UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
MECÁNICA – MECATRÓNICA

INTUICIÓN ARTIFICIAL APLICADA A UN MANIPULADOR MÓVIL QUE OPERA DENTRO DE UN ESPACIO INTELIGENTE

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

PRESENTA:
ALDANA BECERRA JOSE CRISOGONO

TUTOR PRINCIPAL
DR. VÍCTOR JAVIER GONZÁLEZ VILLELA, FACULTAD DE INGENIERÍA

CIUDAD UNIVERSITARIA, CIUDAD DE MÉXICO, FEBRERO 2018

JURADO ASIGNADO:

Presidente:	Dr. Borja Ramírez Vicente
Secretario:	Dr. Cuenca Jiménez Francisco
Vocal:	Dr. González Villela Víctor Javier
1 er. Suplente:	Dr. Rocha Cózatl Edmundo Gabriel
2 do. Suplente:	Dra. Corona Lira María Del Pilar

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

TUTOR DE TESIS:

DR. VÍCTOR JAVIER GONZÁLEZ VILLELA

FIRMA

(Segunda hoja)

DEDICATORIAS

A Dios, sin su ayuda nada de esto sería posible.

A mi mamá Teodora Becerra Rodríguez y papá José Nicolás Crisogono Aldana López, por su amor, ayuda incondicional, educación y paciencia.

A mi hermana Alma Nidia Aldana Becerra por darme su apoyo siempre en los momentos buenos y malos.

A mi hermano Cristhian Aldana Becerra.

Al M.I. Paúl Ramírez Sánchez por ofrecerme su amistad desde la educación secundaria y haberme mostrado el camino para realizar mis estudios de maestría. Así como a su familia por la amistad y confianza brindada durante todos estos años.

A la Sra. Isabel por abrirme las puertas de su casa y brindarme toda su ayuda, atenciones y amistad en la travesía de mis estudios de maestría.

Al Doctor Víctor Javier González Villela por darme una excelente tutoría y educación a lo largo de mis estudios de maestría, así como por su tiempo, consejos y ejemplo de trabajo disciplinado. Además, agradezco profundamente su infinita paciencia a mi persona y la confianza brindada para la finalización de la presente tesis.

AGRADECIMIENTOS

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado para la realización de los estudios de maestría y elaboración de tesis.

También, agradezco en lo que corresponde a la DGAPA, por el apoyo brindado para la realización de este trabajo, a través de los proyectos UNAM-DGAPA-PAPIIT IN117614: “Robótica intuitiva, adaptable, reactiva, híbrida y móvil aplicada al servicio, el rescate y la medicina” y UNAM-DGAPA-PAPIIT IN118117: "Investigación sobre robótica topofixadaptable aplicada a robots móviles híbridos, que operan en ambientes inteligentes estructurados, en tareas de sujeción, traslación y orientación de objetos con cierto grado de asimetría”.

A la Universidad Nacional Autónoma de México y al departamento de posgrado en ingeniería Mecánica por las atenciones brindadas durante mis estudios de posgrado y su cobijo durante dos años y medio.

A mis compañeros del posgrado: Alejandro Jaramillo, Johnny Amadeus, Rafael Orta, Carlos Vargas, Osiris, Miguel Torreblanca y en especial a Eduardo Valentín Talavera Moctezuma.

Agradezco al ingeniero José Efraín Ramírez Ramírez, al M.I Erik Peña Medina y al M.I Neftali Elorza López por su ayuda y conocimientos brindados para la implementación de los experimentos, así como también a los demás miembros del MRG (Mechatronic Research Group).

A todos mis profesores de maestría por haberme proporcionado los conocimientos y experiencias para poder enfrentar los retos de mi campo disciplinario y la vida.

A mis sinodales el Dr. Rocha Cózatl Edmundo Gabriel, Dr. González Villela Víctor Javier, Dr. Borja Ramírez Vicente, Dra. Corona Lira María Del Pilar y el Dr. Cuenca Jiménez Francisco por la retroalimentación brindada para terminar con éxito la redacción del presente trabajo. Finalmente agradezco al Doctor Alejandro Cuauhtémoc Ramírez Reivich por su ayuda para finalizar mi proceso de titulación.

RESUMEN

En este trabajo de investigación se logra construir un manipulador móvil en el que se implementa como intuición artificial el lugar geométrico basado en los patrones que genera la intuición humana en tareas de traslado.

Se consigue llevar el conjunto de instrucciones dadas por la intuición en tareas de traslado del campo de la teleoperación a la robótica móvil, específicamente en la planeación del lugar geométrico que recorre el efector final del manipulador móvil en su trayecto de ida y regreso, para desarrollar la tarea que consiste en sujetar y colocar a un objeto, dicho lugar geométrico es generado automáticamente por el algoritmo programado, reduciendo la complejidad del conjunto de instrucciones necesarias para resolver la tarea, brindando así soluciones rápidas y certeras.

Finalmente, los resultados demuestran que el modelo matemático sintetizado del espacio de trabajo generado por la tarea intuitiva de traslado realizada por el ser humano en el campo de la teleoperación, puede ser programado, medido y transferido a un manipulador móvil para el desarrollo de tareas de traslado de objetos dentro de un espacio inteligente.

Palabras clave

- Intuición artificial
- Traslado
- Lugar geométrico
- Manipulador móvil
- Espacio inteligente

APORTACIONES

Con respecto al manipulador móvil:

- Se construye y ensambla la plataforma móvil del manipulador en base a un diseño previo.
- Se ensambla el brazo serial y el efector final (gripper) del manipulador móvil en base a un kit de accesorios comercializados por el proveedor y compatible con los servomotores Dynamixel utilizados.
- Se realiza un reacomodo de los componentes electrónicos y de energía dentro de la plataforma móvil.
- Se realiza un módulo electrónico para controlar vía inalámbrica a los servomotores de la marca Dynamixel serie AX-12A que conforman al brazo serial del manipulador.

Con respecto a la programación se encuentra la forma de introducir el espacio intuitivo en el algoritmo de coordinación.

- Se programa en Matlab/Simulink cada módulo del algoritmo coordinante con el fin de que sea compatible con simulaciones en tiempo real, se aclara que el algoritmo de coordinación fue diseñado en un trabajo previo.
- Se programa la cinemática inversa del brazo serial en lenguaje C (S-Function) para optimizar la respuesta del sistema en tiempo real.
- Se programa en Simulink el bloque de control y coordinación del brazo serial.
- Se sincroniza el algoritmo de coordinación, así como el envío de la posición angular, velocidad e identificador (ID) y la recepción del ID, posición angular, velocidad, porcentaje de carga, sentido de giro, voltaje y temperatura de cada servomotor Dynamixel vía inalámbrica.
- Se programa y coordina dentro del bloque de control del brazo serial la fórmula que genera los puntos del lugar geométrico (espacio intuitivo) que debe recorrer el efector final para realizar cada etapa de la tarea “pick and place”.

- Se programa en lenguaje arduino la forma de enviar y recibir datagramas a través de la red inalámbrica en tiempo real.
- Se programa en Simulink la forma de enviar y recibir vía inalámbrica paquetes UDP a una sola tarjeta de adquisición, en este caso la tarjeta arduino.

Con respecto a la teoría de intuición artificial:

- Se utiliza la propiedad de la intuición artificial de ser potencialmente aplicable a cualquier sistema artificial para extenderla del área de la teleoperación a la robótica móvil, específicamente en cada etapa de la tarea conocida como "pick and place", para generar el lugar geométrico que debe recorrer el efector final del manipulador móvil.

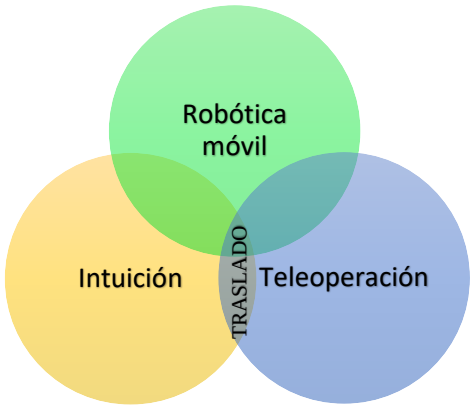


Figura 1.- Vínculo entre la intuición, teleoperación y la robótica móvil

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

Durante las últimas décadas en el área de la robótica el principal objetivo ha sido dotar con inteligencia a los sistemas para lograr autonomía en la toma de decisiones ante nuevos escenarios. Con el fin de ampliar el campo de estudio y mejorar la eficiencia y eficacia de los sistemas (entiéndase por eficiencia según la real academia española como la capacidad de disponer de alguien o de algo para conseguir un efecto determinado y eficacia como la capacidad de lograr el efecto que se desea o se espera) se han desarrollado enfoques basados en procesos cognitivos.

Las ciencias cognitivas son las encargadas del estudio tanto de la mente como de la inteligencia desde un punto de vista interdisciplinario, son fruto de la confluencia entre la filosofía, la psicología, la inteligencia artificial, la neurociencia, la lingüística y la antropología; sus orígenes intelectuales se remontan a mediados de la década de 1950, cuando investigadores provenientes de distintos campos comenzaron a formular teorías de la mente valiéndose de representaciones y procesos computacionales complejos (Thagard, 2008).

El estudio de la mente no es tarea sencilla, puesto que no es posible tener acceso directo a ella, a lo largo de los siglos los filósofos y los psicólogos han recurrido a un conjunto de metáforas y comparaciones para referirse a la mente, se ha afirmado que es como una página en blanco sobre la que se hacen impresiones, o como un instrumento hidráulico en el que se ejercen distintas fuerzas o que es semejante a un conmutador telefónico. En los últimos cincuenta años han surgido nuevas metáforas gracias al desarrollo de nuevas clases de computadoras. Muchos especialistas en ciencia cognitiva, si bien no todos, consideran que el pensamiento es una especie de proceso computacional y utilizan metáforas relacionadas con el campo de la computación para describir y explicar cómo las personas aprenden y resuelven problemas (Thagard, 2008).

La ciencia cognitiva revela dos vías o caminos por los cuales funciona la mente humana (aunque actualmente se añaden nuevas vías como la procedimental) (Mente, 2007), estas son:

- *La mente consciente*, ha sido las más estudiada alrededor del mundo por la comunidad científica. Daniel Kahneman establece que se caracteriza por ser deliberada, analítica, consciente, secuencial, racional y requiere esfuerzo para llevarse a cabo (Corrales Navarro, 2010).

- *La mente inconsciente* es la más compleja debido a que sus funciones vienen determinadas por herencia genética y otros factores. Daniel Kahneman afirma que es automática, *intuitiva*, rápida, asociativa, implícita y no requiere de mayor esfuerzo consciente, además de que regularmente contiene carga emocional. Un ejemplo claro es la función de la respiración cuando dormimos, esta se hace de manera inconsciente (Corrales Navarro, 2010).

Como consecuencia del enfoque basado en procesos cognitivos se ha adoptado a la inteligencia artificial como principal herramienta para dar autonomía a los sistemas robóticos. El campo de la inteligencia artificial se compone de varias áreas de estudio, las más comunes e importantes son: búsqueda de soluciones, sistemas expertos, procesamiento del lenguaje natural, reconocimiento de modelos, robótica, aprendizaje de las máquinas, lógica, incertidumbre y "lógica difusa" (Ponce Cruz, 2010).

La inteligencia artificial posee cuatro categorías principales con las que es posible estudiarla y analizarla. Estas categorías son: sistemas que piensan como humanos, sistemas que actúan como humanos, sistemas que piensan racionalmente y sistemas que actúan racionalmente (Ponce Cruz, 2010).

En la inteligencia artificial como ciencia se han desarrollado a través de los años una gran cantidad de técnicas basadas en las características de la mente consciente. Sin embargo, más tarde que temprano podrá constatarse que lograr la racionalidad perfecta (siempre hacer lo correcto) no es posible en entornos complejos ya que implica cómputo excesivo (Ponce Cruz, 2010).

Actualmente el estudio y aplicación de estrategias basadas en las características de la mente inconsciente han sido de gran interés a la comunidad científica, por ejemplo, la automatización, emociones artificiales y programación de comportamientos con conductas reactivas y deliberativas (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).

Una vía poco explorada para la implementación de inconsciencia artificial es el uso de la intuición como un mecanismo de la mente inconsciente para tomar decisiones rápidas y certeras. *Cuando un agente artificial manifiesta propiedades de la intuición humana, entonces se puede decir que hay una intuición artificial* (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).

A lo largo del marco teórico se aclaran estos conceptos, sin embargo, no se realiza un debate profundo de los mismos, solo se desarrollan clasificaciones para su fácil entendimiento. Si se requiere una revisión bibliográfica más detallada se recomienda a lector consultar la tesis doctoral de (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).

En este trabajo de investigación se implementa la intuición artificial en un manipulador móvil dentro de un espacio inteligente, programando el espacio intuitivo (lugar geométrico) basado en el modelo desarrollado por (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014) que se fundamenta en los patrones que genera la intuición humana en tareas de traslado. Lo que se busca es la reducción de la complejidad del conjunto de instrucciones necesarias para resolver la tarea de traslado, principalmente en cada etapa de la tarea conocida como “pick and place”, en la que dicho espacio intuitivo se encarga de describir el lugar geométrico por el cual el efector final del manipulador móvil llega al punto deseado, de esta manera se logra determinar automáticamente mediante el algoritmo los puntos que debe recorrer el efector final.

El diseño del manipulador móvil, así como el algoritmo de coordinación están basados en la tesis de licenciatura desarrollada por (Escobedo Castillo, 2012), con modificaciones propias, las cuales son descritas a lo largo de la tesis, destacando la reprogramación del algoritmo y la sustitución del brazo serial por uno ensamblado con servomotores profesionales de la marca Dynamixel serie AX-12A.

La tarea que se desarrolla es el traslado de objetos. El espacio inteligente junto con el manipulador móvil como actuador ofrecen la base para poder llevar a cabo la experimentación.

En la figura 2 se muestra el orden en el que se desarrolla la investigación:

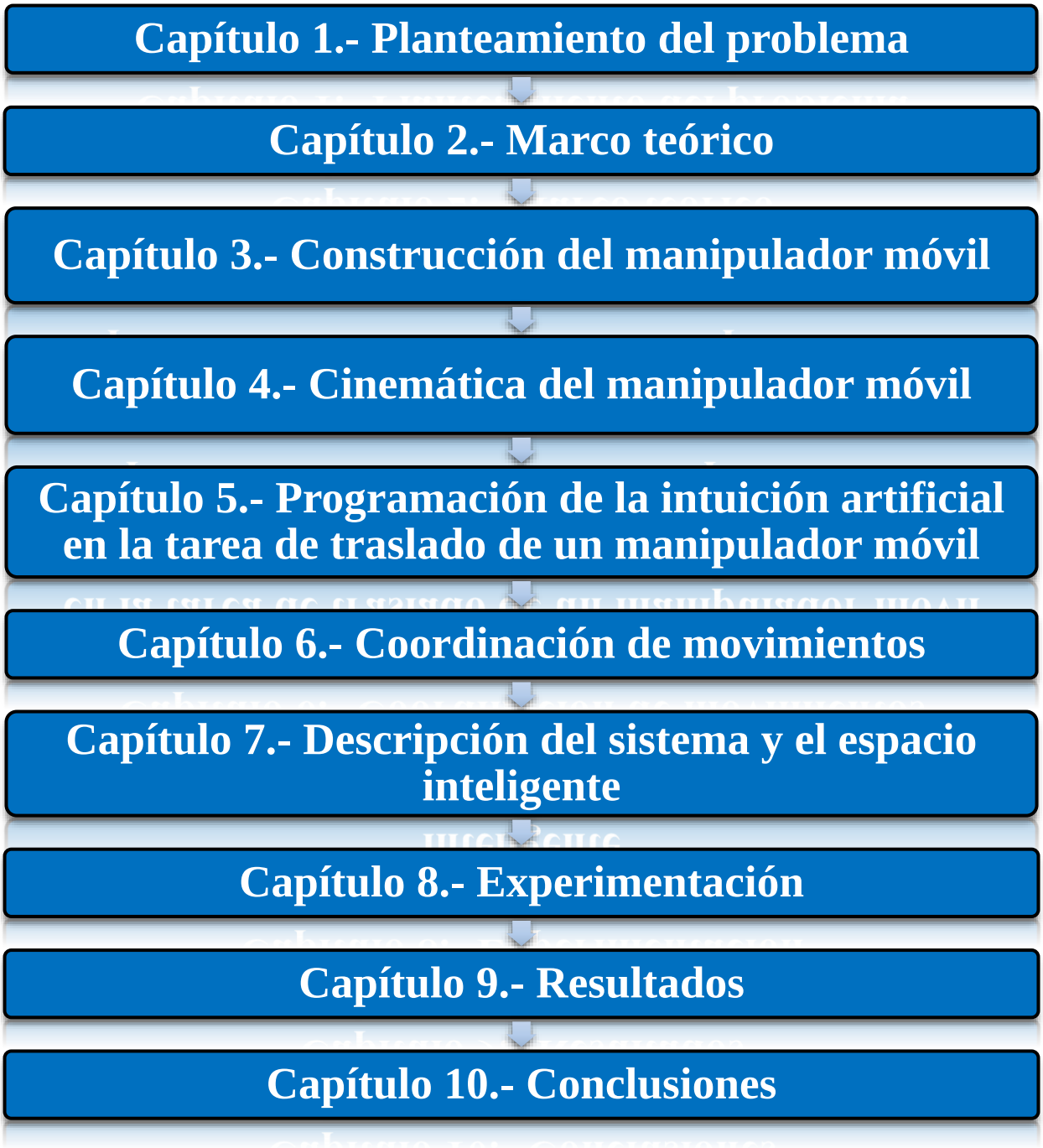


Figura 2.- Orden para el desarrollo de la investigación

Con respecto al software utilizado se aclara que todas las marcas registradas utilizadas en este documento son propiedad de sus respectivos desarrolladores. El uso de cualquier marca registrada en este texto no confiere al autor ningún derecho de propiedad sobre tales marcas. Matlab®, Simulink®, Simulink desktop real-time®, Stateflow® son marcas registradas de The MathWorks™. SolidWorks® (3D CAD) es propiedad de Dassault Systèmes SolidWorks Corp. Dynamixel® y todos sus componentes son propiedad de Robotis Co. Ltd.

Se mencionan únicamente con fines didácticos y de identificación, además todas las simulaciones se realizan en la versión de prueba de cada producto.

- Matlab® software de prueba:

https://la.mathworks.com/programs/trials/trial_request.html?s_iid=hp_ff_p_trial

- Simulink® software de prueba:

https://la.mathworks.com/products/simulink.html

- Simulink desktop real-time® software de prueba:

https://la.mathworks.com/products/simulink-desktop-real-time.html?s_tid=srchtitle

- Stateflow® software de prueba:

https://la.mathworks.com/products/stateflow.html

- SolidWorks® (3D CAD) software de prueba:

http://www.solidworks.es/sw/purchase/product-demonstration.htm?mktid=5486

Los demás softwares utilizados para el desarrollo de los experimentos son open source (código abierto) o ediciones express gratuitas.

CAPÍTULO 1.- PLANTEAMIENTO DEL PROBLEMA

1.1 Planteamiento del problema

Uno de los principales objetivos en la robótica y otras disciplinas es resolver problemas de una manera certera y rápida, sin necesidad de gran cantidad de procesamiento y almacenamiento de información, en la actualidad existen diferentes algoritmos de inteligencia artificial, pero estos son desarrollados en base a complejos modelos matemáticos y que requieren varias iteraciones, lo que conlleva a un gran gasto computacional.

Por otro lado, la intuición humana es un proceso mental inconsciente dirigido a resolver problemas sin necesidad de utilizar un proceso de toma de decisiones racional o iterativo ya que ofrece soluciones únicas caracterizadas por ser rápidas y certeras, mientras tanto, la intuición artificial es una representación limitada de la intuición humana (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014) y los modelos intuitivos son capaces de resolver problemas eficaz y eficientemente con el fin de ser implementados en máquinas.

En este trabajo se implementa el modelo del espacio intuitivo de traslado (lugar geométrico) dado en la tesis doctoral de (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014), ampliando así su campo de aplicación. Para mejorar la planeación, certeza y rapidez de un manipulador móvil en tareas de traslado dentro de un espacio inteligente, reduciendo la complejidad del conjunto de instrucciones necesarias para resolverla.

1.2 Hipótesis

Toda investigación, sea cuantitativa (aquella que usa la recolección de datos con base en la medición numérica y el análisis estadístico) o cualitativa (aquella que usa la recolección de datos sin medición numérica) busca resolver un problema, pero para poder dar solución a un problema antes debemos determinar qué es lo que perseguimos probar con la investigación, por ello es conveniente formular una hipótesis, se presenta la hipótesis correspondiente a esta investigación:

"La intuición artificial existe como un lugar geométrico que representa a la intuición humana en la ejecución de actividades de traslado, siendo programable y transferible a un manipulador móvil dentro de un espacio inteligente"

1.3 Objetivos de la investigación

1.3.1 Objetivo general

Aplicar el modelo del espacio intuitivo de traslado (lugar geométrico) en el algoritmo coordinante del manipulador móvil, para desarrollar la tarea “pick and place” dentro de un espacio inteligente.

1.3.2 Objetivos específicos

- Adquirir el material necesario (acrílico, tornillos, servomotores, etc.) para poder construir y ensamblar un manipulador móvil.
- Construir el manipulador móvil basado en el diseño de (Escobedo Castillo, 2012), pero sustituyendo el brazo serial por uno ensamblado con servomotores de la marca Dynamixel serie AX-12A.
- Implementar un módulo electrónico para poder establecer la comunicación entre la placa arduino mega 2560 R3 y los servomotores Dynamixel serie AX-12A.
- Acoplar los módulos electrónicos, de comunicación y energía en el manipulador móvil.
- Programar y establecer la comunicación entre la placa arduino mega 2560 R3 + arduino WiFi shield, el programa Simulink de Matlab®, el software ReacTIVision y la placa controladora MD25 para motorreductores EMG30.
- Establecer los modelos cinemáticos tanto de la plataforma móvil como del brazo serial.
- Reprogramar el algoritmo de coordinación presentado en la tesis de licenciatura de (Escobedo Castillo, 2012) para realizar las simulaciones en tiempo real y mediante conexión inalámbrica WiFi-UDP.
- Programar en el bloque para el control del brazo serial el lugar geométrico basado en los patrones que genera la intuición humana en tareas de traslado.
- Coordinar el envío y recepción de datos.
- Montar y configurar el espacio inteligente en el cual se desarrolla la tarea de traslado.
- Realizar pruebas y ajustes.
- Recabar resultados y redactar conclusiones.

1.4 Justificación

La historia de la creación humana de máquinas "inteligentes" es similar a la historia de la evolución en la naturaleza. Primero surgieron máquinas u organismos cuyo propósito era solo uno, por lo tanto, eran inflexibles. Un ejemplo son las máquinas de cálculo del siglo XIX que solo servían para una cosa y no eran reprogramables sin cambiar el hardware. Estas máquinas se fueron desarrollando progresivamente hasta hacerse reprogramables, adaptativas e inteligentes (Ponce Cruz, 2010).

La psicología y la filosofía han dado las teorías para desarrollar nueva tecnología en base al estudio del consciente, en general la inteligencia artificial ha sido la encargada de aplicar los métodos de la *inteligencia humana consciente* como *inteligencia artificial*, estos métodos son una respuesta al deseo de aproximar el comportamiento y el pensamiento humano a diversos sistemas para la solución de determinadas problemáticas. Por ello, no es de sorprender que actualmente se tiene sistemas muy avanzados que pueden emular ciertas características humanas, sin embargo, aún nos encontramos muy lejos de poder recrear algunas otras (Ponce Cruz, 2010). En la actualidad los métodos de la inteligencia artificial (IA) tienen un gran auge y muchos investigadores se encuentran estudiando nuevas alternativas en el área. Hoy en día es común el empleo de sistemas que utilizan la IA para su funcionamiento cotidiano, entre ellos los equipos electrodomésticos como lavadoras, hornos de microondas, cámaras de video, e inclusive sistemas de transporte. Lo que se pretende con estos métodos en ingeniería es resolver los problemas, no solo de una manera novedosa, sino sobre todo tener mejores soluciones, más eficientes y mejor planeadas (Ponce Cruz, 2010). Los temas más importantes que abarca la inteligencia artificial son los siguientes (Ponce Cruz, 2010):

- Lógica difusa
- Las redes neuronales
- Los sistemas neuro-difusos
- Los algoritmos genéticos

El problema principal de la inteligencia artificial se basa en generar sistemas eficientes y con una gran precisión, debido al gran procesamiento de datos comúnmente las soluciones resultan demasiado lentas y con un alto costo computacional en el procesamiento de todas las variables, además requieren la inversión de mucho tiempo para el aprendizaje y evolución de los algoritmos programados en el sistema. Por otro lado, un tema que ha generado diversos debates y teorías a través de la historia en las ramas de la filosofía y psicología es el estudio de la mente inconsciente, en especial la intuición humana, comúnmente a la intuición humana alojada en el inconsciente se le atribuyen las siguientes características:

- Ofrece igual o mejor eficacia que el consciente en problemas complejos (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).
- No flaquea ante la complejidad de una situación (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).
- Libera a la consciencia y se ocupa de las cosas automáticamente (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).
- Se parte de la premisa que la solución ya existe y por lo tanto no requiere procesamiento.
- Ofrece la solución óptima que ha sido producto del aprendizaje a través de toda la evolución humana. (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).
- Las intuiciones guían muchas de nuestras acciones y suelen desembocar en resultados que nos satisfacen, y de los cuales generalmente no podemos ofrecer una explicación (Gigerenzer, 2008).
- La intuición se basa en el contexto de la situación y se rige por la elección de una regla general y, por lo tanto, el éxito o fracaso depende de la regla general de cada individuo (Gigerenzer, 2008).

En base a las grandes ventajas que ofrece la intuición, en este trabajo de investigación se busca implementar pequeños fragmentos de la intuición humana como intuición artificial y aplicarlo en sistemas robóticos, específicamente en la tarea de traslado de materiales de un manipulador móvil dentro de un espacio inteligente.

1.5 Alcances

El alcance de esta investigación se limita a la aplicación del modelo del espacio intuitivo de traslado dado en (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014), con el fin de mejorar la planeación, certeza y rapidez de un manipulador móvil en tareas de traslado de objetos dentro de un espacio inteligente.

1.6 Metodología

La metodología que se utiliza como guía para el desarrollo del presente trabajo mostrada en la figura 3, se basa en los apuntes del Dr. Víctor Javier González Villela:

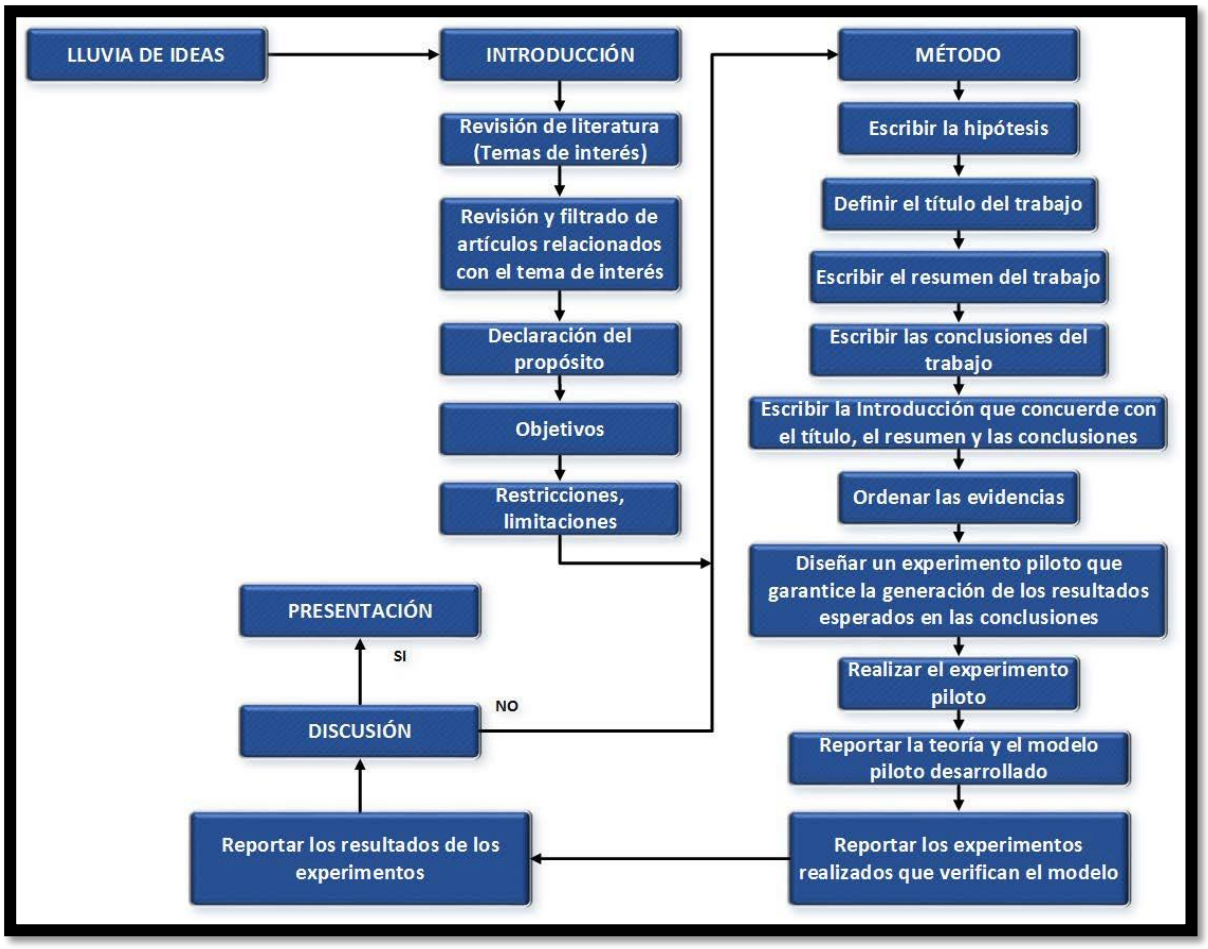


Figura 3. - Metodología propuesta para el desarrollo de la investigación

CAPÍTULO 2.- MARCO TEÓRICO

2.1 Introducción al capítulo

A lo largo del presente capítulo se muestran todos los conceptos necesarios para una buena comprensión por parte del lector respecto al tema que se expone en la presente tesis, como se mencionó en la introducción, no se pretende profundizar en el tema ni crear un debate, ya que en la actualidad existen conceptos que aún en la comunidad científica no se ha llegado a ningún acuerdo, como en el caso de la definición de "mente". Sin embargo, cada concepto se fundamenta en una búsqueda basada en fuentes actuales.

2.2 Espacios inteligentes

La inteligencia de manera general es definida como la capacidad de entender o comprender, de acuerdo con la teoría vigente postulada en 1983 por Howard Gardner (Psicólogo estadounidense, galardonado con el premio príncipe de Asturias de ciencias sociales en 2011)" (Vargas Medina, 2004). De manera general para poder hablar de un espacio inteligente estrictamente tendría que haber una capacidad del propio espacio para comprender lo que ocurre dentro de sí mismo.

2.2.1 Definición de espacio inteligente

Los espacios inteligentes son habitaciones o áreas capaces de percibir y entender lo que ocurre en ellas; están equipados con diferentes tipos de sensores y dispositivos de comunicación, a través de los sensores, los "Espacios inteligentes" captan lo que ocurre en ellos, analizan las diferentes situaciones, reaccionan ante ellas y se comunican con los usuarios a través de los actuadores (Kazuyuki Morioka, Joo-Ho Lee, & Hideki Hashimoto, 2008). En estos espacios los seres humanos y máquinas colaboran entre sí; deben extraer y mantener una consciencia de los eventos y actividades humanas que se producen dentro del entorno (M. Manubhai, K. S. Huang, & I. Mikic, 2005).

2.2.2 Cualidad de un espacio inteligente

Lo que hace que un espacio sea denominado inteligente es cuando toma decisiones de manera autónoma. En el presente trabajo se habla de un espacio inteligente porque será capaz de "ser consciente" de las cosas que ocurren dentro del espacio de trabajo, y en base a ello tomar decisiones dependiendo de la tarea a desarrollar, como primera iteración el transporte de objetos (Carlos Ávila, 2015).

2.2.3 Definición de coexistencia y cooperación

Coexistencia: El humano y el robot comparten el mismo espacio de trabajo al mismo tiempo mientras trabajan en tareas diferentes, concepto extraído de la tesis de (Carlos Ávila, 2015).

Cooperación: El humano y el robot comparten el mismo espacio de trabajo al mismo tiempo mientras trabajan en la misma tarea, concepto extraído de la tesis de (Carlos Ávila, 2015).

2.3 Robótica móvil

2.3.1 Definición de robot

Robot se define, de manera formal en la organización internacional para la estandarización (ISO), como un manipulador multifuncional reprogramable, capaz de mover materiales, piezas, herramientas o dispositivos especiales, a través de movimientos variables programados, para el desempeño de tareas diversas (Saha, 2008), aunque también la palabra robot proviene de la palabra checa *robota* que significa *trabajo* (K. S. Fu, R. C. González, & C. S. G. Lee, 1994). Mientras que el sufijo "ica" se refiere a las *técnicas o ciencias* (-ica, 2003). A continuación, se define la palabra robótica desde el punto de vista de varios autores:

1. Técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones (Robótica, 2014).

2. La robótica es un campo de trabajo muy amplio, tratando con investigación y desarrollo en una serie de áreas interdisciplinarias, que incluyen cinemática, dinámica, planificación de sistemas, control, sensores, lenguajes de programación e inteligencia de máquina (K. S. Fu, R. C. González, & C. S. G. Lee, 1994).
3. La robótica es un campo interdisciplinario porque requiere conocimientos en ingeniería mecánica, ingeniería eléctrica, ciencias de la computación y tecnologías de la información para el diseño y desarrollo de un mejor sistema robótico (Saha, 2008).

2.3.2 Definición de robótica móvil

1. "El desarrollo de la robótica móvil responde a la necesidad de extender el campo de aplicación de la robótica, restringido inicialmente al alcance de una estructura mecánica anclada en unos de sus extremos, se trata también de incrementar la autonomía limitando todo lo posible la intervención humana" (Baturone Ollero , 2001).
2. La robótica móvil puede cambiar su ubicación a través de la locomoción, el tipo más común de robot móvil es un vehículo guiado automáticamente (Nehmzow, 2003).

En base en el análisis de las definiciones anteriores podemos decir que la "Robótica móvil" es una técnica interdisciplinaria que utiliza conocimientos de ingeniería mecánica, electrónica, ingeniería eléctrica e informática, enfocándose principalmente en las áreas de cinemática, dinámica, planificación de sistemas, control, sensores, lenguajes de programación e inteligencia de máquina para poder diseñar y desarrollar sistemas robóticos capaces de cambiar su localización a través de la locomoción, buscando siempre que el sistema sea autónomo, un claro ejemplo es el AGV (vehículo guiado automáticamente) que es un tipo de robot móvil que posee ruedas para su locomoción.

2.3.3 Robot móvil

Los robots móviles son robots con sistemas de locomoción, por lo general ruedas, orugas o patas que permiten el desplazamiento de un lugar a otro. Este tipo de robots cuentan con una gran capacidad de desplazamiento de esta forma su espacio de trabajo aumenta considerablemente (Carlos Ávila, 2015). Existen múltiples tipos de robots móviles, el presente trabajo solo se enfocará a robots móviles terrestres.

2.3.4 Holonomía en un robot móvil

Una forma de clasificar a un robot es por su movilidad, es decir si es o no, holonómico. Un robot se dice que es holonómico si puede modificar su dirección instantáneamente y sin la necesidad de haber rotado previamente, esta consideración no toma en cuenta a la masa del robot, por lo que el robot tiene plena movilidad en el plano. En este caso se usará un robot manipulador móvil que tiene una configuración de ruedas diferencial, es decir, cambia de dirección gracias a una diferencia de velocidades de las dos ruedas que posee, esta configuración es no holonómica (Carlos Ávila, 2015).

2.4 Manipulador móvil

Un manipulador móvil es un sistema mecánicamente unido, comprende de una plataforma moviéndose, sujeta a sus restricciones cinemáticas, y a los grados de libertad del brazo manipulador montado en ella. Estos sistemas combinan las ventajas de las plataformas móviles y los brazos robóticos, además, reducen sus desventajas. Por ejemplo, un manipulador móvil tiene más amplio espacio de trabajo cuando posee una plataforma móvil, pues ofrece mayor funcionalidad durante la operación (Escobedo Castillo, 2012). Existen diversos tipos de manipuladores móviles dependiendo del tipo de tarea que se busque que desarrollen.

2.4.1 Tipos de manipuladores móviles

Los manipuladores móviles se pueden dividir por tipo de manipulador y tipo de plataforma móvil. El manipulador puede ser humanoide o bien un brazo manipulador. A su vez la plataforma móvil puede ser omnidireccional, diferencial, etc. entre otros tipos de locomoción (Escobedo Castillo, 2012).

2.5 Espacio de trabajo de un manipulador

El espacio de trabajo W de un manipulador está definido por el conjunto de puntos (volumen de espacio) que pueden ser alcanzados por su efector final (Craig, 2006).

El análisis y evaluación del espacio de trabajo es fundamental en el diseño de geometrías y algoritmos. Para la descripción del espacio de trabajo se usan las siguientes ecuaciones (Lei, S., X. Mingheng, and L. Bo, 2012):

$$\vec{p} = \begin{Bmatrix} x_E \\ y_E \\ z_E \end{Bmatrix} = \begin{Bmatrix} x_E(q1, q2, \dots, qi \\ y_E(q1, q2, \dots, qi \\ z_E(q1, q2, \dots, qi \end{Bmatrix}$$

Figura 4.- Ecuación para la descripción del espacio de trabajo, fuente: (Lei, S., X. Mingheng, and L. Bo, 2012)

$$W = \left\{ \begin{matrix} x_E(q1, q2, \dots, qi) \\ y_E(q1, q2, \dots, qi) \\ z_E(q1, q2, \dots, qi) \end{matrix} \right| qj^{min} \leq qi \leq qj^{max} \Bigg\}$$

Figura 5.- Ecuación para la descripción del espacio de trabajo, fuente: (Lei, S., X. Mingheng, and L. Bo, 2012)

Donde qj son las variables articulares.

2.5.1 Clasificación de los espacios de trabajo

Los espacios de trabajo se clasifican de la siguiente manera según (Craig, 2006) y (Castelli, Ottaviano, & Ceccarelli, 2008):

- *Espacio de trabajo alcanzable (ETA)*. Es el conjunto de posiciones que pueden ser logrados por el punto de referencia del efector final en una orientación.

- *Espacio de trabajo diestro (ETD)*. Son todas las posiciones del punto de referencia del efector final para las cuales todas las orientaciones son posibles.

- *Espacio de trabajo totalmente orientable (ETTO)*. Son todas las posiciones del punto de referencia del efector final, pueden ser alcanzadas con todas las orientaciones entre un conjunto definido por rangos en los ángulos de orientación.

- *Espacio de trabajo de orientación (ETO)*. Son todas las posibles orientaciones que se pueden lograr cuando el punto de referencia del efector final está en una posición fija.

- *Espacio de trabajo de orientación fija (ETOF)*. Son el conjunto de posibles posiciones que pueden ser alcanzadas por el punto de referencia del efector final con una orientación especificada.

2.6 Traslado

En la figura 6 se muestra la clasificación de los elementos básicos de manipulación basada en (Latombe, J.C., 1991):

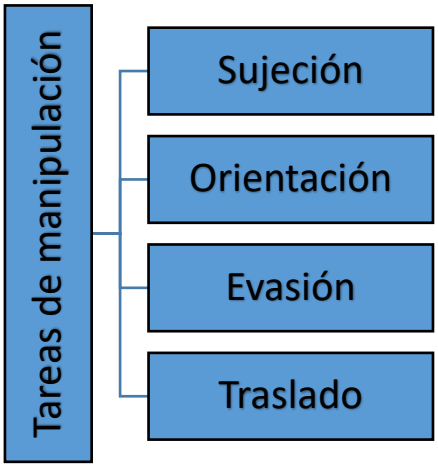


Figura 6.- Clasificación de las tareas de manipulación

Para esta tesis solo se estudia el concepto de traslado, ya que es la tarea principal que desarrolla el manipulador móvil. El concepto es el siguiente:

Traslado: Es el movimiento descrito por el efector final durante un cambio de posición sin tomar en cuenta su orientación, ni otras fracciones de la tarea de manipulación, y se compone de tres elementos de movimiento en el espacio, siendo los tres ejes de posición: x, y, z. El traslado puede estar contenido en cada uno de estos ejes independientemente (x, y, z) o una combinación de ellos (xy, xz, yz, xyz). El traslado requiere de dos especificaciones (Latombe, J.C., 1991):

a) Los puntos en el espacio por los que pasará (lugar geométrico)
b) La información de velocidad y aceleración que se deben cumplir.

2.6.1 Lugar geométrico y trayectoria

El lugar geométrico (locus) es el conjunto de puntos (x, y, z) que cumplen o están determinados por una o más propiedades o condiciones, las cuales pueden ser representadas mediante una ecuación de la forma:

$$f(x, y, z) = 0$$

El conjunto de puntos cuyas coordenadas satisfacen tal ecuación recibe el nombre de gráfica de la ecuación; o bien, su lugar geométrico. Un lugar geométrico puede cumplir con una o más condiciones a la vez. Algunos ejemplos de lugar geométrico son las secciones cónicas en el plano, o las figuras geométricas en el espacio (e.g. esfera, cubo), etc.

Durante el movimiento de un robot existe un historial en el tiempo de la posición, la velocidad y la aceleración que describe cada grado de libertad. *El conjunto de puntos, o el historial de las posiciones, es conocido como el lugar geométrico de la trayectoria. Una vez que se asocian velocidades y/o aceleraciones, el lugar geométrico se convierte en la trayectoria* (Craig, 2006).

2.7 La mente

2.7.1 Mente

(Casacuberta Sevilla, 2001) y (Van Gulick, 2017) afirman que la mente es la responsable del entendimiento, la capacidad de crear pensamientos, la creatividad, el aprendizaje, el raciocinio, la percepción, la emoción, la memoria, la imaginación, la voluntad, *la intuición*, y otras habilidades cognitivas. La mente incluye diversas facultades que le permiten integrar sistemas de recolección de información, procesamiento de información y de ejecución, que responden acorde al dictamen del sistema de procesamiento. *Principalmente hay dos lados reconocidos de la mente: la mente consciente y la mente inconsciente.*

2.7.2 Mente consciente e inconsciente

(G. Myers, 2008) y (Kahneman, A perspective on judgment and choice, 2003) afirman que, desde una perspectiva cognitiva, se considera que la mente funciona por dos vías principales e interrelacionadas: por un lado, está la mente consciente o explícita, discursiva, secuencial, racional, y que requiere un esfuerzo para que funcione. Y por el otro lado, existe una mente inconsciente u oculta, implícita, asociativa, rápida y no requiere de esfuerzo. (Dijksterhuis, Bos, Nordgren, & van Baaren, 2006) afirman que la mente consciente es analítica y convergente, opera con reglas y es exacta, pero se puede desbordar cuando se enfrenta a problemas complejos, en contraposición a la mente inconsciente que apenas se deteriora con situaciones complejas, incluso problemas con múltiples variables.

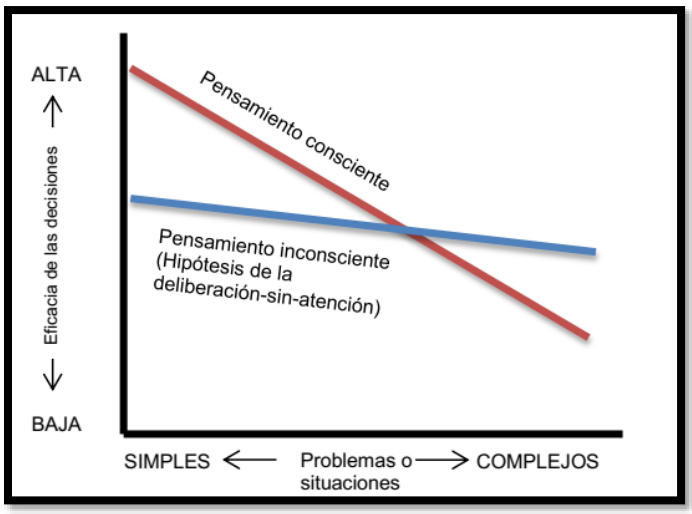


Figura 7.- Relación entre complejidad y calidad de una decisión, según se afronte mediante el pensamiento consciente o inconsciente, fuente: (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).

El inconsciente es la base de lo que emerge en la consciencia, el inconsciente puede procesar hasta 11,000,000 bits/s y necesita 100 ms para hacer todos los procesos pertinentes ante la información que recibe. En contraste con el consciente que procesa información a 50 bits/s y le toma 250 ms para tomar consciencia de la información recibida (Hassin, R.R., J.S. Uleman, & J.A. Bargh, 2005).

La intuición es uno de los mecanismos de la mente inconsciente fortalecida por la experiencia, lo que ayuda al ser humano a llevar a cabo un gran número de tareas.

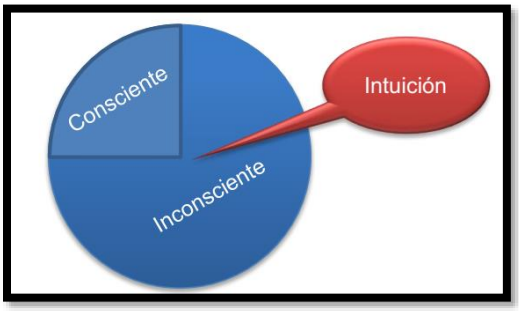


Figura 8.- El consciente, el inconsciente y la intuición, fuente: (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014)

En la figura 9 se muestra una propuesta desarrollada por el Dr. Víctor Javier González Villela y editada por el autor de la presente tesis donde se plantea la interacción de los atajos de la mente cuando se toma una decisión o se resuelve un problema. Lo expuesto en la figura 9 no trata de generalizar los procesos cognitivos ya que intervienen otros factores, pero si se busca brindar un panorama básico del flujo de información:

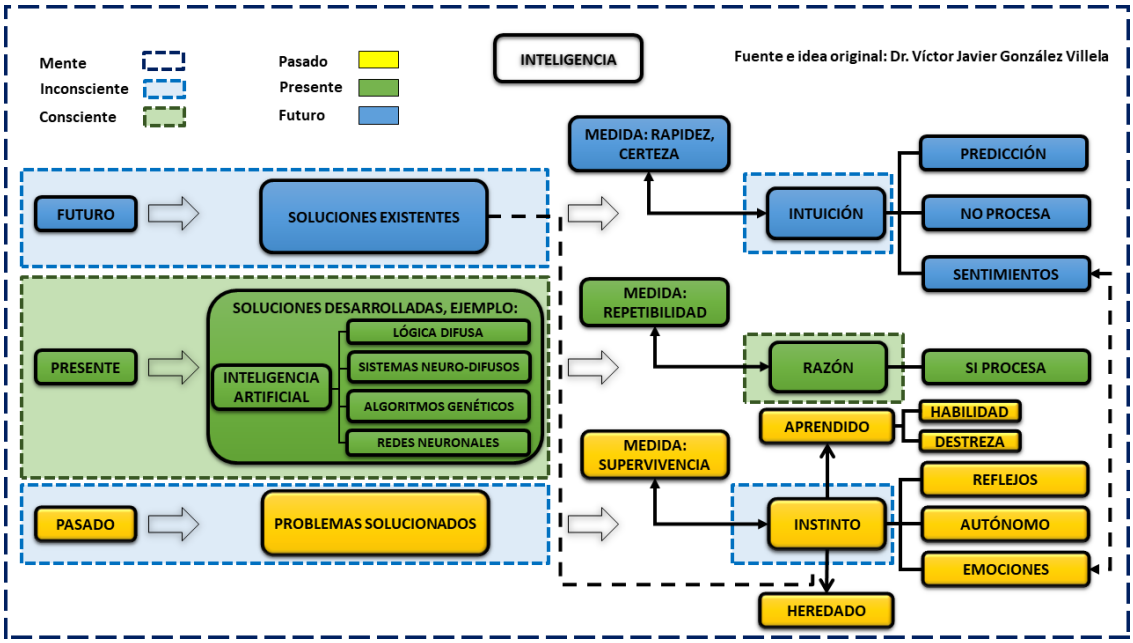


Figura 9.- Propuesta del proceso mental para la toma de decisiones

2.8 Intuición

En la actualidad, el tema de la intuición se ha ido derivando desde el campo de la especulación filosófica al campo de la ciencia positiva, siendo considerado más bien un tema de investigación psicológica y neurológica. La intuición se usa para lidiar con problemas diarios como el control de movimiento muscular y predecir posiciones de los miembros, para entender y generar el habla, leer, analizar lo percibido por los sentidos, también desde manipular objetos con las manos hasta pilotar un vehículo, etc. En el lenguaje popular suele significar con frecuencia presentimiento o "saber sin saber cómo se sabe" (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).

(Kahneman, A perspective on judgment and choice, 2003), (Morsella & Poehlman, 2013), (Hogarth, Educating intuition, 2001), (Erlhoff, Marshall, & Becker, 2008), (Goldstein, 2005), (Krippendorff, 1986), (Ferrater Mora, 1984), (Hogarth, Deciding analytically or trusting your intuition?: The advantages and disadvantages of analytic and intuitive thought, 2002), (Harteis, Koch, & Mergenthaler, 2008), (Kahneman, Pensar rápido, pensar despacio, 2012), (Simon & Frantz, 2003), (Seligman & Kahana, 2009) y (Kant, 2002) afirman que la palabra intuición proviene del latín, *"intueri"*, que significa *"a considerar"* o *"mirada a"*, por lo que podría entenderse como una percepción, pero además como un conocimiento implícito de la realidad en la mente inconsciente de los seres humanos. Los mecanismos de intuición realizan un reconocimiento de símbolos o señales que es llevado a cabo sin el procesamiento consciente o racional del individuo. Después del reconocimiento, automáticamente se sintetizan juicios, pensamientos, decisiones o acciones acorde a la información procesada. Este tipo de juicios son caracterizados por ser apropiados, certeros, coherentes, rápidos e incluso evidentes.

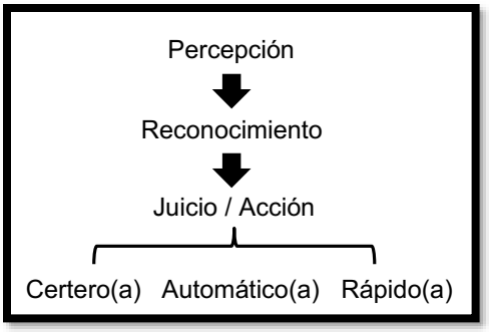


Figura 10.- La intuición, fuente: (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014)

La intuición puede ser organizada como se muestra en la figura 11, con el fin de establecer la analogía con la arquitectura común de agentes artificiales, incluyendo robots, es decir entradas (e.g., sensores), procesos (e.g., microcontroladores y programas) y salidas (e.g., actuadores) (Díaz Hernández & González Villela, Analysis of human intuition towards artificial intuition synthesis for robotics, 2015).

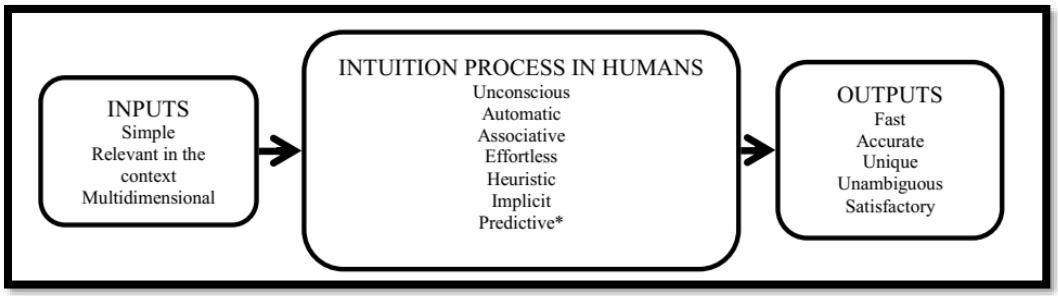


Figura 11.- Intuición como un sistema, fuente: (Díaz Hernández & González Villela, Analysis of human intuition towards artificial intuition synthesis for robotics, 2015)

2.8.1 La intuición aplicada a sistemas artificiales en la actualidad

Hasta el momento se ha establecido que la intuición artificial está basada en micro-intuiciones programadas como algoritmos de bajo gasto computacional (Anderson, 2007), también que la intuición artificial puede ser el diseño de software cooperativo entre programadores y usuarios (Weidong & H. Ping, 2009).

Además, se ha propuesto que la intuición artificial puede programarse como un reconocimiento de patrones usando técnicas convencionales de inteligencia artificial (Dundas J. & D. Chik, 2013). Actualmente en el MIT (Massachusetts Institute of Technology) investigadores de ciencias de la computación e inteligencia artificial trabajan en mejorar e implementar algoritmos para la planeación automática de rutas, basados en los beneficios que otorga la intuición humana, los resultados muestran haber mejorado el rendimiento de los algoritmos de planeación entre un 10% y 15%, en comparación a los antiguos algoritmos implementados (Researchers add a splash of human intuition to planning algorithms, 2017).

Es importante recalcar que ninguno de los autores previos ha definido claramente a la intuición artificial, además, aún se requiere una metodología para adquirir datos relacionados a la intuición humana (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).

2.9 Intuición artificial

Según (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014), la intuición artificial puede definirse como:

- La intuición artificial es un área multidisciplinaria que, a través de ciencias, tales como la informática, psicología y robótica, estudia la creación y diseño de entidades capaces de emular ciertos rasgos de la intuición humana.
- La intuición artificial es una rama de la inteligencia artificial, que se dedica a emular el reconocimiento automático de patrones de información para generar respuestas rápidas y normalmente certeras o confiables.
- La intuición artificial es una representación limitada de las capacidades intuitivas del ser humano.
- La intuición artificial es un procesamiento basado en el reconocimiento automático de patrones, lo que lo hace extremadamente rápido y no hace uso exhaustivo de la capacidad de memoria ni de procesamiento en la búsqueda razonada o iterativa de soluciones.

- La intuición artificial emula la vía intuitiva de la obtención de soluciones que no necesitaron un largo proceso de búsqueda, más bien hay reconocimiento de fragmentos relevantes de información que conllevan al descubrimiento de una solución que mejorará la calidad de las respuestas.
- La intuición artificial funciona en base a un conjunto de algoritmos sintetizados a partir del estudio de las capacidades humanas intuitivas de realizar una tarea en condiciones que estimularon la intuición. Estos algoritmos representan a mecanismos inconscientes compuestos por fragmentos o plantillas de conocimiento pre-adquirido, las funciones de cada algoritmo funcionan independientemente o se presentan bajo el principio de superposición.
- La intuición artificial está embebida en los elementos de procesamiento del agente artificial como parte de su arquitectura.
- La intuición artificial se implementa con ecuaciones o fórmulas, cuyas entradas son piezas clave o relevantes de información para la tarea, lo que cumple con la propiedad de ser un procesamiento automático.
- La intuición artificial encuentra la respuesta a un problema en cuanto recibe las entradas, discriminando otras posibles soluciones, en consecuencia, se disminuye el tiempo de búsqueda y la cantidad de procesamiento.

La intuición artificial, en robótica, es un subespacio de configuraciones específicas por una tarea realizada con intuición, al que llamamos Espacio Intuitivo de la Tarea (EIT), o $A(q)_{\text{intuición}}$.

$$A(q)_{\text{intuición}} \subseteq A(q)_{tarea} \subseteq W \subseteq Mundo$$

Ecuación 1

Se denomina subespacio debido a que es parte del espacio de trabajo de la tarea, que a su vez pertenece al espacio de trabajo del robot.

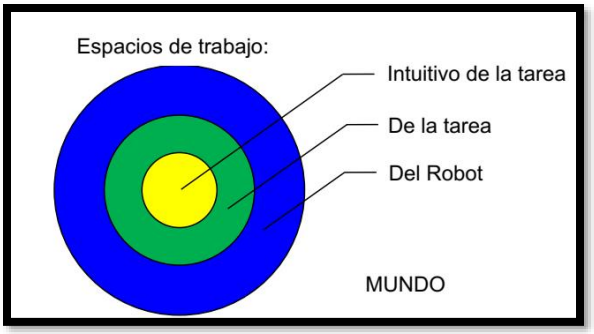


Figura 12.- Espacio intuitivo de la tarea (EIT), fuente: (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014)

2.9.1 Clasificación de la intuición artificial

Según (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014), la intuición artificial puede organizarse en dos tipos de algoritmos para programar un ente artificial:

I. Algoritmos de intuición artificial primitiva (InAP):

- Se le reconoce como primitivo a aquel algoritmo que el "diseñador" del agente artificial considera necesario como una estructura elemental y necesaria para la continuidad y preservación del agente artificial. Una de las características es que su existencia en la interacción del humano con el agente artificial se reemplaza un componente reactivo del "operador" (humano o artificial) durante una tarea. Los algoritmos de InAP representan al conjunto de atajos heurísticos desarrollados a lo largo de la evolución.

II. Algoritmos de intuición artificial adquirida (InAA):

- Se le llama adquirido al algoritmo que fue definido a partir de la tarea a ejecutar, se asocia a la repetición de eventos y estos algoritmos pueden ser inspirados en la intuición para ejecutar tareas por el humano. Su finalidad es que reemplace uno o más elementos deliberativos del "operador" (humano o artificial) empleados en una tarea. Los algoritmos de InAA representan a las asociaciones aprendidas a lo largo de una historia vital.

Los algoritmos así organizados pueden diseñarse por separado para generar una respuesta de intuición artificial; pueden coexistir más de uno en cada tipo y requieren ciertas reglas de interacción.

La intuición artificial no posee mecanismos de aprendizaje: si la intuición artificial debe ser aprendida automáticamente por el ente artificial, pierde el sentido de ser un proceso inconsciente, porque rompe con el esquema de ser un procesamiento rápido y de bajo costo "computacional" (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).

2.9.2 Síntesis de la intuición

El diseño del experimento para capturar la intuición humana y sintetizarla como intuición artificial conlleva la siguiente metodología (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014):

- Capturar datos provenientes de las habilidades intuitivas del humano durante el desarrollo de la tarea a estudiar (en este caso el traslado de objetos).
- Analizar la información contenida en los datos, en busca de patrones y expresarlos en términos de la robótica.
- Usar la información para sintetizar ecuaciones programables en un sistema robótico.

CAPÍTULO 3.- CONSTRUCCIÓN DEL MANIPULADOR MÓVIL

3.1 Introducción al capítulo

En este capítulo se abarca la fase que consiste en construir al manipulador móvil para realizar los experimentos dentro del espacio inteligente, así como para demostrar el algoritmo de coordinación y principalmente el espacio intuitivo (lugar geométrico) programado para llevar a cabo la tarea de traslado de objetos. La construcción del manipulador móvil se basa en el diseño desarrollado por (Escobedo Castillo, 2012), por lo que se conservan las mismas propiedades y especificaciones para la construcción de la plataforma móvil, sustituyendo el brazo serial por uno ensamblado con servomotores Dynamixel de la serie AX-12A, con el fin de lograr una mayor precisión y otras características que se explican a lo largo del capítulo.

3.2 Requerimientos y especificaciones

Para establecer las especificaciones del manipulador móvil se utiliza la teoría de dominios, la cual afirma que la síntesis de máquinas consiste en el establecimiento sucesivo de cuatro sistemas, cada uno corresponde a un dominio de trabajo (mental), este sistema representa cuatro diferentes aspectos de las máquinas:

1. Sistema de proceso
2. Sistema funcional
3. Sistema orgánico
4. Sistema de partes

Antes de describir cada sistema de la teoría de dominios, se definen las tareas a desarrollar por el manipulador móvil dentro del espacio inteligente.

3.2.1 Descripción de la tarea

La tarea principal consiste en trasladar a un objeto de una posición y orientación inicial a una final, mediante un brazo serial antropomórfico montado sobre una plataforma móvil en configuración diferencial, con movimientos coordinados basados en el modelo matemático del espacio intuitivo de traslado y las técnicas “pick and place”, “docking and predocking” y campos potenciales.

3.2.2 Tarea local

Los pasos que se deben seguir para el desarrollo de la tarea son:

1. Trasladar desde una configuración inicial (punto inicial) al efector final (gripper) del manipulador con cierta posición y orientación a un punto final, sujetar o colocar el objeto con el gripper según sea el caso y finalmente trasladarse a su configuración inicial, *la tarea de traslado es descrita por un lugar geométrico basado en el estudio de habilidades intuitivas de personas y extraído a partir de patrones observados en tareas de traslado*.

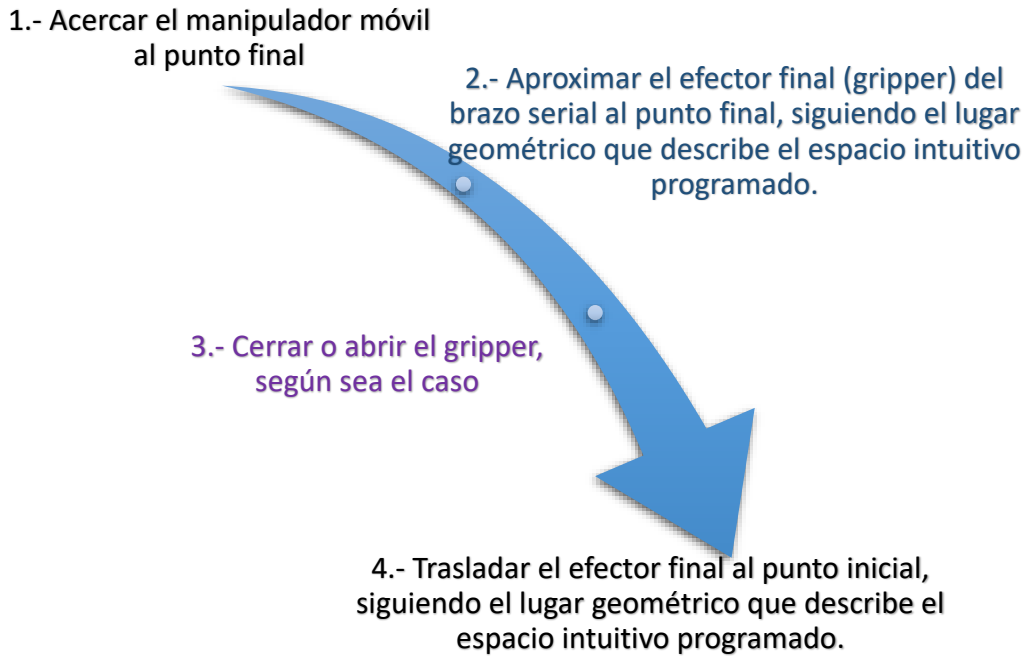


Figura 13.- Resumen tarea local

3.2.3 Tarea global

La tarea global se lleva a cabo cuando se sujeta y transporta un objeto a una meta distante, los pasos a seguir para el desarrollo de la tarea son:

1. Acercar el manipulador móvil en su configuración preferida al objetivo o punto inicial donde se encuentra un objeto con ciertas dimensiones, dicho objeto se caracteriza por tener posición y orientación dentro del espacio inteligente.
2. Sujetar al objeto con el efector final (gripper) del brazo serial antropomórfico.
3. Una vez tomado el objeto, regresar el brazo a su posición inicial.
4. Mover al manipulador móvil a la meta o punto final mediante el sistema de visión.
5. Una vez situado el manipulador en el punto final donde se quiere depositar al objeto, ejecutar el movimiento para posicionar el objeto.
6. Abrir el gripper para colocar al objeto.
7. Por último, regresar el brazo serial a su posición inicial.

Es importante señalar que la planeación de la trayectoria de ida y regreso del brazo serial en la tarea de sujetar y colocar al objeto, es dada automáticamente en base al lugar geométrico o espacio intuitivo programado.

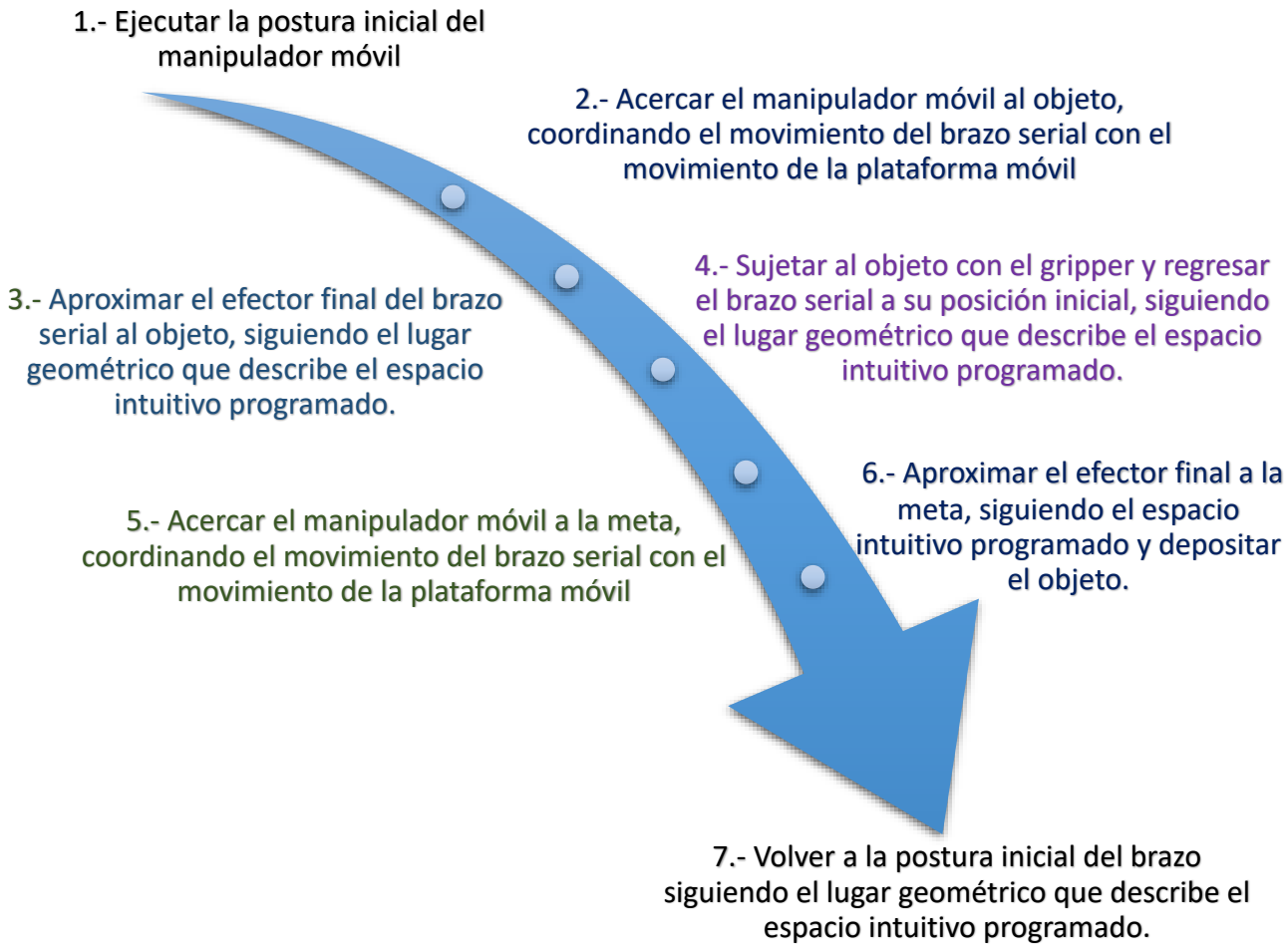


Figura 14.- Resumen tarea global

3.3 Sistema de proceso

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

Procesos:

1. Se le indica al manipulador móvil a través de un sistema de visión las distancias y orientaciones iniciales entre el manipulador y los objetivos (objeto a manipular y meta).
2. Se acopla el brazo serial y la plataforma móvil en un solo punto de referencia, con el fin de coordinar su movimiento hasta la posición y orientación en la cual se encuentra el objeto y la meta.
3. Se calcula la velocidad y orientación necesaria de la plataforma móvil para llegar al punto deseado, en este caso el objeto.
4. Se definen y ejecutan los movimientos que debe seguir el manipulador móvil dentro del espacio inteligente mediante el algoritmo de coordinación.
5. Se calcula el espacio intuitivo que debe seguir el efector final para sujetar al objeto.
6. Con cinemática inversa se determinan las posiciones de cada una de las articulaciones del brazo serial para llegar al objeto y sujetarlo.
7. Se calcula el espacio intuitivo que debe seguir el efector final para regresar a su posición inicial con el objeto sujetado.
8. Mediante cinemática inversa se determinan las posiciones de cada una de las articulaciones del brazo serial para que regrese a su posición articular inicial.
9. Termina la ejecución de la tarea local.
10. Se repiten los pasos del 3 al 8 solo que ahora el punto deseado es la meta y en lugar de sujetar el objeto este es depositado.
11. Termina la ejecución de la tarea global.

Se comenta que el sistema es retroalimentado, es decir, si existe alguna variación en las posiciones y orientaciones estas se actualizan en tiempo real.

3.4 Sistema funcional

El sistema funcional es una estructura de funciones propósito o efectos necesarios en las máquinas para establecer las transformaciones deseadas.

Esto es mejor entendido como la transformación de entradas a salidas, en este paso se muestran las funciones esenciales para cumplir el proceso:

Funciones:

1. Se le indica al manipulador móvil a través de un sistema de visión las distancias y orientaciones iniciales entre el manipulador y los objetivos (objeto a manipular y meta).
    - Función de solicitar posiciones y orientaciones.
    - Función de realizar todos los cálculos.
    - Función de coordinar módulos programados e intercambio de información.
    - Función de procesamiento de datos.
    - Función de cálculo y envío de posiciones y orientaciones.
    - Función de calibración del espacio de navegación.
    - Función de sensado del espacio inteligente.
    - Función de recepción de datos.
    - Función de puente con el módulo inalámbrico.
2. Se acopla el brazo serial y la plataforma móvil en un solo punto de referencia, con el fin de coordinar su movimiento hasta la posición y orientación en la cual se encuentra el objeto y la meta.
    - Función de realizar todos los cálculos.
    - Función de navegación y control.
    - Función de coordinar al manipulador móvil.
3. Se calcula la velocidad y orientación necesaria de la plataforma móvil para llegar al punto deseado, en este caso el objeto.
    - Función de realizar todos los cálculos.
    - Función de navegación y control.
    - Función de coordinar módulos programados e intercambio de información.
    - Función de coordinar al manipulador móvil.
    - Función de procesamiento de datos.
    - Función de recepción de datos.

- Función de puente con el módulo inalámbrico.
- Función de envío de datos por WiFi.
- Función de recepción de datos por WiFi.

4. Se definen y ejecutan los movimientos que debe seguir el manipulador móvil dentro del espacio inteligente mediante el algoritmo de coordinación.
    - Función de control de velocidad y voltaje.
    - Función de retroalimentación de posición y voltaje.
    - Función de recepción de datos.
    - Función de filtrado de datos.
    - Función de puente con el módulo inalámbrico.
    - Función de ejecución de instrucciones programadas.
    - Función de envío de datos por WiFi.
    - Función de recepción de datos por WiFi.
    - Función de suministrar el voltaje adecuado a cada componente.
    - Función de proteger al sistema ante variaciones de energía.
    - Función de establecer la correcta comunicación entre el microcontrolador y demás componentes.
    - Función de distribuir energía y conversión de datos entre cada componente.
    - Función de suministrar energía a todo el sistema mediante una batería.
    - Función de coordinar módulos programados e intercambio de información.
    - Función de coordinar al manipulador móvil.
    - Función de procesamiento de datos.
    - Función de medición de variables físicas de cada componente.
    - Función de retroalimentación de posición, temperatura, velocidad, voltaje y corriente.
    - Función de envío de las variables medidas.
5. Se calcula el espacio intuitivo que debe seguir el efector final para sujetar al objeto.
    - Función de cálculo del espacio intuitivo.
    - Función de realizar todos los cálculos.

6. Con cinemática inversa se determinan las posiciones de cada una de las articulaciones del brazo serial para llegar al objeto y sujetarlo.
    - Función de realizar todos los cálculos.
    - Función de envío de datos.
    - Función de procesamiento de datos.
    - Función de medición de variables físicas de cada componente.
    - Función de retroalimentación de posición, temperatura, velocidad, voltaje y corriente.
    - Función de envío de las variables medidas.
    - Función de recepción de datos.
    - Función de filtrado de datos.
    - Función de puente con el módulo inalámbrico.
    - Función de ejecución de instrucciones programadas.
    - Función de envío de datos por WiFi.
    - Función de recepción de datos por WiFi.
    - Función de suministrar el voltaje adecuado a cada componente.
    - Función de proteger al sistema ante variaciones de energía.
    - Función de establecer la correcta comunicación entre el microcontrolador y demás componentes.
    - Función de distribuir energía y conversión de datos entre cada componente.
    - Función de suministrar energía a todo el sistema mediante una batería.
7. Se calcula el espacio intuitivo que debe seguir el efector final para regresar a su posición inicial con el objeto sujetado.
    - Función de cálculo del espacio intuitivo.
    - Función de realizar todos los cálculos.
8. Mediante cinemática inversa se determinan las posiciones de cada una de las articulaciones del brazo serial para que regrese a su posición articular inicial.
    - Función de realizar todos los cálculos.
    - Función de envío de datos.
    - Función de procesamiento de datos.

- Función de medición de variables físicas de cada componente.
- Función de retroalimentación de posición, temperatura, velocidad, voltaje y corriente.
- Función de envío de las variables medidas.
- Función de recepción de datos.
- Función de filtrado de datos.
- Función de puente con el módulo inalámbrico.
- Función de ejecución de instrucciones programadas.
- Función de envío de datos por WiFi.
- Función de recepción de datos por WiFi.
- Función de suministrar el voltaje adecuado a cada componente.
- Función de proteger al sistema ante variaciones de energía.
- Función de establecer la correcta comunicación entre el microcontrolador y demás componentes.
- Función de distribuir energía y conversión de datos entre cada componente.
- Función de suministrar energía a todo el sistema mediante una batería.

9. Termina la ejecución de la tarea local.
10. Se repiten los pasos del 3 al 8 solo que ahora el punto deseado es la meta y en lugar de sujetar al objeto este es depositado.
    - Función de coordinar al manipulador móvil.
11. Termina la ejecución de la tarea global.

3.5 Sistema orgánico

El sistema orgánico es una estructura de partes (órganos) en la que se realiza una o más funciones a través de efectos físicos.

Órganos:

- Unidad de procesamiento central (CPU)

- Sistema de visión
- Sensores
- Microcontrolador
- Controlador de motores
- Módulo WiFi
- Reguladores de energía
- Módulos electrónicos
- Energía

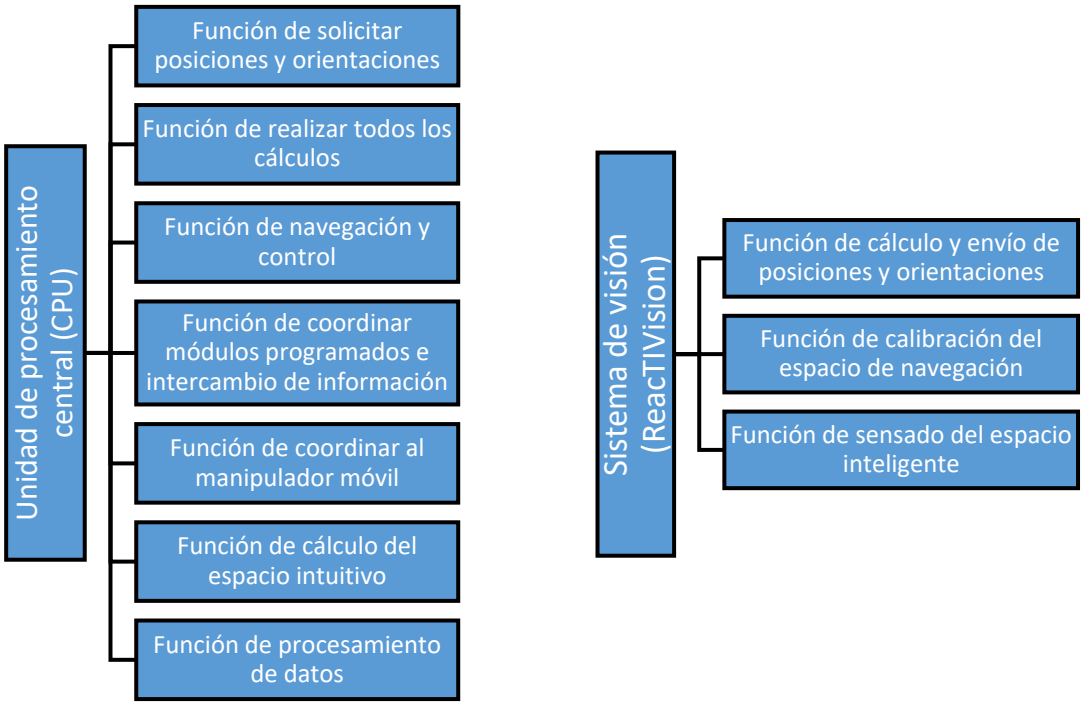


Figura 15.- Funciones de la unidad de procesamiento central (CPU) y el sistema de visión

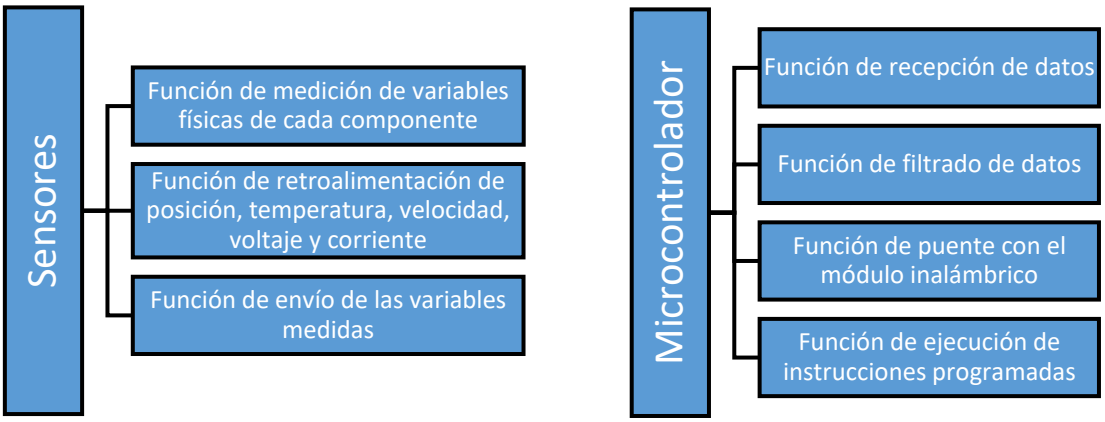


Figura 16.- Funciones de los sensores y el microcontrolador

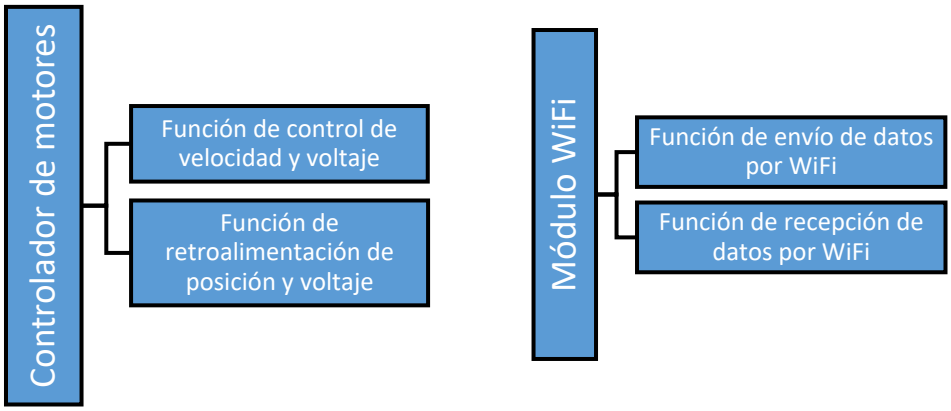


Figura 17.- Funciones del controlador de motores y el módulo WiFi

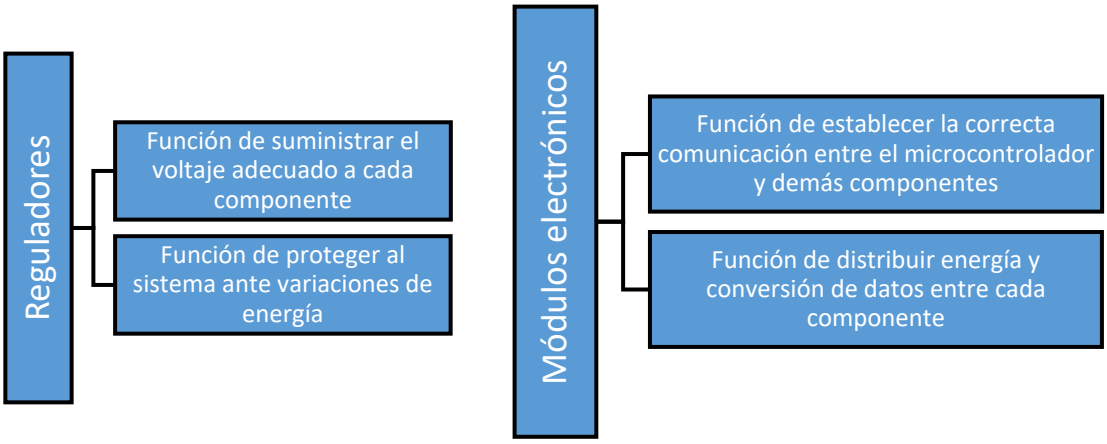


Figura 18.- Funciones de los reguladores y los módulos electrónicos

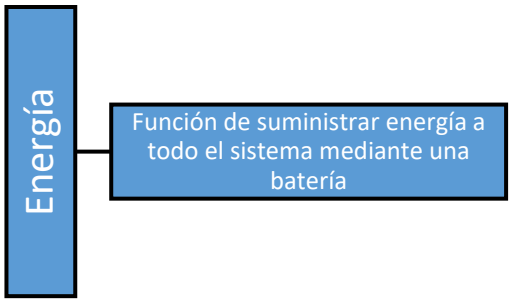


Figura 19.- Funciones del módulo de energía

3.6 Sistema de partes

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

Partes:

- Elementos electrónicos
- Llantas
- Motores
- Servomotores
- Cables
- Conectores
- Tornillos
- Batería
- Estructura de la plataforma móvil
- Elementos mecánicos y de soporte para el brazo serial
- Base del brazo serial
- Gripper
- Soportes
- Espaciadores
- Acoplamientos

En base a los sistemas previamente definidos, se realiza una lista de especificaciones a cumplir para la construcción del manipulador móvil:

Especificaciones:

- ✓ La comunicación entre la computadora y el manipulador móvil deberá ser mediante el protocolo WiFi-UDP.
- ✓ El microcontrolador que se utiliza es una placa arduino mega 2560 R3.
- ✓ Emplear la placa arduino WiFi shield para establecer la comunicación inalámbrica.
- ✓ La alimentación del manipulador móvil deberá proporcionarse mediante una pila de 12 volts y no directamente a corriente alterna.
- ✓ El sistema deberá ser capaz de desplazarse por superficies lisas.
- ✓ El sistema deberá contener todos los componentes dentro de la plataforma móvil.
- ✓ Utilizar la tarjeta controladora MD25.
- ✓ Utilizar motorreductores de corriente continua EMG30.
- ✓ Emplear servomotores de precisión Dynamixel de la serie AX-12A para el brazo serial.
- ✓ Establecer la comunicación entre la placa arduino y los servomotores Dynamixel con el circuito integrado 74LS241.
- ✓ Incluir un sistema de enfriamiento con ventiladores de corriente continua de 5 y 12 volts para los componentes electrónicos.
- ✓ Usar reguladores para proteger al sistema ante variaciones de energía y suministrar el voltaje adecuada a cada componente.
- ✓ El ensamble debe ser modular, con el fin de ofrecer flexibilidad en futuras modificaciones.
- ✓ Utilizar acrílico para el ensamble de la plataforma móvil.
- ✓ Los soportes y demás componentes mecánicos deben ser preferentemente de aluminio o materiales resistentes y durables.

3.7 Concepto y ensamble final de cada subsistema

Las figuras que muestran el concepto de cada subsistema del manipulador móvil y el concepto de ensamble final son realizadas por el escritor de la presente tesis en el software CAD SolidWorks versión de prueba, sin olvidar dar crédito a las aportaciones de la comunidad GRABCAD por el diseño de varias piezas en 3D de libre uso.

El manipulador móvil debe estar formado por tres partes o subsistemas:

- *La plataforma móvil*
- *El brazo serial*
- *El efector final*

La plataforma móvil debe cumplir con la función de desplazamiento dentro del espacio inteligente, en el que mediante un software de visión se lleva al manipulador a la posición que se asigna con unos marcadores llamados fiducials, por lo tanto, tiene que trazar el camino que lleve a la plataforma móvil a la postura que permita cumplir con la tarea. Además de proporcionar movimiento, la plataforma debe de contar con espacio para la batería y todos los componentes electrónicos que controlan al sistema.

El brazo serial debe posicionar al efector final en cualquier lugar de su espacio de trabajo alcanzable y orientarlo para que describa el lugar geométrico o espacio intuitivo que lleva a cada objeto de una posición inicial a una final.

Finalmente, *el efector final* debe cumplir con la función de sujetar y colocar al objeto, lo anterior es propio de la tarea a realizar, es decir, si la tarea es distinta o consiste en sujetar un objeto con geometría diferente, entonces se tiene que elegir o diseñar otro tipo de efector final.

Como se mencionó al inicio del presente capítulo, la construcción del manipulador móvil se basa en el diseño de (Escobedo Castillo, 2012) presentado en su tesis de licenciatura, por lo que si se requiere analizar la metodología y detalles del mismo se recomienda consultar dicha tesis, sin embargo, al ser un diseño con arquitectura modular se sustituye el brazo serial por uno basado en los servomotores Dynamixel de la serie AX-12A y demás componentes que comercializa la empresa Robotis para su correcto ensamblaje, conservando las mismas propiedades y especificaciones para la construcción de la plataforma móvil. También se realiza un reacomodo de los componentes electrónicos dentro de la plataforma, ya que se añaden nuevos dispositivos.

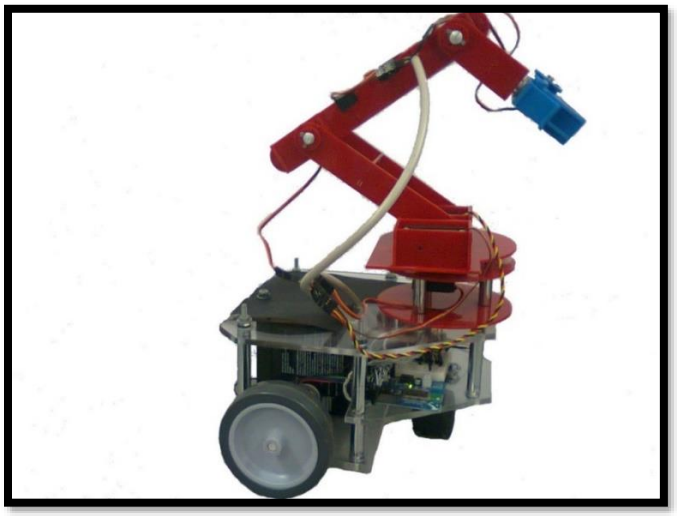


Figura 20.- Manipulador móvil diseñado por (Escobedo Castillo, 2012)

3.7.1 Plataforma móvil

La plataforma móvil conserva el mismo diseño dado por (Escobedo Castillo, 2012), el concepto de la plataforma móvil realizado en el software SolidWorks se muestra en las siguientes figuras:

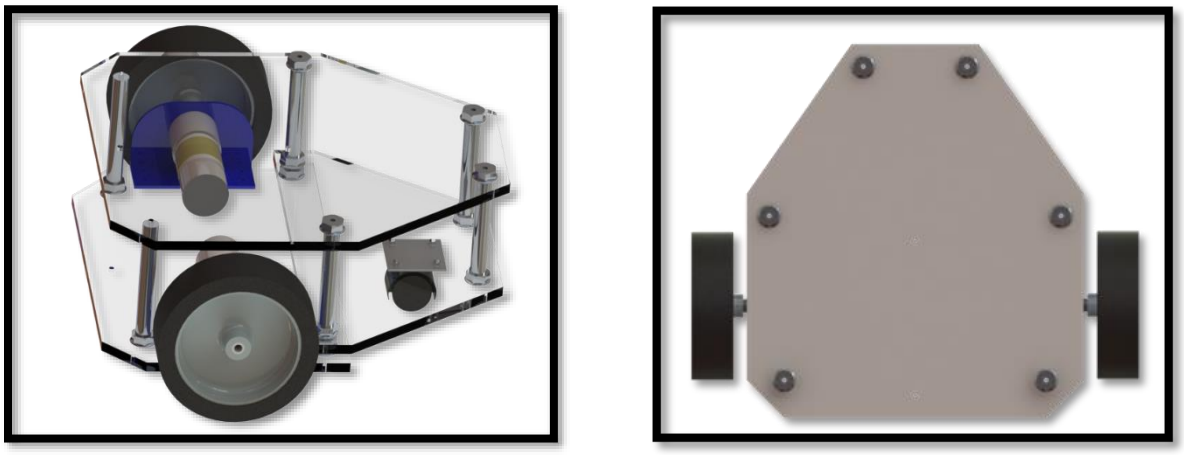


Figura 21.- Concepto final de la plataforma móvil sin componentes

En la figura 22 se observa el concepto final propuesto por el autor para el reacomodo de los componentes electrónicos en la plataforma móvil.

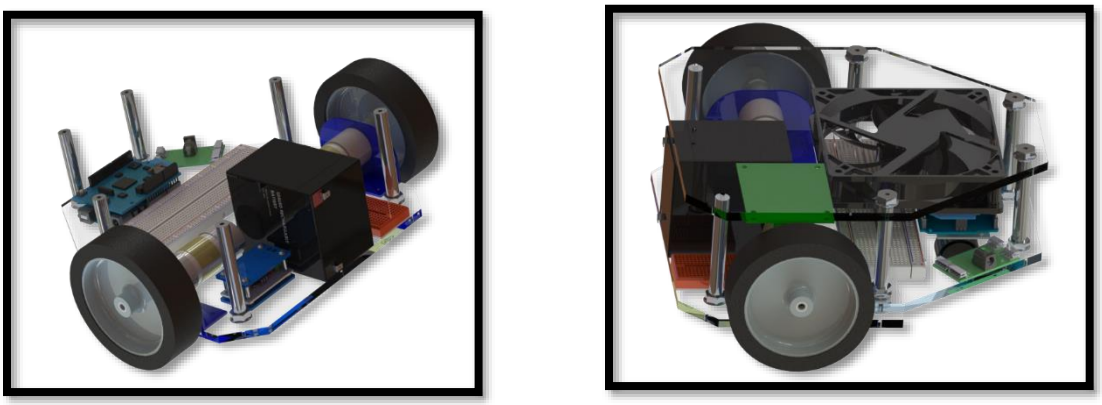


Figura 22.- Concepto final de la plataforma móvil con componentes

Para lograr la locomoción de la plataforma móvil se utilizan los siguientes materiales: 2 motorreductores EMG30, 1 tarjeta controladora MD25 para motorreductores EMG30, 2 llantas de 100 milímetros de diámetro y 2 soportes para los motorreductores.

Figura 23.- Kit para la locomoción de la plataforma móvil, fuente: (RD02 - 12v robot drive, 2009)

En las siguientes figuras se muestra el ensamble final de la plataforma móvil:

Figura 24.- Plataforma móvil sin componentes

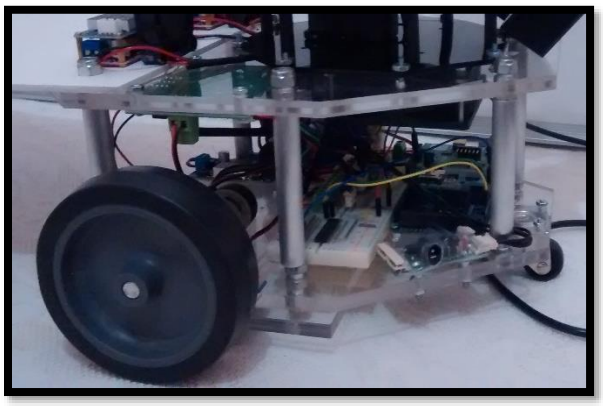


Figura 25.- Plataforma móvil con componentes

3.7.2 Brazo serial

Al sustituirse el brazo serial de tipo antropomórfico por un diseño totalmente nuevo, se elige el concepto basado en el brazo comercializado por la empresa TrossenRobotics.

Figura 26.- Brazo serial TrossenRobotics, fuente: (Robotics, 2012)

Con el fin de poder realizar un ensamble cuyos componentes sean comerciales y de fácil adquisición, se seleccionan los ofrecidos por la empresa Robotis y comercializados por proveedores nacionales. Los servomotores elegidos para posicionar cada articulación del brazo serial son de la marca Dynamixel serie AX-12A, están diseñados específicamente para aplicaciones de robótica y mini robótica, se utilizan por su facilidad en cuanto a programación y ofrecen un alto torque, además de esto son servomotores de uso profesional y con un costo aceptable.

Cada servomotor esta micro controlado de tal manera que se puede tener acceso a diferentes variables: velocidad rotacional, temperatura, carga y voltaje, esto lo logra ya que tiene un microcontrolador ATmega8 el cual se comunica a través de Half Duplex UART-TTL. Estos servomotores están protegidos contra variaciones de voltaje, sobrecalentamiento y condiciones predefinidas de error. Las características que ofrece cada servomotor son las siguientes (Castañeda Espinosa & Pedro Mendoza, 2016):

- Peso: 54.6 gr
- Dimensión: 32 mm x 50 mm x 40 mm
- Resolución: 0.29°
- Ratio de reducción: 254: 1
- Par motor: 1.52 N.m (a 12.0 V, 1.5 A)
- Velocidad sin carga: 59 rpm (a 12 V)
- Grados de giro: 0° ~ 300°
- Rotación continua
- Temperatura de trabajo: -5 ºC ~ +70 ºC
- Tensión de operación: 9 ~ 12 V (tensión de operación recomendada 11.1 V)
- Señal de comandos: paquete digital
- Tipo de protocolo: comunicación serie asíncrona half duplex (8 bit, 1 stop, no parity)
- Conexión física: TTL level multi drop (conector tipo daisy chain)
- ID: 254 ID (0~253)
- Velocidad de comunicación: 7343 bps ~ 1 Mbps
- Feedback: posición, temperatura, carga, tensión de entrada, etc.
- Material: plástico

	AX-12W			AX-12A	AX-18A
Weight	52.9 g (1.86 oz)			54.6 g (1.88 oz)	54.6 g (1.88 oz)
Dimension(mm) / (inch)	32×50×40(mm) 1.25×1.97×1.57(inch)			32×50×40(mm) 1.25×1.97×1.57(inch)	32×50×40(mm) 1.25×1.97×1.57(inch)
Gear Ratio (material)	32 : 1 (enpla)			254 : 1 (enpla)	254 : 1 (enpla+metal)
Network Interface	TTL			TTL	TTL
Position Sensor (Resolution)	Potentiometer (300°/1024)			Potentiometer (300°/1024)	Potentiometer (300°/1024)
Motor	Cored Motor			Cored Motor	Coreless Motor
Operation Voltage (V)	9.0	11.1	12.0	9.0~12.0	9.0~12.0
Stall Torque (N.m)	N/A			1.5 at 12.0V	1.8 at 12.0V
Stall Current (A)	1.1	1.3	1.4	1.5	2.2
No Load Speed (RPM)	360	430	470	59	97

Figura 27.- Características de los servomotores Dynamixel AX-12A, fuente: (AX Series, 2014)

La lista de componentes que se utilizan para el ensamble del brazo serial mostrado en la figura 29 es:

Tabla 1.- Lista de materiales para el ensamble del brazo serial

Cantidad	Descripción
6	Servomotores Dynamixel serie AX-12A
6	Cables para Robot-3P de 180 mm de longitud
3	Brackets metálicos compatibles con Bioloid AX F3, incluye set de tornillos
1	Bracket metálico compatible con Bioloid AX F4, incluye set de tornillos y coples
2	Brackets metálicos compatible con Bioloid AX F2, incluye set de tornillos y coples
1	Set de cables-3P BCS-3P01
4	Adaptadores para servomotores con kit de tornillos y tuercas
1	Tubo redondo de aluminio de alta calidad de 0.50″ de diámetro y 1.5″ de largo, especial para montarse con el HUB 08
1	Tubo redondo de aluminio de alta calidad de 0.50″ de diámetro y 2.25″ de largo, especial para montarse con el HUB 08
2	Par HUB-08 con tornillos
1	Gripper PhantomX Parallel AX-12, con set de tornillos
1	Kit de base giratoria PhantomX para servomotores Dynamixel

El concepto final del brazo a ensamblar se observa en la figura 28:

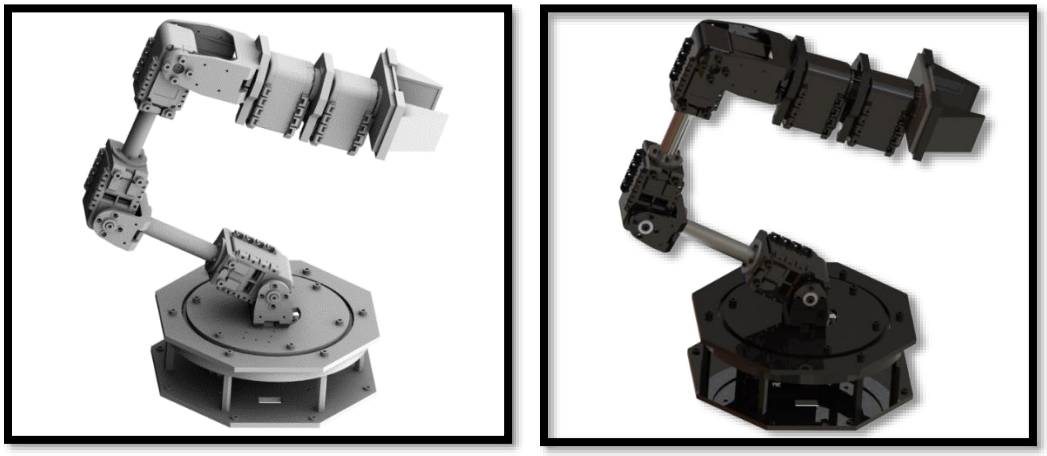


Figura 28.- Concepto en 3D del brazo serial

En la figura 29 se muestra el ensamble final del brazo serial:

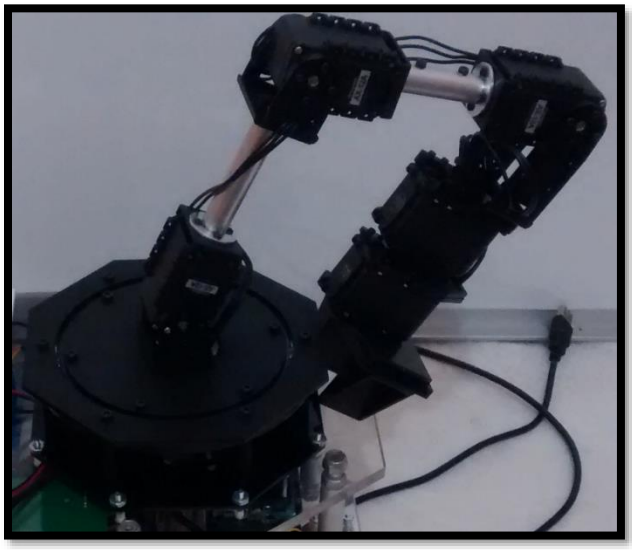


Figura 29.- Brazo serial ensamblado, versión final

3.7.3 Efector final

El gripper o efector final se adquiere como un kit que incluye todos los componentes para poder ensamblarlo. El diseño del gripper seleccionado cumple con su objetivo ya que el objeto por manipular no posee geometría compleja.

Figura 30.- Gripper PhantomX Parallel AX-12, fuente: (PhantomX Parallel AX-12 Gripper, 2012)

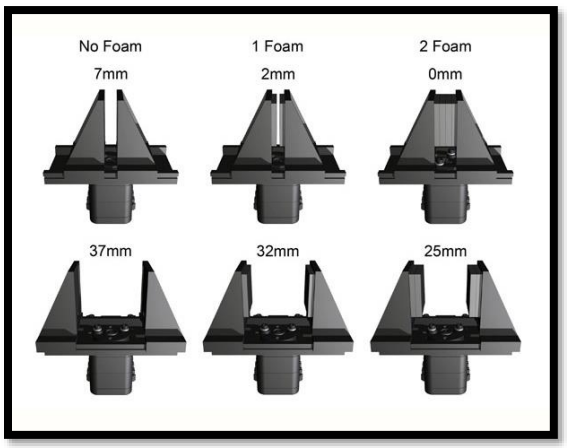


Figura 31.- Rangos de apertura y cierre del gripper PhantomX Parallel AX-12, fuente: (PhantomX Parallel AX-12 Gripper, 2012)

3.8 Ensamble del manipulador móvil

Un manipulador móvil resulta de la unión de la plataforma móvil, el brazo serial y el efector final, por lo tanto, el concepto y el ensamble final realizado se muestra de la figura 32 a la 38.

3.8.1 Concepto

El concepto elegido para colocar el brazo serial sobre la plataforma móvil se muestra de la figura 32 a la 36. La justificación de esta elección se basa en colocar la base del brazo serial lo más cerca posible de la parte frontal de la plataforma para lograr un mayor alcance en la manipulación de objetos.

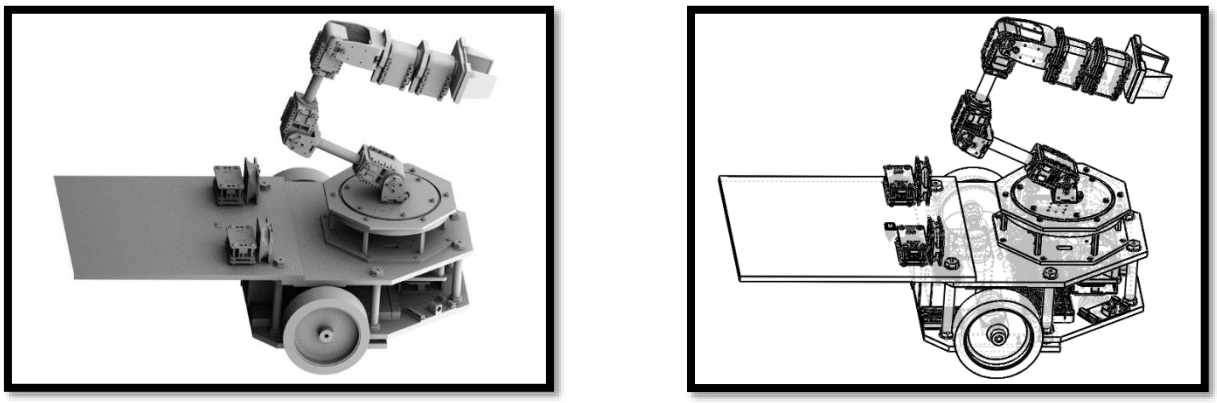


Figura 32.- Vista 1, concepto del manipulador móvil

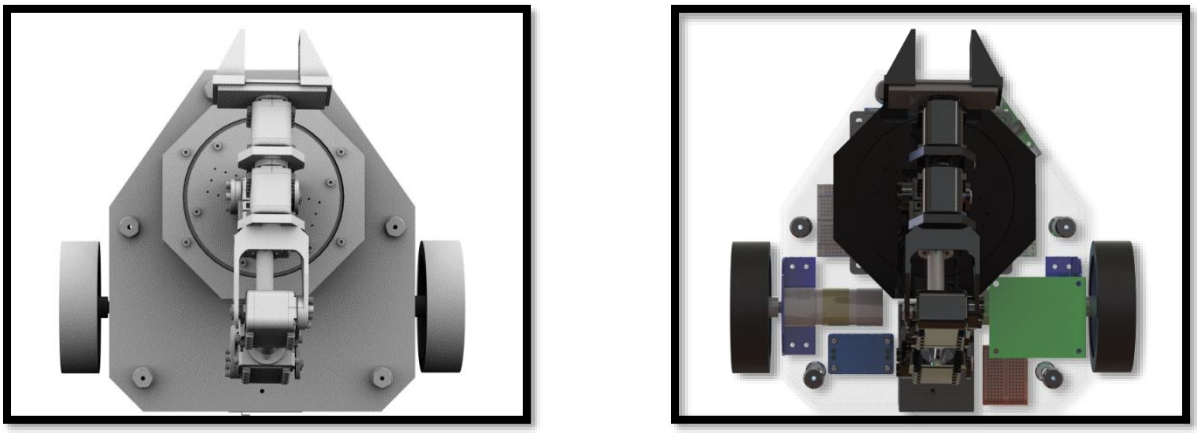


Figura 33.- Vista 2, concepto del manipulador móvil

Figura 34.- Vista 3, concepto del manipulador móvil

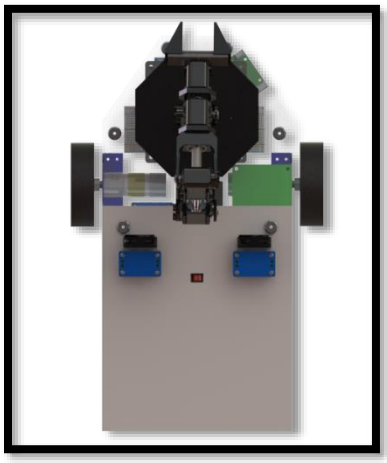


Figura 35.- Vista 4, concepto del manipulador móvil

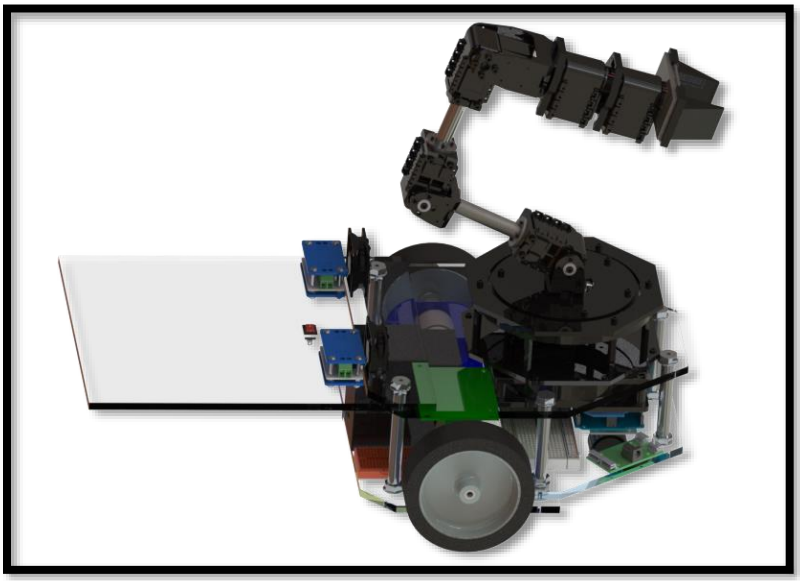


Figura 36.- Vista 5, concepto del manipulador móvil

3.8.2 Prototipo final

El prototipo final de manipulador móvil ensamblado por el autor de la presente tesis como banco de pruebas para el desarrollo de la tarea intuitiva, se muestra en las siguientes figuras:

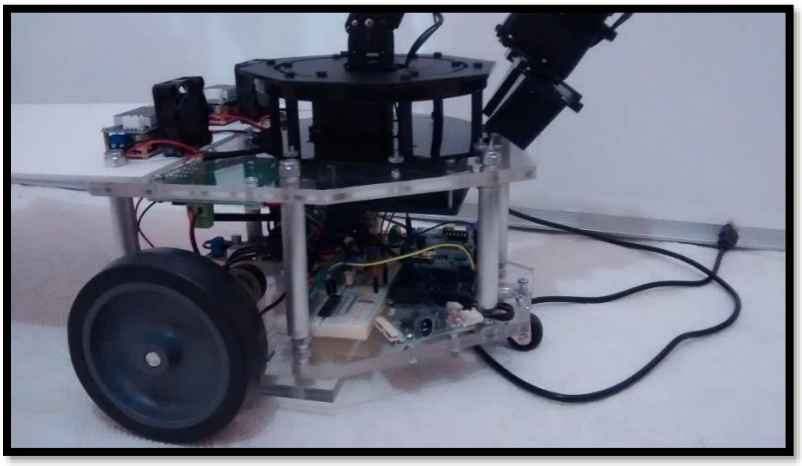


Figura 37.- Vista 1, versión final del manipulador móvil

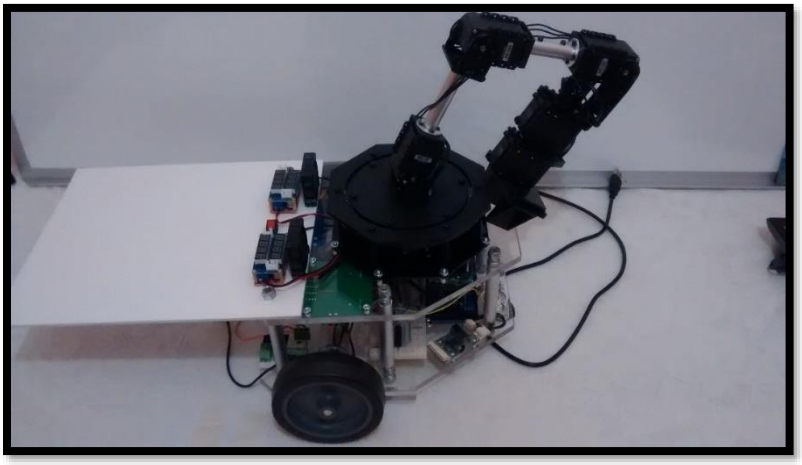


Figura 38.- Vista 2, versión final del manipulador móvil

3.9 Módulo electrónico

En este apartado se definen los principales elementos electrónicos utilizados para la correcta operación y funcionamiento del manipulador móvil.

3.9.1 Microcontrolador arduino mega 2560 R3

Arduino es una compañía de hardware libre, la cual desarrolla placas que integran un microcontrolador y un entorno de desarrollo (IDE), diseñado para facilitar el uso de la electrónica en proyectos multidisciplinarios. El hardware consiste en una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, y puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión (shields) que amplían las características de funcionamiento de la placa arduino. Por otro lado, el software consiste en un entorno de desarrollo (IDE) basado en el entorno de processing y lenguaje de programación basado en wiring, así como en el cargador de arranque (bootloader) que es ejecutado en la placa. El microcontrolador de la placa se programa a través de un computador, haciendo uso de comunicación serial mediante un convertidor de niveles RS-232 a TTL serial (Arduino, 2013).

La primera placa arduino fue introducida en el 2005, ofreciendo un bajo costo y facilidad de uso para novatos y profesionales buscando desarrollar proyectos interactivos con su entorno mediante actuadores y sensores. A partir de octubre del año 2012, se incorporaron nuevos modelos de placas de desarrollo que hacen uso de microcontroladores Cortex-M3, ARM de 32 bits, dichos modelos coexisten con los originales modelos que integran microcontroladores AVR de 8 bits. ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar y compilar bajo el IDE clásico de arduino sin ningún cambio. Las placas arduino están disponibles de forma ensambladas o en forma de kits "hazlo tú mismo" (por sus siglas en inglés “DIY”). Los esquemáticos de diseño del hardware están disponibles bajo licencia libre, permitiendo a cualquier persona crear su propia placa arduino sin necesidad de comprar una prefabricada. Adafruit Industries estimó a mediados del año 2011 que alrededor de 300,000 placas arduino habían sido producidas comercialmente, y en el año 2013 estimó que alrededor de 700,000 placas oficiales de la empresa arduino estaban en manos de los usuarios (Arduino, 2013). Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software tal como Adobe Flash, Processing, Max/MSP, Pure data. Una tendencia tecnológica es utilizar arduino como tarjeta de adquisición de datos desarrollando interfaces en software como JAVA, Visual Basic y LabVIEW 5. Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente (Arduino, 2013).

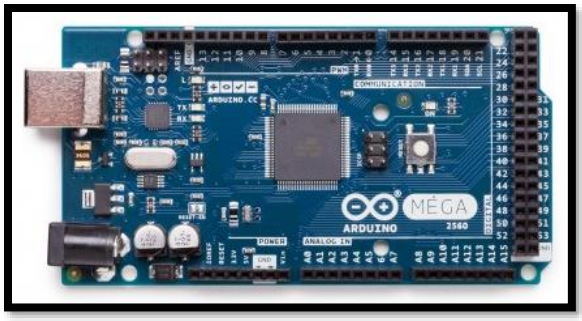


Figura 39.- Arduino mega 2560 R3, fuente: (Arduino Mega, 2011)

Las características generales del microcontrolador ATmega2560 son (Arduino Mega, 2011):

- Trabaja con un voltaje de operación de 5 Volts.
- Cuenta con 54 pines digitales de entrada y salida, de los cuales 15 son salidas pwm.
- Cuenta con 16 pines analógicos de entrada.
- En cada pin de entrada y salida proporciona una corriente DC de 40 mA.
- Cuenta con una memoria flash con capacidad de 256 KB (8 KB usados por el bootloader).
- SRAM: 8 KB.
- EEPROM: 4 KB.
- Clock speed: 16 MHz.
- Protocolos de comunicación: Serial, I2C.
- Interrupciones internas: 6.

De las características más importantes que tiene el microcontrolador arduino mega 2560 solo se utilizan los pines de comunicación serial, los de I2C y pwm. Los pines específicos que se utilizan son:

- Pin 19 (Rx) y 18 (Tx) para la comunicación serial
- Pin 2 (PWM)
- Pin SDA y SCL para la comunicación I2C
- Pin de 5 Volts
- Pin GND

El microcontrolador arduino mega se encarga de establecer la comunicación con los servomotores Dynamixel y la tarjeta MD25, se usa como una tarjeta de adquisición y procesamiento de datos, los cuales se obtendrán del programa Simulink de Matlab.

3.9.2 Conexión Arduino – Dynamixel AX-12A

Se cuenta con un antecedente en el MRG sobre el uso de este tipo de servomotores en la tesis de: (Castañeda Espinosa & Pedro Mendoza, 2016), en la cual se efectúa la comunicación con los servomotores Dynamixel de la serie RX cuya comunicación es RS-485/TTL, el diseño propuesto solo se realiza para la comunicación RS-485 por lo que se utiliza la metodología propuesta y se realiza el circuito para establecer la comunicación TTL.

Para establecer la comunicación entre arduino y los servomotores Dynamixel se emplea la librería que proporciona Josué Alejandro Savage (Savage, 2011). Esta librería es escrita de forma genérica para los servomotores Dynamixel, ya que se puede implementar en las series AX, MX, RX, etc.

Para implementar estas bibliotecas se necesita un buffer tri-estado, se puede hacer con dos circuitos integrados 74HC04 y 74HC126 tal como recomienda Bioloid y utiliza en sus controladores o puede usarse un solo circuito integrado, el 74LS241, en esta tesis se hace uso del circuito integrado 74LS241.

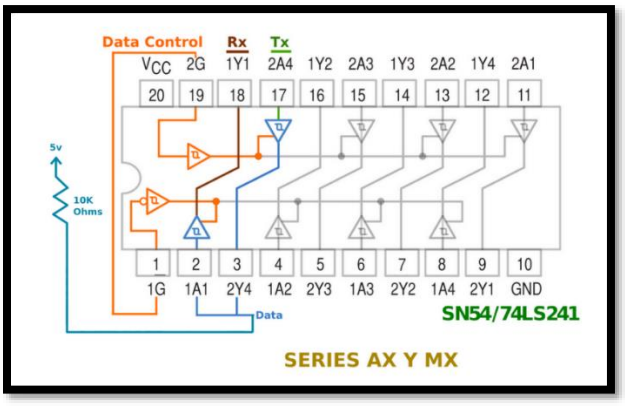


Figura 40.- Circuito integrado 74LS241, fuente: (Savage, 2011)

Los conectores de los servomotores Dynamixel cambian dependiendo el protocolo de comunicación, para la comunicación TTL el conector tiene 3 PINES (VIN, GND y Data).

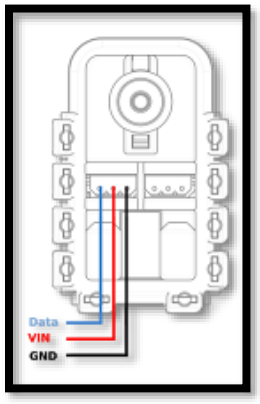


Figura 41.- Pines para la comunicación TTL, fuente: (Savage, 2011)

Los pines de comunicación cambian según sea la biblioteca que se quiera emplear, por lo cual, la librería a utilizar es “dynamixelserial1.h”. Las señales “data control”, RX y TX que se muestran en la figura 42 provienen de los pines 2 (data control), 18 (TX) y 19 (RX) del microcontrolador.

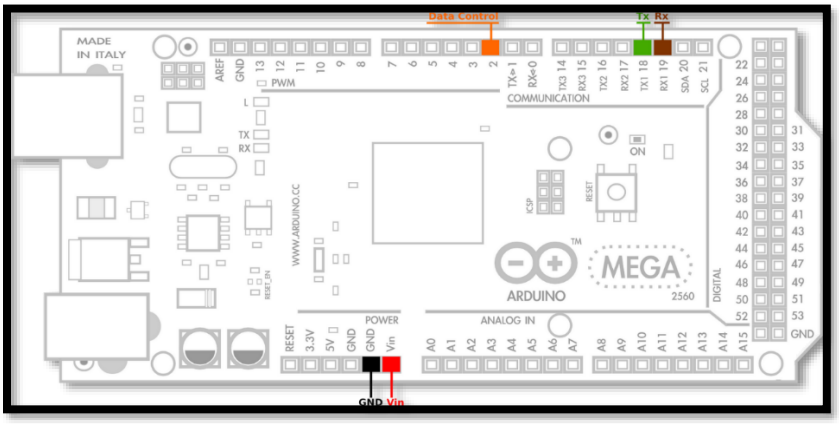


Figura 42.- Circuito Savage Electronics - Dynamixel Serial1, fuente: (Savage, 2011)

Finalmente, el circuito realizado para poder establecer la comunicación entre los servomotores Dynamixel serie AX-12A y la placa arduino es:

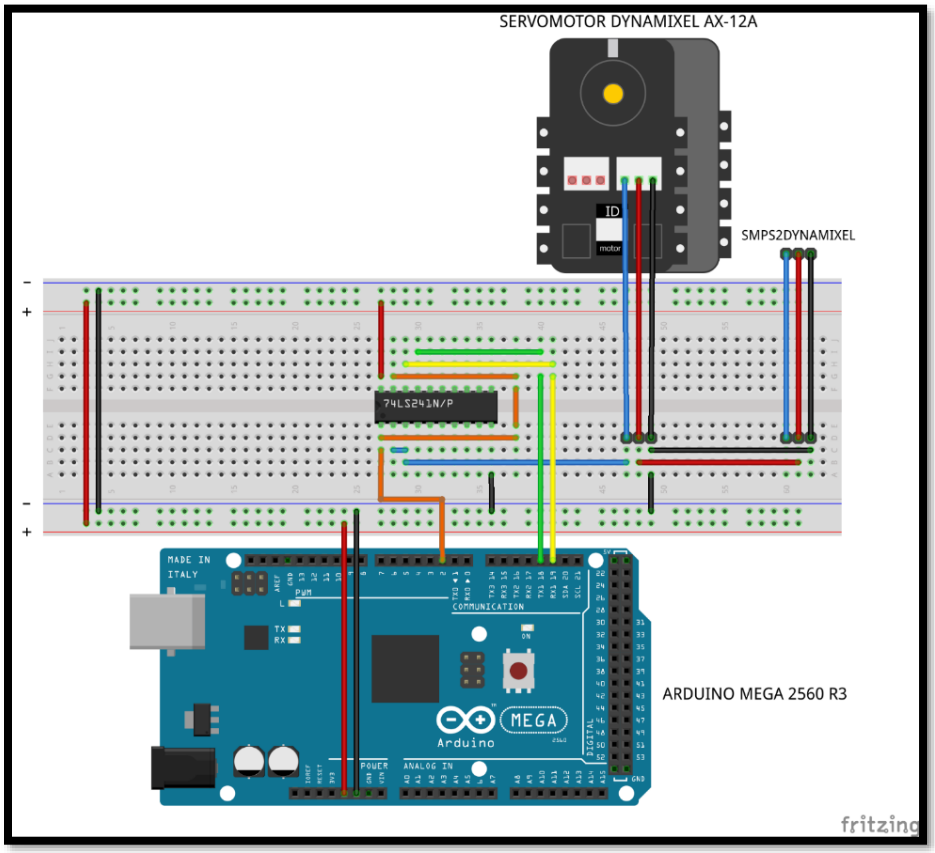


Figura 43.- Módulo electrónico "Dynamixel – Arduino mega"

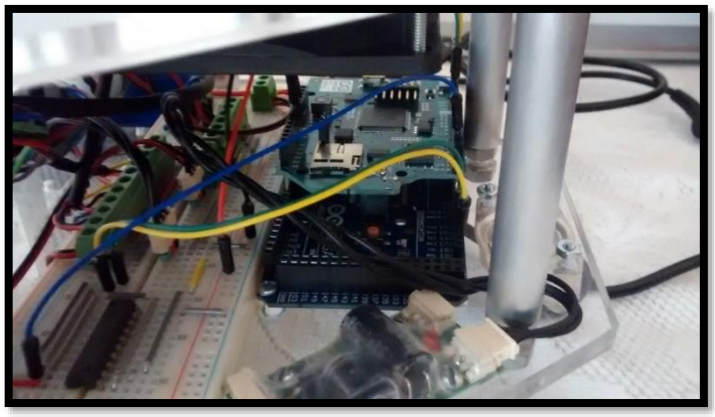


Figura 44.- Comunicación "Arduino-Dynamixel" (versión final)

Es importante señalar que la posición y velocidad proporcionada a cada servomotor se puede controlar con la biblioteca implementada, esto es de relevante importancia para llevar a cabo los experimentos, ya que permite realizar un adecuado "puente" con la programación hecha en Matlab/Simulink y la realizada en la placa arduino.

Se programa en Simulink el "Bloque Dynamixel", este se encarga de enviar los ID, posiciones y velocidades vía WiFi-UDP al microcontrolador arduino y recibir vía WiFi-UDP los valores de voltaje, temperatura, corriente, posición, velocidad y carga de cada servomotor, además, este bloque posee las conversiones necesarias para el correcto envío y recepción de datos vía WiFi, su programación se muestra en el apéndice C.

3.9.3 Conexión Arduino – MD25

La tarjeta MD25 es un controlador de doble puente H desarrollado para el control de dos motores de corriente continua, en particular para el modelo EMG30. Sus características principales son (Castañeda Espinosa & Pedro Mendoza, 2016):

- Lectura de encoders que indican el desplazamiento de los motores y sentido de giro.
- Maneja dos motores con control independiente o combinado.
- Lectura de la corriente que consume el motor.
- Se alimenta con un voltaje único de 12 volts de corriente directa.
- Regulador de +5 Vcc a 300 mA para alimentar la circuitería externa.
- Suministra hasta 2.8 A de corriente para cada motor.
- Interfaz I2C y serial con posibilidad de conectar hasta 8 controladores MD25 en el mismo bus.
- Control de potencia y aceleración.
- La resolución del encoder muestra 360 pulsos por vuelta.

Para realizar la comunicación entre el microcontrolador arduino y la tarjeta MD25 existen dos protocolos de comunicación: Serial e I2C. En la presente tesis se elige el protocolo I2C.

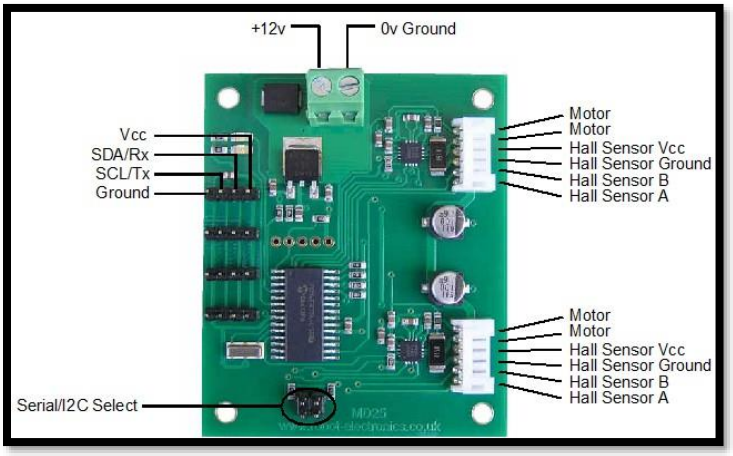


Figura 45.- Tarjeta MD25, fuente: (MD25 - Dual 12Volt 2.8Amp H Bridge Motor Drive, 2009)

Para controlar la velocidad de los motorreductores existen dos opciones diferentes a configurar, que depende de la aplicación. En el presente trabajo se controlan los motores de manera independiente, en donde los datos de velocidad van de 0 a 255. Cero indica la máxima velocidad en un sentido y 255 la máxima velocidad en el sentido contrario, por lo que la velocidad cero está determinado por el valor 128.

Se programa en Simulink el "Bloque MD25", este se encarga de enviar las velocidades de cada motorreductor a la tarjeta arduino vía WiFi-UDP que se obtienen en el bloque cinemático de la plataforma móvil. La regla de correspondencia de la velocidad angular a la velocidad programable en la tarjeta MD25 se muestra en el bloque "Cambio a MD25" (ver apéndice C). La gestión de la comunicación I2C en la tarjeta arduino se lleva a cabo con la librería "Wire.h".

CAPÍTULO 4.- CINEMÁTICA DEL MANIPULADOR MÓVIL

4.1 Introducción al capítulo

En robótica es muy importante conocer la ubicación en el espacio tridimensional y la orientación de los objetos que se verán involucrados en la tarea a realizar. Habitualmente se utiliza un sistema de referencia coordenado (X, Y, Z) global y suele auxiliarse de sistemas de referencia locales. A manera de definición la cinemática es la ciencia que estudia el movimiento de los cuerpos sin tomar en cuenta las causas de este (Carlos Ávila, 2015).

A lo largo del capítulo se explica la cinemática del manipulador móvil, la cual se trata por separado, primero se obtiene la cinemática directa e inversa del brazo serial y por último el análisis cinemático de la plataforma móvil, también se muestra la cinemática de la plataforma en coordenadas generalizadas y la técnica para acoplar el brazo serial y la plataforma móvil. La mayor parte del presente capítulo se basa en la nomenclatura y el análisis cinemático realizado en la tesis de (Escobedo Castillo, 2012), para profundizar en el tema se debe consultar dicha tesis y la bibliografía referenciada en ella. El análisis de la cinemática directa es realizado por el autor de la presente tesis. Ciertas dimensiones cambian, sobre todo las del brazo serial ensamblado y se ven reflejadas en la programación (ver apéndice C).

4.2 Cinemática directa del brazo serial

Un sistema robótico puede describirse completamente a partir del modelo cinemático de cada una de las articulaciones y eslabones que lo conforman. Este concepto se conoce como cinemática directa y su objetivo es describir la geometría y la ubicación espacial de cada uno de los eslabones que constituyen el sistema, buscando definir las características del último eslabón que ejecuta la tarea que se va a realizar. Con frecuencia, este último eslabón se denomina *end-effector* o actuador final (Pérez Cisneros, Cuevas Jiménez, & Zaldívar Navarro, 2015).

4.2.1 Convención Denavit-Hartenberg (DH)

La ubicación del sistema coordenado sobre una articulación para medir su movimiento puede acomodarse de múltiples formas, dando lugar a distintas interpretaciones y excepciones. Sin embargo, con el objetivo de eliminar cualquier indeterminación, la comunidad de robótica y mecatrónica ha adoptado algunos acuerdos fundamentales para definir la posición y orientación inicial para cada sistema coordenado, de forma que sea posible medir el movimiento de la articulación y, por tanto, del eslabón correspondiente. La convención Denavit Hartenberg (DH) permite representar la relación geométrica desde la base de un sistema hasta su último eslabón, considerando una cadena cinemática compuesta por articulaciones y eslabones. Matemáticamente dicha relación se expresa a través de la transformada homogénea ${}^{0}T_{e}$. Los pasos más importantes de la convención DH se discuten detalladamente a continuación (Pérez Cisneros, Cuevas Jiménez, & Zaldívar Navarro, 2015):

1. Enumere cada una de las articulaciones consecutivamente empezando desde el valor 0. Sea cuidadoso en determinar la naturaleza rotacional o prismática de cada articulación.
2. Coloque el eje coordenado z en cada sistema coordenado (uno por cada articulación), de forma que apunte a la misma dirección que el eje de rotación de la articulación o en la misma dirección del movimiento traslacional en el caso de un eslabón prismático.
3. Los siguientes pasos se aplican a cada una de las articulaciones comenzando por la base (nótese que la articulación al inicio del eslabón se etiqueta como $n-1$ y la articulación localizada al final como n):
   3.1 Ajuste la dirección del eje x_{n-1} para que apunte al origen del siguiente sistema coordenado O_n.
   3.2 Mida el largo del eslabón (parámetro a_n) desde el origen del sistema O_{n-1} hasta el origen del sistema O_n, por supuesto sobre la dirección establecida por el eje x, que es una línea recta.

3.3 Determine el ajuste entre eslabones consecutivos (parámetro d_n), que se mide desde la intersección de la dirección x_{n-1} con el eje z_n (punto final de la medición anterior) hasta el origen de dicho sistema coordenado O_n, en la dirección del eje z_{n-1}.

3.4 Frecuentemente, la dirección del eje z_{n-1} es diferente a la dirección del siguiente eje de actuación z_n, por lo que debe realizarse un ajuste a través de una rotación sobre el eje x_{n-1} que permita girar la dirección del eje z_{n-1} para alinearlo con la dirección del eje z_n (parámetro α_n).

4. Complete la tabla al finalizar la determinación de los parámetros para cada eslabón. Cada renglón representa una articulación y su eslabón correspondiente.

4.2.2 Convención Denavit-Hartenberg: una definición formal

La convención DH puede expresarse a través de una matriz homogénea ${}^{n-1}T_n$, que registra cada uno de los pasos de la convención entre los dos sistemas coordenados de la articulación $n-1$ y la articulación n. La descripción se hace con respecto del sistema coordenado base en cada eslabón, lo que explica que la composición de matrices sigue un orden de posmultiplicación, que resulta en:

$$ {}^{n-1}T_n = T_{rotación\_z}(\theta)\cdot T_{traslación\_x}(a)\cdot T_{traslación\_z}(d)\cdot T_{rotación}(\alpha) \quad \textbf{Ecuación 2} $$

Considerando que las matrices HT de traslación representan desplazamientos en ejes separados, es posible fusionar ambas traslaciones en una sola matriz HT, quedando:

$$ {}^{n-1}T_n = T_{rotación\_z}(\theta)\cdot T_{traslación\_xz}(a,d)\cdot T_{rotación}(\alpha) \quad \textbf{Ecuación 3} $$

Lo que de forma matricial se expresa como:

$$ {}^{n-1}T_n = \begin{pmatrix} \cos\theta & -sen\theta & 0 & 0 \\ sen\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -sen\alpha & 0 \\ 0 & sen\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$ **Ecuación 4**

Al multiplicar las tres matrices, se obtiene la fórmula general para la transformada entre dos articulaciones $n-1$ y n, representadas a través de la convención DH:

$$ {}^{n-1}T_n(\theta_n) = \begin{pmatrix} \cos\theta_n & -sen\theta_n \cos\alpha_n & sen\theta_n sen\alpha_n & a_n \cos\theta_n \\ sen\theta_n & \cos\theta_n \cos\alpha_n & -\cos\theta_n sen\alpha_n & a_n sen\theta_n \\ 0 & sen\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} $$ **Ecuación 5**

Es muy importante comentar algunas propiedades de esta matriz de transformación para sistemas basados en la convención DH.

- La convención DH utiliza una variable y tres parámetros: el ángulo θ de magnitud variable que registra la posición de la articulación, así como los parámetros a que representan el largo del eslabón, la distancia de ajuste d y el ángulo de ajuste α.
- Para el caso de eslabones rotacionales, el parámetro θ se convierte en la variable del sistema que registra la posición actual y cambiante del eslabón que maneja dicha articulación.
- Por otra parte, si se modela un eslabón prismático, el parámetro de ajuste d se torna en variable, ya que el desplazamiento del eslabón se mide en la dirección del eje z.

De esta forma, tres parámetros de la convención DH son constantes mientras que, dependiendo de la naturaleza de la articulación, el parámetro θ o el parámetro d se consideran como variables.

Con el objetivo de mantener claridad en la representación DH, una costumbre muy común es adoptar la variable q_n para referirse a la variable de cada eslabón o, mejor dicho, al parámetro DH que se asume como variable, sea el parámetro θ o el parámetro d. La variable q_n se denomina coordenada generalizada (dado que se puede referir a eslabones rotacionales o prismáticos), mientras que el vector que contiene el valor de todas las coordenadas generalizadas se denomina vector de estado o vector de coordenadas generalizadas (q).

Por definición de los eslabones prismáticos y rotacionales, cada uno representa un grado de libertad, por lo que el número de coordenadas generalizadas q_n es igual al número de grados de libertad en el sistema. Fragmento basado en el libro de (Pérez Cisneros, Cuevas Jiménez, & Zaldívar Navarro, 2015). En la siguiente tabla se muestran los parámetros Denavit-Hartenberg obtenidos en base a las características del brazo serial construido.

Tabla 2.- Parámetros Denavit-Hartenberg correspondientes al brazo serial

n	α_n[rad]	d_n [cm]	a_n [cm]	θ_n[rad]
1	$\alpha_1 = \frac{\pi}{2}$	$d_1 = 7.5$	0	θ_1
2	0	0	$a_2 = 13$	θ_2
3	0	0	$a_3 = 12.6$	θ_3
4	0	0	$a_4 = 18$	θ_4

La matriz de transformación homogénea basada en la notación Denavit-Hartenberg para obtener el punto final del efector es:

$$
{}^0T_4 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 = \begin{pmatrix} \cos(\theta_2+\theta_3+\theta_4)\cdot\cos(\theta_1) & -(\sin(\theta_2+\theta_3+\theta_4)\cdot\cos(\theta_1)) & \sin(\theta_1) & \cos(\theta_1)\cdot(a_3\cdot\cos(\theta_2+\theta_3)+a_2\cdot\cos(\theta_2)+a_4\cdot\cos(\theta_2+\theta_3+\theta_4)) \\ \cos(\theta_2+\theta_3+\theta_4)\cdot\sin(\theta_1) & \frac{\cos(\theta_1+\theta_2+\theta_3+\theta_4)}{2}-\frac{\cos(\theta_1-\theta_2-\theta_3-\theta_4)}{2} & -\cos(\theta_1) & \sin(\theta_1)\cdot(a_3\cdot\cos(\theta_2+\theta_3)+a_2\cdot\cos(\theta_2)+a_4\cdot\cos(\theta_2+\theta_3+\theta_4)) \\ \sin(\theta_2+\theta_3+\theta_4) & \cos(\theta_2+\theta_3+\theta_4) & 0 & d_1+a_3\cdot\sin(\theta_2+\theta_3)+a_2\cdot\sin(\theta_2)+a_4\cdot\sin(\theta_2+\theta_3+\theta_4) \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

Ecuación 6

Con fines computacionales, para obtener la cinemática directa del brazo serial se realizan las simulaciones en Matlab con el Robotic Toolbox de Peter Corke.

```
Brazo serial (4 axis, RRRR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|        7.5|          0|     1.571|          0|
|  2|         q2|          0|         13|          0|          0|
|  3|         q3|          0|       12.6|          0|          0|
|  4|         q4|          0|         18|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =    0  base = 1  0  0  0   tool =  1  0  0  0
          0          0  1  0  0           0  1  0  0
       9.81          0  0  1  0           0  0  1  0
                     0  0  0  1           0  0  0  1
```

Figura 46.- Parámetros programados en Matlab

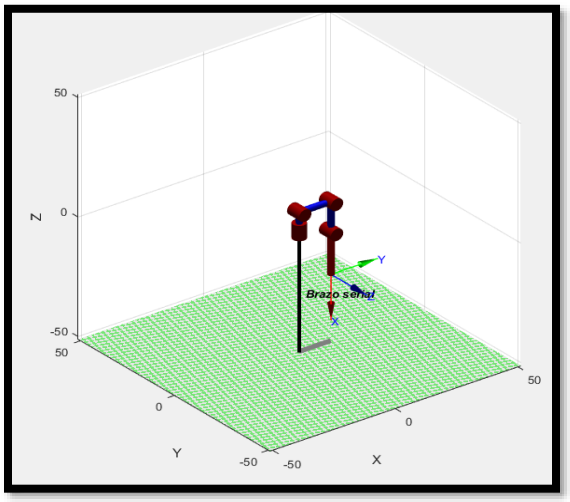


Figura 47.- Simulación en Matlab de la cinemática directa

Finalmente se programa la cinemática directa en un bloque de Simulink con el fin de obtener los cálculos en tiempo real. Es importante señalar que se desacopla el brazo serial para realizar los cálculos de cinemática directa, la explicación de este método se encuentra en (Escobedo Castillo, 2012). De esta manera la penúltima junta se utiliza para orientar al efector final y la última junta controla la longitud de apertura y cierre del gripper o efector final. Más adelante se explica cómo se obtiene la orientación de la penúltima junta.

4.3 Cinemática inversa

El cálculo de la cinemática inversa se refiere al conjunto de ángulos necesarios para hacer que el efector final se mueva de una posición y orientación inicial a una orientación y posición final deseada. Este tipo de cálculos lo hacemos de manera inconsciente al mover nuestra mano de un lado a otro. Pero en el caso de los robots seriales es necesario la realización de algoritmos computacionales para poder controlarlos, aunque este no se vuelve el único medio para controlar un brazo serial, debido a que también se puede controlar de manera manual eligiendo los ángulos que se deseen en cada articulación. El cálculo de la cinemática inversa suele complicarse, debido a la existencia de múltiples soluciones y también a las ecuaciones cinemáticas debido a que no son lineales (Carlos Ávila, 2015). Existen varias formas de resolver el problema cinemático inverso, de estos métodos, se utiliza en este trabajo un método geométrico descrito por (Fu, González, & Lee, 1989) y extraído de la tesis de (Escobedo Castillo, 2012), este método tiene la ventaja de encontrar los valores de las juntas en línea, permitiendo que cualquier alteración en la posición del efector final del manipulador se pueda alcanzar. Esto se debe a que no requiere gran procesamiento para calcular las variables requeridas.

Utilizando este método, se tiene principalmente dos soluciones para este problema, estas son: codo arriba y codo abajo, para este trabajo, se utiliza la configuración codo arriba. Esta elección es de acuerdo con la tarea que va a desempeñar, permitiendo que los objetos encontrados debajo del origen del manipulador puedan ser alcanzados. Lo anterior se debe a que los servomotores con los que trabaja poseen un rango de movimiento limitado (Escobedo Castillo, 2012). En (Escobedo Castillo, 2012) se explica detalladamente el método que lleva a obtener las ecuaciones que se utilizan para programar la cinemática inversa del brazo serial, estas son:

- Para la primera junta

$$\theta_1 = \text{atan2}\left(\frac{x_p}{y_p}\right) \quad \textbf{Ecuación 7}$$

- Para la segunda junta

$$z_c = z_p + l_4 \sin\phi$$ **Ecuación 8**

$$r = \sqrt{x_p{}^2 + y_p{}^2}$$ **Ecuación 9**

$$x_c = r - l_4 \cos\phi$$ **Ecuación 10**

En las ecuaciones se encuentra un parámetro ϕ, este parámetro corresponde a la orientación del objeto a manipular, y es el ángulo respecto a la horizontal con el eslabón cuatro, de longitud l_4. Este ángulo dependerá de cómo se quiera manipular el objeto, es decir, en qué ángulo respecto a la horizontal será tomado el mismo.

$$R = x_c{}^2 + (z_c - l_1)^2$$ **Ecuación 11**

$$\cos\beta = \frac{R^2 + l_2{}^2 - l_3{}^2}{2 l_2 R}$$ **Ecuación 12**

$$\sin\beta = \sqrt{1 - \cos^2\beta}$$ **Ecuación 13**

$$\theta_2 = \text{atan2}\left(\left(\frac{z_c - l_1}{R}\right)\left(\frac{R^2 + l_2{}^2 - l_3{}^2}{2 l_2 R}\right) + \left(\frac{x_c}{R}\right)\left(\sqrt{1 - \cos^2\beta}\right), -\left(\left(\frac{x_c}{R}\right)\left(\frac{R^2 + l_2{}^2 - l_3{}^2}{2 l_2 R}\right) - \left(\frac{z_c - l_1}{R}\right)\left(\sqrt{1 - \cos^2\beta}\right)\right)\right)$$

Ecuación 14

- Para la tercera junta

$$R^2 = l_3{}^2 + l_2{}^2 - 2 l_2 l_3 \cos\gamma$$ **Ecuación 15**

$$\cos\gamma = \frac{l_3^{\,2} + l_2^{\,2} - R^2}{2 l_2 l_3}$$ **Ecuación 16**

$$\sin\gamma = \sqrt{1-\cos^2\gamma}$$ **Ecuación 17**

$$\theta_3 = \operatorname{atan2}\left(\sqrt{1-\cos^2\gamma}, -\frac{l_3^{\,2} + l_2^{\,2} - R^2}{2 l_2 l_3}\right)$$ **Ecuación 18**

- Para la cuarta junta

$$\theta_4 = \operatorname{atan2}\begin{pmatrix} \sin\phi\left(\cos\gamma\cos\theta_2 + \sin\gamma\sin\theta_2\right) + \cos\phi\left(\sin\gamma\cos\theta_2 - \cos\gamma\sin\theta_2\right), \\ \cos\phi\left(\cos\gamma\cos\theta_2 + \sin\gamma\sin\theta_2\right) - \sin\phi\left(\sin\gamma\cos\theta_2 - \cos\gamma\sin\theta_2\right) \end{pmatrix}$$

Ecuación 19

- Para la quinta junta

La posición angular de la penúltima junta del brazo serial debe coincidir con el ángulo de orientación del objeto o el destino, según sea el caso. En la figura 48 se muestra la solución propuesta, para obtener el ángulo que permite al gripper coincidir con la orientación del objeto.

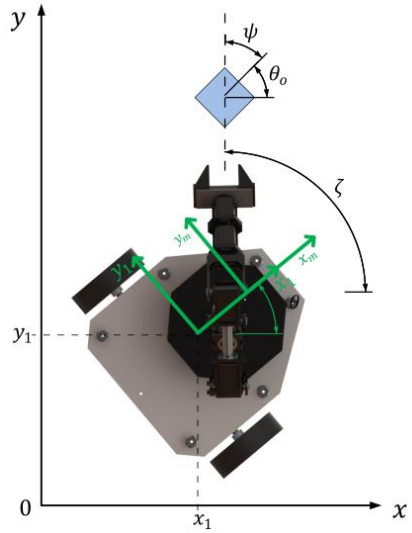


Figura 48.- Ángulo de la penúltima articulación del brazo serial

De la figura 48 se encuentra que el ángulo ψ está dado por la expresión:

$$\psi = \zeta - \theta_o$$

Ecuación 20

Donde θ_o, representa el ángulo de orientación del objeto a manipular y ζ, representa el ángulo entre el origen del manipulador y la posición del objeto a manipular en coordenadas globales.

$$\zeta = \text{atan}\frac{y_o - y_{origen_m}}{x_o - x_{origen_m}}$$

Ecuación 21

El punto (x_o, y_o), se refiere a las componentes del objeto a manipular, las coordenadas del origen del brazo serial en el marco global, se encuentran mediante la siguiente expresión:

$$x_{origen_m} = x_1 + x_0 \cos\theta$$
$$y_{origen_m} = y_1 + x_0 \sin\theta$$

Ecuación 22

- Para la sexta junta

Esta junta viene determinada por la longitud del objeto que se desea manipular, los límites de apertura y cierre del efector final o gripper son:

- Límite de apertura = 3.2 centímetros
- Límite de cierre = 0.2 centímetros

Estas ecuaciones son de vital importancia para el desarrollo de los experimentos, ya que proporcionan los ángulos necesarios para que el efector final describa el lugar geométrico de ida y regreso, en la tarea de sujetar y colocar al objeto (pick and place) en el punto deseado, por lo tanto, para obtener eficacia y eficiencia en el código programado, se emplea una S-Function de Matlab programada en lenguaje C, dicha programación se muestra en el apéndice C, en el apartado "Bloque cinemática inversa". Se hace notar que este tipo de programación es un esfuerzo del autor de la presente tesis para lograr mejorar la respuesta del sistema en tiempo real, además de servir como plantilla para futuros experimentos.

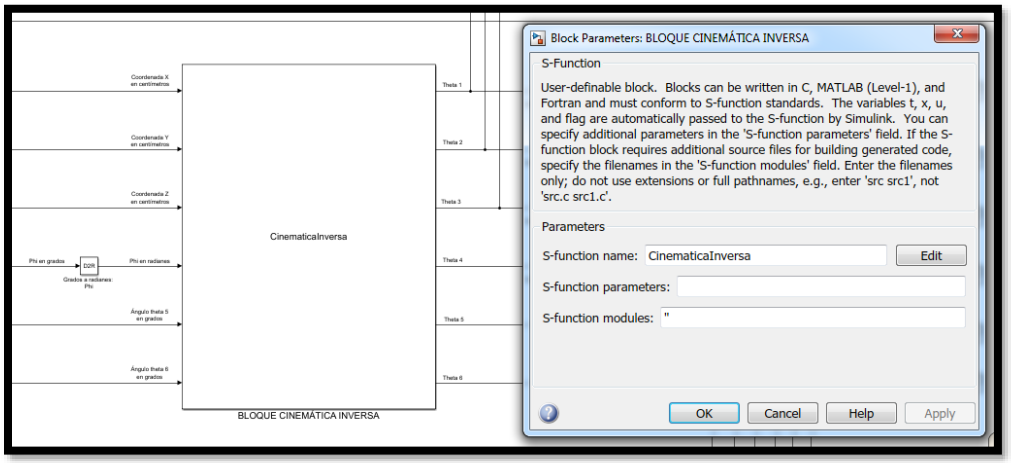


Figura 49.- Cinemática inversa S-Function Matlab

4.4 Plataforma del robot móvil

El manipulador móvil elegido posee una plataforma móvil en configuración diferencial, bajo las consideraciones de que es un cuerpo rígido no deformable, las llantas no presentan deslizamientos y su eje de rotación es paralelo al plano de trabajo. El modelado de la plataforma móvil posee dos sistemas coordenados, un sistema inercial fijado en la plataforma móvil y un sistema de referencia absoluto, en este caso definido por el sistema de visión (Carlos Ávila, 2015). La ubicación de la plataforma móvil dentro del sistema de referencia inercial es descrita por el vector:

$$\xi = \begin{bmatrix} x & y & \theta \end{bmatrix}^T \quad \textbf{Ecuación 23}$$

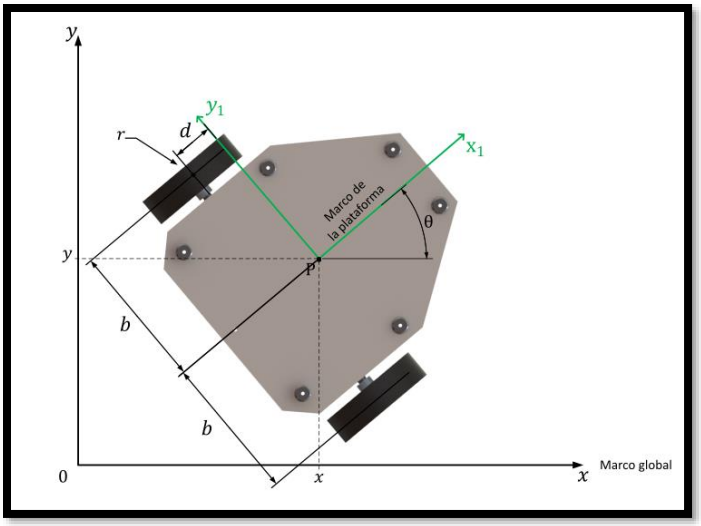


Figura 50.- Sistemas de referencia para la plataforma móvil

La matriz de rotación que describe la ubicación angular del sistema inercial dentro del sistema de referencia absoluto:

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \textbf{Ecuación 24}$$

La velocidad de la plataforma móvil dentro del sistema inercial de la plataforma móvil es descrita por el vector:

$$\xi_1 = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^T \quad \textbf{Ecuación 25}$$

El siguiente vector representa la velocidad del sistema inercial dentro del sistema absoluto de referencia. Es necesario señalar que la velocidad angular del sistema inercial $\dot{\theta}$ se expresa solo dentro del sistema inercial.

$$\dot{\xi} = R^T(\theta)\dot{\xi}_1 \quad \textbf{Ecuación 26}$$

Por propagación de velocidades se calcula la velocidad de las llantas, la ubicación de las llantas dentro del sistema inercial tiene por coordenadas $(\pm d, \pm b)$, según sea el caso. Propagación de velocidades establece que la velocidad angular del eslabón $i+1$, es la velocidad angular del eslabón i más la adición de la componente que agregue el eslabón $i+1$. Matemáticamente:

$$^{i+1}\omega = {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}$$ **Ecuación 27**

En el caso de la velocidad lineal en el eslabón $i+1$ será la misma del eslabón i más la componente de la velocidad angular del eslabón en i. Matemáticamente:

$$^{i}v_{i+1} = {}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}$$ **Ecuación 28**

Tomando en cuenta propagación de velocidades para calcular el vector ξ_1. Y considerando la ubicación de las llantas descritas, se tiene que:

$$^{1}\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}$$ **Ecuación 29**

Para el caso de la llanta derecha:

$$^{1}P_{2D} = \begin{bmatrix} -d \\ -b \\ 0 \end{bmatrix}$$ **Ecuación 30**

Para el caso de la llanta izquierda:

$$^{1}P_{2I} = \begin{bmatrix} -d \\ b \\ 0 \end{bmatrix}$$ **Ecuación 31**

Propagando las velocidades a cada extremo donde se encuentran las llantas, en el presente caso es del sistema inercial a la llanta derecha y del sistema inercial a la llanta izquierda.

La velocidad del sistema inercial:

$$ {}^{1}v_{1} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x}\cos\theta + \dot{y}\sin\theta \\ \dot{y}\cos\theta - \dot{x}\sin\theta \\ 0 \end{bmatrix} \quad \textbf{Ecuación 32} $$

Fórmula para aplicar:

$$ {}^{i}v_{i+1} = {}^{i}v_{i} + {}^{i}\omega_{i} \times {}^{i}P_{i+1} \quad \textbf{Ecuación 33} $$

Aplicando la fórmula en la rueda derecha:

$$ {}^{2}v_{2D} = \begin{bmatrix} \dot{x}\cos\theta + \dot{y}\sin\theta \\ \dot{y}\cos\theta - \dot{x}\sin\theta \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \times \begin{bmatrix} -d \\ -b \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{x}\cos\theta + \dot{y}\sin\theta + b\dot{\theta} \\ \dot{y}\cos\theta - \dot{x}\sin\theta - d\dot{\theta} \\ 0 \end{bmatrix} \quad \textbf{Ecuación 34} $$

Aplicando la fórmula en la rueda izquierda:

$$ {}^{2}v_{2I} = \begin{bmatrix} \dot{x}\cos\theta + \dot{y}\sin\theta \\ \dot{y}\cos\theta - \dot{x}\sin\theta \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \times \begin{bmatrix} -d \\ b \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{x}\cos\theta + \dot{y}\sin\theta - b\dot{\theta} \\ \dot{y}\cos\theta - \dot{x}\sin\theta - d\dot{\theta} \\ 0 \end{bmatrix} \quad \textbf{Ecuación 35} $$

La velocidad lineal de una rueda se puede obtener en base a su velocidad angular $\left[\phi_{d}\right]$ y a su radio [r]. Para la rueda derecha:

$$ \begin{bmatrix} r\phi_{d} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{x}\cos\theta + \dot{y}\sin\theta + b\dot{\theta} \\ \dot{y}\cos\theta - \dot{x}\sin\theta - d\dot{\theta} \\ 0 \end{bmatrix} \quad \textbf{Ecuación 36} $$

Para la rueda izquierda:

$$\begin{bmatrix} r\phi_i \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{x}\cos\theta + \dot{y}\sin\theta - b\dot{\theta} \\ \dot{y}\cos\theta - \dot{x}\sin\theta - d\dot{\theta} \\ 0 \end{bmatrix}$$ **Ecuación 37**

La plataforma móvil que se tiene es una plataforma no holonómica debido a que puede tener desplazamientos sobre su eje de simetría y también puede rotar, pero la configuración de ruedas impide que tenga un desplazamiento lateral. Para poder realizar un movimiento lateral primero tendría que rotar la plataforma para poder después desplazarse en forma recta. De esta restricción cinemática se establece la ecuación:

$$0 = \dot{y}\cos\theta - \dot{x}\sin\theta - d\dot{\theta}$$ **Ecuación 38**

Debido a la restricción cinemática por propagación de velocidades solo nos queda la componente en x de ambas ruedas.

- Para la rueda derecha:

$$r\phi_d = \dot{x}\cos\theta + \dot{y}\sin\theta + b\dot{\theta}$$ **Ecuación 39**

- Para la rueda izquierda:

$$r\phi_i = \dot{x}\cos\theta + \dot{y}\sin\theta - b\dot{\theta}$$ **Ecuación 40**

Texto basado en la tesis de (Carlos Ávila, 2015).

4.5 Coordenadas generalizadas

Las coordenadas generalizadas de la plataforma móvil están dadas por (Escobedo Castillo, 2012):

- Las coordenadas de la postura, que corresponde a x y y que definen la posición del punto P y la orientación θ de la plataforma.
- Las coordenadas de rotación de las llantas a lo largo de su eje horizontal de cada llanta.

El vector de coordenadas generalizadas para la plataforma está determinado por:

$$q = [x \quad y \quad \theta \quad \phi_d \quad \phi_i]^T$$

Ecuación 41

El vector de la postura en velocidad es el siguiente:

$$q = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} & \dot{\phi}_d & \dot{\phi}_i \end{bmatrix}^T$$

Ecuación 42

Una vez determinadas cuales son las coordenadas generalizadas, hay que encontrar una expresión en espacio de estados para la cinemática, el resultado de expresar en variables de estado se muestra en la siguiente ecuación (Escobedo Castillo, 2012):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi}_d \\ \dot{\phi}_i \end{bmatrix} = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \\ 0 & 1 \\ 1/r & b/r \\ 1/r & -b/r \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Ecuación 43

4.6 Acoplamiento

Tomando en cuenta el marco coordenado que se encuentra en el centroide de la plataforma (x_1, y_1), en donde x_1 está dirigido hacia delante de la plataforma, se escoge un punto P_r referenciado al marco coordenado $x_1 - y_1$ como punto de referencia. Por lo que, para la plataforma móvil, sus movimientos se tienen que realizar de manera que este punto de referencia siga una trayectoria deseada. El punto de referencia se denota como $({}^1x_r, {}^1y_r)$, que está referenciado al marco $x_1 - y_1$. Por lo tanto, las coordenadas globales (x_r, y_r) del punto de referencia se encuentra mediante (Escobedo Castillo, 2012):

$$x_r = x_1 + {}^1x_r \cos\theta - {}^1y_r \sin\theta$$
$$y_r = y_1 + {}^1x_r \sin\phi + {}^1y_r \cos\theta$$

Ecuación 44

Donde x_1 y y_1, son las coordenadas globales del punto P. La selección del punto de referencia es el acoplamiento de ambas partes. Esto lleva a que el punto de referencia mencionado es el punto de operación del efector final del brazo (Escobedo Castillo, 2012).

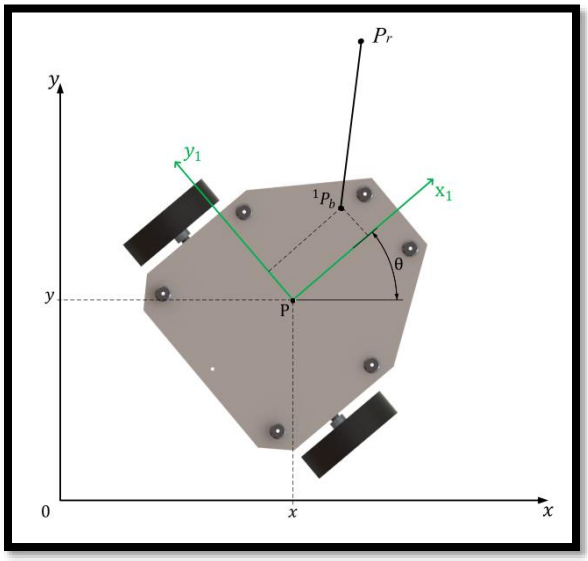


Figura 51.- Acoplamiento de la plataforma móvil y el brazo serial

CAPÍTULO 5.- PROGRAMACIÓN DE LA INTUICIÓN ARTIFICIAL EN LA TAREA DE TRASLADO DE UN MANIPULADOR MÓVIL

5.1 Introducción al capítulo

A lo largo de este capítulo se muestra cómo se aplica la intuición artificial en un manipulador móvil para describir el lugar geométrico que recorre el efector final en la tarea conocida como "pick and place", dicha tarea está implícita en la tarea local y global descritas en el capítulo tres y consiste en trasladar un objeto de una posición y orientación inicial a una posición y orientación final. La intuición artificial se programa en la coordinación del brazo serial montado sobre la plataforma móvil para generar el recorrido o lugar geométrico que este debe seguir con el efector final, el punto inicial está dado por la configuración preferida elegida para el brazo o estado de reposo y el final está descrito por la posición del objeto o la meta, según sea el caso.

Para el movimiento del brazo serial en el espacio cartesiano o global, se requerirá conocer el punto inicial y final del gripper (efector final), mientras que en el espacio articular se encontrará el movimiento de cada junta (articulación) mediante cinemática inversa para alcanzar los puntos que debe recorrer el efector final del lugar geométrico generado.

Las fórmulas programadas son las obtenidas por (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014), en dicha tesis se deducen dos fórmulas, una para el traslado de objetos y la otra para evasión de obstáculos, solo se utiliza la primera.

5.2 Intuición humana en el traslado

Traslado. En un espacio de tres dimensiones están contenidos dos puntos que definen a la tarea de traslado, un punto inicial y un punto final. Una persona puede intuir la forma que tendrá el traslado entre ambos puntos, no es un trayecto estricto, pero con una forma definida. Las observaciones con personas muestran que estos trayectos están contenidos dentro de un "volumen" como se muestra en la figura 52 (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014):

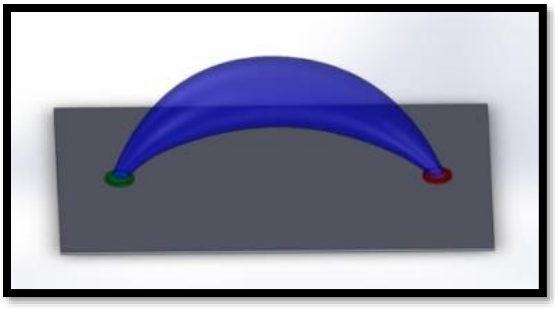


Figura 52.- Espacio intuitivo de la tarea de traslado, fuente: (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014)

El volumen sintetizado por la acción intuitiva durante una tarea de traslado forma un espacio intuitivo de la tarea, está incluido en el espacio de trabajo del instrumento de medición y además es susceptible de ser emulado por un sistema robótico. Una forma de síntesis de un lugar geométrico para establecer un modelo matemático del espacio intuitivo de la tarea puede ser a través del método de centroides que simplifica tal espacio en un lugar geométrico. El método de centroides se basa en la obtención de los centroides de los datos que están contenidos dentro de rebanadas espaciadas uniformemente, los datos son los puntos en función del tiempo generados por las trayectorias intuitivas (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).

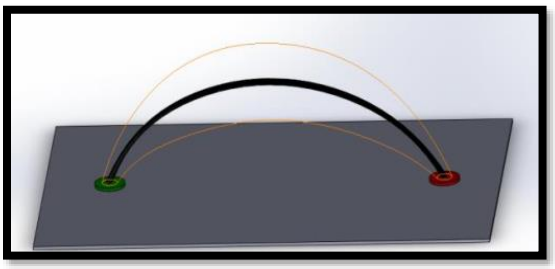


Figura 53.- Representación del lugar geométrico (negro) obtenido a partir de los datos del espacio intuitivo de la tarea de traslado, fuente: (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014)

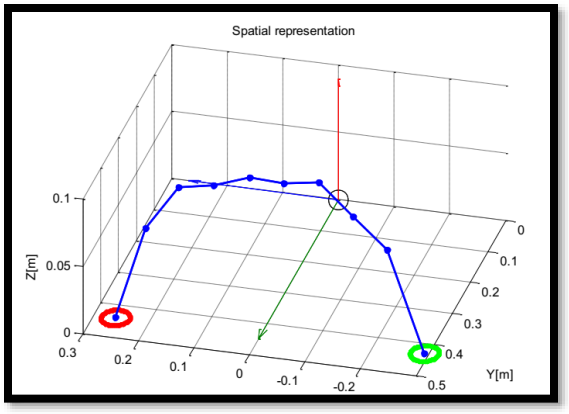


Figura 54.- Promedios del muestreo de traslado, fuente: (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014)

En las siguientes figuras se observan los comportamientos en cada eje de los datos mostrados en la figura 54, siendo primero el eje x donde se describe una recta con pendiente constante.

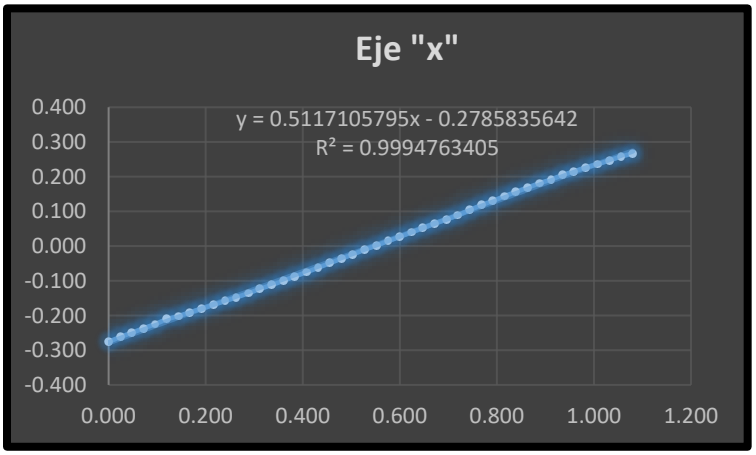


Figura 55.- Eje x

En la figura 56 se observa una recta que permanece cercana a un valor constante en el eje, esto es, el valor inicial es prácticamente el valor final.

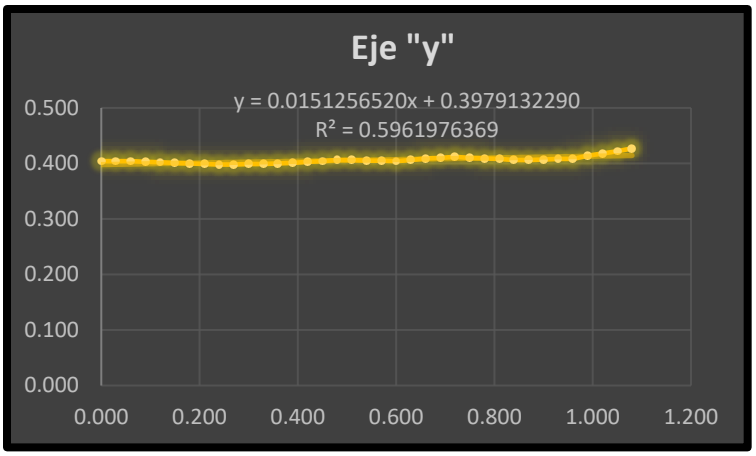


Figura 56.- Eje y

En la figura 57 se observa el comportamiento de un "arco" para el eje z.

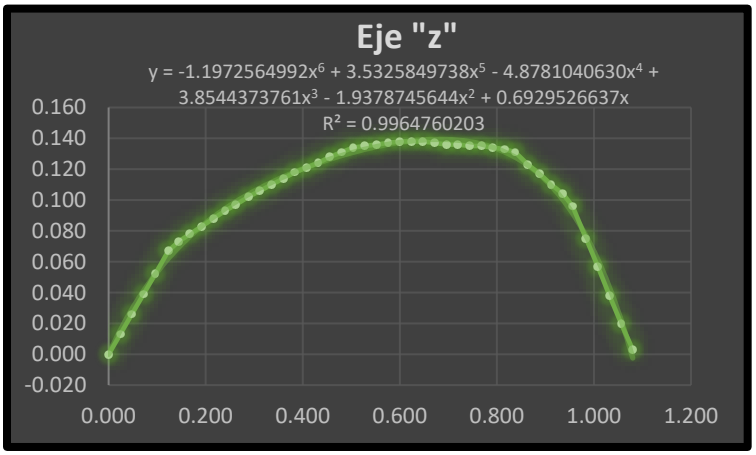


Figura 57.- Eje z

5.3 Ecuaciones para el algoritmo de traslado

Teniendo un punto inicial $I(x_i, y_i, z_i)$ y un punto final $F(x_f, y_f, z_f)$, se utilizan las siguientes ecuaciones extraídas de (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014) que dependen de un avance (OA), dicho avance se discretiza en 512 puntos o posiciones para la etapa de "ida" y 512 para la etapa de "regreso".

Las fórmulas que se utilizan para generar el lugar geométrico que recorre el efector final del brazo serial montado sobre la plataforma móvil, son:

$$x_{ia} = OA \cdot \left(x_f - x_i \right) + x_i \quad \textbf{Ecuación 45}$$

$$y_{ia} = OA \cdot \left(y_f - y_i \right) + y_i \quad \textbf{Ecuación 46}$$

$$z_{ia} = OA \cdot \left(z_f - z_i \right) + z_i + Amplitud \cdot Seno\left(OA \cdot \pi \right) \quad \textbf{Ecuación 47}$$

Donde:

$$Amplitud = \frac{1}{4}\sqrt{\left(x_f - x_i \right)^2 + \left(y_f - y_i \right)^2 + \left(z_f - z_i \right)^2} \quad \textbf{Ecuación 48}$$

$x_{ia} = x$ intuición artificial
$y_{ia} = y$ intuición artificial
$z_{ia} = z$ intuición artificial

Estas fórmulas se programan en el bloque “Bloque para el control del brazo serial” que se encuentra en el apéndice C, las capturas a detalle del código programado se muestran en el apéndice A, así como una breve explicación.

5.4 “Pick and place”

La tarea “pick and place” que ejecuta el brazo serial se compone de cuatro etapas (movimientos), estas son:

- *Etapa pick:* Este movimiento se realiza cuando el brazo montado sobre la plataforma móvil tiene que llegar al objeto que se desea manipular desde su configuración inicial (configuración preferida).
- *Etapa pick-back:* Este movimiento se ejecuta una vez que se ha tomado al objeto, de tal forma que se tiene que seguir el mismo lugar geométrico descrito en la etapa “pick” pero en sentido contrario para colocarse en su configuración inicial.

- *Etapa place:* Este movimiento se lleva a cabo cuando el brazo se encuentra en su configuración inicial y con el efector final sujetando al objeto debe situarlo en su destino y colocarlo.
- *Etapa return:* Finalmente este movimiento se lleva a cabo después de que el efector final ha soltado el objeto en su destino para colocarse en su configuración inicial.

No es necesario planear el movimiento de ida y regreso, ya que es generado automáticamente por las ecuaciones 45, 46 y 47 que representan al espacio intuitivo o lugar geométrico, siendo solo necesario el punto inicial $I(x_i, y_i, z_i)$ y el punto final $F(x_f, y_f, z_f)$.

Las etapas "*pick"* y "*pick back"* (tarea local) describen la tarea de sujetar al objeto, mientras que para depositar el objeto en la meta se ejecutan las etapas *"place"* y *"return"* (tarea local). Para efectuar la tarea global se ejecutan todas las etapas en orden *"pick – pick back – place - return"*.

La variación en la posición angular de la articulación θ_1 a la θ_4 para alcanzar cada punto del lugar geométrico son dadas por la cinemática inversa en cada etapa y en tiempo real, además de ser suficientes para alcanzar el punto deseado con el efector final, siempre y cuando este se encuentre dentro del espacio de trabajo alcanzable del brazo. La articulación θ_5 se utiliza para orientar al efector final como se explica en el capítulo cuatro y la junta θ_6 controla la longitud de apertura y cierre del gripper o efector final.

Por las características de la tarea a realizar y el sistema con el que obtiene la ubicación de los componentes (fiducials), las coordenadas se encuentran referenciadas a un marco global, esto quiere decir, que las coordenadas globales del objeto se tienen que transformar a coordenadas del marco origen del manipulador. Para que el manipulador pueda realizar la operación "pick and place" del objeto (Escobedo Castillo, 2012).

$$x_m = (x_r - x_1)\cos\theta + (y_r - y_1) - x_o$$
$$y_m = (y_r - y_1)\cos\theta - (x_r - x_1)\sin\theta$$

Ecuación 49

Con esta expresión se encuentran las componentes (*x*, *y*) del punto final para realizar la operación pick and place, solo falta la componente *z* del punto en cuestión, debido a que el modelo de la plataforma móvil aplica cuando el sistema se encuentra en el plano *x* – *y*, la componente *z* es la misma y no hace falta convertir (Escobedo Castillo, 2012).

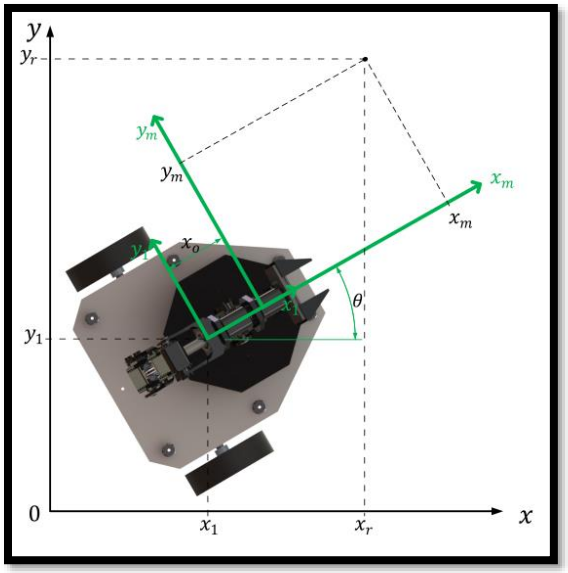


Figura 58.- Transformación de coordenadas globales a coordenadas del manipulador

En la ecuación 49 se encuentra el término x_0, el cual se define por la figura 58 como la distancia entre el origen del marco (x_1, y_1) y el marco (x_m, y_m) en la componente x. Debido a que no hay componente en y no aparece un término que lo represente (Escobedo Castillo, 2012). Por las características del manipulador móvil construido en la presente tesis la longitud tiene como valor numérico:

$$x_o = 1.8\ cm$$

Si se requiere ver la programación completa ir al apéndice C y consultar el apartado “Bloque para la conversión a coordenadas del manipulador”.

5.5 Diagrama de flujo de la programación

El diagrama de flujo que se observa en la figura 59 muestra la forma en cómo se programó la secuencia de instrucciones para ejecutar cada etapa de la tarea “pick and place” en el bloque para el control del brazo serial. La ejecución del bloque se rige por el algoritmo de coordinación mostrado en el capítulo 6. También se observa cómo interactúa la ecuación 47 en la programación para generar el lugar geométrico o espacio intuitivo de traslado, la ecuación 45 y 46 están implícitas dentro de la rutina de ida y de regreso, así como la discretización.

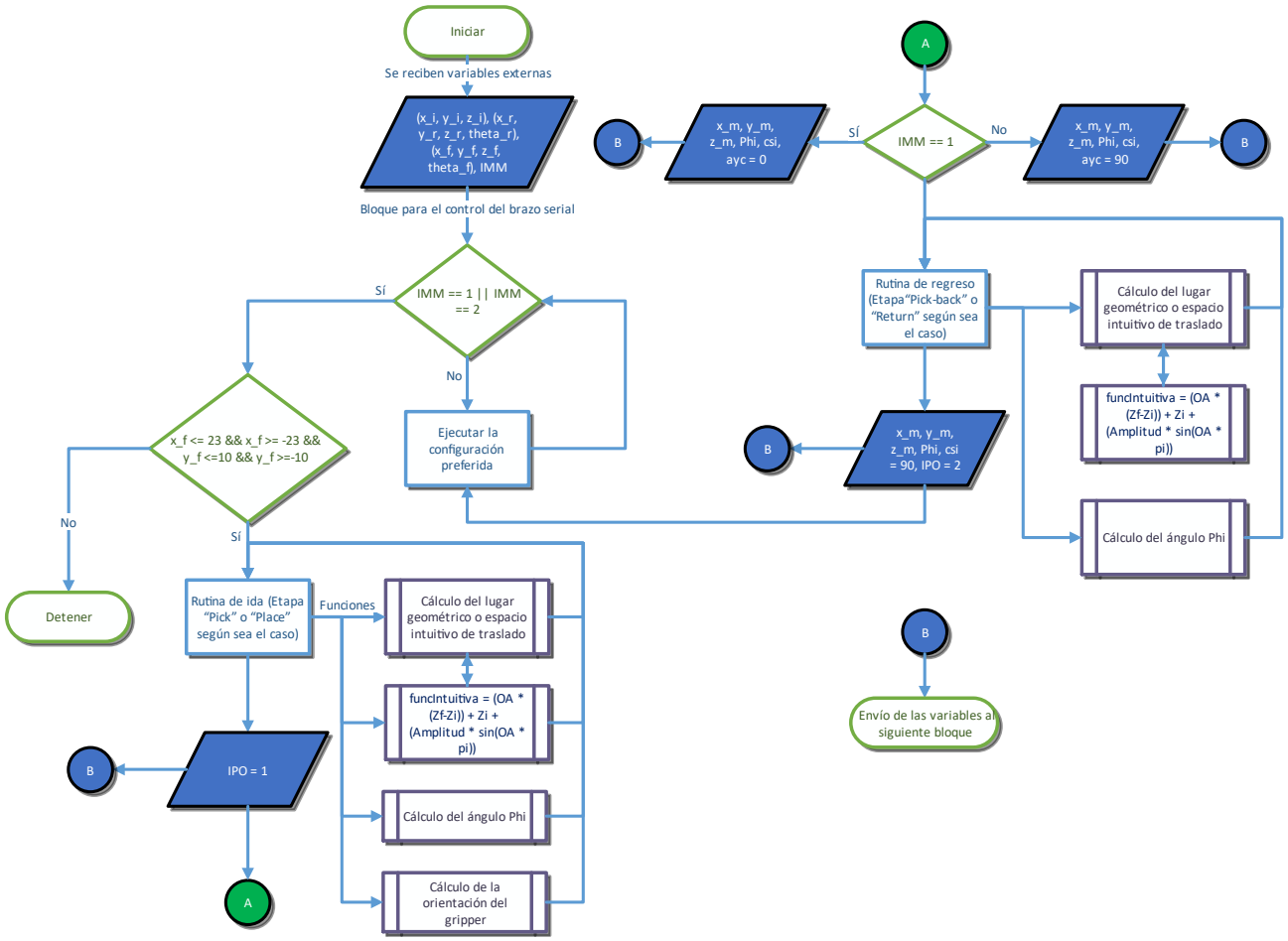


Figura 59.- Diagrama de flujo para ejecutar la tarea “pick and place” del brazo serial y cada una de sus etapas

CAPÍTULO 6.- COORDINACIÓN DE MOVIMIENTOS

6.1 Introducción al capítulo

A lo largo del capítulo se muestra el algoritmo y las técnicas que se utilizan para coordinar al manipulador móvil, principalmente se explica la técnica de campos potenciales para llevar a la plataforma móvil a los puntos deseados, también se incluye la técnica de planeación de movimientos conocida como "docking y predocking" que tiene como fin orientar correctamente al manipulador móvil. El presente texto es extraído del trabajo desarrollado por (Escobedo Castillo, 2012) y (González Villela, 2006).

6.2 Campos potenciales

El enfoque de campos potenciales es útil para que la plataforma móvil se mueva de una posición a otra. Por medio de la técnica de planeación de trayectoria seleccionada, uno puede llevar al manipulador móvil de una postura inicial a una final.

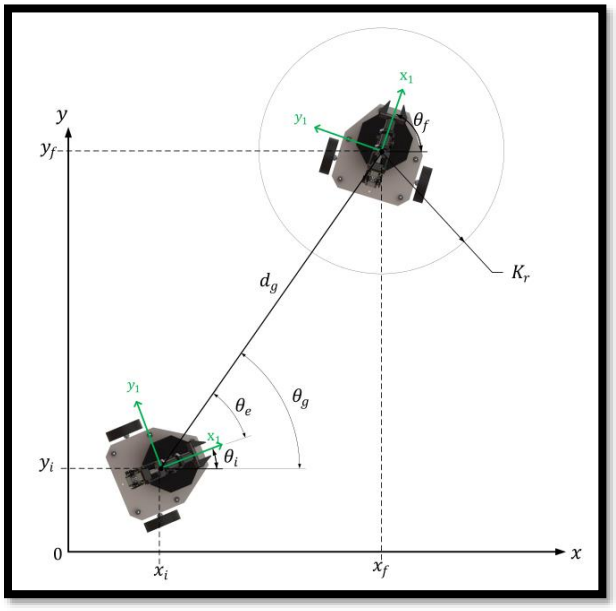


Figura 60.- Campos potenciales

El problema de llegar a la meta consiste en encontrar la velocidad y dirección a la cual el manipulador móvil debe dirigirse para llegar a la meta. Para esto, primero se define un vector de distancia de la plataforma a la meta $d_g = (x_f - x_i, y_f - y_i)$, donde la magnitud de esta distancia se encuentra mediante:

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

Ecuación 50

El ángulo con el cual se orienta a la meta se encuentra con:

$$\theta_g = ang\tan\frac{y_f - y_i}{x_f - x_i}$$

Ecuación 51

El ángulo de error se define como $\theta_e = \theta_g - \theta_i$. Entonces la plataforma móvil es llevada a la meta, tomando las siguientes reglas a seguir.

$$v = \begin{cases} v_{max} & si \quad |d_g| > K_r \\ \frac{v_{max}}{K_r} d_g & si \quad |d_g| \leq K_r \end{cases}$$

Ecuación 52

Donde v_{max} es la máxima velocidad de la plataforma con la que se desplaza y K_r es el radio del área de docking. Para encontrar una expresión que determine la velocidad angular de la plataforma móvil, se tiene lo siguiente:

$$\omega = \omega_{max} \sin\theta_e$$

Ecuación 53

Donde ω_{max} es la máxima velocidad angular que la plataforma móvil puede llegar a alcanzar. Texto basado en la tesis de (Escobedo Castillo, 2012).

6.3 Docking y predocking

Por campos potenciales la velocidad del manipulador móvil dependerá del área donde se encuentre, por ejemplo, si esta fuera del área de docking el manipulador se acercará a la meta a su máxima velocidad, mientras que si está cerca del área de docking el manipulador reduce su velocidad cada vez más hasta llegar a cero. La técnica de campos potenciales solo lleva a la plataforma móvil de un punto inicial a uno final, de forma que, si se quiere llegar a este punto final con una orientación deseada, se tiene que realizar una planeación.

Como se comentó anteriormente el punto de referencia es el punto de operación p del efector final del brazo serial. Por lo tanto, este punto debe de trazar la trayectoria deseada, siendo la guía del manipulador móvil.

El hecho de que el objeto a manipular tenga una orientación, implica que no puede ser tomado desde cualquier posición. Por lo tanto, el manipulador móvil, debe de orientarse lo más posible al ángulo de orientación del objeto en el plano $x-y$, permitiendo que el desplazamiento angular del brazo serial se encuentre en su rango. Por lo mencionado anteriormente, se tiene que ubicar una submeta la cual será un predocking (pD) de la plataforma móvil.

Para calcular el punto pD teniendo como único dato la postura del objeto a manipular, hay que desplazar el punto una distancia d_D, mediante la siguiente ecuación:

$$x_{pD} = x_f - d_D \cos\theta_D$$
$$y_{pD} = y_f - d_D \sin\theta_D$$

Ecuación 54

Con las ecuaciones anteriores (ecuación 54) se encuentran las coordenadas del punto pD, el cual está a una distancia d_D del punto final y a la misma orientación que posee el objeto a manipular, representado por el punto final.

En la figura 61 se puede apreciar dos radios ubicados uno en el punto final y el otro en el punto pD, estos radios son los radios que en la ecuación 52 se expresan como K_r, y son los encargados de reducir la velocidad de la plataforma móvil en cuanto esta entre en el área descrita por este radio. Se aprecia que son de una misma longitud K_D, por lo tanto, para que la plataforma detenga su movimiento debe de incidir el radio K_D con el radio K_{MM}, que se ubica en el punto de referencia del manipulador móvil (punto de operación p), teniendo que:

$$v = 0, \omega = 0 \quad si \quad \begin{matrix} d_{pD} \leq K_{MM_{pD}} \\ d_D \leq K_{MM_D} \end{matrix}$$

Ecuación 55

En la ecuación 55, se indica que si la distancia del punto de operación p al punto de predocking o de docking, según sea al caso, es menor al radio K_{MM} se detiene el robot. También se nota que hay dos diferentes radios K_{MM}, uno para cada punto. Lo anterior se debe a que para el primer punto (pD), este radio tiene que ser muy pequeño, lo suficiente para que la plataforma no pierda su rumbo original, pues este es el punto donde la plataforma "gira" para coincidir con la orientación del objeto a manipular. Finalmente, el radio K_{MM_D} indica que la plataforma ya se encuentra orientándose para recoger el objeto y este radio es de mayor longitud al anterior, pues el manipulador tiene que detenerse a una distancia suficiente, para que el brazo sujete o deposite con el efector final al objeto. Texto basado en la tesis de (Escobedo Castillo, 2012).

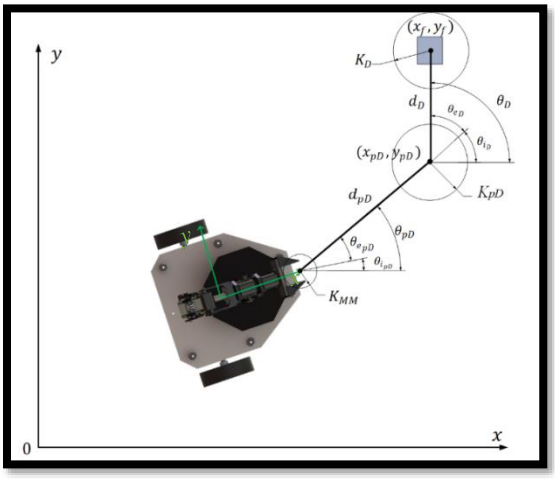


Figura 61.- Docking y predocking

6.4 Algoritmo de coordinación

La estrategia para tratar al brazo serial y la plataforma móvil como una sola se da gracias al acoplamiento y el establecimiento de las técnicas antes descritas, con el fin de realizar el movimiento del sistema en su totalidad. Sin embargo, es necesario establecer el algoritmo coordinante de ambos sistemas para el desarrollo de la tarea dentro del espacio inteligente. El algoritmo y la nomenclatura utilizada para su programación se basa en el capítulo V.3.3 de la tesis de (Escobedo Castillo, 2012), tal algoritmo se adapta para poder realizar la tarea de sujetar y depositar un solo objeto. A manera de resumen los pasos principales para coordinar al manipulador móvil son:

1. Detectar el manipulador móvil, objeto a manipular y su correspondiente destino en coordenadas globales.
2. Seleccionar la secuencia de submetas a las que tiene que desplazarse el manipulador móvil.
3. Evaluar en qué parte de la tarea se encuentra el manipulador móvil y mandar el punto al que debe dirigirse.
4. Llevar el brazo serial a su configuración preferida, en cuanto llegue a ella, ejecutar el siguiente paso.
5. Calcular el predocking y mover la plataforma al mismo.
6. Una vez llegado al predocking, indicar que cambie el punto al docking.
7. Indicar cuando se haya llegado al docking e iniciar la operación “pick and place”, primera parte: sujetar. *Etapas:* “*pick”* y *“pick back”*.
8. Completada la operación de sujetar, indicar que se encuentra nuevamente en la configuración preferida y comenzar el movimiento de la plataforma, cambiando el punto final al destino.
9. Calcular predocking y mover la plataforma al predocking.
10. Una vez llegado al predocking, dirigirse al docking.
11. Cuando se llegue al docking, comenzar con la segunda parte de la operación “pick and place”: colocar. *Etapa:* “*place”*.
12. Colocado el objeto en su destino, regresar a la configuración preferida e indicar que se llegó a ella. *Etapa: “return”*.

Es importante señalar que el algoritmo propuesto por (Escobedo Castillo, 2012), se puede extender a varios objetos, pero al no ser el objetivo de esta tesis, se realiza el experimento para un único objeto. Además, se utilizan tres indicadores para la correcta coordinación y programación, estos son:

- **IPO (indicador de posición óptima)**: determina el instante en el que el brazo serial se encuentra en su configuración preferida, este indicador consiste en tres valores.
  - **0**: Cuando el brazo serial se encuentra en la configuración preferida.
  - **1**: Cuando el brazo serial se encuentra realizando la operación "pick and place".
  - **2**: Cuando el brazo serial termina una parte de la operación "pick and place" y regresa a la configuración preferida.
- **ILP (indicador de llegada al punto)**: indicador de la plataforma móvil que determina el evento que realiza actualmente la plataforma y consiste en dos valores.
  - **0**: La plataforma móvil se encuentra en movimiento, independientemente del punto al que se dirija (no ha llegado).
  - **1**: La plataforma móvil está estacionada, es decir, ya llegó al punto correspondiente.

Los dos indicadores que monitorean lo que realiza cada parte del manipulador móvil, disparan un tercer indicador, que es consecuencia de lo que se realice en el momento, este es:

- **IMM (indicador de evento del manipulador móvil)**: mediante combinaciones de los anteriores indicadores se determina lo que está realmente realizando el manipulador. Consiste en tres valores:
  - **0**: El manipulador móvil está desplazándose en su entorno.
  - **1**: El manipulador móvil está realizando la primera parte de la operación: pick (sujetar).
  - **2**: El manipulador móvil está realizando la segunda parte de la operación: place (colocar).

El diagrama de flujo que representa el algoritmo implementado y la interacción de los indicadores para la correcta coordinación del manipulador móvil se muestra en la figura 62.

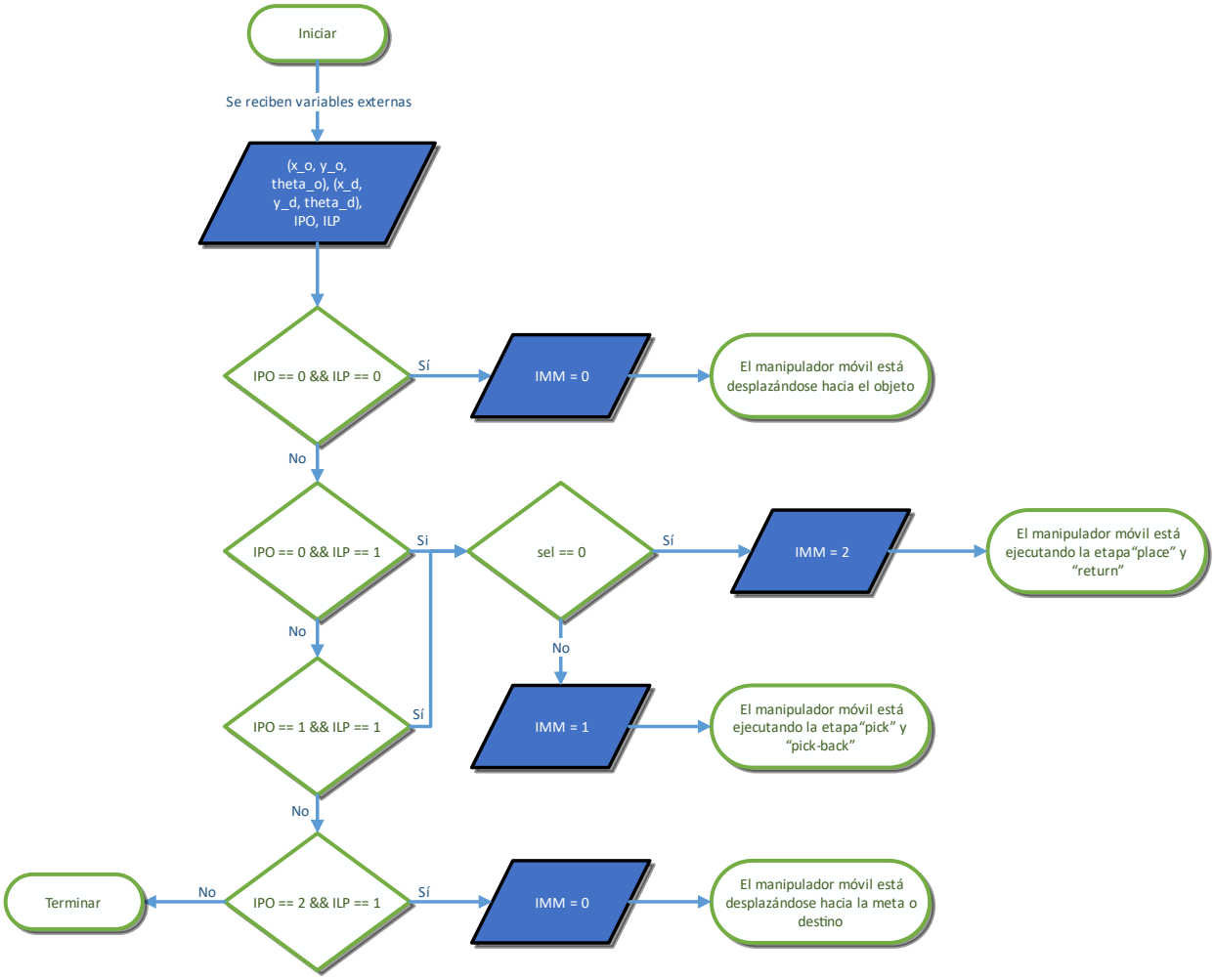


Figura 62.- Diagrama de flujo del algoritmo para la coordinación de movimientos del manipulador móvil

CAPÍTULO 7.- DESCRIPCIÓN DEL SISTEMA Y EL ESPACIO INTELIGENTE

7.1 Introducción al capítulo

Las partes del sistema para el modelo funcional, se divide en el sistema de visión, el sistema de control y la comunicación entre el manipulador móvil y la computadora. El diagrama de la figura 63 describe gráficamente la circulación de la información de forma general.

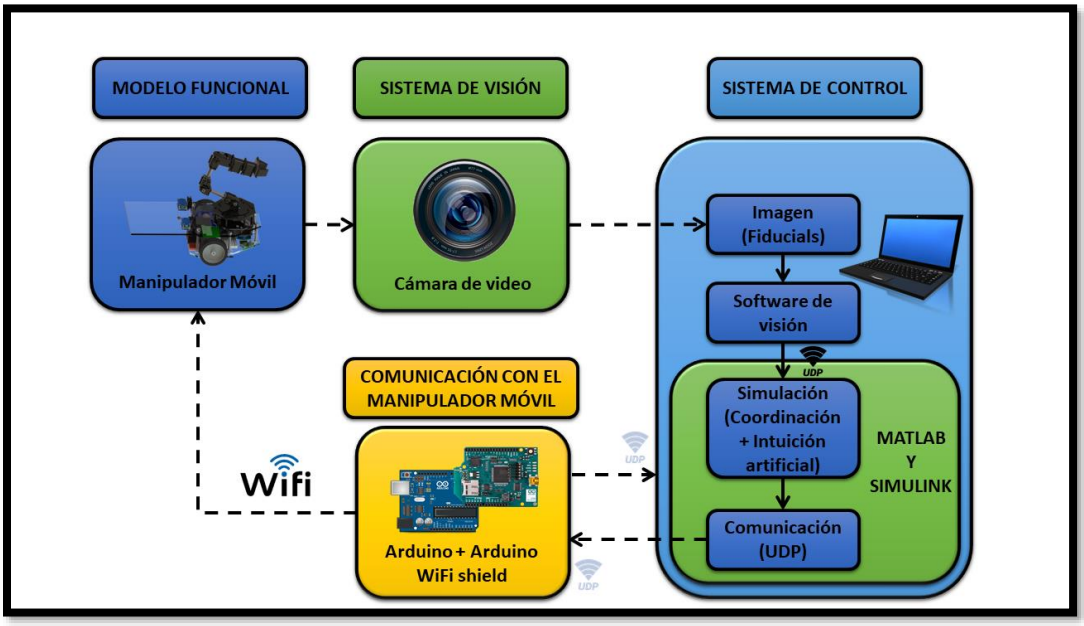


Figura 63.- Diagrama de funcionamiento del sistema

Los agentes y componentes que estructuran el funcionamiento e interconexión entre los elementos que conforman al sistema con intuición artificial están dispuestos en la figura 64:

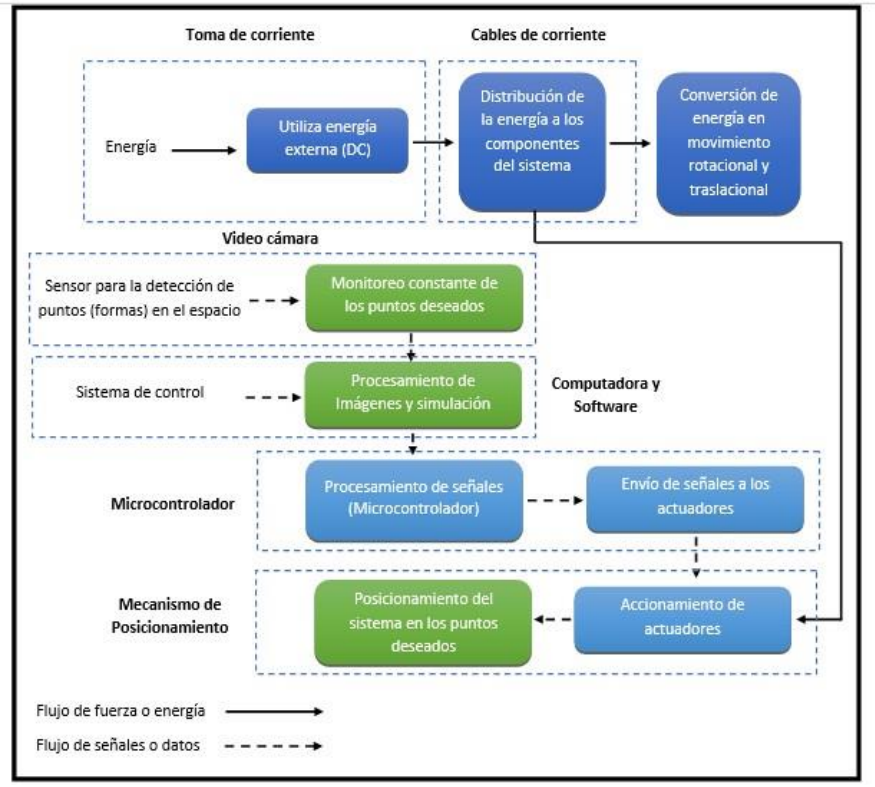


Figura 64.- Diagrama de los elementos constitutivos del sistema

Un espacio inteligente debe ser capaz de "estar consciente" de lo que pasa dentro de él, y así poder actuar de acuerdo con lo deseado dentro del espacio, en este caso el traslado de objetos de un manipulador móvil. Se habla de cuatro capas que conforman al espacio (entiéndase el concepto de capa como un nivel abstracto para el envío de información desde un punto de vista computacional):

Capa pasiva de percepción: Está compuesta de sensores que captan la información, pueden ser de distintos tipos de acuerdo con lo que se requiere percibir. Los sensores pueden estar fijos o estar dentro de la capa de interacción con los usuarios.

Capa de interacción: La componen los elementos que pueden controlarse e interactúan de manera física con el medio.

Capa de comunicaciones: Esta capa se refiere a la red de interconexiones que hay entre los diversos elementos para comunicarse.

Capa de inteligencia: Es la parte encargada de procesar la información generada en la capa de percepción, y quien controla a la capa de interacción, también es la responsable del intercambio de información mediante la capa de comunicaciones. Mediante programación a esta capa se le añade la intuición artificial para el desarrollo de la tarea "pick and place". Los elementos más importantes que conforman y permiten la interacción con el espacio inteligente son los siguientes:

7.2 Arduino WiFi-Shield

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

El arduino Wifi shield viene con una completa biblioteca de funciones para transmisión de datos y configuración del shield desarrollada por MCI electronics. Arduino Wifi shield es ideal para monitoreo y control de equipos usando arduino y comunicándose vía internet. La configuración del módulo se realiza vía web, aunque pueden configurarse algunos parámetros usando el modo de configuración vía serial.

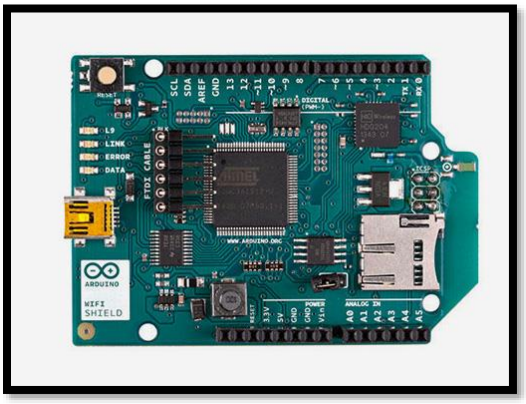


Figura 65.- Módulo WiFi shield, fuente: (Arduino WiFi shield A000058, 2009)

Este módulo se conecta al microcontrolador arduino mega 2560 y es el encargado de establecer la comunicación vía WiFi con Matlab/Simulink mediante el protocolo de datagramas de usuario (UDP), para entender el funcionamiento del protocolo se sugiere consultar la tesis de (Gálvez Valadez, 2015).

7.3 Sistema de visión

El sistema de visión para su funcionamiento requiere de una cámara y un software, este último contiene el algoritmo para el procesamiento de imágenes. De forma general, la cámara funge como sensor para observar al manipulador móvil, el objetivo y la meta desde una vista aérea, y mediante la localización de los identificadores dentro de su campo de visión (delimitado por la cámara), el software procesa la imagen para enviar los datos necesarios (posición, orientación, velocidad, etc.) los cuales se traducen como datos del manipulador móvil, objeto y meta. Por último, estos datos son enviados al sistema de control (el encargado de utilizar dichos datos). La cámara y el software utilizados para el desarrollo de los experimentos son:

7.3.1 Cámara

La cámara que se utiliza es una de la marca Logitech modelo C920, su conexión con la computadora es alámbrica a través del puerto USB. Las principales características técnicas que posee son (Logitech, 2013):

- Videoconferencias Full HD 1080p (hasta 1920 x 1080 píxeles) con la versión más reciente de Skype para Windows
- Videoconferencias HD 720p (1280 x 720 píxeles) con clientes compatibles
- Grabaciones de video Full HD (hasta 1920 x 1080 píxeles)
- Compresión de video H.264
- Micrófonos estéreo integrados con reducción de ruido automática
- Corrección automática de iluminación escasa
- Clip universal compatible con trípodes para monitores LCD, CRT o laptops

Software de cámara web Logitech para Windows:

- Controles de panorámico, inclinación y zoom
- Captura de video y fotos
- Seguimiento facial
- Detección de movimiento

Figura 66.- Cámara web Logitech modelo C920, fuente: (Logitech, 2013)

7.3.2 ReacTIVision

Se utiliza el software llamado ReacTIVision de código abierto, el cual funge como rastreador de imágenes, principalmente orientado al reconocimiento de marcadores fiduciales, estos son capturados a través del flujo de información mediante video en tiempo real. 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 (Carlos Ávila, 2015). Una desventaja de este sistema de visión es el reto de detectar espacialmente (x, y, z) un objeto dentro del espacio de trabajo, ya que ReacTIVision solo es capaz de proporcionarnos las coordenadas en el plano (x, y).

Figura 67.- Ícono del software ReacTIVision, fuente: (Reactivision Engine, 2014)

Se emplean fiduciales porque es un sistema de visión robusto y puede soportar perturbaciones siempre y cuando se vea el fiducial, las deformaciones debido a la cámara pueden ser compensadas. Las rotaciones y traslaciones que puedan sufrir los diferentes objetos son fácilmente detectables, el fondo puede ser relativamente complejo sin que nos afecte, siempre y cuando la superficie de operación sea plana (Carlos Ávila, 2015).

CAPÍTULO 8.- EXPERIMENTACIÓN

8.1 Introducción al capítulo

En el campo de la industria es frecuente hacer experimentos o pruebas con la intención de resolver un problema o comprobar una idea (conjetura, hipótesis); por ejemplo, hacer algunos cambios en los materiales, métodos o condiciones de operación de un proceso, probar varias temperaturas en una máquina hasta encontrar la que da el mejor resultado o crear un nuevo material con la intención de lograr mejoras o eliminar algún problema (Gutiérrez Pulido & de la Vara Salazar, 2008). Por lo tanto, con el fin de demostrar y observar el comportamiento del espacio intuitivo programado en un manipulador móvil para llevar a cabo tareas de traslado, se realizan las siguientes pruebas:

- **Experimento 1:** La unión del punto inicial y final de referencia para el traslado del objeto debe describir una línea paralela al plano del horizonte (línea horizontal).
- **Experimento 2:** Ejecutar la tarea local descrita en el capítulo tres, la unión del punto inicial y final de referencia para el traslado del objeto debe describir una línea perpendicular al plano del horizonte (línea vertical).
- **Experimento 3:** Ejecutar la tarea global descrita en el capítulo tres.

Los resultados de cada experimento se muestran en el capítulo 9, a lo largo de este capítulo se presenta la configuración que se realizó en cada parte componente del sistema para llevar a cabo los experimentos.

8.2 Software empleado para la programación en tiempo real

8.2.1 Arduino IDE

El software de arduino consiste en dos elementos: un entorno de desarrollo (IDE) (basado en el entorno de processing y en la estructura del lenguaje de programación Wiring), y en el cargador de arranque (bootloader) que es ejecutado de forma automática dentro del microcontrolador en cuanto este se enciende. Las placas arduino se programan mediante una computadora, usando comunicación serial (Arduino, 2013).

La programación realizada en el software arduino para la recepción y envío de datagramas mediante conexión WiFi se expone en el apéndice B.

8.2.2 Matlab

Es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes) y las de Simulink con los paquetes de bloques (blocksets).

Es un software muy usado en universidades y centros de investigación y desarrollo, en los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

MATLAB puede llamar funciones y subrutinas escritas en C o Fortran. Se crea una función que permite que sean pasados y devueltos tipos de datos de MATLAB. Los archivos objeto dinámicamente cargables, creados compilando esas funciones se denominan "MEX-files", aunque la extensión de nombre de archivo depende del sistema operativo y del procesador (MATLAB, 2015).

Matlab cuenta con un módulo llamado Real Time Windows Target. Real Time Windows Target soporta dos modos de simulación: modo normal, para una ejecución simple de tiempo real con acceso a dispositivos I/O, y modo externo, para tiempo real de alto rendimiento (Cruz López, 2015).

8.2.3 Configuración de Matlab en tiempo real

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

Matlab tiene incluido un módulo llamado Real Time Windows Target que provee de un motor de tiempo real que se ejecuta en modo kernel para establecer la conexión entre bloques de modelos de Simulink en una computadora corriendo Microsoft Windows y una gran variedad de tarjetas I/O, para poder interactuar con sensores, actuadores y otros dispositivos a desarrollar. Esto permite crear y controlar un sistema en tiempo real para prototipos rápidos o simulación de hardware en ciclo. Real Time Windows Target sincroniza los datos entre el motor en tiempo real y Simulink; soporta dos modos de simulación (Cruz López, 2015):

- *Modo normal*, para una ejecución simple de tiempo real con acceso a dispositivos I/O ejecutándose en paralelo con un modelo de Simulink, alcanzando un rendimiento superior a 500 Hz.

- *Modo externo*, para tiempo real de alto rendimiento en donde el motor en tiempo real carga el archivo binario resultante de la compilación y los controladores de dispositivos de I/O, y establece una conexión con Simulink, alcanzando un rendimiento cercano a 20 kHz.

Durante la ejecución en tiempo real del modelo, el kernel interviene para dar al modelo prioridad para utilizar la CPU con la finalidad de ejecutar cada actualización del modelo en los tiempos de muestreo prescritos. Para garantizar un periodo de muestreo preciso el kernel reprograma el reloj de la computadora a una frecuencia mayor. Durante la ejecución de la aplicación en tiempo real almacena los datos en buffers, y posteriormente el contenido de los buffers es recuperado por Simulink para imprimirlas en pantalla. Una aplicación en tiempo real tiene las siguientes características (Cruz López, 2015):

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

Por su parte Matlab/Simulink garantiza que el programa se ejecute en tiempo real, por otro lado, el sistema de visión (software ReacTIVision) y el microcontrolador arduino no garantizan el cumplimiento de una ejecución en tiempo real, esto se fundamenta en las observaciones realizadas durante los experimentos y en trabajos anteriores al presente, por ejemplo, en la tesis de (Cruz López, 2015). Los parámetros configurados en Simulink para poder establecer la comunicación en tiempo real en el desarrollo de la experimentación se muestran en el Anexo D.

8.2.4 Simulink

Simulink es un entorno de programación visual, que funciona sobre el entorno de programación Matlab. Es un entorno de programación de más alto nivel de abstracción que el lenguaje interpretado Matlab (archivos con extensión .m). Simulink genera archivos con extensión ".mdl", ".slx", etc. Es una herramienta de simulación de modelos o sistemas, con cierto grado de abstracción de los fenómenos físicos involucrados en los mismos. Se hace hincapié en el análisis de sucesos, a través de la concepción de sistemas (cajas negras que realizan alguna operación) (Simulink, 1996).

Es ampliamente usado en ingeniería electrónica en temas relacionados con el procesamiento digital de señales (DSP), involucrando temas específicos de ingeniería biomédica, telecomunicaciones, entre otros, también es muy utilizado en ingeniería de Control y Robótica (Simulink, 1996).

8.2.5 Stateflow

Stateflow® es un entorno para modelar y simular lógica de decisión combinatoria y secuencial basado en máquinas de estado y diagramas de flujo. Stateflow permite combinar representaciones gráficas y tabulares, lo que incluye diagramas de transición de estado, diagramas de flujo, tablas de transición de estado y tablas de verdad, con el fin de modelar la forma en que el sistema reaccionará ante los eventos, las condiciones basadas en el tiempo y las señales de entrada externas. Se utiliza Stateflow a fin de diseñar la lógica para aplicaciones de control de supervisión, planificación de tareas y gestión de fallos. Stateflow incluye animación de diagramas de estado y comprobaciones estáticas y en tiempo de ejecución para determinar si el diseño es coherente y está completo antes de la implementación (Stateflow, 2000).

Los bloques programados en Simulink y Stateflow para llevar a cabo las simulaciones de la presente investigación se muestran en el apartado "Código Matlab" del apéndice C, exponiendo su configuración externa e interna, así como el código programado en cada subsistema.

8.3 Marcadores fiduciales empleados para la experimentación

Para poder adquirir en tiempo real con el sistema de visión la posición y orientación del manipulador móvil, el objeto y la meta (lugar en el cual se deposita al objeto), se utilizan los siguientes marcadores fiduciales, es importante notar el número de ID de cada marcador.

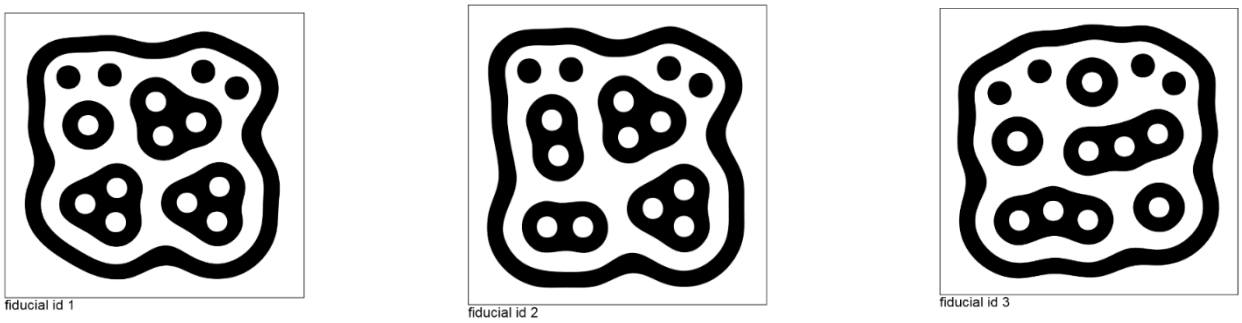


Figura 68.- Marcadores fiducial; ID 1, 2 y 3

Para el desarrollo del experimento el fiducial con ID 1 se utiliza para detectar al manipulador móvil, el ID 2 detecta al objeto y por último el ID 3 detecta la meta.

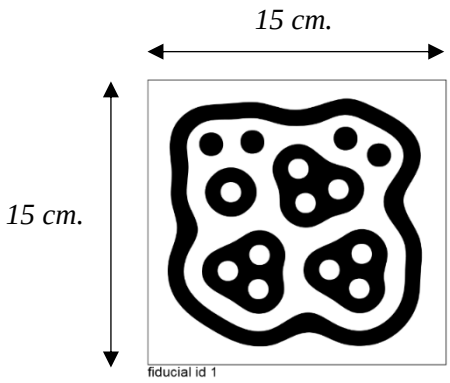


Figura 69.- Dimensiones de los fiducials empleados para los experimentos

Nota importante: el sistema de visión solo se programa para reconocer tres marcadores dentro del espacio inteligente, debido a esto, si la cantidad de fiducials es menor o mayor durante los experimentos se enviarán datos erróneos a Matlab.

8.4 Comunicación WiFi-UDP

User Datagram Protocol (UDP) es un protocolo del nivel de transporte basado en el intercambio de datagramas (encapsulado de capa 4 o de transporte del modelo OSI). 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. Su uso principal es para protocolos como DHCP, BOOTP, DNS y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos. UDP utiliza puertos para permitir la comunicación entre aplicaciones, el campo de puerto tiene una longitud de 16 bits, por lo que el rango de valores válidos va de 0 a 65535. El puerto 0 está reservado, pero es un valor permitido como puerto origen si el proceso emisor no espera recibir mensajes como respuesta.

- Los puertos 1 a 1023 se llaman puertos "bien conocidos" y en sistemas operativos tipo Unix enlazar con uno de estos puertos requiere acceso como superusuario.
- Los puertos 1024 a 49151 son puertos registrados.
- Los puertos 49152 a 65535 son puertos dinámicos y son utilizados como puertos temporales, sobre todo por los clientes al comunicarse con los servidores.

Información extraída de: (User Datagram Protocol, 2015).

La tarjeta arduino WiFi shield permite establecer comunicación con el software Simulink de Matlab por medio del protocolo UDP, de esta manera se puede llevar acabo la transmisión de datos en tiempo real y vía inalámbrica. En el software Matlab se utilizan los siguientes bloques para enviar y recibir paquetes vía UDP; se programa un milisegundo en cada bloque para el envío y recepción de paquetes.

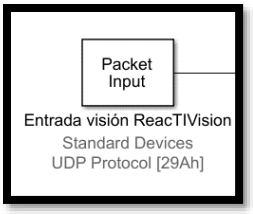


Figura 70.- Bloque para la recepción de datos del software de visión vía UDP

Figura 71.- Bloque de envío y recepción de paquetes UDP a la placa arduino, para el control y retroalimentación de los servomotores Dynamixel

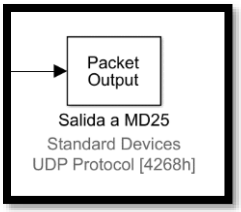


Figura 72.- Bloque para el envío de paquetes UDP a la placa arduino, para el control de las velocidades de los motorreductores EMG30

Los puertos UDP que se programan en Matlab son:

Para el bloque Packet Input (entrada visión ReacTIVision)

- Puerto local: 666
- Dirección remota: 127.0.0.1
- Puerto UDP remoto: 12000

Para el bloque Packet Output (salida a los servomotores Dynamixel)

- Puerto local: 15000
- Dirección remota: 192.168.0.8
- Puerto UDP remoto: 5678

Para el bloque Packet Input (entrada de datos de los servomotores)

- Puerto local: 16000
- Dirección remota: 192.168.0.8
- Puerto UDP remoto: 8765

Para el bloque Packet Output (salida a MD25)

- Puerto local: 17000
- Dirección remota: 192.168.0.8
- Puerto UDP remoto: 1234

8.5 Configuración del espacio inteligente

El espacio de trabajo mostrado en la figura 73 cuenta con 2.04 x 1.70 metros, con la cámara a 2.32 metros de altura. La cámara posee un ancho y alto de fotograma de 1024 x 768 pixeles.

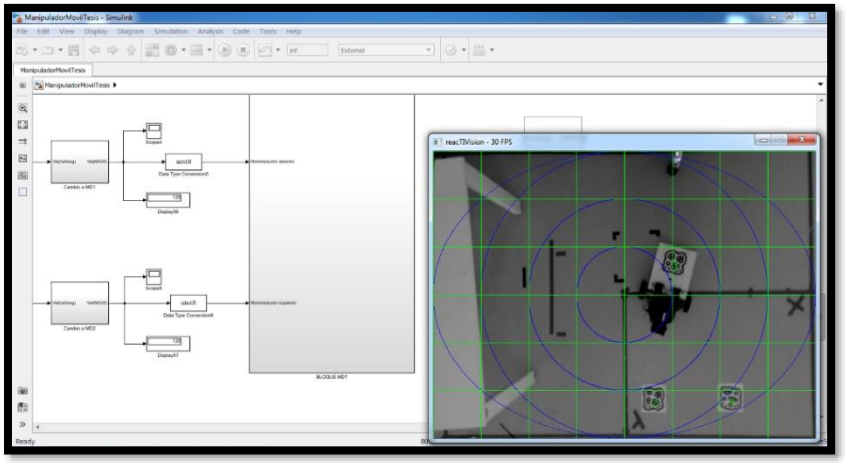


Figura 73.- Dimensiones del espacio inteligente

Como primer paso para la correcta configuración de espacio en el cual se desarrollan los experimentos se debe lograr que la lente de la cámara esté lo más paralela posible al piso, para evitar deformaciones del espacio observado, siendo conveniente utilizar una plomada para la correcta calibración de la cámara y la malla de calibración en el software ReacTIVision.

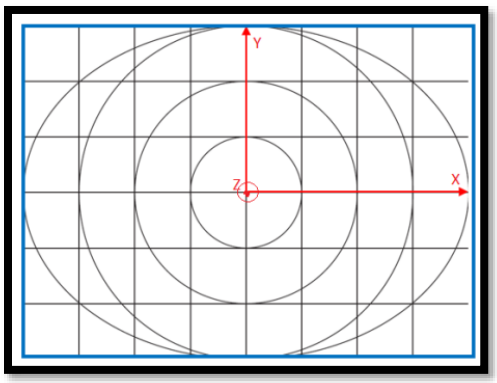


Figura 74.- Malla para calibrar el entorno de trabajo en el software de visión, fuente: (Pacheco Jiménez & Ávila García, 2017)

En el "Módulo de visión" programado en Matlab/Simulink (véase el apéndice C), se realiza un ajuste con el fin de obtener las posiciones en centímetros. Las fórmulas empleadas para realizar el ajuste son:

$$x_s = \left(\frac{x}{100}\right)(-2l) + l \quad \textbf{Ecuación 56}$$

$$y_s = \left(\frac{y}{100}\right)(2a) - a \quad \textbf{Ecuación 57}$$

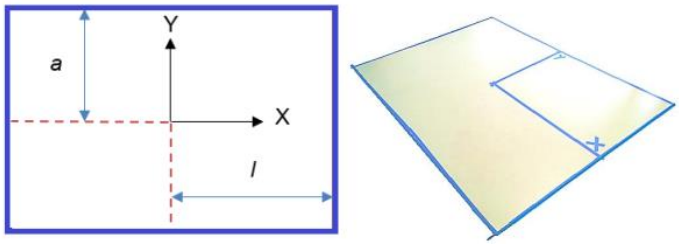


Figura 75.- Configuración del espacio de trabajo a centímetros, fuente: (Hernández Calderón, 2016)

Finalmente, como el análisis cinemático de la plataforma móvil parte del centro de esta, sin embargo, dada la configuración del manipulador móvil no es posible colocar el fiducial en el centro de la plataforma, razón por la cual se tiene que colocar una placa trasera a manera de extensión para situar al fiducial; modificación física que enseguida obliga a compensar de alguna forma este desplazamiento y rotación del punto.

El símbolo fiducial se encontrará una distancia d_o, medida del centroide del fiducial al centro del objeto a manipular o su destino, según sea el caso; con la misma orientación del fiducial. Por otra parte, el fiducial que corresponde al manipulador móvil se encuentra a una distancia d_{MM} medida del punto P al centro del fiducial, lo anterior descrito se realiza con las siguientes expresiones:

$$x_{o_{Ajustado}} = x_o - d_o \cos\theta_o$$
$$y_{o_{Ajustado}} = y_o - d_o \sin\theta_o$$

Ecuación 58

$$x_{d_{Ajustado}} = x_d - d_o \cos\theta_d$$
$$y_{d_{Ajustado}} = y_d - d_o \sin\theta_d$$

Ecuación 59

$$x_{MM_{Ajustado}} = x_{MM} - d_{MM} \cos\theta_{MM}$$
$$y_{MM_{Ajustado}} = y_{MM} - d_{MM} \sin\theta_{MM}$$

Ecuación 60

Donde el índice o representa al objeto y el índice d al destino, así como el índice MM se refiere al manipulador móvil.

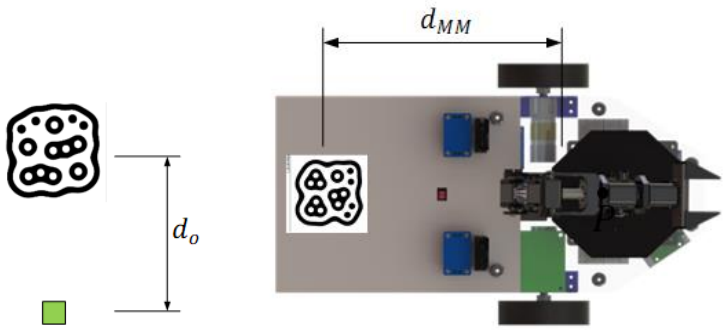


Figura 76.- Traslado de puntos

8.6 Configuración de los servomotores Dynamixel AX-12A

Para configurar el ID, baudrate y otros parámetros de cada servomotor Dynamixel se hace uso del software proporcionado por ROBOTIS, llamado “RoboPlus”.

Figura 77.- Software Roboplus, fuente: (Roboplus 1.0, 2016)

Este software se utiliza para programar los límites de giro de cada servomotor, con el fin de evitar daños o golpes por giros bruscos y por recepción de datos erróneos, para realizar la programación es necesario adquirir un USB2Dynamixel:

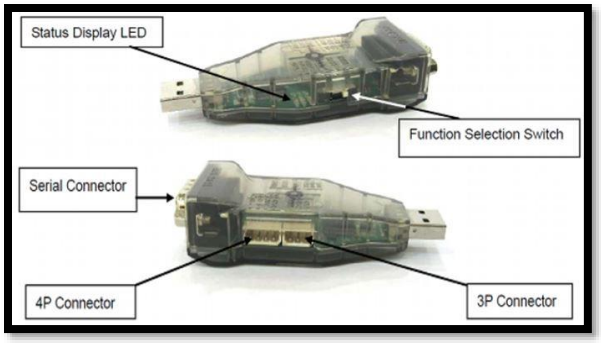


Figura 78.- USB2Dynamixel, fuente: (ROBOTIS e-Manual v1.31.30, 2010)

También se recomienda utilizar un adaptador SMPS2Dynamixel para proporcionar energía a los servomotores.

Figura 79.- SMPS2Dynamixel, fuente: (SMPS2DYNAMIXEL, 2014)

Los límites programados en cada servomotor son los siguientes:

Servomotor ID 1 (θ_1):

Ángulo límite en sentido de las agujas del reloj (ALCC): 60°

Ángulo límite en sentido contrario de las agujas del reloj (ALCCW): 240°

Servomotor ID 2 (θ_2):

ALCC: 60°

ALCCW: 240°

Servomotor ID 3 (θ_3):

ALCC: 48°

ALCCW: 150°

Servomotor ID 4 (θ_4):

ALCC: 12°

ALCCW: 150°

Servomotor ID 5 (θ_5):

ALCC: 60°

ALCCW: 240°

Servomotor ID 6 (θ_6):

ALCC: 60°

ALCCW: 150°

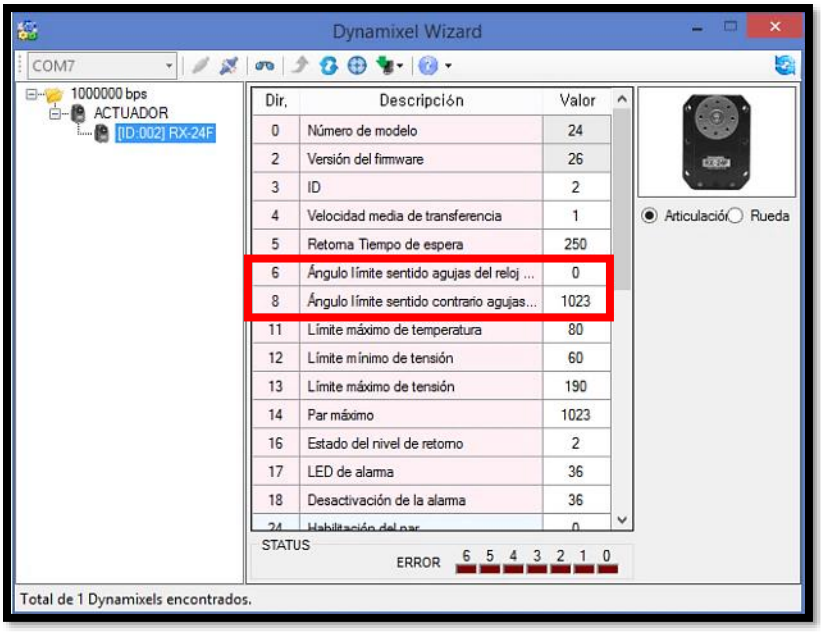


Figura 80.- Ejemplo de la programación de los límites angulares de cada servomotor, fuente: (Castañeda Espinosa & Pedro Mendoza, 2016)

CAPÍTULO 9.- RESULTADOS

En este capítulo se muestran los resultados obtenidos durante las simulaciones realizadas en el software Simulink de Matlab®. Solo se añaden las gráficas más importantes y que denotan el comportamiento de la fórmula programada para el desarrollo de la tarea de traslado.

9.1 Movimiento horizontal

Experimento 1: La unión del punto inicial y final de referencia para el traslado del objeto debe describir una línea paralela al plano del horizonte (línea horizontal, plano "yz")[1]. Es importante destacar que el experimento se lleva a cabo fuera del espacio inteligente, por lo tanto, la plataforma móvil no realiza ningún movimiento y el punto inicial y final es programado por el usuario. Las condiciones iniciales para el desarrollo del experimento son:

Coordenadas iniciales:

$X_{mo1} = 15$ cm.
$Y_{mo1} = -15$ cm.
$Z_{mo1} = 15.3$ cm.

Coordenadas finales:

$X_{mf1} = 15$ cm.
$Y_{mf1} = 15$ cm.
$Z_{mf1} = 15.3$ cm.

[1] Nota: estos puntos son programados en base al marco coordenado del brazo serial (ver figura 58).

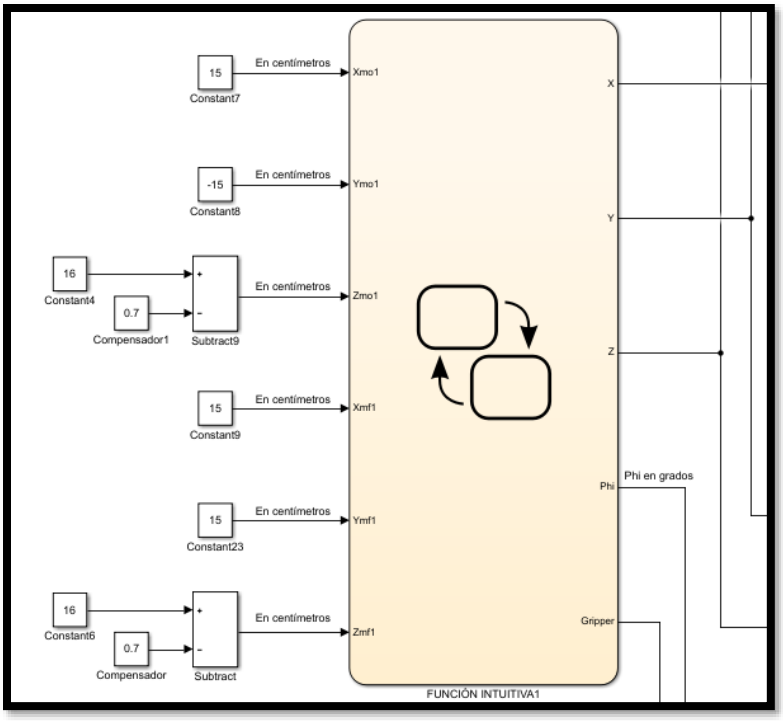


Figura 81.- Condiciones iniciales para realizar el experimento uno

Una vez que se indican las condiciones iniciales, se ejecuta la simulación para realizar los cálculos correspondientes y enviar los datos al brazo serial en tiempo real, las salidas generadas en el bloque de la figura 81 así como su evolución a través del tiempo se muestran en las siguientes gráficas:

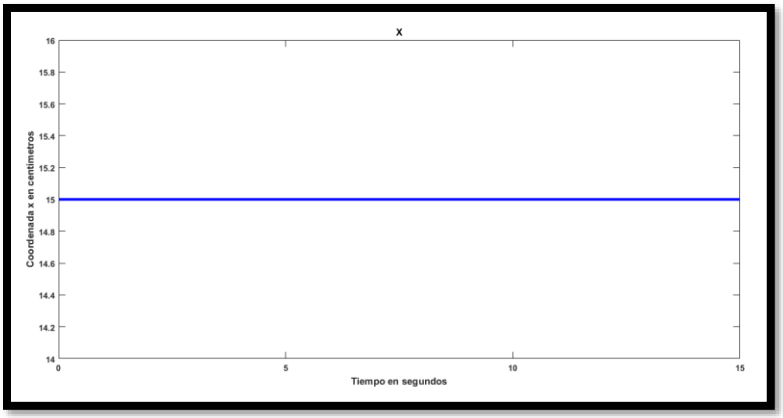


Figura 82.- Coordenada X , experimento 1

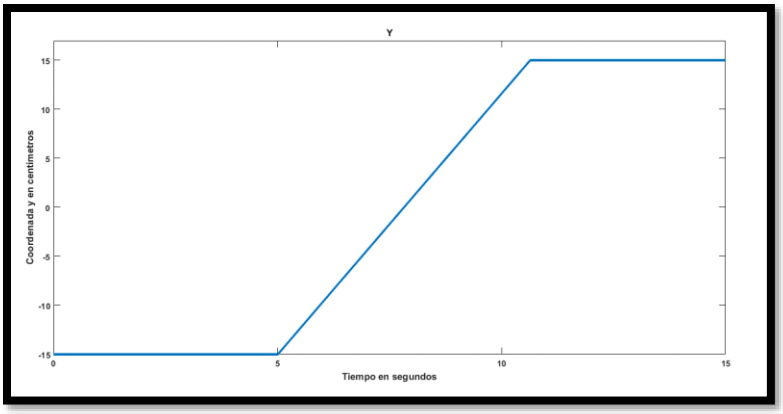


Figura 83.- Coordenada Y , experimento 1

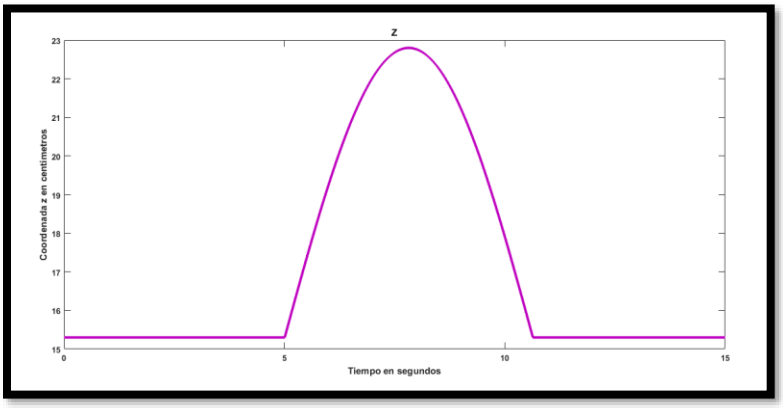


Figura 84.- Coordenada Z , movimiento en el plano " *yz* "

Como se puede apreciar, las figuras 82 (*eje x*), 83 (*eje y*) y 84 (*eje z*) hacen una analogía a las figuras 55 (*eje x*), 56 (*eje y*) y 57 (*eje z*) que representan los patrones intuitivos observados para deducir las ecuaciones 45, 46 y 47, por esa razón se elige realizar un experimento donde el movimiento del efector final se efectúe en el plano " *yz* " del sistema coordenado del brazo serial.

La evolución en el tiempo de la coordenada x en la figura 82 no coincide con la mostrada en la figura 55, lo mismo sucede con la coordenada y de la figura 83 y 56, esto se debe al giro de 90 grados que se realiza en el eje z del sistema coordenado del brazo serial (ver figura 58).

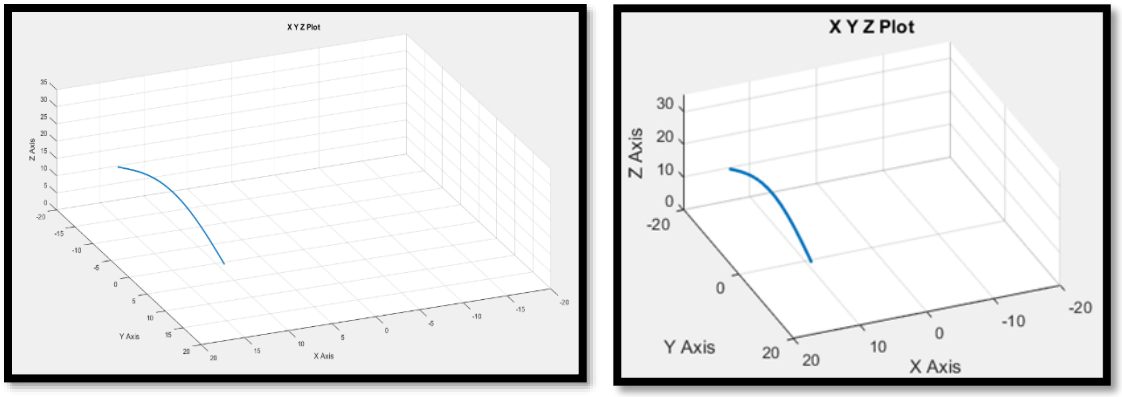


Figura 85.- Lugar geométrico descrito, movimiento en el plano " *yz* "

En el espacio tridimensional las coordenadas X , Y y Z mostradas en las figuras 82, 83 y 84 generan en su conjunto el lugar geométrico (espacio intuitivo) graficado en la figura 85.

En la figura 86 se muestra la variación del ángulo ϕ con el fin de alcanzar cada punto del lugar geométrico, de esta manera se logra orientar a la muñeca del brazo en un ángulo de 90° para que sujete al objeto.

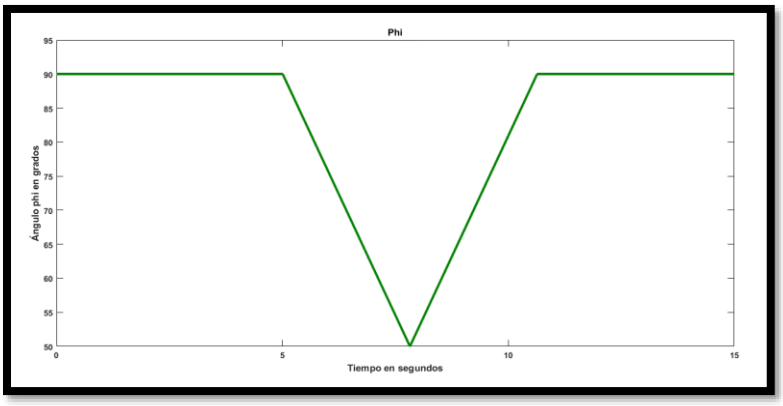


Figura 86.- Ángulo phi (ϕ), experimento 1

El ángulo de giro de la muñeca mostrado en la figura 87 se mantiene constante en 90°.

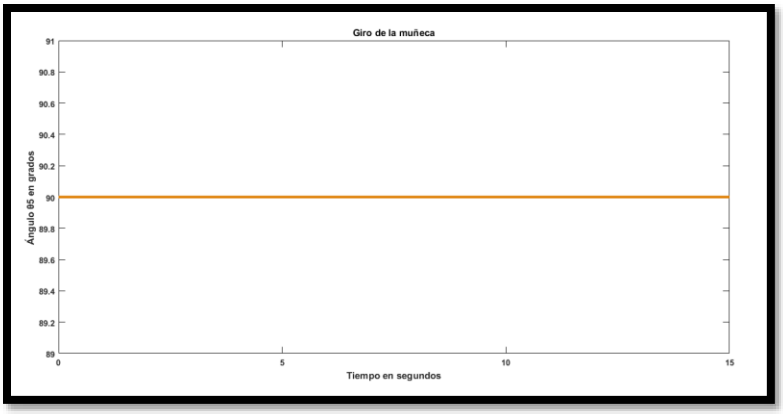


Figura 87.- Giro de la muñeca (θ_5), experimento 1

En la figura 88 se observa que el gripper permanece abierto los primeros 5 segundos, después se cierra para sujetar el objeto y una vez realizado el traslado se vuelve a abrir para colocar al objeto.

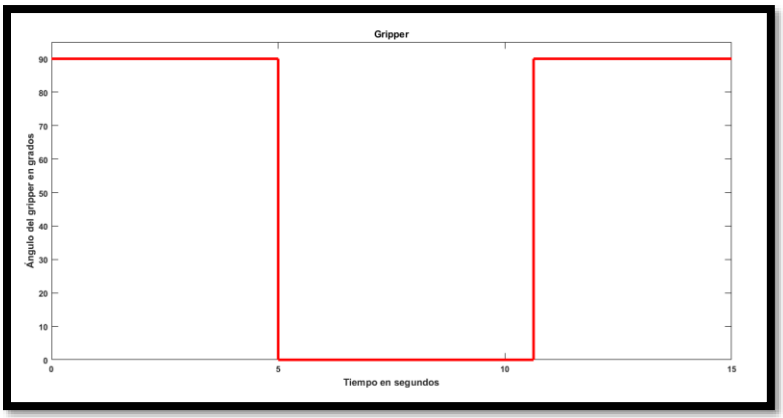


Figura 88.- Gripper (θ_6), abierto = 90° y cerrado = 0°, experimento 1

Los puntos que describen la transición en el tiempo de las coordenadas (X , Y y Z), el ángulo ϕ y el ángulo θ_6 se envían cada 10 milisegundos al bloque de cinemática inversa, donde cada milisegundo se envían al bloque "Coordinación del brazo serial" (ver apéndice C) las variables articulares calculadas para alcanzar el lugar geométrico con el efector final.

En el bloque “Coordinación del brazo serial” se realizan los ajustes necesarios para transmitir cada 20 milisegundos la velocidad, el identificador del servomotor y el ángulo calculado a la tarjeta de adquisición. Los resultados se muestran en las siguientes figuras:

El ángulo θ_1 que representa la base del brazo serial muestra su evolución angular en la figura 89, partiendo de 108.5 grados en el segundo 5 y llegando hasta 198.5 grados en el segundo 10.6.

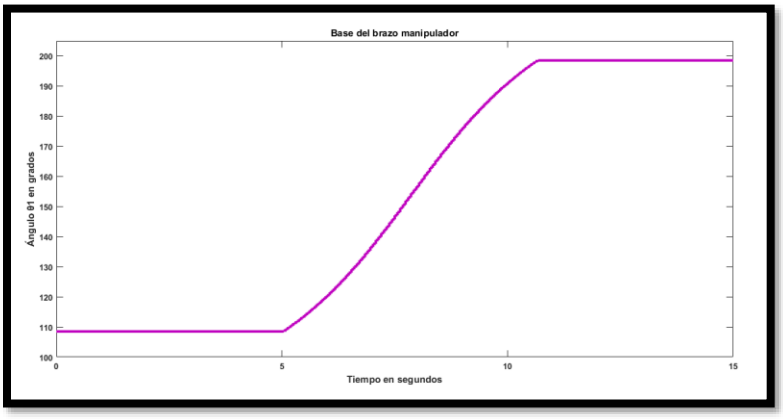


Figura 89.- Base del brazo serial (θ_1), experimento 1

El ángulo θ_2 que representa el hombro del brazo serial muestra su evolución angular en la figura 90, parte de 115.9 grados en el segundo 5 y llega a un pico de 197.1 grados en el segundo 7.8, para después llegar a 115.9 grados en el segundo 10.6.

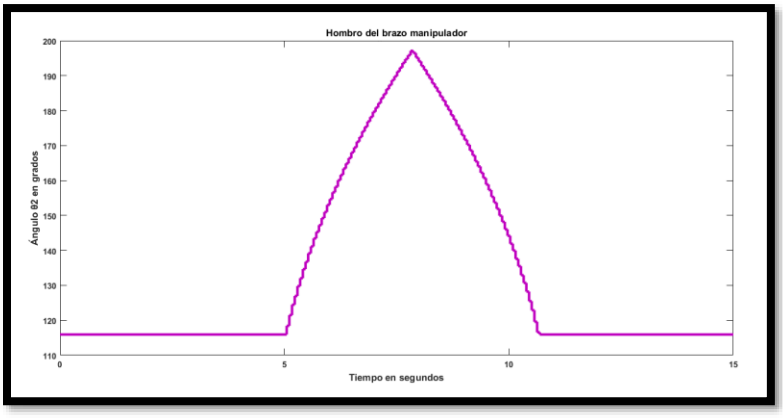


Figura 90.- Hombro del brazo serial (θ_2), experimento 1

El ángulo θ_3 que representa el codo del brazo serial muestra su evolución angular en la figura 91, parte de 119.1 grados en el segundo 5 y llega a un mínimo de 48.8 grados en el segundo 7.8, para después llegar nuevamente a 119.1 grados en el segundo 10.6.

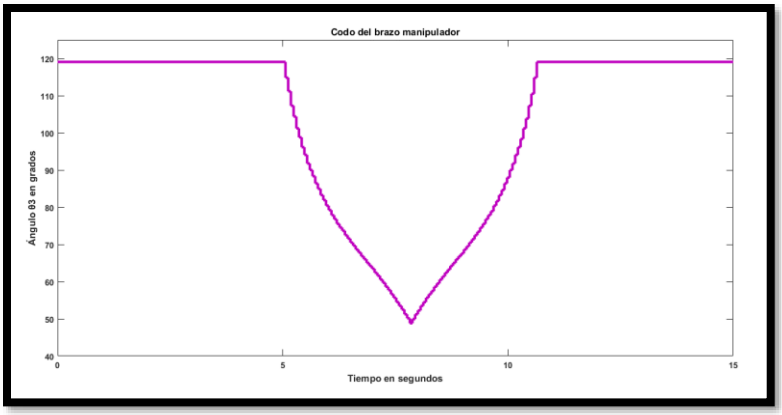


Figura 91.- Codo del brazo serial (θ_3), experimento 1

El ángulo θ_4 que representa la muñeca del brazo serial muestra su evolución angular en la figura 92, parte de 43 grados en el segundo 5 y llega a un pico de 71.7 grados en el segundo 7.8, para después llegar nuevamente a 43 grados en el segundo 10.6.

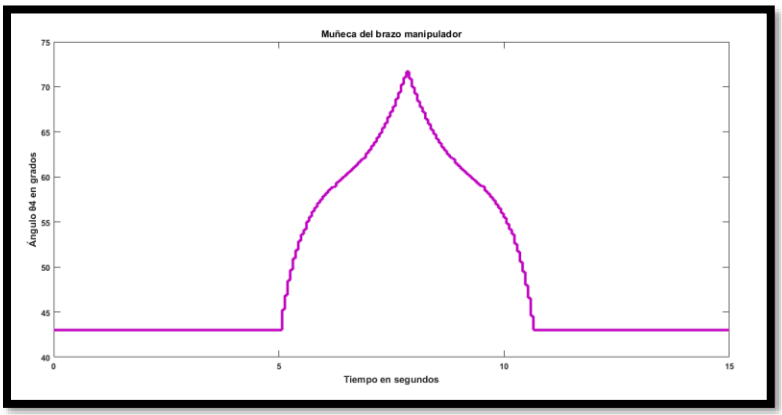


Figura 92.- Muñeca del brazo serial (θ_4), experimento 1

El ángulo θ_5 que representa el giro de la muñeca del brazo serial muestra su evolución angular en la figura 93, partiendo de 105 grados en el segundo 5 y llegando hasta 195 grados en el segundo 10.6.

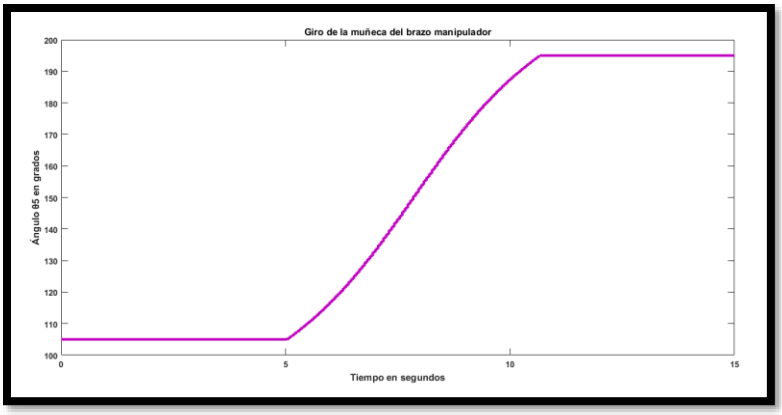


Figura 93.- Giro de la muñeca del brazo serial (θ_5), experimento 1

El ángulo θ_6 que representa la abertura del gripper del brazo serial muestra su evolución angular en la figura 94, donde 150 grados representa al gripper abierto del segundo 0 al 5; 60 grados representa al gripper cerrado del segundo 5 al 10.6, después de los 10.6 segundos este se vuelve a abrir.

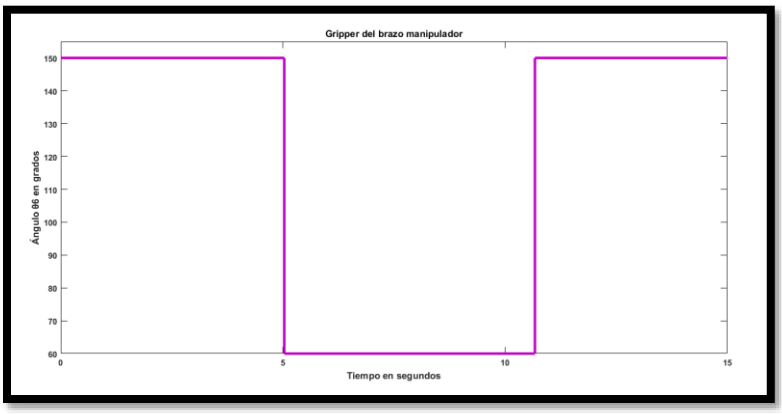


Figura 94.- Gripper del brazo serial (θ_6), abierto = 150° y cerrado = 60°, experimento 1

Las imágenes capturadas durante la ejecución del experimento con el manipulador móvil se muestran en la figura 95:

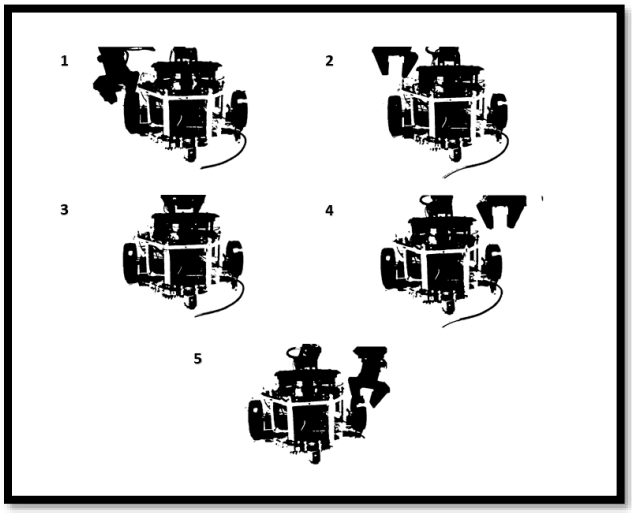


Figura 95.- Capturas del experimento uno

En este experimento no se retroalimentan las posiciones, temperaturas y otras variables, solo se observa el comportamiento del algoritmo programado cuando su movimiento se ejecuta en el plano "*yz*".

9.2 Movimiento vertical

Experimento 2: Ejecutar la tarea local descrita en el capítulo tres, la unión del punto inicial y final de referencia para el traslado del objeto debe describir una línea perpendicular al plano del horizonte (línea vertical). Este experimento radica en ejecutar cada tarea local por separado, la primera consiste en sujetar el objeto y la segunda en colocarlo.

- Para sujetar al objeto se desarrollan las etapas "*pick"* y "*pick back"*
- Para depositar el objeto en la meta se ejecutan las etapas *"place"* y *"return"*

El experimento se lleva a cabo dentro del espacio inteligente, por lo tanto, la plataforma móvil ejecuta campos potenciales y "docking and predocking" para llegar al objeto o la meta según sea el caso, de tal manera que no se ejecuta todo el algoritmo de coordinación. Las condiciones iniciales para el desarrollo del experimento son:

Coordenadas iniciales (configuración preferida del brazo serial):

$X_i = 16$ cm.
$Y_i = 0$ cm.
$Z_i = 28.3$ cm.

Coordenadas finales:

$X_f =$ Coordenada dada por el fiducial con ID número 2 o 3 en cm.
$Y_f =$ Coordenada dada por el fiducial con ID número 2 o 3 en cm.
$Z_f = 0.3$ cm.
$\theta_f =$ Ángulo dado por el fiducial con ID número 2 o 3 en radianes

El fiducial con ID número dos indica la posición y orientación del objeto, mientras que el fiducial con ID número tres proporciona la posición y orientación de la meta.

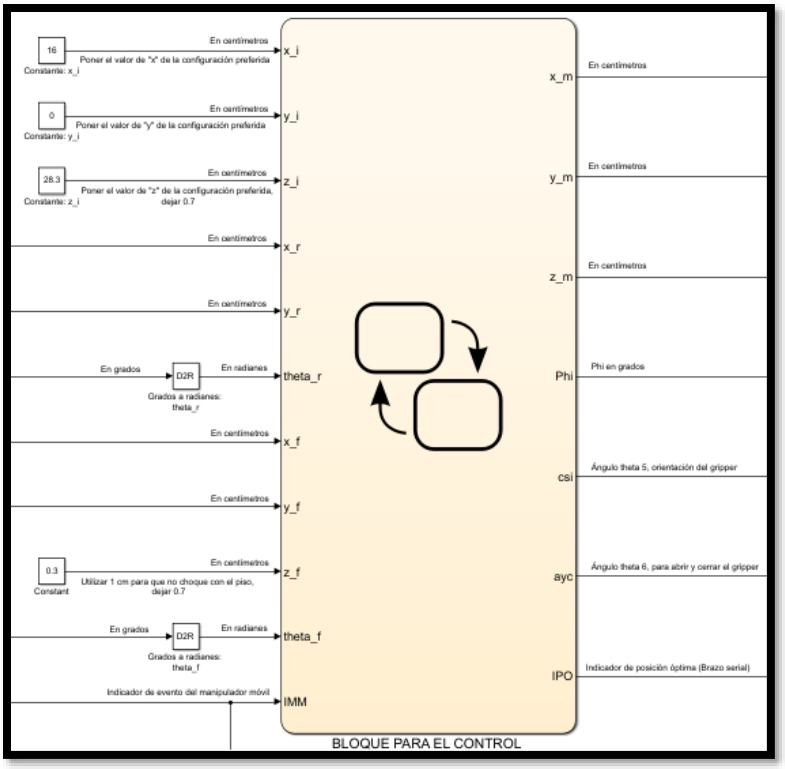


Figura 96.- Condiciones iniciales para realizar el experimento dos

Una vez que se indican las condiciones iniciales, se ejecuta la simulación para realizar los cálculos correspondientes y enviar los datos al manipulador móvil en tiempo real, las salidas generadas en el bloque de la figura 96 así como su variación a través del tiempo se muestran en las siguientes gráficas:

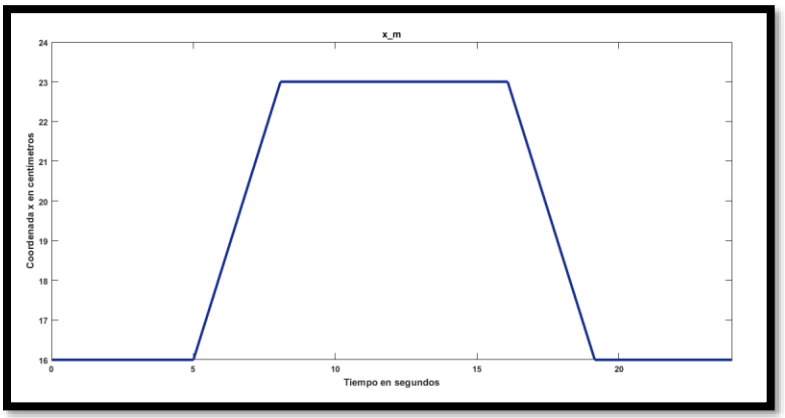


Figura 97.- Coordenada x_m , experimento 2

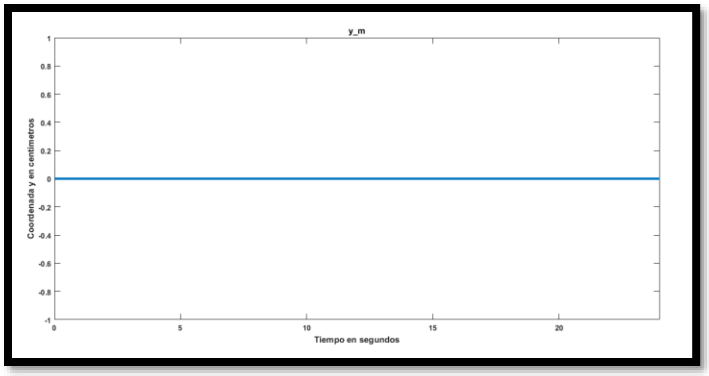


Figura 98.- Coordenada y_m , experimento 2

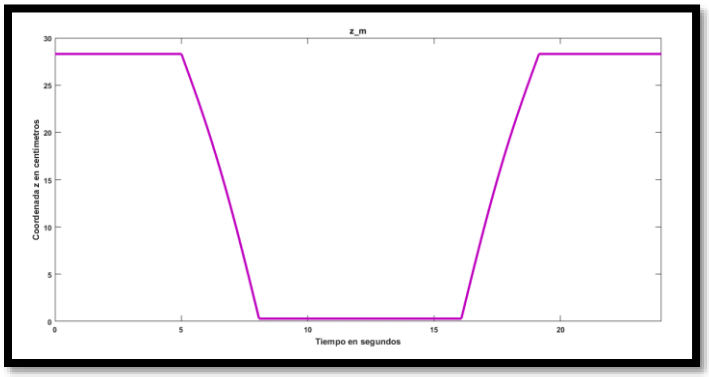


Figura 99.- Coordenada z_m , movimiento en el plano " *xz* "

Cada punto mostrado en la figura 97 (*eje x*), 98 (*eje y*) y 99 (*eje z*) es generado a partir de las ecuaciones 45, 46 y 47.

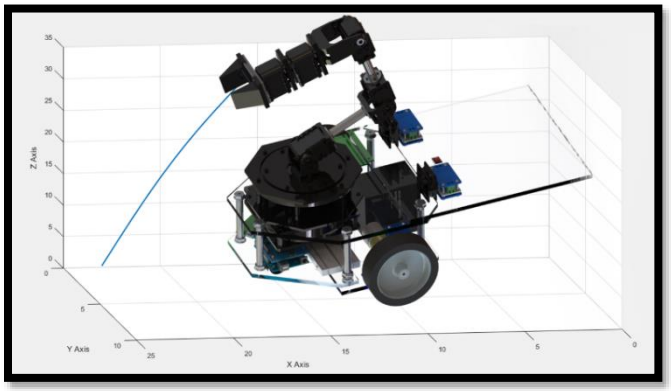


Figura 100.- Lugar geométrico descrito, movimiento en el plano " *xz* " vista 1

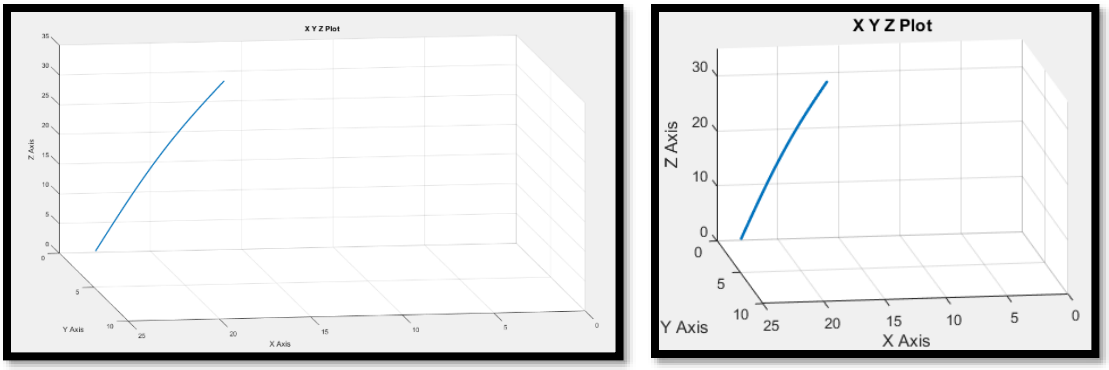


Figura 101.- Lugar geométrico descrito, movimiento en el plano " *xz* " vista 2

En el espacio tridimensional las coordenadas x_m, y_m y z_m mostradas en las figuras 97, 98 y 99 generan en su conjunto el lugar geométrico (espacio intuitivo) graficado en la figura 100 y 101.

En la figura 102 se muestra la variación del ángulo ϕ con el fin de alcanzar cada punto del lugar geométrico, de esta manera se logra orientar a la muñeca del brazo en un ángulo de 90° para que sujete o coloque el objeto.

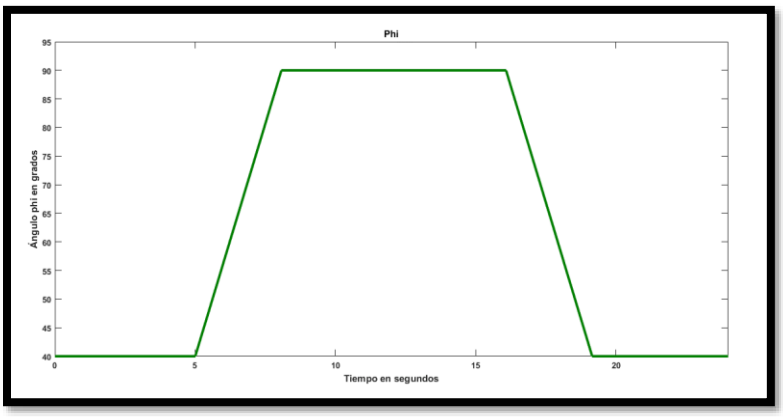


Figura 102.- Ángulo phi (ϕ), experimento 2

El ángulo de giro de la muñeca mostrado en la figura 103 se mantiene constante en 90°.

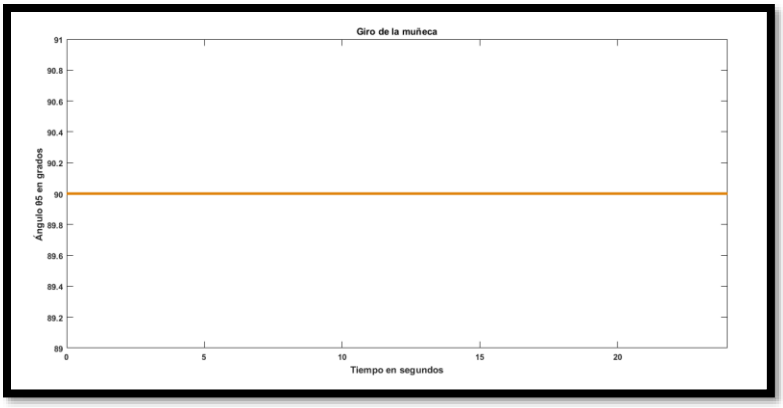


Figura 103.- Giro de la muñeca (θ_5), experimento 2

En la figura 104 se observa que el gripper permanece abierto los primeros 8 segundos, después de los 8 segundos se cierra para sujetar al objeto; mientras que para colocarlo la operación se realiza de manera inversa.

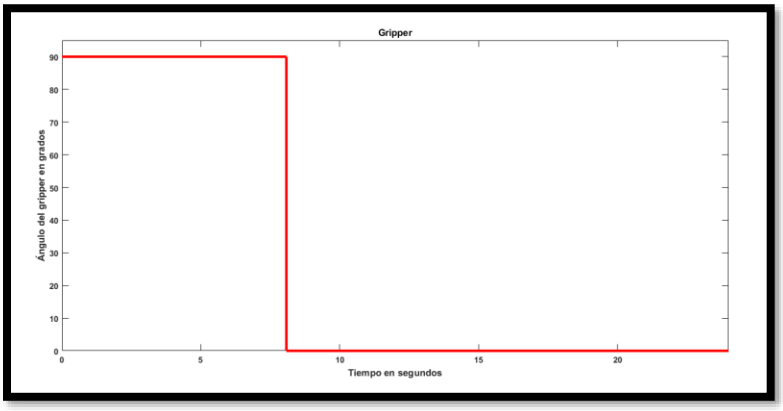


Figura 104.- Gripper (θ_6), abierto = 90° y cerrado = 0°, experimento 2

Se generan las mismas gráficas tanto en la tarea local de sujetar como en la de colocar al objeto, variando solo la gráfica del gripper como se indicó en el párrafo anterior.

Los puntos que describen la evolución en el tiempo de las coordenadas (x_m , y_m y z_m), el ángulo ϕ , el ángulo θ_5 y θ_6 se envían cada 10 milisegundos al bloque de cinemática inversa, donde cada milisegundo se envían al bloque "Coordinación del brazo serial"[2] las variables articulares calculadas para alcanzar el lugar geométrico con el efector final. En el bloque "Coordinación del brazo serial" se realizan los ajustes necesarios para transmitir cada 20 milisegundos la velocidad, el identificador del servomotor y el ángulo calculado a la tarjeta de adquisición.

En las siguientes gráficas se observan las variables articulares para realizar el movimiento de ida y regreso, el movimiento de "ida" se realiza antes de los 10 segundos, mientras que el de "regreso" después de los 15 segundos. Los ángulos mostrados en las figuras son iguales tanto en la tarea local para sujetar al objeto (etapa "pick" y "pick-back") como en la de colocarlo (etapa "place" y "return"), la diferencia solo recae en la articulación θ_6 . Los resultados se exponen en las siguientes figuras:

El ángulo θ_1 que representa la base del brazo serial muestra su evolución angular en la figura 105, manteniéndose constante en 153.5 grados durante el desarrollo de la tarea.

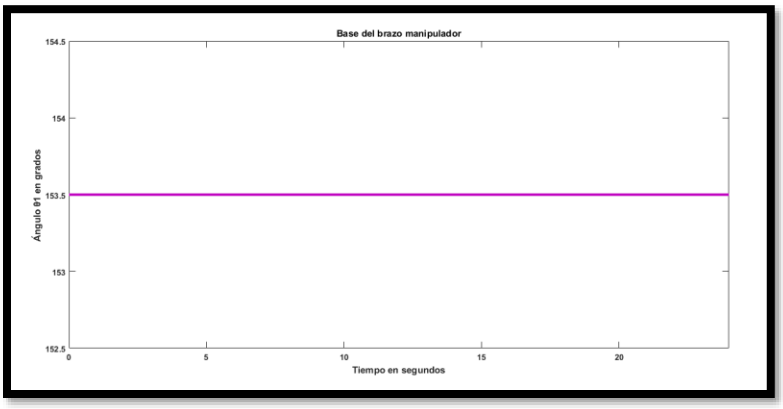


Figura 105.- Base del brazo serial (θ_1), experimento 2

[2] Ver apéndice C

El ángulo θ_2 que representa el hombro del brazo serial muestra su evolución angular en la figura 106, parte de 193.7 grados en el segundo 5 y llega a un mínimo de 89 grados en el segundo 8 (etapa "pick" cuando se sujeta al objeto o etapa "place" cuando se coloca en su objetivo). Después de 8 segundos parte de 89 grados a un máximo de 193.7 grados en el segundo 19.1 (etapa "pick-back" cuando se sujeta al objeto o etapa "return" cuando se coloca en su objetivo).

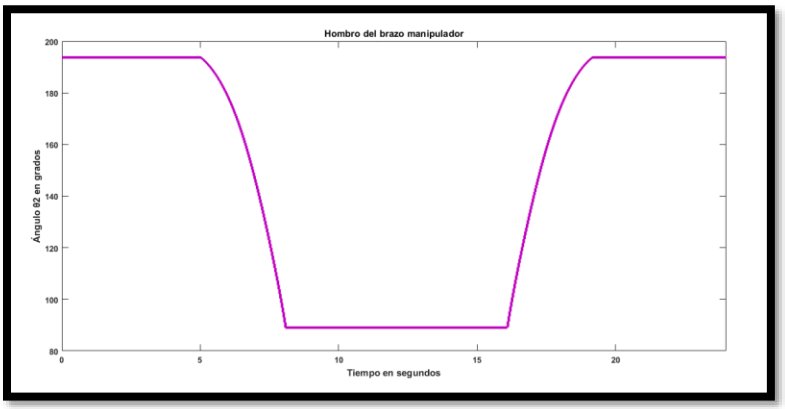


Figura 106.- Hombro del brazo serial (θ_2), experimento 2

El ángulo θ_3 que representa al codo del brazo serial muestra su evolución angular en la figura 107, parte de 67.8 grados en el segundo 5 describiendo una curva que llega a 99.1 grados en el segundo 8 (etapa "pick" cuando se sujeta al objeto o etapa "place" cuando se coloca en su objetivo). Después de 8 segundos parte de 99.1 grados describiendo una curva que llega a 67.8 grados en el segundo 19.1 (etapa "pick-back" cuando se sujeta al objeto o etapa "return" cuando se coloca en su objetivo).

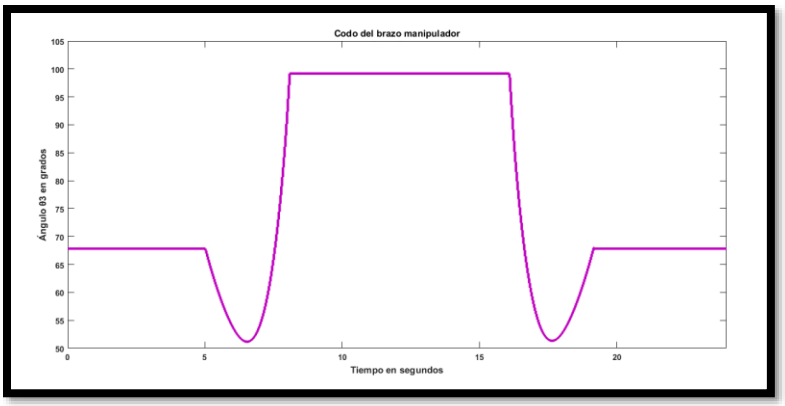


Figura 107.- Codo del brazo serial (θ_3), experimento 2

El ángulo θ_4 que representa a la muñeca del brazo serial muestra su evolución angular en la figura 108, parte de 66.45 grados en el segundo 5 describiendo una curva que llega a 89.7 grados en el segundo 8 (etapa "pick" cuando se sujeta al objeto o etapa "place" cuando se coloca en su objetivo). Después de 8 segundos parte de 89.7 grados describiendo una curva que llega a 66.45 grados en el segundo 19.1 (etapa "pick-back" cuando se sujeta al objeto o etapa "return" cuando se coloca en su objetivo).

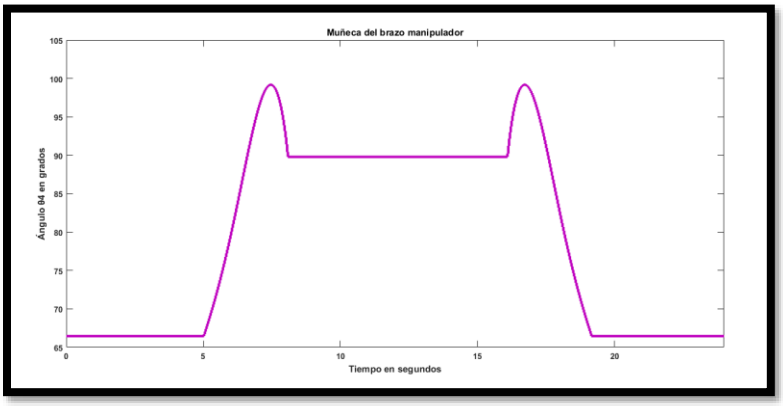


Figura 108.- Muñeca del brazo serial (θ_4), experimento 2

El ángulo θ_5 que representa el giro de la muñeca del brazo serial muestra su evolución angular en la figura 109, manteniéndose constante en 150 grados durante el desarrollo de la tarea.

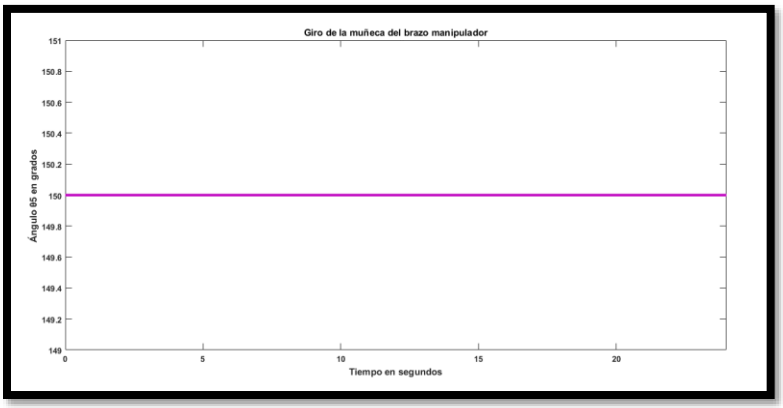


Figura 109.- Giro de la muñeca del brazo serial (θ_5), experimento 2

El ángulo θ_6 que representa la abertura del gripper del brazo serial muestra su evolución angular en la figura 110, donde 150 grados representa al gripper abierto del segundo 0 al 8; 60 grados representa al gripper cerrado del segundo 8 en adelante, ocurre lo inverso cuando se desarrolla la tarea de colocar al objeto en su destino.

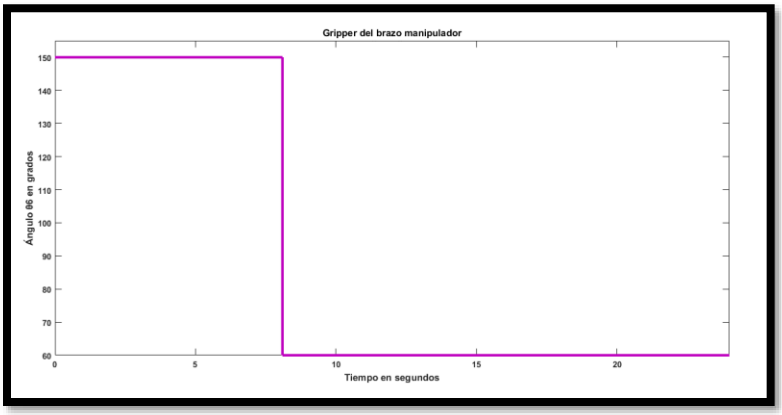


Figura 110.- Gripper del brazo serial (θ_6), evolución de la articulación para sujetar al objeto, donde 150° representa al gripper abierto y 60° al gripper cerrado. En la tarea local para colocar al objeto la transición de la articulación comienza en 60° (gripper cerrado) y termina en 150° (gripper abierto)

Las posiciones y velocidades enviadas a la tarjeta arduino para realizar la tarea local que consiste en sujetar al objeto se muestran en la figura 111, cada milisegundo se envía un ID, posición y velocidad, para controlar a cada servomotor.

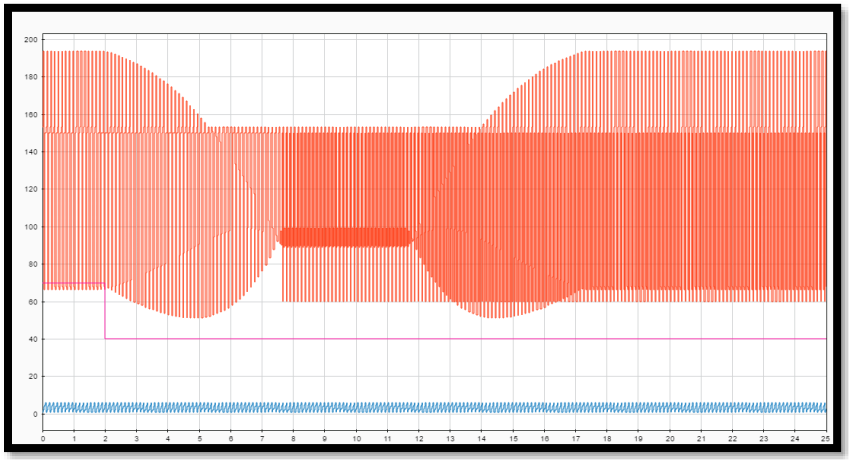


Figura 111.- ID (color azul), posiciones angulares (color rojo) y velocidades (color violeta) enviadas a cada servomotor Dynamixel para sujetar al objeto

Las variables medidas en cada servomotor y retroalimentadas desde la tarjeta arduino a Simulink vía inalámbrica durante la tarea que consiste en sujetar al objeto se muestran en las siguientes figuras:

El barrido de las posiciones angulares retroalimentadas mostradas en la figura 112 se realiza cada 6 milisegundos, iniciando desde el servomotor con ID 1 hasta el que tiene ID 6.

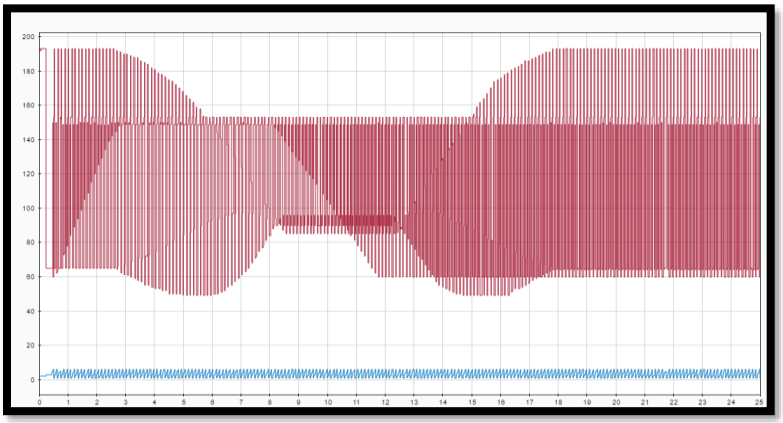


Figura 112.- ID (color azul) y posiciones angulares (color vino) retroalimentadas de cada servomotor en la tarea que consiste en sujetar al objeto

El barrido de las velocidades retroalimentadas en revoluciones por minuto mostradas en la figura 113 se realiza cada 6 milisegundos, iniciando desde el servomotor con ID 1 hasta el que tiene ID 6.

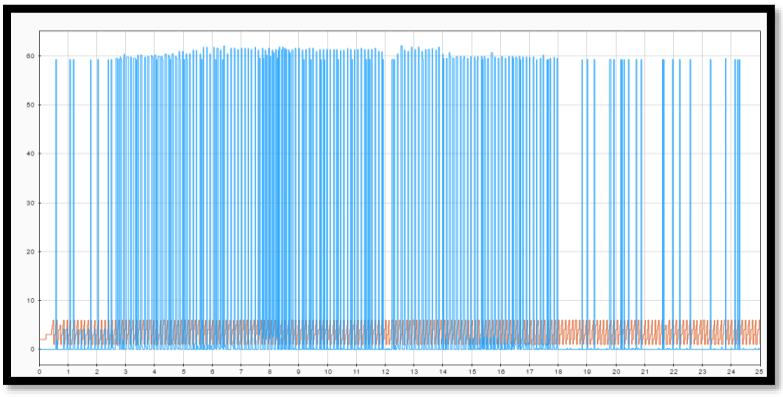


Figura 113.- ID (color rojo) y velocidades (color azul) retroalimentadas de cada servomotor en la tarea que consiste en sujetar al objeto

La retroalimentación de los porcentajes de carga mostrados en la figura 114 se realiza cada 6 milisegundos, iniciando el barrido desde el servomotor con ID 1 hasta el que tiene ID 6; dichos porcentajes tienen un rango de 0% hasta 100%.

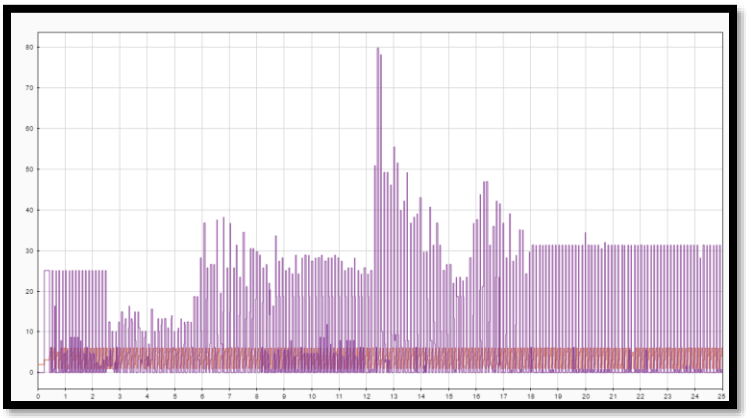


Figura 114.- ID (color rojo) y porcentajes de carga (color violeta) retroalimentados de cada servomotor en la tarea que consiste en sujetar al objeto

La retroalimentación de los sentidos de giro mostrados en la figura 115 se realiza cada 6 milisegundos, iniciando el barrido desde el servomotor con ID 1 hasta el que tiene ID 6; donde 1 representa el sentido horario y 2 el sentido antihorario.

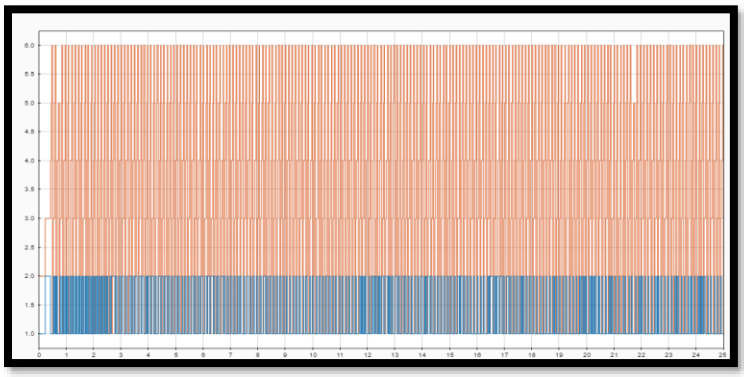


Figura 115.- ID (color rojo) y sentidos de giro (color azul) retroalimentados de cada servomotor en la tarea que consiste en sujetar al objeto

El barrido de los voltajes retroalimentados en volts mostrados en la figura 116 se realiza cada 6 milisegundos, iniciando desde el servomotor con ID 1 hasta el que tiene ID 6.

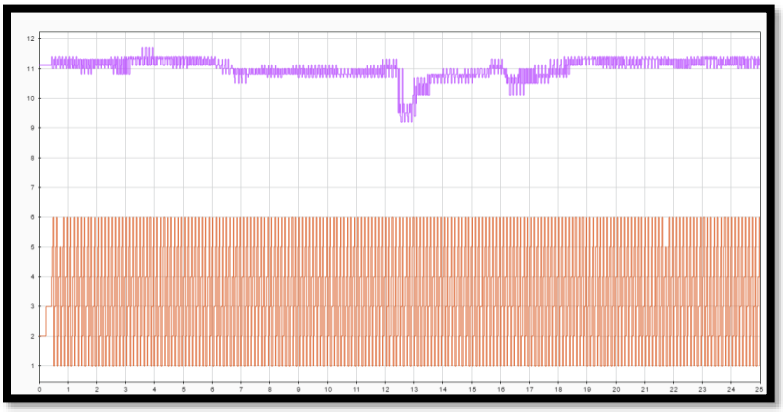


Figura 116.- ID (color rojo) y voltajes (color violeta) retroalimentados de cada servomotor en la tarea que consiste en sujetar al objeto

El barrido de las temperaturas retroalimentadas en grados Celsius mostradas en la figura 117 se realiza cada 6 milisegundos, iniciando desde el servomotor con ID 1 hasta el que tiene ID 6.

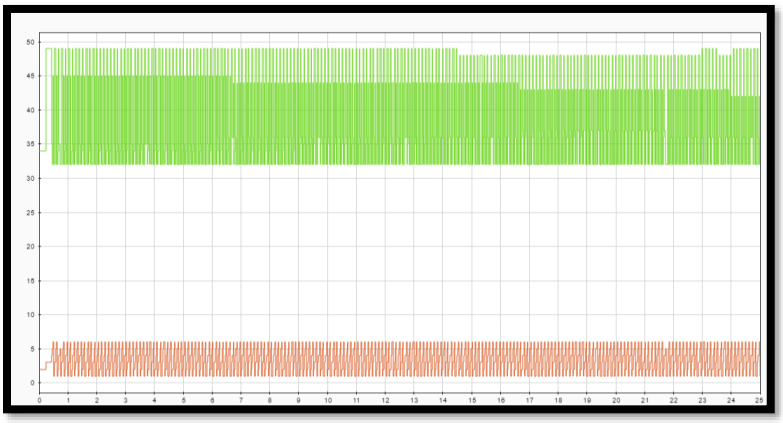


Figura 117.- ID (color rojo) y temperaturas (color verde) retroalimentadas de cada servomotor en la tarea que consiste en sujetar al objeto

Las posiciones y velocidades enviadas a la tarjeta arduino para realizar la tarea local que consiste en colocar al objeto se muestran en la figura 118, cada milisegundo se envía un ID, posición y velocidad, para controlar a cada servomotor.

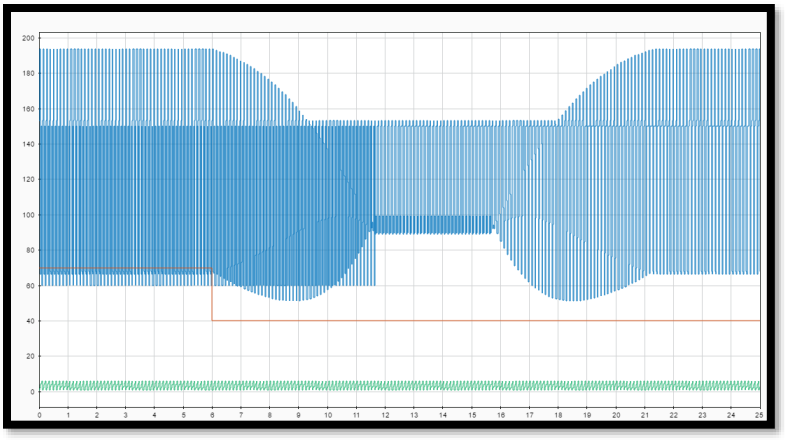


Figura 118.- ID (color azul turquesa), posiciones angulares (color azul fuerte) y velocidades (color anaranjado) enviadas a cada servomotor Dynamixel para colocar al objeto

Las variables medidas en cada servomotor y retroalimentadas desde la tarjeta arduino a Simulink vía inalámbrica durante la tarea que consiste en colocar el objeto se muestran en las siguientes figuras:

El barrido de las posiciones angulares retroalimentadas mostradas en la figura 119 se realiza cada 6 milisegundos, iniciando desde el servomotor con ID 1 hasta el que tiene ID 6.

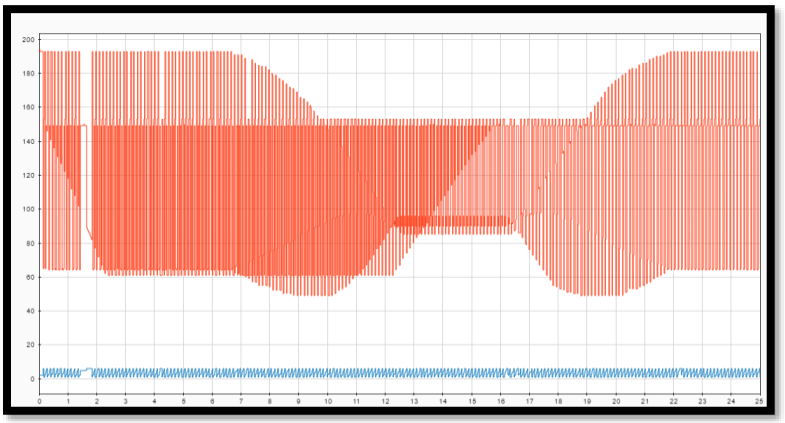


Figura 119.- ID (color azul) y posiciones angulares (color rojo) retroalimentadas de cada servomotor en la tarea que consiste en colocar al objeto

El barrido de las velocidades retroalimentadas en revoluciones por minuto mostradas en la figura 120 se realiza cada 6 milisegundos, iniciando desde el servomotor con ID 1 hasta el que tiene ID 6.

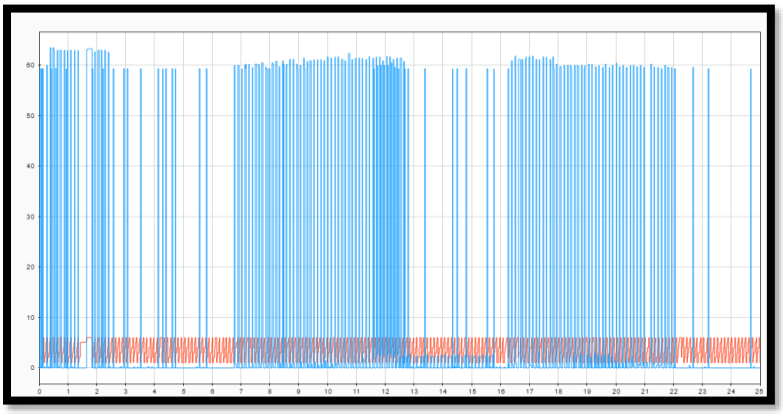


Figura 120.- ID (color rojo) y velocidades (color azul cielo) retroalimentadas de cada servomotor en la tarea que consiste en colocar al objeto

La retroalimentación de los porcentajes de carga mostrados en la figura 121 se realiza cada 6 milisegundos, iniciando el barrido desde el servomotor con ID 1 hasta el que tiene ID 6; dichos porcentajes tienen un rango de 0% hasta 100%.

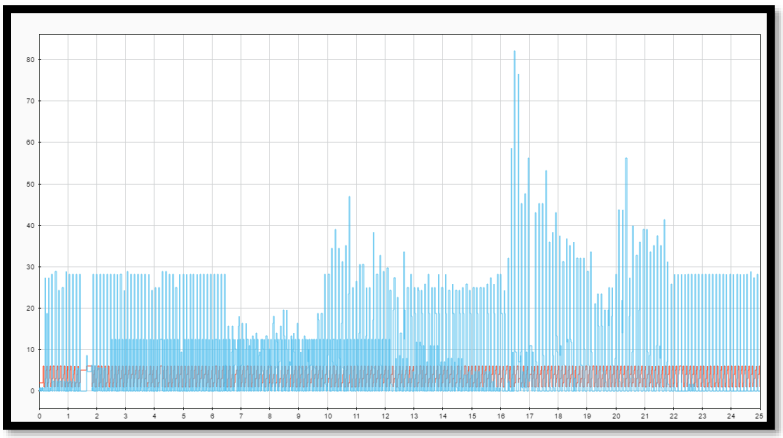


Figura 121.- ID (color rojo) y porcentajes de carga (color azul suave) retroalimentados de cada servomotor en la tarea que consiste en colocar al objeto

La retroalimentación de los sentidos de giro mostrados en la figura 122 se realiza cada 6 milisegundos, iniciando el barrido desde el servomotor con ID 1 hasta el que tiene ID 6; donde 1 representa el sentido horario y 2 el sentido antihorario.

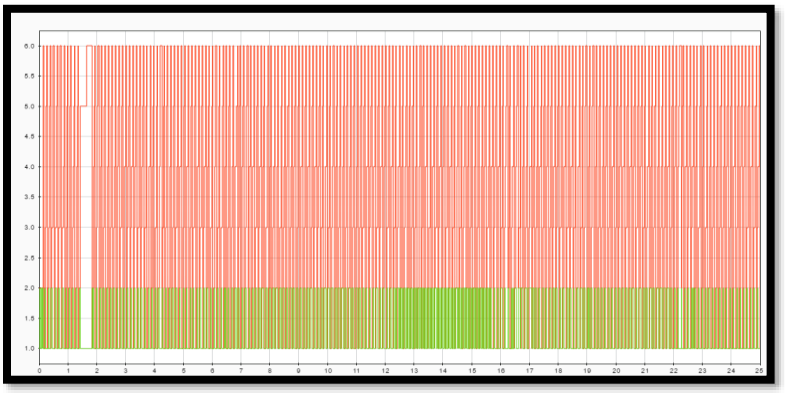


Figura 122.- ID (color rojo) y sentidos de giro (color verde fuerte) retroalimentados de cada servomotor en la tarea que consiste en colocar al objeto

El barrido de los voltajes retroalimentados en volts mostrados en la figura 123 se realiza cada 6 milisegundos, iniciando desde el servomotor con ID 1 hasta el que tiene ID 6.

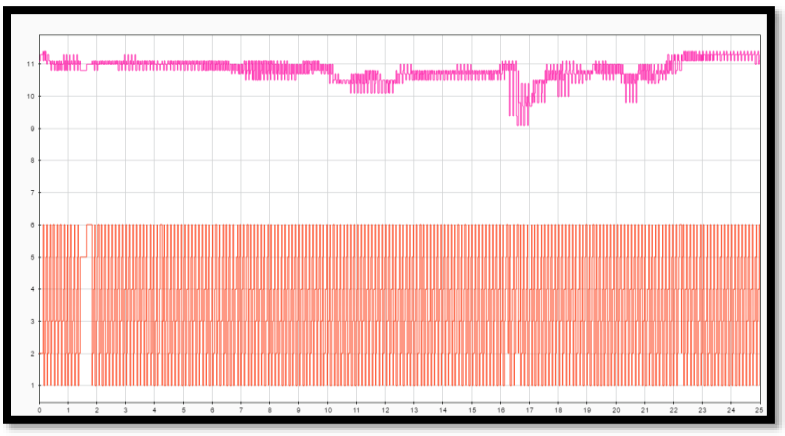


Figura 123.- ID (color rojo) y voltajes (color rosa fuerte) retroalimentados de cada servomotor en la tarea que consiste en colocar al objeto

El barrido de las temperaturas retroalimentadas en grados Celsius mostradas en la figura 124 se realiza cada 6 milisegundos, iniciando desde el servomotor con ID 1 hasta el que tiene ID 6.

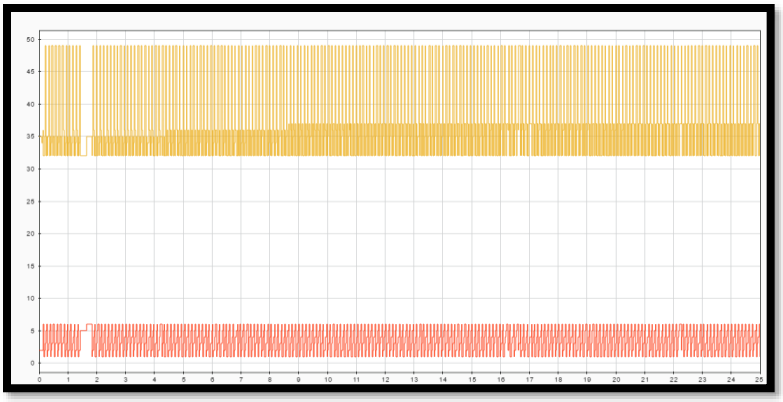


Figura 124.- ID (color rojo) y temperaturas (color amarillo fuerte) retroalimentadas de cada servomotor en la tarea que consiste en colocar al objeto

Las imágenes captadas durante la ejecución del experimento con el manipulador móvil dentro del espacio inteligente para desarrollar la tarea local de sujetar al objeto son:

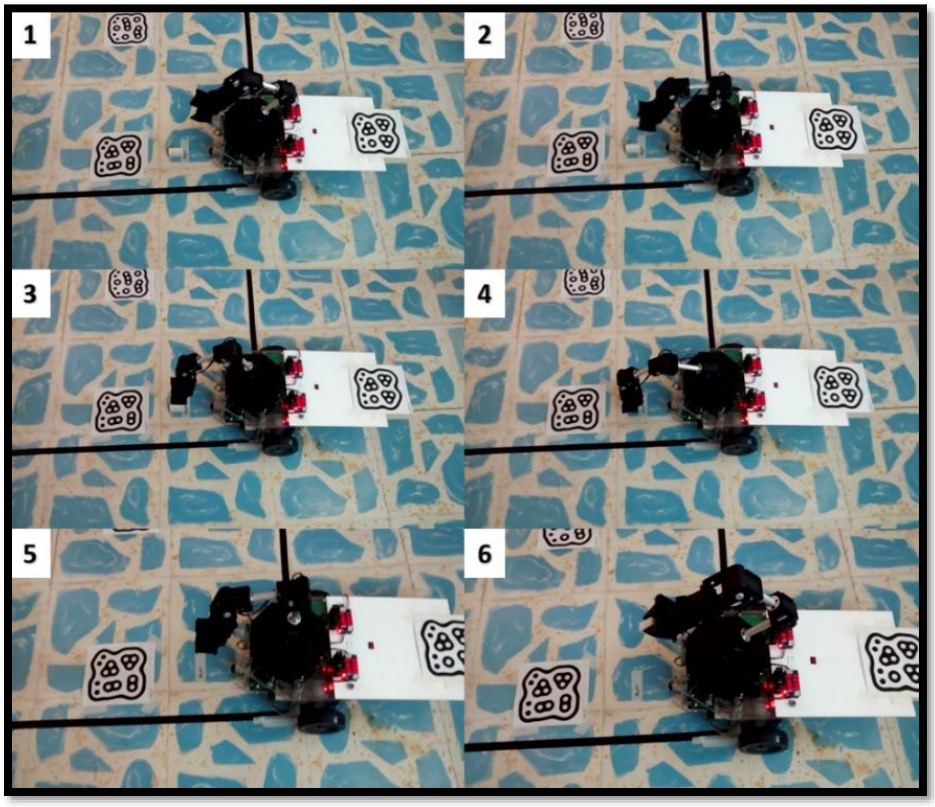


Figura 125.- Capturas del experimento dos en el que se desarrolla la tarea que consiste en sujetar al objeto

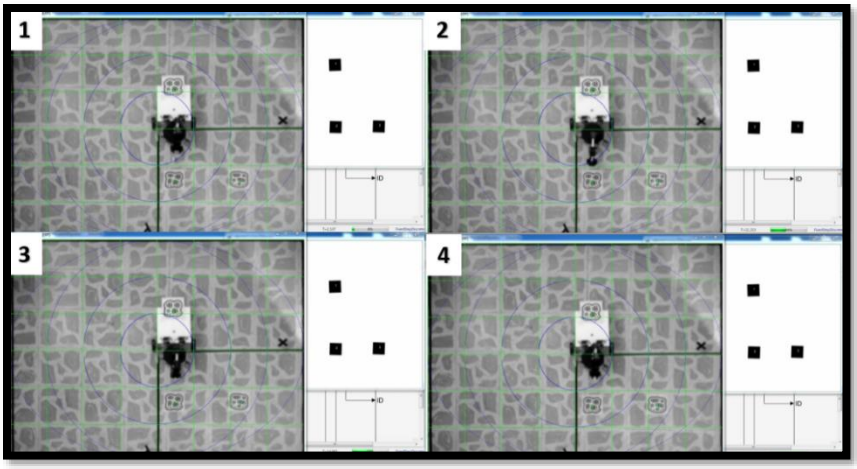


Figura 126.- Capturas del experimento dos en el que se desarrolla la tarea que consiste en sujetar al objeto dentro del espacio inteligente

Las imágenes captadas durante la ejecución del experimento con el manipulador móvil dentro del espacio inteligente para desarrollar la tarea local de colocar al objeto en su destino o meta son:

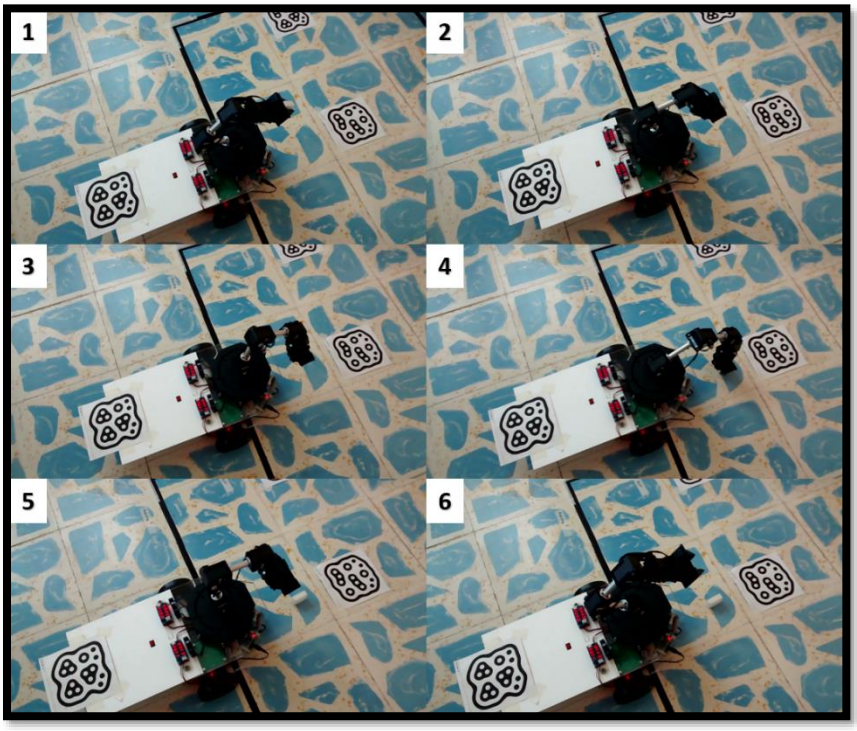


Figura 127.- Capturas del experimento dos en el que se desarrolla la tarea que consiste en colocar al objeto

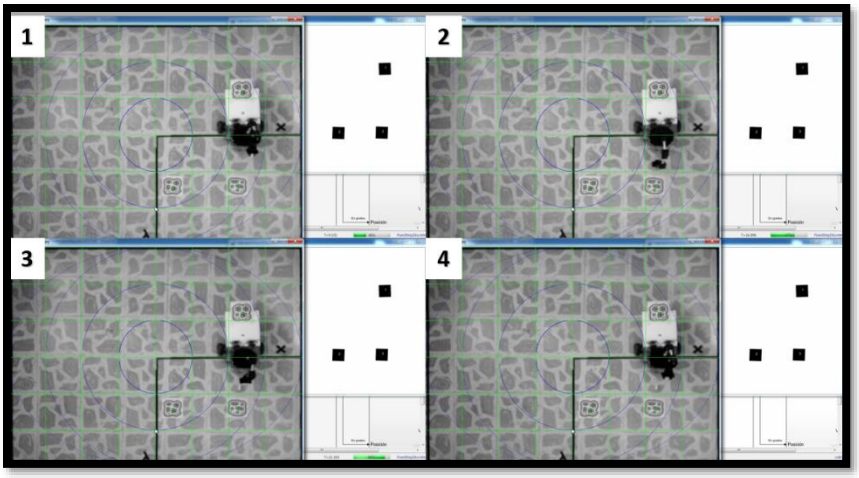


Figura 128.- Capturas del experimento dos en el que se desarrolla la tarea que consiste en colocar al objeto dentro del espacio inteligente

9.3 Traslado del objeto dentro del espacio inteligente

Experimento 3: Ejecutar la tarea global descrita en el capítulo tres.

En este experimento se coordinan las dos tareas locales ejecutadas en el experimento número dos, ejecutándose todas las etapas de la tarea "pick and place" (*"pick"* - *"pick back"* - *"place"* - *"return"*).

El experimento se lleva a cabo dentro del espacio inteligente, por lo tanto, la plataforma móvil ejecuta campos potenciales y "docking and predocking" para llegar al objeto y la meta de manera coordinada. Las condiciones iniciales para el desarrollo del experimento son:

Coordenadas iniciales (configuración preferida del brazo serial):

$X_i = 16$ cm.
$Y_i = 0$ cm.
$Z_i = 28.3$ cm.

Coordenadas finales:

$X_f =$ Coordenada dada por el fiducial con ID número 2 o 3 en cm.
$Y_f =$ Coordenada dada por el fiducial con ID número 2 o 3 en cm.
$Z_f = 0.3$ cm.
$\theta_f =$ Ángulo dado por el fiducial con ID número 2 o 3 en radianes

El fiducial con ID número dos indica la posición y orientación del objeto, mientras que el fiducial con ID número tres proporciona la posición y orientación de la meta.

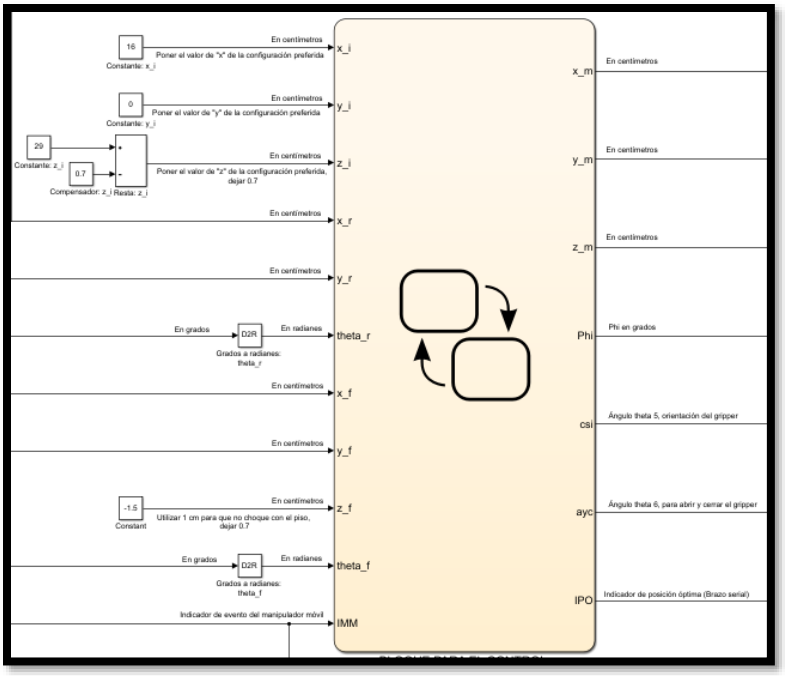


Figura 129.- Condiciones iniciales para realizar el experimento tres

Una vez que se indican las condiciones iniciales, se ejecuta la simulación para realizar los cálculos correspondientes y enviar los datos al manipulador móvil en tiempo real, en este caso se ejecutan todos los bloques programados en Simulink para implementar el algoritmo que coordina a la tarea global[3], las principales salidas generadas en los bloques, así como su evolución a través del tiempo se muestran en las siguientes figuras:

El ángulo θ_1 que representa la base del brazo serial muestra su evolución angular en la figura 130, en seguida se describe cada etapa:

- Etapa “pick”: parte de 153.5 grados en el segundo 24.5 y llega a un mínimo de 146.3 grados en el segundo 30.
- Etapa “pick back”: Después de 4 segundos parte de 146.3 grados a un máximo de 153.5 grados en el segundo 39.7.

[3] Ver apéndice C

- Etapa “place”: Después de 28.6 segundos parte de 153.5 grados a un máximo de 160.3 grados en el segundo 73.8.
- Etapa “return”: Después de 4.08 segundos parte de 160.3 grados a un mínimo de 153.5 grados en el segundo 83.4.

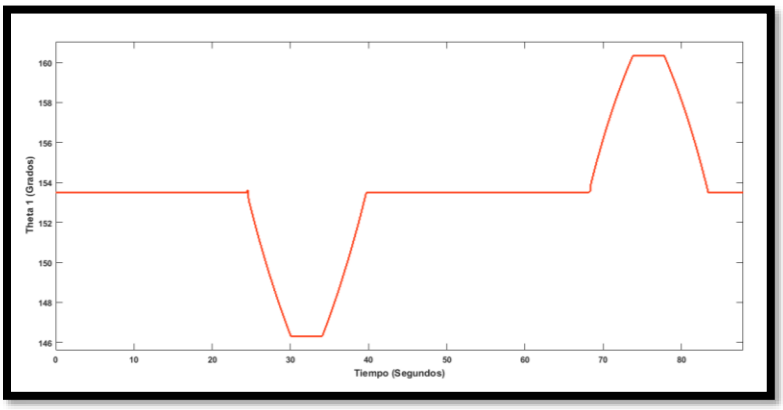


Figura 130.- Base del brazo serial (θ_1), experimento 3

El ángulo θ_2 que representa al hombro del brazo serial muestra su evolución angular en la figura 131, en seguida se describe cada etapa:

- Etapa “pick”: parte de 193.7 grados en el segundo 24.5 y llega a un mínimo de 94.4 grados en el segundo 30.
- Etapa “pick back”: Después de 4 segundos parte de 94.4 grados a un máximo de 193.7 grados en el segundo 39.7.
- Etapa “place”: Después de 28.6 segundos parte de 193.7 grados a un mínimo de 90.4 grados en el segundo 73.8.
- Etapa “return”: Después de 4.08 segundos parte de 90.4 grados a un máximo de 193.7 grados en el segundo 83.4.

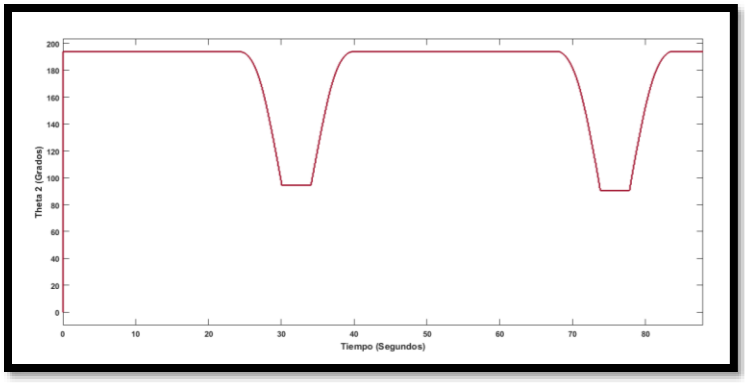


Figura 131.- Hombro del brazo serial (θ_2), experimento 3

El ángulo θ_3 que representa al codo del brazo serial muestra su evolución angular en la figura 132, en seguida se describe cada etapa:

- Etapa “pick”: parte de 67.8 grados en el segundo 24.5 describiendo una curva que llega a 76.1 grados en el segundo 30.
- Etapa “pick back”: Después de 4 segundos parte de 76.1 grados describiendo una curva que llega a 67.8 grados en el segundo 39.7.
- Etapa “place”: Después de 28.6 segundos parte de 67.8 grados describiendo una curva que llega a 85.6 grados en el segundo 73.8.
- Etapa “return”: Después de 4 segundos parte de 85.6 grados describiendo una curva que llega a 67.8 grados en el segundo 83.4.

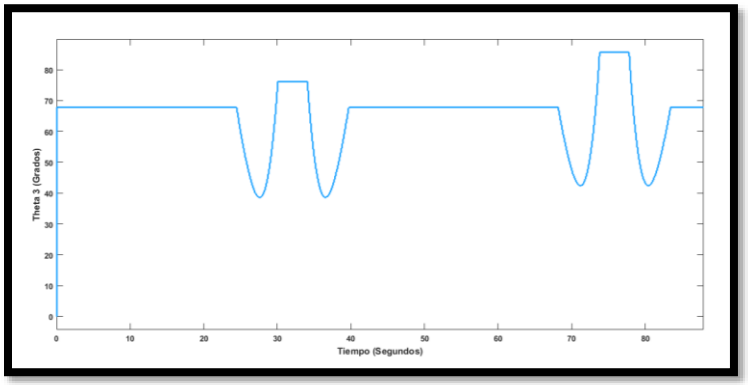


Figura 132.- Codo del brazo serial (θ_3), experimento 3

El ángulo θ_4 que representa a la muñeca del brazo serial muestra su evolución angular en la figura 133, en seguida se describe cada etapa:

- Etapa “pick”: parte de 66.4 grados en el segundo 24.5 describiendo una curva que llega a 107.5 grados en el segundo 30.
- Etapa “pick back”: Después de 4 segundos parte de 107.5 grados describiendo una curva que llega a 66.4 grados en el segundo 39.7.
- Etapa “place”: Después de 28.6 segundos parte de 66.4 grados describiendo una curva que llega a 101.9 grados en el segundo 73.8.
- Etapa “return”: Después de 4 segundos parte de 101.9 grados describiendo una curva que llega a 66.4 grados en el segundo 83.4.

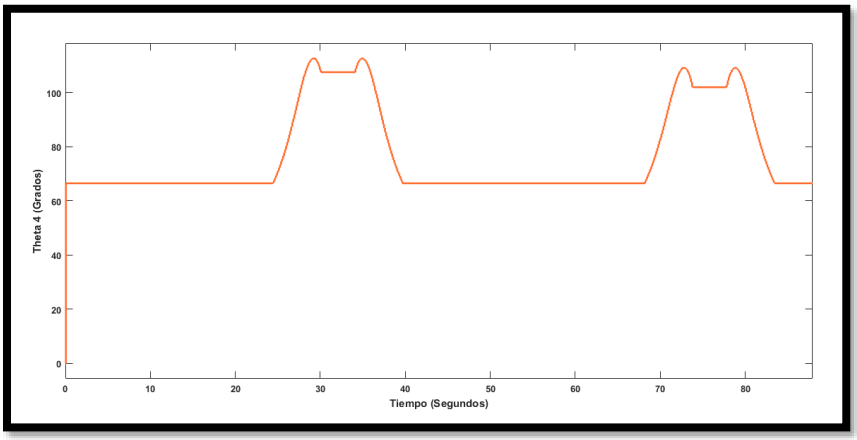


Figura 133.- Muñeca del brazo serial (θ_4), experimento 3

El ángulo θ_5 que representa el giro de la muñeca del brazo serial muestra su evolución angular en la figura 134, presenta pequeñas variaciones angulares de aproximadamente 0.8 grados, manteniéndose prácticamente constante en 150 grados durante el desarrollo de todas las etapas.

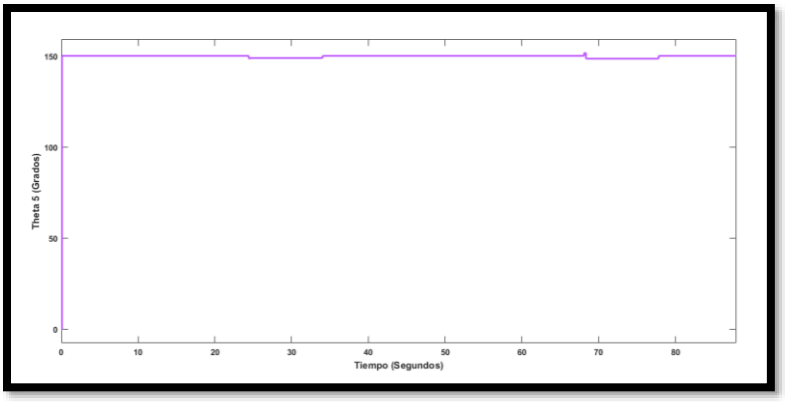


Figura 134.- Giro de la muñeca del brazo serial (θ_5), experimento 3

El ángulo θ_6 que representa la abertura del gripper del brazo serial muestra su evolución angular en la figura 135.

- 60 grados representa al gripper cerrado del segundo 0 al 24.4
- 150 grados representa al gripper abierto del segundo 24.4 al 30.1
- 60 grados representa al gripper cerrado del segundo 30.1 al 73.7
- 150 grados representa al gripper abierto del segundo 73.7 en adelante

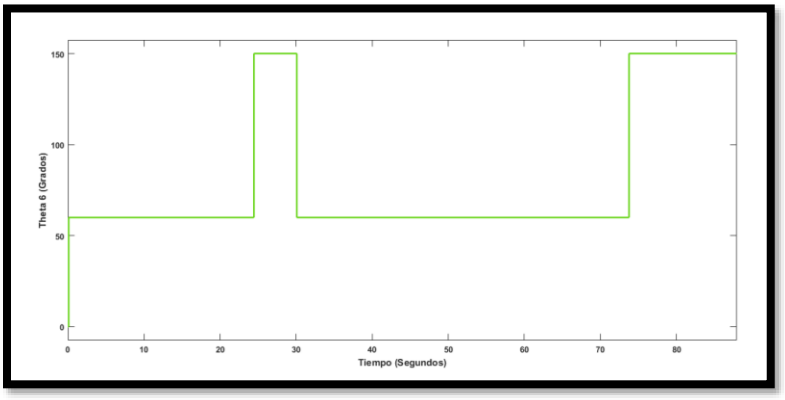


Figura 135.- Gripper del brazo serial (θ_6), abierto = 150° y cerrado = 60°, experimento 3

En las figuras 130 a la 135 se observa que la etapa “pick” se ejecuta aproximadamente del segundo 25 al 30, la etapa “pick back” del segundo 34 al 39, la etapa “place” del segundo 68 al 73 y la etapa “return” del segundo 77 al 82.

Las velocidades en radianes por segundo de la llanta derecha e izquierda se muestran en la figura 136 y 137.

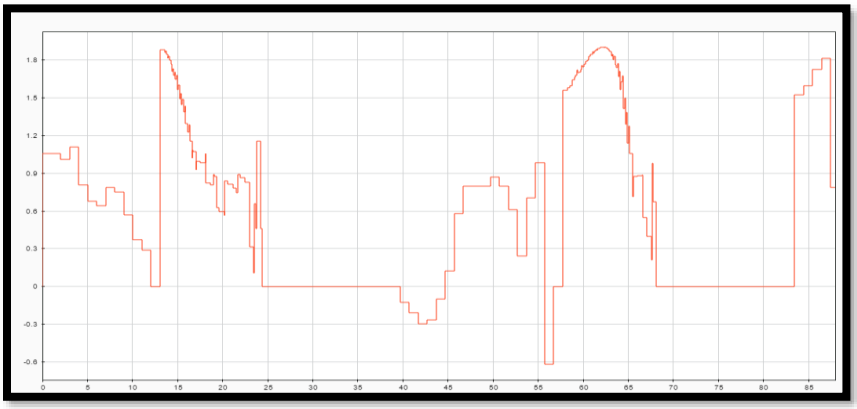


Figura 136.- Velocidad en rad/s, llanta derecha de la plataforma móvil

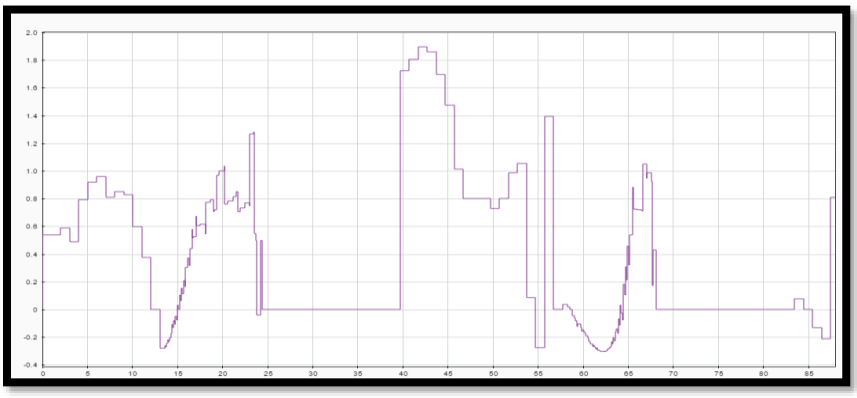


Figura 137.- Velocidad en rad/s, llanta izquierda de la plataforma móvil

Las posiciones y velocidades enviadas a la tarjeta arduino para realizar la tarea que consiste en sujetar y colocar al objeto se muestran en la figura 138, cada milisegundo se envía un ID, posición y velocidad, para controlar a cada servomotor.

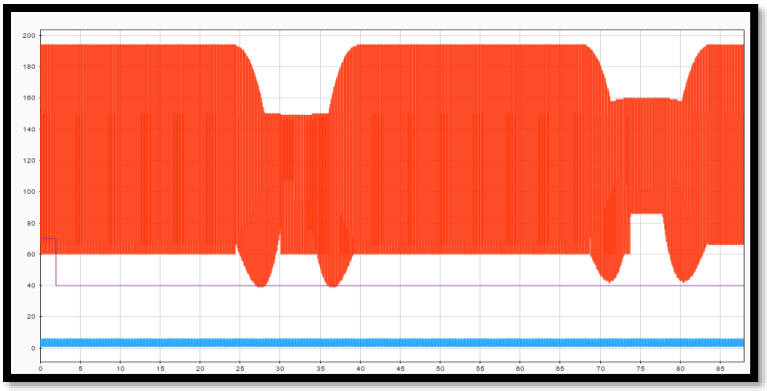


Figura 138.- ID (color azul), posiciones angulares (color rojo) y velocidades (color violeta) enviadas a cada servomotor Dynamixel para sujetar y colocar al objeto

La retroalimentación de los ID, posiciones angulares, velocidades, porcentajes de carga, sentidos de giro, voltajes y temperaturas de cada servomotor son los mismos que se muestran a partir de la figura 112 hasta la 117 para la tarea de sujetar al objeto que comprende las etapas “pick” y “pick-back”.

La retroalimentación de las variables de cada servomotor durante la tarea que consiste en colocar al objeto en su destino y que se desarrolla en dos etapas: “place” y “return”, son iguales a las mostradas desde la figura 119 a la 124.

Los resultados del mapeo que se realiza de rad/s a la velocidad MD25 se observan en la figura 139 y 140, es importante señalar que estas velocidades son las que se envían durante el experimento a cada motorreductor y son resultado de aplicar la técnica de campos potenciales y “docking and predocking” para alcanzar al objeto y el destino, el valor 128 representa cuando la plataforma se detiene para ejecutar una parte de la tarea “pick and place”.

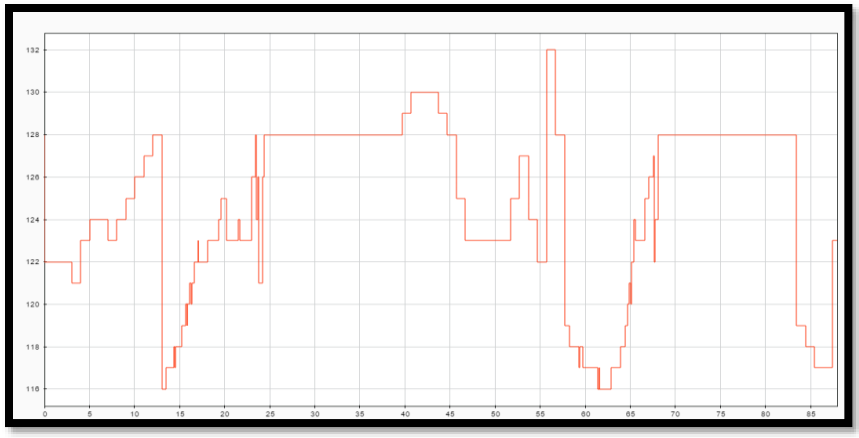


Figura 139.- Resultado del mapeo de rad/s a velocidad MD25 para la llanta derecha, del segundo 24 al 39 se sujeta el objeto, mientras que del segundo 68 al 84 se coloca

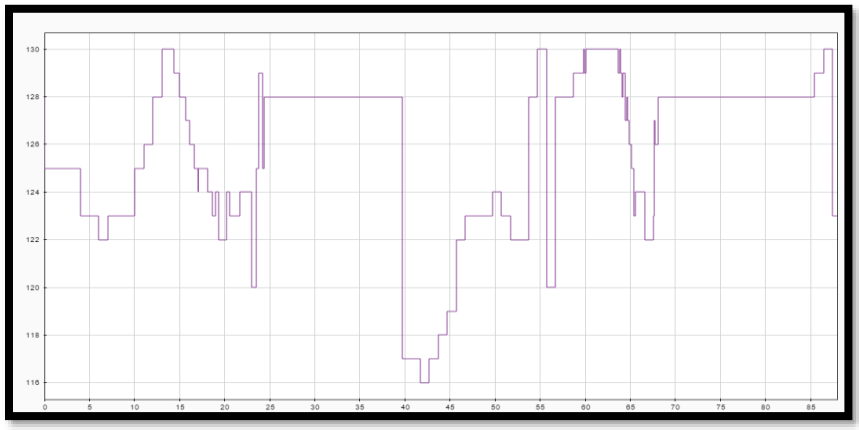


Figura 140.- Resultado del mapeo de rad/s a velocidad MD25 para la llanta izquierda del segundo 24 al 39 se sujeta el objeto, mientras que del segundo 68 al 84 se coloca

Las imágenes capturadas durante la ejecución del experimento con el manipulador móvil dentro del espacio inteligente para desarrollar la tarea global son:

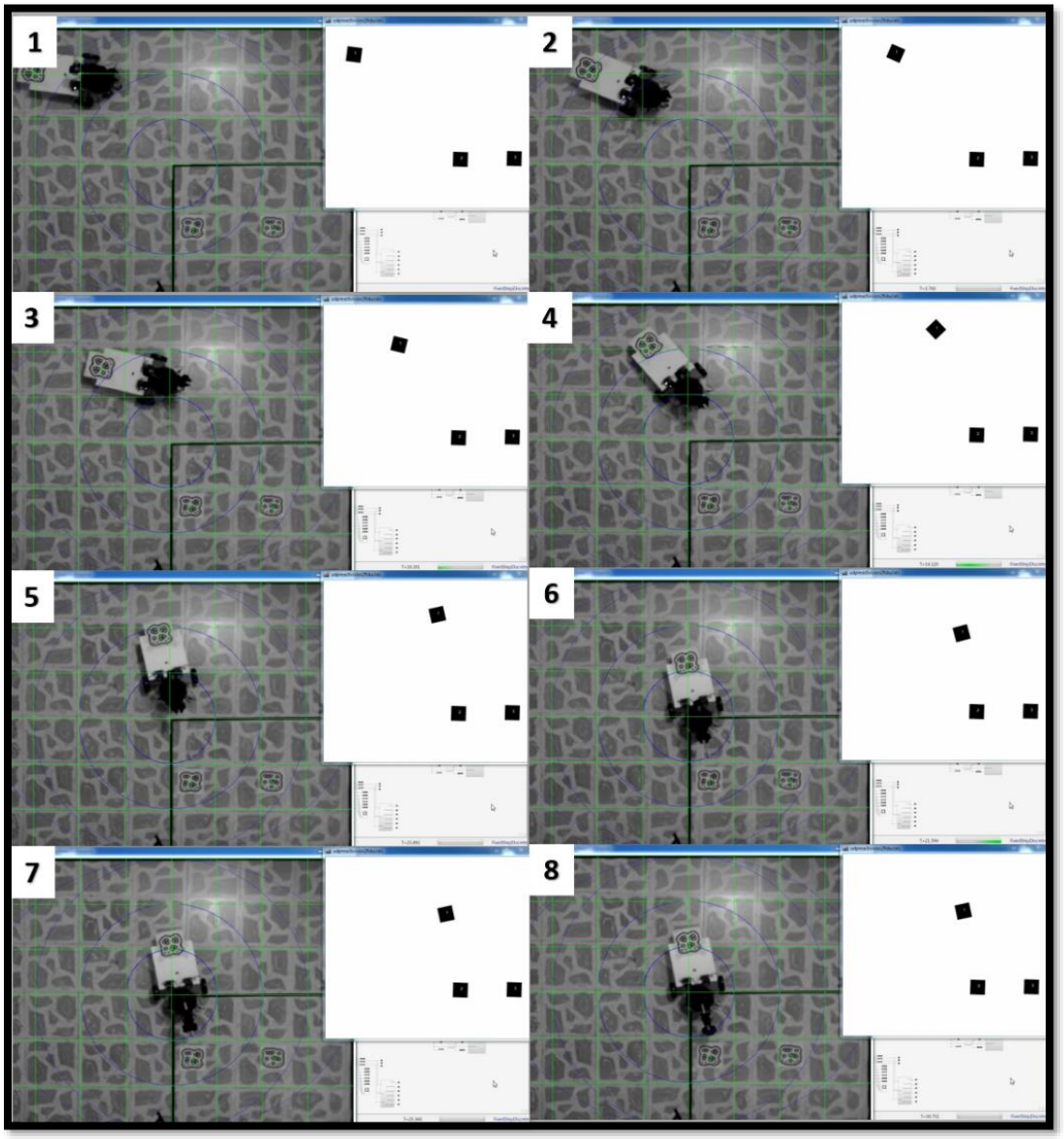


Figura 141.- Capturas del experimento número tres, en el que se ejecuta el algoritmo coordinante para el traslado del objeto dentro del espacio inteligente, parte 1

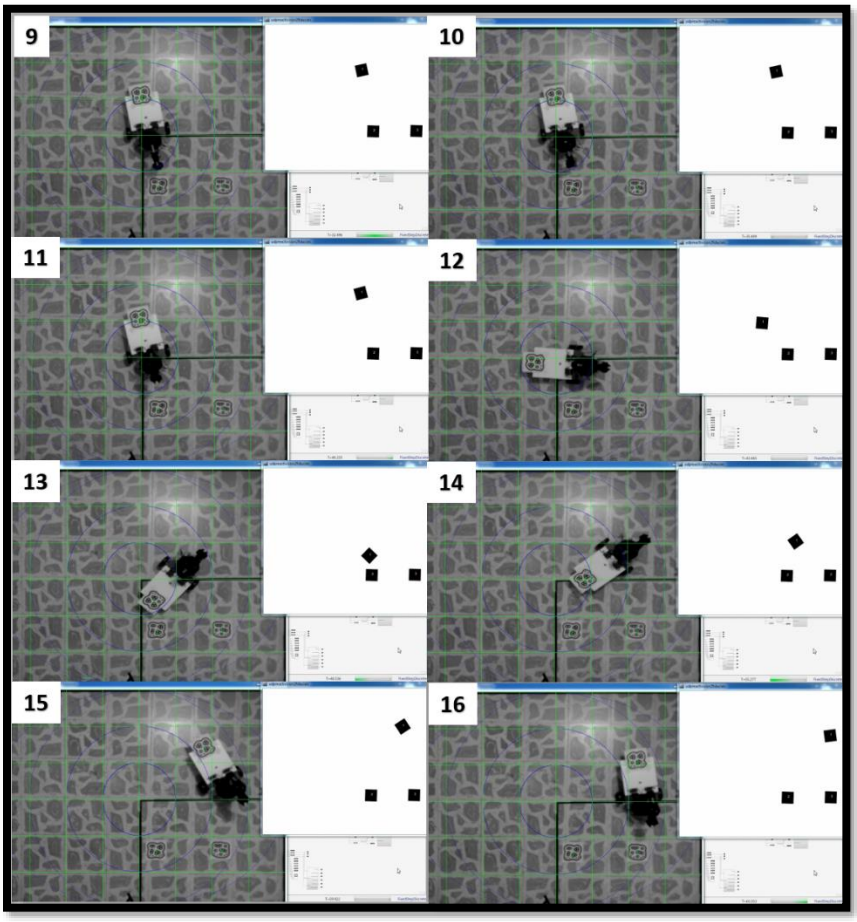


Figura 142.- Capturas del experimento número tres, en el que se ejecuta el algoritmo coordinante para el traslado del objeto dentro del espacio inteligente, parte 2

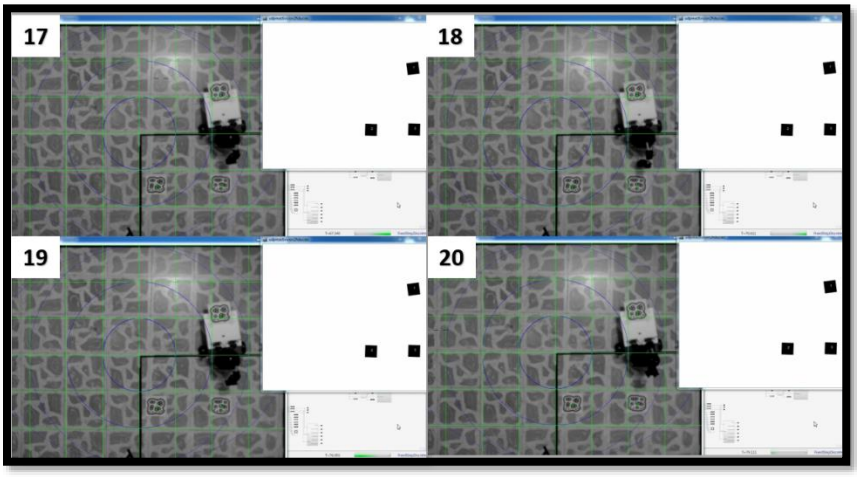


Figura 143.- Capturas del experimento número tres, en el que se ejecuta el algoritmo coordinante para el traslado del objeto dentro del espacio inteligente, parte 3

CAPÍTULO 10.- CONCLUSIONES

10.1 Conclusión

En conclusión, se comprueba la hipótesis planteada ya que el lugar geométrico o espacio intuitivo de traslado basado en los patrones que genera la intuición humana es programado y transferido al algoritmo coordinante del manipulador móvil, para desarrollar la tarea de traslado dentro del espacio inteligente.

10.2 Discusión de resultados

- En esta tesis se aplicaron los patrones que genera la intuición humana en el traslado de objetos para la coordinación de un manipulador móvil, extendiendo el concepto de intuición artificial a la robótica móvil, brindando así sistemas capaces de dar soluciones rápidas y certeras.
- Se programa el algoritmo que genera el lugar geométrico para que se modifique en tiempo real, es decir, si el punto inicial y final cambia, este se actualiza y proporciona una nueva trayectoria, sin tener que realizar una planeación previa.
- Por primera vez se logra programar verdadera intuición artificial en un manipulador móvil, ya que en trabajos previos solo se utilizan los rasgos básicos de los principios de intuición artificial para realizar movimientos suaves y coordinados, pero no se programa ninguna fórmula que denote los patrones generados por la intuición humana, dicha aplicación solo se lleva a cabo en la tesis de (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014), pero enfocada a sistemas teleoperados.
- Se construye un manipulador móvil como banco de pruebas para llevar a cabo experimentos relacionados con intuición artificial y otras teorías de la robótica móvil.
- Se cumple con el objetivo general y cada objetivo específico planteado.

- Con respecto al módulo electrónico se aporta y logra establecer comunicación en tiempo real y vía WiFi-UDP con los servomotores Dynamixel serie AX-12A y todos aquellos cuya comunicación sea TTL, siendo parte esencial en los experimentos, ya que nos permite controlar las posiciones articulares del brazo serial para que el efector final pueda seguir el espacio intuitivo generado, además se retroalimentan a Simulink vía WiFi-UDP los valores de voltaje, temperatura, corriente, posición, velocidad y carga de los servomotores en tiempo real, esto quiere decir que la información se gráfica al instante en el que los experimentos se ejecutan.
- Al realizar las correspondientes pruebas se observa que la comunicación entre la placa arduino y los servomotores se puede realizar directamente sin utilizar el integrado 74LS241, pero al conectar varios servomotores la comunicación falla.
- Con respecto a la retroalimentación se observa que existen pérdida de datos debido a la misma naturaleza de la placa microcontroladora empleada, es decir, no soporta la transmisión de múltiples datos a altas frecuencias o en tiempo real.
- Las velocidades retroalimentadas en las etapas “pick”, “pick back”, “place” y “return” presentan variaciones en un rango de 0 a 65 revoluciones por minuto.
- Los porcentajes de carga de cada servomotor varían en un rango de 0% a 85%, los picos se alcanzan cuando el brazo se extiende por debajo de la plataforma móvil para sujetar o colocar el objeto.
- El voltaje consumido por cada servomotor cambia en un rango de 9 a 12 volts, ya que cada servomotor controla el uso de energía en proporción a la carga que se le aplica.
- Las temperaturas retroalimentadas durante la ejecución de las etapas “pick”, “pick back”, “place” y “return” varían de 35 a 49 grados. Este dato es muy importante ya que se observa que los servomotores no se sobrecalientan en la ejecución del lugar geométrico.
- De acuerdo con las características del brazo serial se logra alcanzar un radio máximo de 24 centímetros durante los experimentos.

- Se observa que el tiempo de muestreo programado en el bloque “Packet output” (ver figura 71) que envía el paquete de datagramas a la tarjeta arduino (ID, posiciones y velocidades de cada servomotor), es determinante para lograr un movimiento suave en el brazo serial, así como también la cantidad de instrucciones programadas en la tarjeta arduino y la velocidad enviada a cada servomotor.
- Finalmente, el error observado en base a las posiciones articulares enviadas vs las retroalimentas varia en el rango de cero a dos grados, dicha variación depende de la fuerza de la señal WiFi, el protocolo utilizado para el envío de datos y el nivel de carga de la pila. En ocasiones por las características de la tarjeta de adquisición en lo que respecta a los tiempos de muestreo, los datos retroalimentados de cada servomotor se desfasan, son negativos o simplemente no se envían a Simulink, por tal motivo se programa en arduino un filtro que evita la transmisión de datos erróneos.

10.3 Recomendaciones

- Utilizar un equipo de cómputo capaz de soportar simulaciones en tiempo real (procesador intel core i7).
- Emplear para las simulaciones una unidad de estado sólido (SSD).
- Verificar que el voltaje suministrado al circuito integrado 74LS241 sea de 5 volts. Un voltaje menor a 4.7 genera errores en la comunicación y retroalimentación de datos con la placa arduino.
- Se recomienda utilizar un sistema de visión más preciso y fácil de calibrar, ya que es un reto lograr una buena calibración debido a varios factores que afectan el sistema, por ejemplo, la iluminación.
- Utilizar el sistema operativo Windows 7, ya que en base a la experiencia del autor de la presente tesis se observa que el kernel que se instala en Matlab para realizar las simulaciones en modo externo y en tiempo real no es compatible con los sistemas operativos Windows 8, 8.1, 10 y todas sus versiones, provocando una aceleración de CPU al 100%.

- En caso de tener ese problema se debe cambiar el plan de energía a economizador o alto rendimiento si se tiene configurado en equilibrado, o restaurarlo a su configuración predeterminada (ver figura 144). Otro problema que ocasiona es el congelamiento de las simulaciones y el sistema operativo, dañando a los discos duros mecánicos.
- Alimentar la placa arduino mega con un mínimo de 7 volts y máximo 12 volts.
- Se recomienda diseñar un circuito electrónico impreso para establecer la comunicación entre todos los componentes.
- Emplear una tarjeta de adquisición de datos profesional y que tenga soporte para comunicación inalámbrica.
- Programar una biblioteca (S-Function) para controlar directamente a los servomotores Dynamixel desde Simulink y obtener una mejor retroalimentación de cada servomotor.
- Cuando se compila el programa en Simulink se debe estar conectado a la red inalámbrica, ya que en ocasiones no se compila correctamente si la red está desconectada.
- Antes de ejecutar los experimentos se debe verificar que cada servomotor esté bien configurado con los parámetros deseados, ya que en ocasiones por la recepción de datos erróneos se desconfiguran los ID y los límites angulares programados.
- La actualización KB4041676 (Windows 10) y KB4041681 (Windows 7) generan problemas con el software “RoboPlus”.
- Se recomienda mejorar el código escrito para calcular la cinemática inversa, con una técnica más robusta.

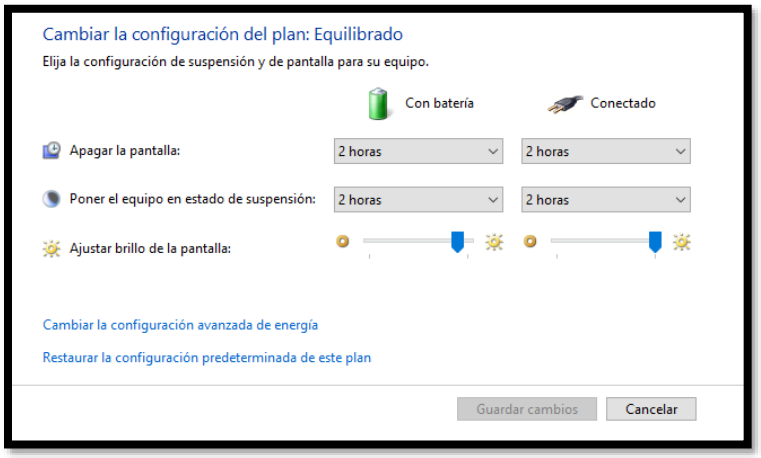


Figura 144.- Plan de energía

10.4 Trabajo a futuro

- Emplear el manipulador móvil construido como un banco de pruebas para implementar el modelo matemático basado en habilidades intuitivas para la *evasión de obstáculos* presentado en la tesis de (Díaz Hernández, Intuición artificial aplicada a la teleoperación, 2014).
- Programar a cada servomotor las velocidades presentadas en la tesis antes mencionada para generar trayectorias intuitivas y no solo lugares geométricos.
- Realizar un diseño de experimentos comparando la efectividad del generador de lugares geométricos programado en esta tesis con otras técnicas, específicamente en la etapa “pick and place” del brazo serial.

APÉNDICES

Apéndice A.- Código programado en el bloque para el control del brazo serial

En las siguientes figuras se muestra la programación hecha en Stateflow de Simulink para coordinar el movimiento de ida y regreso del brazo serial montado sobre la plataforma móvil. La figura A.1 y A.2 muestran las ecuaciones 45, 46, 47 y 48 programadas, las cuales generan el lugar geométrico o espacio intuitivo de traslado dependiendo el punto inicial y final entrantes.

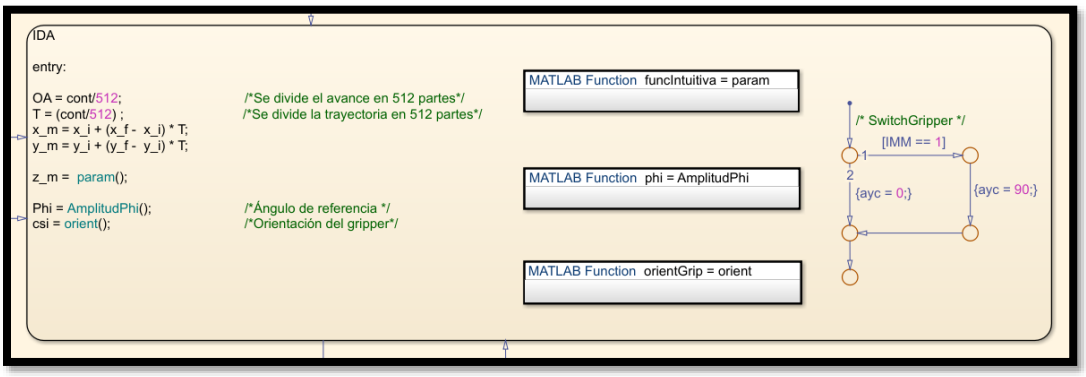


Figura A.1.- Bloque para el control del brazo serial: IDA

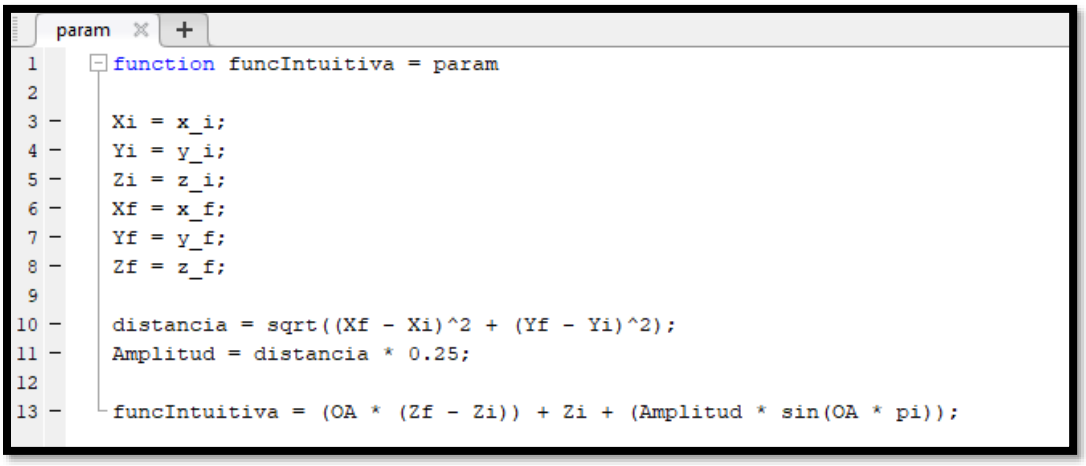


Figura A.2.- Bloque para el control del brazo serial: "funcIntuitiva = param"

En la figura A.3 se observa la programación del parámetro ϕ que es el ángulo respecto a la horizontal con el cuál será tomado el objeto, dicho parámetro debe variarse en su trayecto para alcanzar cada punto del lugar geométrico que se quiere describir, en este caso se parte de 40° y se requiere llegar al objeto con un ángulo de 90°.

AmplitudPhi

```
1   function phi = AmplitudPhi
2
3 -   phi = 40;
4
5 -   if (cont >= 0 && cont <= 512)
6 -           phi = 40 + (50 * T);   %Valor de phi, desde 40° hasta 90°
7     end
```

Figura A.3.- Bloque para el control del brazo serial: "phi = AmplitudPhi"

La figura A.4 describe la programación del ángulo que orienta a la articulación cinco con respecto a el ángulo de orientación del objeto o el destino.

orient

```
1   function orientGrip = orient
2
3 -   h = 1.8;
4
5 -   cl = cos(theta_r);
6 -   sl = sin(theta_r);
7
8 -   ha  = x_r + (h * cl);
9 -   ka  = y_r + (h * sl);
10 -  eta = atan2(y_f - ka, x_f - ha);
11
12 -  orientGrip = eta - theta_f;
13
14 -  if(orientGrip < 0)
15 -      orientGrip = pi + orientGrip;
16    end
17
18    end
```

Figura A.4.- Bloque para el control del brazo serial: "orientGrip = orient"

La figura A.5 muestra la programación para ejecutar el movimiento de regreso, se observa que se programan las mismas fórmulas que en el movimiento de ida, para describir el espacio geométrico basado en intuición.

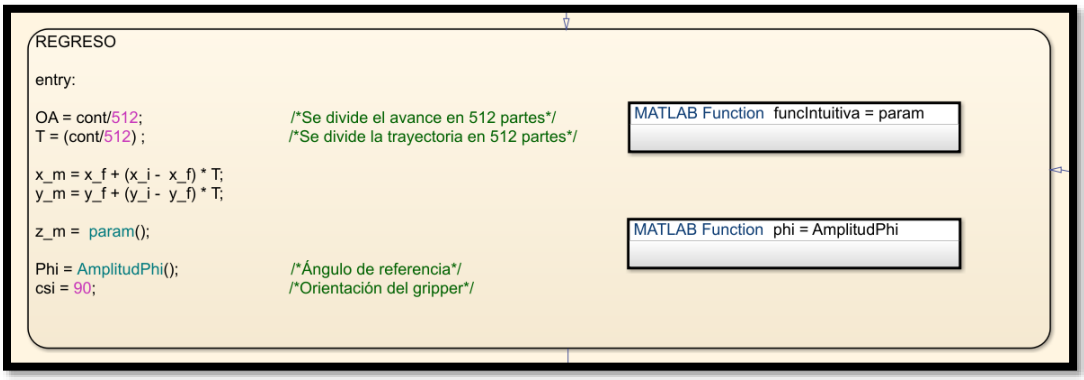


Figura A.5.- Bloque para el control del brazo serial: REGRESO

En la figura A.6 se reprograma el parámetro ϕ para que parta de 90° y regrese al ángulo inicial de 40°, mientras que el ángulo de orientación de la articulación cinco se programa a 90°.

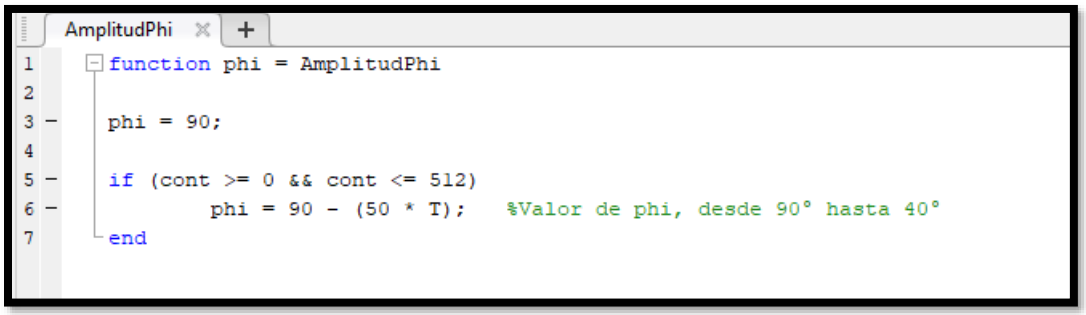


Figura A.6.- Bloque para el control del brazo serial: "phi = AmplitudPhi"

En la figura A.7 y A.8 se añade en la programación del movimiento de ida y regreso un polinomio de quinto orden para suavizar el movimiento, se obtienen datos y se experimenta, pero al notarse que el perfil geométrico varia se decide no implementarlo en la prueba final.

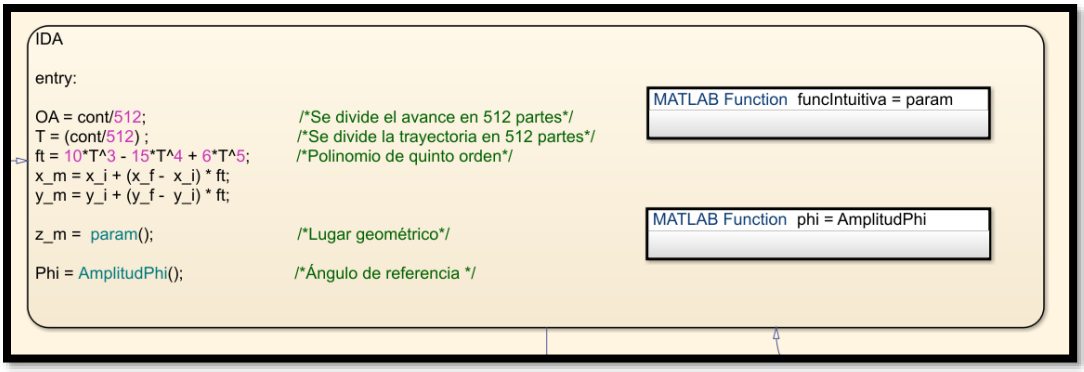


Figura A.7.- Bloque para el control del brazo serial: IDA con polinomio

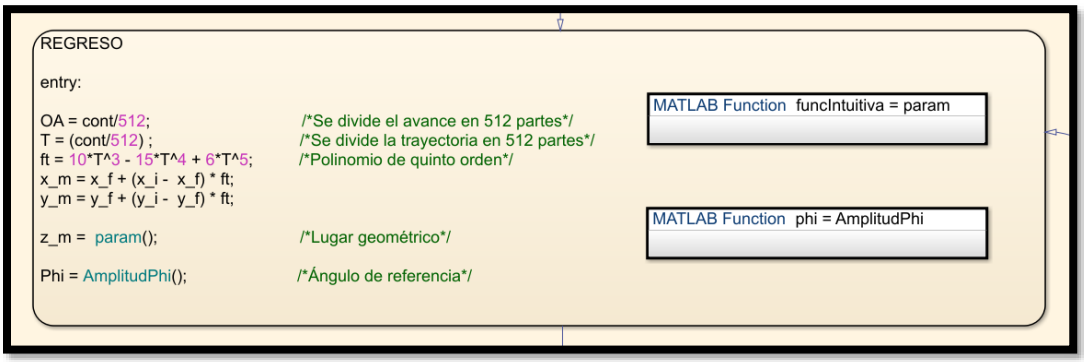
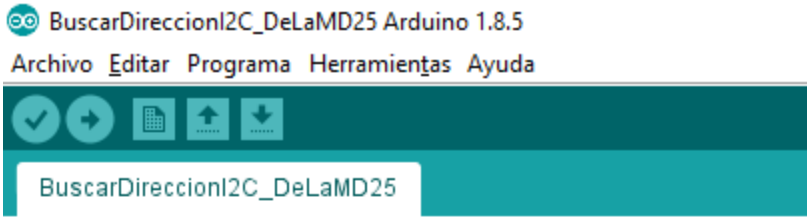


Figura A.8.- Bloque para el control del brazo serial: REGRESO con polinomio

Apéndice B.- Código arduino

B.1 Programa para buscar las direcciones I2C de la tarjeta MD25




```
/*--------------------------------------------------------
  Programa para buscar dispositivos I2C conectados.
  Manda las direcciones que encuentra a la puerta serie

  Encontrado en:
       http://playground.arduino.cc/Main/I2cScanner

--------------------------------------------------------
*/

#include <Wire.h>

void setup()
{
  Wire.begin();

  Serial.begin(9600);
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
```

```
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknow error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);           // wait 5 seconds for next scan
}
```

B.2 Programa para controlar el manipulador móvil mediante conexión WiFi-UDP

TesisManipuladorMovilUDPDynamixelAX12yMD25VS Arduino 1.8.5

Archivo Editar Programa Herramientas Ayuda

TesisManipuladorMovilUDPDynamixelAX12yMD25VS

```
//Programa arduino para controlar el Manipulador Móvil mediante conexión Wifi UDP
//José Crisogono Aldana Becerra
//Librerias Dynamixel: Josué Alejandro Savage

//Bibliotecas

#include<DynamixelSerial1.h>     //Libreria para los servomotores Dynamixel
#include <Wire.h>                //Libreria para comunicarse con dispositivos I2C/TWI en este caso la tarjeta MD25
#include <WiFi.h>                //Libreria para establecer la conexión Wifi
#include <WiFiUdp.h>             //Libreria para enviar y recibir mensajes UDP

//Configuración de parámetros para la tarjeta MD25

#define DIRECCIONI2CMD25 0x58     //Dirección de la tarjeta
#define VELMOTORIZQ      0x00     //Dirección del registro de velocidad del motor izquierdo
#define VELMOTORDER      0x01     //Dirección del registro de velocidad del motor derecho

byte id, posd_h, posd_l, vel_h, vel_l;
byte posa_h, posa_l, Speed_h, Speed_l, Load_h, Load_l;
int posd, posd_val, vel;
int posa, posa_val, Speed, Load, Voltage, Temperature;

byte SpeedI, SpeedD;

byte ID1, ID2, ID3, ID4, ID5, ID6;

byte bandera;

//*********************************************************************************************Clave de la red inalámbrica*********
//char ssid[] = "Kratos";             //Nombre de la red inalámbrica SSID
//char pass[] = "82497ee384EWtxGT?";  //Clave de la red con protección WPA

const int NTP_PACKET_SIZE = 9;        //Costante que indica el número de datos que se envian a simulink (Datos Dynamixel)
byte packet[NTP_PACKET_SIZE];         //Buffer para almacenar los datos que se envian a simulink (Datos Dynamixel)

unsigned int localPort1 = 1234;       //Puerto local que recibe y envia los datos (Datos MD25)
unsigned int localPort2 = 5678;       //Puerto local que recibe y envia los datos (Datos Dynamixel)

int status = WL_IDLE_STATUS;          //Radio de alcance de la señal

char packetBuffer[255];               //Buffer para mantener el paquete entrante (Datos MD25)
char packetBuffer1[255];              //Buffer para mantener el paquete entrante (Datos Dynamixel)

WiFiUDP Udp, Udp2, Udp3;              //Se crean instancias para enviar y recibir mensajes UDP, Udp = Recibe, Udp2 = Recibe, Udp3 = Envia

//*********************************************************************************************Void setup()************************************

void setup() {

  Wire.begin();                 //Se inicia la comunicación I2C
  Serial.begin(9600);           //Se inicia la comunicación serial a 9600 bps
  Dynamixel.begin(1000000,2);   //Se inicia el servomotor a 1 Mbps, conectado al pin 2
  bandera = 0;                  //Se inicia la variable bandera en 0 para poder controlar el orden de las instrucciones, así como también

  ID1 = 1;     //Variables para identificar el ID de cada servomotor
  ID2 = 2;
  ID3 = 3;
  ID4 = 4;
```

```
  ID5 = 5;
  ID6 = 6;

  BrazoPosturaInicial();     //Posición en reposo del brazo

  while (!Serial) {
    ; //Espera a que el puerto serie se conecte. Necesario sólo para el puerto USB nativo
  }

  //Se verifica que el shield arduino Wifi esté conectado
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("Wifi shield desconectado");
    //No continuar hasta que se conecte
    while(true);
  }

  //Versión del shield arduino Wifi
  String fv = WiFi.firmwareVersion();
  if (fv != "1.1.0") {
    Serial.println("Porfavor actualiza el firmware de tu shield arduino Wifi");
  }

  //Intenta conectarse a la red Wifi
  while (status != WL_CONNECTED) {
    Serial.print("Intentando conectarse a WPA SSID: ");
    Serial.println(ssid);
    //Conectarse a la red WPA/WPA2:
    status = WiFi.begin(ssid, pass);
    //Esperar 5 segundos para la conexión:
   delay(5000);
  }

  //Muestra la información de tu red:
  Serial.print("Estas conectado a la red ");
  printCurrentNet();
  printWifiData();

  Udp.begin(localPort1);      //Inicia el socket WiFiUDP, para leer datos del puerto local
  Udp2.begin(localPort2);     //Inicia el socket WiFiUDP, para leer datos del puerto local

}

//*************************************************************************************************Void loop()*******

void loop() {

  int packetSize = Udp.parsePacket();       //Si hay datos disponibles, leer el paquete de datos
  int packetSize1 = Udp2.parsePacket();     //Si hay datos disponibles, leer el paquete de datos

  switch(bandera){

  //PLATAFORMA MÓVIL
  case 0:

  if(packetSize) {

    //Serial.print("Se ha recibido un paquete de tamaño: ");
    //Serial.println(packetSize);
```

```
  //Serial.print("De ");
  IPAddress remoteIp = Udp.remoteIP();
  //Serial.print(remoteIp);
  //Serial.print(", puerto ");
  //Serial.println(Udp.remotePort());

  int len = Udp.read(packetBuffer,255);     //Lee el paquete dentro del packetBufffer

  //Serial.println("Contiene:");     //Se muestra en pantalla los datos recibidos
  //Serial.println(packetBuffer);

if (len > 0) {
  packetBuffer[len] = 0;
  }

  //Velocidades de las llantas
  SpeedI = packetBuffer[0];     //Velocidad 1
  SpeedD = packetBuffer[1];     //Velocidad 2

  //Serial.print(SpeedI);
  //Serial.print("   " );
  //Serial.print(SpeedD);
  //Serial.print("   " );

  //Transmisión de datos para la tarjeta MD25, se acciona el motor 1 y 2 con el valor a
  Move(SpeedI,SpeedD);
}

  bandera = 1;
break;

//Programa para controlar los servomotores Dynamixel AX-12A

case 1:

 if(packetSize1) {

  //Serial.print("Se ha recibido un paquete de tamaño: ");
  //Serial.println(packetSize1);
  //Serial.print("De ");
  IPAddress remoteIp1 = Udp2.remoteIP();
  //Serial.print(remoteIp1);
  //Serial.print(", puerto ");
  //Serial.println(Udp2.remotePort());

  int len1 = Udp2.read(packetBuffer1,255);     //Lee el paquete dentro del packetBufffer

  //Serial.println("Contiene:");     //Se muestra en pantalla los datos recibidos
  //Serial.println(packetBuffer1);

if (len1 > 0) {
  packetBuffer1[len1] = 0;
  }

    id     = packetBuffer1[0];
    posd_h = packetBuffer1[1];
    posd_l = packetBuffer1[2];
    vel_h  = packetBuffer1[3];
```

```
    vel_l  = packetBuffer1[4];

    posd = posd_h * 256 + posd_l;
    vel = vel_h * 256 + vel_l;

 }

  bandera = 2;

break;

case 2:

  posd_val = map(posd,0,300,0,1024);
  Dynamixel.moveSpeed(id, posd_val, vel);

  Speed = Dynamixel.readSpeed(id);
  posa_val = Dynamixel.readPosition(id);
  Load = Dynamixel.readLoad(id);
  Voltage = Dynamixel.readVoltage(id);
  Temperature = Dynamixel.readTemperature(id);

  posa = map(posa_val,0,1024,0,300);
  posa_h = posa>>8;
  posa_l = posa;
  Speed_h = Speed>>8;
  Speed_l = Speed;
  Load_h = Load>>8;
  Load_l = Load;

  bandera = 3;

break;

case 3:

  if (Speed >= 0 && posa_val >= 0 && Load >= 0 && Voltage >= 0 && Temperature >= 0)    //Instrucción nec
  {
  packet[0] = id;
  packet[1] = posa_h;
  packet[2] = posa_l;
  packet[3] = Speed_h;
  packet[4] = Speed_l;
  packet[5] = Load_h;
  packet[6] = Load_l;
  packet[7] = Voltage;
  packet[8] = Temperature;
  Udp3.beginPacket(Udp2.remoteIP(), 16000);     //Importante en simulink: programar en el "Packet input"
                                                //diferente al que se programó en el "Packet output"
  Udp3.write(packet,NTP_PACKET_SIZE);           //Instrucción para enviar los datos a simulink
  Udp3.endPacket();
  }

  bandera = 0;

break;
```

```
  }

}

//*****************************************************************************************

void BrazoPosturaInicial(){

  Dynamixel.ledStatus(ID1,ON);                  //Enciende el led del servomotor
  Dynamixel.torqueStatus(ID1,ON);               //Se habilita el torque del servo
  Dynamixel.setMaxTorque(ID1,1023);             //Se habilita el máximo torque (c
  Dynamixel.setAngleLimit(ID1,205,819);         //Se establecen los límites de gi
  Dynamixel.moveSpeed(ID1,512,97);              //Se hace girar al servomotor en
  Dynamixel.ledStatus(ID1,OFF);                 //Apaga el led del servomotor

  Dynamixel.ledStatus(ID2,ON);
  Dynamixel.torqueStatus(ID2,ON);
  Dynamixel.setMaxTorque(ID2,1023);
  Dynamixel.setAngleLimit(ID2,205,819);
  Dynamixel.moveSpeed(ID2,666,97);
  Dynamixel.ledStatus(ID2,OFF);

  Dynamixel.ledStatus(ID3,ON);
  Dynamixel.torqueStatus(ID3,ON);
  Dynamixel.setMaxTorque(ID3,1023);
  Dynamixel.setAngleLimit(ID3,205,512);
  Dynamixel.moveSpeed(ID3,205,97);
  Dynamixel.ledStatus(ID3,OFF);

  Dynamixel.ledStatus(ID4,ON);
  Dynamixel.torqueStatus(ID4,ON);
  Dynamixel.setMaxTorque(ID4,1023);
  Dynamixel.setAngleLimit(ID4,205,512);
  Dynamixel.moveSpeed(ID4,205,97);
  Dynamixel.ledStatus(ID4,OFF);

  Dynamixel.ledStatus(ID5,ON);
  Dynamixel.torqueStatus(ID5,ON);
  Dynamixel.setMaxTorque(ID5,1023);
  Dynamixel.setAngleLimit(ID5,205,819);
  Dynamixel.moveSpeed(ID5,512,97);
  Dynamixel.ledStatus(ID5,OFF);

  Dynamixel.ledStatus(ID6,ON);
  Dynamixel.torqueStatus(ID6,ON);
  Dynamixel.setMaxTorque(ID6,1023);
  Dynamixel.setAngleLimit(ID6,205,512);
  Dynamixel.moveSpeed(ID6,512,97);
  Dynamixel.ledStatus(ID6,OFF);

  }

void printWifiData() {

  //Muestra la dirección IP del Wifi shield:
  IPAddress ip = WiFi.localIP();
  Serial.print("Dirección IP: ");
```

```
  Serial.println(ip);

  //Muestra la dirección MAC del Wifi shield:
  byte mac[6];
  WiFi.macAddress(mac);
  Serial.print("Dirección MAC: ");
  Serial.print(mac[5], HEX);
  Serial.print(":");
  Serial.print(mac[4], HEX);
  Serial.print(":");
  Serial.print(mac[3], HEX);
  Serial.print(":");
  Serial.print(mac[2], HEX);
  Serial.print(":");
  Serial.print(mac[1], HEX);
  Serial.print(":");
  Serial.println(mac[0], HEX);

}

void printCurrentNet() {

  //Imprime el SSID (Nombre de la red inalámbrica) a la que está conectado
  Serial.print("SSID (Nombre de la red inalámbrica): ");
  Serial.println(WiFi.SSID());

  //Muestra la dirección MAC del router al que está conectado:
  byte bssid[6];
  WiFi.BSSID(bssid);
  Serial.print("BSSID (Dirección MAC del router al que está conectado): ");
  Serial.print(bssid[5], HEX);
  Serial.print(":");
  Serial.print(bssid[4], HEX);
  Serial.print(":");
  Serial.print(bssid[3], HEX);
  Serial.print(":");
  Serial.print(bssid[2], HEX);
  Serial.print(":");
  Serial.print(bssid[1], HEX);
  Serial.print(":");
  Serial.println(bssid[0], HEX);

  //Muestra la intensidad de la señal recibida:
  long rssi = WiFi.RSSI();
  Serial.print("Indicador de fuerza de la señal recibida (RSSI): ");
  Serial.print(rssi);
  Serial.println(" dBm");

  //Muestra el tipo de encriptación:
  byte encryption = WiFi.encryptionType();
  Serial.print("Tipo de encriptación: ");
  printEncryptionType(encryption);
  Serial.println();
}

void printEncryptionType(int thisType) {

  //Leer el tipo de encriptación y mostrarlo en pantalla
```

```
  switch (thisType) {
    case ENC_TYPE_WEP:
      Serial.println("WEP");
      break;
    case ENC_TYPE_TKIP:
      Serial.println("WPA");
      break;
    case ENC_TYPE_CCMP:
      Serial.println("WPA2");
      break;
    case ENC_TYPE_NONE:
      Serial.println("None");
      break;
    case ENC_TYPE_AUTO:
      Serial.println("Auto");
      break;
  }
}

void Move(byte SpeedI, byte SpeedD){

  Wire.beginTransmission(byte(DIRECCIONI2CMD25));
  Wire.write(VELMOTORIZQ);
  Wire.write(SpeedI);
  Wire.endTransmission();

  Wire.beginTransmission(byte(DIRECCIONI2CMD25));
  Wire.write(VELMOTORDER);
  Wire.write(SpeedD);
  Wire.endTransmission();
  }
```

Guardado.

Apéndice C.- Código Matlab

- Módulos programados en Simulink de Matlab para el “Experimento número 3”

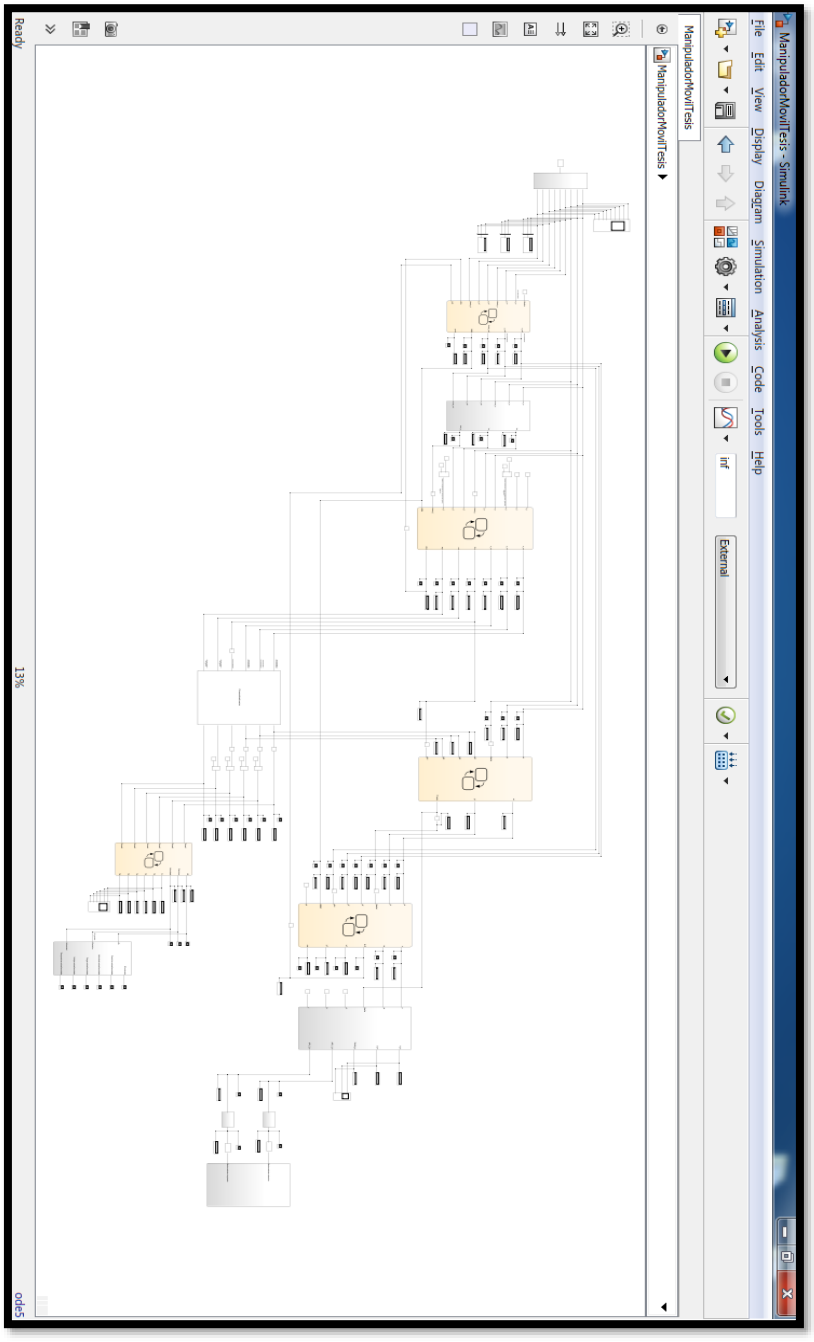


Figura C.1.- Diagrama de bloques del algoritmo de coordinación

C.1 Módulo de visión

- Vista externa

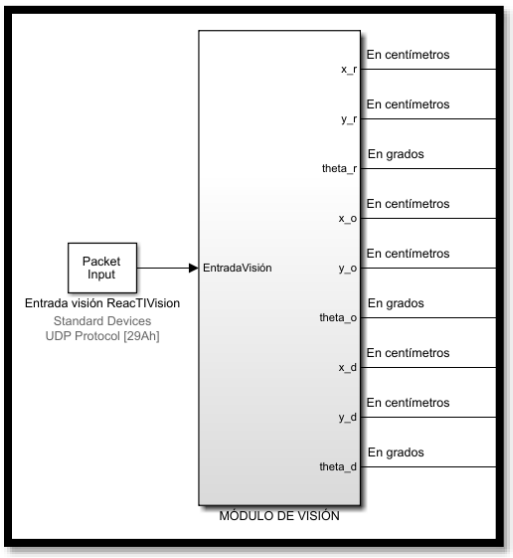


Figura C.2.- Bloque del módulo de visión vista externa

- Vista interna

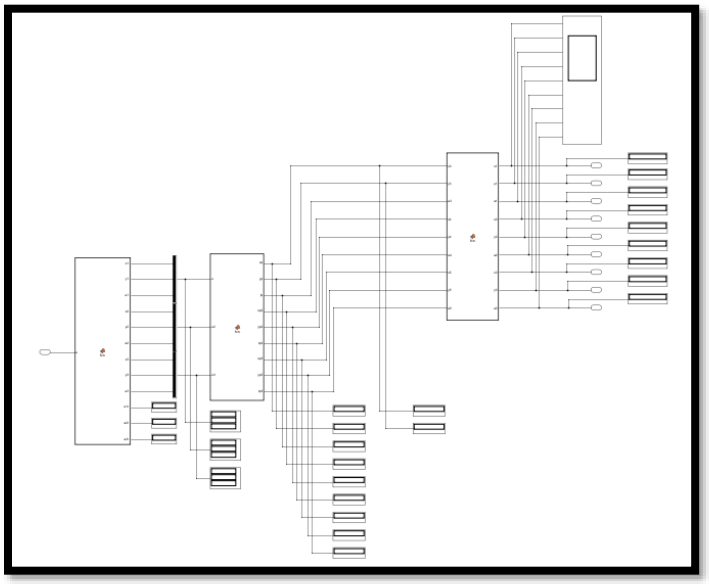


Figura C.3.- Bloque del módulo de visión vista interna

- Código programado en el bloque: 3 fiducials, 28 datos de entrada, ajuste inicial

```
function [x1,y1,a1,x2,y2,a2,x3,y3,a3,a1r,a2r,a3r]= 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*pi/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*pi/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*pi/180;
```

- Código programado en el bloque: Ajuste a centímetros y sentido de giro

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

x  = u(1);
y  = u(2);
th = u(3);      %En grados

x2  = u2(1);
y2  = u2(2);
th2 = u2(3);    %En grados
```

```
x3  = u3(1);
y3  = u3(2);
th3 = u3(3);   %En grados

%*************************************Coordenadas del robot y del
objetivo**********************************

lfid1 = 95;   %Largo del eje x, en centímetros
afid1 = 80.5;  %Largo del eje y, en centímetros

l = 108.4;   %Largo del eje x, en centímetros
a = 88.8;  %Largo del eje y, en centímetros

%Conversion de pixeles a centímetros
xs   = (x/100)*(-2*lfid1)+lfid1;
xp   = (xs*-1);                  %En centímetros
ys   = (y/100)*(2*afid1)-afid1;
yp   = ys;                       %En centímetros

xs2  = (x2/100)*(-2*l)+l;
xp2   = (xs2*-1);          %En centímetros
ys2  = (y2/100)*(2*a)-a;
yp2   = ys2;               %En centímetros


xs3  = (x3/100)*(-2*l)+l;
xp3   = (xs3*-1);          %En centímetros
ys3  = (y3/100)*(2*a)-a;
yp3   = ys3;               %En centímetros

%%Ajuste sentido antihorario positivo de los ángulos

if th >= 0 && th <= 90
     tp = 90-th;
elseif th > 90 && th <= 180
     tp = 270+180-th;
elseif th > 180 && th <= 270
     tp = 180+270-th;
elseif th > 270 && th <= 360
     tp = 90+360-th;
else
     tp = th;
end

if th2 >= 0 && th2 <= 90
     tp2 = 90-th2;
elseif th2 > 90 && th2 <= 180
     tp2 = 270+180-th2;
elseif th2 > 180 && th2 <= 270
     tp2 = 180+270-th2;
elseif th2 > 270 && th2 <= 360
     tp2 = 90+360-th2;
else
     tp2 = th2;
```

```
end

if th3 >= 0 && th3 <= 90
     tp3 = 90-th3;
elseif th3 > 90 && th3 <= 180
     tp3 = 270+180-th3;
elseif th3 > 180 && th3 <= 270
     tp3 = 180+270-th3;
elseif th3 > 270 && th3 <= 360
     tp3 = 90+360-th3;
else
     tp3 = th3;
end
```

- Código programado en el bloque: Ajustes fiducials (transformación)

```
function [x1,y1,a1,x2,y2,a2,x3,y3,a3] = fcn(x3,y3,a3,x4,y4,a4,x5,y5,a5)
%#codegen
%Los ángulos entrantes se encuentran en grados y las coordenadas en metros

x3m = x3;  %Las unidades se encuentran en centímetros
y3m = y3;
x4m = x4;
y4m = y4;
x5m = x5;
y5m = y5;

a = 32.1;  %32.1 cm.  = 0.321 m. Distancia del centro del fiducial al centro
de la plataforma
b = 18;    %18 cm.    = 0.18 m.  Distancia del centro del fiducial al centro
del objeto a trasladar
c = 18;    %18 cm.    = 0.18 m.  Distancia del centro del fiducial al centro
del objetivo

x1 = (x3m + (a*cos(a3 * (pi/180)))); %En centímetros
y1 = (y3m + (a*sin(a3 * (pi/180)))); %En centímetros
a1 = a3;                             %En grados

x2 = (x4m - (b*cos(a4 * (pi/180)))); %En centímetros
y2 = (y4m - (b*sin(a4 * (pi/180)))); %En centímetros
a2 = a4;                             %En grados

x3 = (x5m - (c*cos(a5 * (pi/180)))); %En centímetros
y3 = (y5m - (c*sin(a5 * (pi/180)))); %En centímetros
a3 = a5;                             %En grados

end
```

C.2 Bloque de selección

- Vista externa

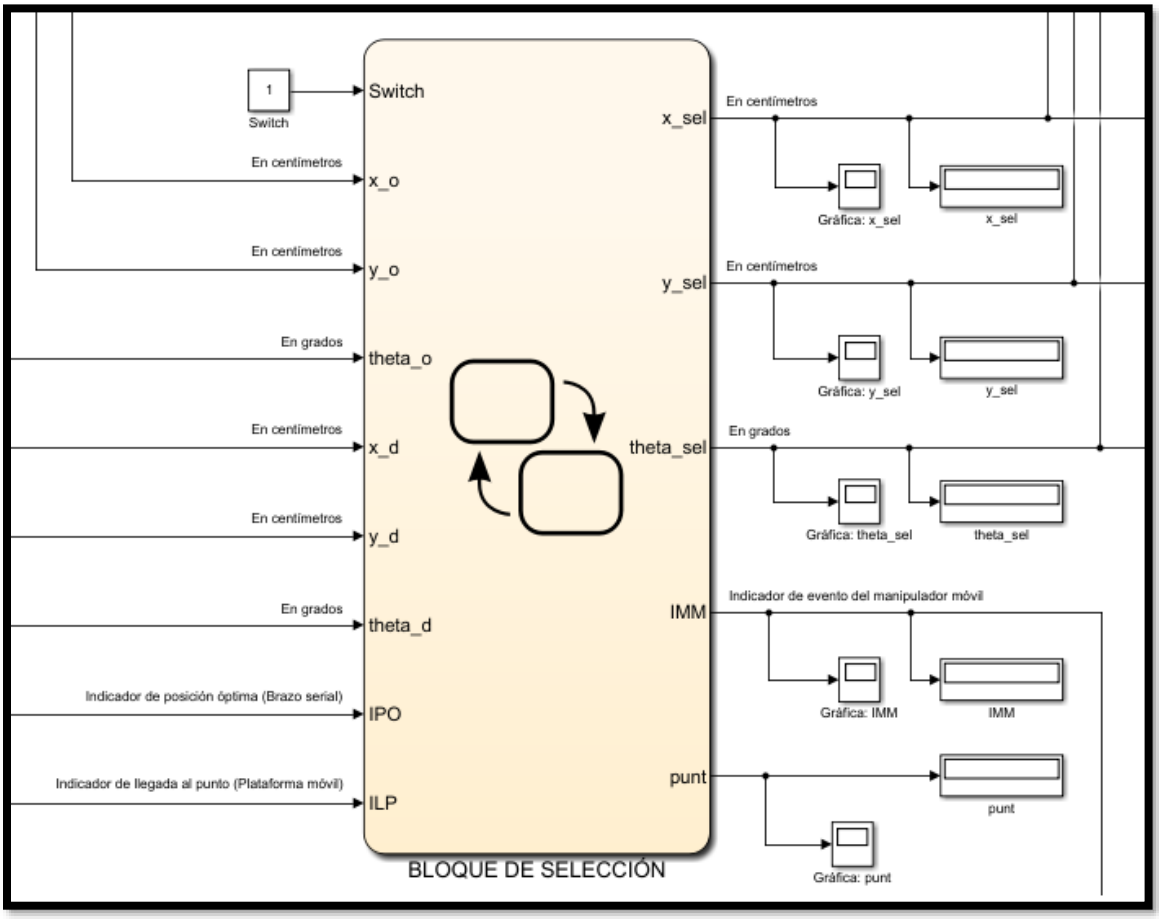


Figura C.4.- Bloque de selección vista externa

- Vista interna

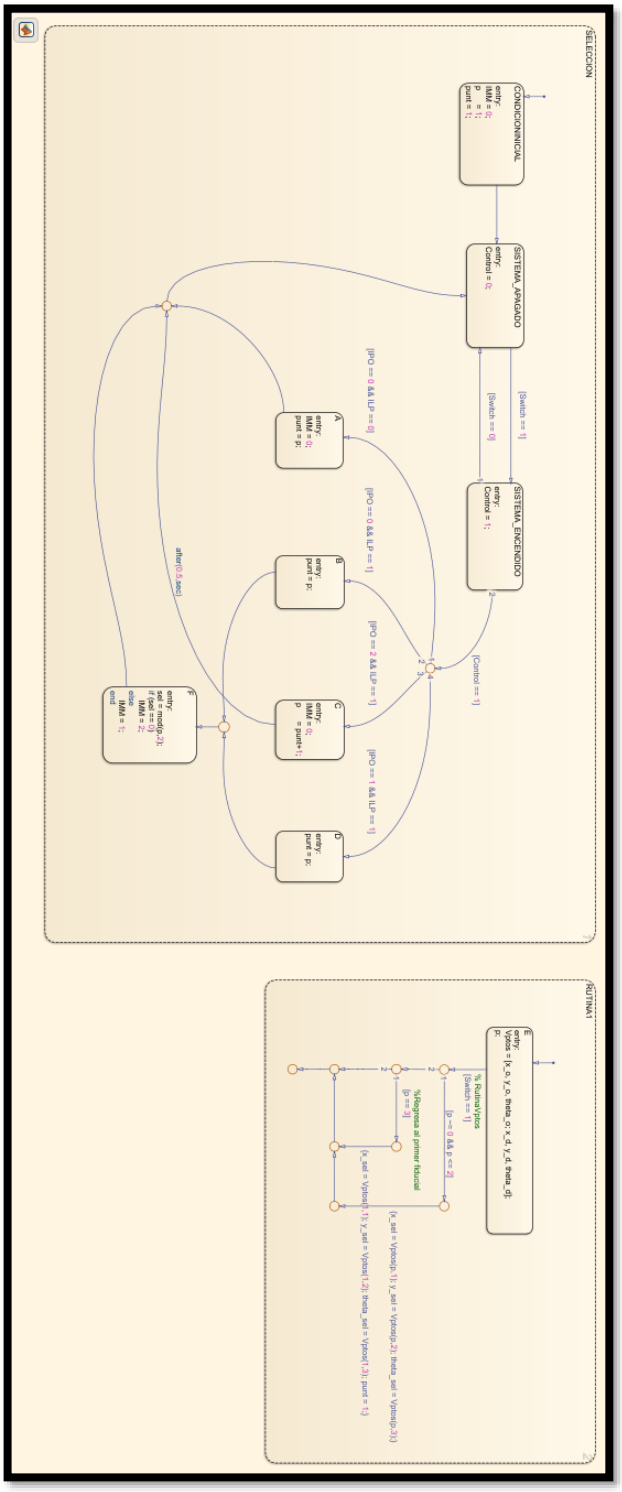


Figura C.5.- Bloque de selección vista interna

C.3 Bloque para la conversión a coordenadas del manipulador

- Vista externa

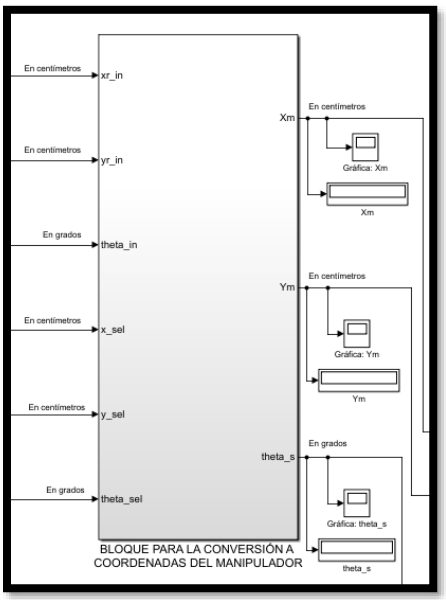


Figura C.6.- Bloque para la conversión a coordenadas del manipulador vista externa

- Vista interna

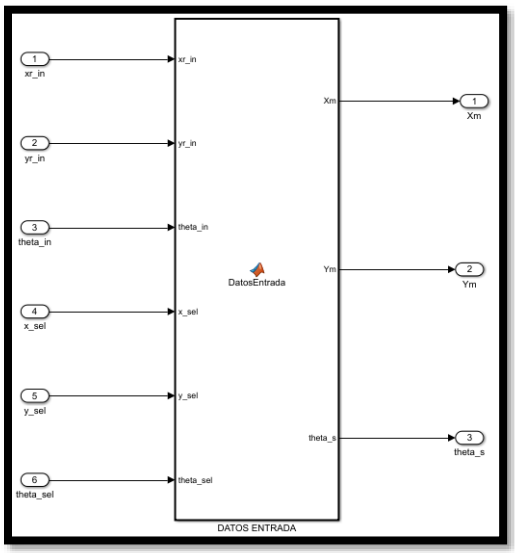


Figura C.7.- Bloque para la conversión a coordenadas del manipulador vista interna

- Código programado en el bloque: DATOS ENTRADA

```
function [Xm, Ym, theta_s] = DatosEntrada(xr_in, yr_in, theta_in, x_sel,
y_sel, theta_sel)
%Los ángulos entrantes se encuentran en grados, se realiza la conversión a
radianes en los cálculos

h = 1.8;                %1.8 cm.   = 0.018 m.    Distancia del centro de la
plataforma al centro del brazo serial

xr      = xr_in;        %En centímetros
yr      = yr_in;        %En centímetros
                        %theta_in se encuentra en grados

x_s     = x_sel;        %En centímetros
y_s     = y_sel;        %En centímetros
theta_s = theta_sel;    %En grados

ct_r = cos(theta_in * (pi/180));
st_r = sin(theta_in * (pi/180));

Xm = ct_r*(x_s - xr) + st_r*(y_s - yr) - h;  %En centímetros
Ym = ct_r*(y_s - yr) - st_r*(x_s - xr);      %En centímetros
```

C.4 Bloque para el control del brazo serial

- Vista externa

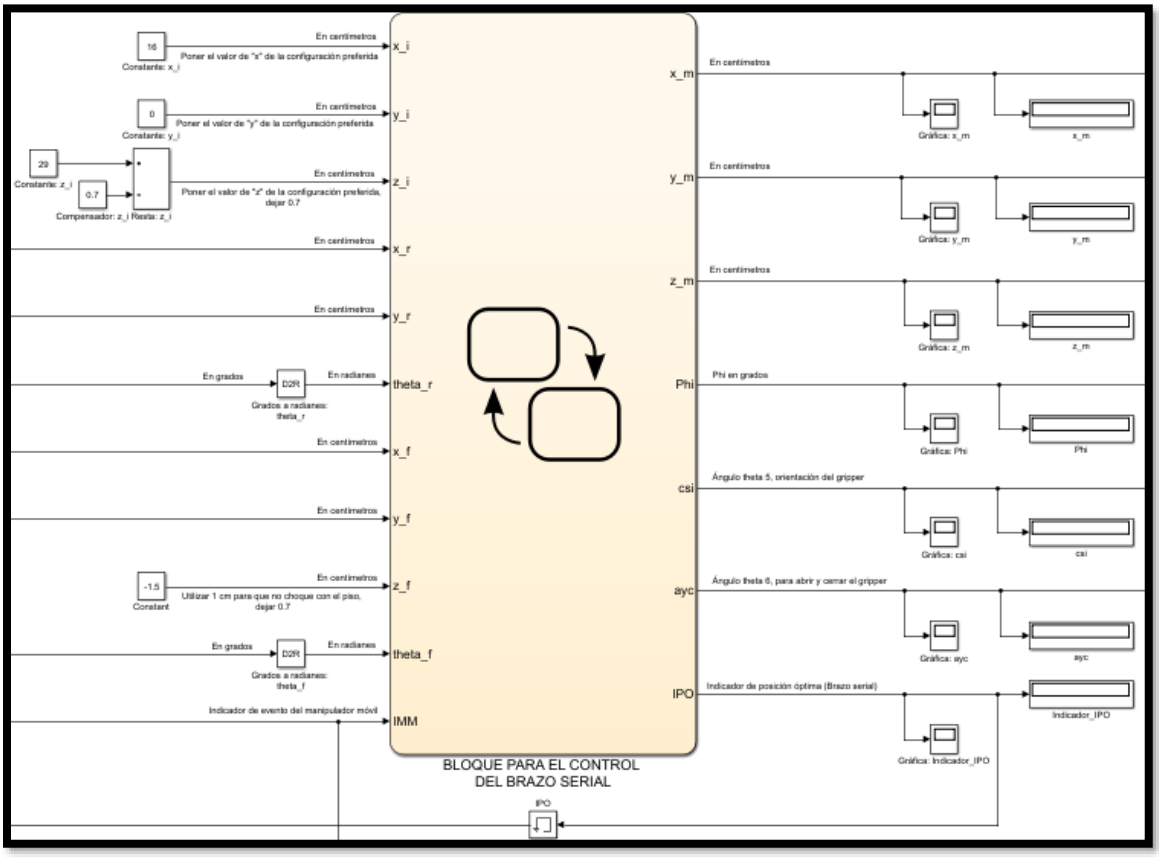


Figura C.8.- Bloque para el control del brazo serial vista externa

- Vista interna

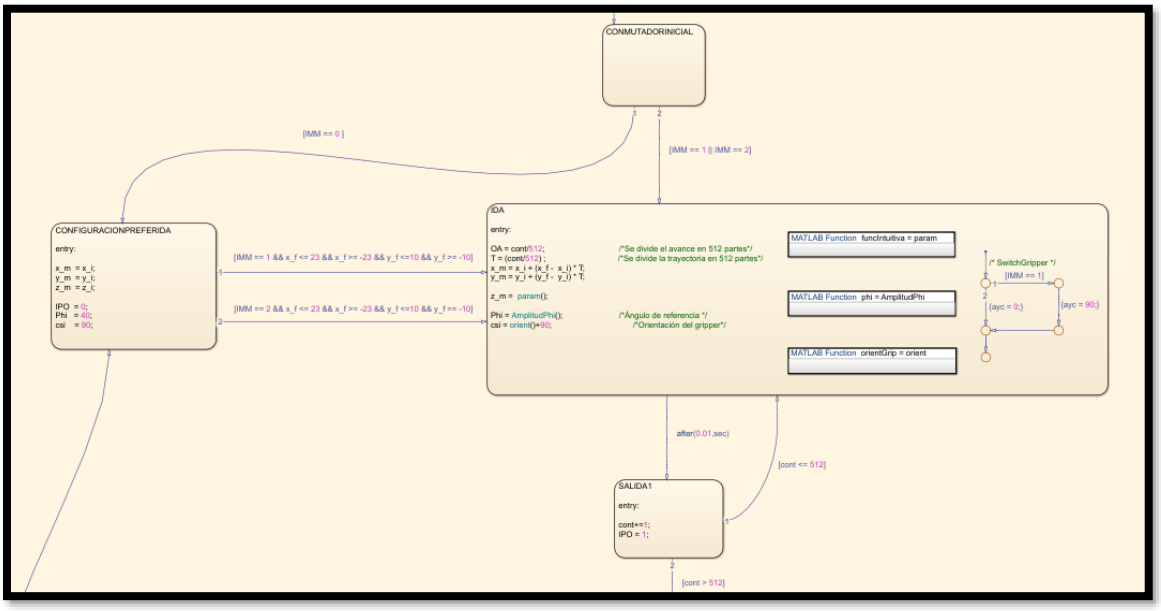
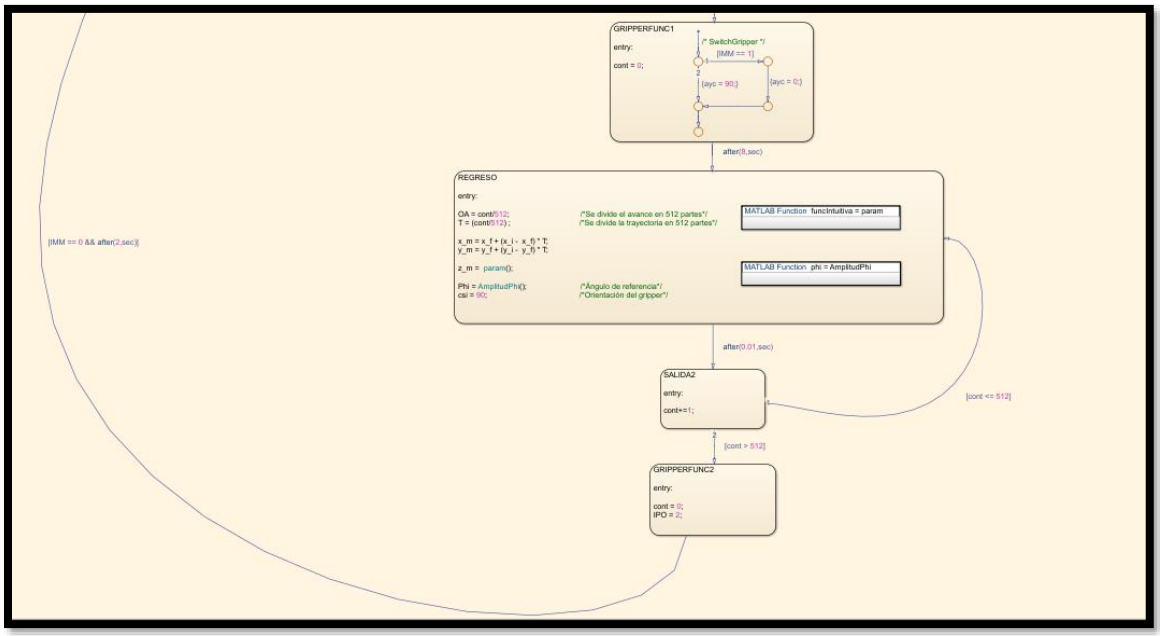


Figura C.9.- Bloque para el control del brazo serial vista interna parte 1



.

Figura C.10.- Bloque para el control del brazo serial vista interna parte 2

- Código programado en el bloque: param

```
function funcIntuitiva = param

Xi = x_i;
Yi = y_i;
Zi = z_i;
Xf = x_f;
Yf = y_f;
Zf = z_f;

distancia = sqrt((Xf - Xi)^2 + (Yf - Yi)^2);
Amplitud = distancia * 0.25;

funcIntuitiva = (OA * (Zf - Zi)) + Zi + (Amplitud * sin(OA * pi));
```

- Código programado en el bloque: AmplitudPhi (IDA)

```
function phi = AmplitudPhi

phi = 40;

if (cont >= 0 && cont <= 512)
        phi = 40 + (50 * T);   %Valor de phi, desde 40° hasta 90°
end
```

- Código programado en el bloque: orient

```
function orientGrip = orient

h = 1.8;

c1 = cos(theta_r);
s1 = sin(theta_r);

ha  = x_r + (h * c1);
ka  = y_r + (h * s1);
eta = atan2(y_f - ka, x_f - ha);

orientGrip = eta - theta_f;

if(orientGrip < 0)
    orientGrip = pi + orientGrip;
end

end
```

- Código programado en el bloque: param (REGRESO)

```
function funcIntuitiva = param

Xi = x_i;
Yi = y_i;
Zi = z_i;
Xf = x_f;
Yf = y_f;
Zf = z_f;

distancia = sqrt((Xf - Xi)^2 + (Yf - Yi)^2);
Amplitud = distancia * 0.25;

funcIntuitiva = (OA * (Zf - Zi)) + Zi + (Amplitud * sin(OA * pi));
```

- Código programado en el bloque: AmplitudPhi (REGRESO)

```
function phi = AmplitudPhi

phi = 90;

if (cont >= 0 && cont <= 512)
        phi = 90 - (50 * T);    %Valor de phi, desde 90° hasta 40°
end
```

C.5 Bloque para el traslado de puntos

- Vista exterior

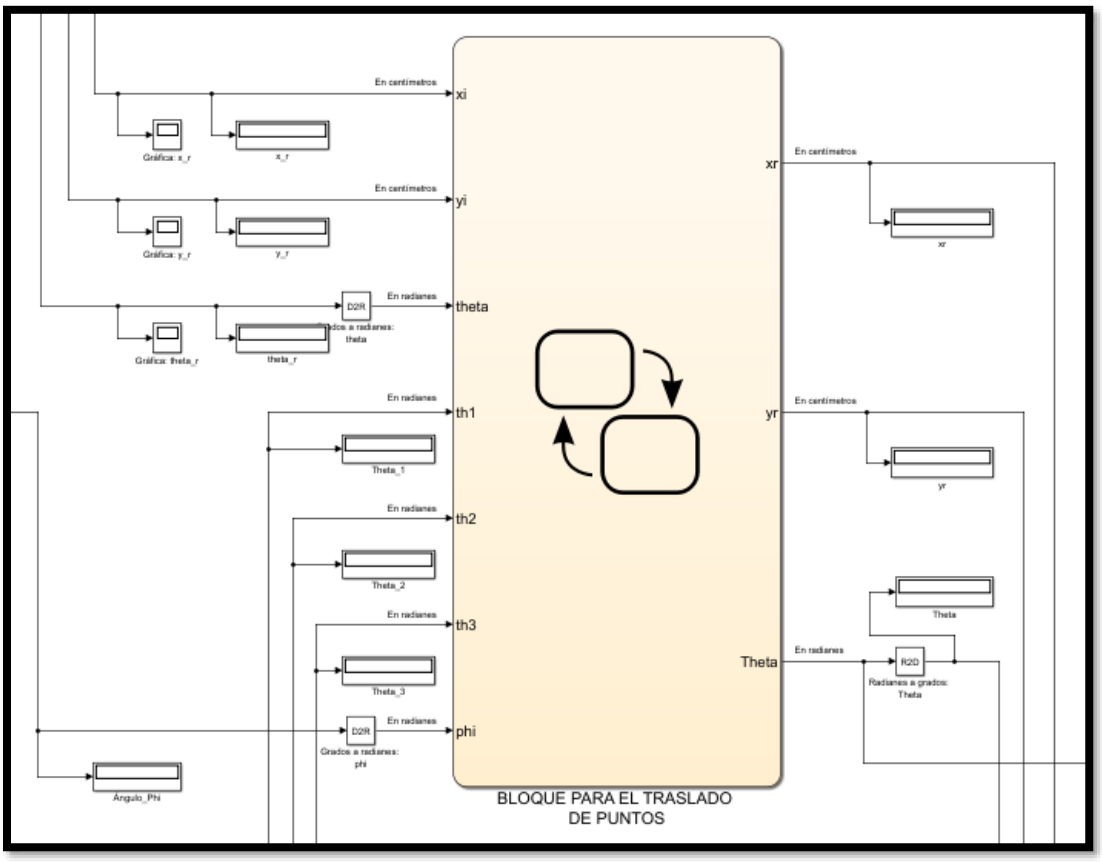


Figura C.11.- Bloque para el traslado de puntos vista exterior

- Vista interior

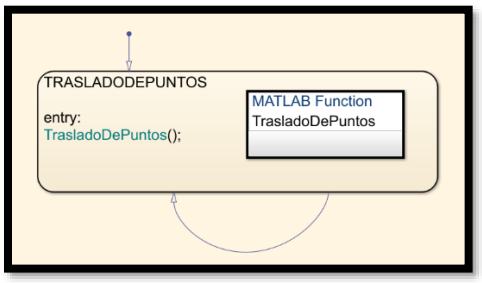


Figura C.12.- Bloque para el traslado de puntos vista interior

- Código programado en el bloque: TrasladoDePuntos

```
function TrasladoDePuntos
%% Parámetros del manipulador móvil

L1  = 13;
L2  = 12.6;
L3  = 18;
Xbc = 1.8;   %Distancia del centro de la plataforma al centro del brazo
Ybc = 0;

c1 = cos(pi - th2 - th3);
c2 = cos(th2);
c3 = cos(phi);

c0 = cos(th1);
s0 = sin(th1);

c = cos(theta);
s = sin(theta);

%% Acoplamiento

la = L2*c1 + L3*c3 - L1*c2;

xrc = Xbc + la*c0;
yrc = Ybc + la*s0;

xr = xi + xrc*c - yrc*s;   %Salida xr
yr = yi + xrc*s + yrc*c;   %Salida yr

Theta = theta;             %Salida theta
```

C.6 Bloque llegando al objetivo

- Vista exterior

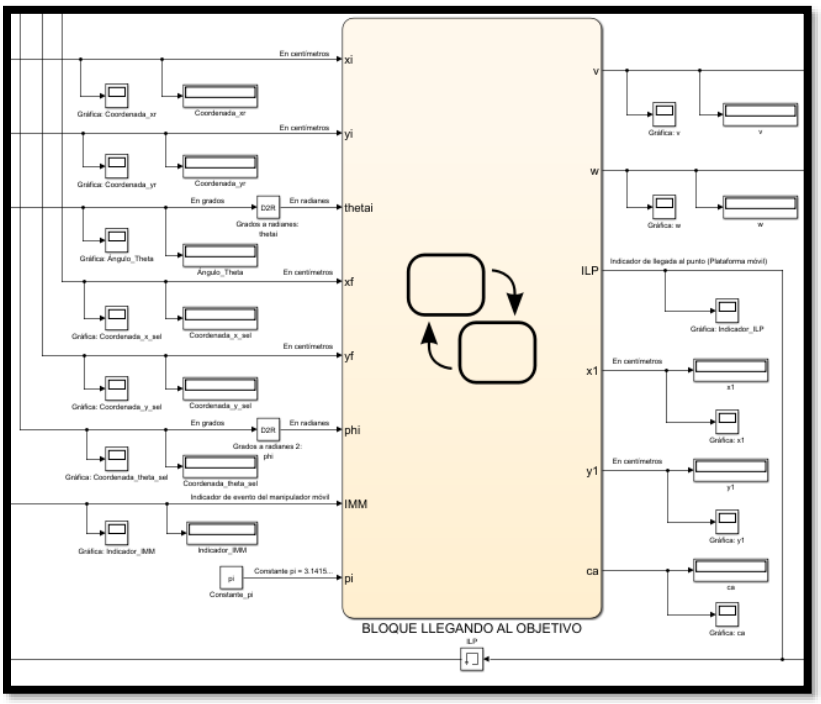


Figura C.13.- Bloque llegando al objetivo vista exterior

- Vista interior

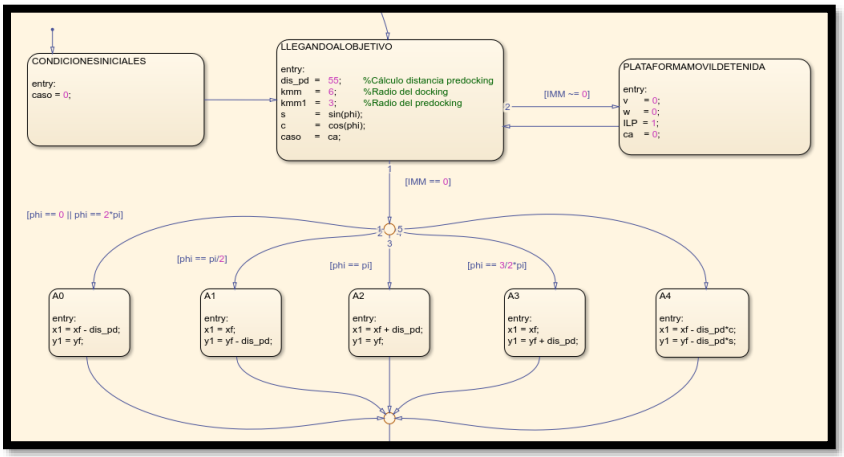


Figura C.14.- Bloque llegando al objetivo vista interior parte 1

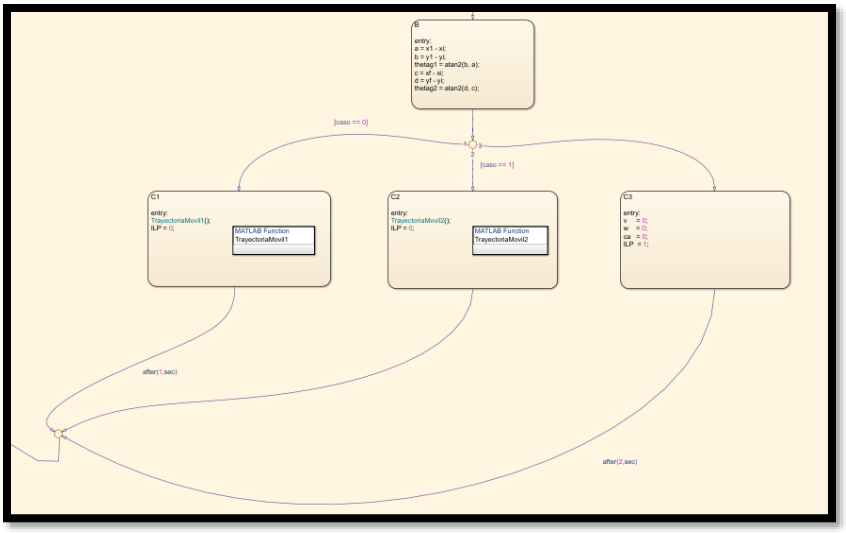


Figura C.15.- Bloque llegando al objetivo vista interior parte 2

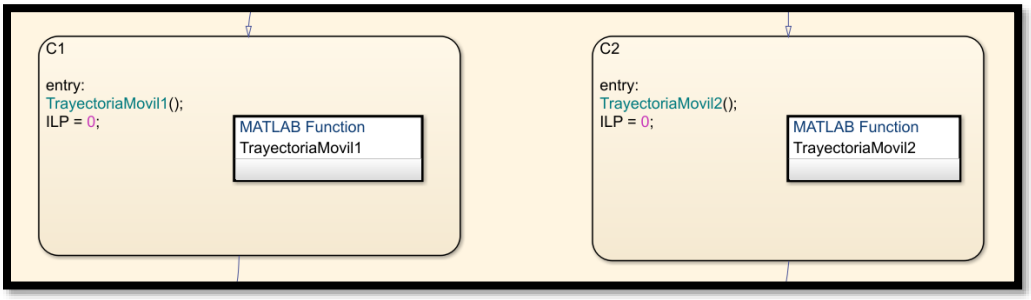


Figura C.16.- Bloque llegando al objetivo vista interior parte 3

- Código programado en el bloque: TrayectoriaMovil1

```
function TrayectoriaMovil1
%%Llegando a la meta por campos potenciales

Kr    = 10;  %10
vmax = 4;    %4
wmax = 0.4; %0.4

a       = x1 - xi;
b       = y1 - yi;
dg      = norm([a b]);
thetae = thetag1 - thetai;
s       = sin(thetae);

if(dg <= kmm1)
   v    = 0;
```

```
   w    = 0;
   ca   = 1;
else
    ca = 0;
    if(dg > Kr)
        v = vmax;
        w = wmax*s;
    elseif(dg <= Kr)
        v = (vmax/Kr)*dg;
        w = wmax*s;
    end
end
```

- Código programado en el bloque: TrayectoriaMovil2

```
function TrayectoriaMovil2
%%Llegando a la meta por campos potenciales

Kr   = 10;  %10
vmax = 4;   %4
wmax = 0.4; %0.4

a      = xf - xi;
b      = yf - yi;
dg     = norm([a b]);
thetae = thetag2 - thetai;
s      = sin(thetae);

if(dg <= kmm)
   v    = 0;
   w    = 0;
   ca   = 2;
else
    ca = 1;
    if(dg > Kr)
        v = vmax;
        w = wmax*s;
    elseif(dg <= Kr)
        v = (vmax/Kr)*dg;
        w = wmax*s;
    end
end
```

C.7 Bloque cinemático de la plataforma móvil

- Vista exterior

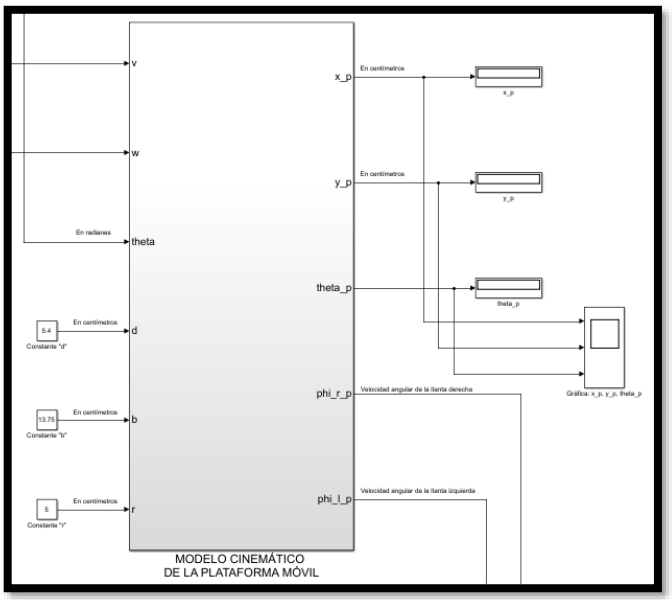


Figura C.17.- Bloque cinemático de la plataforma móvil vista exterior

- Vista interior

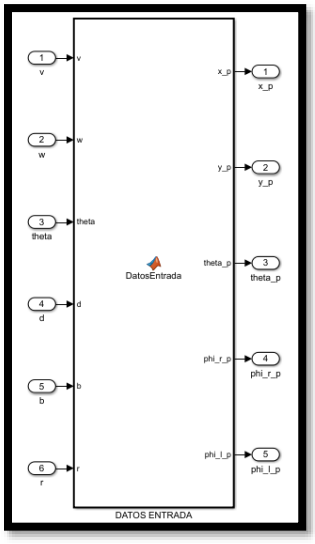


Figura C.18.- Bloque cinemático de la plataforma móvil vista interior

- Código programado en el bloque: DATOS ENTRADA

```
function [x_p, y_p, theta_p, phi_r_p, phi_l_p] = DatosEntrada(v, w, theta, d,
b, r)

U = [v;w];

A = cos(theta);
B = sin(theta);
C = (1/r);

Sq = [A -d*B; B d*A; 0 1; C b*C; C -b*C];

Xp = Sq * U;

x_p     = Xp(1,1);
y_p     = Xp(2,1);
theta_p = Xp(3,1);
phi_r_p = Xp(4,1);
phi_l_p = Xp(5,1);
```

C.8 Cambio a MD25

- Vista exterior

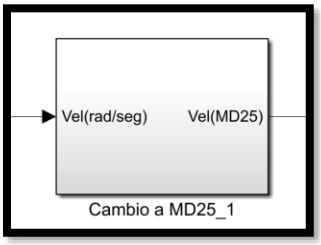


Figura C.19.- Bloque cambio a MD25_1 vista exterior

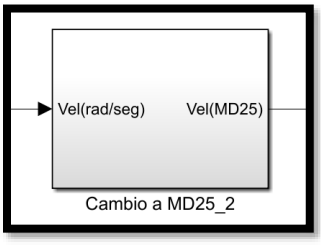


Figura C.20.- Bloque cambio a MD25_2 vista exterior

- Código programado en el bloque: Cambio a MD25_1

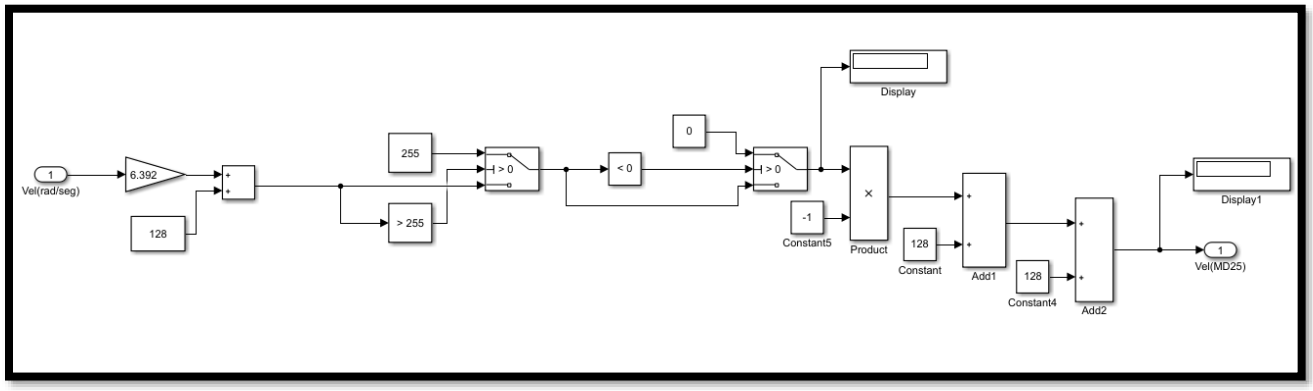


Figura C.21.- Bloque cambio a MD25_1 vista interior

- Código programado en el bloque: Cambio a MD25_2

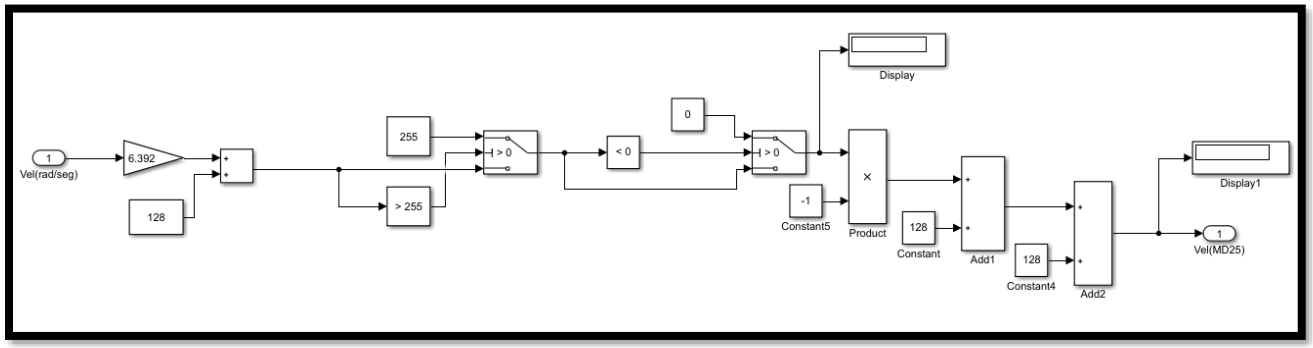


Figura C.22.- Bloque cambio a MD25_2 vista interior

C.9 Bloque MD25

- Vista exterior

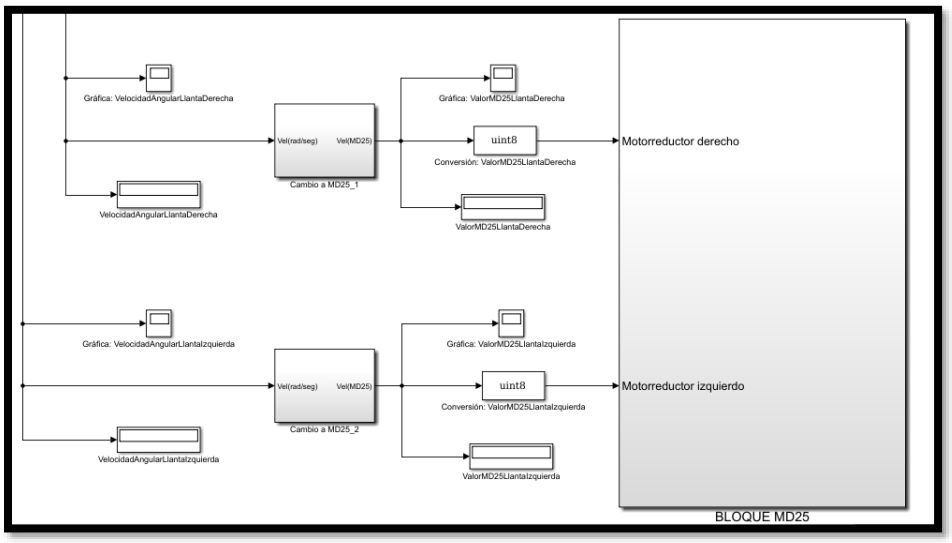


Figura C.23.- Bloque MD25 vista exterior

- Vista interior

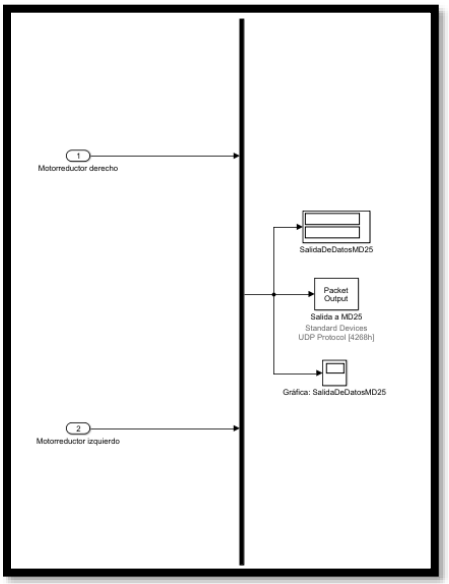


Figura C.24.- Bloque MD25 vista interior

C.10 Bloque cinemática inversa

- Vista exterior

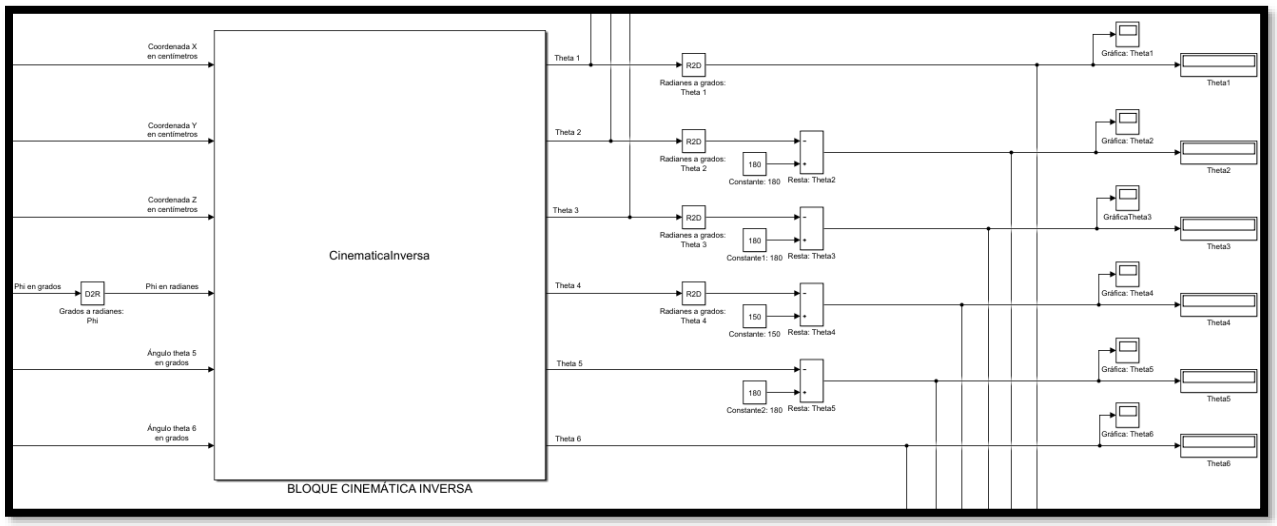


Figura C.25.- Bloque cinemática inversa vista exterior

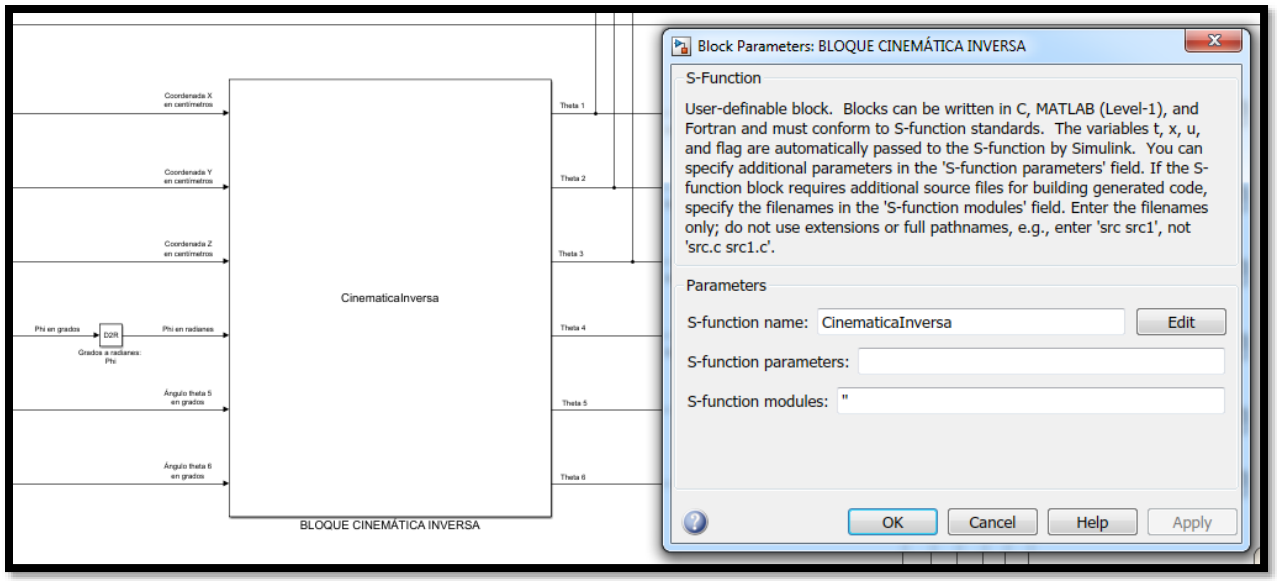


Figura C.26.-Bloque cinemática inversa vista exterior, parámetros

- Código programado en el bloque: CINEMÁTICA INVERSA

```
/*
 * File : CinematicaInversa.c
 * Abstract:
 *       An example C-file S-function for multiplying an input by 2,
 *       y  = 2*u
 *
 * Real-Time Workshop note:
 *   This file can be used as is (noninlined) with the Real-Time Workshop
 *   C rapid prototyping targets, or it can be inlined using the Target
 *   Language Compiler technology and used with any target. See
 *     matlabroot/toolbox/simulink/blocks/tlc_c/timestwo.tlc
 *   the TLC code to inline the S-function.
 *
 * Copyright 1990-2013 The MathWorks, Inc.
 */


#define S_FUNCTION_NAME  CinematicaInversa     //Verificar que coincida con
el nombre del archivo .c
#define S_FUNCTION_LEVEL 2

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


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

    if (!ssSetNumInputPorts(S, 6)) return;     //Se declara el número de
entradas al sistema, en este caso 6
    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,6)) return;     //Se declara el número de
salidas del sistema, en este caso 6
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 1, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 2, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 3, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 4, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 5, DYNAMICALLY_SIZED);

    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);     //Hereda el tiempo de
muestreo
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}
/* Function: mdlOutputs
=======================================================
 * Abstract:
 *    y = 2*u
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    //Declaración de variables
    float   xp, yp, zp, fi, theta5, theta6;
    float   d1, a2, a3, a4;
    float   cosFi, sinFi, x_c, z_c, r, R, sin_alfa, cos_alfa, sin_beta,
cos_beta, sin_gamma, cos_gamma;
    float   s1, c1, s2, c2, s3, c3, s4, c4, theta2, costh2, sinth2;

    //Variables de entrada en este caso las coordenadas "x","y" y "z", así
como el valor de "fi", "theta 5" y "theta 6"
    InputRealPtrsType  uPtrs0 = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType  uPtrs1 = ssGetInputPortRealSignalPtrs(S,1);
    InputRealPtrsType  uPtrs2 = ssGetInputPortRealSignalPtrs(S,2);
    InputRealPtrsType  uPtrs3 = ssGetInputPortRealSignalPtrs(S,3);
    InputRealPtrsType  uPtrs4 = ssGetInputPortRealSignalPtrs(S,4);
    InputRealPtrsType  uPtrs5 = ssGetInputPortRealSignalPtrs(S,5);
```

```
    //Variables de salida, en este caso los ángulos de theta 1 a theta 6
    real_T            *t1  = ssGetOutputPortRealSignal(S,0);
    real_T            *t2  = ssGetOutputPortRealSignal(S,1);
    real_T            *t3  = ssGetOutputPortRealSignal(S,2);
    real_T            *t4  = ssGetOutputPortRealSignal(S,3);
    real_T            *t5  = ssGetOutputPortRealSignal(S,4);
    real_T            *t6  = ssGetOutputPortRealSignal(S,5);

    //Asignación de variables a las entradas
    xp     = *uPtrs0[0];
    yp     = *uPtrs1[0];
    zp     = *uPtrs2[0];
    fi     = *uPtrs3[0];
    theta5 = *uPtrs4[0];
    theta6 = *uPtrs5[0];

    //Parámetros del brazo manipulador
    d1  = 20.7;  //Con 20.7 llega hasta el piso
    a2  = 13;    //13
    a3  = 12.6;  //12.6
    a4  = 18;    //18


//*********************************************************************************
*******************************

    sinFi = sin(fi);
    cosFi = cos(fi);
    r = sqrt(pow(xp,2) + pow(yp,2));
    x_c = r - (a4 * cosFi);
    z_c = zp + (a4 * sinFi);
    R = sqrt(pow(x_c,2) + pow((z_c-d1),2));
    sin_alfa = (z_c-d1)/R;
    cos_alfa = x_c/R;
    cos_beta = (pow(R,2) + pow(a2,2) - pow(a3,2)) / (2*a2*R);
    sin_beta = sqrt(1 - pow(cos_beta,2));
    cos_gamma = (pow(a3,2) + pow(a2,2) - pow(R,2)) / (2*a2*a3);
    sin_gamma = sqrt(1 - pow(cos_gamma,2));

    //_____________________________________________CÁLCULOS CINEMÁTICA
INVERSA_____________________________________________

    s1 = yp;
    c1 = xp;

    *t1 = atan2(s1, c1);     //Theta 1


//_____________________________________________________________________________
_____________________________________

    s2 = (sin_alfa*cos_beta) + (cos_alfa*sin_beta);
    c2 = -((cos_alfa*cos_beta) - (sin_alfa*sin_beta));
```

```
    *t2 = atan2(s2,c2);      //Theta 2
    theta2 = *t2;


//_____________________________________________________________________________
_____________________________

    s3 = sin_gamma;
    c3 = -1*cos_gamma;

    *t3 = atan2(s3,c3);      //Theta 3


//_____________________________________________________________________________
_____________________________

    sinth2 = sin(theta2);
    costh2 = cos(theta2);

    s4  = sin(fi)*((cos_gamma*costh2)+(sin_gamma*sinth2)) +
cos(fi)*((sin_gamma*costh2)-(cos_gamma*sinth2));
    c4  = cos(fi)*((cos_gamma*costh2)+(sin_gamma*sinth2)) -
sin(fi)*((sin_gamma*costh2)-(cos_gamma*sinth2));

    *t4 = atan2(s4,c4);      //Theta 4


//_____________________________________________________________________________
_____________________________

    *t5 = theta5;      //Theta 5, variable para la orientación del efector
final


//_____________________________________________________________________________
_____________________________

    *t6 = theta6;      //Theta 6, ángulo de abertura y cierre del gripper

}

/* 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
```

C.11 Coordinación del brazo serial

- Vista exterior

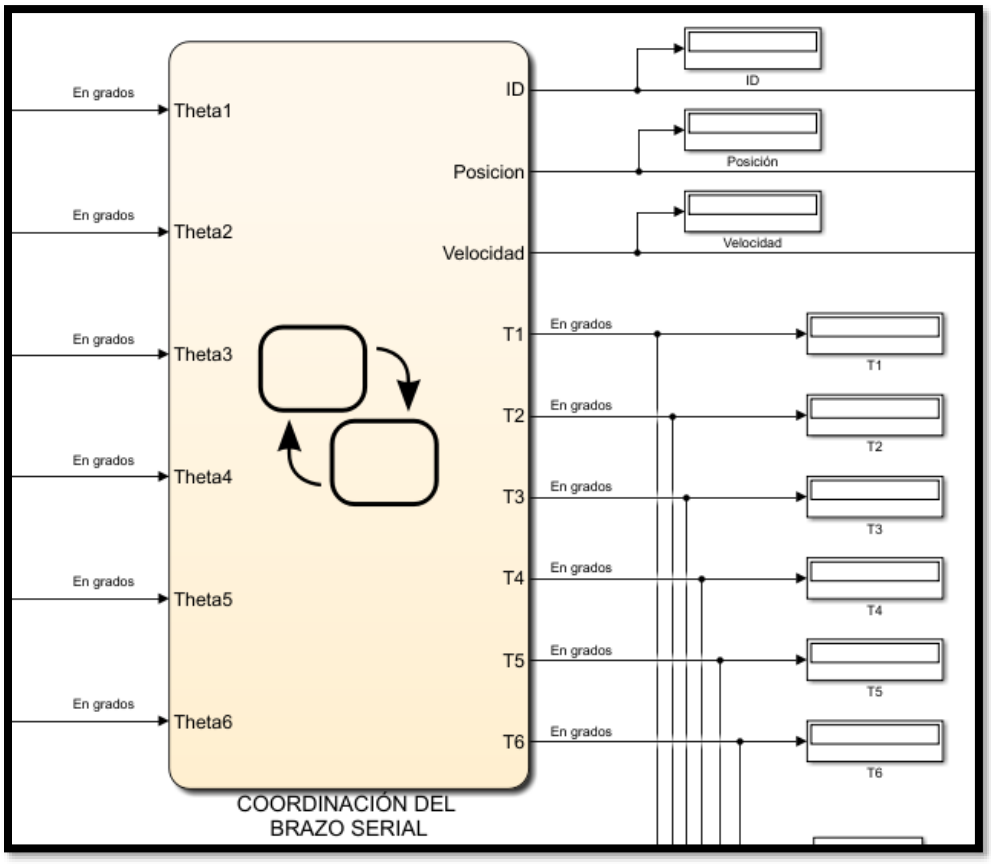


Figura C.27.- Coordinación del brazo serial vista exterior

- Vista interior

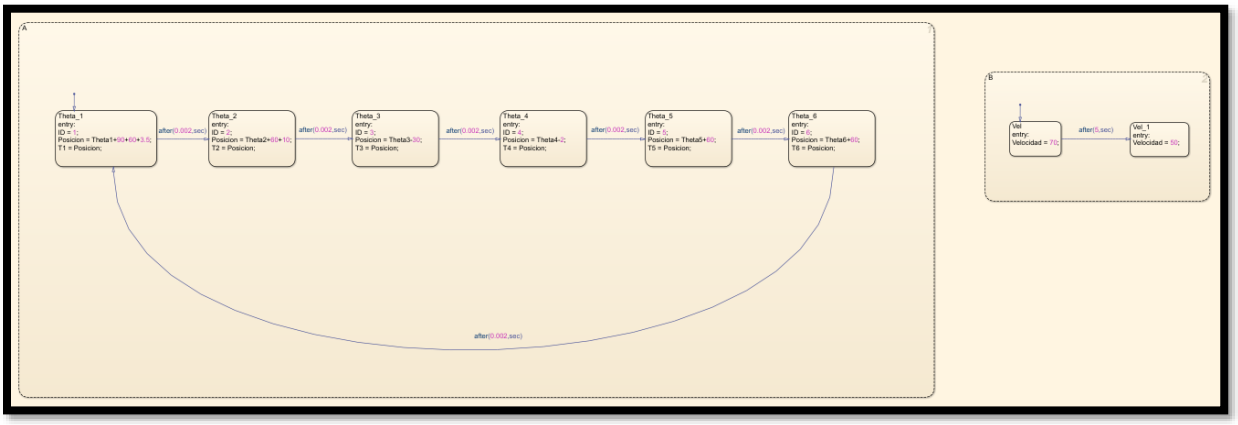


Figura C.28.- Coordinación del brazo serial vista interior

C.12 Bloque Dynamixel

- Vista exterior

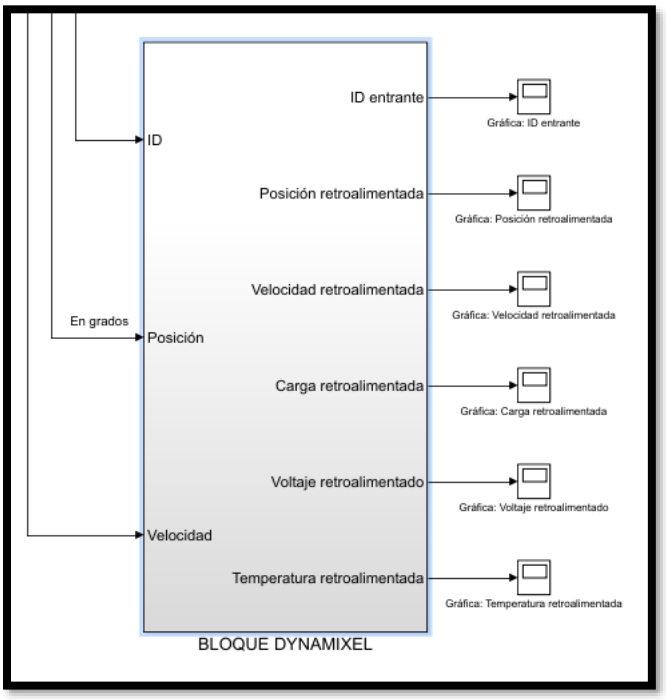


Figura C.29.- Bloque Dynamixel vista exterior

- Vista interior

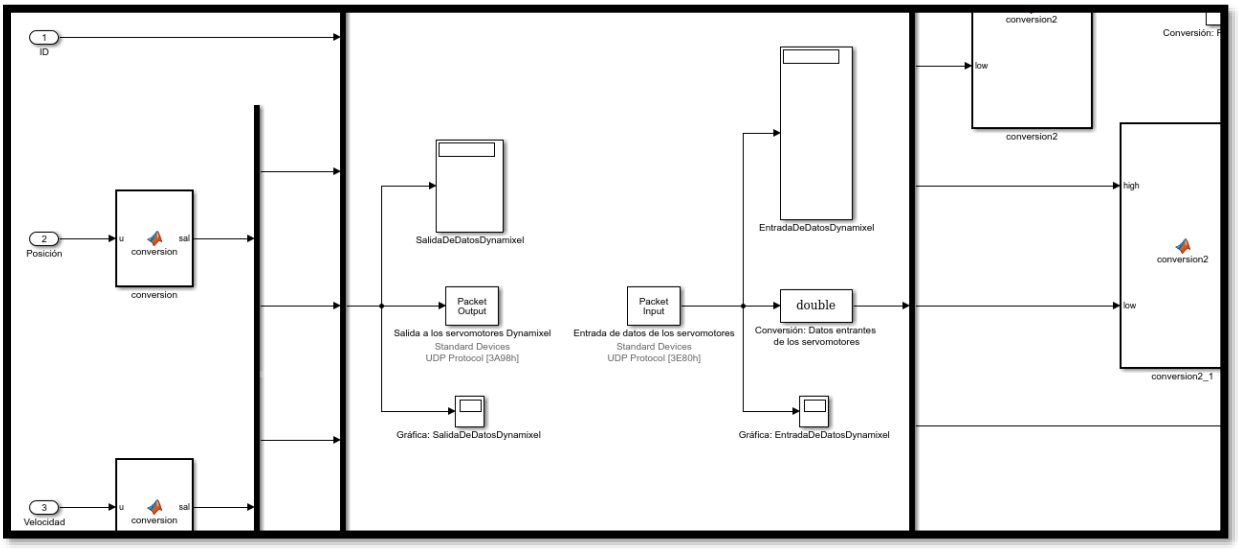


Figura C.30.- Bloque Dynamixel vista interior

- Código programado en el bloque: conversión

```
function sal = conversion(u)

dat = uint16(u);
mas1 = uint16(255);

low = double(bitand(dat, mas1));
high = double(bitand(swapbytes(dat), mas1));

sal = [high; low];
```

- Código programado en el bloque: conversión2

```
function sal = conversion2(high, low)

dato = double(uint16(high) * uint16(256)) + uint16(low);

sal = dato;
```

- Código programado en el bloque: velocidad

```
unction sal = velocidad(dato)

%velocidad = dato * 0.111;      %Conversión para 114 rpm
velocidad = dato * 0.05767350; %El servomotor funciona a 59 rpm sin carga,
con 12 volts de alimentación

sal = velocidad;
```

- Código programado en el bloque: carga

```
function sal = carga(dato)

sentido = 0;

if dato <= 1023
    dato = (dato*100)/1023;
    sentido = 1;  %Sentido antihorario
end

if dato > 1023
    dato = dato - 1024;
    dato = (dato*100)/1023;
    sentido = 2;  %Sentido horario
end

sal = [dato; sentido];
```

- Código programado en el bloque: voltaje

```
function sal = voltaje(dato)

dato = dato/10;

sal = dato;
```

Apéndice D.- Configuración de Matlab en tiempo real

En las siguientes figuras se muestran los parámetros configurados en Simulink para poder establecer la comunicación en tiempo real.

En la figura D.1 se señala con el marco de color verde la opción que se debe elegir en el software, comúnmente esta opción por defecto se encuentra en modo normal, para el desarrollo de los experimentos se debe configurar en modo “External”.

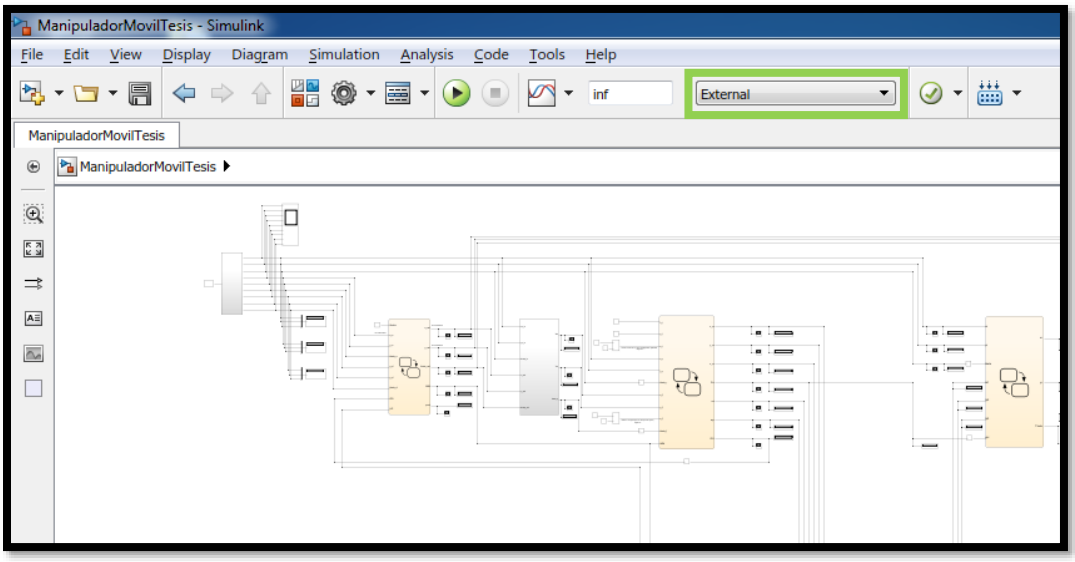


Figura D.1.- Configuración del sistema en modo externo

En la pestaña “Commonly Used Parameters” de “Configuration parameters” se deben configurar los siguientes parámetros (ver figura D.2):

- Simulation time: Start time: 0, Stop time: inf
- Solver options: Type: Fixed-step, Solver: Ode5 (Dormand-Prince)
- Additional options: Fixed-step size (fundamental sample time): 0.001 y elegir Treat each discrete rate as a separate task.

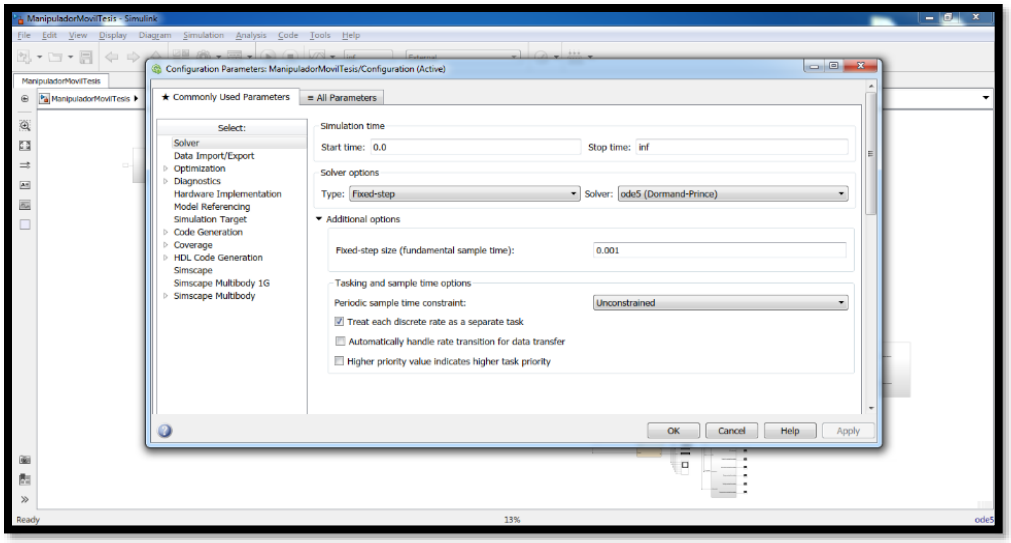


Figura D.2.- Configuración de los parámetros para la simulación en tiempo real

En la pestaña “Commonly Used Parameters” elegir el submenú “Code Generation” y configurar los siguientes parámetros (ver figura D.3 y D.4):

- System target file: sldrt.tlc
- Language: C

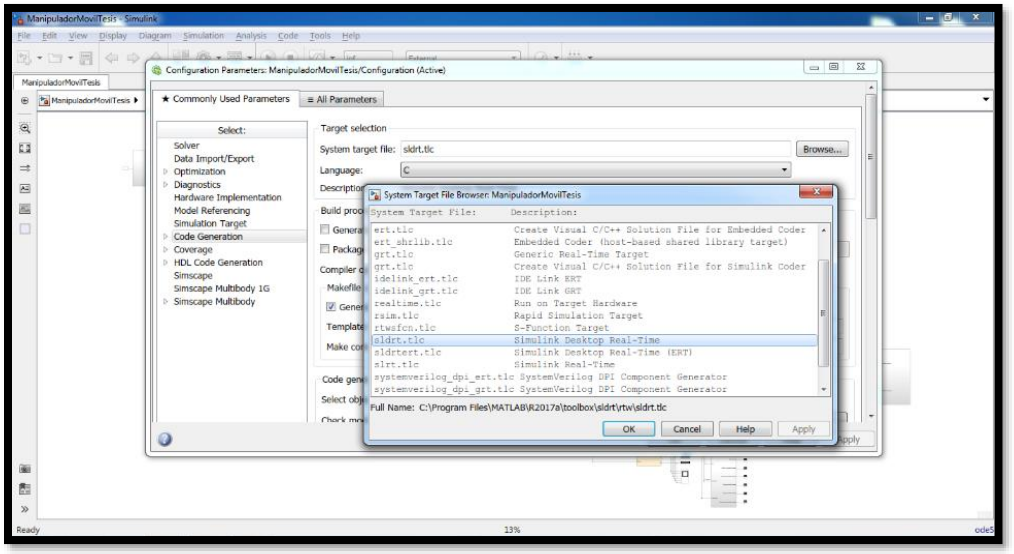


Figura D.3.- Configuración de los parámetros para la simulación en tiempo real y generación de código en lenguaje C, parte 1

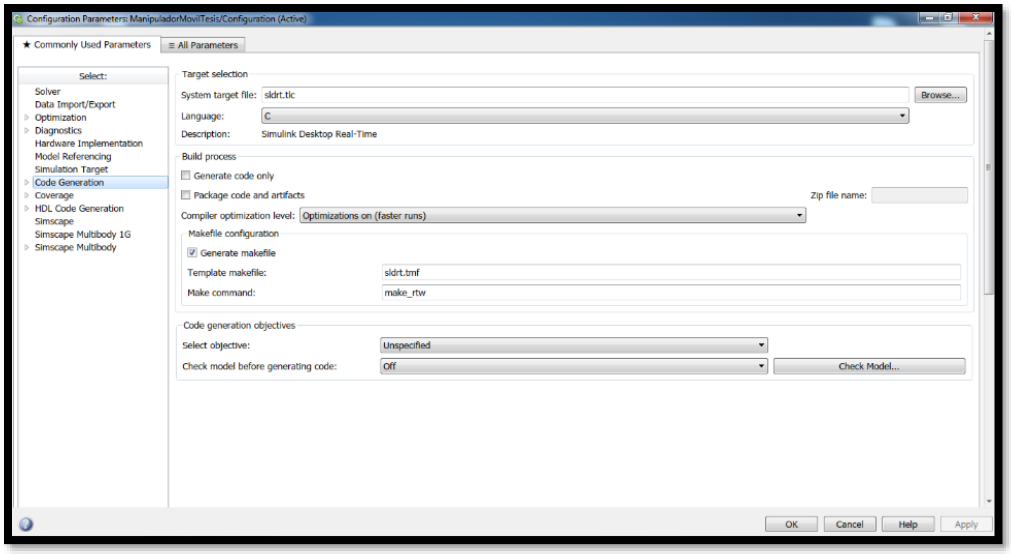


Figura D.4.- Configuración de los parámetros para la simulación en tiempo real y generación de código en lenguaje C, parte 2

REFERENCIAS

Anderson. (2007). *Artificial Intuition: A new possible path to artificial intelligence*. Obtenido de http://artificial-intuition.com/anderson.html

Arduino. (22 de Diciembre de 2013). Recuperado el 12 de Octubre de 2017, de Wikipedia la enciclopedia libre: https://es.wikipedia.org/wiki/Arduino

Arduino Mega. (10 de Febrero de 2011). Recuperado el 2 de Noviembre de 2017, de ARDUINO: https://www.arduino.cc/en/Main/arduinoBoardMega

Arduino WiFi shield A000058. (1 de Enero de 2009). Recuperado el 23 de Enero de 2018, de HETPRO: https://hetpro-store.com/arduino-wifi-shield-r3/

AX Series. (9 de Julio de 2014). Recuperado el 14 de Diciembre de 2017, de Robotis: http://www.robotis.us/ax-series/

Baturone Ollero , A. (2001). *Robótica manipuladores y robots móviles.* Barcelona: Marcobo.

Carlos Ávila, P. (2015). *Espacios inteligentes con manipuladores móviles dotados de intuición artificial.* Tesis de licenciatura, Universidad Nacional Autónoma de México, Ciudad de México.

Casacuberta Sevilla, D. (2001). *La mente humana.* Oceano.

Castañeda Espinosa, J., & Pedro Mendoza, E. E. (2016). *Diseño de un banco de pruebas para un robot serial adaptable.* Tesis de licenciatura, Universidad Nacional Autónoma de México, Ciudad de México.

Castelli, G., Ottaviano, E., & Ceccarelli, M. (2008). A fairly general algorithm to evaluate workspace characteristics of serial and parallel manipulators. *Mechanics based design of structures and machines, 36*(1), 14-33.

Corrales Navarro, E. (2010). La intuición como proceso cognitivo. *Comunicación, 19*(2), 33-42.

Craig, J. J. (2006). *Introduction to robotics: Mechanics and Control.* Prentice Hall.

Cruz López, I. C. (2015). *Manipulador móvil omnidireccional redundante: coordinación de movimientos en ambientes inteligentes.* Tesis de maestría, Universidad Nacional Autónoma de México, Ciudad de México.

Díaz Hernández, O. (2014). *Intuición artificial aplicada a la teleoperación.* Tesis doctoral, Universidad Nacional Autónoma de México, Ciudad de México.

Díaz Hernández, O., & González Villela, V. J. (2015). Analysis of human intuition towards artificial intuition synthesis for robotics. *Mechatronics and Applications: An International Journal (MECHATROJ), 1*(1), 23-39.

Dijksterhuis, A., Bos, M., Nordgren, L., & van Baaren, R. (2006). On making the right choice: the deliberation-without-attention effect. *Science*, *311*(5763), 1005-7.

Dundas J., & D. Chik. (2013). Machine implementation of human-like intuition mechanism in Artificial Intelligence. *ICIC Express Letters*, 2231-2235.

Erlhoff, M., Marshall, T., & Becker, S. (2008). *Design Dictionary.* Basel: Birkhäuser.

Escobedo Castillo, P. M. (2012). *Manipulador móvil: estudio sobre la coordinación de movimientos de un manipulador serial acoplado.* Tesis de licenciatura, Universidad Nacional Autónoma de México, Ciudad de México.

Ferrater Mora, J. (1984). *Diccionario de Filosofía.* Barcelona: A. Diccionarios.

Fu, K., González, R., & Lee, C. (1989). *Robótica: Control, detección, visión e inteligencia.* México: Mc Graw Hill.

G. Myers, D. (2008). El poder y los peligros de la intuición. *Mente y cerebro*(33), 22-29.

Gálvez Valadez, P. (2015). *Desarrollo de un banco de pruebas para el estudio de robots móviles mediante el protocolo IEEE 802.11 en tiempo real y con retroalimentación visual.* Tesis de maestría, Universidad Nacional Autónoma de México, Ciudad de México.

Gigerenzer, G. (2008). *Decisiones instintivas: La inteligencia del inconsciente.* Barcelona, España: Ariel.

Goldstein, E. (2005). *Cognitive psychology: Connecting mind, research, and everyday experience.* Australia Belmont: Thomson/Wadsworth.

González Villela, V. J. (2006). *"Research on a semiautonomous mobile robot for loosely structured environments focused on transporting mail trolleys".* Tesis doctoral, Loughborough University, United Kingdom.

Gutiérrez Pulido, H., & de la Vara Salazar, R. (2008). *Análisis y diseño de experimentos.* México: McGraw-Hill.

Harteis, C., Koch, T., & Mergenthaler, B. (2008). How intuition contributes to high performance: an educational perspective. *US-China Education Review, 5*(1), 68-80.

Hassin, R.R., J.S. Uleman, & J.A. Bargh. (2005). *The new unconscious.* The Oxford University Press: T.O.U. Press.

Hernández Calderón, I. (2016). *Manipulador móvil omnidireccional, su coordinación de movimientos en ambientes inteligentes.* Tesis de licenciatura, Universidad Nacional Autónoma de México, Ciudad de México.

Hogarth, R. (2001). *Educating intuition.* Chicago: University of Chicago.

Hogarth, R. (2002). Deciding analytically or trusting your intuition?: The advantages and disadvantages of analytic and intuitive thought. *UPF Economics and business working*(654), 1-46.

-ica. (16 de Febrero de 2003). Recuperado el 5 de Marzo de 2016, de Raíces griegas: sufijos: http://etimologias.dechile.net/griego/?Sufijos

K. S. Fu, R. C. González, & C. S. G. Lee. (1994). *Robótica: Control, detección, visión e inteligencia.* México: McGraw Hill.

Kahneman, D. (2003). A perspective on judgment and choice. *American Psychologist, 58*(9), 697-720.

Kahneman, D. (2012). *Pensar rápido, pensar despacio.* España: Penguin random house grupo editorial.

Kant, I. (2002). *Crítica de la razón pura.* Tecnos.

Kazuyuki Morioka, Joo-Ho Lee, & Hideki Hashimoto. (2008). Intelligent Space for human Centered Robotics. *Advances in service robotics*, 182-192.

Krippendorff, A. (1986). *A dictionary of cybernetics.* Philadelphia: Norfolk VA.

Laplante, P. A. (2004). *Real-time systems design and analysis.* Wiley.

Latombe, J.C. (1991). *Robot Motion Planning.* USA: Kluwer.

Lei, S., X. Mingheng, and L. Bo. (2012). A fast algorithm for workspace of a robotic. *Advanced Materials Research*, 538-541.

Logitech. (19 de Abril de 2013). *Logitech*. Recuperado el 20 de Diciembre de 2017, de HD PRO WEBCAM C920: https://www.logitech.com/es-mx/product/hd-pro-webcam-c920?crid=34

M. Manubhai, K. S. Huang, & I. Mikic. (2005). Dynamic Context Capture and Distributed Video Arrays. *IEEE transactions on systems, man, and cybernetics part a: systems and humans, 35*(1).

MATLAB. (19 de Septiembre de 2015). Recuperado el 8 de Diciembre de 2017, de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/MATLAB

MD25 - Dual 12Volt 2.8Amp H Bridge Motor Drive. (3 de Diciembre de 2009). Recuperado el 19 de Enero de 2018, de Robot electronics: https://www.robot-electronics.co.uk/htm/md25tech.htm

Mente. (5 de Mayo de 2007). Recuperado el 17 de Enero de 2016, de Wikipedia, la enciclopedia libre: http://es.wikipedia.org/wiki/Mente

Morsella, E., & Poehlman, T. (2013). The inevitable contrast: conscious vs. unconscious processes in action control. *Front Psychol, 4*, 590.

Nehmzow, U. (2003). *Mobile Robotics a practical introduction.* London: Springer.

Pacheco Jiménez, A. J., & Ávila García, A. (2017). *Manipulador móvil híbrido autónomo: coordinación de movimientos para el transporte de materiales.* Tesis de licenciatura, Universidad Nacional Autónoma de México, Ciudad de México.

Pérez Cisneros, M., Cuevas Jiménez, E., & Zaldívar Navarro, D. (2015). *Fundamentos de robótica y mecatrónica con Matlab y Simulink.* México: Alfaomega.

PhantomX Parallel AX-12 Gripper. (10 de Septiembre de 2012). Recuperado el 12 de Enero de 2018, de Trossen Robotics: http://www.trossenrobotics.com/p/phantomx-parallel-ax12-gripper.aspx

Ponce Cruz, P. (2010). *Inteligencia artificial con aplicaciones a la ingeniería.* D.F., México: Alfaomega grupo editor S.A. de C.V.

RD02 - 12v robot drive. (3 de Diciembre de 2009). Recuperado el 10 de Enero de 2018, de Robot electronics: http://www.robot-electronics.co.uk/products/drive-systems/drive-kits/rd02-12v-robot-drive.html

Reactivision Engine. (9 de Septiembre de 2014). Recuperado el 17 de Enero de 2018, de HKU MAP LAB: http://maplab.hku.nl/tool/fiducial-engine/

Researchers add a splash of human intuition to planning algorithms. (7 de Febrero de 2017). Recuperado el 7 de Enero de 2018, de MIT News: http://news.mit.edu/2017/human-intuition-planning-algorithms-0207

Roboplus 1.0. (23 de Junio de 2016). Recuperado el 19 de Enero de 2018, de ROBOTIS: http://www.robotis.us/roboplus1/

Robótica. (4 de Octubre de 2014). Recuperado el 21 de Febrero de 2016, de Real Academia Española: http://dle.rae.es/?id=WYTm4uf

Robotics, T. (30 de Noviembre de 2012). *WidowX Robot Arm*. Obtenido de Trossen Robotics: http://www.trossenrobotics.com/widowxrobotarm

ROBOTIS e-Manual v1.31.30. (17 de Febrero de 2010). Recuperado el 27 de Diciembre de 2017, de ROBOTIS: http://support.robotis.com/en/techsupport_eng.htm#product/manipulator/getting_started/preparation.htm

Saha, S. K. (2008). *Introducción a la Robótica.* México: McGraw Hill.

Savage, J. A. (22 de Enero de 2011). *Arduino y Dynamixel AX-12*. Recuperado el 14 de Febrero de 2017, de Savage electronics: http://savageelectronics.blogspot.mx/2011/01/arduino-y-dynamixel-ax-12.html

Seligman, & Kahana, M. (2009). Unpacking intuition: a conjecture. *Perspective psycholgy science, 4*(4), 399-402.

Simon, H., & Frantz, R. (2003). Artificial intelligence as a framework for understanding intuition. *Journal of economic psychology, 24*, 265-277.

Simulink. (1 de Enero de 1996). Recuperado el 18 de Junio de 2017, de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Simulink

SMPS2DYNAMIXEL. (9 de Septiembre de 2014). Recuperado el 2017 de Noviembre de 15, de ROBOTIS: http://www.robotis.us/smps2dynamixel/

Stateflow. (1 de Enero de 2000). Recuperado el 13 de Julio de 2017, de MathWorks: https://es.mathworks.com/products/stateflow.html

Thagard, P. (2008). *La mente, introducción a las ciencias cognitivas.* Madrid, España: Katz editores.

User Datagram Protocol. (11 de Marzo de 2015). Recuperado el 10 de Septiembre de 2017, de Wikipedia la enciclopedia libre: https://es.wikipedia.org/wiki/User_Datagram_Protocol

Van Gulick, R. (2017). *Consciousness* (Summer 2017 ed.). (S. U. Metaphysics Research Lab, Ed.) Edward N. Zalta. Obtenido de https://plato.stanford.edu/archives/sum2017/entries/consciousness

Vargas Medina, J. (1 de Septiembre de 2004). *Tipos de inteligencia*. Recuperado el 22 de Noviembre de 2016, de CSMV Emprendedor: http://www.ingenieria.unam.mx/~guiaindustrial/entorno/menu6.htm

Weidong, & H. Ping. (2009). Intuitive Learning and Artificial Intuition Networks. in Second International Conference on Education Technology and Training. *ETT '09*.