muestra un EMG30® que es un motor de corriente directa que trabaja a 12 V y se caracteriza por traer integrado un encoder o codificador de cuadrante que manda un tren de pulsos 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 cuántas 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. El motor cuenta con condensadores internos de filtro que ayudan a minimizar el ruido generado por el motor al girar [24] .

5.2.1 Características:

- Tensión nominal: 12 V
- Par: 1,5 kg/cm
- Velocidad nominal: 170 rpm
- Corriente nominal: 530 mA
- Velocidad sin carga: 216 rpm
- Corriente sin carga: 150 mA
- Velocidad angular: 360 ppr
- Longitud total: 86,6 mm
- Diámetro motor: 30 mm
- Diámetro eje: 5 mm
- Longitud eje: 9 mm

Figura 40 Motor EMG30®.

5.2.2 Conexión

El motor EMG30® en su conector JST cuenta con 6 terminales, que tienen un color diferente. En la Tabla 7 se puede apreciar a qué conexión corresponde cada color de cable.

Tabla 7 Conexión del motor EMG30®.

Color de cable	Conexión
Morado (1)	Sensor Hall B Vout
Azul (2)	Sensor Hall A Vout
Verde (3)	Sensor Hall tierra
Café (4)	Sensor Hall Vcc
Rojo (5)	+ Motor
Negro (6)	- Motor

5.2.3 Soporte

El soporte para el motor EMG30®, Figura 41, está diseñado para ser fácil de montar dicho motor sobre él, está fabricado de lámina de aluminio de 2 mm de espesor con acabado esmalte en azul.

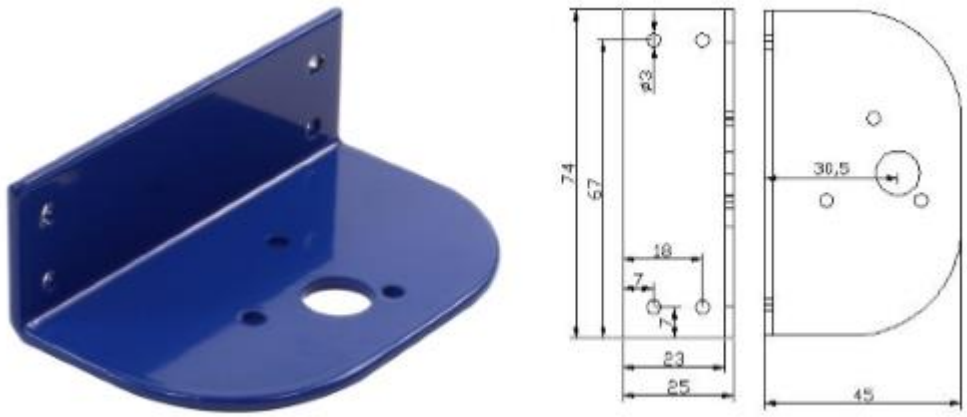


Figura 41 Soporte para motor EMG30® [25].

La siguiente figura, muestra cómo lucen los motores, soportes y llantas montados sobre la base del MMO.

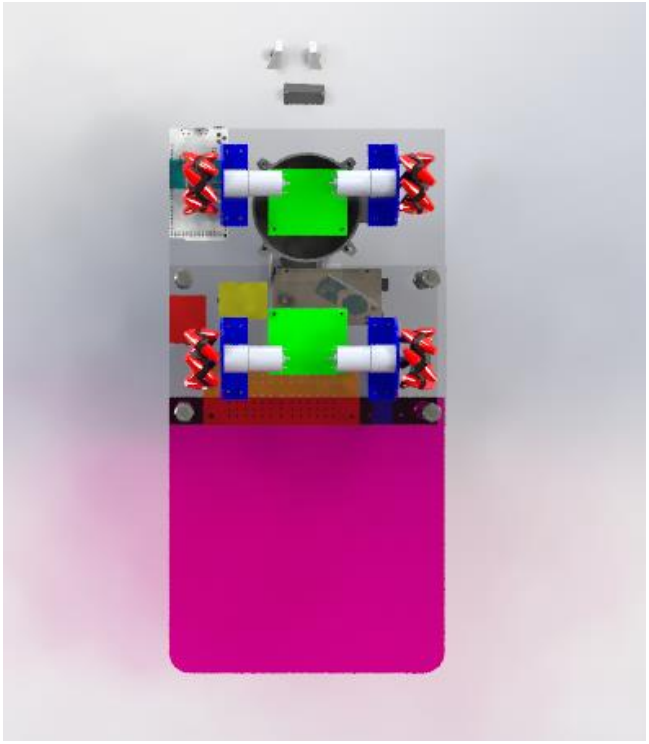


Figura 42. Ensamble soportes-motor-ruedas

5.3 RUEDAS

Las ruedas seleccionadas para el prototipo son las ruedas Mecanum con configuración AB (Figura 43). Estas ruedas, como ya se mencionaba en el subcapítulo 2.4 permiten a la plataforma, y al robot en general, realizar movimientos hacia atrás, delante y girar sobre su propio eje. Estas ruedas poseen una corona de rodillos orientados a 45º.

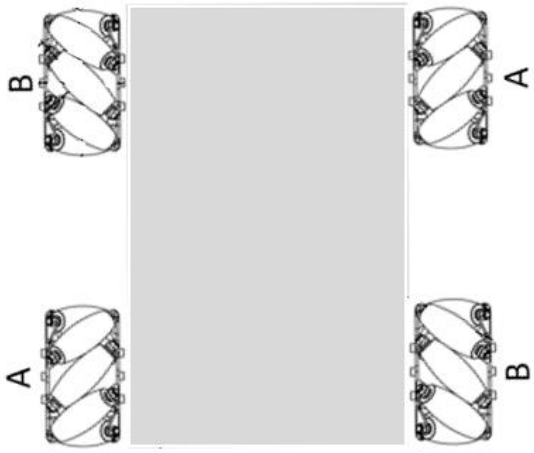


Figura 43 Configuración AB de cuatro ruedas suecas.

5.4 BATERÍA

El prototipo es alimentado con una batería LiPo (abreviatura de Litio y Polímero) de tres celdas recargable mostrada en la Figura 44, se optó por una batería de este tipo ya que es ligera, de tamaño compacto y ofrece alto voltaje y corriente de salida adecuados para el prototipo.

Sus especificaciones técnicas son las siguientes:

Carga: 5000 mAh
Voltaje de salida: 11.1 V
Peso: 354 g
Dimensiones: 162x21x46 mm
Plug de carga: JST-XH
Plug de descarga: 5.5 mm

Figura 44 Batería de LiPo.

5.5 ARDUINO MEGA

Se seleccionó este microcontrolador basado en el ATmega1280 de la Figura 45, principalmente por ser una plataforma computacional flexible y por permitir la comunicación serial, necesaria entre este y la placa ESPino, la comunicación I2C para el control del MD25 y finalmente por sus 54 pines de entrada/salida digital suficientes para controlar las tarjetas MD25, la placa ESPino y los seis servomotores utilizados.

Figura 45 Microcontrolador Arduino MEGA.

.

5.6 TARJETA MD25

Módulo desarrollado por Devantech Ltd. el cual consiste en un driver con dos puentes H para motores de CD, está diseñado para controlar motores EMG30® pero puede utilizarse en cualquier otro motor de corriente directa que funcione a 12 V y hasta 2.8 A.

Utiliza una interfaz I2C para la comunicación con dispositivos como PIC y Arduino, entre otros.

De entre sus características destacan las siguientes:

- Dispone de entradas para codificadores de cuadratura de los motores
- Capaz de controlar dos motores de forma individual o combinada
- Permite conocer el consumo de cada motor
- Alimentación única de 12 V
- Corriente de salida de hasta 2.8 A para cada motor
- Control de potencia y aceleración [26]

5.6.1 Conexiones

Como se muestra en la Figura 46, existen dos conectores que permiten la conexión directa con los motores EMG30®. Los dos conectores son idénticos y cada uno dispone de seis terminales:

- Dos salidas que se conectan a los bornes del motor
- +Vcc para alimentación de los sensores de efecto Hall de los codificadores

- GND de alimentación de los sensores de efecto Hall
- Señal del codificador A
- Señal del codificador B

La tarjeta MD25 dispone de cuatro conectores idénticos de 4 vías para la interfaz I2C:

- +5 Vcc para alimentar a los dispositivos I2C; intensidad de corriente máxima admitida de 300 mA
- SDA/Rx línea de datos del bus I2C o recepción de datos en el modo serie
- SCL/Tx línea de reloj del bus I2C o transmisión de datos en el modo serie
- GND [27]

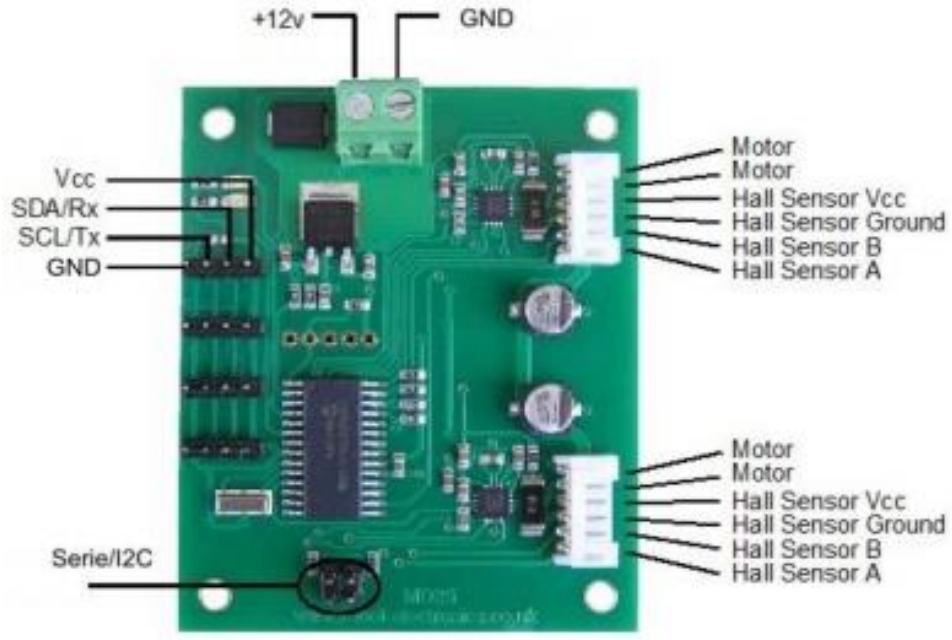


Figura 46 Conexiones del driver MD25.

5.6.2 Leds

El driver cuenta con leds que indican el estado en el cual se encuentra, el led rojo indica que el MD25 está debidamente alimentado. El led verde indica que existe actividad entre el bus I2C/Serie y el módulo; y se enciende intermitentemente cada vez que se asigna una nueva dirección I2C. Por defecto se emplea la dirección 0xB0 y cada vez que se conecta el módulo MD25, se enciende durante 500 ms.

5.7 ESPINO

ESPino es una placa de desarrollo integrada con el módulo de comunicación WiFi ESP8266. Viene programada con el firmware node-mcu, el cual permite programarlo en el lenguaje de programación Lua. También puede ser programado en C o C++ por medio de herramientas como el IDE de Arduino con soporte para ESP8266. Es precisamente este IDE el que se utilizó para establecer comunicación WiFi entre MATLAB® y el MMO. El programa desarrollado puede ser consultado en el Anexo D Programas de ESPino. Las dimensiones físicas y la disposición de pines se muestran en las Figura 47 y Figura 48.

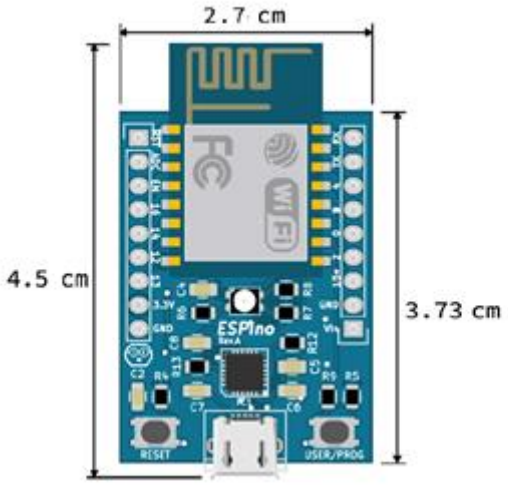


Figura 47 Dimensiones físicas y pines del ESPino.

5.7.1 Alimentación

ESPino se puede alimentar directamente desde la entrada USB (5 V) o, desde el pin Vin con una fuente de entre 4.4 V a 15 V. El regulador LDO integrado se encarga de regular este voltaje a los 3.3 V necesarios para el funcionamiento de la placa.

Los pines de alimentación son los siguientes:

- Vin: Entrada de alimentación de CD no regulada externa. El regulador integrado en la placa acepta entradas de entre 4.4 V a 15 V y proporciona una salida de 3.3 V con una intensidad de corriente máxima de 800 mA.

- 3.3 V: Salida del regulador interno de 3.3 V.
- GND: Pines de tierra

5.7.2 Comunicación

El chip ESP8266 integrado en la placa suministra comunicación inalámbrica por medio del estándar WiFi (802.11 b/g/n/d/e/i/k/r), con lo que se puede comunicar de forma fácil directamente a Internet. Puede ser configurado para funcionar como estación (cliente de un router), punto de acceso o access point (crea una red a la cual se pueden conectar otros dispositivos, por ejemplo, un teléfono inteligente o smartphone), o modo WiFi direct (P2P).

Además, incluye soporte para los siguientes métodos de comunicación alámbrica:

- UART (Serial)
- SPI
- I2C

5.7.3 Programación serial

ESPino® cuenta con un chip USB-Serial y un conector micro-USB que permite que sea conectado a una PC, la cual lo reconocerá como un puerto serial estándar pudiendo así programarla y utilizar monitores seriales para comunicarse con ella.

En caso de que la placa no sea reconocida automáticamente por el sistema operativo, puede que sea necesario instalar los controladores del chip USB-Serial, estos pueden ser descargados en la página referida en [28].

5.7.4 Programación

Una vez que el sistema operativo ha reconocido la placa, ésta puede ser programada de dos formas:

1 Con el lenguaje de programación Lua del firmware integrado node-mcu (sin entrar a modo “Bootloader”)
2 Con un firmware propio escrito en C o C++, por ejemplo, desde el IDE de Arduino con soporte para ESP8266. Para esto es necesario entrar en modo Bootloader.

5.7.5 Modo bootloader

El modo bootloader es necesario si se requiere subir nuevo firmware al ESP8266, es decir cada vez que se quiera cargar un nuevo programa, o restaurar el firmware node-mcu que viene por defecto.

Para poner el ESPino en este modo, es necesario seguir los siguientes pasos:

1 Presionar los botones RESET y USER/PROG al mismo tiempo sin soltarlos
2 Soltar el botón RESET, manteniendo USER/PROG presionado
3 Esperar un segundo y soltar USER/PROG.

Una vez que el sistema está en modo bootloader, es posible cargar nuevos programas.

5.7.6 Dispositivos integrados

La placa cuenta con un led RGB (rojo, verde y azul) y dos botones, de los cuales uno es RESET y el otro USER/PROG que está conectado internamente al pin 0 del ESP8266, y sirve para ponerlo en modo bootloader.

Estos elementos pueden ser apreciados en la Figura 48 y se encuentran conectados de la siguiente forma:

- led rojo: pin 2
- led verde: pin 5
- led azul: pin 4
- Botón USER/PROG: pin 0
- Botón RESET: pin RST.

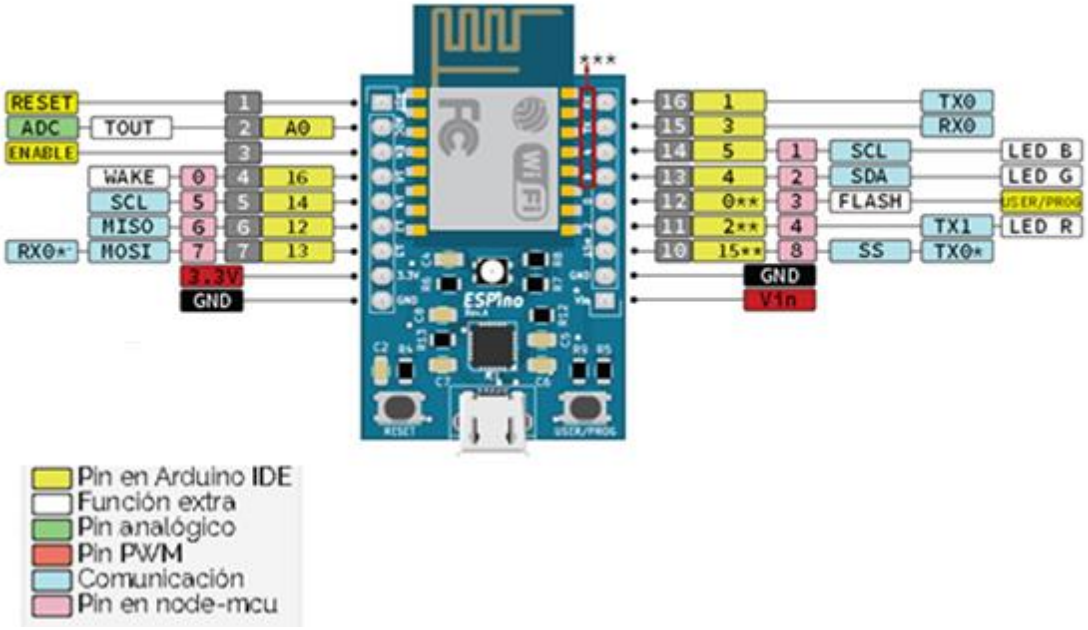


Figura 48 ESPino, disposición de pines.

Notas:

- En los ESPino Rev. A, las etiquetas RX y TX, 4 y 5 están invertidas por error (consultado en http://www.espino.io/es).
- Los pines 13 y 15 pueden ser usados como puerto serial en sustitución de TX0 y RX0. En el IDE de Arduino esto se logra llamando Serial.swap() después de Serial.begin().

El manual de instalación de esta placa en el IDE de Arduino puede ser consultada en el Anexo B.

5.8 CONVERTIDOR DE NIVELES LÓGICOS

Debido a que la placa ESPino trabaja con voltajes de entrada y salida que van desde los 0 a los 3.3 V, fue necesario utilizar un convertidor de niveles con el fin de acoplar las terminales de entrada y salida tanto del ESPino como del Arduino MEGA a sus voltajes de operación correspondientes. El convertidor que se seleccionó fue el convertidor de nivel lógico bidireccional Sparkfun, Figura 49, por ser pequeño y poseer cuatro canales que convierten señales de 5 a 3.3 V y de 3.3 a 5 V de forma bidireccional. Este convertidor de

nivel también funciona con dispositivos de 2.8 V y 1.8 V. Sus dimensiones son de 0.63 x 0.52 " (16.05 x 13.33mm).

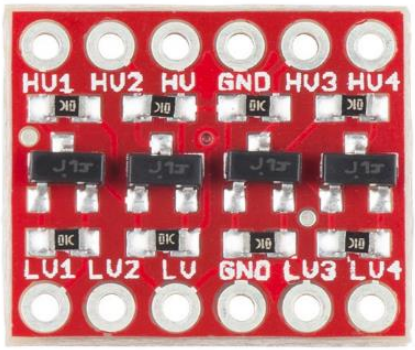


Figura 49 Convertidor de niveles lógicos.

5.9 MANIPULADOR

El manipulador que se seleccionó y que se muestra en la Figura 50, es un brazo antropomórfico con cinco grados de libertad operados cada uno por un servomotor (con excepción del segundo grado de libertad que emplea dos acoplados).

Los tres primeros grados de libertad corresponden a la base, hombro y codo y se encargan de ubicar el efector final en cualquier posición del espacio. Los últimos dos forman la muñeca y es el encargado de orientar el efector final con el fin de llevar a cabo la tarea planeada [22].

Figura 50 Manipulador antropomórfico seleccionado.

Regulador

Debido a que las baterías de LiPo seleccionadas tienen un voltaje de salida de 11.1 V, fue necesario construir un regulador de voltaje que proporcionara 5 V que es el voltaje de alimentación de los servomotores utilizados en el brazo. Para esto se colocaron dos etapas de regulación en serie una de 9V y otra de 5V, en ambas se dispusieron tres reguladores en paralelo, con el fin de aumentar la corriente de salida en cada etapa y disminuir la disipación de calor (Figura 51).

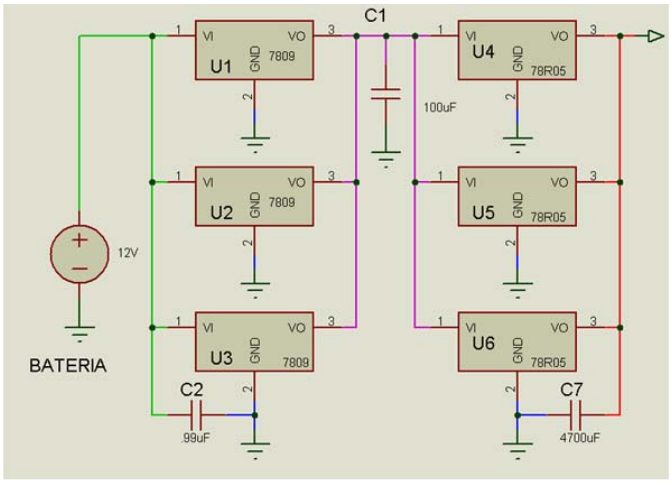


Figura 51 Regulador de voltaje con salida de 5 V [9].

6 IMPLEMENTACIÓN

El espacio inteligente que se diseñó para llevar a cabo las pruebas del prototipo robótico está constituido por varias capas que aseguran su carácter inteligente, a continuación se habla de manera amplia sobre ellas.

6.1 CAPA PASIVA DE PERCEPCIÓN

En esta capa se hace uso de la visión artificial proporcionada por el software reacTIVision® y una cámara. Dicha cámara fue colocada de modo que el eje principal del lente quedara perpendicular al suelo del ambiente inteligente (delimitado por el alcance de visión que posee la cámara.

ReacTIVision®

ReacTIVision® es un software multiplataforma de visión por computadora de código abierto, desarrollado para el seguimiento robusto y rápido de marcadores de referencia denominados fiduciales (Figura 52) adjuntos a objetos físicos.

El software envía mensajes TUIO a través del puerto UDP 3333 para cualquier aplicación cliente activado TUIO. El protocolo TUIO fue inicialmente diseñado para codificar el estado de los objetos tangibles y eventos multi-tacto de una superficie interactiva

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 adecuada, como la mayoría de las que tienen conexión USB, FireWire y cámaras DV

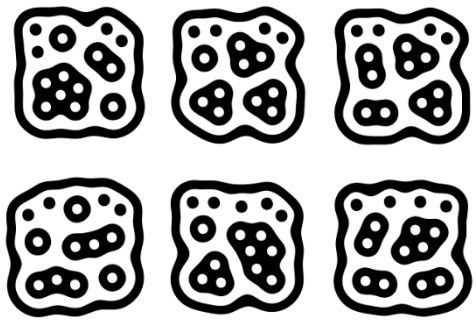


Figura 52 Marcadores de referencia fiduciales.

Ajustes sobre la cámara y fiduciales

Tal como se mencionó anteriormente, el ambiente inteligente está delimitado por el alcance de la cámara; las dimensiones de dicho ambiente cambian según la altura y el ángulo, con respecto a la horizontal, al que aquélla se encuentre. De acuerdo con lo anterior, se deben de realizar ajustes con el fin de que los datos proporcionados por el software de enlace, ver la referencia [19], entre reacTIVision® y MATLAB® Simulink®, sean lo más fieles posibles a la realidad.

El primer ajuste que se realizó fue el de mover el centro del fiducial de modo que coincidiera con el centro físico del MMO, Figura 53, ya que fue el que se utilizó en el subcapítulo 4.2 en la determinación del modelo cinemático de la base omnidireccional.

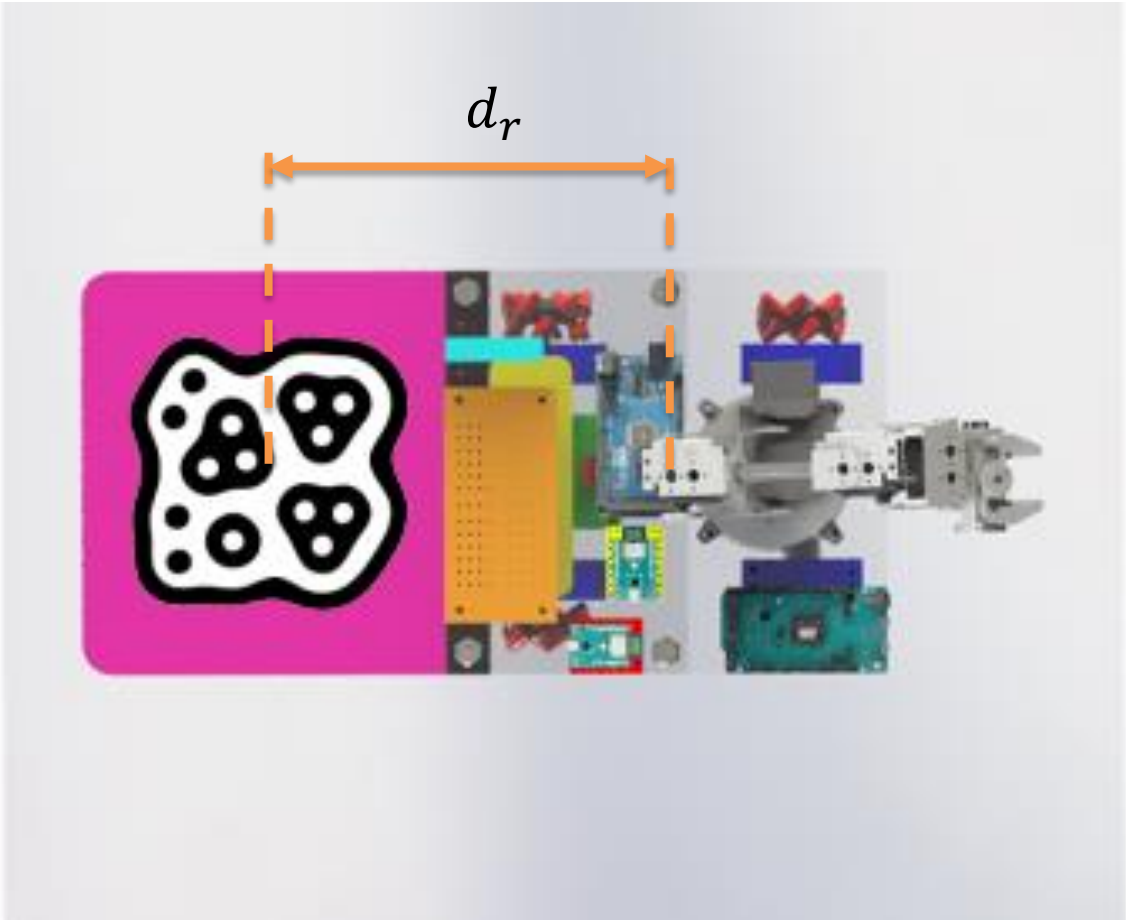


Figura 53 Distancia entre el centro del fiducial y el centro del robot.

Similar a lo anterior, la mesa objetivo y el elemento pasivo, tienen un fiducial (Figura 54) para su reconocimiento, el cual está desplazado una distancia d_o del centro geométrico del objeto a manipular (mostrado de color rojo en dicha figura).

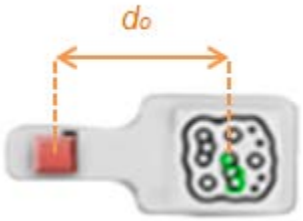


Figura 54 Distancia entre centro del fiducial y el centro del objeto a manipular.

El traslado del centro del fiducial al centro físico del MMO se realizó mediante una rotación sobre el eje z y una traslación sobre el eje Xr (coordenada en x de la posición del MMO respecto al sistema global), además de su correspondiente referencia del sistema del robot al sistema global de coordenadas, como sigue:

$$H_{MMO} = \begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} d_r \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} d_r cos\theta \\ d_r sen\theta \\ 0 \end{bmatrix}$$

Ecuación 27.

De lo cual se obtiene:

$$x_{MMO_{real}} = x_{MMO} + d_r cos\theta_{MMO}$$

Ecuación 28.

$$y_{MMO_{real}} = y_{MMO} + d_r sen\theta_{MMO}$$

Ecuación 29.

Se realizó lo anterior para las coordenadas de posición de las dos mesas utilizadas, identificadas por los índices t (tomar) y d (dejar), obteniendo lo siguiente:

$$x_{t_{real}} = x_t - d_o cos\theta_t$$

Ecuación 30.

$$Y_{t_{real}} = y_t - d_o sen\theta_t$$

Ecuación 31.

$$x_{d_{real}} = x_d - d_o cos\theta_d$$

Ecuación 32.

$$Y_{d_{real}} = y_d - d_o sen\theta_d$$

Ecuación 33.

Las ecuaciones anteriores se ven reflejadas en la programación que ms adelante se explicará.

El segundo ajuste realizado se debió a qué información de la posición y orientación de los fiduciales, que es proporcionada por el bloque de visión en Simulink®, no está dada en unidades de distancia, sino en pixeles. Por esto se optó por hacer la conversión de pixeles a centímetros.

Dicho ajuste consistió en escalar los pixeles de la imagen proporcionada por la cámara entre valores de 0 y 1 (los cuales se multiplicaron por 100 para su fácil manipulación) donde 0 corresponde a la posición inicial del marco coordenado de la imagen y 1 a la posición máxima alcanzada por ésta.

Posteriormente se orientó la cámara paralela al piso y en esta posición se obtuvieron las medidas de su alcance, tanto sobre el eje *x* como sobre el *y*. La conversión final de pixeles a centímetros se realizó utilizando las ecuaciones

$$x_s = \frac{x}{100} * (-2l) + l$$

Ecuación 34.

$$y_s = \frac{y}{100} * (2a) - a$$

Ecuación 35.

donde *a* es la distancia absoluta del origen coordenado medida desde el eje *x* al perímetro del área de trabajo y l es la distancia absoluta del origen del sistema coordenado al perímetro del área de trabajo medida desde el eje *y* (Figura 55) [19].

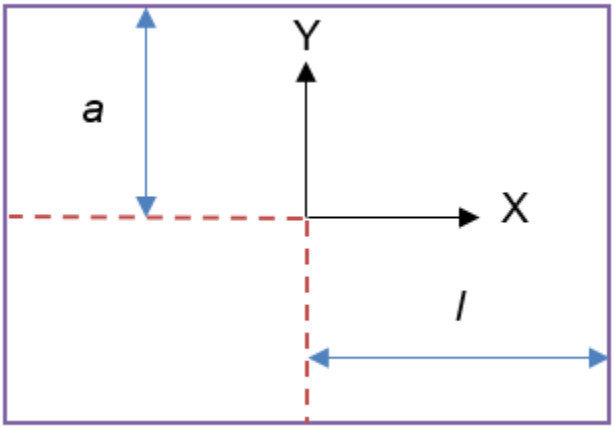


Figura 55 Ajuste sobre el área de visión de la cámara.

6.2 CAPA DE INTERACCIÓN

Esta capa está conformada principalmente por el manipulador móvil omnidireccional, ya que es el que interactuará con el ambiente para modificarlo y llevar a cabo la tarea. Además, los sensores que éste posee (como son los encoders contenidos en los motores de corriente directa) complementarán la información obtenida por la cámara, en la capa anterior.

6.3 CAPA DE COMUNICACIONES

Esta capa está compuesta por la cámara, la computadora, los microcontroladores y el dispositivo de comunicación WiFi ESPino, los cuales se describen de manera amplia en el Capítulo 5.

En la Figura 56 se puede apreciar el esquema básico de comunicación que se desarrolló para interconectar los elementos del sistema.

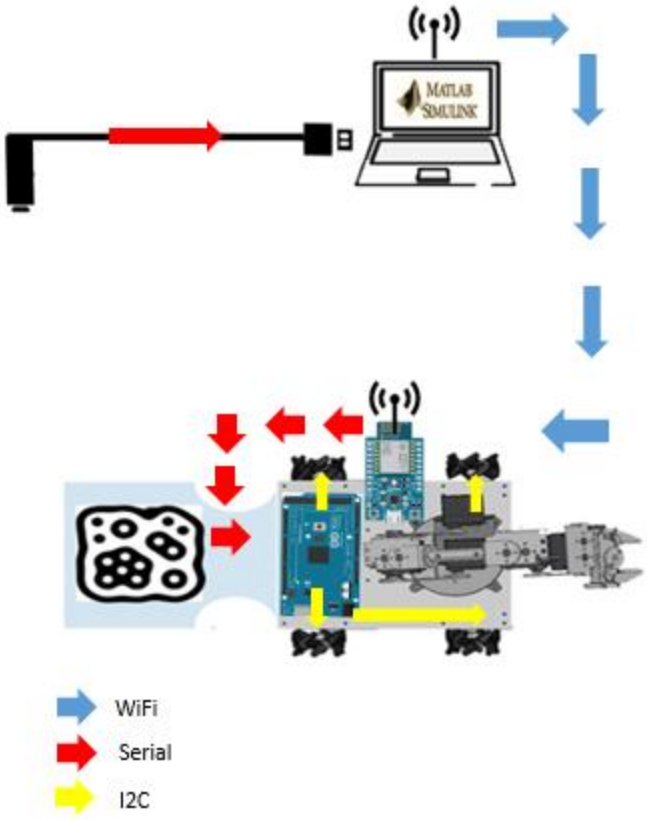


Figura 56 Esquema de comunicaciones.

Comunicación WiFi

Se empleó la comunicación WiFi (Figura 56, flechas azules) para comunicar la computadora con el dispositivo ESPino. Se eligió para dotar de mayor movilidad al robot por no involucrar cables de conexión.

El protocolo de comunicación UDP (User Datagran Protocol) utilizado, permite el envío de datagramas a través de la red 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.

Comunicación serial

Este tipo de comunicación fue utilizada principalmente para comunicar el dispositivo ESPino con el microcontrolador. En el esquema básico del modo asíncrono de dicho sistema, mostrado en la Figura 57, se encuentran dos terminales: TX/CK y RX/DT, la primera es la línea de transmisión por donde sale la señal de datos y la segunda es la terminal de recepción de datos.

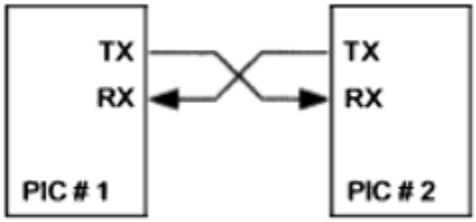


Figura 57 Diagrama de conexión por puerto serie entre dos microcontroladores.

El funcionamiento del puerto serie USART (Universal Synchronous Asynchronus Receiver Transmitter) en dicho modo, se caracteriza por permitir la comunicación bidireccional simultánea de datos, lo cual significa que durante la comunicación ente dos dispositivos USART (como lo son el ESPino y el Arduino), cada dispositivo puede transmitir y recibir datos a la vez.

Comunicación I2C

Este protocolo de comunicación se usó en el prototipo para comunicar el microcontrolador Arduino MEGA con la tarjeta MD25 (Figura 56, flechas amarillas).

El bus i2C (Inter-Integrated Circuit) es utilizado en la interconexión y transferencia sincrónica de datos en serie entre diferentes dispositivos cercanos. Se utilizan, a parte de la línea de masa, dos líneas entre los dispositivos: una para trasferencia de datos (SDA: Serial Dara Line) y otra para la señal de reloj (SCL: Serial Clock Line).

En una comunicación, uno de los dispositivos se comporta como servidor (Arduino MEGA) y los restantes como clientes (tarjetas MD25). Servidores y clientes pueden ser indistintamente transmisores o receptores. El dispositivo servidor es el que inicia la comunicación, genera la señal de reloj y termina la comunicación.

Una vez que el servidor genera la condición de inicio, coloca en la línea SDA la dirección del cliente con el que se quiere comunicar. A partir de ese momento, el servidor indica que

va a transmitir o recibir datos. Hecha esta indicación, se comienza la trasferencia de los bytes de datos.

Diagrama de conexión

En la Figura 58 se presenta el diagrama de conexión que se realizó con el fin de establecer los protocolos de comunicación antes citados. La programación desarrollada puede ser consultada en el Anexo D Programas de ESPino y en el Anexo E Programas de Arduino.

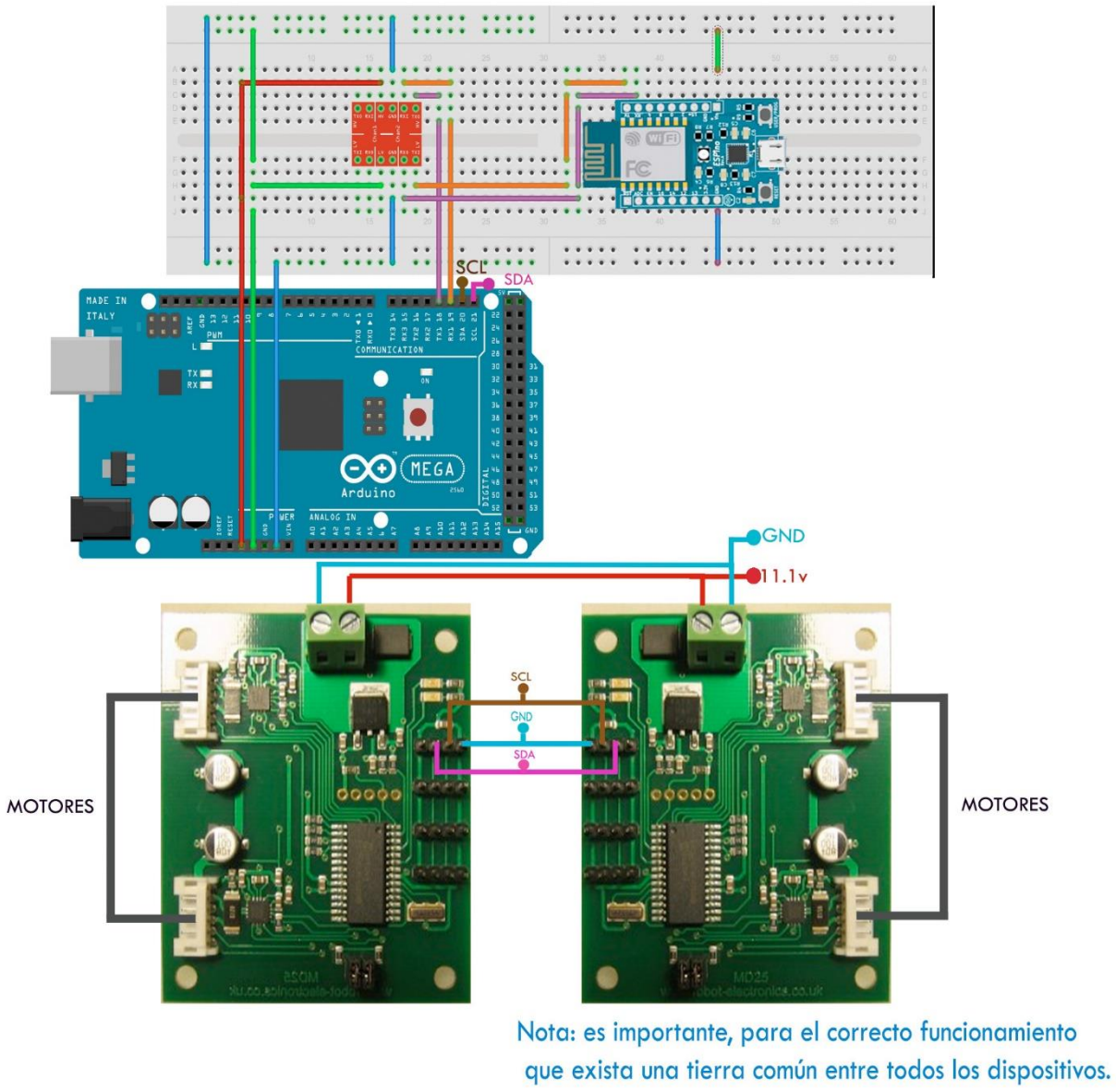


Figura 58 Diagrama de conexión Arduino- convertidor de niveles- ESPino.

6.4 CAPA DE INTELIGENCIA

Esta capa está compuesta por una computadora cuyo objetivo es el de procesar la información recabada en las distintas capas. El procesamiento de dicha información es llevado a cabo utilizando Simulink® de MATLAB®. A continuación, se hace una breve descripción de este software y se muestra el programa de bloques que se desarrolló para este trabajo.

MATLAB®

MATLAB® es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, visualización y programación. MATLAB® permite el análisis de datos, desarrollar algoritmos y crear modelos y aplicaciones. De la amplia gama de aplicaciones que se le pueden dar al software, destacan el procesamiento de señales y comunicaciones, procesamiento de imágenes y de vídeo, sistemas de control, prueba y medición, finanzas computacionales, y la biología computacional.

Simulink®

Simulink® es un entorno de diagrama de bloques para la simulación multidominio y diseño basado en modelos. Es compatible con el diseño a nivel de sistema, la simulación, la generación automática de código, y la prueba continua y verificación de sistemas embebidos. Simulink® ofrece un editor gráfico, bibliotecas de bloques personalizables y solucionadores para el modelado y simulación de sistemas dinámicos. Está integrado con MATLAB®, lo que le permite incorporar algoritmos de MATLAB® en modelos y resultados de la simulación de exportación a MATLAB® para su posterior análisis [48].

Como ya se había mencionado, la versión de MATLAB® que se utilizó fue la 2005Ra por ser la versión más actual en el momento en que se programó el prototipo. A continuación, se muestran los bloques que conforman el programa que se desarrolló para llevar a cabo las pruebas sobre el MMO.

Bloque de Visión

Este es el primer bloque (Figura 59) del programa y es quien recibe la posición y orientación de los fiduciales presentes en el espacio de visión de la cámara. Lo hace mediante un módulo Packet Input que obtiene la información que reacTIVision® manda a través de Processing, dicha información es acondicionada en el bloque ”3 Fiduciales”

(Figura 60, izquierda) para expresar la posición de tres marcadores fiduciales en pixeles. Finalmente estas posiciones pasan a un tercer bloque (Figura 60, derecha) que le aplica los ajustes mencionados en el Capítulo 6 para obtener la posición y orientación en cm y radianes, respectivamente, de los tres fiduciales.

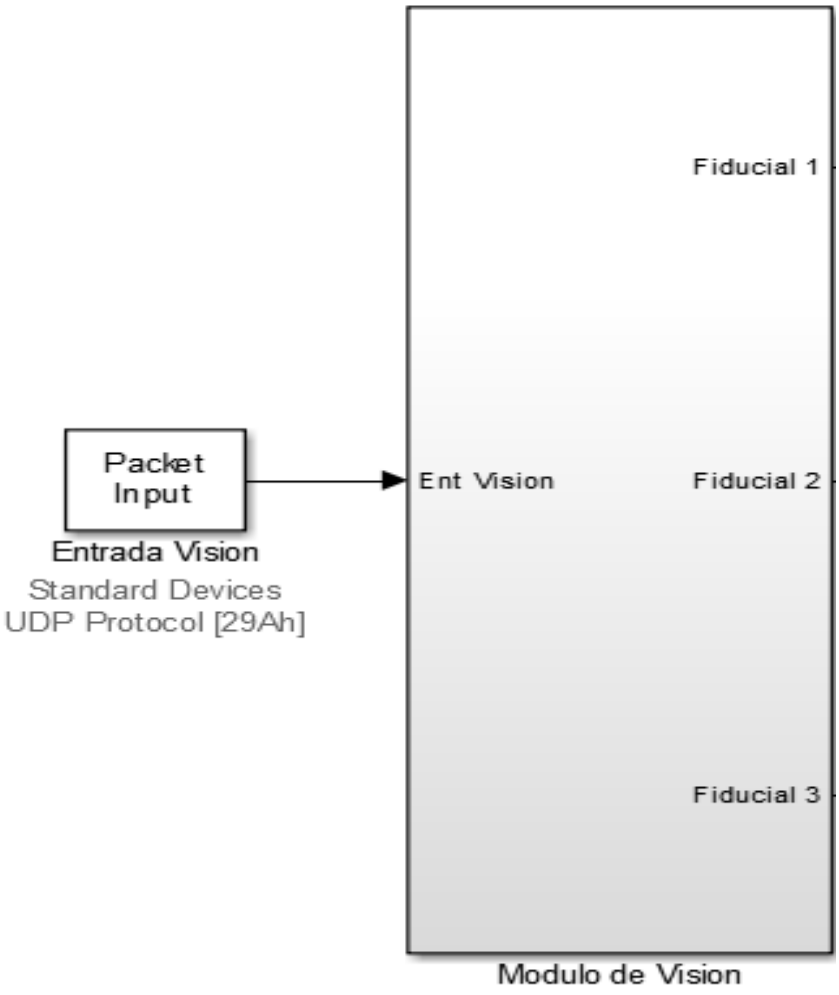


Figura 59 Bloque de visión.

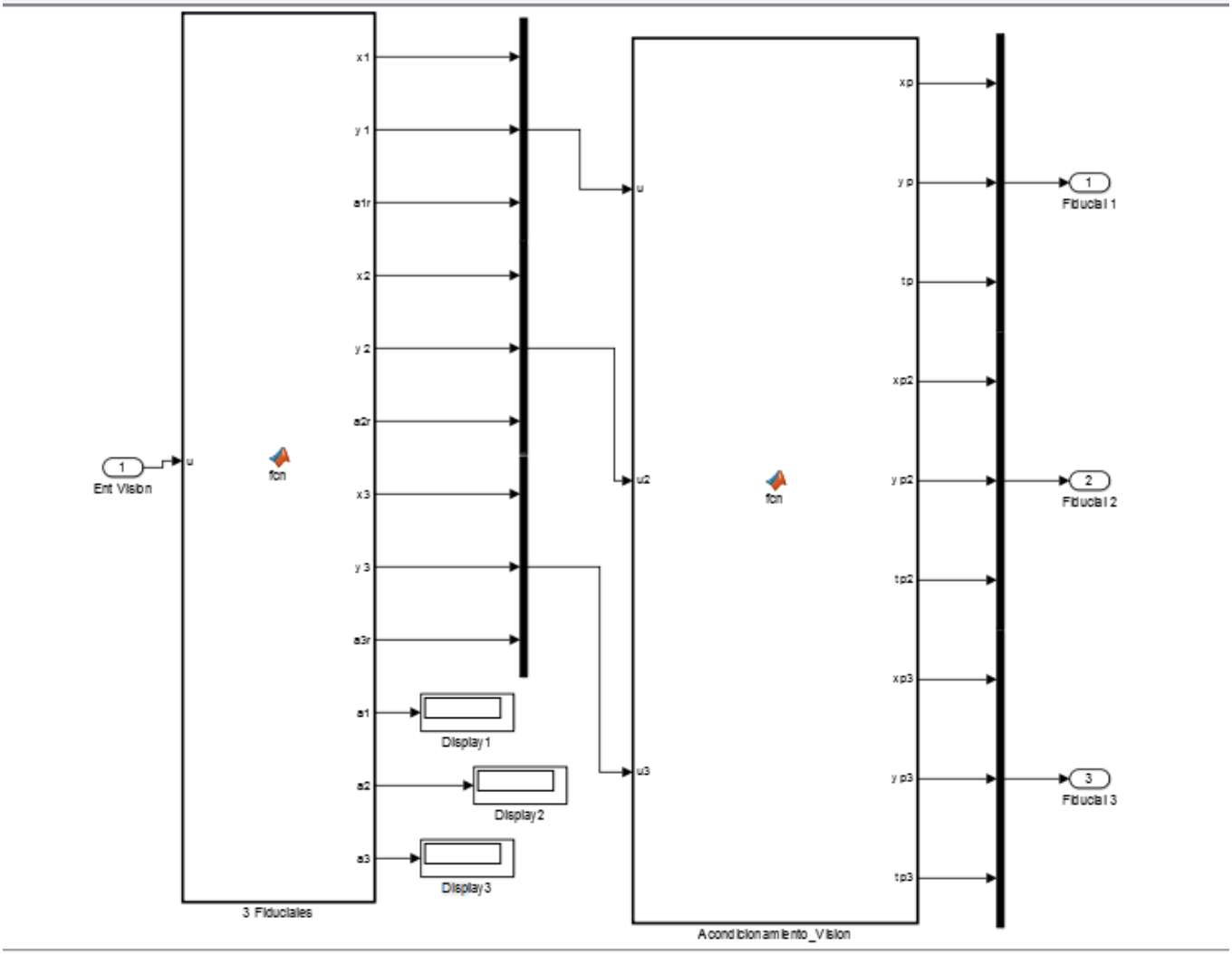


Figura 60 Bloque de acondicionamiento a pixeles (izquierda) y a centímetros (derecha).

Bloque de medición de distancia

Este bloque es el encargado de tomar en todo momento la posición del robot y de la meta para calcular la distancia existente entre ellos

Es por esto que tiene como entradas las posiciones de los fiduciales y como salidas la posición actual del robot, la posición siguiente que se espera que tome (posición de la meta) y la distancia entre ambas.

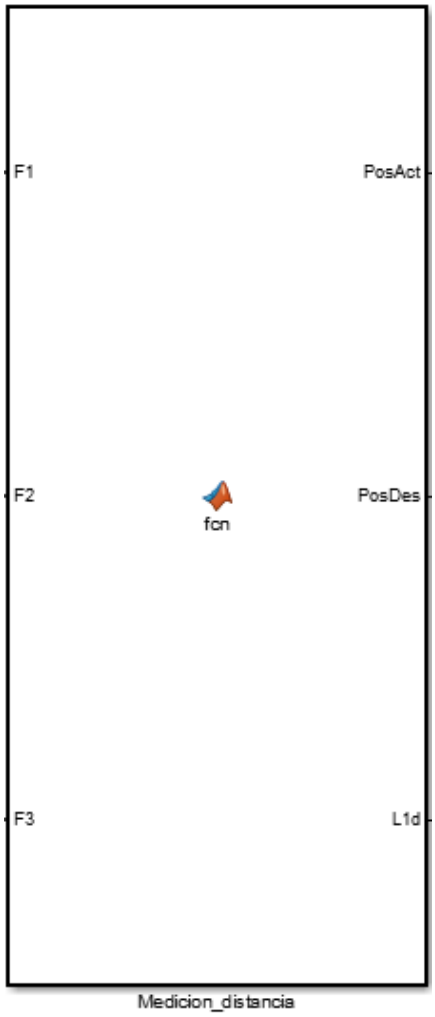


Figura 61 Bloque para medir distancia entre robot y meta.

Bloque de campos potenciales

Este bloque es el encargado de aplicar la técnica de campos potenciales que se trató en el Capítulo 4.

Tiene como entradas las salidas del bloque anterior, que corresponden a los datos de la posición del robot y de la meta; las salidas del bloque son la velocidad en la dirección del eje *x*, la velocidad en dirección del eje *y* y la velocidad angular para una rueda.

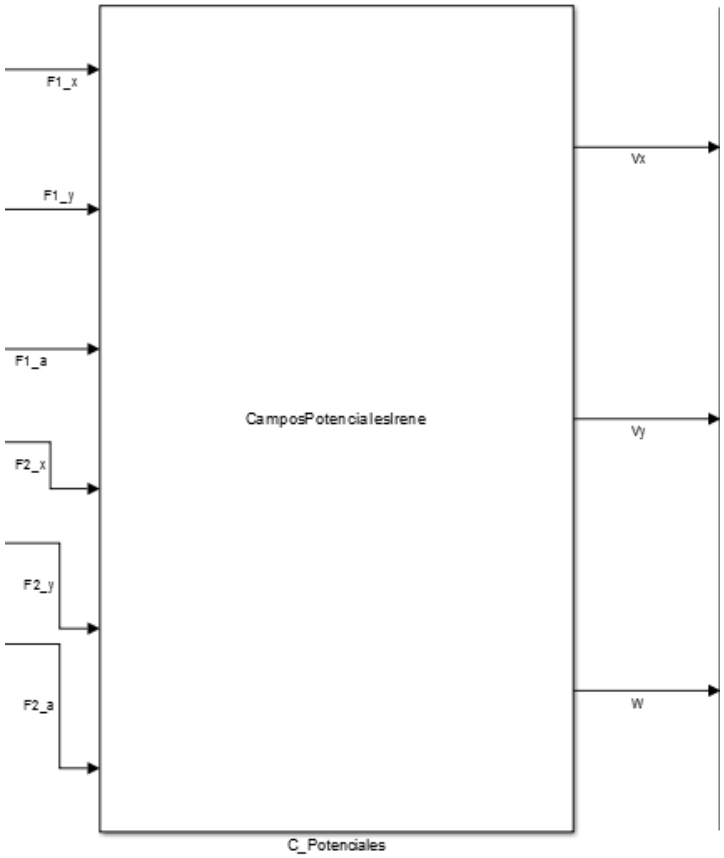


Figura 62 Bloque de campos potenciales.

Bloque cinemático

El bloque mostrado en la Figura 62, contiene las ecuaciones cinemáticas obtenidas mediante la técnica de propagación de velocidades vista en el subcapítulo 4.2, específicamente las desglosadas de la Ecuación 15.

Tiene como salidas las velocidades que deben aplicarse en cada una de las ruedas.

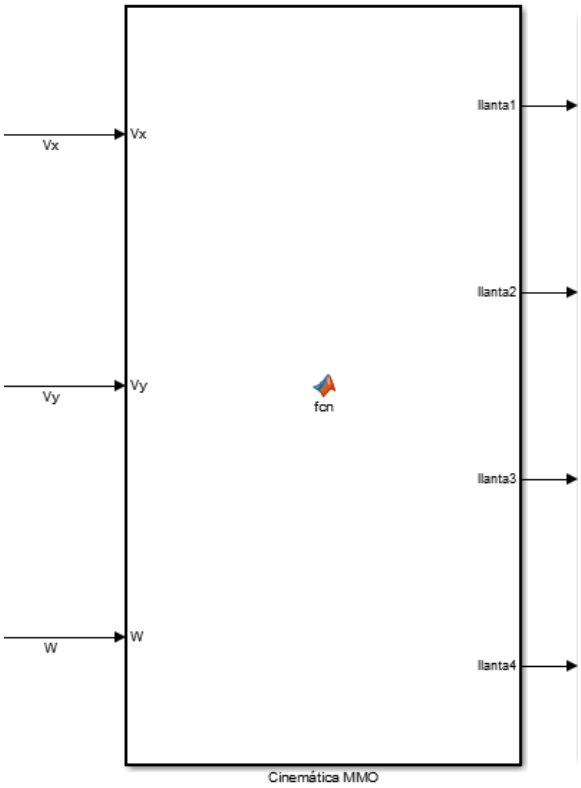


Figura 63 Bloque cinemático MMO.

Bloque de acondicionamiento de datos

Este bloque está destinado a convertir los valores obtenidos por la cinemática a valores que puedan ser enviados a las tarjetas MD25, es decir, números enteros sin signo que estén entre el intervalo de 0 a 255. Además, realiza un corrimiento sobre el número cero arrojado por la cinemática de modo que este sea el cero de la MD25 (128).

Lo anterior es contenido dentro del sistema en cuatro funciones idénticas, una para cada rueda, que pueden ser consultadas en el Anexo C.

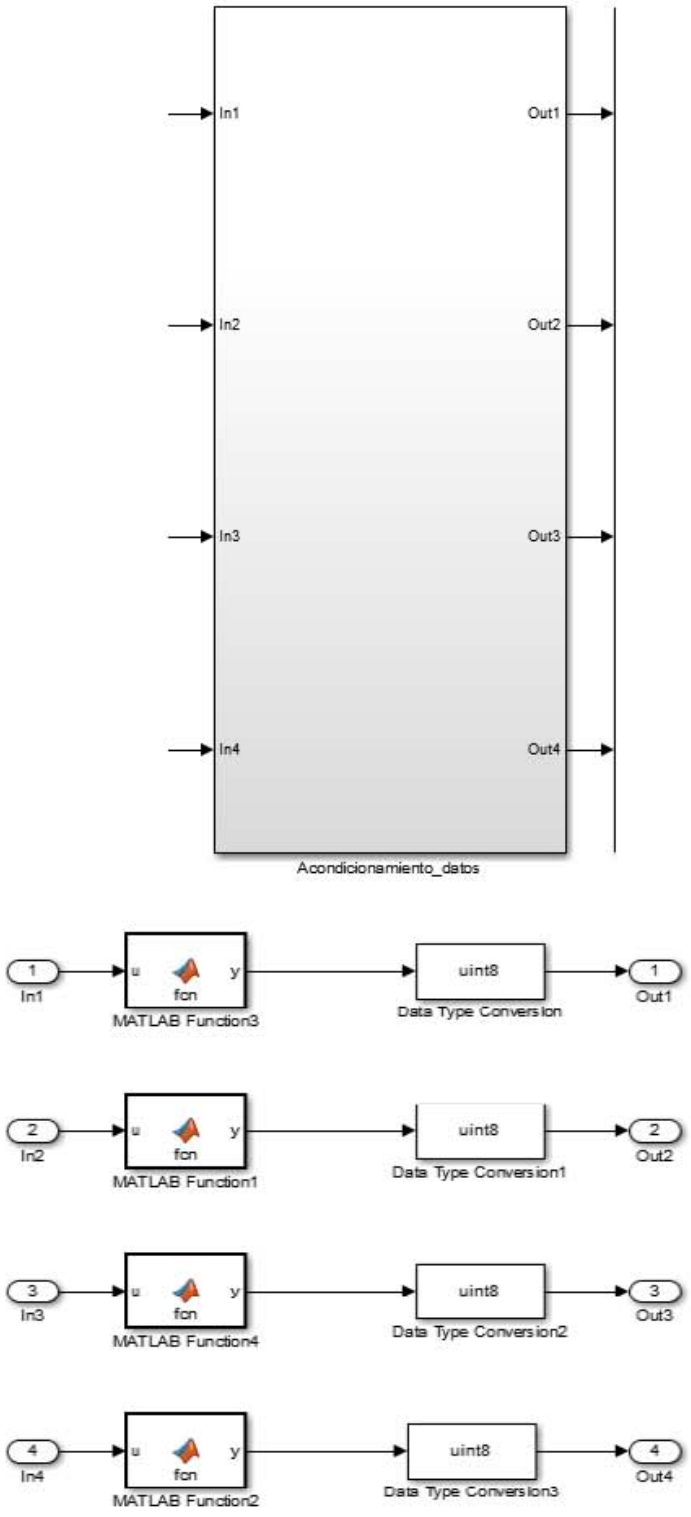


Figura 64 Bloque de acondicionamiento de datos.

Bloque de envío de datos

Similar al bloque que recibe los datos de la cámara, se tienen dos bloques Packet Output al final del programa, para enviar los datos de la simulación a los módulos ESPino. Específicamente se mandan cuatro datos, correspondientes a las velocidades de las cuatro ruedas, al módulo ESPino-Base y el dato de la distancia del MMO a la meta al módulo ESPino-Meta.

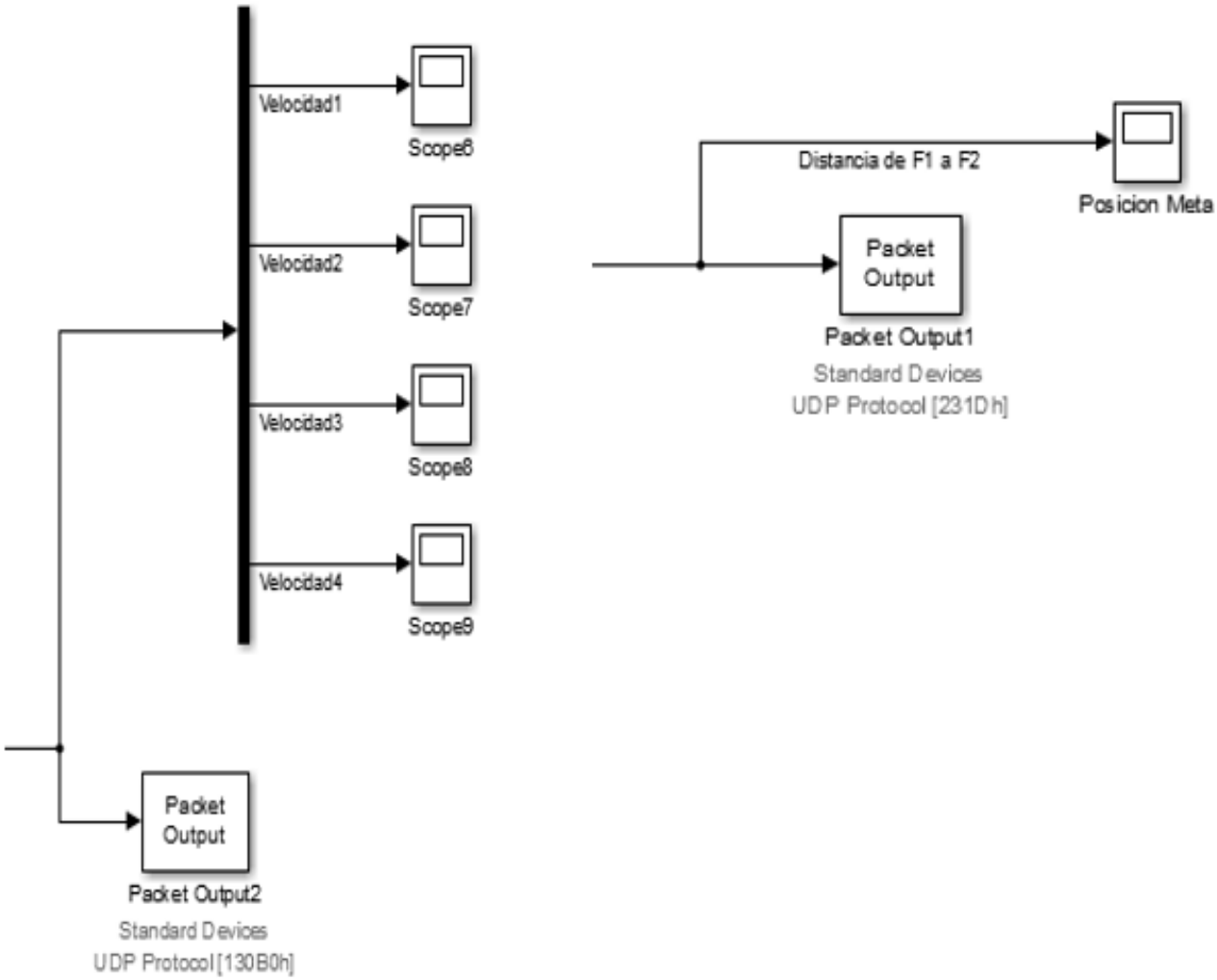


Figura 65 Envío de datos a la base (izquierda) y al brazo (derecha).

Cada uno de los módulos de salida de datos Packet Output están configurados con una IP y un puerto UDP, de modo que sólo puedan establecer comunicación con los módulos destinados a estos (Figura 66 y Figura 67).

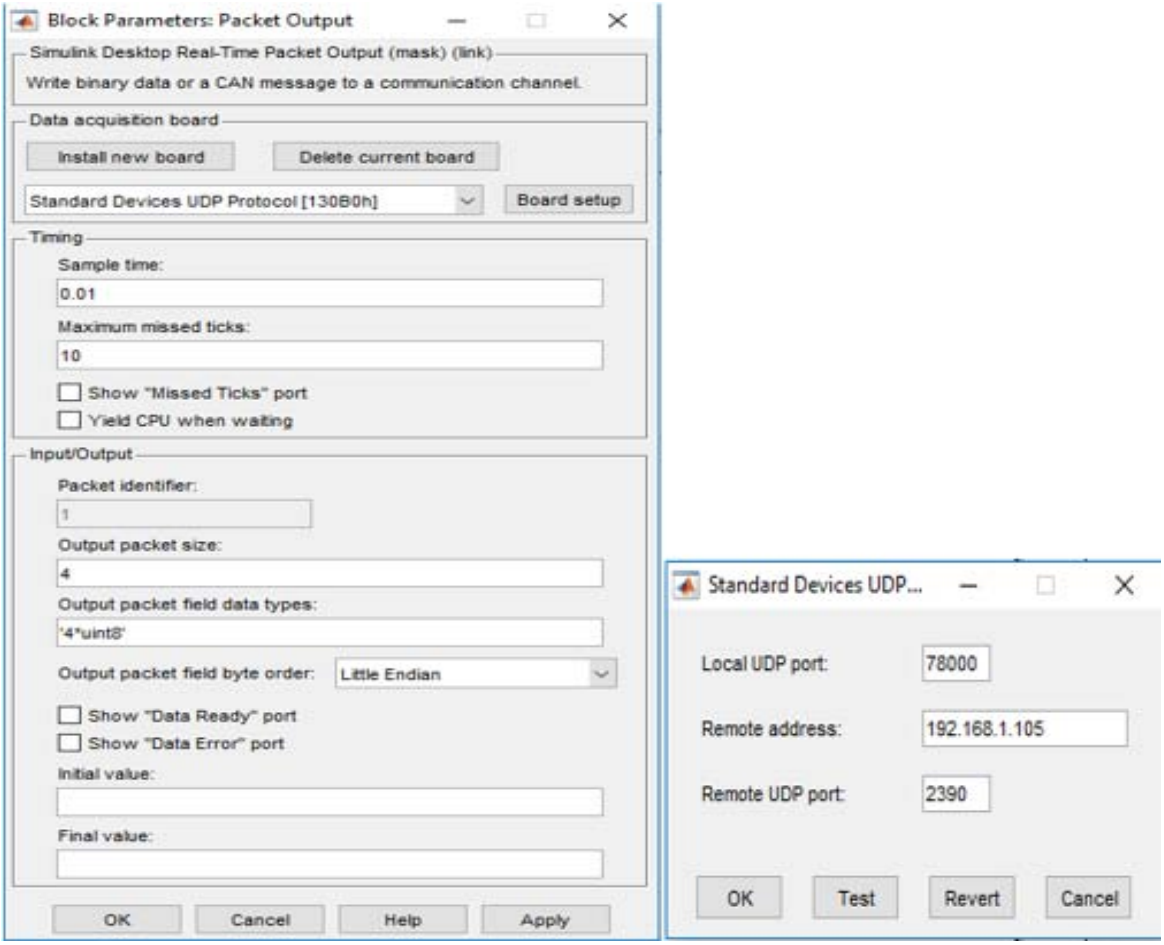


Figura 66 Configuración del bloque de salida de datos a la base.

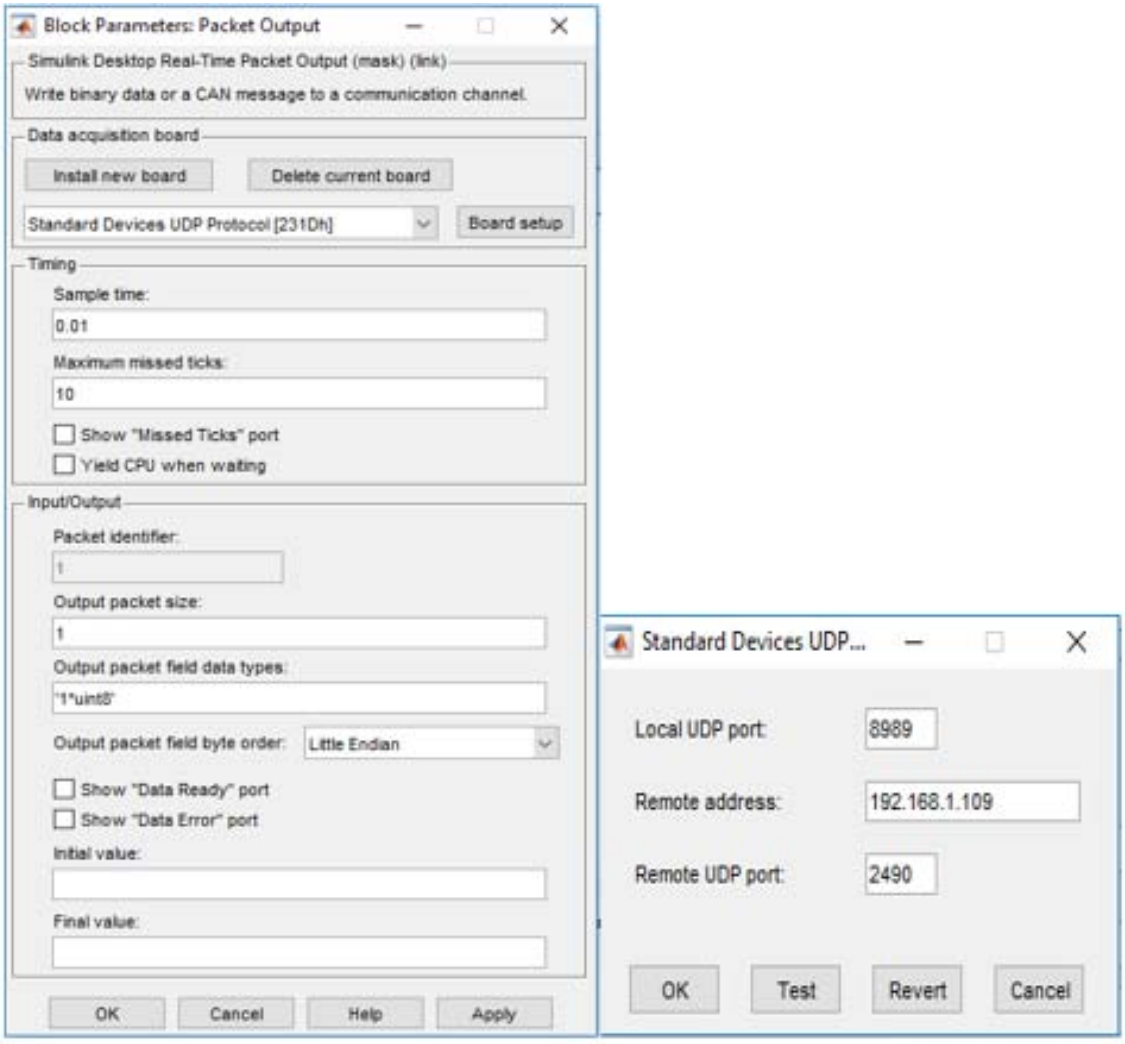


Figura 67 Configuración del bloque de salida de datos al manipulador.

Bloques de documentación

Se dispusieron varios bloques Scope en el programa, con el fin de recabar datos que se consideraron de interés para su documentación, Los ajustes realizados sobre estos se observan en la Figura 68.

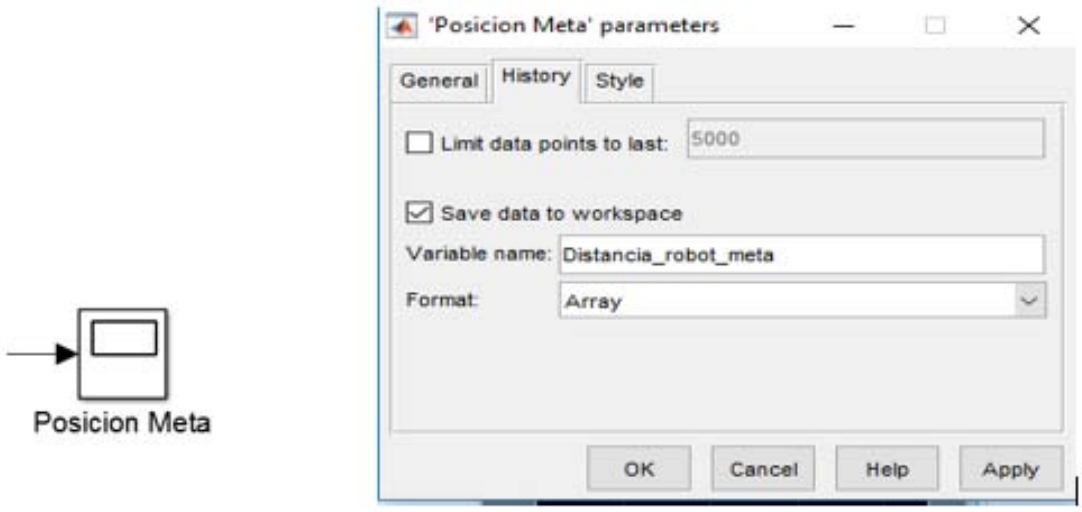


Figura 68 Bloques para documentación.

Diagrama general

En la Figura 69 se muestra el diagrama general de conexiones entre los bloques, identificados cada uno mediante colores para su fácil lectura.

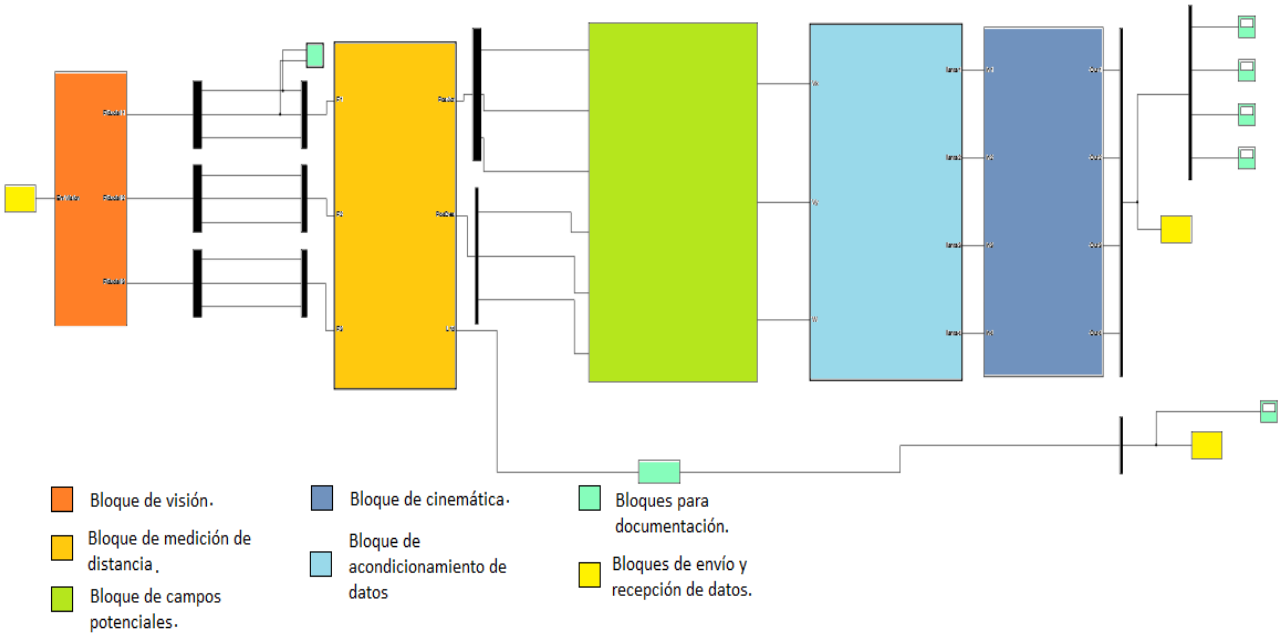


Figura 69 Diagrama general.

7 PRUEBAS, RESULTADOS Y ANÁLISIS

Como se mencionaba en el Capítulo 3, la tarea que se pretende realice el manipulador móvil omnidireccional es el transporte de materiales en un ambiente inteligente, para lo cual se colocó un objeto en una de las mesas meta y el MMO en alguna parte del ambiente inteligente.

Se llevaron a cabo distintas pruebas en las cuales el objetivo era que el MMO, sin importar la posición y orientación en que éste y la mesa fueran colocados, el primero lograra tomar el objeto de la mesa coordinando la base y el manipulador en función de la posición, de modo que el movimiento realizado fuera lo más suave posible.

7.1 PRUEBA 1: DIAGONAL NEGATIVA

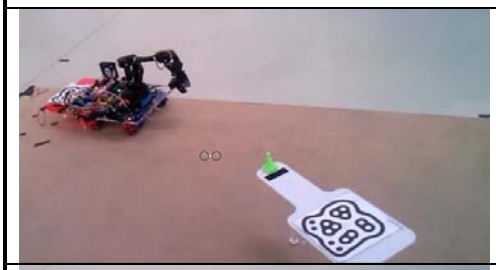
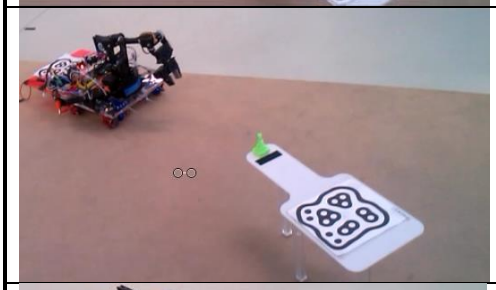
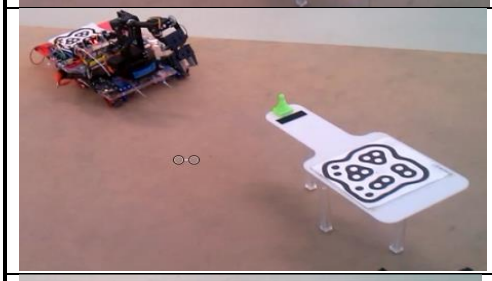
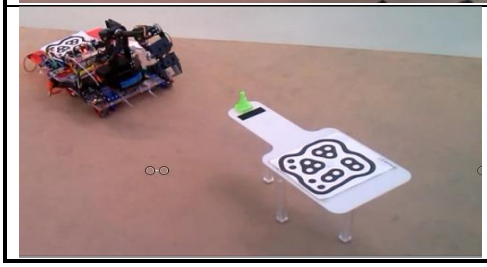
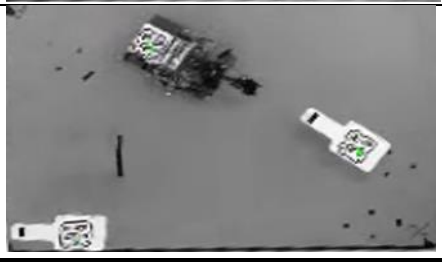
El objetivo de esta prueba fue que el MMO tomara el objeto de la mesa partiendo una orientación de 0° respecto a un eje horizontal dirigido hacia la derecha, mientras que la mesa se encontrara rotada 318° tomando la misma referencia.

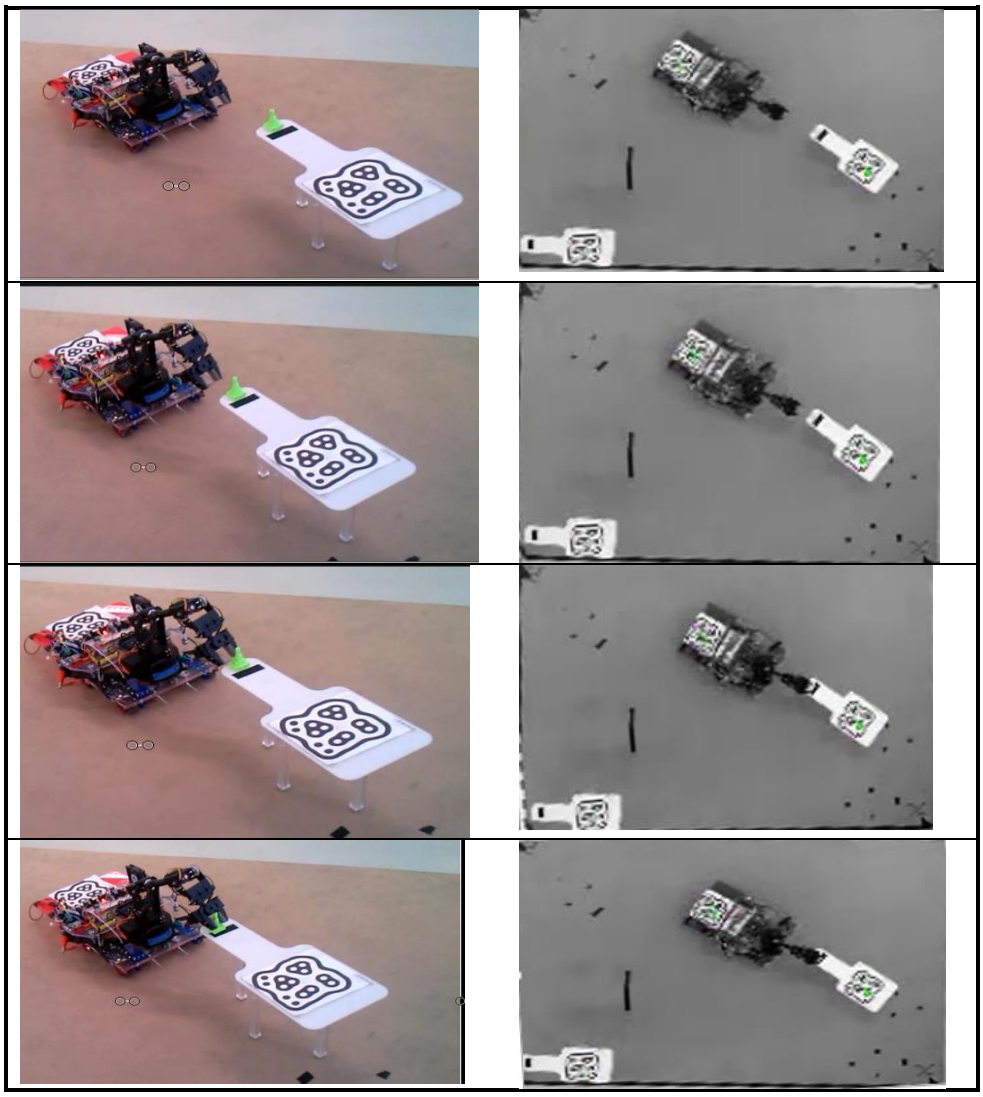
A lo largo de la tarea se recabaron datos de importancia, los cuales se expresan en las siguientes tablas, imágenes y gráficas. El MMO llevó a cabo la tarea sin contratiempo y de manera satisfactoria como se verá a continuación.

7.1.1 Trayectoria punto a punto

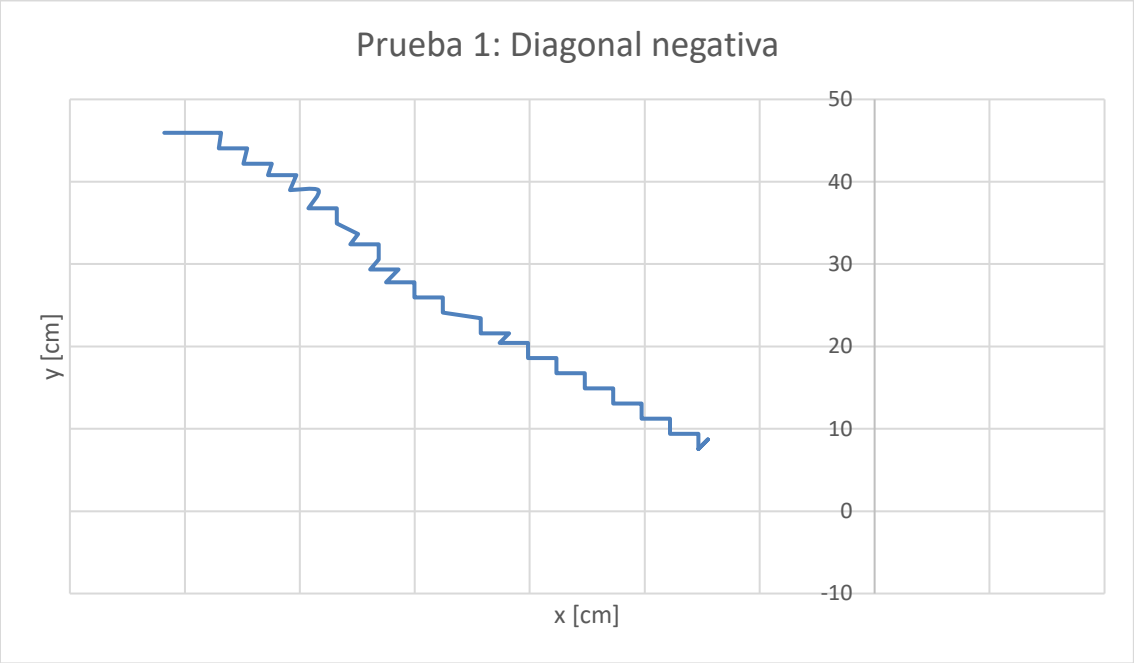
La siguiente tabla muestra la trayectoria que realizó el MMO para llegar a la meta, a la derecha se pueden observar las imágenes tomadas con la cámara, colocada de forma perpendicular al piso, mientras que las de la izquierda fueron tomadas por un observador y ofrecen otra perspectiva en la que el brazo puede ser apreciado en mayor detalle.

Tabla 8 Trayectoria del MMO a la meta, prueba 1.

Observador	reacTIVision®
	
	
	
	



A lo largo de la prueba, como ya se mencionó, se tomaron datos considerados de gran importancia, de entre los que destacan las coordenadas, tanto en *x* como en *y,* de la posición del MMO y de la mesa, la cual se mantuvo estática durante toda la prueba. Estos datos fueron recabados en la Gráfica 7, en la cual se puede apreciar en color azul la trayectoria que el MMO realizó para poder llegar a la mesa representada por un punto rojo.



Gráfica 7. Posicionamiento del MMO, prueba 1.

Tabla 9 Posiciones finales, MMO y meta, prueba 1.

Objeto	Coordenada en x, en cm	Coordenada en y, en cm	Ángulo, en rad
MMO	-15.32853	7.54874	5.63732
Meta	8.85229	-4.62138	5.56751

Utilizando los últimos datos de posición del MMO y la meta, mostrados en la Tabla 9, se calculó la distancia entre ambos mediante la Ecuación 36, dando como resultado una distancia de 27 cm, la cual debe guardarse siguiendo lo establecido en el método de capos potenciales (subcapítulo 4.4).

$$d = \sqrt{\left(x_{f-meta} - x_{f-MMO}\right)^2 + \left(y_{f-meta} - y_{f-MMO}\right)^2}$$

Ecuación 36

$$d = \sqrt{(8.8522 - (-15.3285))^2 + (-4.6213 - 7.5487)^2} = 27.07055\text{cm}$$

Ecuación 37

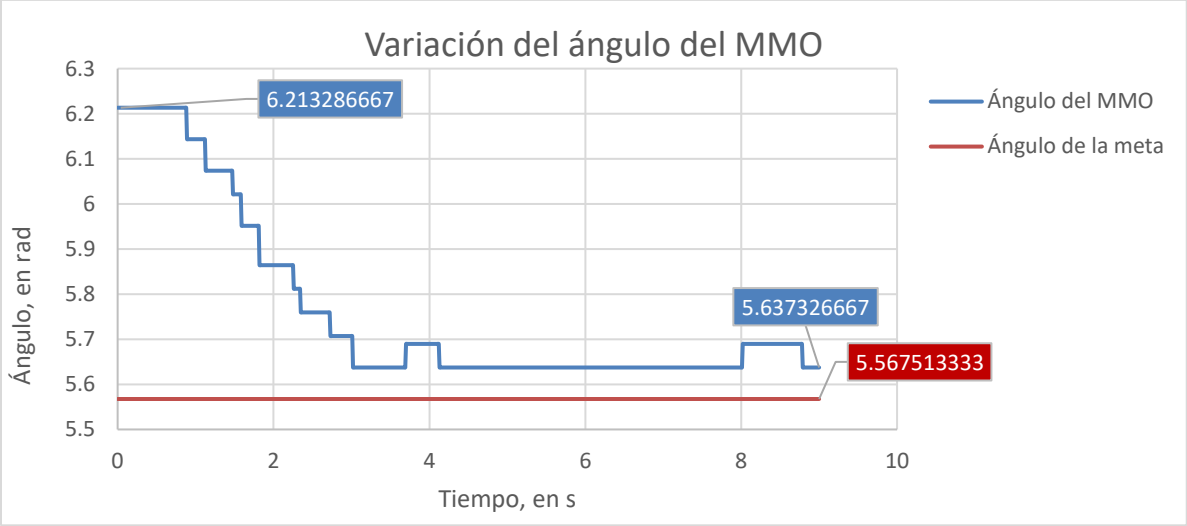
7.1.2 Porcentaje de error del ángulo final

Similar a lo que se realizó con el seguimiento de la trayectoria punto a punto, se graficaron los ángulos que el MMO fue tomando a medida que avanzaba hacia la mesa. Como se observa, el ángulo final del MMO fue de 5.6373 radianes o 322°, siendo el de la meta (que se mantuvo estática a lo largo de la prueba) de 5.5675 radianes o 318°.

$$\%error\ de\ ángulo = |ángulo_{meta} - ángulo_{MMO}| * 100$$

Ecuación 38

El porcentaje de error entre dichos ángulos fue calculado usando la Ecuación 38, y fue del 6.98% el cual no afectó en absoluto la realización de la tarea.

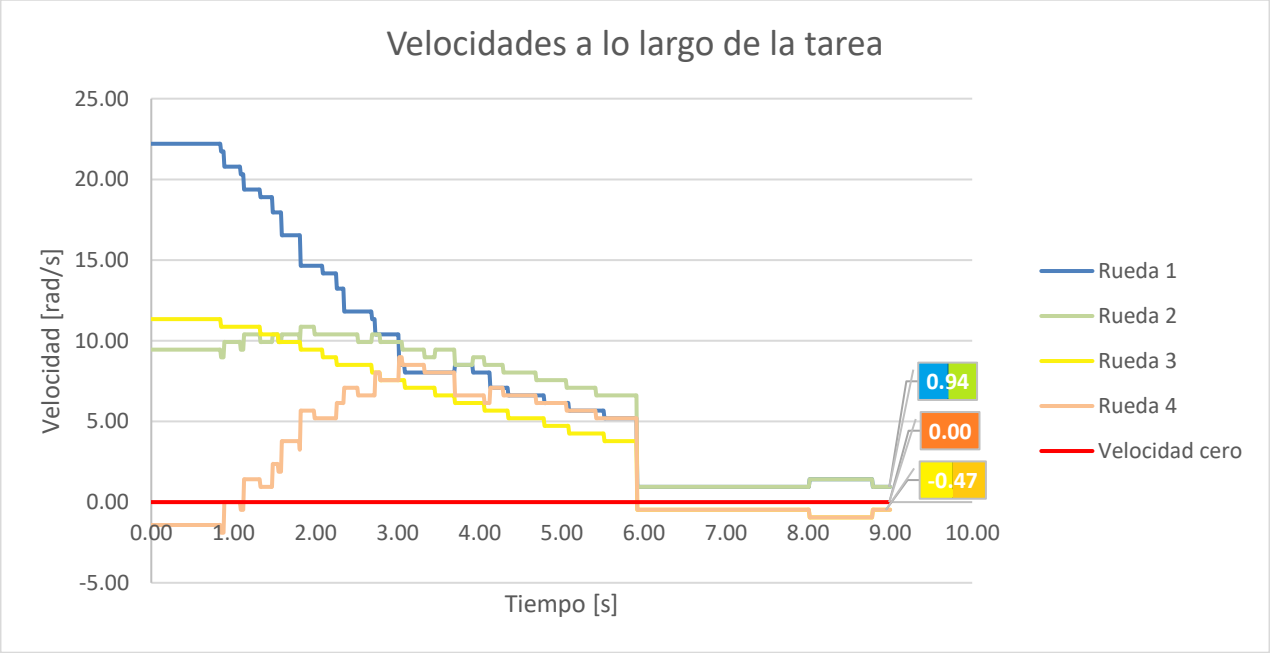


Gráfica 8. Variación de ángulo del MMO, prueba 1.

7.1.3 Variación de las velocidades del MMO

Finalmente, en la Gráfica 9 se muestran los cambios de velocidad que experimentó cada rueda a lo largo de la tarea. La línea en color rojo representa el valor de 128, reconocido por la tarjeta MD25 como cero o velocidad nula, valor al cual las cuatro ruedas van

convergiendo a medida que el MMO se acerca a la meta (o dicho de otra manera, a medida que transcurre el tiempo de la prueba, eje *y*).



Gráfica 9. Velocidades a lo largo de la tarea, prueba 1.

7.2 PRUEBA 2: DIAGONAL POSITIVA

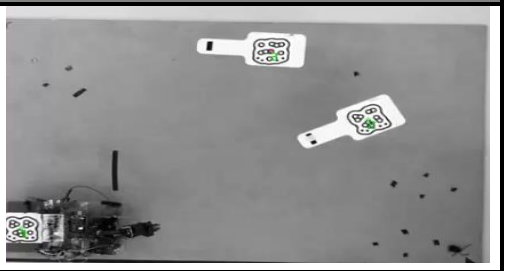
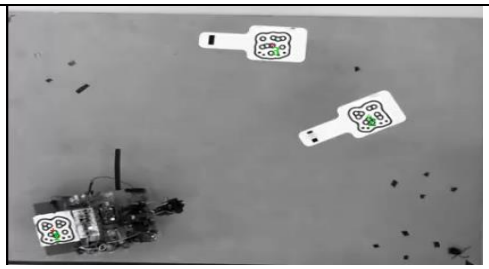
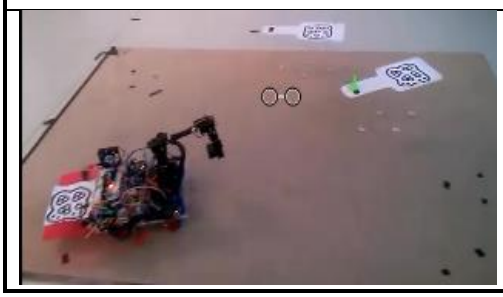
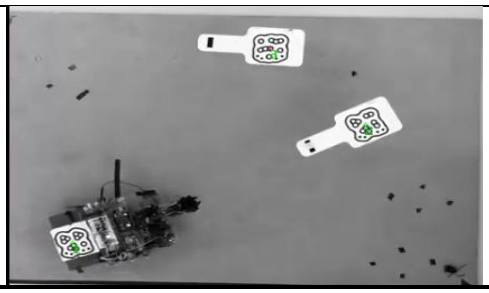
Similar a la prueba anterior, se orientó a 0° al MMO, respecto al eje horizontal dirigido hacia la derecha, y la mesa a 25° tomando la misma referencia. El objetivo fue que el MMO tomara el objeto colocado en la mesa meta.

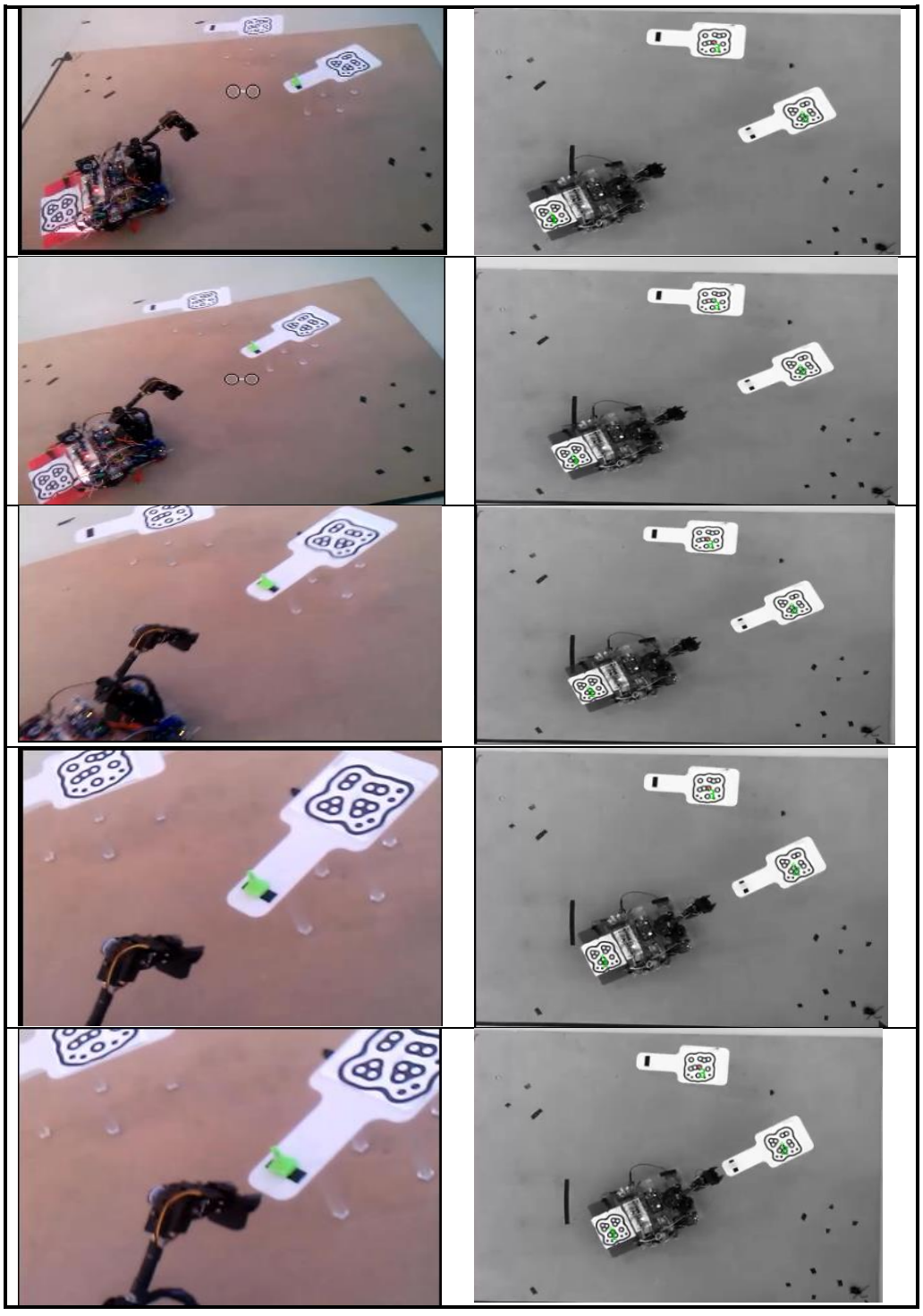
A continuación se muestran tablas, imágenes y gráficas que se construyeron a partir de los datos recabados durante la prueba, los cuales son analizados más adelante.

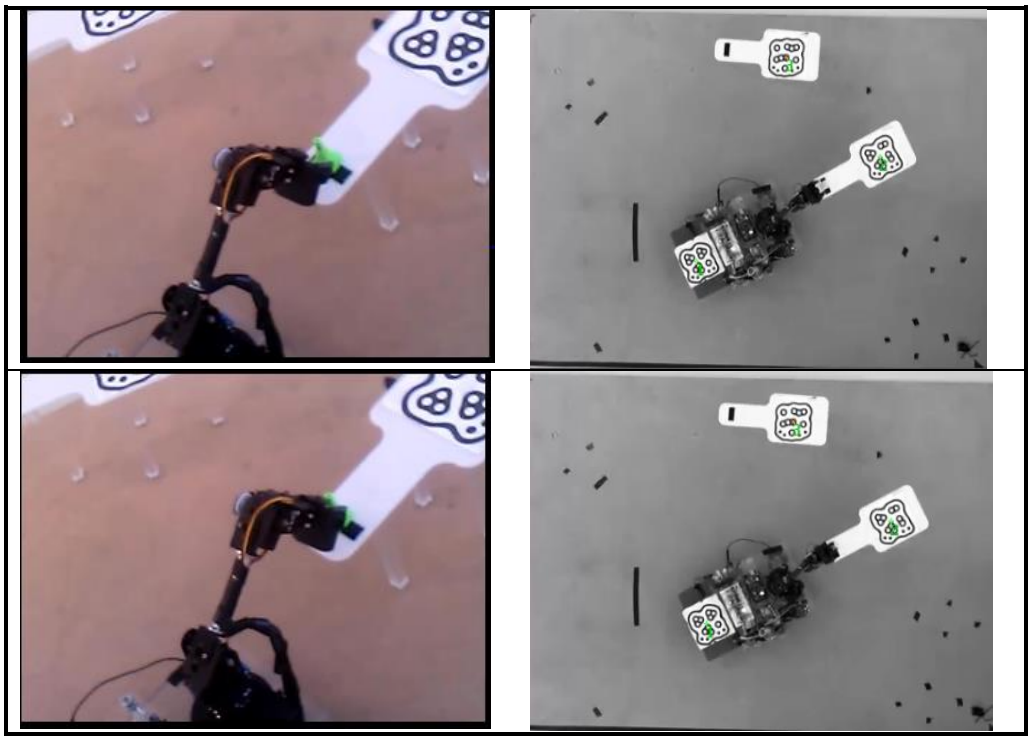
7.2.1 Trayectoria punto a punto

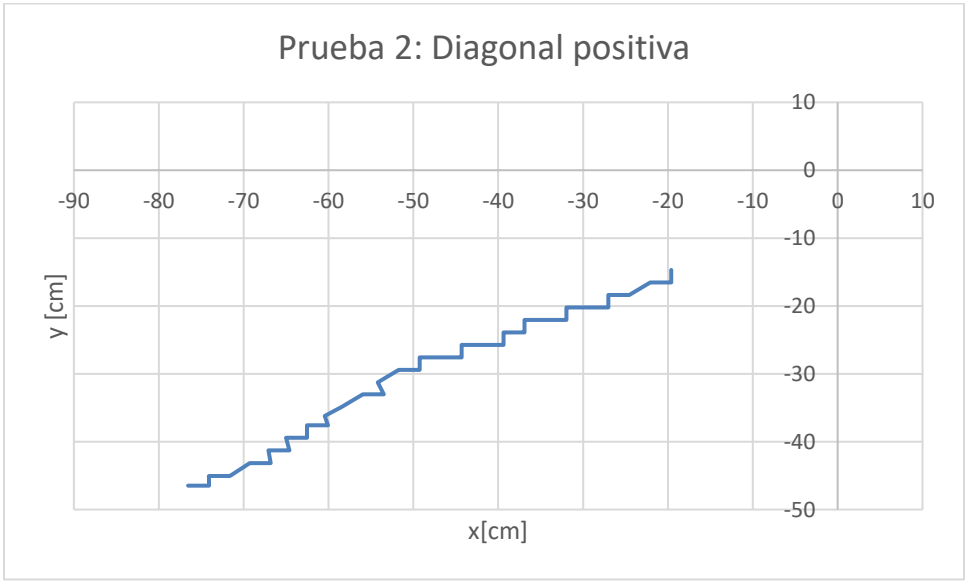
Se realizó el seguimiento de la trayectoria que el MMO hizo para llegar de su posición inicial a la posición esperada. Dicha trayectoria puede ser observada en las imágenes de la Tabla 10 y en la Gráfica 10. En ambas se aprecia que la tarea fue llevada a cabo de manera satisfactoria.

Tabla 10 Trayectoria realizada por el MMO, prueba 2.

Observador	reacTIVision®
	
	
	







Gráfica 10.Posicionamiento del MMO, prueba 2.

Tabla 11 Posiciones finales, MMO y meta, prueba 2.

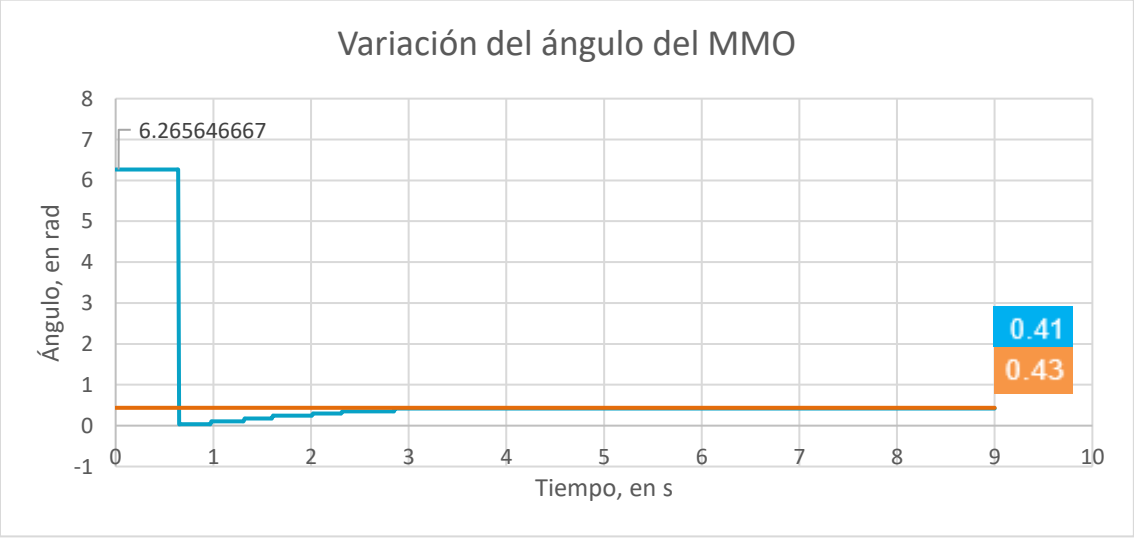
Objeto	Coordenada en x, en cm	Coordenada en y, en cm	Ángulo, en rad
MMO	-19.61046	-14.69922	0.41878
Meta	3.196658	0.32690	0.43623

En la tabla anterior se muestran las coordenadas finales tanto del MMO como de la mesa y el ángulo final de ambos. Utilizando la Ecuación 36 se calculó la distancia a la cual se detuvo el MMO de la meta, que fue de 27.31 cm, que al igual que en la prueba anterior, es la distancia que debe guardarse siguiendo el método de campos potenciales.

7.2.2 Porcentaje de error del ángulo final

En la Gráfica 11 se muestra en azul cómo fue variando el ángulo del MMO a medida que iba realizando la tarea, y en naranja el ángulo, que se mantuvo constante, de la meta.

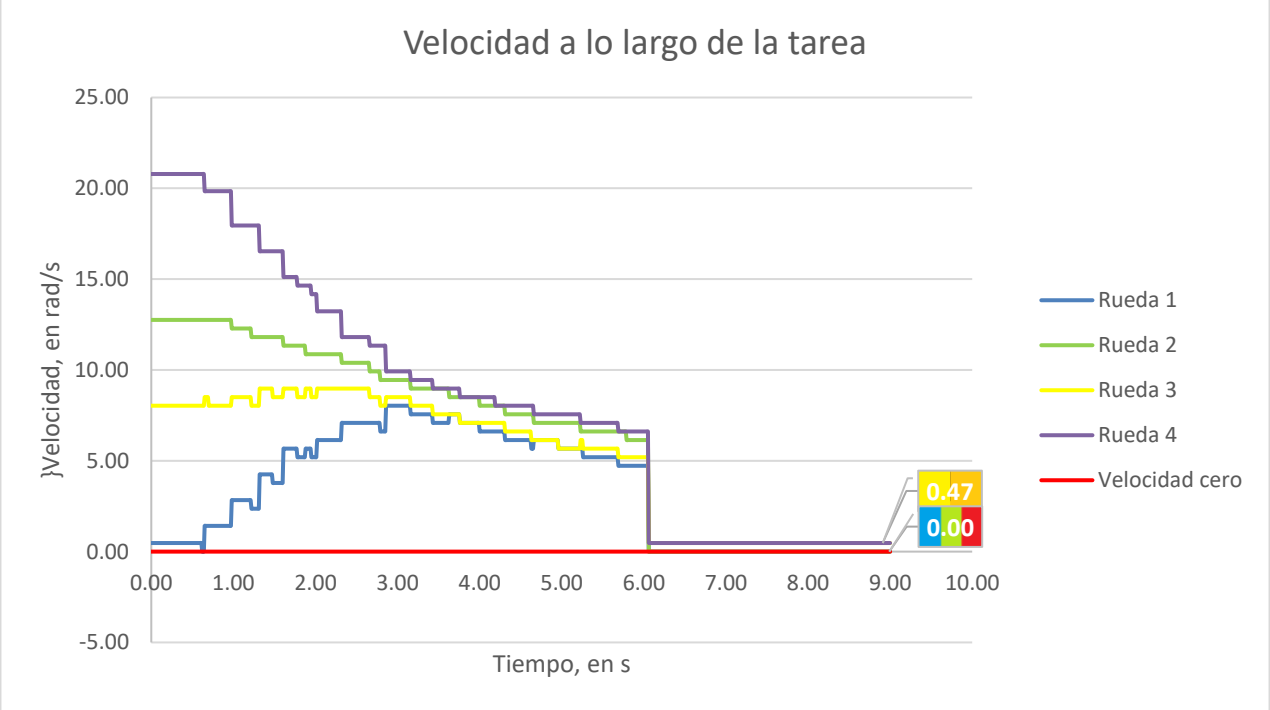
Utilizando la Ecuación 38 se calculó el porcentaje de error, el cual resultó en 1.74%, error que no influyó de manera significativa en la realización de la tarea.



Gráfica 11. Variación de ángulo del MMO, prueba 2.

7.2.3 Variación en la velocidad

Similar a la prueba anterior, se esperaba que a medida que el MMO fuera acercándose a la meta, la velocidad de cada una de las ruedas convergiera al valor 128, velocidad nula para la tarjeta MD25. Las ruedas 1 y 2 alcanzaron esta velocidad, mientras que la 3 y 4 llegaron al valor de 127, esto debido al error existente en el ángulo de llagada, es decir, el MMO no se detiene por completo ya que trata de orientarse con el mismo ángulo de la meta. Sin embargo, se da por terminada la tarea a pesar de no llegar al valor esperado, ya que el error es mínimo y no afecta en el cumplimiento de la tarea, ni de la base ni del brazo.



Gráfica 12. Velocidades a lo largo de la tarea, prueba 2.

8 CONCLUSIONES Y TRABAJO A FUTURO

Tras realizar varias pruebas se pudo llegar a las siguientes conclusiones:

1 Se logró diseñar un espacio inteligente que incluyera e identificara al MMO y a la mesa como tal y se sirviera del primero utilizándolo como actuador para la realización del trasporte de materiales.

2 Se dio solución al problema de coordinar la base y el brazo sumando las ventajas que cada uno de los robots posee para llevar a cabo la tarea propuesta, dicha coordinación pudo ser expresada como un movimiento sincronizado entre base y brazo, de tal forma que el brazo partió de una configuración inicial y fue posicionándose hasta llegar a una final, a medida en que la plataforma se iba acercando a la meta.

3 Tras analizar los resultados obtenidos, se puede concluir también que no importa cómo se coloquen el robot y la mesa, el primero siempre llegará a la meta guardando una distancia de seguridad, que tenderá a ser la misma (27 cm) siguiendo la técnica de campos potenciales. Se presentaron en las dos pruebas reportadas porcentajes de error del 6.98% y 1.74%, errores que no afectaron significativamente el cumplimiento de la tarea. El efecto de estos errores se muestra en las Gráficas 3 y 6, ya que algunas ruedas no se detuvieron por completo al llegar al área de seguridad entre el MMO y la mesa debido a que el primero trataba de orientarse con el mismo ángulo que el del objetivo.

Como trabajo a futuro, se sugiere mejorar la comunicación WiFi, específicamente en el acomodo de datos recibidos, de tal modo que se pueda utilizar un solo módulo de comunicación WiFi ESPino y un sólo Arduino para el control total del robot.

También, es importante que sobre el prototipo puedan ser probados otros tipos de coordinación entre la base y el brazo, específicamente un modelo completo del robot en el cual definida la trayectoria por la que pasará el efector final en el espacio cartesiano, se obtengan las velocidades necesarias en el espacio articular, con objeto de cumplir con dicha trayectoria.

ANEXOS

ANEXO A: VECTOR DE LA BASE DEL BRAZO MANIPULADOR AL EFECTOR FINAL

$$\begin{bmatrix} \left(\begin{array}{l} \text{Sin}[\theta 1]\ \left(-12.\ \text{Sin}[\theta 2]\ \left(\text{Cos}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right]\ \text{Sin}\left[\frac{25\,\pi}{102} - \theta 3\right] + \text{Cos}\left[\frac{25\,\pi}{102} - \theta 3\right]\ \left(-1.19167 + \text{Sin}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right]\right)\right) + 12.\ \text{Cos}[\theta 2]\ \left(-0.766667 + \text{Cos}\left[\frac{25\,\pi}{102} - \theta 3\right]\ \text{Cos}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right] + \text{Sin}\left[\frac{25\,\pi}{102} - \theta 3\right]\ (1.19167 \\ \text{Cos}[\theta 1]\ \left(12.\ \text{Sin}[\theta 2]\ \left(\text{Cos}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right]\ \text{Sin}\left[\frac{25\,\pi}{102} - \theta 3\right] + \text{Cos}\left[\frac{25\,\pi}{102} - \theta 3\right]\ \left(-1.19167 + \text{Sin}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right]\right)\right) - 12.\ \text{Cos}[\theta 2]\ \left(-0.766667 + \text{Cos}\left[\frac{25\,\pi}{102} - \theta 3\right]\ \text{Cos}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right] + \text{Sin}\left[\frac{25\,\pi}{102} - \theta 3\right]\ (1.19167 \\ 13.\ + 14.3\ \text{Cos}[\theta 2]\ \text{Cos}\left[\frac{25\,\pi}{102} - \theta 3\right] + 9.2\ \text{Sin}[\theta 2] - 14.3\ \text{Sin}[\theta 2]\ \text{Sin}\left[\frac{25\,\pi}{102} - \theta 3\right] + 12\ \left(-\text{Cos}\left[\frac{25\,\pi}{102} - \theta 3\right]\ \left(\text{Cos}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right]\ \text{Sin}[\theta 2] + \text{Cos}[\theta 2]\ \text{Sin}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right]\right) + \text{Sin}\left[\frac{25\,\pi}{102} - \theta 3\right]\ \left(-\text{Cos}[\theta 2]\ \text{Cos}\left[\frac{1}{9}\ (\end{array}\right) \\ \left(\begin{array}{l} -0.34202\ \text{Cos}[\theta 4] + 0.939693\ \text{Sin}[\theta 4]\right)\Big) \\ -0.34202\ \text{Cos}[\theta 4] + 0.939693\ \text{Sin}[\theta 4]\Big)\Big) \\ (\pi - 9\,\theta 4)\Big] + \text{Sin}[\theta 2]\ \text{Sin}\left[\frac{1}{9}\ (\pi - 9\,\theta 4)\right]\Big)\Big) \end{array}\right) \end{bmatrix}$$

ANEXO B INSTALACIÓN DE LA PLACA ESPINO EN EL IDE DE ARDUINO

Como se mencionó anteriormente, se utilizó el IDE de Arduino para la programación del ESPino a continuación se muestran los pasos a seguir para que este IDE soporte al ESP8266.

Paso 1. Lo primero que hay que hacer es instalar la placa; para esto es necesario ir al IDE de Arduino y seleccionar la opción “preferencias” localizada en la pestaña “Archivo”, tal cual se muestra en la siguiente figura.

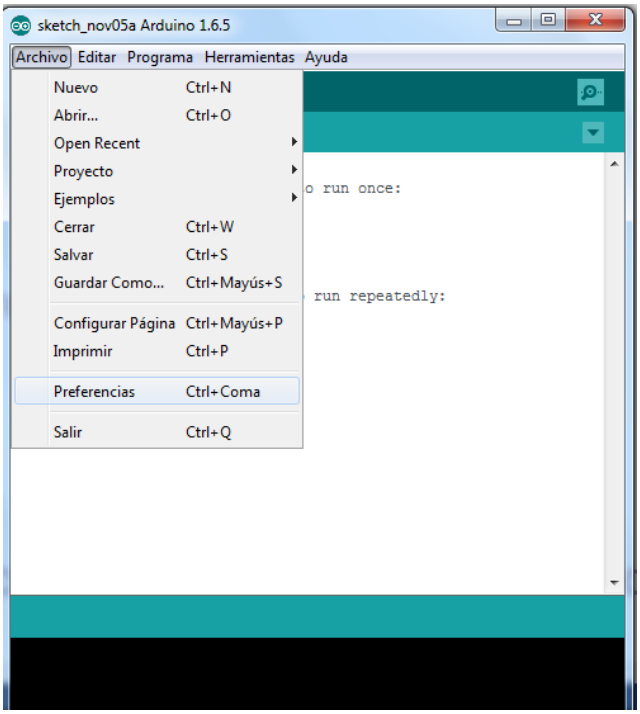


Figura 70 Arduino IDE.

Pegar el siguiente link (sin comillas):

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Justo en el cuadro de texto de “Additional Boards Manager URLs”, Figura 71, y seleccionar Ok.

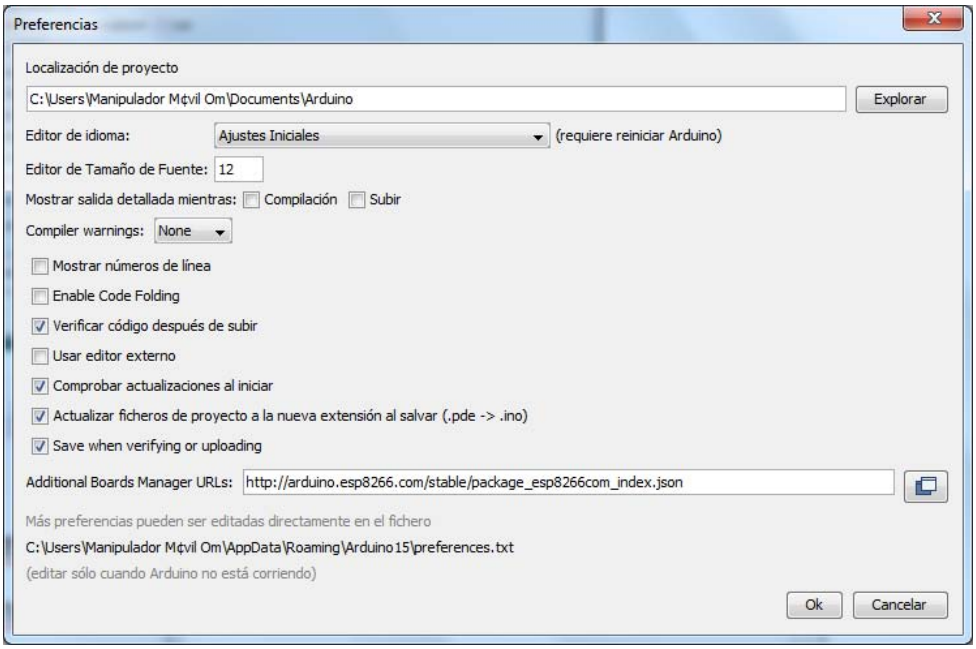


Figura 71 Arduino IDE, preferencias.

Paso 2. Una vez pegada la liga, hay que seleccionar “Board Manager” en la opción Placa de la pestaña de Herramientas, como se muestra en la siguiente imagen.

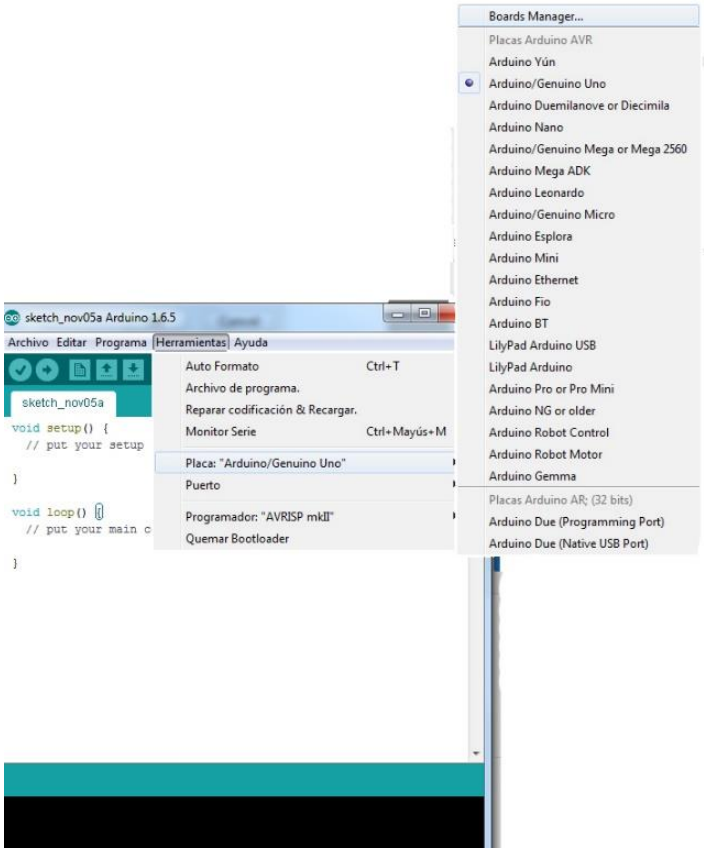


Figura 72 Boards manager.

Ya en esta opción, se instala el paquete del ESP8266 seleccionando la opción “esp8266 by ESP8266 Community”.

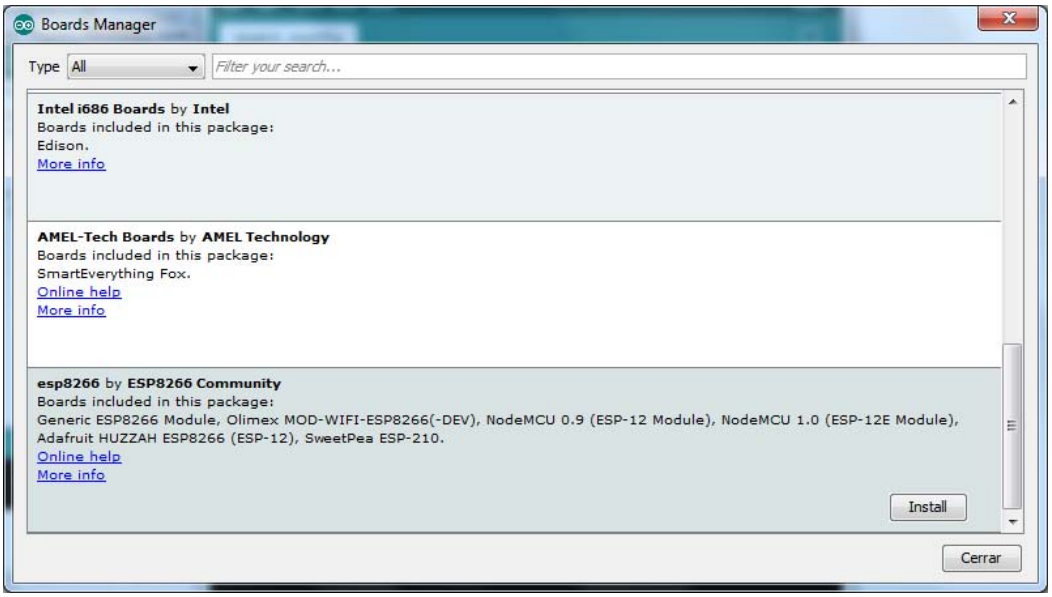


Figura 73 Instalación del ESP8266.

Paso 3. Finalmente, se verifica que en la sección de placas se encuentre la “Generic ESP8266 Module”, los ajustes de la placa dependen de las aplicaciones que se le den.

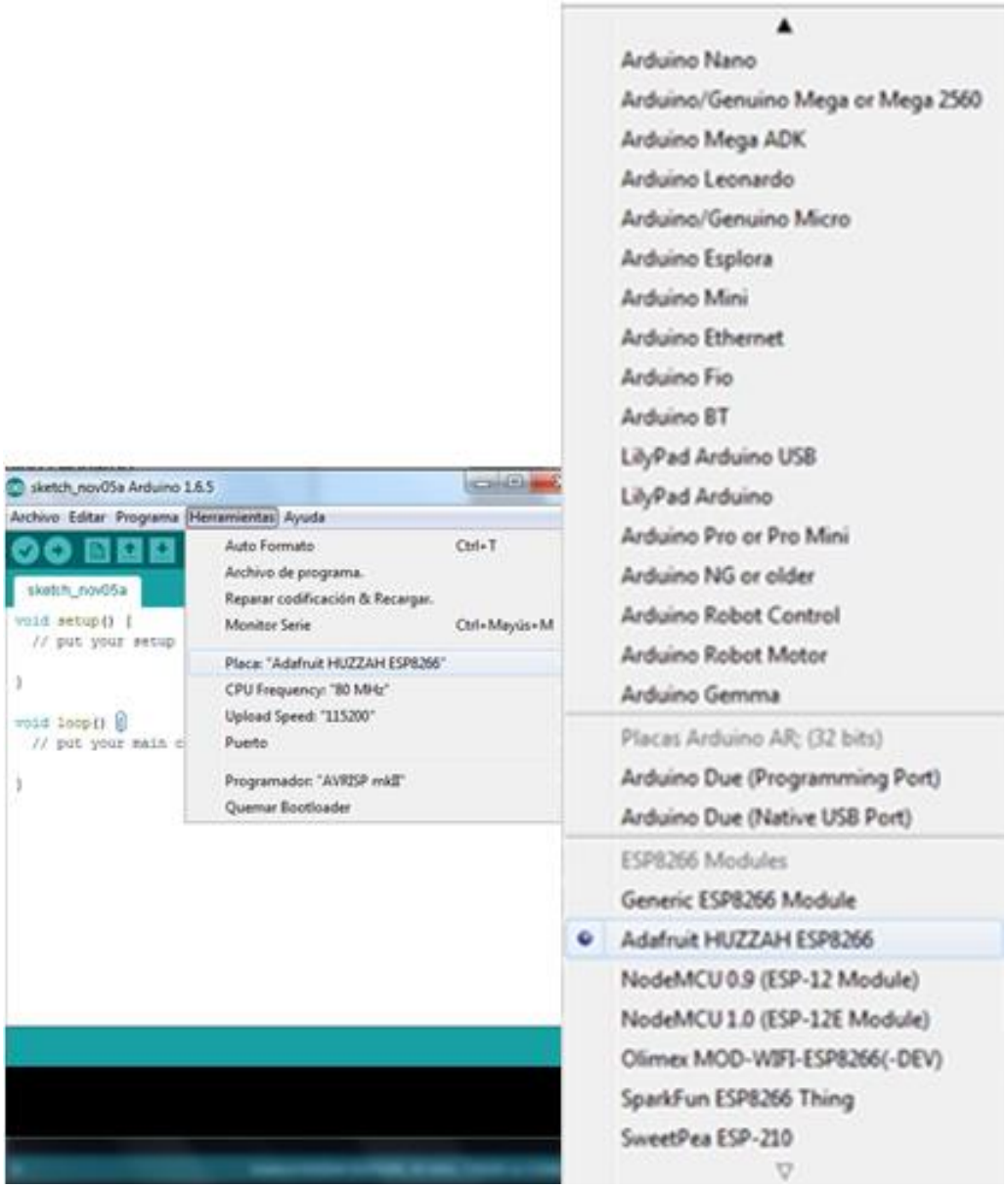


Figura 74 Instalación completa.

ANEXO C BLOQUES DE SIMULINK®

Bloque de visión

3-Fiduciales

```
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;
```

Acondicionamiento Visión

```
function [xp,yp,tp,xp2,yp2,tp2,xp3,yp3,tp3]= fcn(u,u2,u3)
dpi=6.2831;
x     = u(1);
y     = u(2);
th    =dpi- u(3);

x2    = u2(1);
y2    = u2(2);
th2   = dpi-u2(3);

x3    = u3(1);
y3    = u3(2);
th3   = dpi-u3(3)+0.174;


%Coordenadas del robot y del objetivo
xs   = (x/100)*(-247)+123.5;
ys   = (y/100)*(+184)-92;

xs2  = (x2/100)*(-247)+123.5;
ys2  = (y2/100)*(+184)-92;

xs3  = (x3/100)*(-247)+123.5;
ys3  = (y3/100)*(+184)-92;

%Ajuste de parámetros de visión (Transformación)

xp = xs+(27.2*cos(th));
yp = ys+(27.2*sin(th));
tp = th;

xp2 = xs2-(21*cos(th2));
yp2 = ys2-(21*sin(th2));
tp2 = th2;

xp3 = xs3-(21*cos(th3));
yp3 = ys3-(21*sin(th3));
tp3 = th3;
```

Bloque de medición de distancia

```
function [PosAct,PosDes,L1d] = fcn(F1,F2,F3)
%#codegen

PosAct=F1;
PosDes=F2;
manipulador = 0;

%%distancias del robot(F1) a la metas
L1d=sqrt(power((F2(1)-F1(1)),2)+power((F2(2)-F1(2)),2)); %meta F2
PosDes=F2;
manipulador=L1d;
```

Bloque de campos potenciales

```
#define S_FUNCTION_NAME  CamposPotencialesIrene
#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:
 *    y = 2*u
 */
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 = 0.3;
vymax = 0.3;
wmax  = 1;
//Posición y orientación del MMO
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
  k_OWR = 35; //robot
kr  =50 ; //mesa



//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
```

Bloque de cinemática

```
function [rueda1, rueda2, rueda3, rueda4]= fcn(Vx,Vy,W)
%datos del robot
r=0.025;
b=0.1425;
d=0.0825;

rueda1t=  (Vx/r) - (Vy/r) - (((b+d)/r)*W);
rueda2t=  (Vx/r) + (Vy/r) - (((b+d)/r)*W);
rueda3t=  (Vx/r) - (Vy/r) + (((b+d)/r)*W);
rueda4t=  (Vx/r) + (Vy/r) + (((b+d)/r)*W);

rueda1 = rueda1t;
rueda2 = rueda2t;
rueda3 = rueda3t;
rueda4 = rueda4t;
```

Bloque de acondicionamiento de datos

```
function y = fcn(u)
%#codegen

   y=((-127*u/60)+128);

   if y>255
     y=255;

   end

   if y<0
     y=0;
   end

 end
```

ANEXO D PROGRAMAS DE ESPINO

ESPino-base

```
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
const char* ssid     = "Nombre_de_red";
const char* password = "contraseña";
unsigned int localPort = 2390;
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
byte a=128,b=128,c=128,d=128;
WiFiUDP Udp;
void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println();// salto de linea
  Serial.println();//salto de linea
  Serial.print("Conectando a: ");// mando mensaje "conectando a ...
  Serial.println(ssid);//imprimo nombre de la red
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  }
  Serial.println("");//imprimo ""
  Serial.println("WiFi conectado"); // si se conectó notifico
  Serial.println(WiFi.localIP());//imprimo IP
```

```
  Udp.begin(localPort);
}
int value = 0;
int x;
byte g=155;
void loop() {
  for (int i=0; i<=255; i++)
 {
  packetBuffer[i]=0;
 }
  int packetSize = Udp.parsePacket();
  if (packetSize)
  {
    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
     a=packetBuffer[0];
     b=packetBuffer[1];
     c=packetBuffer[2];
     d=packetBuffer[3];
     delay(1);
     Serial.write(a);
     delay(1);
     Serial.write(b);
     delay(1);
     Serial.write(c);
     delay(1);
     Serial.write(d);
     delay(1);
     for (int i=0; i<=255; i++)
      {
         packetBuffer[i]=0;
      }
```

```
  }
  else
    {
      WiFi.begin(ssid, password);
        for (int i=0; i<=255; i++)
          {
            packetBuffer[i]=0;
          }
    }
}
```

ESPino-Manipulador

```
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
const char* ssid     = "Nombre_de_red";
const char* password = "contraseña";
unsigned int localPort = 2490;
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
byte a=128,b=128,c=128,d=128;
WiFiUDP Udp;
void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println();// salto de linea
  Serial.println();//salto de linea
  Serial.print("Conectando a: ");// mando mensaje "conectando a ...
  Serial.println(ssid);//imprimo nombre de la red
  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  }
  Serial.println("");//imprimo ""
  Serial.println("WiFi conectado"); // si se conectó notifico
  Serial.println(WiFi.localIP());//imprimo IP
  Udp.begin(localPort);
}
int value = 0;
int x;
void loop() {
  for (int i=0; i<=255; i++)
 {
  packetBuffer[i]=0;
 }
 int packetSize = Udp.parsePacket();
  if (packetSize)
  {
    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
      e=packetBuffer[0];
      delay(1);
      Serial.write(e);
      delay(1);
 for (int i=0; i<=255; i++)
 {
  packetBuffer[i]=0;
 }
    }
```

```
    else
  {
  WiFi.begin(ssid, password);
  for (int i=0; i<=255; i++)
 {
  packetBuffer[i]=0;
 }
  }
```

ANEXO E PROGRAMAS DE ARDUINO

Arduino-base

```
#include <Wire.h> //biblioteca para  comunicación I2C
//configuración de parámetros para la md25
#define i2cAddress 0x59      //dirección de la primer MD25
#define i2cAddress2 0x58      //dirección de la segunda MD25
#define vel2 0x00             //dirección del registro de velocidad 1
#define vel1 0x01             //dirección del registro de velocidad 2
#define VOLTREAD 0x0A
int a=128, c=128, d=128, b=128; //Velocidades
void setup() {
Wire.begin(); //Inicio comunicación
Serial2.begin(115200); //inicio comunicación serial, recibirá datos desde ESPino
Serial.begin(9600);//inicio el monitor serial para
//tener retroalimentación en la computadora
}
void loop() {
  if(Serial2.available()>0)
    {
       delay(1);
       a=Serial2.read();
       Serial.println(a);
       delay(1);
       b=Serial2.read();
       Serial.println(b);
       delay(1);
       c=Serial2.read();
       Serial.println(c);
```

```
     delay(1);
     d=Serial2.read();
     Serial.println(d);
     delay(1);
 }
else
{
 a=128;
 c=128;
 d=128;
 b=128;
}
 //Inicio la trasmición a las MD25
digitalWrite(13,HIGH);
Wire.beginTransmission(i2cAddress);
Wire.write(vel1);
Wire.write(b);
Wire.endTransmission();
Wire.beginTransmission(i2cAddress);
Wire.write(vel2);
Wire.write(c);
Wire.endTransmission();
Wire.beginTransmission(i2cAddress2);
Wire.write(vel1);
Wire.write(a);
Wire.endTransmission();
Wire.beginTransmission(i2cAddress2);
Wire.write(vel2);
Wire.write(d);
Wire.endTransmission();
digitalWrite(13,LOW);
```

```
   }
```

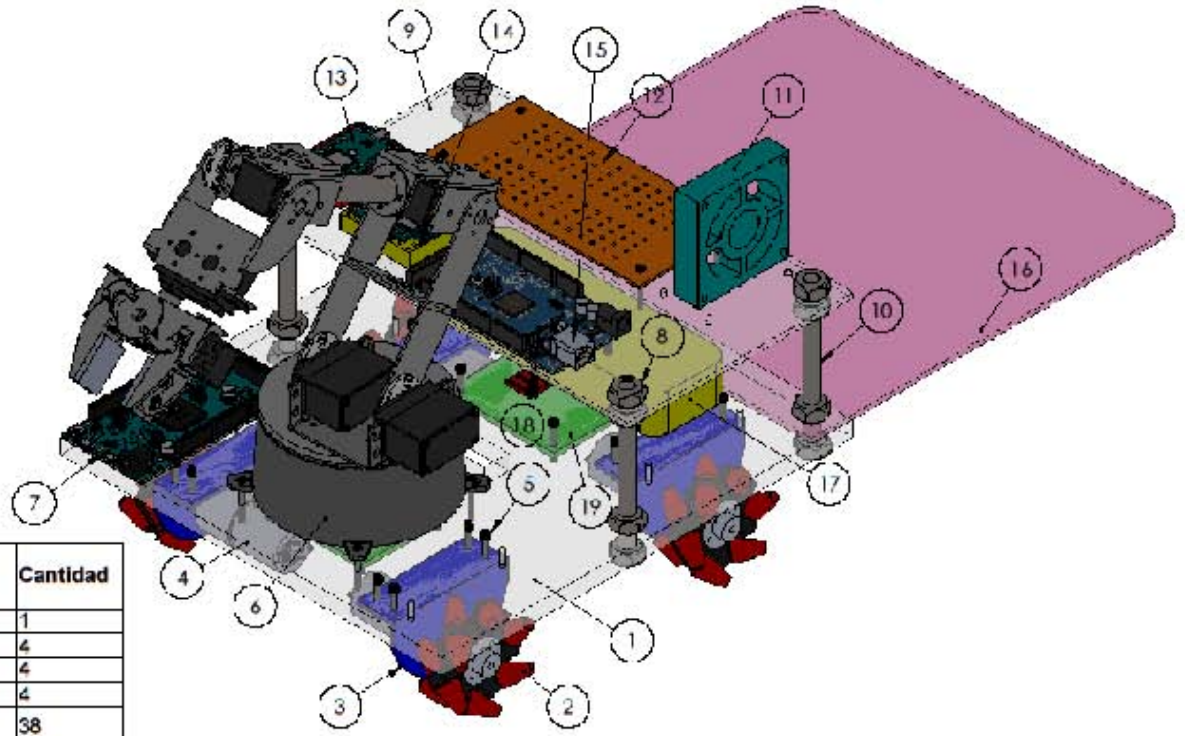
Arduino-manipulador

```
#include <Servo.h>//Biblioteca para servomotores
Servo base;
Servo hombro;
Servo hombro2;
Servo codo;
Servo muneca;
Servo artmuneca;
Servo efector;
int u;
int s;
int m;
int y;
float h,c;
int ih,ic;
void setup() {
Serial1.begin(115200); //inicio comunicación serial, recibirá datos desde ESPino
Serial.begin(9600);//inicio el monitor serial para
//tener retroalimentación en la computadora
base.attach(2);
 hombro.attach(3);
 codo.attach(4);
 muneca.attach(5);
 artmuneca.attach(6);
 efector.attach(7);
```

```
 hombro2.attach(8);
}
void loop() {
  y=0;
  if(Serial1.available()>0)
    {
      delay(1);
      m=Serial1.read();
      u=m;
      delay(1);
      h=0.00002*pow(u,3)-0.0044*pow(u,2)+0.0457*u+90.466;
      ih=(int)h;
      c=0.00003*pow(u,3)-0.0064*pow(u,2)-0.0168*u+76.782;
      ic=int(c);
  }
  else
    {
      hombro.write(90);
      codo.write(30);
      muneca.write(30);
      efector.write(90);
      delay(10);
    }
  if (u>= 107)
   {
      Serial.println("mayor a 107");
      hombro.write(90);
```

```
      codo.write(30);
      muneca.write(30);
      efector.write(90);
      delay(10);
   }
  if (u<107 && u>=35)
   {
      Serial.println("en funcion");
      codo.write(ic);
      muneca.write(30);
      efector.write(90);
      delay(10);
  }
 if (u <35 )
  {
     hombro.write(90);
     codo.write(70);
     muneca.write(30);
     efector.write(160);
     delay(10);
  }
}
```

ANEXO F: PLANOS



No. De parte	Nombre de parte	Cantidad
1	Base para ruedas	1
2	Rueda Mecanum	4
3	Soporte para motor EMG30	4
4	Motor EMG30	4
5	Tornillo con hexágono interior 3 x 2.3 mm	38
6	Brazo manipulador	1
7	Arduino DUE	1
8	Tuerca hexagonal 8 x 8.5 mm	12
9	Base para electrónica	1
10	Espárrago 8 x10 mm	4
11	Ventilador	1
12	Placa fenólica para regulador	1
13	Tarjeta ESPino	2
14	Protoboard mini	2
15	Arduino MEGA	1
16	Base para fiducial	1
17	Batería LIPo	1
18	Convertido de niveles lógicos	1
19	Tarjeta MD25	2

Manipulador móvil omnidireccional, su coordinación de movimientos en ambientes inteligentes

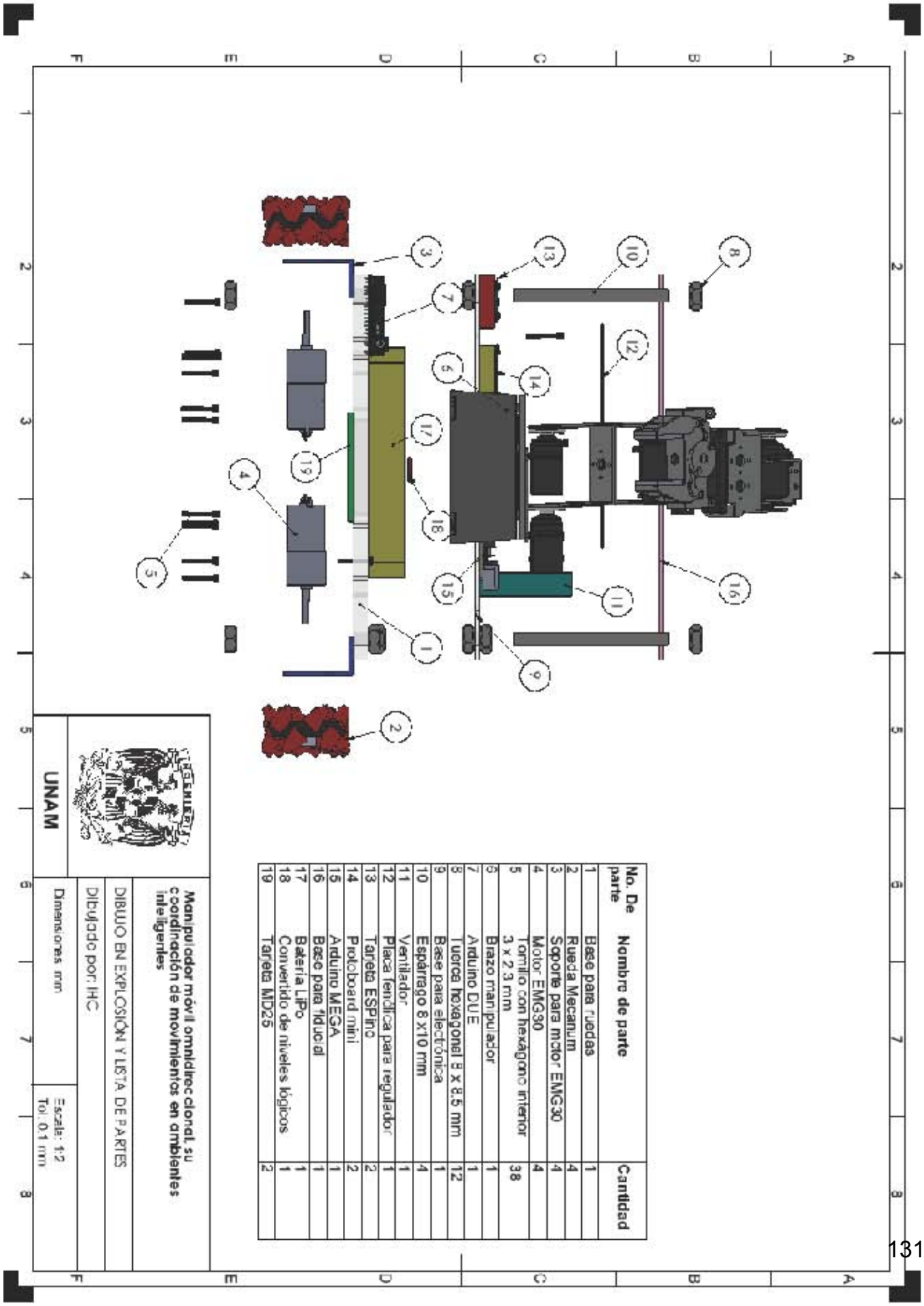
ENSAMBLE COMPLETO Y LISTA DE PARTES

Dibujado por: HC

UNAM

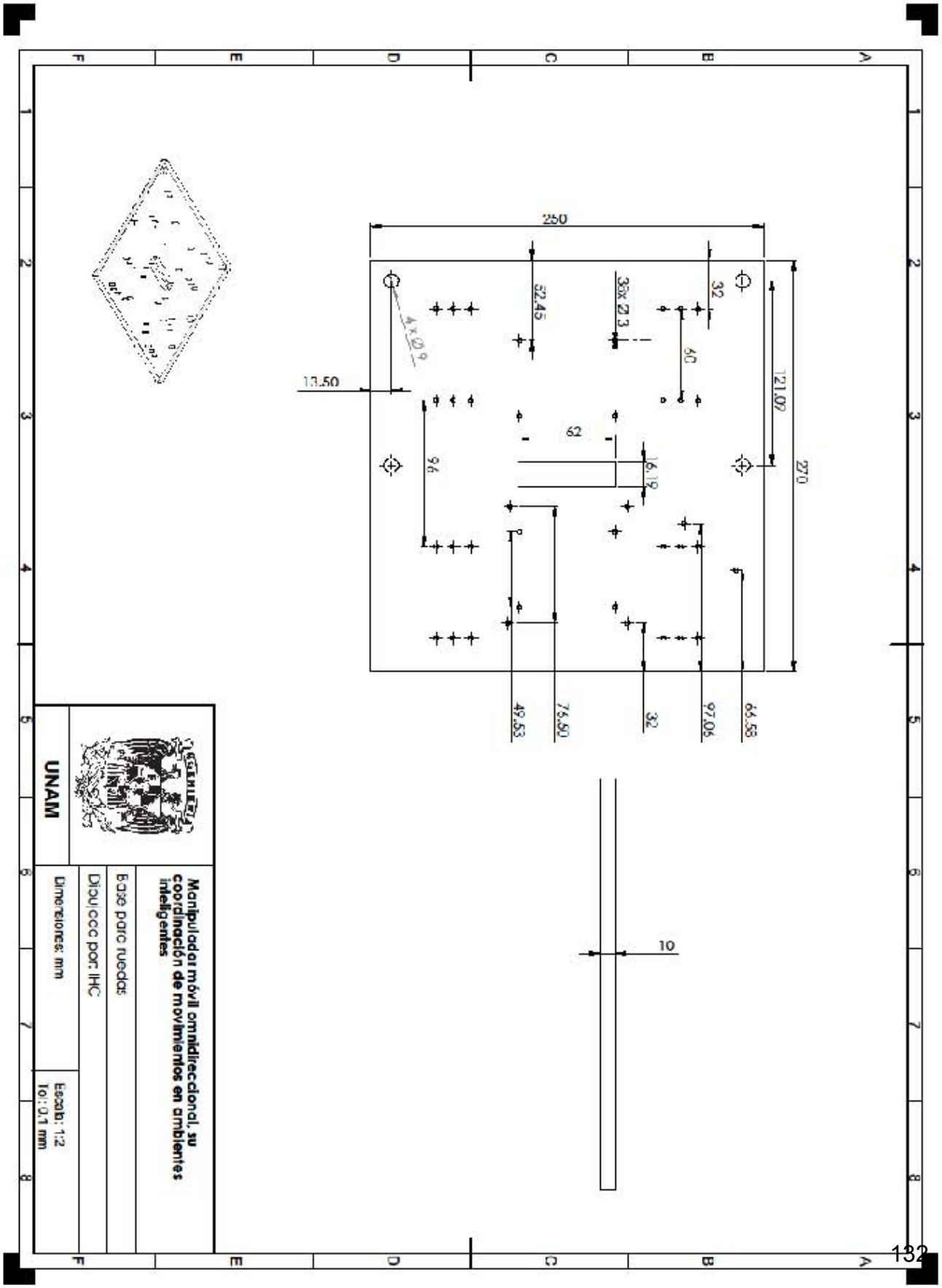
Dimensiones: mm

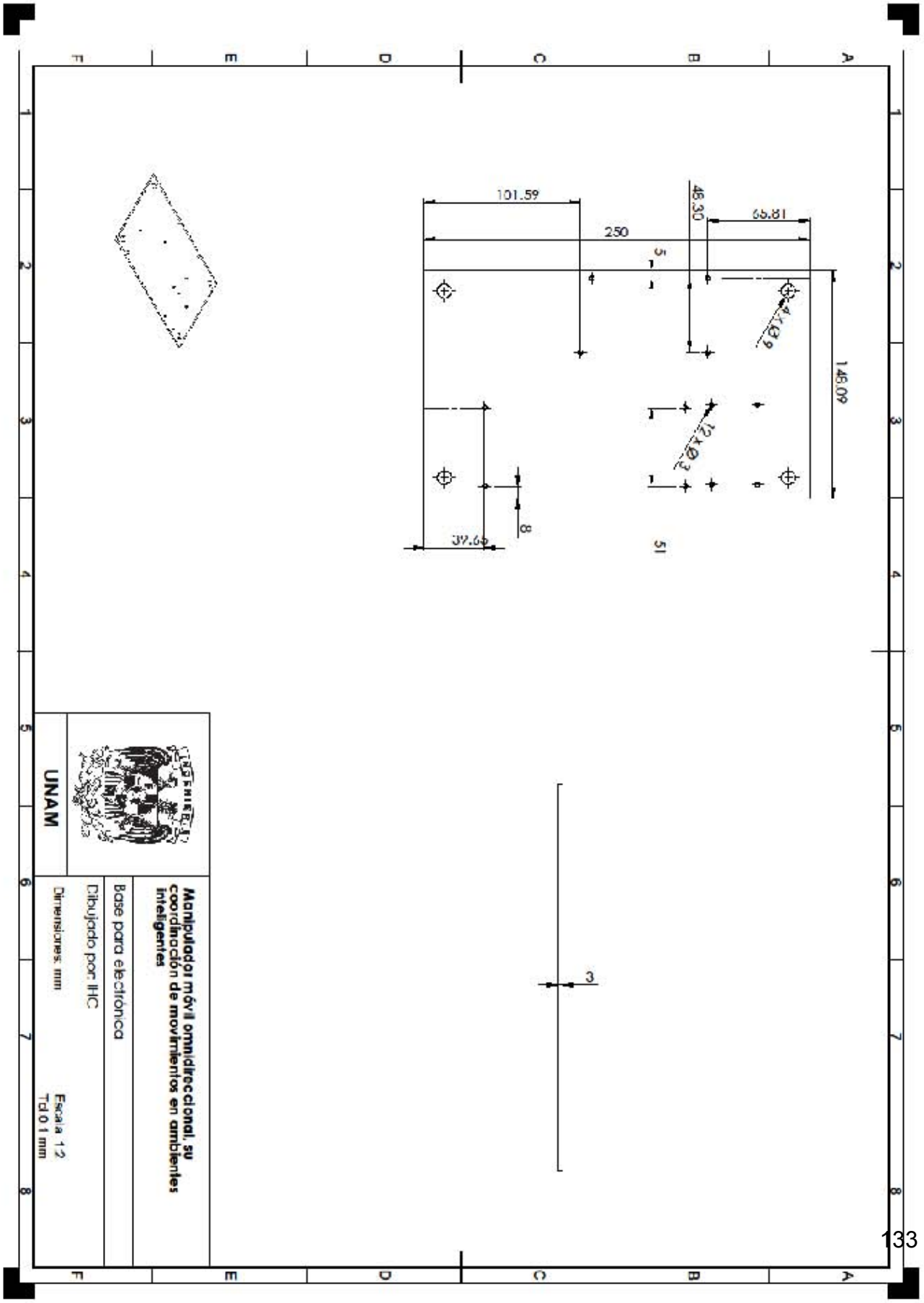
Escala: 1:2
Tol: 0.1 mm

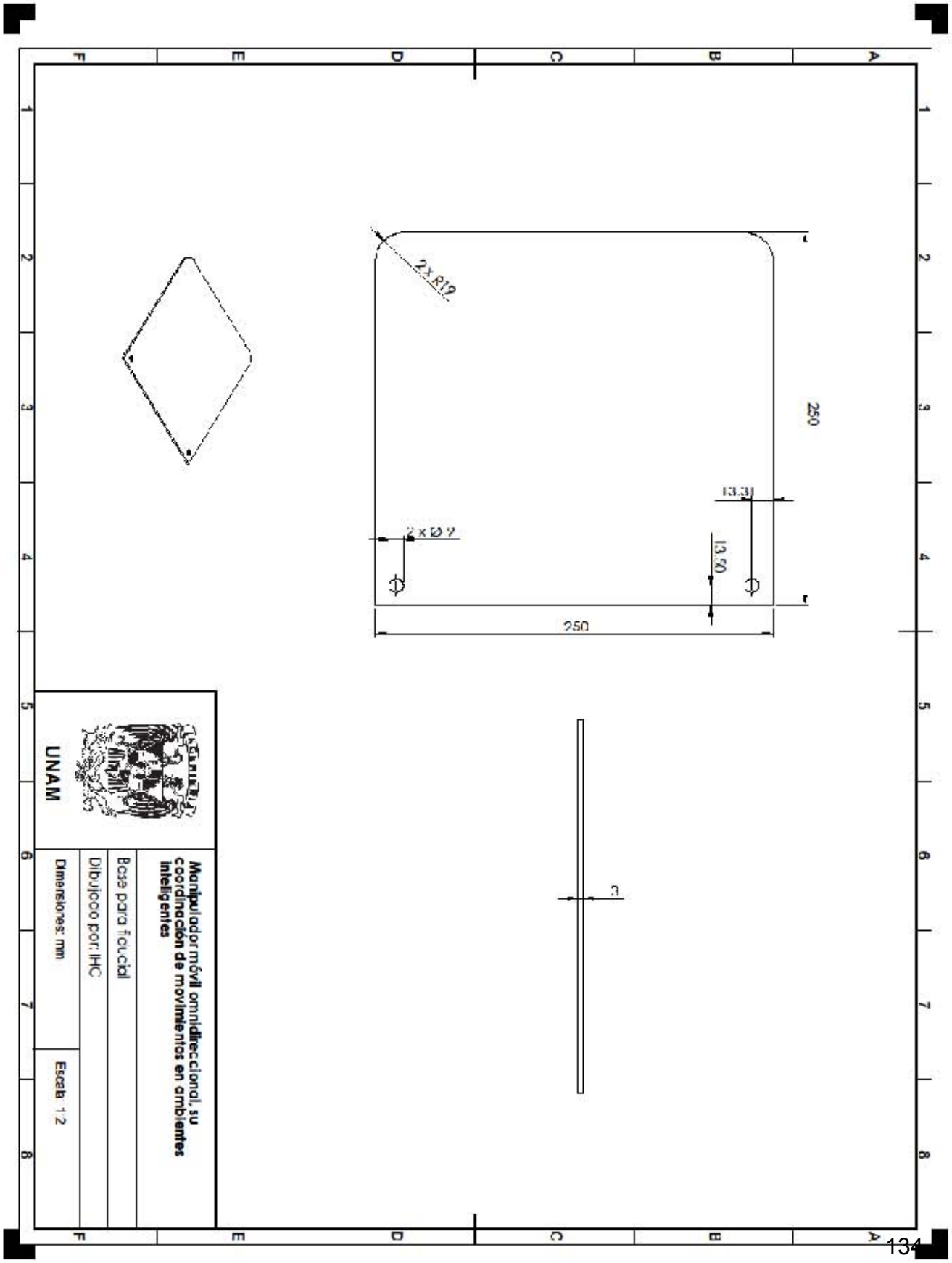


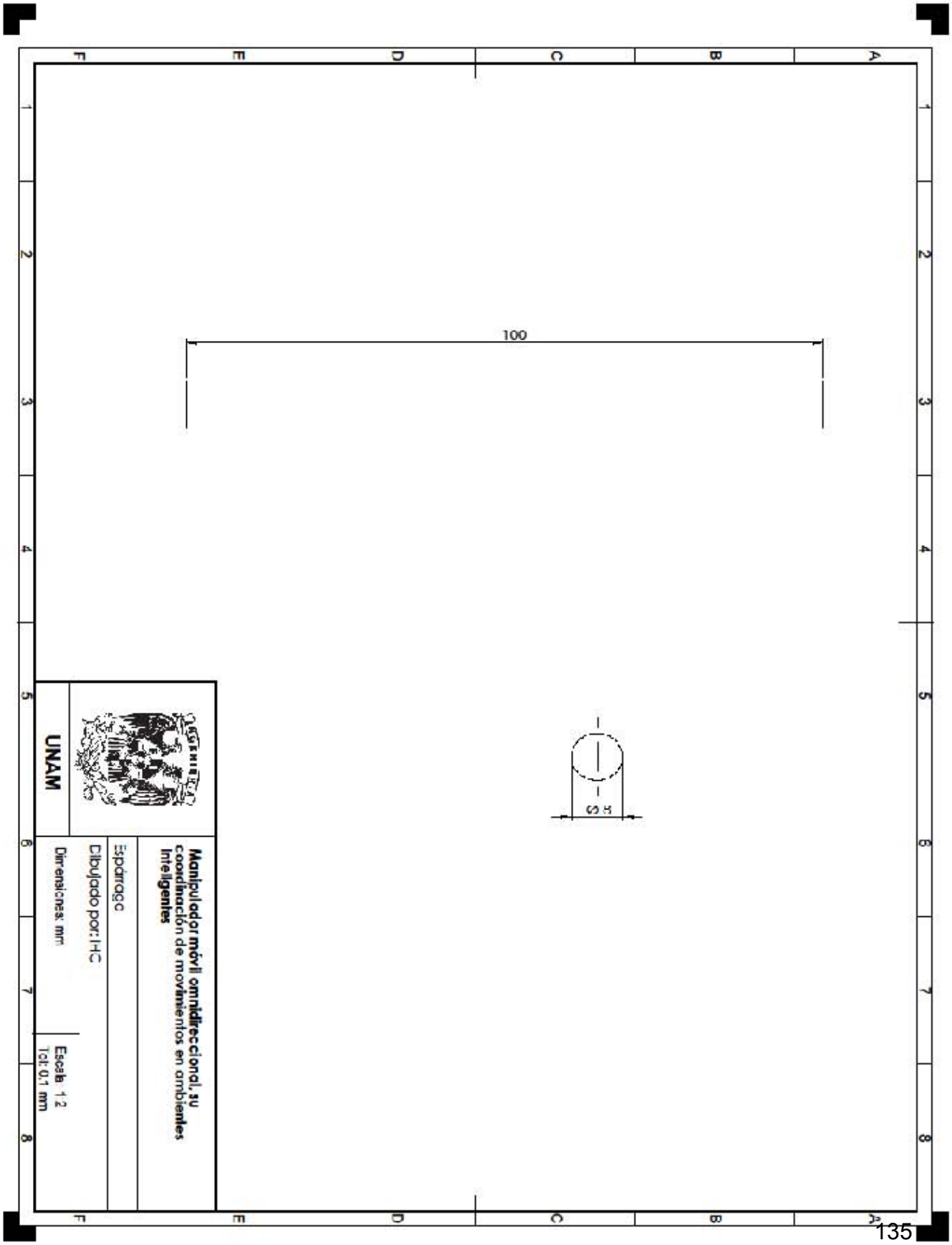
No. De parte	Nombre de parte	Cantidad
1	Base para ruedas	1
2	Rueda Mecanum	4
3	Soporte para motor EMG30	4
4	Motor EMG30	4
5	Tornillo con hexágono interior 3 x 2.3 mm	38
6	Brazo manipulador	1
7	Arduino DUE	1
8	Tuerca hexagonal 3 x 8.5 mm	12
9	Base para electrónica	1
10	Espárrago 8 x10 mm	4
11	Ventilador	1
12	Placa fenólica para regulador	1
13	Tarjeta ESPinc	2
14	Protoboard mini	2
15	Arduino MEGA	1
16	Base para fiducial	1
17	Batería LiPo	1
18	Convertidor de niveles lógicos	1
19	Tarjeta MD25	2

UNAM	Manipulador móvil omnidireccional su coordinación de movimientos en ambientes inteligentes
	DIBUJO EN EXPLOSIÓN Y LISTA DE PARTES
	Dibujado por: IHC
	Dimensiones: mm / Escala: 1:2 / Tol: 0.1 mm


250
270
121.07
32
32.45
36x Ø 3
60
4x Ø 9
13.50
62
96
16.19
49.53
75.50
32
97.05
64.58
10
UNAM
Manipulador móvil omnidireccional, su coordinación de movimientos en ambientes inteligentes
Base para ruedas
Dibujado por: IHC
Dimensiones: mm
Escala: 1:2
Tol: 0.1 mm


101.59
250
46.30
65.81
146.09
37.69
8
51
3
UNAM
Manipulador móvil omnidireccional, su coordinación de movimientos en ambientes inteligentes
Base para electrónica
Dibujado por: IHC
Dimensiones: mm
Escala 1:2
Tol 0.1 mm


2x R19
250
13.33
2x Ø 7
13.50
250
3
UNAM
Manipulador móvil omnidireccional, su coordinación de movimientos en ambientes inteligentes
Base para fecucial
Dibujado por: IHC
Dimensiones: mm
Escala 1:2


100
UNAM
Manipulador móvil omnidirecccional, su coordinación de movimientos en ambientes Inteligentes
Esparrago
Dibujado por: IHC
Dimensiones: mm
Escala 1:2
Tol: 0.1 mm

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICAS

REFERENCIAS

[1] A. A. Souma y M. Chaffari, «Mobile Robotics, Moving Intelligence,» *the Advanced Robotic Systems International and pro literatur Verlag,* 2006.

[2] F. Reyes Cortés , Robótica, control de robots manipuladores, Alfaomega, 2011.

[3] A. K. Solutions, Robotics, Infinity Science Press, 2007.

[4] A. Ollero Baturone, Robótica, manipuladores y robots móviles, Barcelona: Marcombo, 2001.

[5] F. R. Rodríguez Chirinos, «Scribd,» [En línea]. Available: http://es.scribd.com/doc/138251282/Articulaciones-de-Robots#scribd. [Último acceso: 20 04 2015].

[6] T. Bajd y a. et, Robotics, Springer, 2010.

[7] RobotWorx, «RobotWorx, a Scott Technology Ltd. Company,» [En línea]. Available: http://www.robots.com/faq/show/what-is-a-robot-manipulator. [Último acceso: 04 28 2015].

[8] F. R. Cortéz, Matlab aplicada a robotica y mecatrónica, Alfaomega, 2012.

[9] P. Ávila Carlos, Espacios inteligentes con manipuladores móviles dotados de intición artificial para el transporte de materiales. tesis de Licenciatura, 2014.

[10] B. A. J. N van, «A User-Interface Robot for Ambient Intelligent Environments,» *International Conference on Advances on Service Robots, At Bardolino, Italy,* 2003.

[11] R. C. Arkin, «Towards Cosmopolitan Robots: Intelligent Navigation in Extended Man-Made Environments,» *Georgia Institute of Technology,* 1987.

[12] A. Hoover y O. B.D, «A Real-Time Occupancy Map From Multiple Video Stream,» *International Conference on Robotics & Automation,* 1999.

[13] D. Pinzarro Pérez, Localización de robots móviles en espacios inteligentes utilizando cámaras externas y marcas naturales, Tesis de doctorado, 2008.

[14] K. Watanabe, S. Kazuya y I. Kiyotaka, «Analysis and Control for an Omnidirectional Mobile Manipulator,» *Journal of Intelligent and Robotic Systems,* 2000.

[15] M. H. Korayem, «Maximum load-carrying capacity of autonomous mobile manipulator in an environment with obstacle considering tip ober stability,» *Springer-Verlag London Limited,* 2009.

[16] H. Asama , M. Sato y L. Bogoni, «Development of an Omni-Directional Mobile Robot with 3 DOF Decoupling Drive Mechanism,» *IEEE International Conference on Robotics and Automation,* 1995.

[17] X.-m. Tan, «Unified Model and Robust Neural-Network Control of Omnidirectional Mobile Manipulators,» *6th IEEE Int. Conf. on Cognitive Informatics (ICCI´07),* 2007.

[18] K. Osussama, «Mobile manipulation: The robotic assistant,» *Elsevier Science B.V,* 1998.

[19] I. C. Cruz López, manipulador móvil omnidireccional redundante: coordinación de movimientos en ambientes inteligentes, Tesis de Maestría, 2015.

[20] M. Z. Patricio, Manipulador robótico de seis grados de libertad cinfiguración anropomórfica, 2007.

[21] J. J. Craig, Introduction to Robotics, Mechanics and Control, Third Edition ed., Pearson Education International, 2005.

[22] E. C. P. Mariano, Manipulador móvil: estudio sobre la coordinación de movimientos de un manipulador serial acoplado, Tesis de Licenciatura, 2012.

[23] A. E. S. A. y. A. M. S. Balpuesta, Plataforma móvil omnidirecional de cuatro llantas suecas (mecanum) en configuración “ab”, Tesis de Licenciatura, 2015.

[24] INTPLUS, «Super Robotica,» [En línea]. Available: http://www.superrobotica.com/S330100.htm. [Último acceso: 2015 Octubre 28].

[25] Robot Electronics, «EMG30, mounting bracket and wheel specification,» [En línea]. Available: http://www.robot-electronics.co.uk/htm/emg30.htm. [Último acceso: 2015 Octubre 28].

[26] I. d. M. P. S.L., «Microsystems Engineering,» [En línea]. Available: http://www.msebilbao.com/tienda/product_info.php?cPath=53_102&products_id=572. [Último acceso: 2015 Octubre 29].

[27] I. d. M. P. S.L, «MD25: Driver doble puente H para motores,» *Engineering, Microsystemas.*

[28] M. I. e. I. S. d. C.V., «ESPino,» [En línea]. Available: http://www.espino.io/es. [Último acceso: 2015 noviembre 2].

[29] C. R. Fernando, Matlab aplicado a robótica y mecatrónica, alfaomega, 2012.

[30] « EMG30, mounting bracket and wheel specification,» [En línea]. Available: http://www.robot-electronics.co.uk/htm/emg30.htm. [Último acceso: 2015 Octubre 28].

[31] C.V., Makerlab Ingeniería e Innovación S.A.P.I. de, «ESPino,» [En línea]. Available: http://www.espino.io/es. [Último acceso: 2015 noviembre 2].

[32] INTPLUS, «Super Robotica,» [En línea]. Available: http://www.superrobotica.com/S330100.htm. [Último acceso: 2015 Octubre 28].

[33] Robotnik, «Robotnik,» [En línea]. Available: http://www.robotnik.es/. [Último acceso: 28 08 2015].

[34] S.L, Ingeniería de Microsistemas Programados, «MD25: Driver doble puente H para motores,» *Engineering, Microsystemas.*

[35] S.L., Ingenieria de Microsistemas Programados, «Microsystems Engineering,» [En línea]. Available:

http://www.msebilbao.com/tienda/product_info.php?cPath=53_102&products_id=572. [Último acceso: 2015 Octubre 29].

[36] O. Díaz Hernández, Intuición artificial aplicada a la teleoperación, Tesis de Doctorado, México, 2014.

[37] V. J. González Villela, R. Parkin y M. López Parra, «A wheeled mobile robot with obstacle avoidance capability,» 2004.