



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

**MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA
OPCIÓN EN AUTOMATIZACIÓN**

**“Implementación de algoritmos de consenso en un sistema multiagente de
robots manipuladores”**

T E S I S

Presentada para obtener el título de:
Maestro en Ciencias de la Electrónica, Opción en Automatización

Presenta:

Ing. José Domingo Pájaro Adrián*

Directores:

Dr. Jaime Julián Cid Monjaraz

Dr. Rodolfo Reyes Báez (RUG**)

Dr. Fernando Reyes Cortés (FCE-BUAP)

Puebla, México

ENERO 2020

*Becario CONACYT.

** University of Groningen.

Agradecimientos

Agradezco a Dios, a Maribel por todo su apoyo y comprensión quien en todo momento me animo a continuar y concluir este trabajo, a José Mateo por las horas que no te dedique te agradezco, a mis padres Hilario Pájaro y Hilaria Adrián por todo su apoyo, a mi hermano Anastasio por su buen ejemplo y apoyo, a Carolina por su invaluable apoyo, a Rosario y a Hilario por su apoyo.

A mis directores de tesis, Dr. Jaime Julián Cid Monjaraz, Dr. Rodolfo Reyes Báez y al Dr. Fernando Reyes Cortés por las enseñanzas y conocimientos que me compartieron mismos que me permitieron concluir este trabajo de tesis.

A mi comité tutorial, Dra. María Aurora Diozcora Vargas Treviño, Dra. Amparo Dora Palomino Merino y al Dr. Arnulfo Luis Ramos, quienes con sus valiosas observaciones y consejos enriquecieron el contenido de este trabajo de tesis.

A la Dra. Olga Guadalupe Félix Beltrán coordinadora de la MCEA. A mis amigos Abril, Miguel, Daniel Clemente, Edwin, Carlos, Daniel y Carolina por todos sus consejos.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado para poder realizar mis estudios de maestría.

A la Benemérita Universidad Autónoma de Puebla (BUAP) y a la Facultad en Ciencias de la Electrónica (FCE), el lugar que ha sido mi casa de estudio y a la Maestría en Ciencias de la Electrónica opción Automatización, por darme las herramientas necesarias para el desarrollo de este trabajo de tesis.

Dedicatoria

Dedico esta tesis a mi amada Maribel por estar a mi lado por toda su paciencia, apoyo y comprensión, a mi hijo José Mateo. A mis padres Hilario e Hilaria por darme la vida, todas las enseñanzas y apoyo. A mis hermanos, Anastacio, Carolina, Rosario y Hilario por todo su apoyo. A todos los que me apoyaron para escribir y concluir esta tesis.

Para ellos es esta dedicatoria de tesis, pues es a ellos a quienes se las debo por su apoyo incondicional.

Resumen

En este trabajo de tesis se realiza la implementación de algoritmos no lineales de consenso en un sistema multiagente. El término “consenso” se refiere al acuerdo que se produce entre los miembros de un grupo o equipo, el término “sistema multiagente” se refiere a un grupo de individuos o agentes que normalmente tienen las mismas características, que son capaces de tomar decisiones de forma autónoma y pueden interactuar con otros miembros del equipo intercambiando información. En este trabajo se considera como agente a un robot manipulador de n grados de libertad que está conectado a través de un sistema de intercambio de información a otro(s) agente(s) para formar al sistema multiagente y cuyo objetivo de consenso es alcanzar una posición o trayectoria común.

Se presenta un breve estudio de sistemas no lineales ya que se requiere para realizar la implementación de controladores en sistemas cuya respuesta natural a la salida no es proporcional a la entrada, con lo que es posible diseñar y realizar el análisis de estabilidad y implementar algoritmos de control. Además, se realiza el estudio de teoría de grafos la cual está relacionada con el intercambio de información entre agentes, además, está estrechamente relacionada con el diseño de algoritmos de consenso y el estudio de estabilidad.

La implementación de algoritmos de consenso en una red de robots manipuladores requiere de instrumentación electrónica por lo que también se presenta la electrónica que se encarga de realizar el control del robot, así como la etapa de acondicionamiento de las señales de control que van desde la tarjeta de control a las entradas de control del robot. La etapa de la comunicación entre agentes es una de las características más importantes por lo que también se presenta la electrónica que se encargará de la transmisión de información entre agentes y el acoplamiento de estas señales entre emisor y receptor.

Los resultados obtenidos que se presentan en este trabajo son los obtenidos por simulación haciendo uso de software especializado y los resultados obtenidos por la implementación física de los algoritmos de consenso.

Índice general

Agradecimientos	I
Dedicatoria	II
Resumen	III
Índice de figuras	VII
Índice de tablas	X
1. Introducción	1
1.1. Planteamiento	1
1.1.1. Objetivo general	3
1.1.2. Objetivos Particulares	3
1.2. Estado del arte	4
1.3. Robot manipulador de transmisión directa	5
1.3.1. Articulaciones	5
1.3.2. Eslabones	6
1.3.3. Espacio de trabajo	7
1.3.4. Efector final	7
1.4. Modelo dinámico del robot manipulador	8
1.4.1. Propiedades del modelo dinámico	13
1.5. observaciones	14
2. Preliminares Matemáticos	15
2.1. Teoría de Grafos	15
2.2. Álgebra de teoría de grafos	19
2.2.1. Matriz de adyacencia	19
2.2.2. Matriz de grados	20
2.2.3. Matriz Laplaciana	21

2.2.4. Producto de Kronecker	22
2.3. Pasividad	23
2.4. Observaciones	24
3. Algoritmos de Consenso	26
3.1. Problema de consenso	26
3.2. Consenso en sistemas de integrador simple	28
3.3. Algoritmos de consenso para sistemas con integrador doble	30
3.4. Algoritmos de consenso para robots manipuladores	34
3.4.1. Algoritmo de consenso fundamental	35
3.4.2. Algoritmo de consenso no lineal	36
3.5. Observaciones	36
4. Instrumentación	37
4.1. Casos de estudio	37
4.1.1. Robot manipulador de transmisión directa	37
4.1.2. Tarjeta de adquisición de datos y control	40
4.2. Acoplamiento de señales de entrada y ajuste de señales de salida	41
4.2.1. Acoplamiento de los encoders	41
4.3. Adaptación de los convertidores digital-analógico de la tarjeta Arduino Due	42
4.4. Validación experimental de la instrumentación	46
4.5. Hardware in the loop	51
4.5.1. Validación experimental del sistema Hardware In the Loop	52
4.6. Observaciones	59
5. Resultados	60
5.1. Sistema multiagente de robots manipuladores	60
5.2. Implementación de los algoritmos de consenso	66
6. Conclusiones	77
Anexos	79
A. Códigos realizados, Capítulo 3	79
A.1. Unión implementada en Arduino DUE	79
A.2. Control de trayectoria: Implementación en Arduino DUE	80
A.3. Control de trayectoria: Modelo dinámico Robot de transmisión directa HIL	82

B. Códigos realizados, capítulo 5	84
B.1. Implementación en simulink	84
B.1.1. Script: Robot de transmisión directa 1	84
B.1.2. Script: Robot de transmisión directa 2	86
B.1.3. Script: Robot de transmisión directa 3	88
B.2. Implementación en Arduino Due	89
B.2.1. Sketch: Obtención, envió de datos y algoritmo de consenso para el robot	89
B.2.2. Sketch: Obtención, envió de datos y algoritmo de consenso para el robot	91
B.2.3. Sketch: Obtención, envió de datos y algoritmo de consenso para el robot	94
C. Publicaciones y participaciones en congreso	97
Bibliografía	125

Índice de figuras

1.1. Ejemplo de sistemas multiagente.	2
1.2. Diagrama a bloques general del sistema multiagente.	3
1.3. Robot manipulador de transmisión directa de 3 grados de libertad.	5
1.4. Servo motor de transmisión directa.	6
1.5. Cadena en cinemática abierta.	6
1.6. Eslabones del robot manipulador de transmisión directa.	7
1.7. Espacio de trabajo de un robot manipulador.	7
1.8. Diferentes efectores finales para robots manipuladores.	8
1.9. Robot manipulador de 2 grados de libertad.	9
2.1. Ejemplo de un grafo formado por siete nodos y ocho bordes.	15
2.2. Grafo con $V = \{1, 2, 3\}$ y $E = \{\{1, 2\}, \{2, 3\}, \{3, 1\}, \{3, 2\}\}$	16
2.3. Ejemplo de grafos dirigidos.	17
2.4. Ejemplo de grafos no dirigidos.	18
2.5. Ejemplo de un dígrafo y subgrafo.	18
2.6. Dígrafo con $V = \{1, 2, 3\}$ y $E = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$	19
2.7. Grafo no dirigido con $V = \{1, 2, 3\}$ y $E = \{\{1, 2\}, \{1, 3\}, \{2, 1\}, \{2, 3\}, \{3, 1\}, \{3, 2\}\}$	20
3.1. Topología de interacción para un sistema con tres agentes interconectados.	28
3.2. Consenso en un sistema de primer orden.	29
3.3. Topología de interacción para un sistema con tres agentes interconectados.	31
3.4. Consenso en espacio cartesiano de un grupo de agentes formado por péndulos.	32
3.5. Consenso en posición articular.	33
3.6. Consenso en velocidad articular.	33
3.7. Consenso en señal de control.	34
4.1. Servo amplificador Dynaserv G3.	39
4.2. Señal de salida de los encoders.	39

4.3. Tarjeta de adquisición de datos y control, Arduino DUE.	40
4.4. Señal de salida del encoder.	41
4.5. Acondicionamiento de señales de salida del encoder.	42
4.6. Señal del encoder después de la etapa de acoplamiento.	42
4.7. Ajuste de offset de la salida del DAC de Arduino Due.	44
4.8. Ajuste de offset de la salida del DAC de Arduino Due.	44
4.9. Respuesta del DAC0 de Arduino Due.	45
4.10. Respuesta del DAC1 de Arduino Due.	45
4.11. Diagrama general de la planta experimental.	46
4.12. Diagrama de flujo para la implementación del algoritmo de control Backstepping.	48
4.13. Gráfica comparativa, trayectoria realizada y trayectoria deseada por el robot manipulador de transmisión directa.	49
4.14. a) Posiciones articulares. b) Error de posición.	50
4.15. Torques empleados para realizar la trayectoria.	51
4.16. Diagrama de implementación Hardware In the Loop.	52
4.17. Adaptador FTDI232R.	53
4.18. Diagrama de conexión del Arduino due y el adaptador FTDI.	53
4.19. Diagrama de implementación Hardware In the Loop en la tarjeta Arduino Due.	54
4.20. Implementación del modelo dinámico en Simulink para el sistema HIL.	55
4.21. Implementación del modelo dinámico en Simulink para el sistema HIL.	56
4.22. Implementación del modelo dinámico en Simulink para el sistema HIL.	56
4.23. Comparación de resultados.	57
4.24. a) Posiciones articulares. b) Error de posición.	58
4.25. Torques empleados para realizar la trayectoria.	58
5.1. Implementación Hardware In the Loop del robot de transmisión directa de tres grados de libertad en Simulink y la tarjeta Arduino Due.	63
5.2. Tarjetas donde se realiza la implementación de los algoritmos de consenso de forma distribuida.	63
5.3. Sistema multiagente de robots manipuladores de tres grados de libertad, Hardware In the Loop.	64
5.4. Diagrama de flujo para la implementación del algoritmo de consenso.	65
5.5. Topología de interacción no dirigida para un sistema con tres agentes interconectados transmitiendo su posición y velocidad articular.	66
5.6. Consenso alcanzado por los tres robots manipuladores en el espacio cartesiano empleando el algoritmo de consenso (5.35).	68

5.7. Consenso alcanzado para los eslabones $q_{i(1)}$ empleando el algoritmo (5.35).	68
5.8. Consenso alcanzado para los eslabones $q_{i(2)}$ empleando el algoritmo (5.35).	69
5.9. Consenso alcanzado para los eslabones $q_{i(3)}$ empleando el algoritmo (5.35).	69
5.10. Velocidad articular $\dot{q}_{i(1)}$ empleando el algoritmo (5.35).	69
5.11. Velocidad articular $\dot{q}_{i(2)}$ empleando el algoritmo (5.35).	70
5.12. Velocidad articular $\dot{q}_{i(3)}$ empleando el algoritmo (5.35).	70
5.13. Torque $\tau_{i(1)}$ empleado con el algoritmo (5.35).	70
5.14. Torque $\tau_{i(2)}$ empleado con el algoritmo (5.35).	71
5.15. Torque $\tau_{i(3)}$ empleado con el algoritmo (5.35).	71
5.16. Consenso alcanzado por los tres robots manipuladores en el espacio cartesiano empleando el algoritmo de consenso (5.39).	73
5.17. Consenso alcanzado para los eslabones $q_{i(1)}$ empleando el algoritmo (5.39).	74
5.18. Consenso alcanzado para los eslabones $q_{i(2)}$ empleando el algoritmo (5.39).	74
5.19. Consenso alcanzado para los eslabones $q_{i(3)}$ empleando el algoritmo (5.39).	74
5.20. Velocidad articular de los eslabones $\dot{q}_{i(1)}$ empleando el algoritmo (5.39).	75
5.21. Velocidad articular de los eslabones $\dot{q}_{i(2)}$ empleando el algoritmo (5.39).	75
5.22. Velocidad articular de los eslabones $\dot{q}_{i(3)}$ empleando el algoritmo (5.39).	75
5.23. Energía suministrada para $\tau_{i(1)}$ empleando el algoritmo (5.39).	76
5.24. Energía suministrada para $\tau_{i(2)}$ empleando el algoritmo (5.39).	76
5.25. Energía suministrada para $\tau_{i(3)}$ empleando el algoritmo (5.39).	76
C.1. Constancia de participación en el primer congreso de tópicos avanzados de óptica y electrónica.	98
C.2. Constancia de participación en el seminario de la maestría en ciencias de la electrónica, opción automatización.	99
C.3. Constancia de participación en el 18° congreso nacional de mecatrónica.	100
C.4. Constancia de participación en el congreso nacional de control automático.	101
C.5. Constancia de participación en el primer foro en investigación y desarrollo tecnológico.	102

Índice de tablas

1.1. Parámetros del robot manipulador de 2 grados de libertad.	13
3.1. Parámetros del péndulo simple.	31
4.1. Servomotores que conforman la estructura del robot manipulador de transmisión directa.	39
4.2. Ganancias para el control backstepping.	49
4.3. Ganancias para el control backstepping en el sistema HIL.	57
5.1. Ganancias elegidas para la implementación del algoritmo de control (5.35).	67
5.2. Ganancias elegidas para la implementación del algoritmo de control (5.39).	72

Capítulo 1

Introducción

1.1. Planteamiento

La robótica se ha convertido en un área de estudio importante para muchos países en vías de crecimiento económico ya que representa desarrollo tecnológico industrial y de servicios, con lo que se introducen un mejor desarrollo de prototipos, procesos y productos lo que representa mejores empleos que llevan a la sociedad bienestar. Actualmente existe una gran variedad de robots para diversas aplicaciones como por ejemplo robots móviles terrestres, aéreos, humanoides, entre otros, de manera particular en este trabajo se hace referencia a robots manipuladores.

En los últimos años se ha tenido la necesidad de hacer que los robots trabajen en conjunto con la finalidad de obtener mejoras en tiempo y ahorro de energía en procesos automatizados, por tal motivo los científicos han puesto su atención en el comportamiento cooperativo, ver Figura 1.1, por esta razón es necesario el desarrollo de estrategias de colaboración que permitan que los componentes individuales del sistema puedan trabajar de forma cooperativa para alcanzar objetivos globales al combinar sus capacidades de decisión y actuación, haciendo frente de forma dinámica a perturbaciones y cambios en su entorno [1].

En la actualidad el problema de consenso se ha estado abordando desde el panorama de control de sistemas dinámicos, es decir, se considera a un sistema dinámico como un agente y la comunicación se determina como una red alámbrica o inalámbrica, la toma de decisiones entre agentes se basa en los estados del sistema y de los estados de los vecinos que integran el equipo en la red de comunicación. El problema real para conseguir el consenso en sistemas multiagente no solo se refleja en los problemas habituales en el control de sistemas dinámicos, el problema del consenso se centra en encontrar una topología de

comunicación adecuada entre agentes que permita establecer un adecuado intercambio de información de los estados entre cada miembro. El consenso de sistemas multiagente se ha estudiado por medio del uso de sistemas de primer y segundo orden, representando la comunicación que existe entre agentes como un grafo, como se presenta en [2, 3] los cuales presentan un algoritmo entre múltiples agentes y configuración de topologías cambiantes con el tiempo en el cual el consenso es alcanzado cuando este contiene un árbol de expansión dirigida. En el trabajo presentado en [4], se explica el hecho de la cooperación de robots móviles, con obstáculos fijos y móviles con ayuda de la teoría de grafos y la teoría de Lyapunov. Por otra parte, en [5], analizaron la convergencia entre los casos en donde se tiene topologías fijas y dirigidas, al igual cuando la topología es dirigida y conmutada y por último presenta el caso cuando la topología de comunicación es fija y con un retraso de tiempo. Otros trabajos muestran el estudio de sistemas Lagrangianas como en [6] entre otros. Todos estos avances, han permitido resolver algunas cuestiones sobre al consenso de sistemas multiagente, sin embargo, existen todavía algunos que no se han podido responder. Por ello resulta de nuestro interés estudiar, estos sistemas con aplicación en la ingeniería, así como analizar los protocolos de comunicación y su instrumentación para lograr un adecuado consenso de sistemas multiagente y así acoplar este a una aplicación real como se presenta en este trabajo de tesis.

En este trabajo de tesis se utilizan dos robots manipuladores completamente actualizados en los cuales se resolvería el problema de consenso en los que la posición deseada sea la misma en cada robot en su espacio de trabajo utilizando la teoría del sistema multiagente.



Figura 1.1: Ejemplo de sistemas multiagente.

1.1.1. Objetivo general

Realizar un estudio comparativo experimental del desempeño de algoritmos de consenso (por lo menos dos) en un sistema multiagente basado en sistemas mecánicos completamente actuados.

1.1.2. Objetivos Particulares

- Modelar matemáticamente el sistema mecánico de un robot manipulador.
- Realizar un estudio comparativo del desempeño de algoritmos de consenso lineales y no lineales en un sistema multiagente de sistemas Lagrangianos.
- Desarrollar la instrumentación necesaria para la implementación de los algoritmos de consenso en robots manipuladores completamente actuados de tres grados de libertad.
- Publicar los resultados de tesis en revista (arbitrada/indexada) y/o congreso arbitrado.

A continuación en la Figura 1.2 se presenta el diagrama de bloques general del sistema multiagente formado por robots manipuladores de transmisión directa.

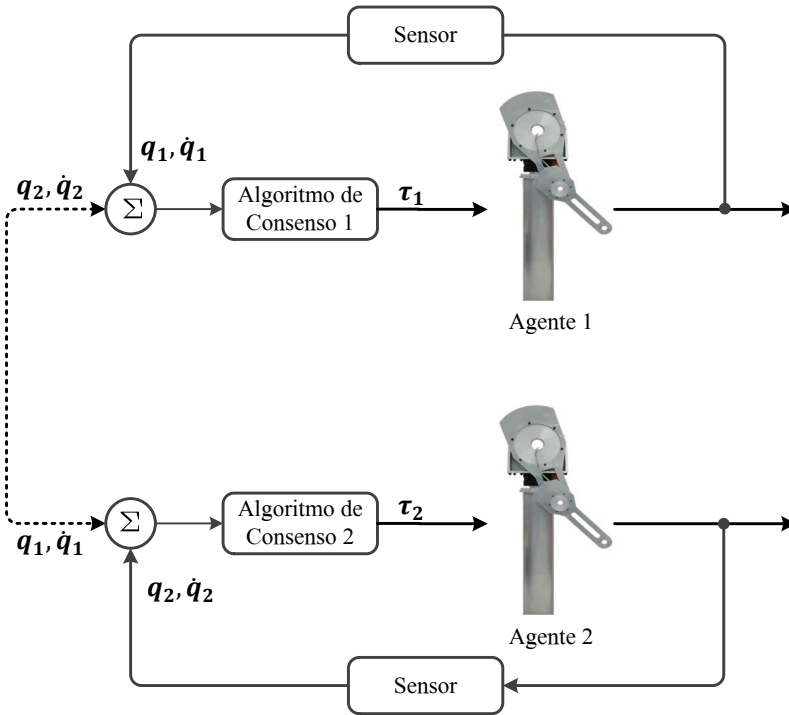


Figura 1.2: Diagrama a bloques general del sistema multiagente.

1.2. Estado del arte

Desde hace varios años se ha estado observando a la naturaleza y el comportamiento de los seres vivos, lo que ha permitido grandes avances científicos. Un comportamiento que ha llamado la atención de la comunidad científica es la capacidad de algunas especies incluidos los humanos en coordinarse para alcanzar objetivos comunes a lo que denominamos consenso. Algunos ejemplos de este comportamiento puede ser fácilmente observado en el vuelo de las aves durante periodos de migración o ante la presencia de algún depredador de igual forma ocurre en la formación de cardúmenes o incluso en la coordinación de las hormigas para transportar de forma más eficiente sus alimentos [7]. Uno de los primeros estudios sobre el comportamiento cooperativo en diferentes especies se presenta en [8]. Debido a los múltiples sistemas robóticos actuales se han realizado diversos estudios en sistemas multiagente aplicando algoritmos de consenso en diferentes plantas experimentales, por ejemplo en vehículos aéreos no tripulados (UAV's) como se presenta en [9–12], de igual forma se realizó en vehículos submarinos autónomos (USV's) como se presenta en [13].

La implementación de algoritmos de consenso en robots manipuladores es tratado con diferentes enfoques para el diseño de controladores y su implementación, por ejemplo, en [14] se presenta el diseño del control mediante el intercambio local de información, es decir, la información se evalúa de robot manipulador a robot manipulador y además se asume que no el sistema no cuenta con la existencia de un líder, por otro lado, en [15] se presenta el diseño de un algoritmo de consenso diseñado mediante el enfoque de pasividad en cada agente, además, en este también se aborda el problema de consenso en sistemas multiagente en presencia de un líder. Otros enfoques para el diseño y estudio de algoritmos de consenso mediante de pasividad en combinación con otras técnicas de diseño de controladores como backstepping en [16], entre otros [1, 6, 17, 18].

En la literatura se puede encontrar dos métodos para el estudio de sistemas multi-agente que permiten alcanzar un objetivo común, uno trata al problema del consenso mediante un enfoque centralizado, es decir, todas las operaciones se realizan en una sola computadora lo que hace al sistema ineficiente ante cualquier perturbación en cualquiera de los integrantes (agentes) del sistema, el segundo enfoque utilizado es el descentralizado o distribuido, este aborda el problema de consenso agregando a cada agente la capacidad de realizar su propia toma de decisiones por medio de un sistema de comunicación y un microcontrolador el cual procesa los estados de información que recibe del agente más próximo lo que hace al método distribuido más eficiente, flexible y escalable, este enfoque se presenta en [19–22].

1.3. Robot manipulador de transmisión directa

Un robot manipulador es esencialmente un brazo articulado, sin embargo, según la definición de Robot del Institute of America, un robot es un manipulador multifuncional reprogramable diseñado para mover materiales, partes, herramientas o dispositivos especializados a través de movimientos programados para la ejecución de una variedad de tareas. Los elementos que componen a un robot manipulador pueden variar según sus características y los requerimientos de aplicación, pero en general comparten algunos como son: Unidad de control, interfaz de control, drivers, motores y sensores. En la Figura 1.3 se presenta un robot manipulador de 3 grados de libertad compuesto por eslabones rígidos unidos a través de articulaciones giratorias de transmisión directa.



Figura 1.3: Robot manipulador de transmisión directa de 3 grados de libertad.

A continuación, se describen algunas de las características del robot manipulador de transmisión directa de 3 grados de libertad:

1.3.1. Articulaciones

Las articulaciones o uniones son servomotores que permiten la conexión y movimiento entre eslabones del robot los cuales forman una cadena cinemática, según el movimiento que se produce por las articulaciones este puede ser lineal (prismáticas) o rotacional. El robot que se toma como caso de estudio en este trabajo posee en las articulaciones

motores de transmisión directa en estos el rotor del motor funciona como una fuente de par o torque debido a que la flecha ha desaparecido y no requiere el sistema de engranes para disminuir la velocidad de giro y amplificar el par. En la Figura 1.4 se muestra la configuración del motor de transmisión directa.

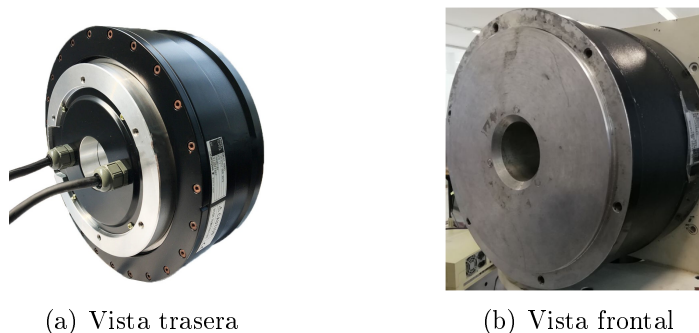


Figura 1.4: Servo motor de transmisión directa.

1.3.2. Eslabones

Un eslabón es un cuerpo rígido que posee por lo menos dos nodos, que son los puntos de unión con otros elementos. Al ensamblaje de eslabones y juntas interconectadas se le denomina cadena cinemática. Ver Figura 1.5¹. Dicho ensamblaje se realiza de modo que proporcione un movimiento de salida controlado en respuesta a un movimiento de entrada proporcionado. Existen dos tipos de cadenas cinemáticas las cuales pueden ser abiertas o cerradas. En la Figura 1.6 se muestra el diseño de los eslabones de los que está compuesto el robot manipulador de transmisión directa que es objeto de estudio en este trabajo.



Figura 1.5: Cadena en cinemática abierta.

¹Imagen recuperada de [23]

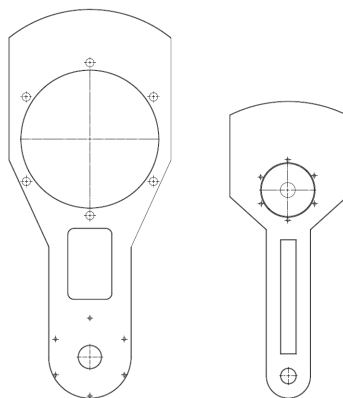


Figura 1.6: Eslabones del robot manipulador de transmisión directa.

1.3.3. Espacio de trabajo

El espacio de trabajo de un robot manipulador es el espacio o lugar donde el robot puede realizar todos sus posibles movimientos. El espacio de trabajo está determinado por la geometría del robot y la naturaleza de sus articulaciones, en la Figura 1.7² se presenta un ejemplo de espacio de trabajo para un robot manipulador.

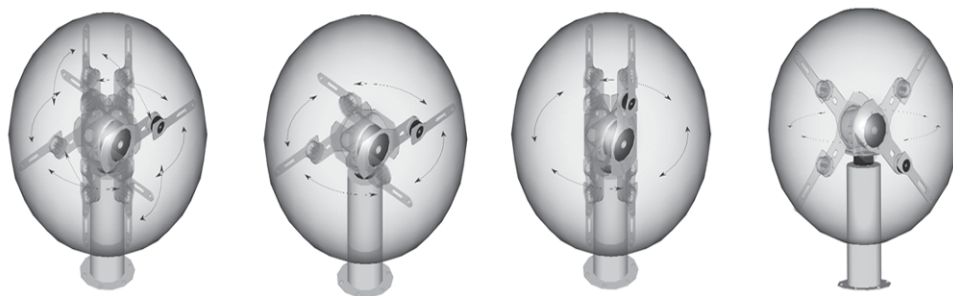


Figura 1.7: Espacio de trabajo de un robot manipulador.

1.3.4. Efector final

El efector final es la parte del último eslabón, la cual se destina a colocar la herramienta adecuada para una tarea específica. La posición del extremo final se representa por $[x, y, z]^T$ y su orientación se denota a través de los ángulos de Euler.

²Imagen recuperada de [23]



Figura 1.8: Diferentes efectores finales para robots manipuladores.

1.4. Modelo dinámico del robot manipulador

El modelo dinámico de un robot tiene como objetivo mostrar la relación que existe entre el movimiento del robot y las fuerzas que ha este se le aplican, el cual puede ser obtenido por diferentes enfoques, de manera particular en este trabajo se realiza a través del método de Euler-Lagrange. Las ecuaciones de movimiento de Euler-Lagrange para la obtención del modelo dinámico de un robot manipulador de n grados de libertad están dadas por

$$\frac{d}{dt} \left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_i} \right] - \frac{\partial \mathcal{L}(q, \dot{q})}{\partial q_i} = \tau_i, \quad i = 1, 2, \quad (1.1)$$

donde $q = [q_1, q_2, \dots, q_n]^\top \in \mathbb{R}^n$ es el vector de posiciones articulares o coordenadas generalizadas, $\dot{q} = [\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n]^\top \in \mathbb{R}^n$ es el vector de velocidades articulares, $\tau = [\tau_1, \tau_2, \dots, \tau_n]^\top \in \mathbb{R}^n$ es el vector de pares aplicados, además el lagrangiano $\mathcal{L}(q, \dot{q})$ de un robot manipulador de n grados de libertad se define como la diferencia entre la energía cinética $\mathcal{K}(q, \dot{q})$ y la energía potencial $\mathcal{U}(q)$, es decir,

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{U}(q). \quad (1.2)$$

La obtención del modelo dinámico del robot de 2 grados de libertad de la Figura 1.9 se realiza considerando el marco de referencia cartesiano cuyo origen se ubica en el eje de rotación de la primera articulación del robot con el plano xy . Por lo que las coordenadas del centro de masa del eslabón 1 en el plano xy son las siguientes:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} l_{c1} \sin(q_1) \\ -l_{c1} \cos(q_1) \end{bmatrix}, \quad (1.3)$$

las coordenadas del centro de masa para el eslabón 2 son las siguientes

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} l_1 \sin(q_1) + l_{c2} \sin(q_1 + q_2) \\ -l_1 \cos(q_1) - l_{c2} \cos(q_1 + q_2) \end{bmatrix}. \quad (1.4)$$

La determinación de la velocidad con respecto al centro de masa de cada eslabón se obtiene de derivar las ecuaciones (1.3) y (1.4), por lo que se tiene lo siguiente:

$$v_1 = \frac{d}{dt} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} l_{c1} \sin(q_1) \\ -l_{c1} \cos(q_1) \end{bmatrix} = \begin{bmatrix} l_{c1} \cos(q_1) \dot{q}_1 \\ l_{c1} \sin(q_1) \dot{q}_1 \end{bmatrix}, \quad (1.5)$$

$$v_2 = \frac{d}{dt} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} l_1 \sin(q_1) + l_{c2} \sin(q_1 + q_2) \\ -l_1 \cos(q_1) - l_{c2} \cos(q_1 + q_2) \end{bmatrix} \quad (1.6)$$

$$= \begin{bmatrix} l_1 \cos(q_1) + l_{c2} \cos(q_1 + q_2) & l_{c2} \cos(q_1 + q_2) \\ l_1 \sin(q_1) + l_{c2} \sin(q_1 + q_2) & l_{c2} \sin(q_1 + q_2) \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}. \quad (1.7)$$

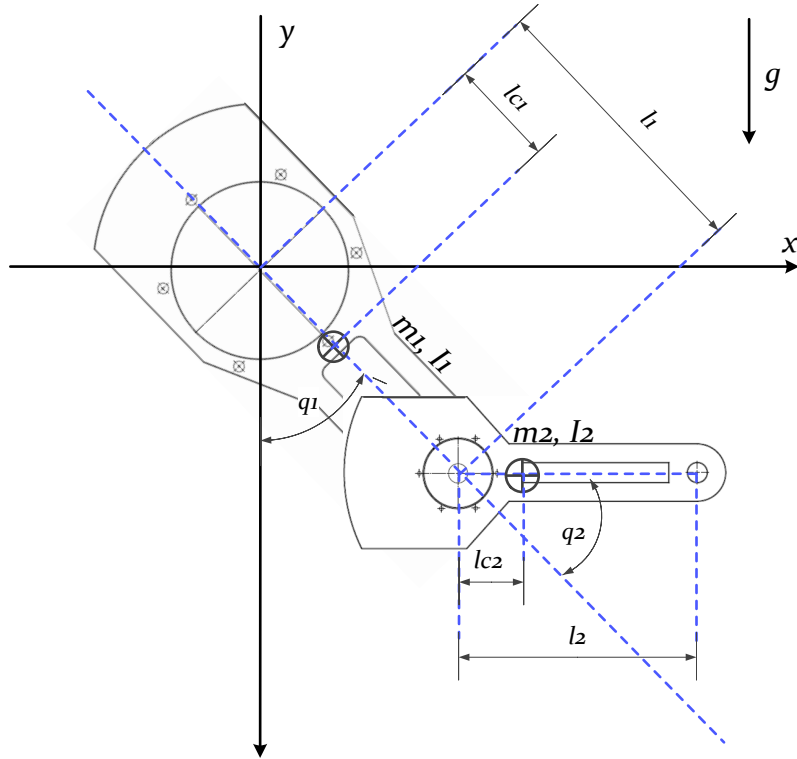


Figura 1.9: Robot manipulador de 2 grados de libertad.

Por lo tanto, la rapidez de cada eslabón se obtiene de la velocidad al cuadrado

$$v_1^\top v_1 = \begin{bmatrix} l_{c1} \cos(q_1) \dot{q}_1 \\ l_{c1} \sin(q_1) \dot{q}_1 \end{bmatrix}^\top \begin{bmatrix} l_{c1} \cos(q_1) \dot{q}_1 \\ l_{c1} \sin(q_1) \dot{q}_1 \end{bmatrix} \quad (1.8)$$

$$= l_{c1}^2 \cos^2(q_1) \dot{q}_1^2 + l_{c1}^2 \sin^2(q_1) \dot{q}_1^2 \quad (1.9)$$

$$= l_{c1}^2 \dot{q}_1^2, \quad (1.10)$$

$$v_2^\top v_2 = \begin{bmatrix} (l_1 \cos(q_1) + l_{c2} \cos(q_1 + q_2)) \dot{q}_1 + l_{c2} \cos(q_1 + q_2) \dot{q}_2 \\ (l_1 \sin(q_1) + l_{c2} \sin(q_1 + q_2)) \dot{q}_1 + l_{c2} \sin(q_1 + q_2) \dot{q}_2 \end{bmatrix}^\top \quad (1.11)$$

$$\begin{aligned} & \begin{bmatrix} (l_1 \cos(q_1) + l_{c2} \cos(q_1 + q_2)) \dot{q}_1 + l_{c2} \cos(q_1 + q_2) \dot{q}_2 \\ (l_1 \sin(q_1) + l_{c2} \sin(q_1 + q_2)) \dot{q}_1 + l_{c2} \sin(q_1 + q_2) \dot{q}_2 \end{bmatrix} \\ &= l_1^2 \cos^2(q_1) \dot{q}_1^2 + 2l_1 \cos(q_1) l_{c2} \cos(q_1 + q_2) \dot{q}_1^2 + 2l_1 \cos(q_1) \dot{q}_1 l_{c2} \cos(q_1 + q_2) \dot{q}_2 + \\ & \quad l_{c2}^2 \cos^2(q_1 + q_2) \dot{q}_1^2 + 2l_{c2} \cos(q_1 + q_2) \dot{q}_1 l_{c2} \cos(q_1 + q_2) \dot{q}_2 + l_{c2}^2 \cos^2(q_1 + q_2) \dot{q}_2^2 + \\ & \quad l_1^2 \sin^2(q_1) \dot{q}_1^2 + 2l_1 \sin(q_1) l_{c2} \sin(q_1 + q_2) \dot{q}_1^2 + 2l_1 \sin(q_1) \dot{q}_1 l_{c2} \sin(q_1 + q_2) \dot{q}_2 + \\ & \quad l_{c2}^2 \sin^2(q_1 + q_2) \dot{q}_1^2 + 2l_{c2} \sin(q_1 + q_2) \dot{q}_1 l_{c2} \sin(q_1 + q_2) \dot{q}_2 + l_{c2}^2 \sin^2(q_1 + q_2) \dot{q}_2^2. \end{aligned} \quad (1.12)$$

Haciendo uso de las identidades trigonométricas, $\cos^2(x) + \sin^2(x) = 1$ y $\cos(x)\cos(x+y) + \sin(x)\sin(x+y) = \cos(y)$ se obtiene lo siguiente:

$$v_2^\top v_2 = l_1^2 \dot{q}_1^2 + 2l_1 l_{c2} \cos(q_2) \dot{q}_1^2 + 2l_1 l_{c2} \cos(q_2) \dot{q}_1 \dot{q}_2 + l_{c2}^2 \dot{q}_1^2 + 2l_{c2}^2 \dot{q}_1 \dot{q}_2 + l_{c2}^2 \dot{q}_2^2 \quad (1.13)$$

$$= l_1^2 \dot{q}_1^2 + 2l_1 l_{c2} \cos(q_2) [\dot{q}_1^2 + \dot{q}_1 \dot{q}_2] + l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2]. \quad (1.14)$$

La energía cinética $\mathcal{K}(q, \dot{q})$ asociada a cada eslabón dada como $\mathcal{K}(q, \dot{q}) = \mathcal{K}_1(q, \dot{q}) + \mathcal{K}_2(q, \dot{q})$ se expresa de la siguiente forma

$$\begin{aligned} \mathcal{K}(q, \dot{q}) &= \frac{1}{2} m_1 l_{c1}^2 \dot{q}_1^2 + \frac{1}{2} I_1 \dot{q}_1^2 + \frac{1}{2} m_2 [l_1^2 \dot{q}_1^2 + 2l_1 l_{c2} \cos(q_2) [\dot{q}_1^2 + \dot{q}_1 \dot{q}_2] + l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2]] \\ & \quad + \frac{1}{2} I_2 (\dot{q}_1 + \dot{q}_2)^2, \end{aligned} \quad (1.15)$$

de forma similar, la energía potencial $\mathcal{U}(q)$ asociada a cada eslabón dada como $\mathcal{U}(q) = \mathcal{U}_1(q) + \mathcal{U}_2(q)$ se expresa de la siguiente forma

$$\mathcal{U}(q) = -m_1 l_{c1} g \cos(q_1) - m_2 l_1 g \cos(q_1) - m_2 l_{c2} g \cos(q_1 + q_2). \quad (1.16)$$

El lagrangiano del robot de la Figura 1.9 se obtiene sustituyendo la ecuación (1.15)

y la ecuación (1.16) en la ecuación (1.2), con lo que se tiene la siguiente expresión

$$\begin{aligned} \mathcal{L}(q, \dot{q}) = & \frac{1}{2}m_1l_{c1}^2\dot{q}_1^2 + \frac{1}{2}I_1\dot{q}_1^2 + \frac{1}{2}m_2[l_1^2\dot{q}_1^2 + 2l_1l_{c2}\cos(q_2)[\dot{q}_1^2 + \dot{q}_1\dot{q}_2] + l_{c2}^2[\dot{q}_1^2 + 2\dot{q}_1\dot{q}_2 + \dot{q}_2^2]] \\ & + \frac{1}{2}I_2(\dot{q}_1 + \dot{q}_2)^2 + m_1l_{c1}g\cos(q_1) + m_2l_1g\cos(q_1) + m_2l_{c2}g\cos(q_1 + q_2). \end{aligned} \quad (1.17)$$

Entonces de la ecuación (1.1) para la coordenada generalizada q_1 presenta las siguientes expresiones

$$\begin{aligned} \frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_1} = & m_1l_{c1}^2\dot{q}_1 + I_1\dot{q}_1 + m_2l_1^2\dot{q}_1 + 2m_2l_1l_{c2}\cos(q_2)\dot{q}_1 + m_2l_1l_{c2}\cos(q_2)\dot{q}_2 \\ & + m_2l_{c2}^2\dot{q}_1 + m_2l_{c2}^2\dot{q}_2 + I_2(\dot{q}_1 + \dot{q}_2), \end{aligned} \quad (1.18)$$

$$\begin{aligned} \frac{d}{dt} \left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_1} \right] = & m_1l_{c1}^2\ddot{q}_1 + I_1\ddot{q}_1 + m_2l_1^2\ddot{q}_1 + 2m_2l_1l_{c2}\cos(q_2)\ddot{q}_1 + m_2l_1l_{c2}\cos(q_2)\ddot{q}_2 \\ & + m_2l_{c2}^2\ddot{q}_1 + m_2l_{c2}^2\ddot{q}_2 + I_2(\ddot{q}_1 + \ddot{q}_2) - 2m_2l_1l_{c2}\sin(q_2)\dot{q}_1\dot{q}_2 \\ & - m_2l_1l_{c2}\sin(q_2)\dot{q}_2^2, \end{aligned} \quad (1.19)$$

$$\frac{\partial \mathcal{L}(q, \dot{q})}{\partial q_1} = - (m_1l_{c1} + m_2l_1)g\sin(q_1) - m_2l_{c2}g\sin(q_1 + q_2). \quad (1.20)$$

Las ecuaciones de movimiento para la q_2 tienen la siguiente forma

$$\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_2} = m_2l_1l_{c2}\cos(q_2)\dot{q}_1 + m_2l_{c2}^2\dot{q}_1 + m_2l_{c2}^2\dot{q}_2 + I_2(\dot{q}_1 + \dot{q}_2), \quad (1.21)$$

$$\begin{aligned} \frac{d}{dt} \left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_2} \right] = & m_2l_1l_{c2}\cos(q_2)\ddot{q}_1 + m_2l_{c2}^2\ddot{q}_1 + m_2l_{c2}^2\ddot{q}_2 + I_2(\ddot{q}_1 + \ddot{q}_2) - m_2l_1l_{c2}\sin(q_2)\dot{q}_1\dot{q}_2, \\ & (1.22) \end{aligned}$$

$$\frac{\partial \mathcal{L}(q, \dot{q})}{\partial q_2} = - m_2l_1l_{c2}\sin(q_2)[\dot{q}_1^2 + \dot{q}_1\dot{q}_2] - m_2l_{c2}g\sin(q_1 + q_2). \quad (1.23)$$

Por lo que las ecuaciones del modelo dinámico para un robot de 2 grados de libertad tienen la siguiente forma

$$\begin{aligned} \tau_1 = & [m_1l_{c1}^2 + m_2l_1^2 + m_2l_{c2}^2 + 2m_2l_1l_{c2}\cos(q_2) + I_1 + I_2] \ddot{q}_1 \\ & + [m_2l_{c2}^2 + m_2l_1l_{c2}\cos(q_2) + I_2] \ddot{q}_2 \\ & - 2m_2l_1l_{c2}\sin(q_2)\dot{q}_1\dot{q}_2 - m_2l_1l_{c2}\sin(q_2)\dot{q}_2^2 \\ & + (m_1l_{c1} + m_2l_1)g\sin(q_1) + m_2l_{c2}g\sin(q_1 + q_2) \end{aligned} \quad (1.24)$$

y

$$\begin{aligned}\tau_2 = & [m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2] \ddot{q}_1 + [m_2 l_{c2}^2 + I_2] \ddot{q}_2 \\ & + m_2 l_1 l_{c2} \sin(q_2) \dot{q}_1^2 + m_2 l_{c2} g \sin(q_1 + q_2),\end{aligned}\quad (1.25)$$

donde τ_1 y τ_2 son los torques que se entregan a los servomotores en las articulaciones hombro y codo.

Reescribiendo el modelo dinámico, es decir, las ecuaciones (1.24) y (1.25) obtenidas mediante el método de Euler-Lagrange del robot manipulador de dos grados de libertad de la Figura 1.9 en forma compacta, es decir,

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q), \quad (1.26)$$

se tiene lo siguiente

$$\tau = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + g \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad (1.27)$$

donde

$$M_{11} = m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2} \cos(q_2) + I_1 + I_2, \quad (1.28)$$

$$M_{12} = m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2, \quad (1.29)$$

$$M_{21} = m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2, \quad (1.30)$$

$$M_{22} = m_2 l_{c2}^2 + I_2, \quad (1.31)$$

$$C_{11} = -m_2 l_1 l_{c2} \sin(q_2) \dot{q}_2, \quad (1.32)$$

$$C_{12} = -m_2 l_1 l_{c2} \sin(q_2) (\dot{q}_1 + \dot{q}_2), \quad (1.33)$$

$$C_{21} = m_2 l_1 l_{c2} \sin(q_2) \dot{q}_1, \quad (1.34)$$

$$C_{22} = 0, \quad (1.35)$$

$$g_1 = l_{c1} m_1 \sin(q_1) + m_2 l_1 \sin(q_1) + m_2 l_{c2} \sin(q_1 + q_2), \quad (1.36)$$

$$g_2 = m_2 l_{c2} \sin(q_1 + q_2). \quad (1.37)$$

Para la implementación y evaluación experimental de los algoritmos de control que se emplean en este trabajo, considere los valores numéricos de la siguiente tabla, para los parámetros de los componentes de la matriz de inercia, la matriz de Coriolis y el vector de pares gravitacionales del modelo dinámico de la ecuación (1.27).

Articulación	Notación	Unidades
Hombro	m_1	$6.94kg$
	l_1	$0.35m$
	l_{c1}	$0.057m$
	I_1	$1.422Nm \frac{s^2}{rad}$
	b_1	$2.155Nm \frac{s}{rad}$
Codo	m_2	$1.18kg$
	l_2	$0.35m$
	l_{c2}	$0.041m$
	I_2	$0.097Nm \frac{s^2}{rad}$
	b_2	$0.215Nm \frac{s}{rad}$
	g	$9.81 \frac{Nm}{s^2}$

Tabla 1.1: Parámetros del robot manipulador de 2 grados de libertad.

1.4.1. Propiedades del modelo dinámico

En esta sección se presentan algunas de las propiedades del modelo dinámico (1.26) que facilitan el diseño de controladores y el análisis de estabilidad del punto de equilibrio [24–28], donde $q \in \mathbb{R}^n$ es el vector de coordenadas generalizadas o posiciones articulares, $\dot{q} \in \mathbb{R}^n$ es el vector de velocidades articulares, $\ddot{q} \in \mathbb{R}^n$ es el vector de aceleraciones articulares, $\tau \in \mathbb{R}^n$ es el vector de pares aplicados a los servomotores, $M(q) \in \mathbb{R}^{n \times n}$ es la matriz de inercia, $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$ es la matriz de fuerzas centrípetas y de Coriolis y $g(q) \in \mathbb{R}^n$ es el vector de pares gravitacionales.

Propiedad 1.4.1. *La matriz de inercia $M(q)$ es simétrica, $M(q) = M(q)^\top$ y definida positiva $M(q) > 0$.*

Propiedad 1.4.2. *Para todo vector $q, x, y \in \mathbb{R}^n$ se satisface,*

$$C(q, x)y = C(q, y)x. \quad (1.38)$$

Propiedad 1.4.3. *La matriz de fuerzas centrípetas y de Coriolis $C(q, \dot{q})$ y la derivada con respecto al tiempo de la matriz de inercia $\dot{M}(q)$ satisfacen*

$$x^\top \left[\frac{1}{2} \dot{M}(q) - C(q, \dot{q}) \right] x = 0, \quad \forall q, \dot{q}, x \in \mathbb{R}^n, \quad (1.39)$$

donde $\frac{1}{2} \dot{M}(q) - C(q, \dot{q})$ es antisimétrica. Equivalentemente, la matriz $\dot{M}(q) - C(q, \dot{q})$ es

antisimétrica, y también cumple

$$\dot{M}(q) = C(q, \dot{q}) + C(q, \dot{q})^\top. \quad (1.40)$$

1.5. observaciones

En este capítulo se revisó el estado del arte de este trabajo de tesis, también se realizó el estudio del modelo dinámico de un robot manipulador de transmisión directa mediante el método de Euler Lagrange, además, se presentan algunas propiedades del modelo dinámico que usualmente se emplean en el análisis de estabilidad por Lyapunov.

Capítulo 2

Preliminares Matemáticos

El intercambio de información entre agentes es uno de los enfoques más importantes es este trabajo por lo que se presenta en este capítulo la teoría de grafos, con esta es posible modelar el intercambio de información a través de grafos de tal forma que sea fácil interpretar la interacción entre agentes y el flujo de información entre agentes, la cual puede ser enviada de un agente a otro o en ambos sentidos.

2.1. Teoría de Grafos

Para realizar la implementación de algoritmos de consenso se requiere que cada miembro del sistema (agente) pueda comunicarse. La teoría de grafos es una topología que en el control de consenso nos permite modelar como es que se comunica cada agente con otro de tal forma que sea más sencillo observar la conexión entre elementos de un sistema multiagente sin importar la dinámica de cada agente. Representamos como nodos a los agentes que conforman al sistema multiagente y las aristas (bordes) son las líneas que representan el flujo de información entre cada agente.

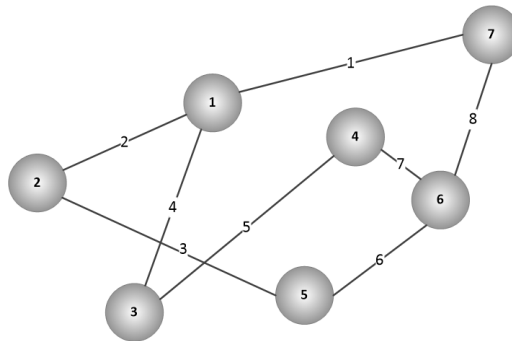


Figura 2.1: Ejemplo de un grafo formado por siete nodos y ocho bordes.

A continuación, se presenta la definición de grafo.

Definición 2.1.1 (Grafo). *Un grafo consiste en un conjunto o par $\mathcal{G} = (V, E)$ con $V = \{\mathcal{V}_1, \dots, \mathcal{V}_N\}$ siendo un conjunto de N nodos o vértices y $E \subseteq V \times V$ un conjunto ordenado de aristas llamado bordes [29] [2].*

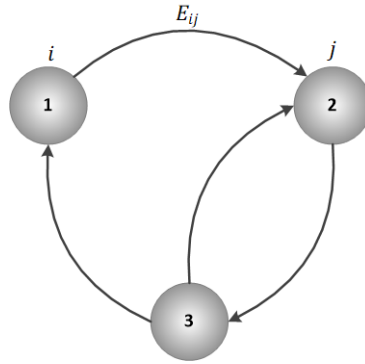


Figura 2.2: Grafo con $V = \{1, 2, 3\}$ y $E = \{\{1, 2\}, \{2, 3\}, \{3, 1\}, \{3, 2\}\}$.

La cardinalidad (número de elementos en un conjunto) del conjunto de vértices V se le denomina *orden* y al cardinal del conjunto de aristas (bordes) E , *tamaño*. Por lo que en el grafo de la Figura 2.2 se tiene que este es un grafo de orden 3 y tamaño 4. La notación $E_{i,j}$ de la Figura 2.2, presenta un borde el cual indica que el agente j puede obtener información del agente i , pero no siempre en sentido contrario ya que depende de las características o tipo de grafo. Los tipos de grafos que en este trabajo se presentan son en los que la comunicación de cada agente puede fluir en un solo sentido y los grafos en los que la comunicación puede fluir en ambos sentidos. Considere las siguientes definiciones.

Definición 2.1.2 (Grafo dirigido). *Un grafo dirigido o dígrafo de orden n es un par $\mathcal{G} = (V, E)$, donde V es un conjunto con n elementos llamados nodos y E es un conjunto de pares ordenados de nodos llamados bordes de la forma $\{\{i_1, i_2\}, \{i_2, i_3\}, \dots\}$.*

En grafos dirigidos los nodos de estos pares representan a los agentes dentro del equipo cuyos bordes representan a las líneas de comunicación con una flecha indicando la dirección.

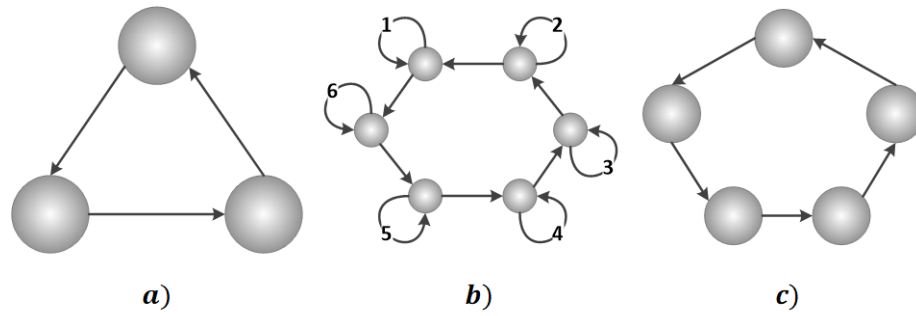


Figura 2.3: Ejemplo de grafos dirigidos.

En la Figura 2.3, se presentan ejemplos de grafos dirigidos con diferentes topologías, el inciso *a)* presenta una topología triangular con tres nodos y tres bordes representados con flechas, en el inciso *b)* se presenta una topología tipo anillo con seis nodos y en cada uno un auto bucle denotados por los números $\{1, 2, 3, 4, 5, 6\}$, el inciso *c)* presenta una topología tipo pentágono con cinco nodos y cinco bordes.

A continuación, se presentan algunas propiedades importantes para grafos dirigidos [29]:

- $\mathcal{G}$ está fuertemente conectado si existe un borde dirigido desde cualquier nodo a cualquier otro nodo.
- $\mathcal{G}$ está débilmente conectado si la versión no dirigida del grafo está conectada.
- $\mathcal{G}$ posee un nodo accesible globalmente si uno de sus nodos puede ser alcanzado desde cualquier otro nodo atravesando un borde dirigido.
- $\mathcal{G}$ posee un árbol de expansión dirigida si uno de sus nodos es la raíz de las rutas dirigidas a cualquier otro nodo.

Definición 2.1.3 (Grafo no dirigido). *Un grafo no dirigido de orden n es un par $\mathcal{G} = (V, E)$, donde V es un conjunto con n elementos llamados nodos y E es un conjunto de pares no ordenados de nodos llamados bordes [30].*

En un grafo no dirigido la comunicación de los agentes puede ser bidireccional. En un dígrafo y grafo no dirigido el auto bucle indica que hay una arista del nodo hacia sí mismo, aunque los auto bucles no están permitidos en los grafos, solo si se indica previamente en el diseño.

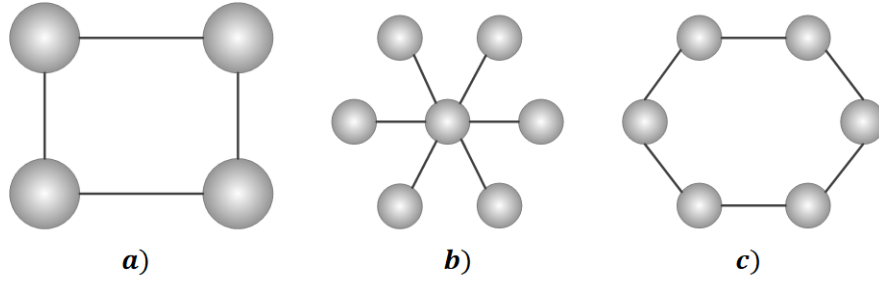


Figura 2.4: Ejemplo de grafos no dirigidos.

En la Figura 2.4, se presentan ejemplos de grafos no dirigidos con diferentes topologías, el inciso a) presenta un topología tipo cuadro, en el inciso b) se presenta una topología tipo estrella y c) presenta una topología tipo anillo.

Definición 2.1.4 (Subgrafo). *Un dígrafo (V', E') es un subgrafo de un dígrafo (V, E) si $V' \subseteq V$ y $E' \subseteq E$. Un dígrafo (V', E') es un subgrafo extendido (spanning subgraph) de (V, E) si este es un subgrafo y $V' = V$. El subgrafo (V, E) inducido por $V' \subseteq V$ es el dígrafo (V', E') , donde E' contiene todos los bordes en E entre dos nodos en V' [29].*

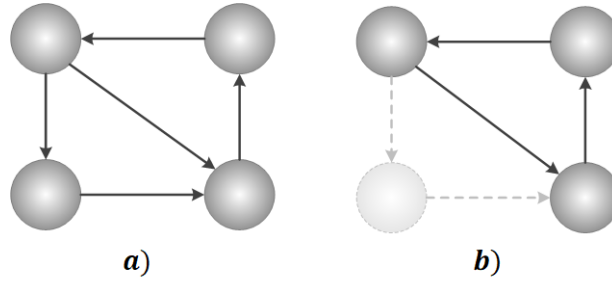


Figura 2.5: Ejemplo de un dígrafo y subgrafo.

En la Figura 2.5, se presenta en el inciso a) un dígrafo con el que se forma el subgrafo del inciso b).

Definición 2.1.5 (Nodos vecinos). *Los nodos i y j de un grafo son vecinos si $\{i, j\}$ es un borde no dirigido. Dado un grafo $\mathcal{G}$, denotamos como $N_{\mathcal{G}}(i)$ al conjunto de vecinos de i .*

Definición 2.1.6 (Grado de un vértice). *El grado de i es el número de vecinos de i . Es decir, la suma de los pesos de las aristas que inciden a un vértice y se denotan de la siguiente forma, $d_{i,j} = \sum_j W_{i,j}$.*

2.2. Álgebra de teoría de grafos

En esta sección se presenta el álgebra que nos permite obtener la información de cada forma y tipo de grafo, lo que nos permite implementar el flujo de información de un sistema multiagente. La forma más común de obtener la información de cómo está dada la conexión entre cada nodo (agente), de cada grafo es a través de matrices. Las matrices que se asocian a un grafo son, la matriz de adyacencia, la matriz de grado y la matriz Laplaciana.

2.2.1. Matriz de adyacencia

Dado un grafo $\mathcal{G} = (V, E)$ con $V = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$ y $E = \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. La matriz de adyacencia de $\mathcal{G}$ denotada por $\mathcal{A}_{\mathcal{G}}$, es una matriz $\mathcal{A}_{\mathcal{G}} = [a_{ij}] \in \mathbb{R}^{n \times n}$. Consideramos la siguiente definición

Definición 2.2.1 (Matriz de adyacencia de un dígrafo). *Las filas y columnas de $\mathcal{A}_{\mathcal{G}}$ están indexadas por V . Si $i \neq j$ entonces la (i, j) -entrada de $\mathcal{A}_{\mathcal{G}}$ es 0 para vértices i y j no adyacentes, y la (i, j) -entrada es 1 para i y j adyacentes. La (i, i) -entrada de $\mathcal{A}_{\mathcal{G}}$ es 0 para $i = 1, \dots, n$ [30].*

Definición 2.2.2 (Matriz de adyacencia de un grafo no dirigido). *Las filas y columnas de $\mathcal{A}_{\mathcal{G}}$ están indexadas por V . Si $i \neq j$ entonces la (i, j) -entrada de $\mathcal{A}_{\mathcal{G}}$ es 0 para vértices i y j no adyacentes, y la (i, j) -entrada es 1 para i y j adyacentes. La (i, i) -entrada de $\mathcal{A}_{\mathcal{G}}$ es 0 para $i = 1, \dots, n$ [30].*

A continuación, se presentan ejemplos que muestran cómo se obtiene la matriz de adyacencia para un grafo dirigido y un grafo no dirigido.

Ejemplo 2.2.1. Considerando el grafo $\mathcal{G} = (V, E)$ de la siguiente figura

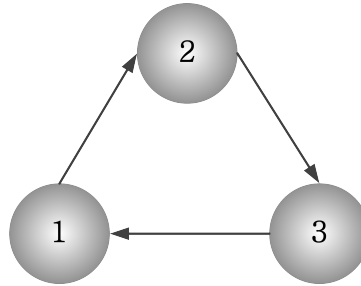


Figura 2.6: Dígrafo con $V = \{1, 2, 3\}$ y $E = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$.

y dada la definición (2.2.1), se obtiene la siguiente matriz de adyacencia

$$\mathcal{A}_{\mathcal{G}} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (2.1)$$

Ejemplo 2.2.2. Considerando el grafo $\mathcal{G} = (V, E)$ de la siguiente figura

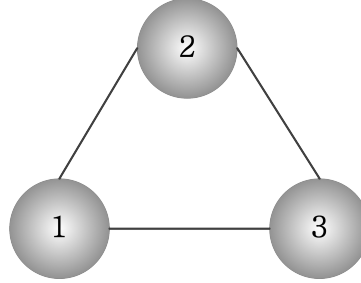


Figura 2.7: Grafo no dirigido con $V = \{1, 2, 3\}$ y $E = \{\{1, 2\}, \{1, 3\}, \{2, 1\}, \{2, 3\}, \{3, 1\}, \{3, 2\}\}$.

y dada la definición (2.2.2), se obtiene la matriz de adyacencia del grafo de la Figura 2.7.

$$\mathcal{A}_{\mathcal{G}} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad (2.2)$$

2.2.2. Matriz de grados

La matriz de grados de un grafo dirigido se forma a partir de los bordes que inciden a un nodo indicados por el extremo final de una flecha, la matriz de grado para un grafo no dirigido se forma apartir de los bordes que inciden en él, aunque estos no estén indicados con flechas. A continuación, se presenta la definición de la matriz de grado.

Definición 2.2.3 (Matriz de grados). *Sea $\mathcal{G} = (V, E)$ un grafo, donde $|V| = n$ la matriz de grados $\mathcal{D}$ de $\mathcal{G}$ es una matriz diagonal donde, $\mathcal{D} \in \mathbb{R}^{n \times n}$ y dada la definición 2.1.6 se define $\mathcal{D}$ como [30]*

$$d_{i,j} = \begin{cases} W_i & \text{si, } i = j \\ 0 & \text{otra cosa.} \end{cases} \quad (2.3)$$

Se muestra un ejemplo para ilustrar de forma practica la matriz de grados de un grafo, primero se realiza para grafos dirigidos y después para grafos no dirigidos

Ejemplo 2.2.3. Considerando el grafo dirigido de la Figura 2.6 y la definición (2.2.3) se tiene lo siguiente

$$\mathcal{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.4)$$

Ahora consideramos el grafo no dirigido de la Figura 2.7 y la definición (2.2.3), con lo que se obtiene lo siguiente

$$\mathcal{D} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \quad (2.5)$$

2.2.3. Matriz Laplaciana

Ahora se presenta la matriz Laplaciana la cual está directamente relacionada al estudio de sistemas dinámicos multiagente y que están representados en grafos. A continuación, se presenta la definición de una matriz Laplaciana [29].

Definición 2.2.4 (Matriz Laplaciana). *Dado un grafo dirigido (dígrafo) $\mathcal{G}$ con su matriz de adyacencia $\mathcal{A}$ y su matriz de grados $\mathcal{D}$, la matriz Laplaciana de $\mathcal{G}$ se obtiene como*

$$\mathcal{L}_{\mathcal{G}} = \mathcal{D} - \mathcal{A}_{\mathcal{G}} \quad (2.6)$$

en componentes $\mathcal{L}_{\mathcal{G}} = (\ell_{ij})_{i,j \in \{1, \dots, n\}}$

$$\ell_{ij} = \begin{cases} -a_{ij}, & \text{si } i \neq j, \\ \sum_{h=1, h \neq i}^n a_{ih}, & \text{si } i = j. \end{cases} \quad (2.7)$$

Para un grafo no dirigido

$$\ell_{ij} = \begin{cases} -1, & \text{si } \{i, j\} \text{ es un borde y no un auto-bucle,} \\ d(i), & \text{si } i = j, \\ 0, & \text{otra cosa.} \end{cases} \quad (2.8)$$

A continuación, se presenta un ejemplo para ilustrar la matriz Laplaciana.

Ejemplo 2.2.4. Considerando el grafo dirigido de la Figura 2.6 la ecuación (2.6) se

escribe de la siguiente forma

$$\mathcal{L}_{\mathcal{G}} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (2.9)$$

$$\mathcal{L}_{\mathcal{G}} = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \quad (2.10)$$

Para un grafo no dirigido como el de la Figura 2.7 la ecuación (2.6) toma la siguiente forma

$$\mathcal{L}_{\mathcal{G}} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad (2.11)$$

$$\mathcal{L}_{\mathcal{G}} = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} \quad (2.12)$$

Por definición, la matriz laplaciana $\mathcal{L}_{\mathcal{G}}$ tiene un valor propio cero con un eigenvector $\mathbf{1}_n$, es decir, $\mathcal{L}_{\mathcal{G}}\mathbf{1} = 0$, donde $\mathbf{1}_n = [1 \cdots 1]^\top \in \mathbb{R}^n$ [6]. Algunas de las propiedades de la matriz Laplaciana se presentan a continuación [1] [2].

Propiedad 2.2.1. *La matriz Laplaciana $\mathcal{L}_{\mathcal{G}}$ de un grafo no dirigido es simétrica y semidefinida positiva.*

Propiedad 2.2.2. *Todos los grafos no dirigidos tienen una matriz Laplaciana $\mathcal{L}_{\mathcal{G}}$ simétrica por lo que sus valores propios son reales.*

Propiedad 2.2.3. *Un grafo no dirigido está conectado si y solo si el segundo valor propio más pequeño de su matriz Laplaciana es estrictamente positivo.*

Propiedad 2.2.4. *La suma de los elementos de cada fila es igual a cero, esto para que todos los grafos tengan el primer valor propio en $\lambda_1 = 0$.*

2.2.4. Producto de Kronecker

Para el diseño e implementación de algoritmos de consenso es necesario representar la sincronización del sistema de forma global haciendo operaciones entre dos matrices, para lograrlo se hace uso del producto de Kronecker el cual se denota con el operador $\otimes$.

El producto de Kronecker de dos matrices $A \in \mathbb{R}^{n \times m}$ y $B \in \mathbb{R}^{q \times r}$ es la matriz de $A \otimes B$ dada por

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{bmatrix} \in \mathbb{R}^{nq \times mr}, \quad (2.13)$$

donde a_{ij} es el (i, j) -elemento de A . El producto de Kronecker satisface las siguientes propiedades [1]

- $(A \otimes B)^\top = A^\top \otimes B^\top$
- $(A \otimes I_P)(C \otimes I_P) = (AC) \otimes I_P$

donde se supone que es posible realizar multiplicación de matrices entre A y C .

2.3. Pasividad

En este trabajo se hace uso de las propiedades de sistemas pasivos para abordar el problema de consenso de sistemas multiagente, para este caso consideramos a sistemas no lineales como agentes que son pasivos de entrada-salida y que su intercambio de información puede ser modelado por un grafo.

Considere un sistema dinámico en espacio de estados con entrada y salida

$$\Sigma \begin{cases} \dot{x} = f(x, u), \\ y = h(x, u), \end{cases} \quad (2.14)$$

donde $x \in \mathbb{R}^n$ es el estado, $u \in \mathbb{R}^m$ es la entrada, y $y \in \mathbb{R}^m$ es la salida. Tome en cuenta que Σ , es un sistema físico con la propiedad de que su energía se puede aumentar sólo a través del suministro de una fuente.

Definición 2.3.1 (Disipatividad [31]). Σ es disipativo con respecto a la tasa de suministro $s(u, y) : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ si y solo si existe una función de almacenamiento $S : \mathbb{R}^n \rightarrow \mathbb{R}$, tal que

$$S(x(T)) \leq S(x(0)) + \int_0^T s(u(t), y(t)) dt, \quad (2.15)$$

para todo $u(t) \in U$, todo $x(t) \in X$ y para todo $t \in [0, T]$. La función $S(x)$ es también llamada función de almacenamiento.

La definición anterior establece que un sistema es disipativo si el aumento en su energía (función de almacenamiento) durante el intervalo $(0, T)$ no es mayor que la energía suministrada (a través de la tasa de suministro) a ella.

Definición 2.3.2 (Pasividad [32]). Σ es pasivo si es disipativo con respecto a la tasa de suministro $s(u, y) = u^\top y$.

- Σ es de entrada estrictamente pasiva si existe $\delta > 0$ tal que Σ es disipativo con tasa de suministro $s(u, y) = u^\top y - \delta \|u\|^2$.
- Σ es de salida estrictamente pasiva si existe $\varepsilon > 0$ tal que Σ es disipativo con respecto a $s(u, y) = u^\top y - \varepsilon \|y\|^2$.
- Σ es sin pérdida si es conservativo con respecto a $s(u, y) = u^\top y$.

La pasividad es la propiedad de que la tasa de aumento de almacenamiento no es más alta que la tasa de suministro. En otras palabras, cualquier aumento de almacenamiento en un sistema pasivo se debe únicamente a fuentes externas. La propiedad de pasividad en sistemas dinámicos puede ser usada como una herramienta de estabilización en el sentido de Lyapunov tomando a la función $S(x)$ como una función candidata de Lyapunov, consideramos el siguiente teorema:

Teorema 2.3.1. (Pasividad y estabilidad) Consideramos que el sistema Σ es pasivo con función de almacenamiento $S(x)$, entonces se cumplen las siguientes propiedades:

- (i) Si $S(x)$ es definida positiva, entonces el equilibrio $x = 0$ de Σ con $u = 0$ es estable.
- (ii) Si Σ es pasivo estricto de salida, entonces es $\mathcal{L}_2$ estable con ganancia finita.
- (iii) Si Σ es pasivo estricto de estado con $S(x)$ definida positiva, entonces el equilibrio $x = 0$ de Σ es asintóticamente estable.

Cuando la función de almacenamiento $S(x)$ es una función radialmente desacotada, estas propiedades son globales, es decir, el equilibrio del sistema es global-asintóticamente estable.

2.4. Observaciones

En este capítulo se ha revisado la definición de grafos, en la que, además, se ha visto que estos pueden ser dirigidos o no dirigidos y que empleando el álgebra de grafos es posible obtener tres matrices fundamentales, la matriz de adyacencia, la matriz de grado

y la matriz Laplaciana. Esto es importante debido que el estudio y análisis de algoritmos de consenso se encuentra directamente relacionado con grafos ya que es con estos que se modela como es que se establece la comunicación entre un agente y sus vecinos.

Capítulo 3

Algoritmos de Consenso

Los algoritmos de consenso son reglas de interacción que especifican el intercambio de información entre un agente y sus vecinos en la red que conforma al sistema multiagente, en estos algoritmos las variables de información que se intercambian son los estados de cada agente. Como consecuencia, cada agente, actualiza su información en función a los estados de sus vecinos. En este trabajo se considera a un robot manipulador como un agente y los estados son su posición y velocidad articular.

3.1. Problema de consenso

Para poder entender el problema de consenso se da su definición según la real academia española.

Definición 3.1.1 (Consenso). *Acuerdo producido por consentimiento entre todos los miembros de un grupo o entre varios grupos.*

Debido a la propiedad de pasividad presente en una gran cantidad de sistemas, como en el caso de robots manipuladores [24, 32], es posible abordar el problema de consenso de salida de sistemas de múltiples agentes, con cada agente modelado por un sistema no lineal de entrada y que es pasivo de entrada-salida [2, 33]. Por lo tanto, el problema de consenso se presenta como en [33] a continuación.

Considere N agentes y una topología de interacción modelada por un grafo que está fuertemente conectado. Cada i -ésimo agente está modelado por un sistema no lineal,

$$\Sigma_i \begin{cases} \dot{x}_i = f_i(x_i, u_i), \\ y_i = h_i(x_i, u_i), \end{cases} \quad (3.1)$$

donde $x_i \in \mathbb{R}^n$ es el estado, $u_i \in \mathbb{R}^m$ es la entrada de control y $y_i \in \mathbb{R}^m$ es la salida. Se

dice que los agentes logran un acuerdo, consenso, si

$$\lim_{t \rightarrow \infty} |y_i(t) - y_j(t)| = 0 \quad \forall i, j = 1, \dots, N. \quad (3.2)$$

Supongo que el i -ésimo agente (3.1) es pasivo, para $i = 1, \dots, N$. Entonces, El siguiente algoritmo de control dado por

$$u_i = \sum_{j \in \mathcal{N}} K(y_j - y_i), \quad i = 1, \dots, N, \quad (3.3)$$

permite que los N agentes converjan a un valor común, donde K es un constante definida positiva y $\mathcal{N}_i$ es el conjunto de agentes transmitiendo sus salidas al i -ésimo agente.

Teorema 3.1.1. [33] *Considere un sistema de N agentes cuya dinámica está dada por la expresión (3.1), para todo $i = 1, \dots, N$. Suponga que el dígrafo que modela la interacción $\mathcal{G}$ está fuertemente conectado. Suponga también que la dinámica del i -ésimo agente es pasiva con función de almacenamiento V_i , para $i = 1, \dots, N$. Entonces, la salida de los N agentes se sincroniza o logra el consenso, es decir, $y_i = y_j, \forall i, j = 1, \dots, N$.*

Demostración. Considere una función candidata de Lyapunov para el grupo de N agentes definido como

$$V = 2(V_1 + V_2 + \dots + V_N). \quad (3.4)$$

La derivada de la función de Lyapunov (3.4) está dada como

$$\dot{V} = 2 \sum_{i=1}^N (-S_i(x_i) + y_i^\top u_i) \quad (3.5)$$

$$= -2 \sum_{i=1}^N S_i(x_i) + 2 \sum_{i=1}^N \sum_{j \in \mathcal{N}} y_i^\top K(y_j - y_i) \quad (3.6)$$

$$= -2 \sum_{i=1}^N S_i(x_i) - 2K \sum_{i=1}^N \sum_{j \in \mathcal{N}} y_i^\top y_i + 2K \sum_{i=1}^N \sum_{j \in \mathcal{N}} y_j^\top y_i \quad (3.7)$$

Como el grafo que modela el intercambio de información está fuertemente conectado y balanceado

$$2K \sum_{i=1}^N \sum_{j \in \mathcal{N}} y_i^\top y_i = K \sum_{i=1}^N \sum_{j \in \mathcal{N}} y_i^\top y_i + K \sum_{i=1}^N \sum_{j \in \mathcal{N}} y_j^\top y_j \quad (3.8)$$

Por lo tanto, se tiene

$$\dot{V} = -2 \sum_{i=1}^N S_i(x_i) - K \sum_{i=1}^N \sum_{j \in \mathcal{N}} (y_i + y_j)^\top (y_i - y_j) \leq 0. \quad (3.9)$$

Por lo tanto, el sistema es globalmente estable y todas las señales están acotadas. Observe que finalmente la sincronización es la salida pasiva de cada agente, por lo tanto (ver teorema 2.3.1), el estado x_i también se sincroniza o alcanza el consenso. $\square$

La convergencia hacia un valor de interés común en un sistema multiagente está estrechamente ligado a la topología de interacción la cual se representa por un grafo y como se ha visto puede ser dirigido o no dirigido. Por lo tanto, se enuncia el siguiente Lemma.

Lemma 3.1.1. *El intercambio de información es condición necesaria para la cooperación.*

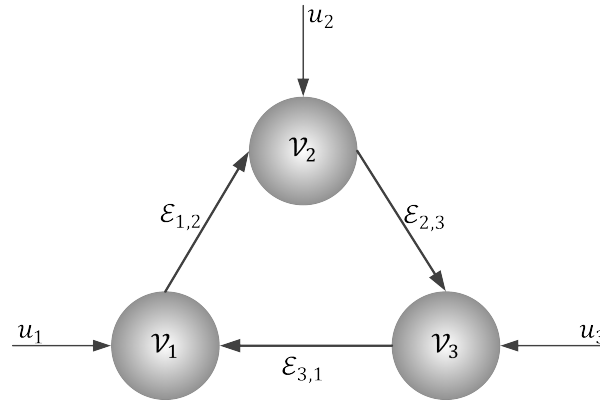


Figura 3.1: Topología de interacción para un sistema con tres agentes interconectados.

Dado el Lema (3.1.1) consideremos el grafo dirigido de la Figura 3.1, el cual representa a un sistema formado por tres agentes transmitiendo sus estados de información según la topología dada. Se observa que cada nodo al que se le denomina agente tiene una entrada de control o regla de interacción individual la cual define el comportamiento del agente según los estados que obtiene y comparte.

3.2. Consenso en sistemas de integrador simple

Considere N agentes modelados con dinámica de integrador simple y que son pasivos de entrada-salida descritos como

$$\dot{\xi}_i = u_i, \quad i, \dots, N, \quad (3.10)$$

donde $\xi_i \in \mathbb{R}^m$ es el estado de información y $u_i \in \mathbb{R}^m$ es la entrada de control que permite alcanzar el consenso del i -ésimo agente. El algoritmo de consenso [20], para

(3.10) está dado por

$$u_i = - \sum_{j=1}^N a_{ij}(t) [\xi_i - \xi_j], \quad i = 1, \dots, N, \quad (3.11)$$

donde $a_{ij}(t)$ es el valor de la entrada (i, j) de la matriz de adyacencia $\mathcal{A}_n \in \mathbb{R}^{n \times n}$ asociado con el grafo $\mathcal{G}_n$ en el tiempo t y ξ representa el estado de información en los agentes. Con (3.11), el consenso es alcanzado por el sistema de múltiples agentes si, para todo $\xi_i(0)$ y todo $i, j = 1, \dots, n$,

$$\|\xi_i(t) - \xi_j(t)\| \rightarrow 0, \quad \text{como } t \rightarrow \infty. \quad (3.12)$$

Por lo que un sistema multiagente con topología de intercambio de información como el de la Figura 3.1 con el algoritmo de consenso (3.11) y condiciones iniciales $\xi_i(0) = [0 \ 0.785398 \ 1.5708]^\top$, tiene el siguiente comportamiento.

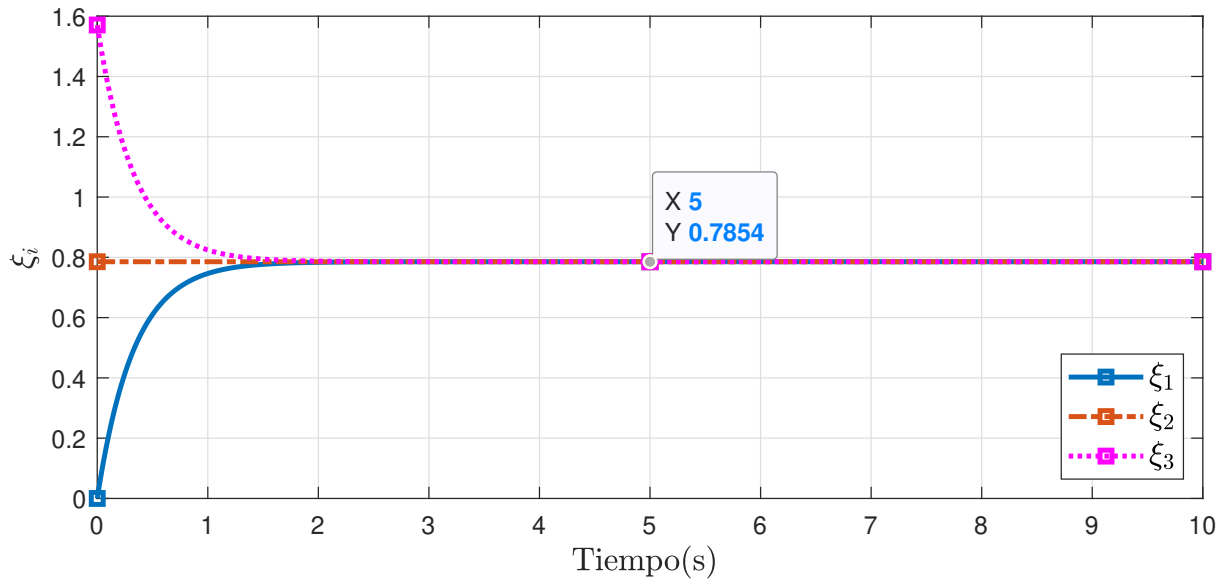


Figura 3.2: Consenso en un sistema de primer orden.

En la Figura 3.2 se observa que el sistema multiagente alcanza el consenso en un valor igual a 0.7854 en un tiempo aproximado a 1.7 segundos.

3.3. Algoritmos de consenso para sistemas con integrador doble

En sistemas donde las ecuaciones de movimiento requieren un modelo dinámico de doble integrador descritos como

$$\dot{\xi}_i = \zeta_i, \quad (3.13)$$

$$\dot{\zeta}_i = u_i, \quad i = 1, \dots, n, \quad (3.14)$$

donde $\xi_i \in \mathbb{R}^m$ es el estado de información, $\zeta_i \in \mathbb{R}^m$ es la derivada del estado de información, y $u_i \in \mathbb{R}^m$ es la entrada de control de consenso asociada con el i -ésimo agente. Con el uso de un grafo dirigido como el de la Figura 3.1, donde, $\mathcal{G}_n^A \triangleq (V_n, E_n^A)$, donde $V_n = \{1, \dots, n\}$ es el nodo establecido y $\mathcal{E}_n^A \subseteq \mathcal{V}_n \times \mathcal{V}_n$ es el conjunto de bordes, y $\mathcal{G}_n^B \triangleq (V_n, E_n^B)$, donde $\mathcal{V}_n$ es el conjunto de nodos y $\mathcal{E}_n^B \subseteq \mathcal{V}_n \times \mathcal{V}_n$ es el conjunto de bordes, para el modelar las topologías de interacción para ξ_i y ζ_i respectivamente, entre los n vehículos. Sea $\mathcal{A}_n = [a_{ij}] \in \mathbb{R}^{n \times n}$ y $\mathcal{B}_n = [b_{ij}] \in \mathbb{R}^{n \times n}$ es la matriz adyacente asociada con $\mathcal{G}_n^A$ y $\mathcal{G}_n^B$ respectivamente. También $\mathcal{L}_n^A = [\ell_{ij}] \in \mathbb{R}^{n \times n}$ y $\mathcal{L}_n^B = [\ell_{ij}] \in \mathbb{R}^{n \times n}$ las cuales son las matrices laplacianas no simétricas asociadas con $\mathcal{G}_n^A$ y $\mathcal{G}_n^B$ respectivamente.

El algoritmo que permite alcanzar el consenso al sistema formado por agentes cuya dinámica esta descrita por (3.13) tiene la siguiente forma como se presenta en [19]

$$u_i = - \sum_{j=1}^n a_{ij}(t) [(\xi_i - \xi_j) - \gamma(t)(\zeta_i - \zeta_j)], \quad i = 1, \dots, n, \quad (3.15)$$

donde $a_{ij}(t)$ es la (i, j) es la entrada de la matriz adyacente $\mathcal{A}_n(t) \in \mathbb{R}^{n \times n}$ asociada con $\mathcal{G}_n$ en el tiempo t y $\gamma(t)$ es un escalar positivo en t . Además, cuando ξ_i y ζ_i denotan, respectivamente, la posición y velocidad del i -ésimo agente, (3.15) representa la aceleración de estos agentes. Con (3.15) el consenso es alcanzado por el equipo de vehículos si, para todo $\xi_i(0)$ y $\zeta_i(0)$ y todo $i, j = 1, \dots, n$,

$$\|\xi_i(t) - \xi_j(t)\| \rightarrow 0, \quad (3.16)$$

y

$$\|\zeta_i(t) - \zeta_j(t)\| \rightarrow 0, \quad \text{como } t \rightarrow \infty. \quad (3.17)$$

Sea $\xi \triangleq [\xi_1^\top, \dots, \xi_n^\top]^\top$ y $\zeta \triangleq [\zeta_1^\top, \dots, \zeta_n^\top]^\top$, aplicando el algoritmo (3.15), (3.13) puede escribirse en forma matricial como

$$\begin{bmatrix} \dot{\xi} \\ \dot{\zeta} \end{bmatrix} = (\Theta(t) \otimes I_m) \begin{bmatrix} \xi \\ \zeta \end{bmatrix}, \quad (3.18)$$

donde

$$\Theta = \begin{bmatrix} 0_{n \times n} & I_n \\ -\mathcal{L}_n(t) & -\gamma(t)\mathcal{L}_n(t) \end{bmatrix} \quad (3.19)$$

con $\mathcal{L}_n \in \mathbb{R}^{n \times n}$ como la matriz Laplaciana no simétrica asociada con el grafo $\mathcal{G}_n$ en el tiempo t .

Para observar el comportamiento del algoritmo de consenso (3.15) considere agentes cuyo modelo dinámico esta descrito por

$$\tau_i = m_i l_{c_i}^2 \ddot{q}_i + m_i g_i l_{c_i} \text{sen}(q_i). \quad (3.20)$$

La dinámica en lazo cerrado de (3.20) se expresa como

$$\frac{d}{dt} \begin{bmatrix} q_i \\ \dot{q}_i \end{bmatrix} = \begin{bmatrix} \dot{q}_i \\ \frac{1}{m_i l_{c_i}^2} (\tau_i - m_i g_i l_{c_i} \text{sen}(q_i)) \end{bmatrix}, \quad (3.21)$$

donde $q \in \mathbb{R}^n$ es el vector de posición, $\dot{q} \in \mathbb{R}^n$ es el vector de velocidad y $\tau \in \mathbb{R}^n$ es la señal de control. Los parámetros que se emplean para (3.20) se muestran en la Tabla 3.1.

Articulación	Notación	Unidades
Péndulo	m	$1.18kg$
	l	$0.35m$
	l_c	$0.041m$
	g	$9.81 \frac{Nm}{s^2}$

Tabla 3.1: Parámetros del péndulo simple.

Considerando la topología de intercambio de información de la Figura 3.3, la dinámica (3.20) de los agentes con los parámetros de la Tabla 3.1.

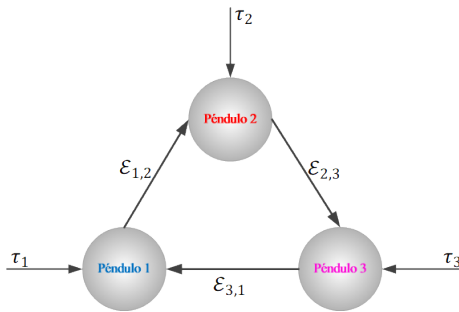


Figura 3.3: Topología de interacción para un sistema con tres agentes interconectados.

El algoritmo de consenso (3.15) más compensación de gravedad para cada agente está dada por

$$\tau_1 = -a_{12}[(q_1 - q_2) + \gamma_1(\dot{q}_1 - \dot{q}_2)] + g_1(q_1), \quad (3.22)$$

$$\tau_2 = -a_{23}[(q_2 - q_3) + \gamma_2(\dot{q}_2 - \dot{q}_3)] + g_2(q_2), \quad (3.23)$$

$$\tau_3 = -a_{31}[(q_3 - q_1) + \gamma_3(\dot{q}_3 - \dot{q}_1)] + g_3(q_3), \quad (3.24)$$

donde $\gamma_{1,2,3} = 2.5$ con condiciones iniciales $q_{1,2,3} = [0 \ 3.14159 \ 0.78539]^\top$ y $\dot{q}_{1,2,3} = [0 \ 0 \ 0]^\top$, Con lo que se tiene el siguiente comportamiento.

En la gráfica de la Figura 3.4, se muestra al sistema multiagente formado por tres péndulos en el espacio cartesiano. Los marcadores cuadrados indican la posición inicial y la posición donde el sistema multiagente logra el consenso. El recuadro que está indicado por una flecha marca el punto de consenso, los recuadros azul y rojo no se visualizan ya que estos se sobreponen.

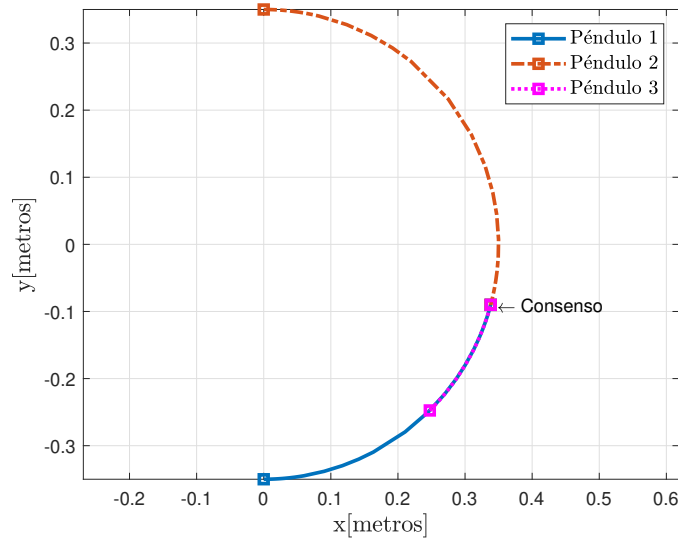


Figura 3.4: Consenso en espacio cartesiano de un grupo de agentes formado por péndulos.

En la Figura 3.5, se observan las posiciones articulares de cada agente y se muestra el valor cuando las posiciones logran el consenso siendo este igual a 1.309 radianes en un tiempo aproximado de 3 segundos. Por otro lado en la Figura 3.6, se muestran las velocidades articulares correspondientes a cada agente, debido a que los agentes logran el consenso la velocidad tiende a cero ya que los agentes permanecen sin movimiento. En la Figura 3.7, se muestran los torques que se emplean para que el sistema logre el consenso manteniendo un torque igual a 0.4585 Nm para cada agente.

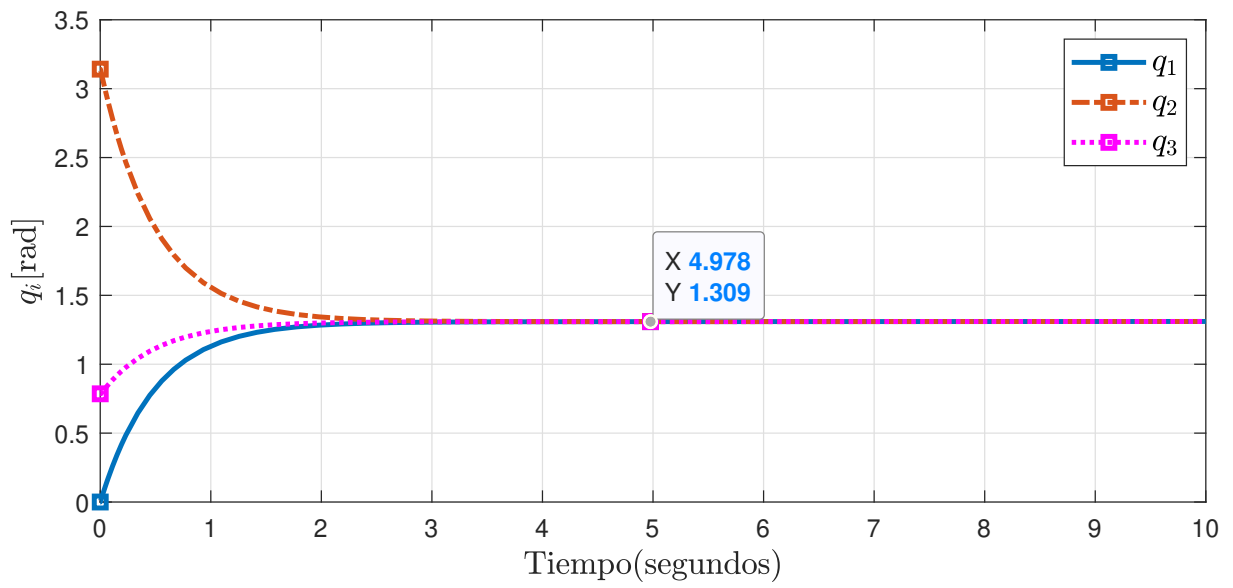


Figura 3.5: Consenso en posición articular.

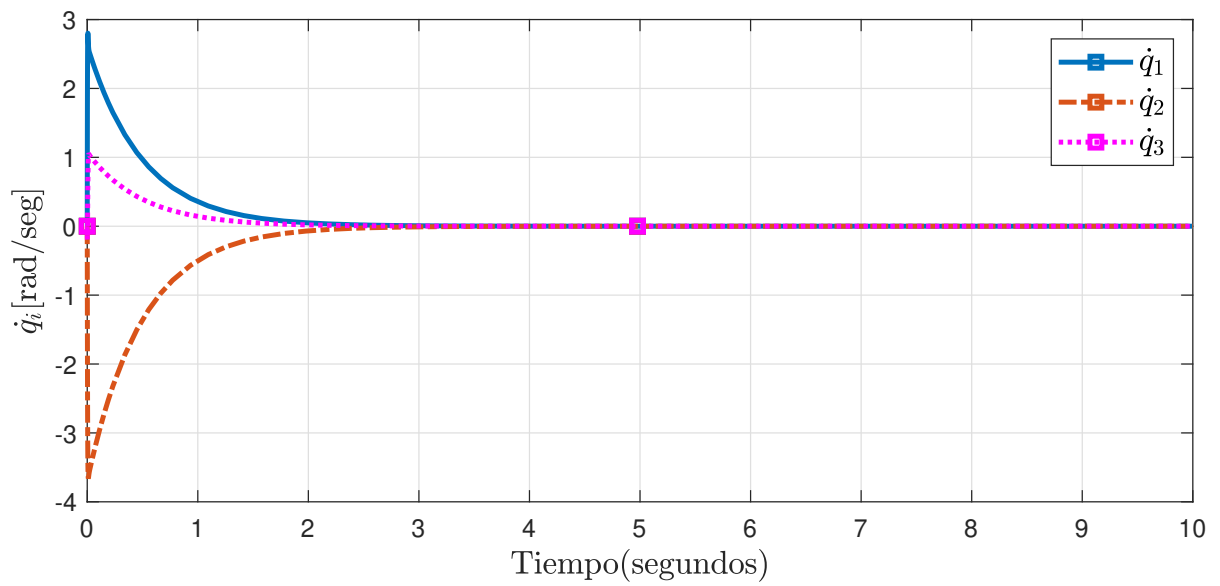


Figura 3.6: Consenso en velocidad articular.

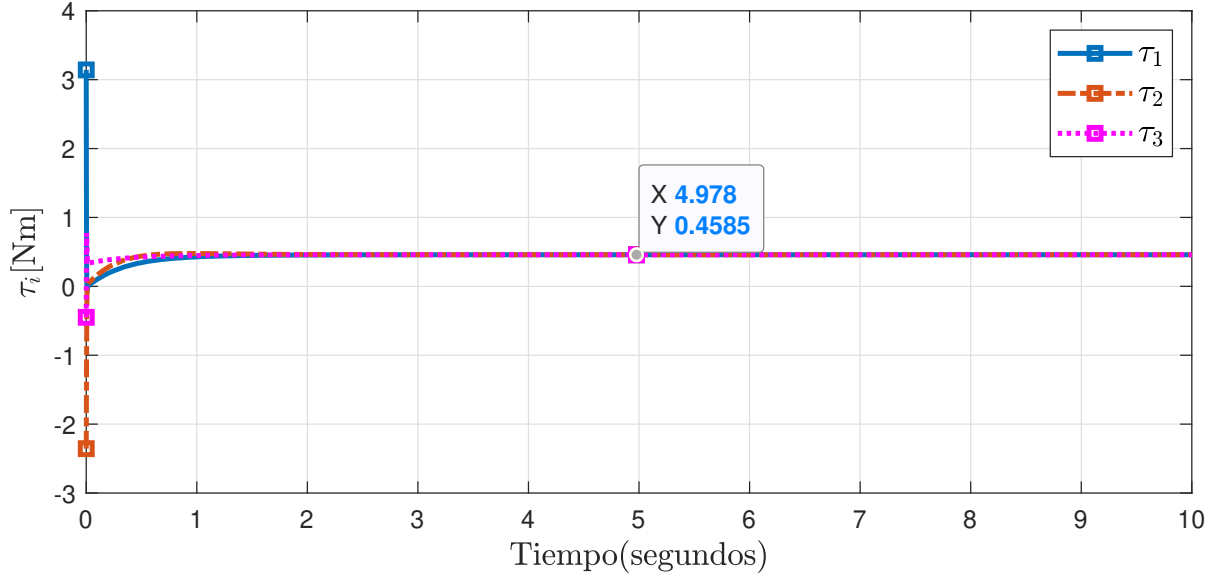


Figura 3.7: Consenso en señal de control.

3.4. Algoritmos de consenso para robots manipuladores

Los algoritmos de consenso pueden ser usados para sincronizar brazos robóticos de n grados de libertad empleando su modelo dinámico y de forma general es posible sincronizar sistemas Euler-Lagrange afines a (3.1) que son representados por

$$\tau_i = M_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i), \quad i = 1, \dots, N, \quad (3.25)$$

donde $q_i \in \mathbb{R}^p$ es el vector de coordenadas generalizadas o posiciones articulares, $\dot{q}_i \in \mathbb{R}^p$ es el vector de velocidades articulares, $\ddot{q}_i \in \mathbb{R}^p$ es el vector de aceleraciones articulares, $M_i(q_i) \in \mathbb{R}^{p \times p}$ es la matriz de inercia la cual es simétrica y definida positiva, $C_i(q_i, \dot{q}_i) \in \mathbb{R}^{p \times p}$ es la matriz de fuerzas centrípetas y de Coriolis, $g_i(q_i) \in \mathbb{R}^p$ es el vector de pares gravitacionales y $\tau_i \in \mathbb{R}^p$ es el vector de pares aplicados a los servomotores asociados con al i -ésimo agente. Aunque las ecuaciones de movimiento anteriores son acopladas y no lineales, poseen ciertas propiedades fundamentales debido a su estructura dinámica y que pueden ser consultadas más a detalle en [25, 26].

El modelo dinámico (3.25) con la siguiente función candidata de Lyapunov

$$V_i(q_i, \dot{q}_i) = \frac{1}{2} \dot{q}_i^\top M_i(q_i) \dot{q}_i + g_i(q_i), \quad (3.26)$$

Como en [24, 32] haciendo uso de la Propiedad 1.39 se puede verificar

$$V_i(t) - V_i(0) = \int_0^t \tau_i(s)^\top \dot{q}_i(s) ds \quad (3.27)$$

por lo tanto, la dinámica es pasiva con $(\tau_i, \dot{q}_i)$ como el par de entrada-salida y (3.26) como función de almacenamiento. Se puede observar que la función de almacenamiento (3.26) es definido positivo para todo $i = 1, \dots, N$.

Considere una topología de interconexión entre los N agentes que satisfaga que el grafo está fuertemente conectado, y el sistema logre el consenso utilizando el control

$$\tau_i = \sum_{j \in \mathcal{N}_i} K(\dot{q}_j, \dot{q}_i), \quad i = 1, \dots, N \quad (3.28)$$

Empleando el Teorema 3.1.1 con

$$V = 2(V_1 + \dots + V_N) \quad (3.29)$$

como función candidata de Lyapunov el sistema se sincroniza a la salida y

$$\lim_{t \rightarrow \infty} |\dot{q}_i(t) - \dot{q}_j(t)| = 0 \quad \forall i, j = 1, \dots, N \quad (3.30)$$

Debido a que todas las salidas son pasivas, convergen por pasividad, todos los agente se van a una posición constante.

3.4.1. Algoritmo de consenso fundamental

El objetivo como se muestra en [34], es diseñar algoritmos para (3.25) que ante la ausencia de una señal de referencia, la posición del efector final de todos los agentes debe alcanzar un consenso a un valor constante $q_c \in \mathbb{R}^P$, además, sus velocidades deben converger a cero, es decir,

$$\lim_{t \rightarrow \infty} q_i(t) = q_c, \quad (3.31)$$

$$\lim_{t \rightarrow \infty} |\dot{q}_i(t)| = 0. \quad (3.32)$$

Considere el algoritmo de consenso fundamental como se presenta en [20] para (3.25) el cual está dado por

$$\tau_i = - \sum_{j=1}^n a_{ij}(q_i - q_j) - \sum_{j=1}^n b_{ij}(\dot{q}_i - \dot{q}_j) - K_i \dot{q}_i + g_i(q_i), \quad (3.33)$$

donde $i = 1, \dots, n$, a_{ij} es la (i, j) entrada de la matriz de adyacencia $\mathcal{A} \in \mathbb{R}^{n \times n}$ asociada con el grafo $\mathcal{G}_A = (V_A, E_A)$ la cual caracteriza la interacción entre los n agentes para q_i , b_{ij} es la (i, j) entrada de la matriz de adyacencia $\mathcal{B} \in \mathbb{R}^{n \times n}$ asociada al grafo $\mathcal{G}_B = (V_B, E_B)$ la cual caracteriza la interacción entre los n agentes para $\dot{q}_i$, $K_i \in \mathbb{R}^{p \times p}$ es simétrica definida positiva.

3.4.2. Algoritmo de consenso no lineal

Los algoritmos de consenso no lineales para sistemas Euler-lagrange están dados por

$$\tau_i = - \sum_{j=1}^n a_{ij} \psi [K_q(q_i - q_j)] - \sum_{j=1}^n b_{ij} \psi [K_{\dot{q}}(\dot{q}_i - \dot{q}_j)] - K_i \psi (K_{d_i} \dot{q}_i) + g_i(q_i), \quad (3.34)$$

donde $i = 1, \dots, n$, a_{ij} es la (i, j) entrada de la matriz de adyacencia $\mathcal{A} \in \mathbb{R}^{n \times n}$ asociada con el grafo $\mathcal{G}_A = (V_A, E_A)$ la cual caracteriza la interacción entre los n agentes para q_i , b_{ij} es la (i, j) entrada de la matriz de adyacencia $\mathcal{B} \in \mathbb{R}^{n \times n}$ asociada al grafo $\mathcal{G}_B = (V_B, E_B)$ la cual caracteriza la interacción entre los n agentes para $\dot{q}_i$, $K_i \in \mathbb{R}^{p \times p}$ es simétrica definida positiva. $\psi(\cdot)$ se define como en [20] en el siguiente lemma.

Lemma 3.4.1. *Considere que $\psi : \mathbb{R} \rightarrow \mathbb{R}$ es una función impar continua que satisfaga $\psi(x) > 0$ si $x > 0$. Además, $\psi(\cdot)$ es continuamente diferenciable.*

3.5. Observaciones

En este capítulo se realizó el estudio de los algoritmos de consenso, en el cual primero se define el problema de consenso y después se realizó ejemplos en sistemas que nos permitan poder entender de forma más simple la implementación de algoritmos de consenso. Después se mostró los algoritmos de consenso que se emplean para permitir que un grupo de robots manipuladores pueda converger hacia una posición común. En el siguiente capítulo se presenta el sistema que es caso de estudio en este trabajo de tesis y es en el que se implementan algoritmos de consenso.

Capítulo 4

Instrumentación

En este capítulo se presenta la electrónica que se emplea para la adquisición de datos, el control del robot manipulador de transmisión directa y el sistema multiagente. Por lo que se da una breve descripción del servo driver que controla al servomotor y se describe la tarjeta electrónica que por su arquitectura permite que se le realicen los ajustes necesarios para controlar el robot manipulador de transmisión directa de forma correcta. También se describe al sistema Hardware In the Loop, donde se implementa una red de robots manipuladores y se implementa los algoritmos de consenso para controlar al sistema multiagente el cual está formado por tres robots virtuales. Parte importante de este trabajo es la comunicación entre agentes, este problema se aborda haciendo uso del sistema UART (Universal Asynchronous Receiver-Transmitter).

4.1. Casos de estudio

En esta sección se presentan los casos de estudio que en este trabajo se abordan. Primero se consideran los aspectos técnicos que nos permitan controlar al robot manipulador de transmisión directa y después se consideran las especificaciones técnicas de la tarjeta electrónica que permitan la correcta implementación y evaluación de algoritmos de control en un robot manipulador de transmisión directa.

4.1.1. Robot manipulador de transmisión directa

A los robots antropomórficos que por diseño poseen eslabones rígidos unidos por articulaciones giratorias compuestas por servomotores de transmisión directa se les denomina “robot de transmisión directa”, como en [23]. Ver Figura 1.3. Transmisión directa (Direct drive) significa que el servomotor funciona como una fuente ideal de par o torque, no tiene pérdidas de energía como sucede en motores provistos de engranaje. Direct drive

elimina el sistema de engranaje tradicional, por lo que el rotor del motor está directamente acoplado al eslabón, además, con esta tecnología se elimina el juego mecánico y se reduce significativamente el fenómeno de fricción.

Un servomotor está compuesto por motor eléctrico, sensor de posición y el amplificador electrónico (electronic driver). Los servomotores de transmisión directa tienen tres modos de operación [35,36]:

- **Modo Posición:** El modo posición permite mover al motor a una posición preestablecida con la desventaja de que en este modo de operación no es posible desplazar objetos o aplicar fuerza.
- **Modo velocidad:** El modo velocidad permite controlar el movimiento del robot a una velocidad deseada, tanto en modo posición y modo velocidad no es posible implementar otro tipo de controladores conservando las características de sistemas con arquitectura cerrada.
- **Modo par o torque:** El modo de operación por par hace que el servomotor funcione como una fuente de par permitiendo que el par se mantenga en magnitud y signo. En este modo el servomotor posee características de sistemas con arquitectura abierta, haciendo posible implementar y evaluar algoritmos de control.

Los amplificadores electrónicos de los que está provisto el robot manipulador de transmisión directa que se toma como caso de estudio en este trabajo son de 3ra-generación, Figura 4.1, y poseen integrados a su estructura un encoder rotacional que para el hombro puede alcanzar hasta 4,096,000 pulsos por revolución y para la base y el codo 2,621,440. Sin embargo, en los servo amplificadores de 3ra-generación es posible ajustar la cantidad de pulsos por revolución del encoder por lo que estos cuentan con 10,260 pulsos por revolución es decir 0.0350877192° de resolución como se presenta en [37]. Los encoder proporcionan dos señales A y B de 5V las cuales están desfasadas 90° como se muestra en la Figura 4.2.

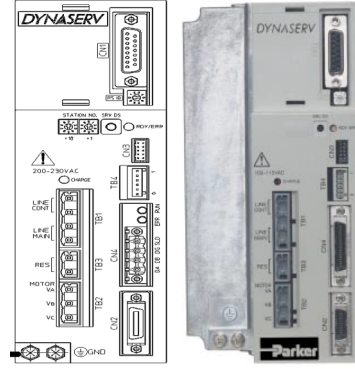


Figura 4.1: Servo amplificador Dynaserv G3.

En el modo par, la señal de control que se debe enviar corresponde a un voltaje el cual puede estar en el rango de $\pm 8V$, de esta forma si se envía 0V el servo motor dará como respuesta 0 par o torque, el par máximo se alcanzará cuando el servomotor reciba $\pm 8V$ [36]. A continuación, se presentan los modelos de los servomotores que conforman la estructura del robot manipulador.

Articulación	Modelo	Resolución	Torque
Base	DMG3-1015B-115B	10260 $\frac{\text{pulsos}}{\text{rev}}$	15 Nm
Eslabón 1	DMG3-1050A-115B	10260 $\frac{\text{pulsos}}{\text{rev}}$	50 Nm
Eslabón 2	DMG3-1004C-115B	10260 $\frac{\text{pulsos}}{\text{rev}}$	4 Nm

Tabla 4.1: Servomotores que conforman la estructura del robot manipulador de transmisión directa.

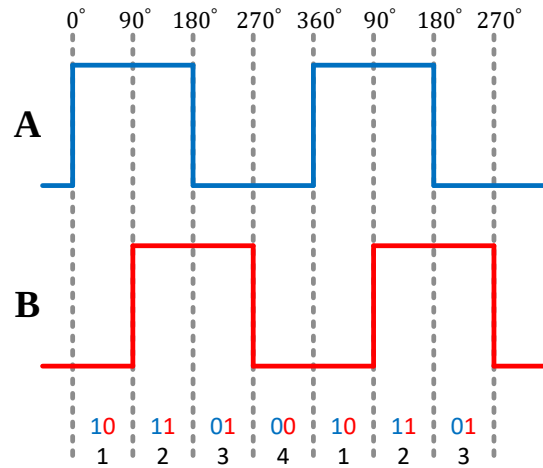


Figura 4.2: Señal de salida de los encoders.

4.1.2. Tarjeta de adquisición de datos y control

Además de la arquitectura mecánica del robot manipulador es necesario contar con arquitectura electrónica que permita realizar la implementación de algoritmos de control. Arduino es un sistema electrónico de arquitectura abierta con una amplia gama de aplicaciones en robótica y mecatrónica como se presenta en [38]. El Arduino Due, Figura 4.3, está basado en un microcontrolador ARM Cortex-M3. Tiene 54 pines de entrada-salida digital (de los cuales 12 se pueden usar como salidas PWM), 12 entradas analógicas, 4 UART (puertos seriales), un reloj de 84 MHz, conexión compatible con USB, 2 DAC (convertidor digital a analógico), 2 TWI, un conector de alimentación, un conector SPI, un botón de reinicio y un botón de borrado, además de otras características que pueden ser consultadas más a detalle en [39]. Sin embargo, para su correcto funcionamiento requiere de acoplar las señales de entrada a la tarjeta y al mismo tiempo ajustar las salidas que por lo general son las señales que se requiere para el control de un sistema físico.

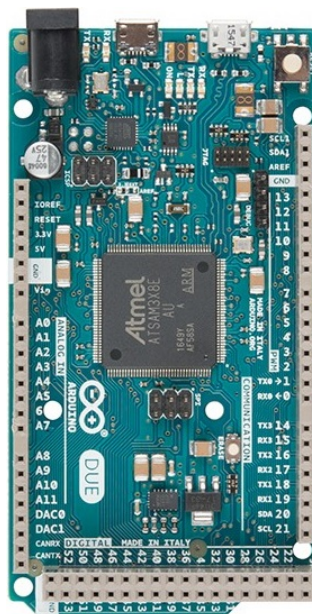


Figura 4.3: Tarjeta de adquisición de datos y control, Arduino DUE.

La tarjeta Arduino Due funciona a 3.3V por lo que para la implementación de algoritmos de control se debe considerar que voltajes superiores podrían dañar la tarjeta, también se debe considerar que las salidas de los DAC's incorporados no son ideales ya que estos presentan un offset de aproximadamente 0.500mV por lo que es necesario realizar una etapa de ajuste a 0V, además estos DAC's solo proporcionan voltaje positivo haciendo necesaria la etapa electrónica para la obtención de voltajes negativos y voltajes positivos.

4.2. Acoplamiento de señales de entrada y ajuste de señales de salida

En esta sección se presenta los ajustes realizados a la tarjeta Arduino Due para poder realizar la implementación de algoritmos de control en un robot manipulador de transmisión directa , primero se presenta la electrónica que permite acoplar las salidas de los encoders a la tarjeta Arduino Due y después se presenta la electrónica que permite ajustar y amplificar la salida de los convertidores digital-analógico.

4.2.1. Acoplamiento de los encoders

Para realizar una correcta lectura de la posición actual del robot manipulador a través de la tarjeta Arduino Due se requiere que la señal se conserve lo más posible sin ruido, es decir, libre de señales no deseadas que alteren la señal de salida del encoder. En la Figura 4.4, se presentan las señales de salida del encoder antes de la etapa de acoplamiento.

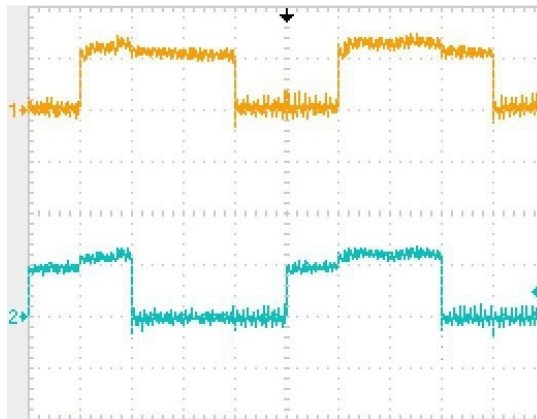


Figura 4.4: Señal de salida del encoder.

Para ajustar la señal de salida de los encoder se realiza la implementación de un circuito basado en un transistor tipo npn el cual es configurado en modo saturación, en el colector del transistor se coloca el voltaje de referencia en este caso corresponde a los 3.3 V que se requieren a la entrada del Arduino Due, en la base del transistor se inserta la señal del encoder lo que nos permitirá tener en el emisor la señal del encoder acondicionada para la correcta lectura del encoder, el circuito se muestra en la Figura 4.5.

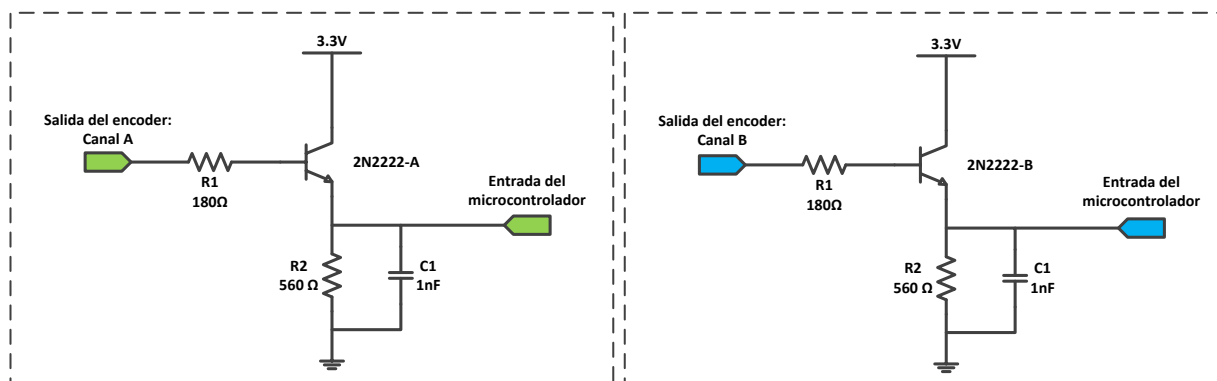


Figura 4.5: Acondicionamiento de señales de salida del encoder.

Después de la etapa de acoplamiento con el circuito propuesto se obtienen las siguientes señales a la salida del encoder como se muestra en la Figura 4.6.

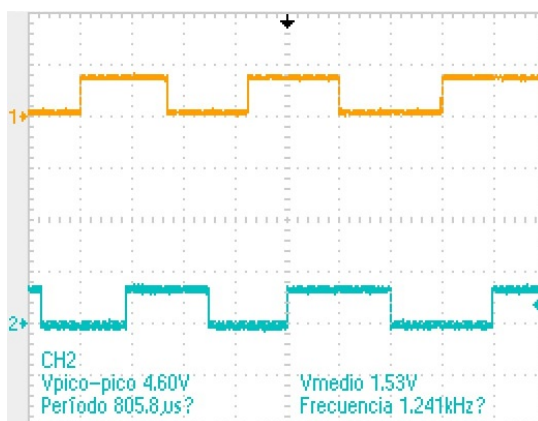


Figura 4.6: Señal del encoder después de la etapa de acoplamiento.

4.3. Adaptación de los convertidores digital-analógico de la tarjeta Arduino Due

Los convertidores digital-analógico con los que cuenta la tarjeta Arduino Due pueden ser configurados para operar con resoluciones de 10 o 12 bits, por conveniencia en este trabajo se configuran para operar a 12 bits con lo que el voltaje de salida puede tomar 4096 valores distintos. Considerando las necesidades de adaptación para los DAC del Arduino Due se realiza un circuito electrónico que adapte los DAC's del Arduino a los requerimientos necesarios y se presenta en dos partes como en [40].

En la primera etapa se realiza el ajuste del offset del DAC interno del Arduino DUE, a través del diseño de un divisor de voltaje haciendo uso de la ecuación (4.1) con el cual

se pueda obtener aproximadamente 500mV los cuales se emplean para ajustar el offset de los DAC's del Arduino DUE.

$$V_S = \frac{R_2 \times V_E}{R_1 + R_2} \quad (4.1)$$

$$= \frac{580\Omega \times 5V}{4.8K\Omega + 580\Omega} \cong 0.539V. \quad (4.2)$$

La salida del divisor de voltaje se conecta a un amplificador operacional configurado como seguidor esto con la finalidad de acoplar las impedancias entre el divisor de voltaje y el resto del circuito, a continuación la salida del amplificador seguidor se conecta a la terminal negativa del siguiente amplificador operacional que está configurado como restador inversor el cual se encarga de realizar la resta del offset que tiene el DAC de arduino DUE como se muestra en las ecuaciones (4.3) y (4.4).

$$V_{S1} = V_{ADAC} \left(\frac{(R_3 + R_1)R_4}{(R_4 + R_2)R_1} \right) - V_S \left(\frac{R_3}{R_1} \right) \quad (4.3)$$

$$= V_{ADAC} \left(\frac{(1K\Omega + 1K\Omega)1K\Omega}{(1K\Omega + 1K\Omega)1K\Omega} \right) - V_S \left(\frac{1K\Omega}{1K\Omega} \right) \quad (4.4)$$

En la Figura 4.7, se muestra el circuito de la primera etapa.

En la segunda etapa, el voltaje obtenido a la salida del amplificador operacional U4B de la Figura 4.7, es de aproximadamente 2.35V, por lo que es necesario implementar un siguiente amplificador operacional que nos permita ajustar a 5V, esto se realiza con el amplificador operacional U4C el cual tiene un arreglo de resistencias R8 y R7 para obtener una ganancia de 2.15 con la que podremos ajustar al voltaje que se requiere esto a través de la siguiente ecuación:

$$V_{S2} = V_{S1} \left(1 + \frac{R_7}{R_8} \right), \quad (4.5)$$

$$= 2.33V \left(1 + \frac{2.2K\Omega}{2K\Omega} \right) = 5.06V. \quad (4.6)$$

El amplificador operacional U4D nos permite ajustar el voltaje del DAC de tal forma que nos permita obtener un voltaje que este entre -5V y 5V, como se muestra en la Figura 4.8, con lo que es posible que los servomotores del robot manipulador puedan trabajar en ambos sentidos de giro.

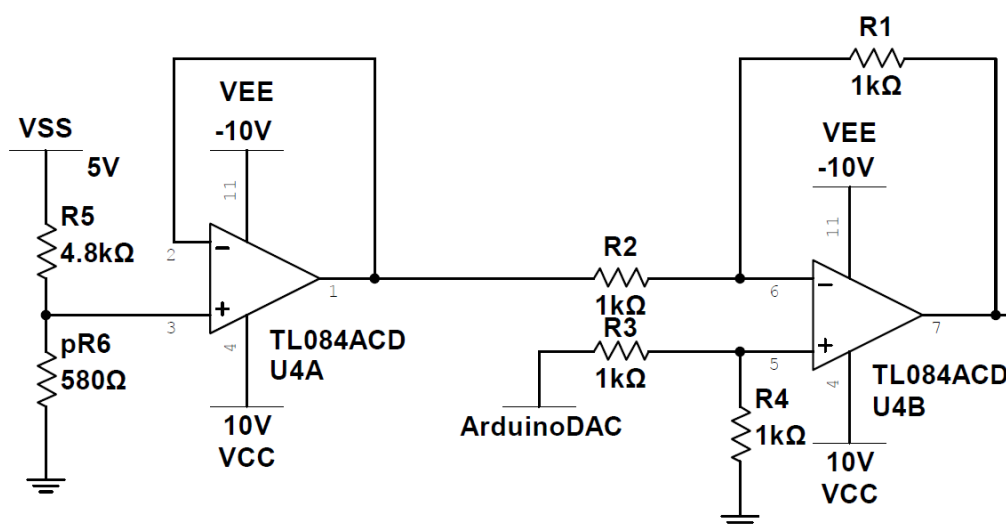


Figura 4.7: Ajuste de offset de la salida del DAC de Arduino Due.

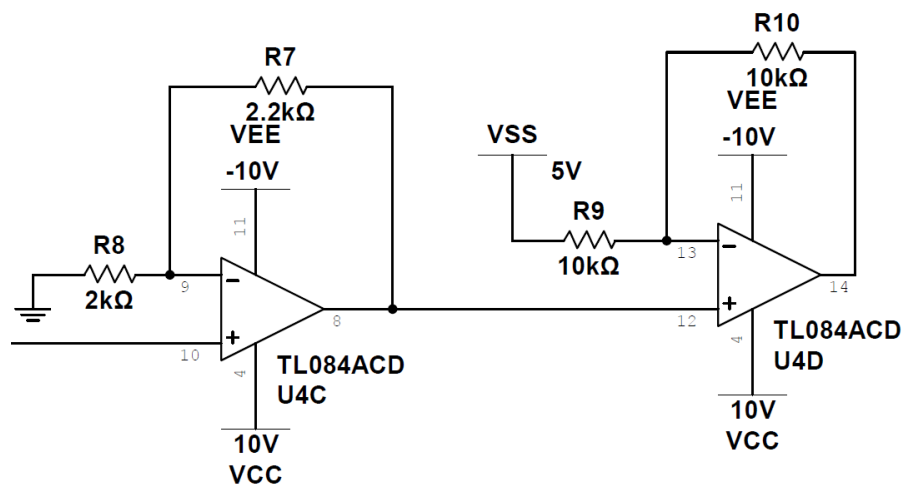


Figura 4.8: Ajuste de offset de la salida del DAC de Arduino Due.

Se deben realizar dos circuitos, uno para cada DAC del Arduino Due.

Para verificar el correcto funcionamiento del circuito se le realiza un barrido de voltaje, esto con la finalidad de observar la respuesta y asegurar que su comportamiento es lineal. El barrido de voltaje se realiza enviando al DAC del Arduino Due valores en intervalos de 50 con lo que se obtienen 82 muestras de valores analógicos en el rango de -5V a 5V, con un multímetro se toman los valores analógicos obtenidos de cada muestra, este procedimiento se realiza a ambos circuitos con lo que se obtienen las siguientes

gráficas.

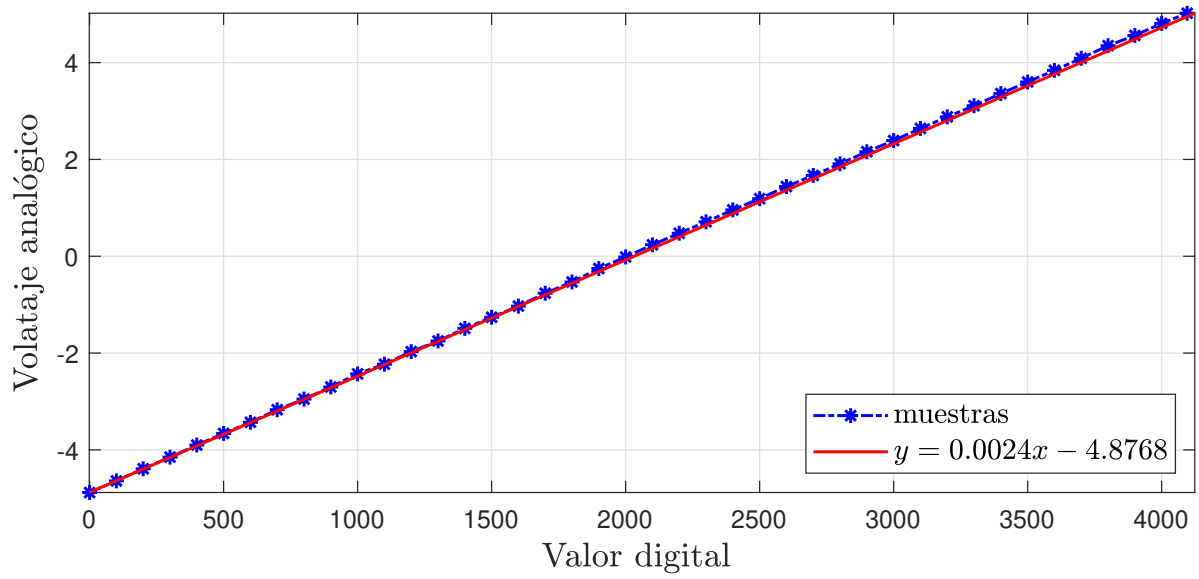


Figura 4.9: Respuesta del DAC0 de Arduino Due.

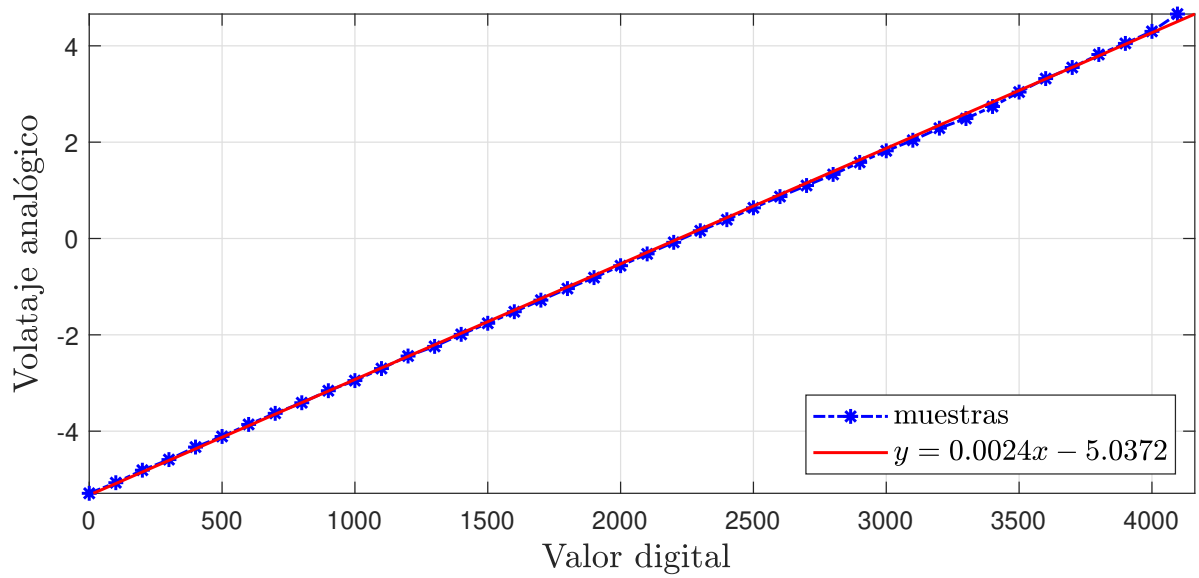


Figura 4.10: Respuesta del DAC1 de Arduino Due.

En la Figura 4.11 se presenta el diagrama general del sistema donde se observa la electrónica que se emplea para la lectura de la posición actual del robot y la electrónica para el envío de la señal de control τ , también se presenta la tarjeta que nos permite

realizar adquisición de datos la cual se realiza mediante un cable USB conectado del Arduino Due a una computadora en la que se realizan las gráficas con los datos obtenidos y procesados por la tarjeta Arduino Due.

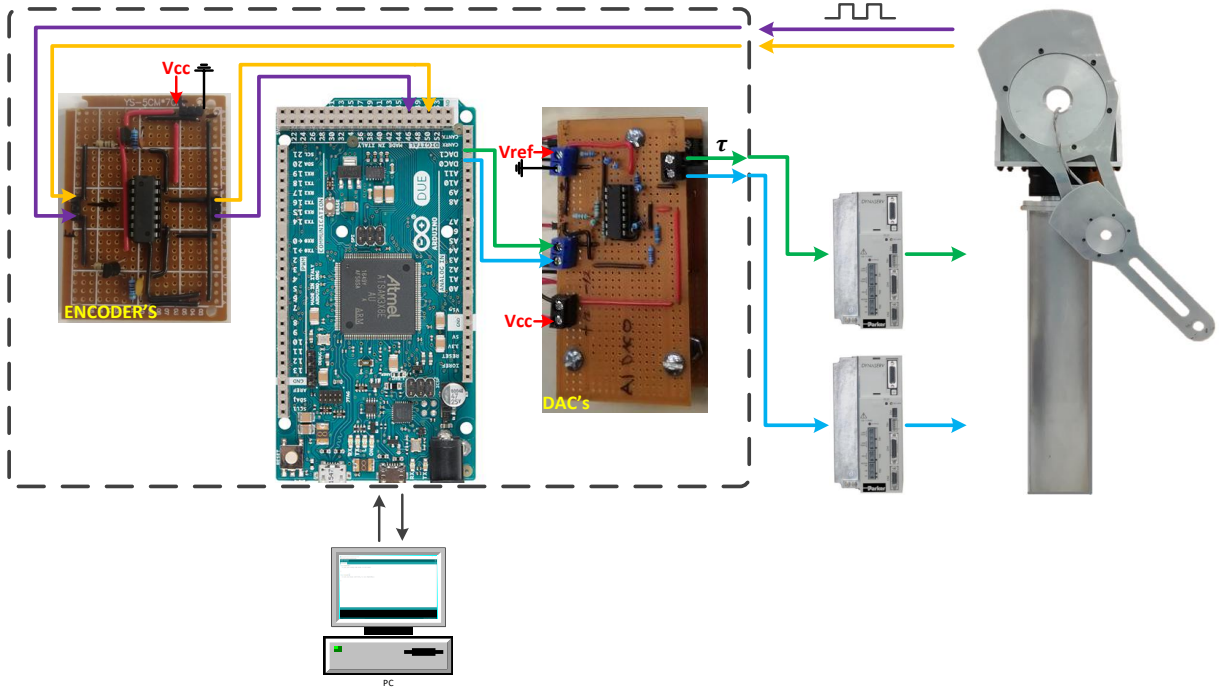


Figura 4.11: Diagrama general de la planta experimental.

4.4. Validación experimental de la instrumentación

Para validar experimentalmente la electrónica de acoplamiento de las señales de salida de los encoders del robot manipulador, así como la electrónica que nos permite enviar la señal de control de la tarjeta Arduino Due al robot manipulador de transmisión directa, se realiza la implementación del algoritmo de control de trayectoria Backstepping [41] como se muestra en la siguiente ecuación:

$$\tau = M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + g(q) - K_v s - K_p \tilde{q}, \quad (4.7)$$

donde $\tilde{q} = q_d - q$ es el error de posición, $K_v \in \mathbb{R}^{n \times n}$ y $K_p \in \mathbb{R}^{n \times n}$ son matrices diagonales de ganancias definidas positivas y q_r es la referencia virtual de velocidad y se define como

$$\dot{q}_r := -\dot{q}_d - \Lambda \tilde{q}, \quad (4.8)$$

además, s es la entrada virtual de control que se define como

$$s := \dot{\tilde{q}} + \Lambda \tilde{q}, \quad (4.9)$$

donde $\Lambda \in \mathbb{R}^{n \times n}$ es una matriz diagonal de ganancias definida positiva.

La trayectoria deseada q_d es un círculo con centro en el plano cartesiano $(x_c, y_c) = (0.3, -0.3)$ y un radio $r = 0.2m$. La ecuación que describe al círculo se muestra a continuación

$$x_d = x_c + r \sin(\omega t), \quad (4.10)$$

$$y_d = -y_c + r \cos(\omega t), \quad (4.11)$$

donde $\omega = 0.5$ representa a la frecuencia, por lo tanto, le tomara al robot manipulador de transmisión directa 12.571 segundos en realizar el círculo.

Debido a que el algoritmo de control (4.7) requiere del cálculo de la velocidad y aceleración deseada es necesario calcular la primera y segunda derivada de (4.10) y (4.11) por lo tanto se tiene

$$\dot{x}_d = r \cos(\omega t), \quad (4.12)$$

$$\dot{y}_d = -r \sin(\omega t). \quad (4.13)$$

La segunda derivada de las ecuaciones (4.12) y (4.15) se expresa como

$$\ddot{x}_d = -r \sin(\omega t), \quad (4.14)$$

$$\ddot{y}_d = -r \cos(\omega t). \quad (4.15)$$

Las posiciones deseadas en coordenadas articulares se obtienen a partir de la cinemática inversa del robot manipulador de 2 grados de libertad asociada al cuarto cuadrante como se realiza en [25] donde se tiene

$$q_{d1} = \frac{\pi}{2} + \tan^{-1} \left(\frac{y_d}{x_d} \right) - \tan^{-1} \left(\frac{l_2 \sin(q_{d2})}{l_1 + l_2 \cos(q_{d2})} \right), \quad (4.16)$$

$$q_{d2} = \frac{\cos^{-1}(x_d^2 + y_d^2 - l_1^2 - l_2^2)}{2l_1 l_2}. \quad (4.17)$$

La velocidad y la aceleración articular deseada se obtienen como

$$\dot{q}_d = J^{-1}(q_d)\dot{x}_d, \quad (4.18)$$

$$\ddot{q}_d = J^{-1}(q_d)\ddot{x}_d - J^{-1}(q_d)\dot{J}(q_d)J^{-1}(q_d)\dot{x}_d, \quad (4.19)$$

donde $J^{-1}(q_d)$ es el Jacobiano inverso, $\dot{J}(q_d)$ es la deriva temporal del Jacobiano y $\dot{x}_d$, $\ddot{x}_d$ son la primera y segunda derivada de la trayectoria en coordenadas cartesianas. La implementación del control en el ambiente de programación de arduino se realiza como se presenta en el diagrama de flujo de la Figura 4.12.

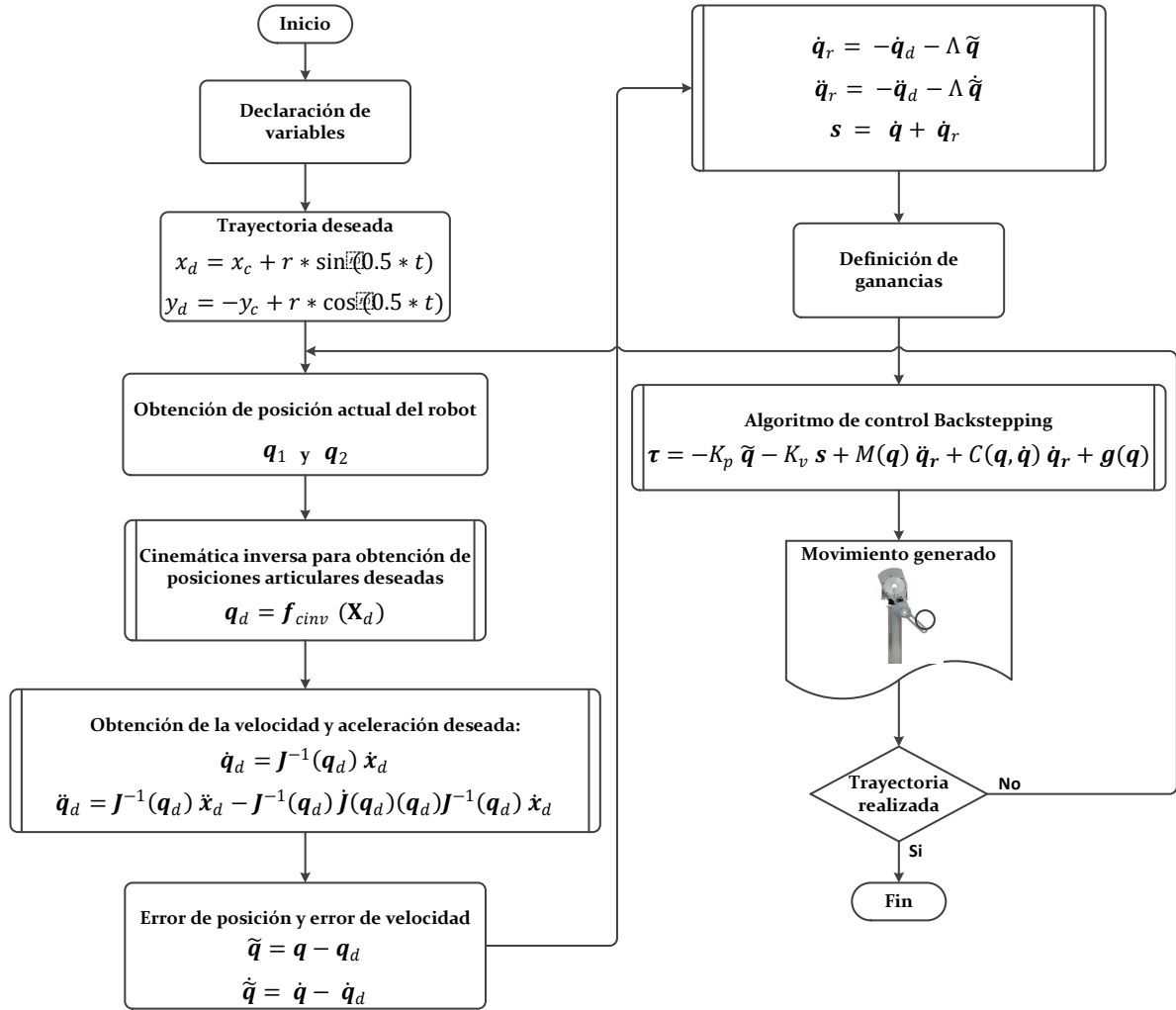


Figura 4.12: Diagrama de flujo para la implementación del algoritmo de control Backstepping.

Las ganancias que se emplean para la implementación del algoritmo de control backs-

tepping se presentan en la siguiente tabla.

Ganancia	Valor
K_{p1}	128
K_{p2}	28
K_{v1}	$0.5 * K_{p1}$
K_{v2}	$0.5 * K_{p2}$
Λ_1	1.5
Λ_2	2

Tabla 4.2: Ganancias para el control backstepping.

En la Figura 4.13, se muestra la gráfica de trayectoria deseada en línea azul y la trayectoria realizada por el robot manipulador de transmisión directa de dos grados de libertad en la línea roja, se observa que con las ganancias propuestas para realizar la trayectoria deseada el robot manipulador no alcanza a la posición deseada $q_d(t)$ teniendo un error de posición en su punto máximo de aproximadamente 3 grados.

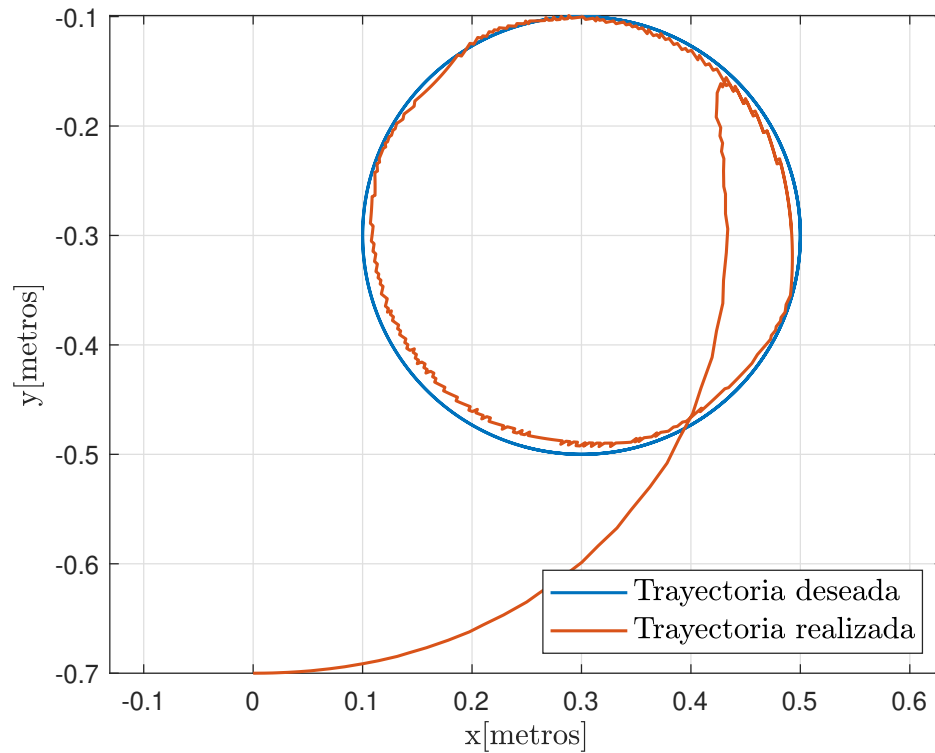


Figura 4.13: Gráfica comparativa, trayectoria realizada y trayectoria deseada por el robot manipulador de transmisión directa.

Aunque la trayectoria deseada no fue alcanzada por el robot manipulador de transmisión directa no significa que para el sistema sea difícil alcanzar a la posición deseada, lo que significa es que se necesita realizar una correcta estrategia matemática de sintonización de ganancias con las que a través del empleo de cotas máximas y mínimas en pares máximos permitidos así como las cotas en velocidad máxima permitida por los servo motores que influyen directamente en la estructura matemática de el algoritmo de control de trayectoria se obtendría una mejora en los resultados obtenidos ya que en este caso la elección de las ganancias empleadas se realizó de forma empírica.

En la Figura 4.14, se muestra primero la gráfica de las posiciones articulares que realiza el robot manipulador de transmisión directa con las que se obtiene la trayectoria deseada, después se presenta la gráfica de los errores de posición para el eslabón 1 (q_1) y el eslabón 2 (q_2), donde se visualiza lo que ya se había evidenciado en la gráfica de la Figura 4.13, que el robot manipulador tiene un error de posición. También se puede visualizar que al robot manipulador le toma 2 segundos en aproximarse a las posiciones deseadas partiendo desde su posición de reposo la cual está ubicado sobre el eje negativo y del plano cartesiano.

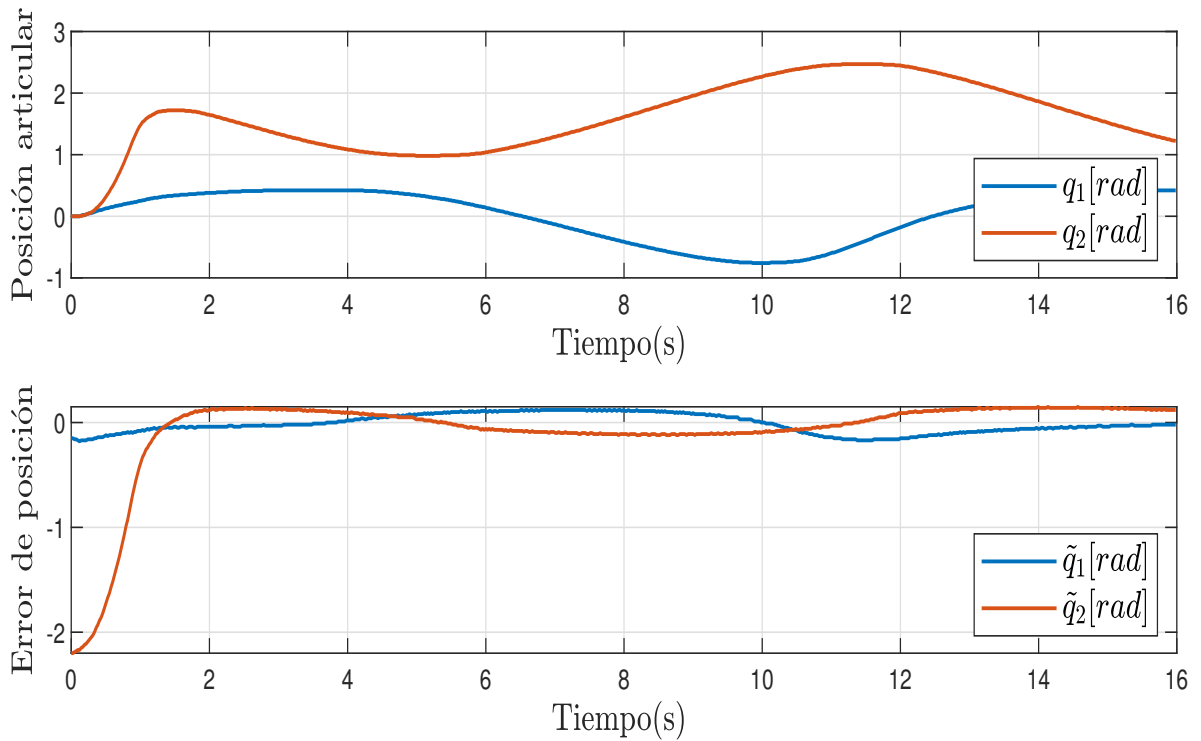


Figura 4.14: a) Posiciones articulares. b) Error de posición.

En la Figura 4.15, se presentan las gráficas de torques o pares que se envían a los servomotores del robot manipulador como señal de control, en la primera parte se presenta el torque empleado para el servomotor de la articulación 1 se observa que el torque máximo no sobrepasa los 50 Nm permitidos por el servomotor, de igual forma se observa la gráfica de torque para el servomotor de la articulación 2, aunque el torque máximo que puede proporcionar el servomotor es de apenas 4 Nm es suficiente para realizar la trayectoria.

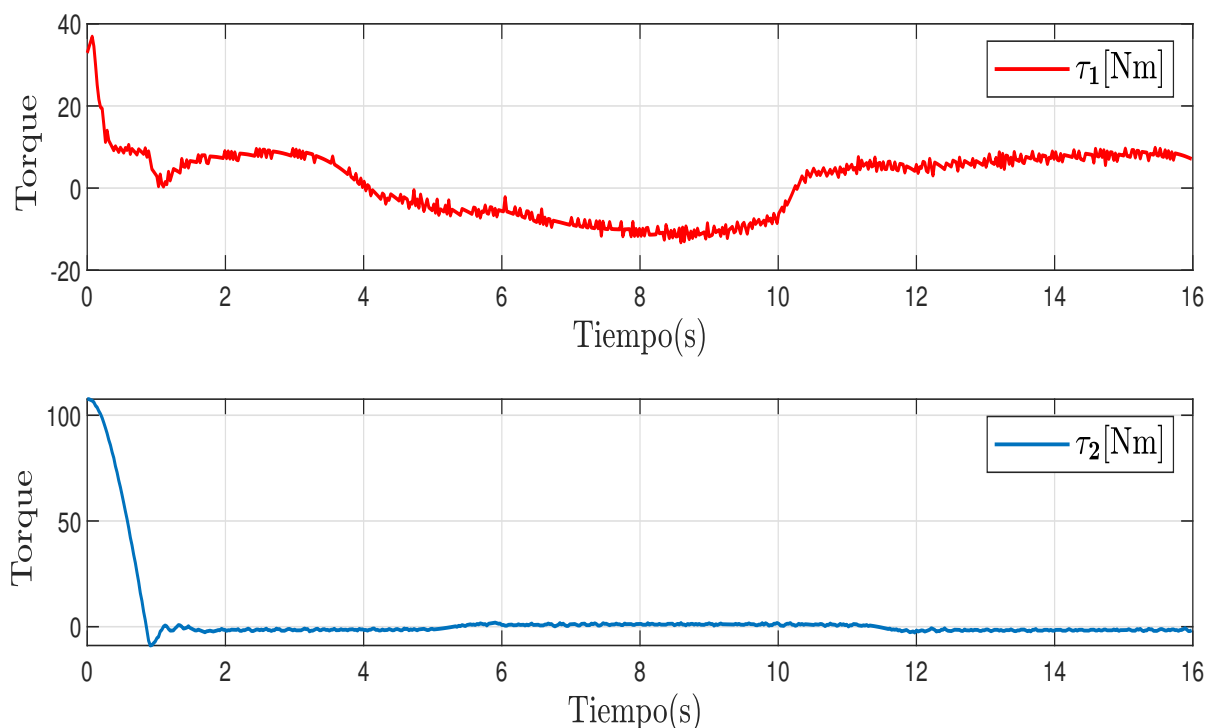


Figura 4.15: Torques empleados para realizar la trayectoria.

4.5. Hardware in the loop

En la actualidad existen sistemas que nos permitan implementar controladores en una plataforma electrónica (microcontrolador) y poder observar su comportamiento en una planta experimental virtual. Hardware In the Loop (HIL) es una técnica que permite realizar pruebas en sistemas integrados de forma simple pero que asegura que la respuesta obtenida es repetible y confiable. Además, garantiza que es fácilmente adaptable a un sistema físico. En este trabajo se realiza la implementación de un sistema Hardware In the Loop (HIL), en el que se integra el uso de una tarjeta Arduino Due y se combina

con el uso de software de simulación de sistemas Simulink, como se muestra en la Figura 4.16.

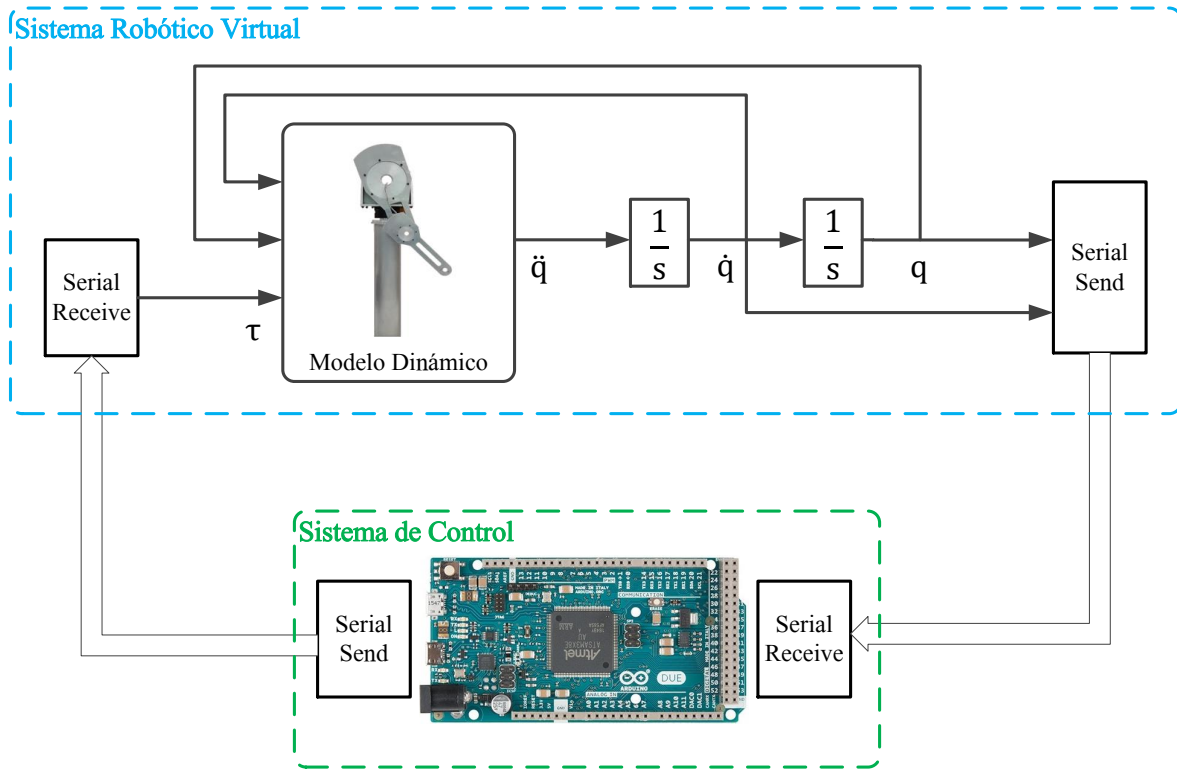


Figura 4.16: Diagrama de implementación Hardware In the Loop.

4.5.1. Validación experimental del sistema Hardware In the Loop

Para validar de forma experimental que el sistema Hardware In the Loop funciona de forma correcta y que los resultados obtenidos se aproximan con los resultados de la implementación del sistema físico de la Figura 4.11 se propone que el sistema Hardware In the Loop realice la misma trayectoria que el robot manipulador de transmisión directa físico, con el mismo control de trayectoria (4.7).

La forma en que se realiza es implementando el algoritmo de control en la tarjeta Arduino Due a la vez que esta se comunica de forma serial a una computadora en la que se encuentra el software Simulink de acuerdo al diagrama de la Figura 4.16. Como se observa en la Figura 4.16 se hace uso de 2 puertos seriales del Arduino Due, el primero es el nativo de arduino y no necesita ser adecuado para poder conectarlo a la computadora. Sin embargo, el segundo puerto se trata del serial 1 y este solo es accesible por los pines RX1 y TX1, para poder emplear el puerto serial 1 del Arduino Due el cual se encarga de la recepción de datos se debe considerar que la comunicación serial en los pines RX y TX

de la placa Arduino Due usa niveles lógicos TTL de 3.3V, por lo que es necesario hacer uso de un adaptador USB a serial, por lo que en este trabajo se hace uso del adaptador FTDI232RL, Figura 4.17.

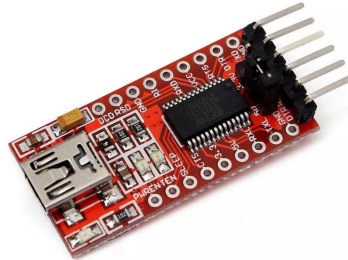


Figura 4.17: Adaptador FTDI232R.

El diagrama de conexiones del adaptador FTDI232R a Arduino Due se muestra en la Figura 4.18.

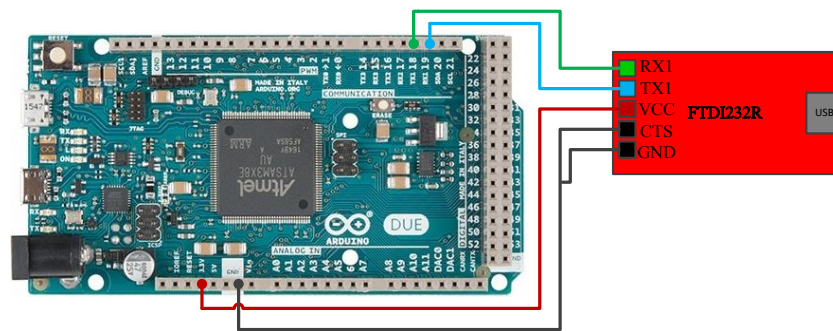


Figura 4.18: Diagrama de conexión del Arduino due y el adaptador FTDI.

La implementación del algoritmo de control (4.7) en el script de arduino se realiza como se muestra en el diagrama de flujo de la Figura 4.19, en este se comienza con la declaración de variables necesarios para la programación después se inicializan dos puertos seriales, *Serial.begin()* que envía la señal de control al robot virtual en Simulink y *Serial1.begin()* que recibe los estados que envía Simulink al Arduino Due y se configuran a 115200 baudios para la transmisión y recepción de datos. En el script de arduino también se hace uso de una unión la cual en C es un tipo de dato especial que está formada por varios campos de variables de diferentes tipos de datos que comparten la misma localidad o espacio de memoria, de tal forma que si una variable modifica su valor, de la misma manera lo harán los demás campos como se muestra en [38]. Los programas empleados para el Arduino Due se pueden consultar en A.1 y A.2.

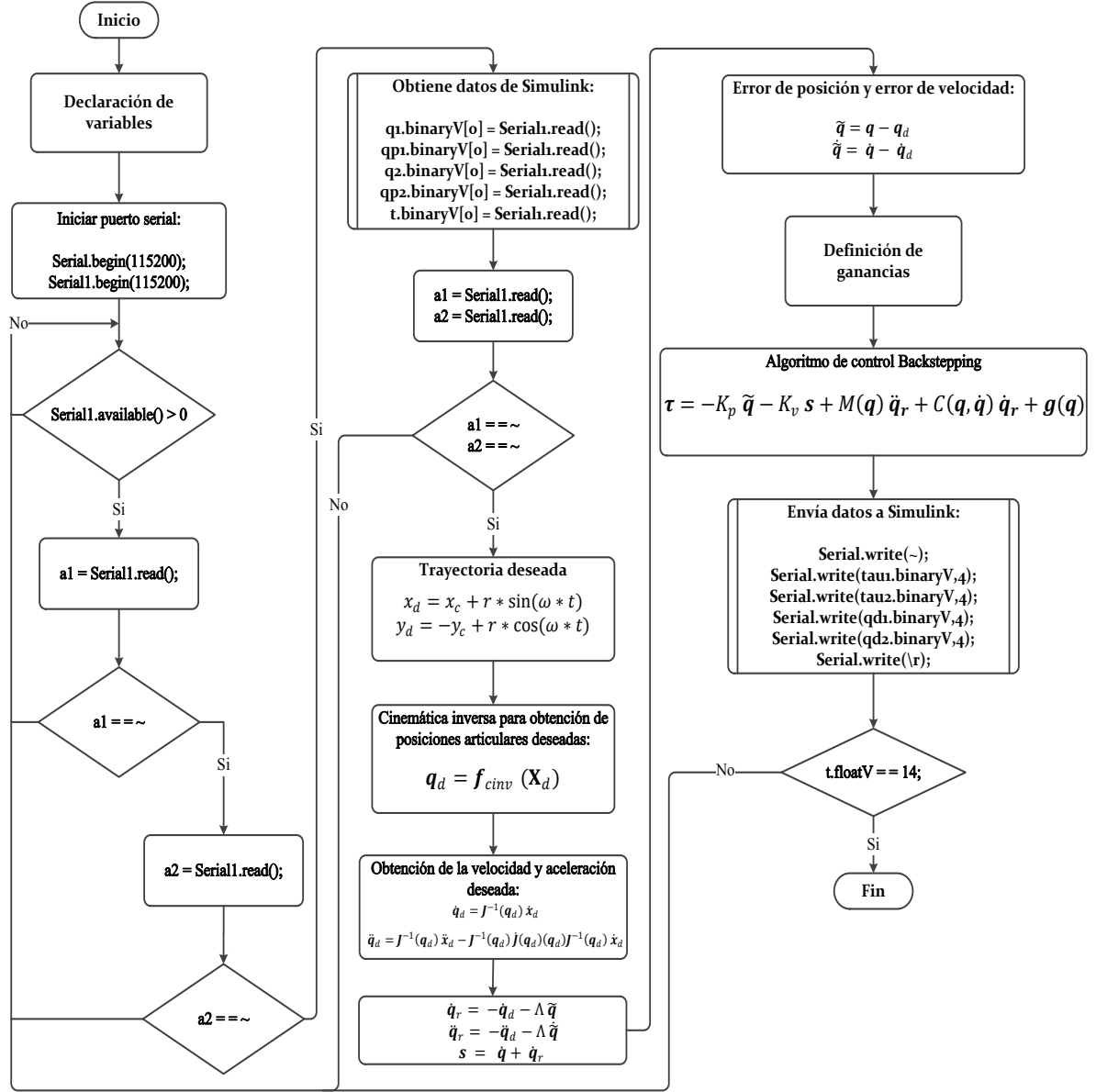


Figura 4.19: Diagrama de implementación Hardware In the Loop en la tarjeta Arduino Due.

En Simulink se implementa el modelo dinámico (1.27) del robot manipulador de transmisión directa de la Figura 1.3, este retroalimenta su posición y velocidad articular haciendo uso de comunicación serial a la tarjeta Arduino Due.

En el ambiente de programación de Simulink se hace uso del bloque *Serial Receive* y *Serial Send* de la caja de herramientas *Instrument Control*. *Serial Receive* se emplea para recibir la señal de control τ que procesa arduino, en este bloque es necesario configurar los siguientes parámetros

- **Communication port:** Especifica el puerto serial para recibir datos. Cada bloque *Serial Receive* debe tener un bloque *Serial Configuration*. Si se utiliza varios puertos, se debe configurar cada puerto por separado.
- **Header:** especifica los datos que marcan el inicio de la recepción de datos. El encabezado indica el comienzo de un nuevo bloque de datos y la simulación ignorará los datos que ocurran antes del encabezado. Por defecto, no se especifica ninguno o ningún encabezado.
- **Terminator:** especifica los datos que marcan el final del bloque de datos. El terminador indica el final del bloque de datos y la simulación tendrá en cuenta cualquier dato que ocurra después del terminador como un nuevo bloque de datos. Por defecto no se especifica ningún terminador. Otros formatos de terminador disponibles son:
 - CR ($\backslash r$): retorno de carro
 - LF ($\backslash n$): nueva línea
 - CR/LF ($\backslash r \backslash n$): retorno de carro y nueva línea
 - NULL ($\backslash 0$): byte nulo
- **Data size:** Especifica el tamaño de los datos de salida o el número de valores que deben leerse en cada paso de tiempo de la simulación.
- **Data type:** Especifica el tipo de datos de salida para recibir del bloque.

En la siguiente figura se muestra la configuración realizada del bloque *Serial Receive*.

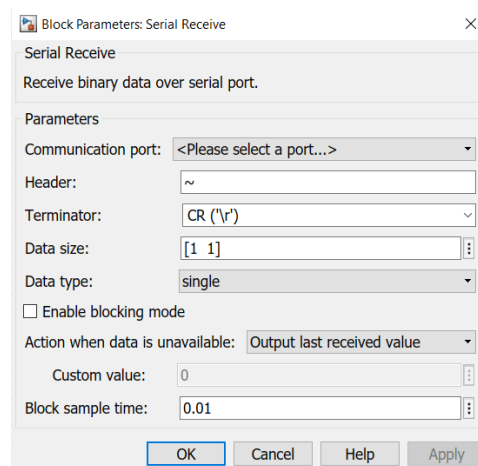


Figura 4.20: Implementación del modelo dinámico en Simulink para el sistema HIL.

El bloque *Serial Send* se emplea para enviar datos de posiciones articulares $q_{1,2}$ y velocidades articulares $\dot{q}_{1,2}$, a través del puerto serial a la tarjeta Arduino Due. La configuración del bloque *Serial Send* solo requiere de los parámetros, Communication port, Header y Terminator. En la siguiente figura se muestra la configuración del bloque *Serial Send*.

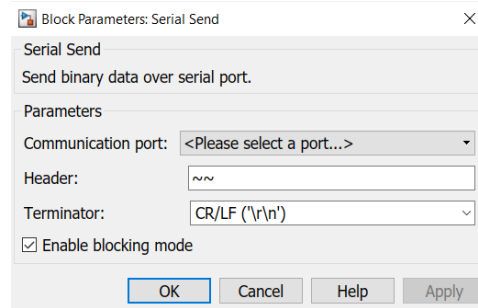


Figura 4.21: Implementación del modelo dinámico en Simulink para el sistema HIL.

En la Figura 4.22, se presenta la implementación del sistema Hardware In the Loop, con el que se simula al robot manipulador de transmisión directa de dos grados de libertad y el control es realizado por la tarjeta Arduino Due. Los programas empleados pueden ser consultados en A.3.

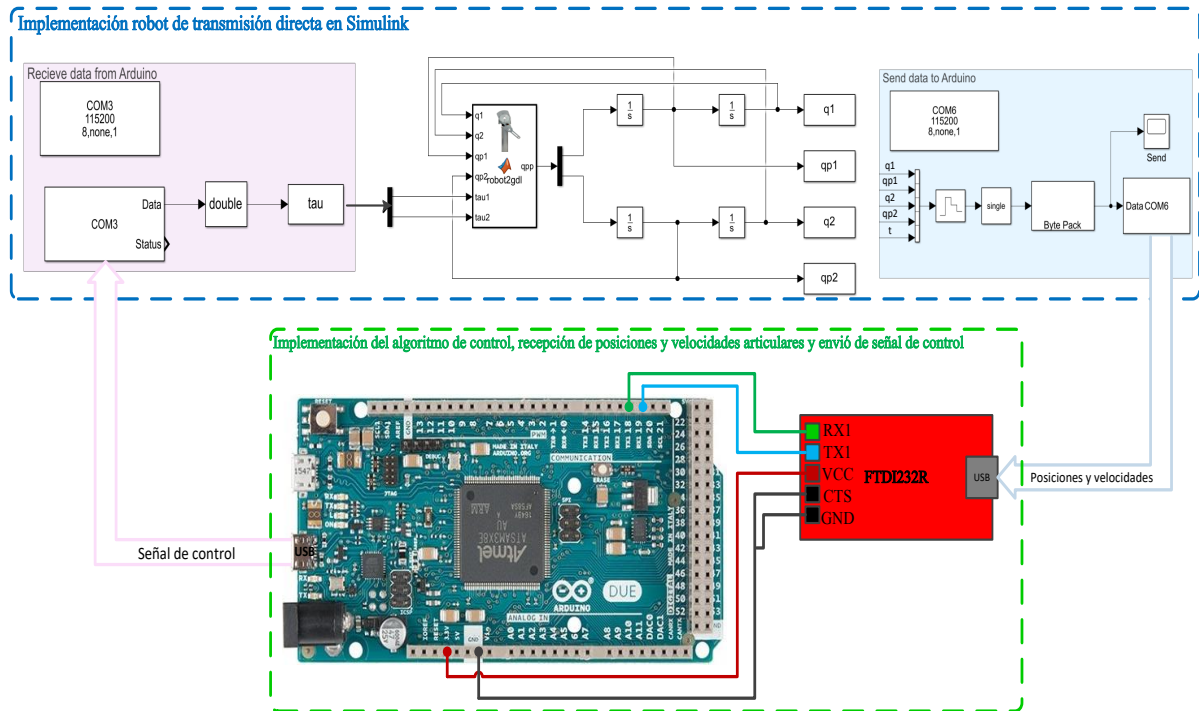


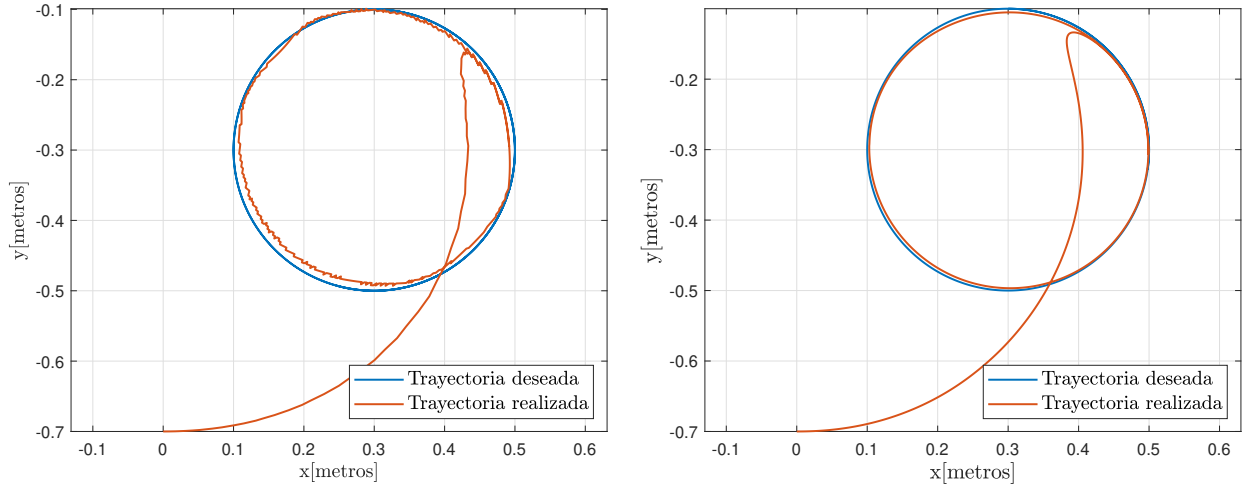
Figura 4.22: Implementación del modelo dinámico en Simulink para el sistema HIL.

Las ganancias que se emplean para la implementación del algoritmo de control backstepping en Hardware In the Loop se presentan en la siguiente tabla.

Ganancia	Valor
K_{p1}, K_{p2}	128, 28
K_{v1}, K_{v2}	$0.5 * K_{p1}, 0.5 * K_{p2}$
Λ_1, Λ_2	1.5, 2

Tabla 4.3: Ganancias para el control backstepping en el sistema HIL.

En la Figura 4.23, se presenta la gráfica de la trayectoria realizada por el sistema HIL comprándola con la trayectoria deseada que realiza el robot de transmisión directa de la Figura 1.3. Se puede observar que el comportamiento del sistema es similar al del sistema físico como se presenta, sin embargo, este presenta pequeños errores en su comportamiento debido a que en este no están modelados los fenómenos de fricción que podemos encontrar de forma natural en el sistema físico.



(a) Trayectoria realizada y trayectoria deseada por el robot manipulador de transmisión directa HIL.

(b) Trayectoria realizada y trayectoria deseada por el robot manipulador de transmisión directa físico.

Figura 4.23: Comparación de resultados.

En la Figura 4.24, se presenta la gráfica de las posiciones articulares en la parte superior y en la parte inferior se presenta el error de posición durante el seguimiento de la trayectoria deseada. En la Figura 4.25, se presenta la gráfica de torques con las que el

robot manipulador de transmisión directa virtual realiza la trayectoria. En estas también se pueden observar pequeñas perturbaciones como se observa en la Figura 4.15.

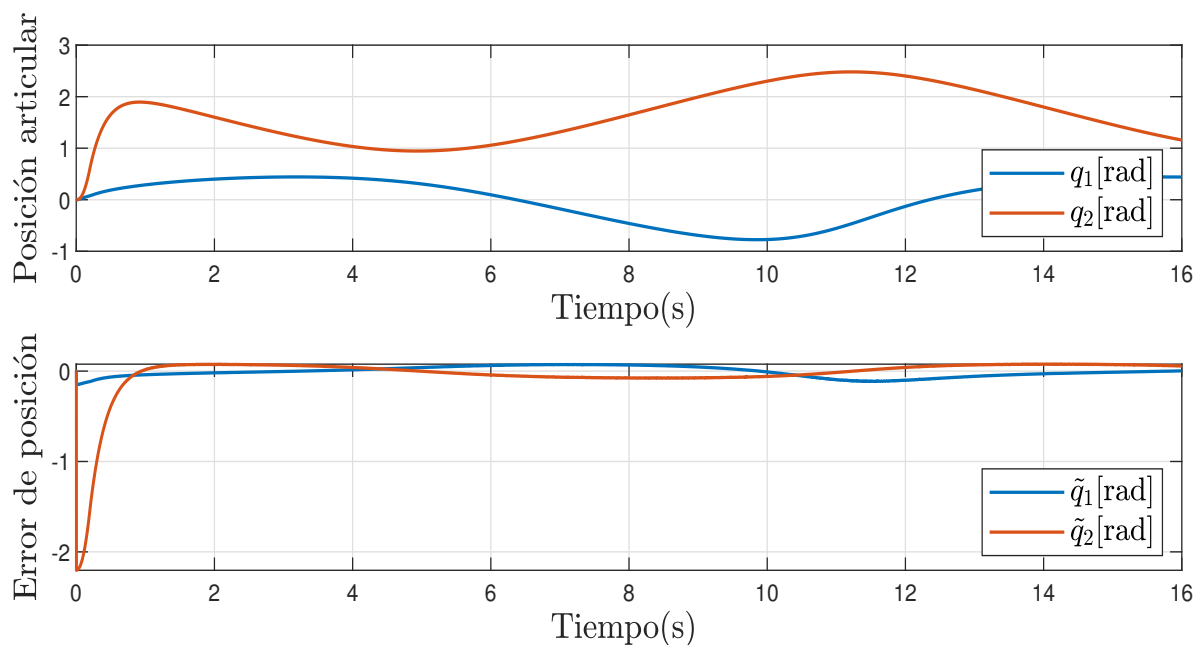


Figura 4.24: a) Posiciones articulares. b) Error de posición.

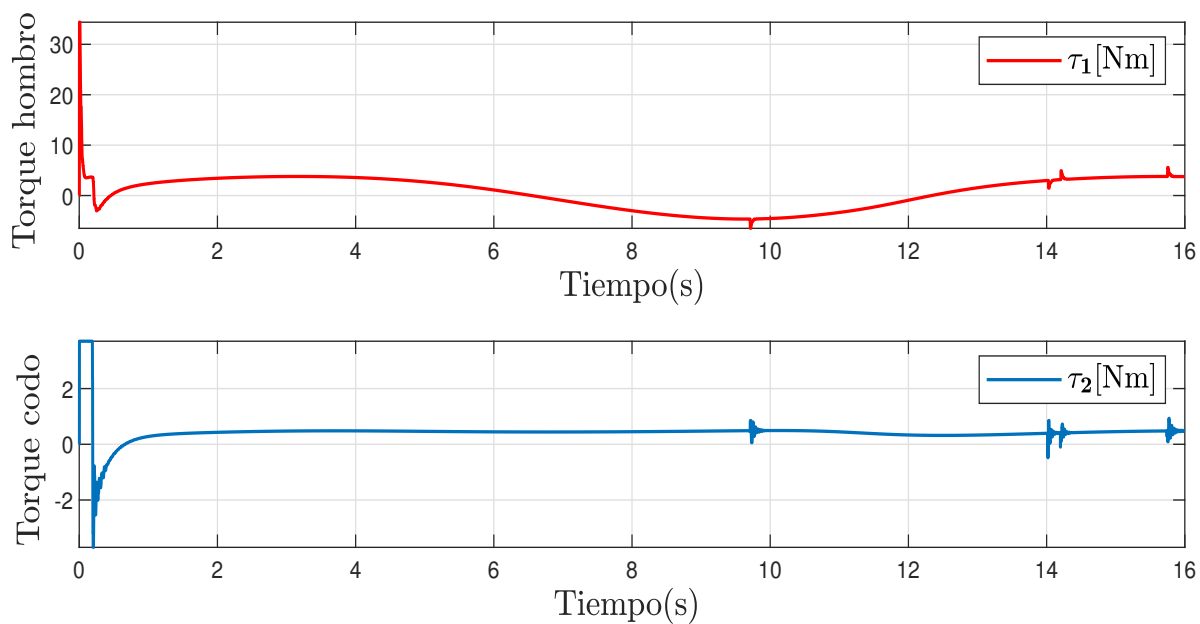


Figura 4.25: Torques empleados para realizar la trayectoria.

4.6. Observaciones

En este capítulo se realizó el estudio de instrumentación de la planta experimental con la que se integra al sistema multiagente. Primero se realizó el estudio a la planta física y después se propuso el uso del sistema Hardware In the Loop la cual nos permite poder ampliar a un grupo más amplio de robots manipuladores en los que se pueda implementar los algoritmos de consenso considerando que se emplean las mismas condiciones iniciales y modelo dinámico del robot manipulador.

Capítulo 5

Resultados

En este capítulo se muestran los resultados obtenidos. Se muestra la propuesta del sistema multiagente el cual está formado por robots manipuladores de dos grados de libertad y se encuentran interconectados bajo una topología no dirigida. Además, se muestran las gráficas que nos permiten verificar que el sistema multiagente ha alcanzado un punto en común o valor constante lo que permite validar la respuesta de los algoritmos de consenso.

5.1. Sistema multiagente de robots manipuladores

Los algoritmos de consenso que han sido objeto de estudio en este trabajo son implementados en un sistema multiagente formado por tres robots de transmisión directa de tres grados de libertad idénticos. Empleando el siguiente modelo dinámico y los valores numéricos de la Tabla 1.1 se expresa la matriz de inercia, la matriz de fuerzas centrípetas y de Coriolis y el vector de pares gravitacionales para cada uno de los agentes como

$$M_i(q_i) = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}, \quad (5.1)$$

$$C_i(q_i, \dot{q}_i) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}, \quad (5.2)$$

$$g_i(q_i) = g \begin{bmatrix} gq_{11} \\ gq_{21} \\ gq_{31} \end{bmatrix}, \quad (5.3)$$

donde $i = 1, 2, 3$.

Los componentes para la matriz de inercias $M_i(q_i)$ tienen la siguiente forma.

$$M_{11} = I_{y2} \sin(q_{i(2)}) \sin(q_{i(2)}) + I_{y3} \sin(q_{i(2)} + q_{i(3)}) \sin(q_{i(2)} + q_{i(3)}) + I_{z1} + I_{z2} \cos(q_{i(2)}) \cos(q_{i(2)}) + I_{z3} \cos(q_{i(2)} + q_{i(3)}) \cos(q_{i(2)} + q_{i(3)}) + m_2 l_{c2}^2 \cos(q_{i(2)}) \cos(q_{i(2)}) + m_3 (l_2 \cos(q_{i(2)}) + l_{c3} \cos(q_{i(2)} + q_{i(3)})) (l_2 \cos(q_{i(2)}) + l_{c3} \cos(q_{i(2)} + q_{i(3)})), \quad (5.4)$$

$$M_{12} = 0, \quad (5.5)$$

$$M_{13} = 0, \quad (5.6)$$

$$M_{21} = 0, \quad (5.7)$$

$$M_{22} = I_{x2} + I_{x3} + m_3 l_2^2 + m_2 l_{c2}^2 + m_3 l_{c3}^2 + 2m_3 l_2 l_{c3} \cos(q_{i(3)}), \quad (5.8)$$

$$M_{23} = I_{x3} + m_3 l_{c3}^2 + m_3 l_2 l_{c3} \cos(q_{i(3)}), \quad (5.9)$$

$$M_{31} = 0, \quad (5.10)$$

$$M_{32} = I_{x3} + m_3 l_{c3}^2 + m_3 l_2 l_{c3} \cos(q_{i(3)}), \quad (5.11)$$

$$M_{33} = I_{x3} + m_3 l_{c3}^2. \quad (5.12)$$

Componentes para la matriz de fuerzas centrípetas y de coriolis $C_i(q_i, \dot{q}_i)$.

$$\gamma_{112} = (I_{y2} - I_{x2} - m_2 l_{c2}^2) \cos(q_{i(2)}) \sin(q_{i(2)}) + (I_{y3} - I_{z3}) \cos(q_{i(2)} + q_{i(3)}) \sin(q_{i(2)} + q_{i(3)}) - m_3 (l_2 \cos(q_{i(2)}) + l_{c3} \cos(q_{i(2)} + q_{i(3)})) (l_2 \sin(q_{i(2)}) + l_{c3} \sin(q_{i(2)} + q_{i(3)})), \quad (5.13)$$

$$\gamma_{113} = (I_{y3} - I_{z3}) \cos(q_{i(2)} + q_{i(3)}) \sin(q_{i(2)} + q_{i(3)}) - m_3 l_{c3} \sin(q_{i(2)} + q_{i(3)}) (l_2 \cos(q_{i(2)}) + l_{c3} \cos(q_{i(2)} + q_{i(3)})), \quad (5.14)$$

$$\gamma_{121} = (I_{y2} - I_{z2}) \cos(q_{i(2)} + q_{i(3)}) \sin(q_{i(2)} + q_{i(3)}) - m_3 l_{c3} \sin(q_{i(2)} + q_{i(3)}) (l_2 \cos(q_{i(2)}) + l_{c3} \cos(q_{i(2)} + q_{i(3)})), \quad (5.15)$$

$$\gamma_{131} = (I_{y3} - I_{z3}) \cos(q_{i(2)} + q_{i(3)}) \sin(q_{i(2)} + q_{i(3)}) - m_3 l_{c3} \sin(q_{i(2)} + q_{i(3)}) (l_2 \cos(q_{i(2)}) + l_{c3} \cos(q_{i(2)} + q_{i(3)})), \quad (5.16)$$

$$\gamma_{211} = (I_{y2} - I_{x2} - m_2 l_{c2}^2) \cos(q_{i(2)}) \sin(q_{i(2)}) + (I_{y3} - I_{z3}) \cos(q_{i(2)} + q_{i(3)}) \sin(q_{i(2)} + q_{i(3)}) + m_3 (l_2 \cos(q_{i(2)}) + l_{c3} \cos(q_{i(2)} + q_{i(3)})) (l_2 \sin(q_{i(2)}) + l_{c3} \sin(q_{i(2)} + q_{i(3)})), \quad (5.17)$$

$$\gamma_{223} = -l_2 m_3 l_{c3} \sin(q_{i(3)}), \quad (5.18)$$

$$\gamma_{232} = -l_2 m_3 l_{c3} \sin(q_{i(3)}), \quad (5.19)$$

$$\gamma_{233} = -l_2 m_3 l_{c3} \sin(q_{i(3)}), \quad (5.20)$$

$$\gamma_{311} = (I_{z3} - I_{y3}) \cos(q_{i(2)} + q_{i(3)}) \sin(q_{i(2)} + q_{i(3)}) + m_3 l_{c3} \sin(q_{i(2)} + q_{i(3)}) (l_2 \cos(q_{i(2)}) + l_{c3} \cos(q_{i(2)} + q_{i(3)})), \quad (5.21)$$

$$\gamma_{322} = l_2 m_3 l_{c3} \sin(q_{i(3)}). \quad (5.22)$$

$$c_{11} = \gamma_{112}\dot{q}_2 + \gamma_{113}\dot{q}_3, \quad (5.23)$$

$$c_{12} = \gamma_{121}\dot{q}_1, \quad (5.24)$$

$$c_{13} = \gamma_{131}\dot{q}_1, \quad (5.25)$$

$$c_{21} = \gamma_{211}\dot{q}_1, \quad (5.26)$$

$$c_{22} = \gamma_{223}\dot{q}_3, \quad (5.27)$$

$$c_{23} = \gamma_{232}\dot{q}_2 + \gamma_{233}\dot{q}_3 \quad (5.28)$$

$$c_{31} = \gamma_{311}\dot{q}_1, \quad (5.29)$$

$$c_{32} = \gamma_{322}\dot{q}_2, \quad (5.30)$$

$$c_{33} = 0. \quad (5.31)$$

Componentes para el vector de pares gravitacionales $g_i(q_i)$.

$$gq_{11} = 0, \quad (5.32)$$

$$gq_{21} = (l_{c2}m_1 + m_2 * l_2)\text{sen}(q_{i(1)}) + m_2l_{c3}\text{sen}(q_{i(1)} + q_{i(2)}), \quad (5.33)$$

$$gq_{31} = m_2l_{c3}\text{sen}(q_{i(1)} + q_{i(2)}). \quad (5.34)$$

Las condiciones iniciales para el robot de transmisión directa 1 son $q_1(0) = [0 \ 0 \ 0]^\top$ rad, $\dot{q}_1(0) = [0 \ 0 \ 0]^\top$ rad/s, para el robot de transmisión directa 2 son $q_2(0) = [0.785398 \ 1.5708 \ 0.785398]^\top$ rad, $\dot{q}_2(0) = [0 \ 0 \ 0]^\top$ rad/s y para el robot de transmisión directa 3 son $q_3(0) = [1.5708 \ 0.785398 \ 1.5708]^\top$ rad, $\dot{q}_3(0) = [0 \ 0 \ 0]^\top$ rad/s.

La implementación y evaluación de los algoritmos de consenso, se realiza en una planta experimental Hardware In the Loop, por lo que cada robot está configurado como se mostró en el Capítulo 3, Figura 4.22, se realiza la adaptación para un robot de transmisión directa de tres grados de libertad empleando (5.1), (5.2) y (5.3) para (1.26). En la Figura 5.1, se muestra la implementación Hardware In the Loop de un robot manipulador de transmisión directa de tres grados de libertad, empleando una tarjeta Arduino Due para implementar los algoritmos de control. De forma individual se implementa para cada agente el algoritmo de consenso, es decir, de forma distribuida en diferentes microcontroladores con características propias del sistema Arduino Due. Ver Figura 5.2.

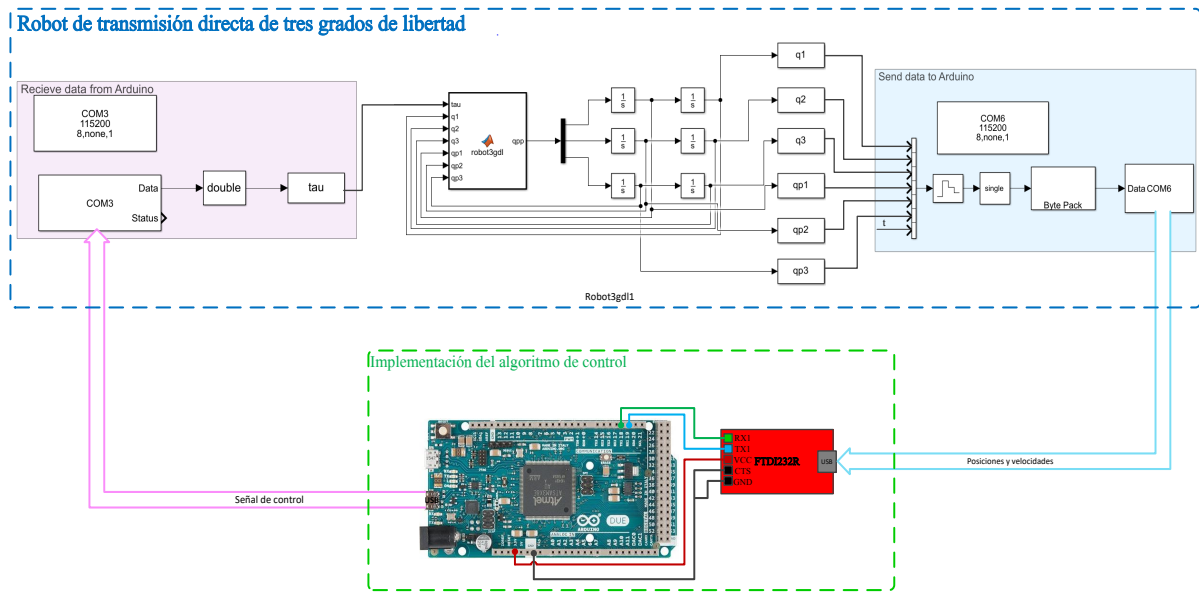


Figura 5.1: Implementación Hardware In the Loop del robot de transmisión directa de tres grados de libertad en Simulink y la tarjeta Arduino Due.

Por lo tanto, el sistema multiagente de robots manipuladores de transmisión directa, que está formado por tres agentes intercambiando su información, se implementa mediante el enfoque Hardware In the Loop como se muestra en la Figura 5.3.

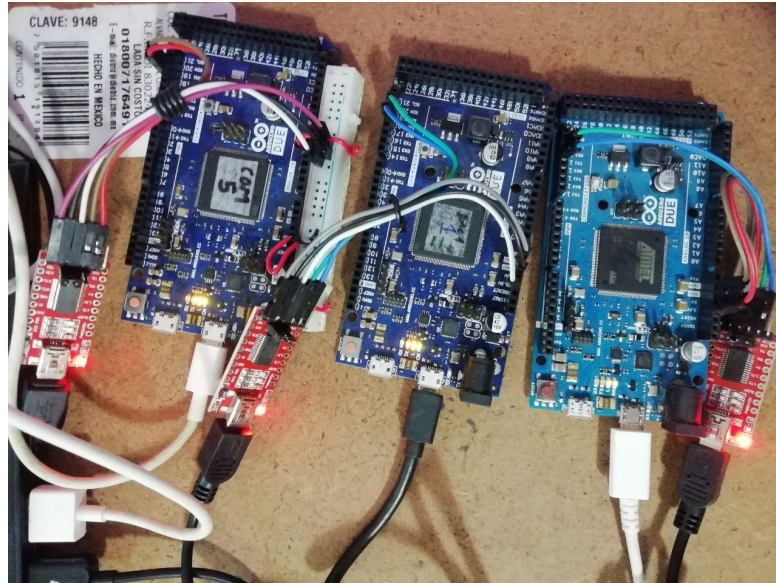


Figura 5.2: Tarjetas donde se realiza la implementación de los algoritmos de consenso de forma distribuida.

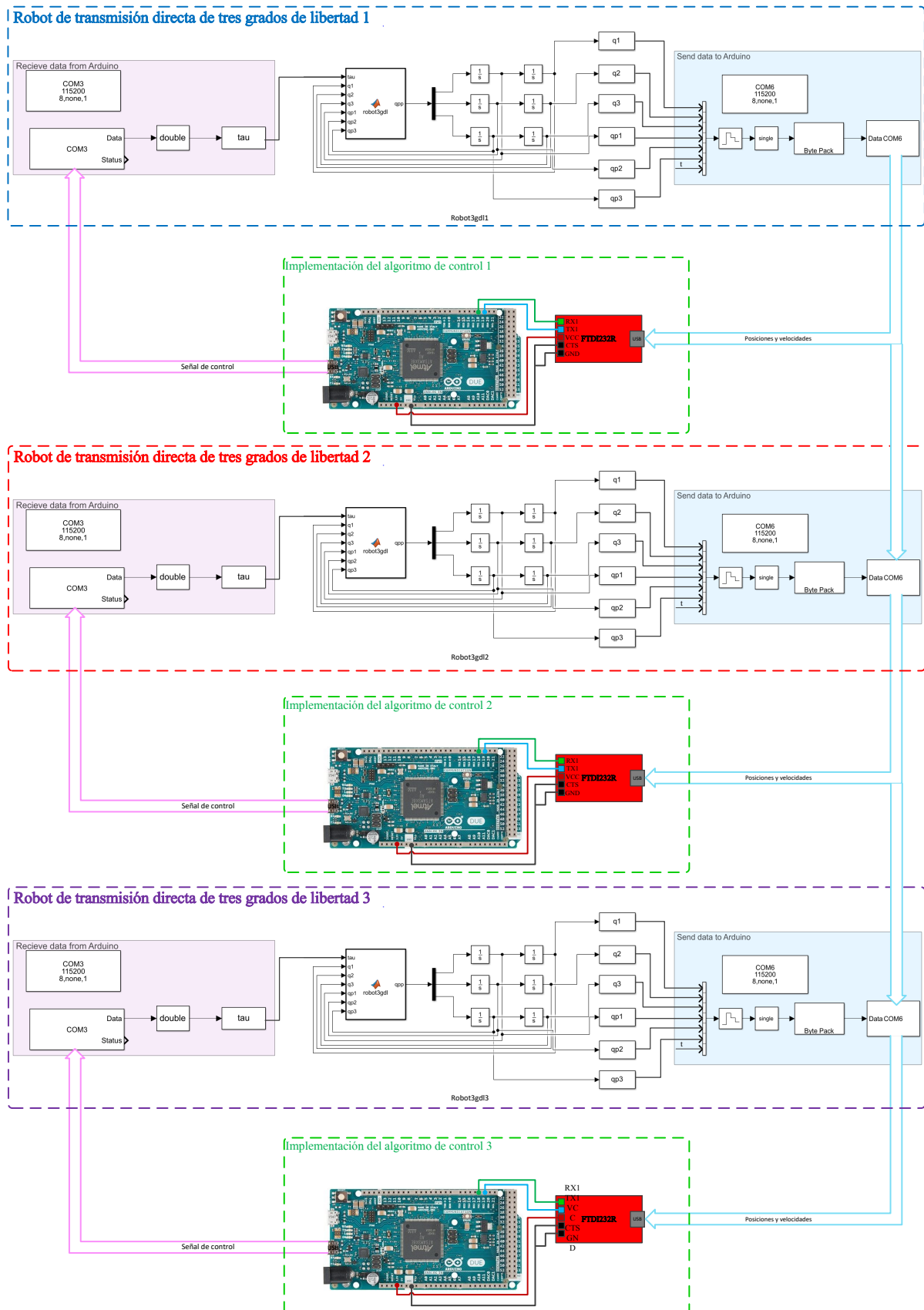


Figura 5.3: Sistema multiagente de robots manipuladores de tres grados de libertad, Hardware In the Loop.

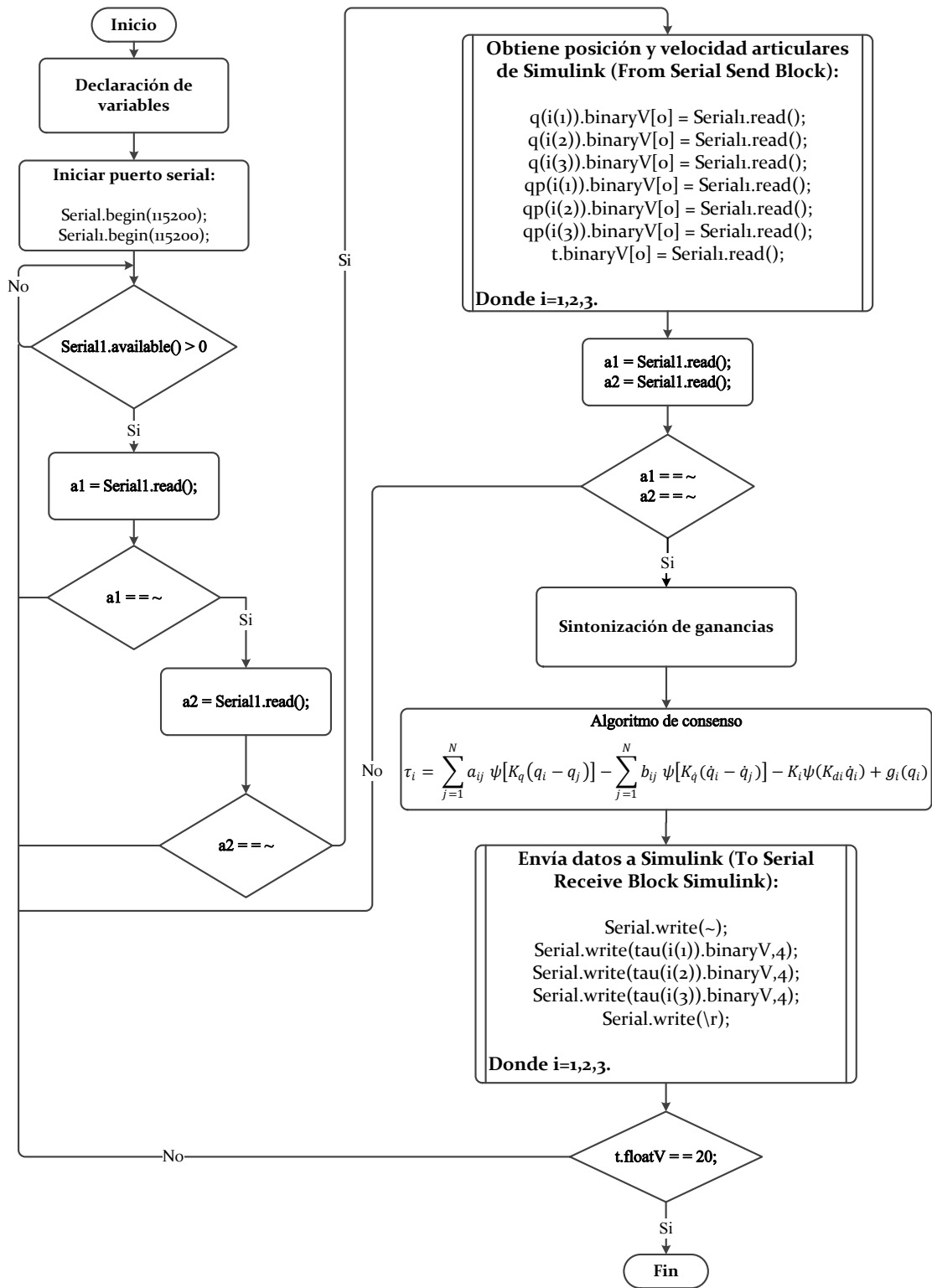


Figura 5.4: Diagrama de flujo para la implementación del algoritmo de consenso.

El intercambio de información entre agentes se modela a través de un grafo no dirigido, por lo que se considera que $\mathcal{G}_A = \mathcal{G}_B$ para el intercambio de posiciones y velocidades articulares como se muestra en la siguiente figura y se toman los valores de la matriz de adyacencia (2.2).

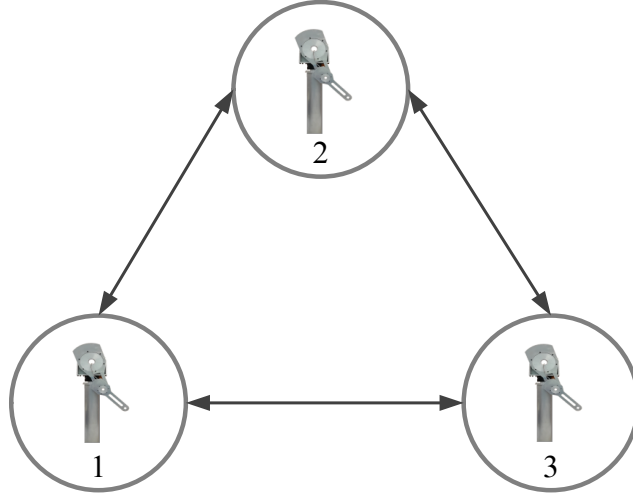


Figura 5.5: Topología de interacción no dirigida para un sistema con tres agentes interconectados transmitiendo su posición y velocidad articular.

5.2. Implementación de los algoritmos de consenso

Considerando el algoritmo de consenso no lineal fundamental (3.34) y la definición de $\psi(\cdot)$, se emplean las funciones $\text{sen}(\cdot)$ y $\text{tanh}(\cdot)$. Por lo que (3.34) toma la siguiente forma para $\text{sen}(\cdot)$,

$$\tau_i = - \sum_{j=1}^n a_{ij} \text{sen}(K_q(q_i - q_j)) - \sum_{j=1}^n b_{ij} \text{sen}(K_{\dot{q}}(\dot{q}_i - \dot{q}_j)) - K_i \text{sen}(K_{d_i} \dot{q}_i) + g_i(q_i). \quad (5.35)$$

El algoritmo de consenso (5.35) para cada agente se expresa como

$$\begin{aligned} \tau_1 = & -a_{12} \text{sen}(K_q(q_1 - q_2)) - b_{12} \text{sen}(K_{\dot{q}}(\dot{q}_1 - \dot{q}_2)) \\ & -a_{13} \text{sen}(K_q(q_1 - q_3)) - b_{13} \text{sen}(K_{\dot{q}}(\dot{q}_1 - \dot{q}_3)) \\ & -K_1 \text{sen}(K_{d1} \dot{q}_1) + g_1(q_1), \end{aligned} \quad (5.36)$$

$$\begin{aligned}
\tau_2 = & -a_{21}\text{sen}(K_q(q_2 - q_1)) - b_{21}\text{sen}(K_{\dot{q}}(\dot{q}_2 - \dot{q}_1)) \\
& - a_{23}\text{sen}(K_q(q_2 - q_3)) - b_{23}\text{sen}(K_{\dot{q}}(\dot{q}_2 - \dot{q}_3)) \\
& - K_2\text{sen}(K_{d2}\dot{q}_2) + g_2(q_2),
\end{aligned} \tag{5.37}$$

$$\begin{aligned}
\tau_3 = & -a_{31}\text{sen}(K_q(q_3 - q_1)) - b_{31}\text{sen}(K_{\dot{q}}(\dot{q}_3 - \dot{q}_1)) \\
& - a_{32}\text{sen}(K_q(q_3 - q_2)) - b_{32}\text{sen}(K_{\dot{q}}(\dot{q}_3 - \dot{q}_2)) \\
& - K_3\text{sen}(K_{d3}\dot{q}_3) + g_3(q_3).
\end{aligned} \tag{5.38}$$

Los valores de las ganancias $K_q, K_{\dot{q}}, K_i$ y k_{di} se presentan en la Tabla 5.1.

Ganancia	Valor
K_{q_1}	1.25
K_{q_2}	1.25
K_{q_3}	1.25
$K_{\dot{q}_1}$	0.5
$K_{\dot{q}_2}$	0.5
$K_{\dot{q}_3}$	0.5
$k_{1,2,3}$	0.25
$k_{d1,2,3}$	0.025

Tabla 5.1: Ganancias elegidas para la implementación del algoritmo de control (5.35).

Con los valores de ganancias elegidos se obtienen las siguientes gráficas que muestran los resultados de la implementación experimental. En la Figura 5.6, se muestra la gráfica de los agentes en el espacio cartesiano indicando con un recuadro la posición inicial del efector final del robot manipulador 1, con un triángulo la posición inicial del robot manipulador 2 y con una estrella la posición inicial del robot 3, también se observa cuando estos se sobreponen indicando que han alcanzado una posición constante, consenso.

Por otro lado, la Figura 5.7 muestra las posiciones articulares $q_{i(1)}$ alcanzando el consenso en un valor igual a 0.785 radianes, la Figura 5.8, muestra las posiciones articulares $q_{i(2)}$ las cuales alcanzan el valor de consenso igual a 0.777 radianes y las posiciones articulares $q_{i(3)}$ en la Figura 5.9 alcanzan un valor de consenso igual a 0.7608 rad.

Además de alcanzar un valor constante que define el consenso, parte del objetivo es hacer que las velocidades articulares converjan a cero partiendo del reposo, como se muestra en la Figura 5.10 para $\dot{q}_{i(1)}$, en la Figura 5.11 para $\dot{q}_{i(2)}$ y en la Figura 5.12 para $\dot{q}_{i(3)}$. La energía que se subministra a los servomotores para que el sistema multiagente alcance el consenso se muestra en la Figura 5.13 para $\tau_{i(1)}$, en la Figura 5.14 para $\tau_{i(2)}$ y en la Figura 5.15 para $\tau_{i(3)}$.

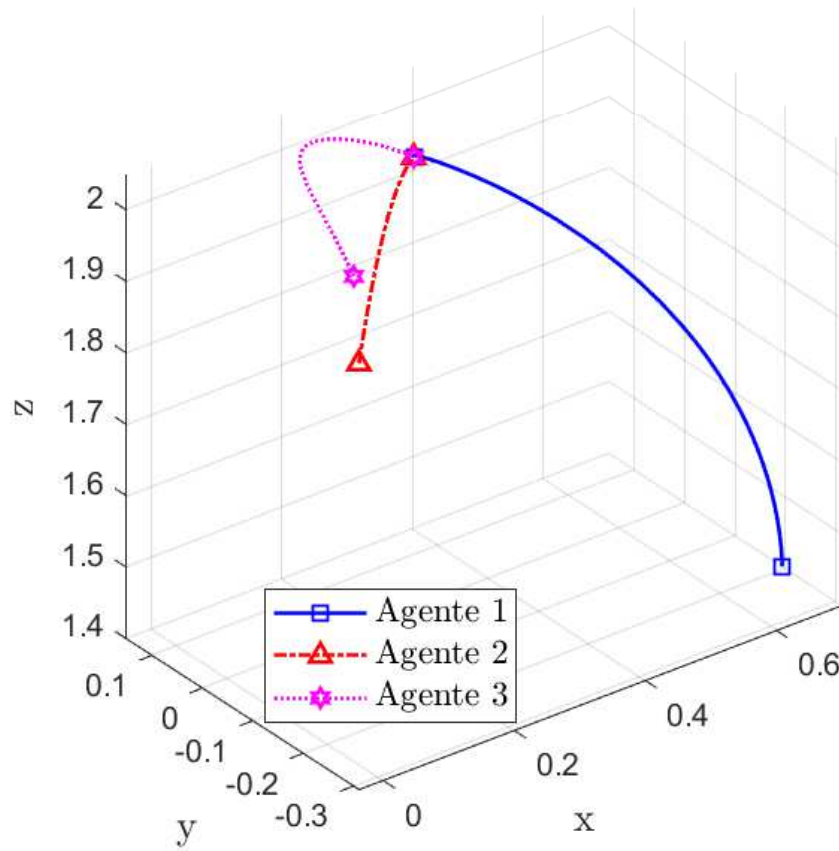


Figura 5.6: Consenso alcanzado por los tres robots manipuladores en el espacio cartesiano empleando el algoritmo de consenso (5.35).

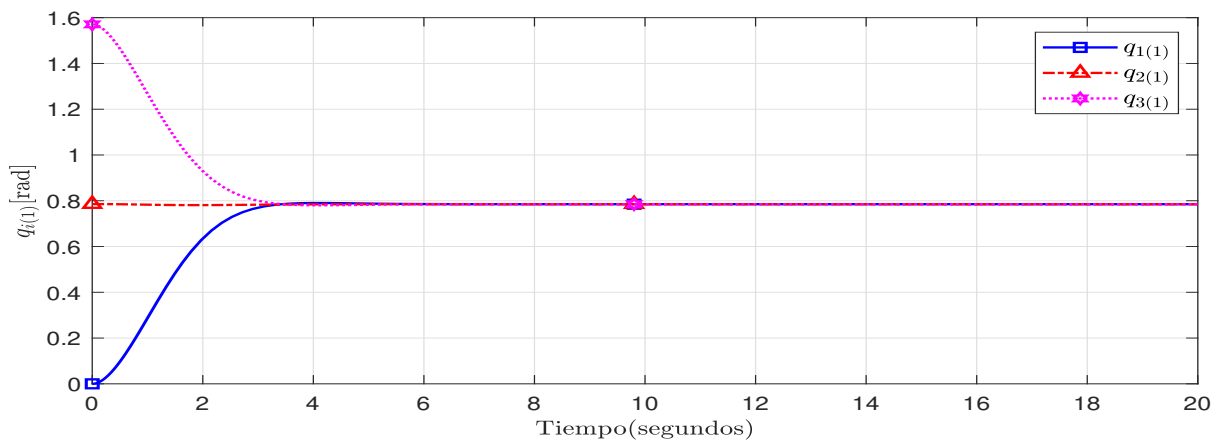


Figura 5.7: Consenso alcanzado para los eslabones $q_{i(1)}$ empleando el algoritmo (5.35).

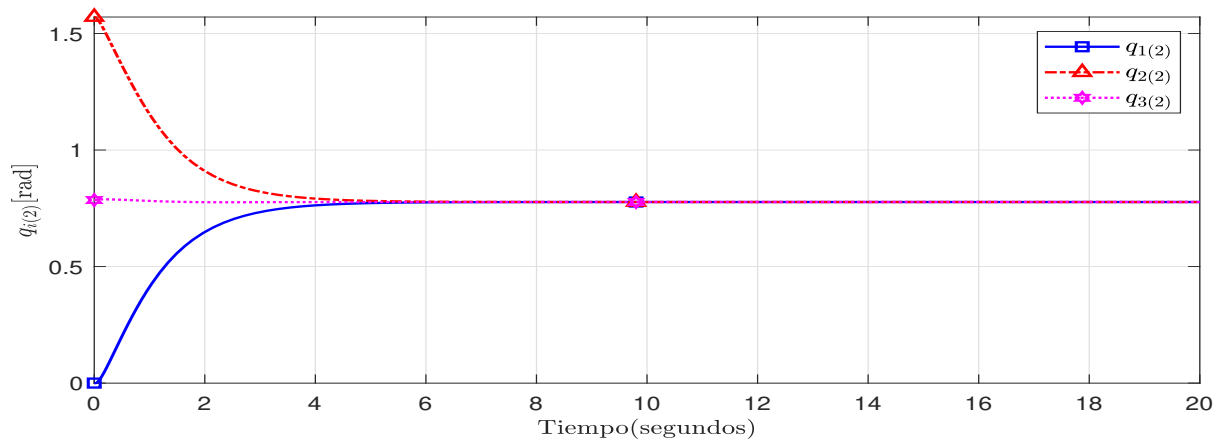


Figura 5.8: Consenso alcanzado para los eslabones $q_{i(2)}$ empleando el algoritmo (5.35).

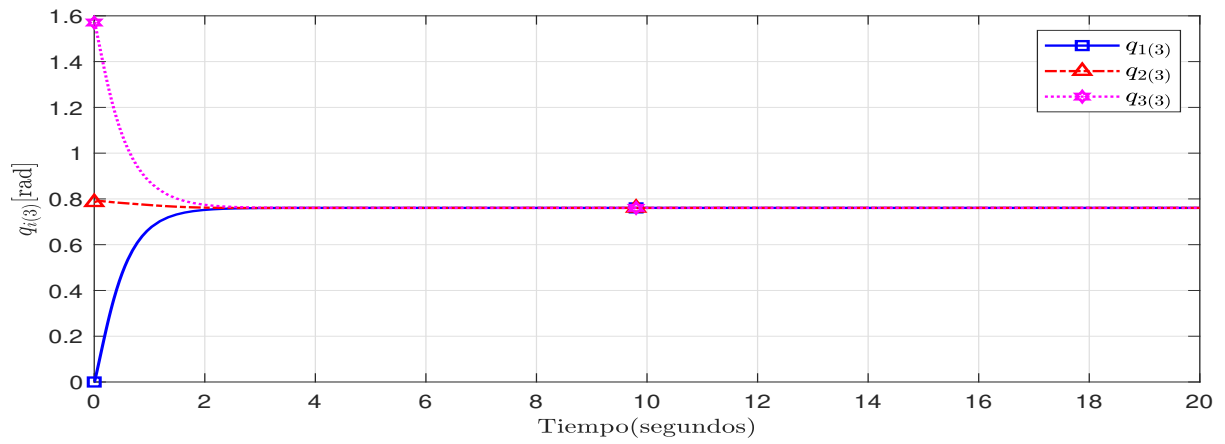


Figura 5.9: Consenso alcanzado para los eslabones $q_{i(3)}$ empleando el algoritmo (5.35).

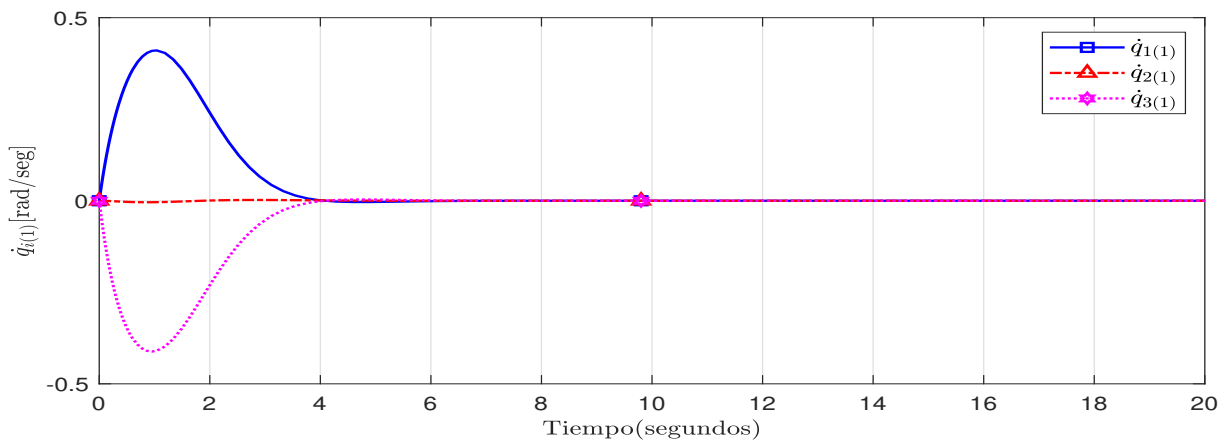


Figura 5.10: Velocidad articular $\dot{q}_{i(1)}$ empleando el algoritmo (5.35).

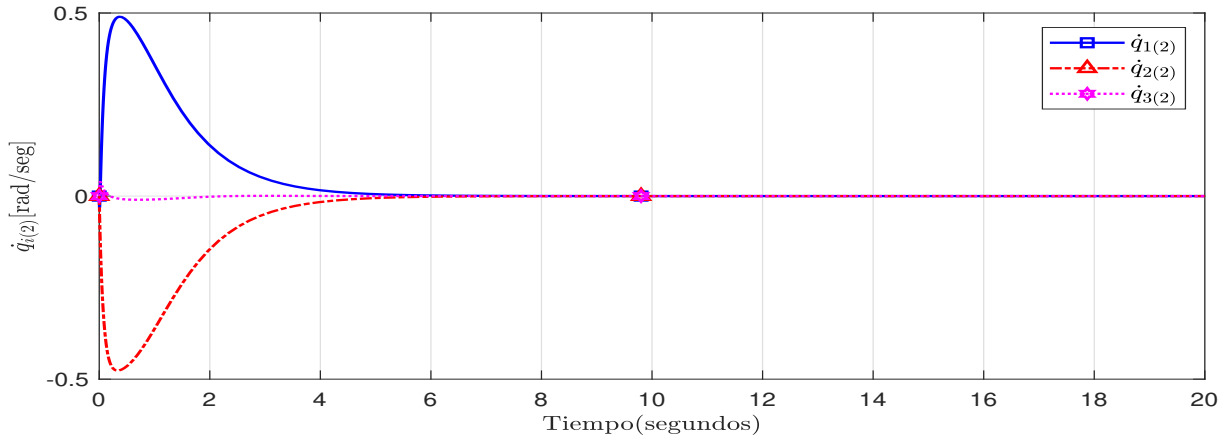


Figura 5.11: Velocidad articular $\dot{q}_{i(2)}$ empleando el algoritmo (5.35).

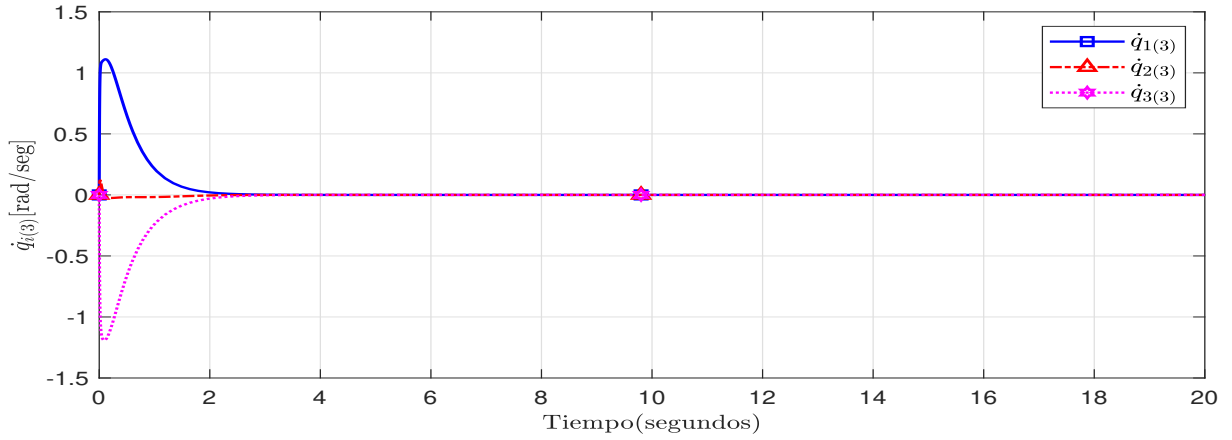


Figura 5.12: Velocidad articular $\dot{q}_{i(3)}$ empleando el algoritmo (5.35).

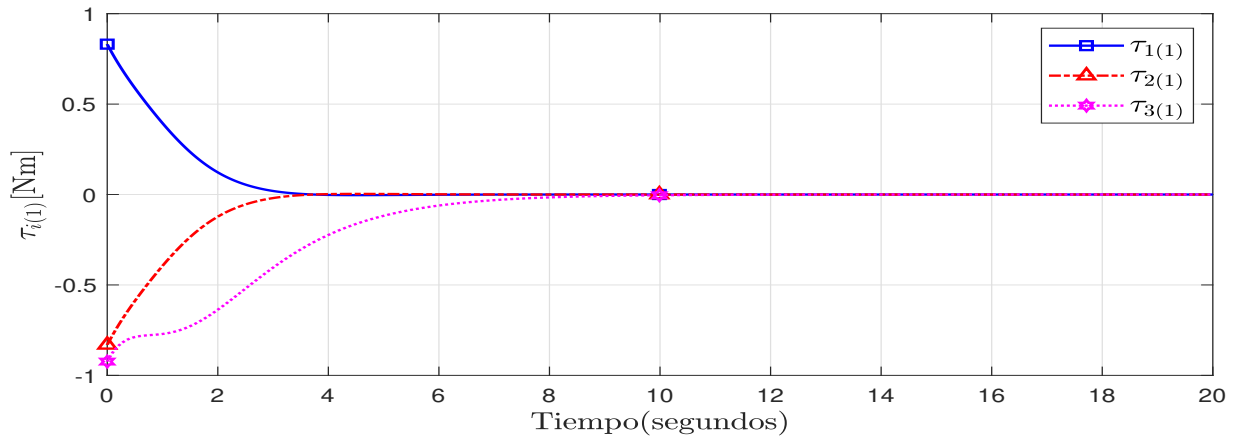


Figura 5.13: Torque $\tau_{i(1)}$ empleado con el algoritmo (5.35).

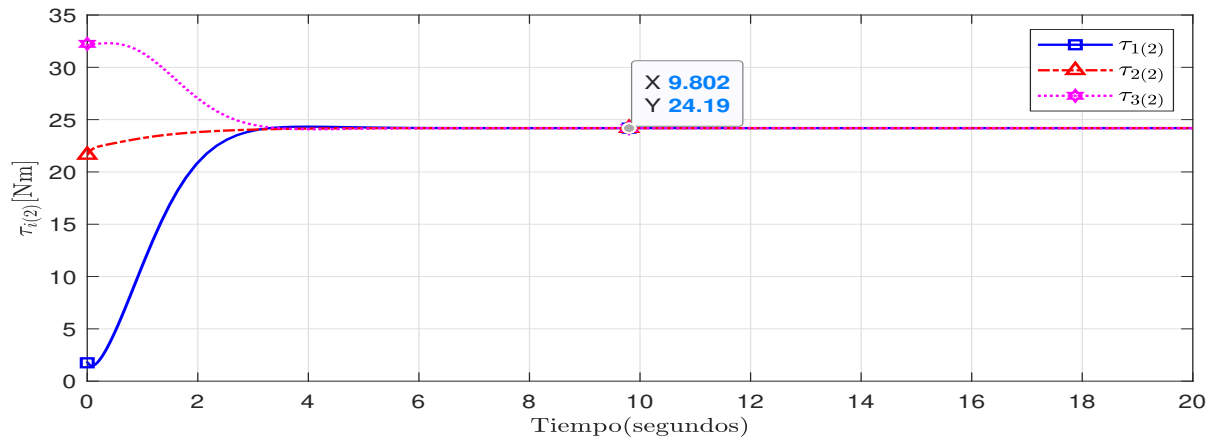


Figura 5.14: Torque $\tau_{i(2)}$ empleado con el algoritmo (5.35).

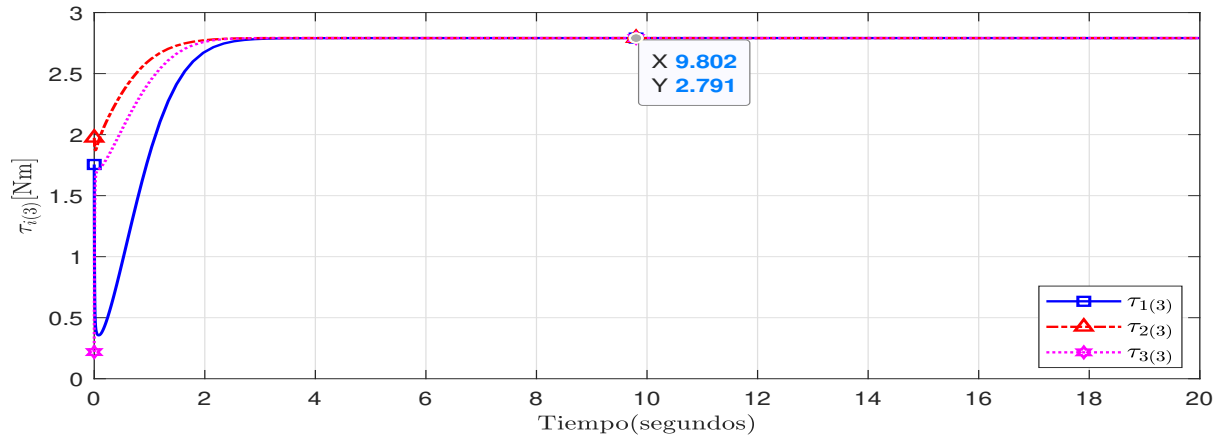


Figura 5.15: Torque $\tau_{i(3)}$ empleado con el algoritmo (5.35).

La implementación del algoritmo de consenso (5.35) considerando $\tanh(\cdot)$ tiene la siguiente forma

$$\tau_i = - \sum_{j=1}^n a_{ij} \tanh(K_q(q_i - q_j)) - \sum_{j=1}^n b_{ij} \tanh(K_{\dot{q}}(\dot{q}_i - \dot{q}_j)) - K_i \tanh(K_{d_i} \dot{q}_i) + g_i(q_i). \quad (5.39)$$

El algoritmo de consenso (5.39) para cada agente se expresa como

$$\begin{aligned} \tau_1 = & -a_{12} \tanh(K_q(q_1 - q_2)) - b_{12} \tanh(K_{\dot{q}}(\dot{q}_1 - \dot{q}_2)) \\ & - a_{13} \tanh(K_q(q_1 - q_3)) - b_{13} \tanh(K_{\dot{q}}(\dot{q}_1 - \dot{q}_3)) \\ & - K_1 \tanh(K_{d1} \dot{q}_1) + g_1(q_1), \end{aligned} \quad (5.40)$$

$$\begin{aligned} \tau_2 = & -a_{21} \tanh(K_q(q_2 - q_1)) - b_{21} \tanh(K_{\dot{q}}(\dot{q}_2 - \dot{q}_1)) \\ & - a_{23} \tanh(K_q(q_2 - q_3)) - b_{23} \tanh(K_{\dot{q}}(\dot{q}_2 - \dot{q}_3)) \\ & - K_2 \tanh(K_{d2} \dot{q}_2) + g_2(q_2), \end{aligned} \quad (5.41)$$

$$\begin{aligned} \tau_3 = & -a_{31} \tanh(K_q(q_3 - q_1)) - b_{31} \tanh(K_{\dot{q}}(\dot{q}_3 - \dot{q}_1)) \\ & - a_{32} \tanh(K_q(q_3 - q_2)) - b_{32} \tanh(K_{\dot{q}}(\dot{q}_3 - \dot{q}_2)) \\ & - K_3 \tanh(K_{d3} \dot{q}_3) + g_3(q_3). \end{aligned} \quad (5.42)$$

Para propósitos de comparación de desempeño se emplean los valores de las ganancias de la Tabla para el algoritmo de consenso (5.39).

Ganancia	Valor
K_{q_1}	1.25
K_{q_2}	1.25
K_{q_3}	1.25
$K_{\dot{q}_1}$	0.5
$K_{\dot{q}_2}$	0.5
$K_{\dot{q}_3}$	0.5
$k_{1,2,3}$	0.25
$k_{d1,2,3}$	0.025

Tabla 5.2: Ganancias elegidas para la implementación del algoritmo de control (5.39).

Con valores anteriores se obtienen las siguientes gráficas que muestran los resultados de la implementación experimental. Como en el caso anterior la Figura 5.16, se muestra la gráfica de los agentes en el espacio cartesiano, donde se indica con un recuadro la posición inicial del efector final del robot manipulador 1, con un triángulo la posición

inicial del robot manipulador 2 y con una estrella la posición inicial del robot 3, también se observa cuando estos se sobreponen indicando que han alcanzado una posición constante, consenso.

Por otro lado, la Figura 5.17 muestra las posiciones articulares $q_{i(1)}$ alcanzando el consenso en un valor igual a 0.7851 radianes, la Figura 5.18, muestra las posiciones articulares $q_{i(2)}$ las cuales alcanzan el valor de consenso en un valor igual a 0.7775 radianes, en la Figura 5.19 $q_{i(3)}$ el consenso es alcanzado con un valor igual a 0.7622 radianes.

Además de alcanzar un valor constante que define el consenso parte del objetivo es hacer que las velocidades articulares converjan a cero partiendo del reposo, como se muestra en la Figura 5.20 para $\dot{q}_{i(1)}$, en la Figura 5.21 para $\dot{q}_{i(2)}$, en la Figura 5.22 para $\dot{q}_{i(3)}$. La energía que se subministra a los servomotores para que el sistema multiagente alcance el consenso se muestra en la Figura 5.23 para $\tau_{i(1)}$ la energía necesaria para el sistema multiagente es de 0 Nm, en la Figura 5.24 para $\tau_{i(2)}$ la energía necesaria es de 24.19 Nm y en la Figura 5.25 para $\tau_{i(3)}$ la energía necesaria es de 2.791 Nm.

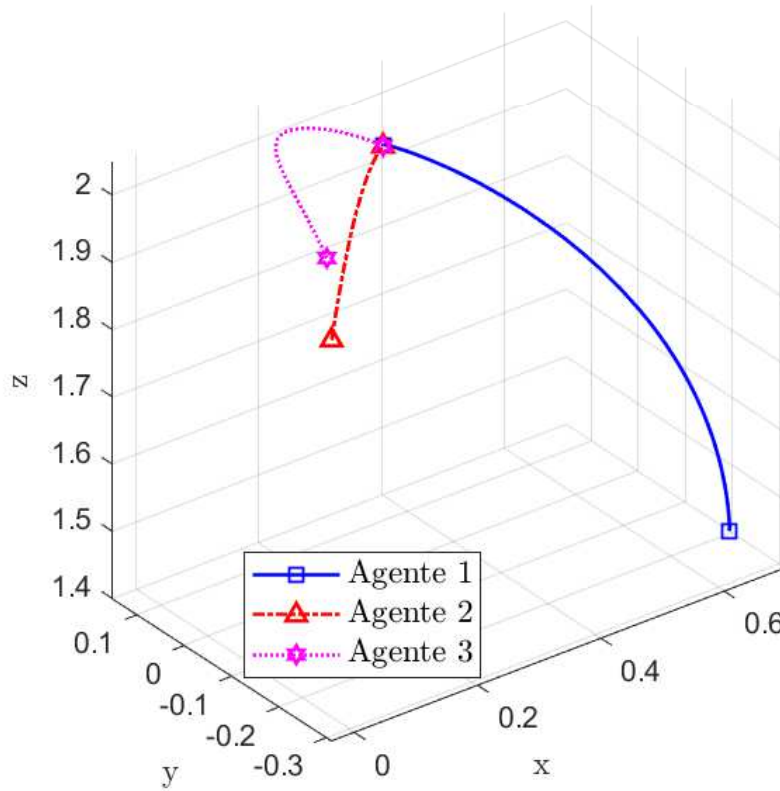


Figura 5.16: Consenso alcanzado por los tres robots manipuladores en el espacio cartesiano empleando el algoritmo de consenso (5.39).

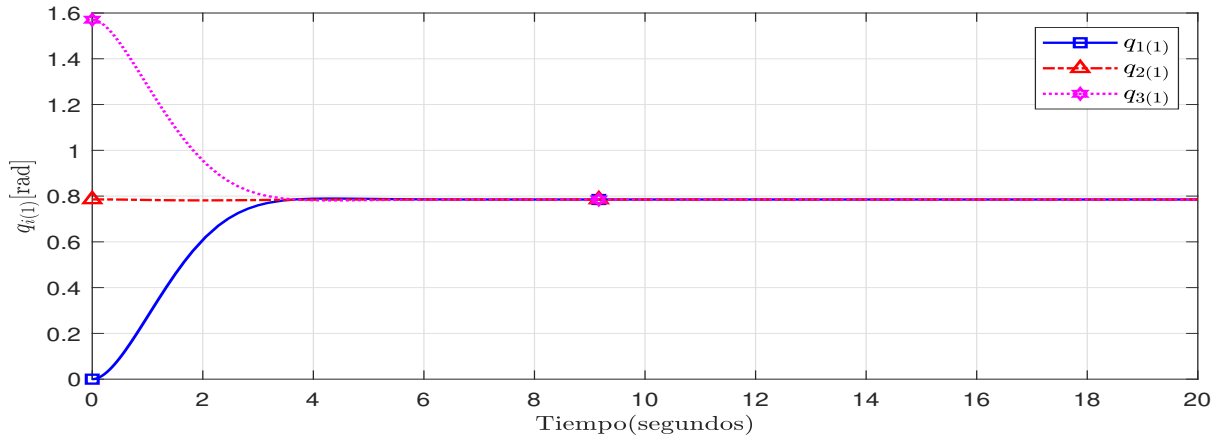


Figura 5.17: Consenso alcanzado para los eslabones $q_{i(1)}$ empleando el algoritmo (5.39).

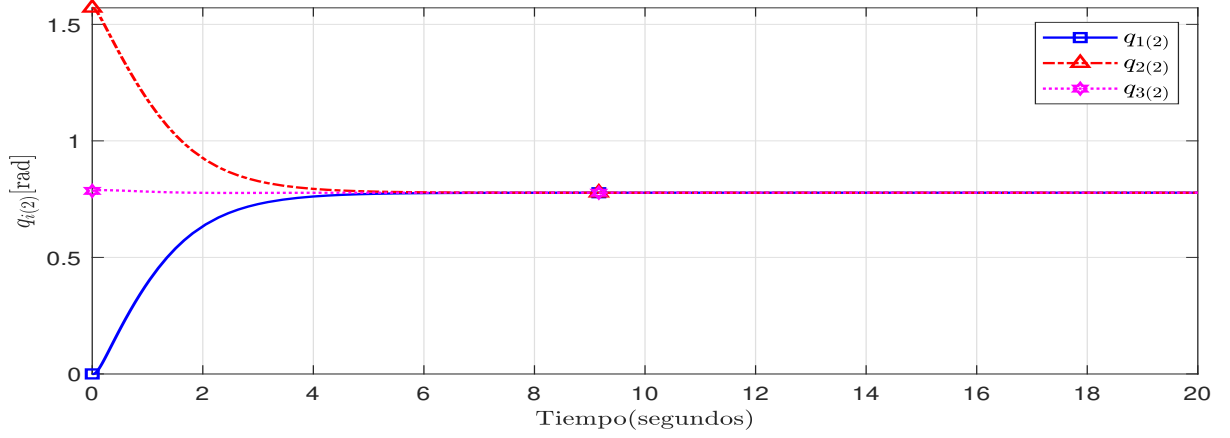


Figura 5.18: Consenso alcanzado para los eslabones $q_{i(2)}$ empleando el algoritmo (5.39).

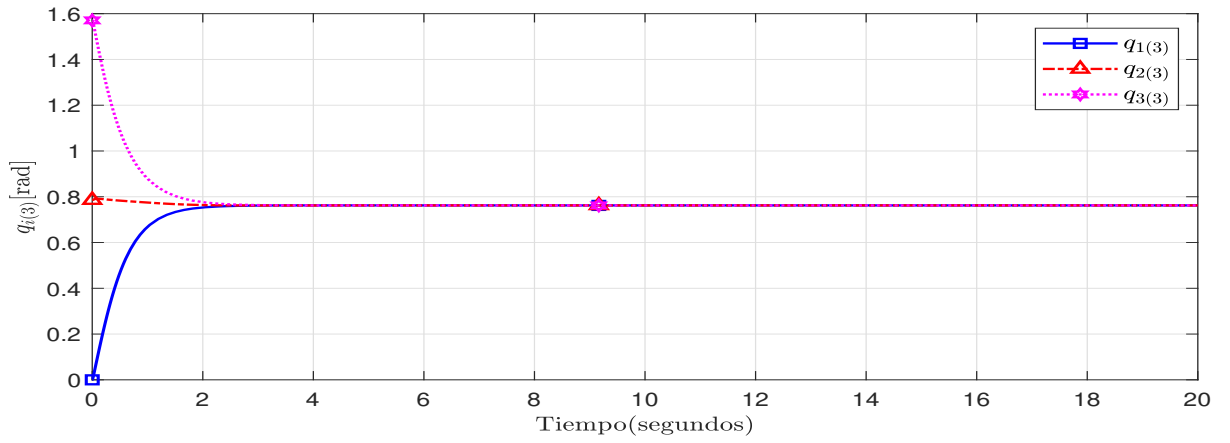


Figura 5.19: Consenso alcanzado para los eslabones $q_{i(3)}$ empleando el algoritmo (5.39).

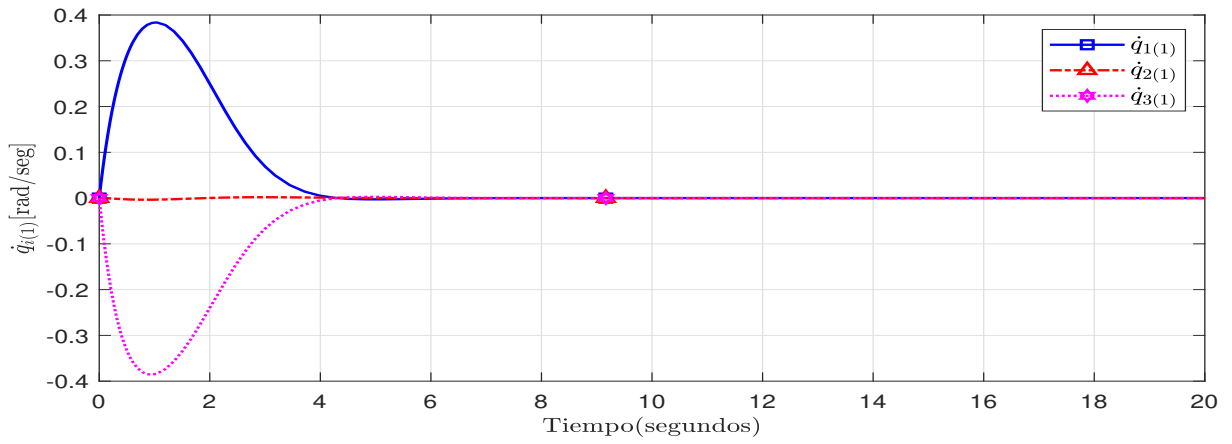


Figura 5.20: Velocidad articular de los eslabones $\dot{q}_{i(1)}$ empleando el algoritmo (5.39).

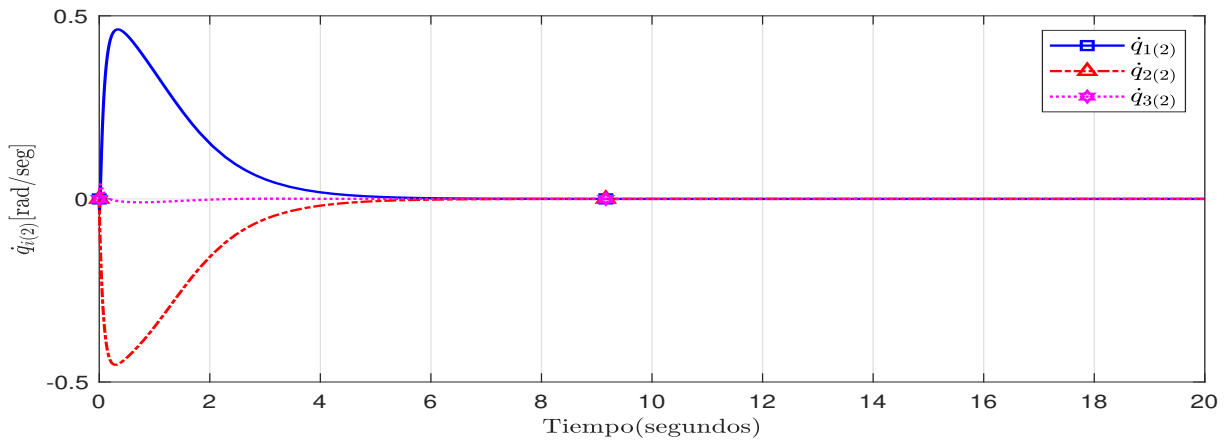


Figura 5.21: Velocidad articular de los eslabones $\dot{q}_{i(2)}$ empleando el algoritmo (5.39).

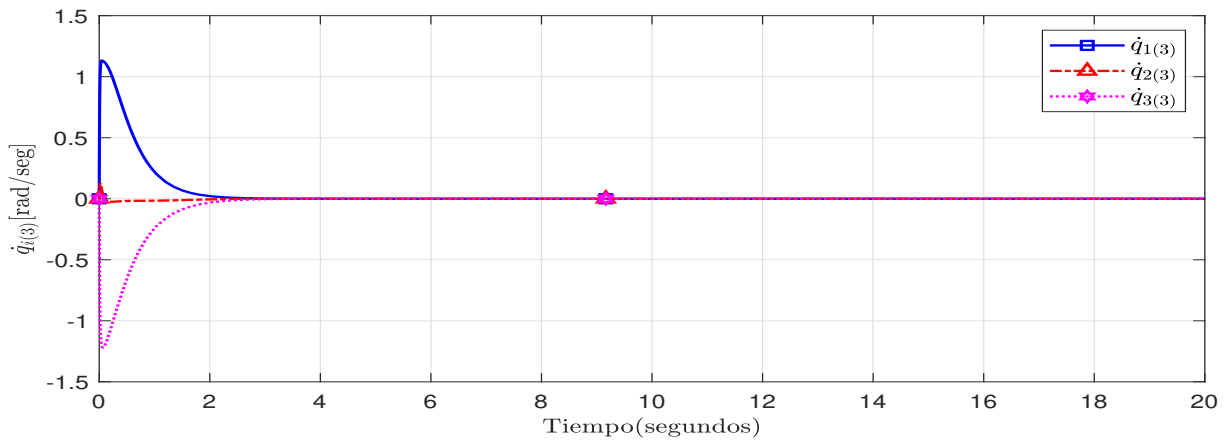


Figura 5.22: Velocidad articular de los eslabones $\dot{q}_{i(3)}$ empleando el algoritmo (5.39).

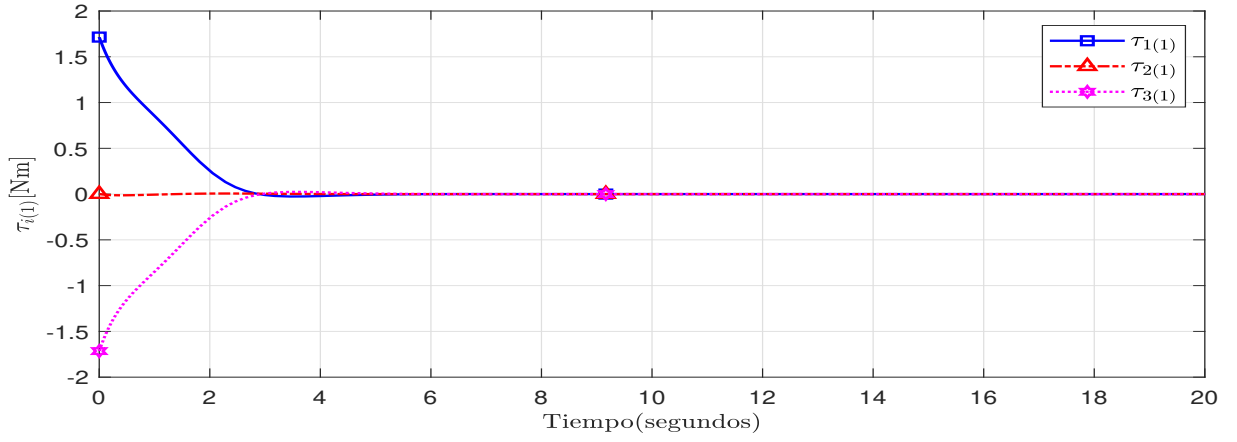


Figura 5.23: Energía suministrada para $\tau_{i(1)}$ empleando el algoritmo (5.39).

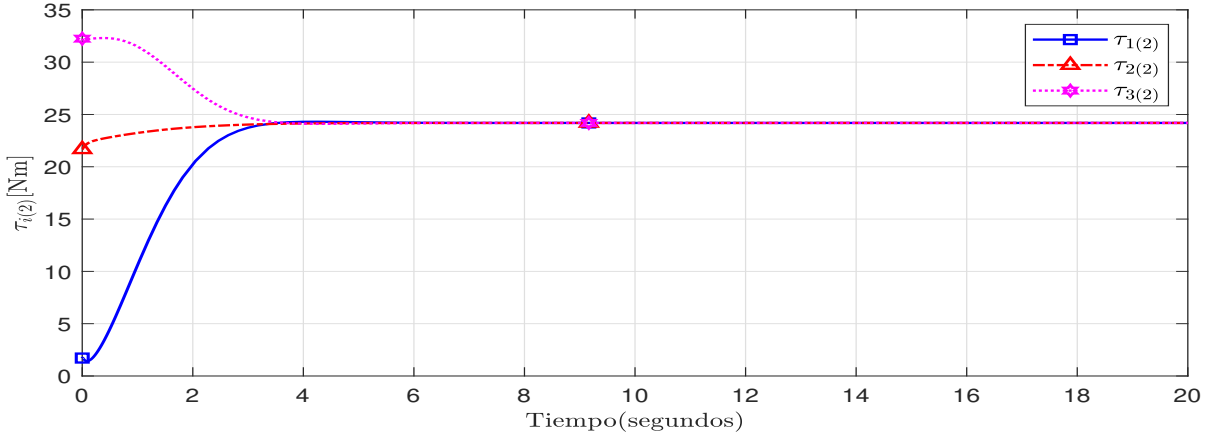


Figura 5.24: Energía suministrada para $\tau_{i(2)}$ empleando el algoritmo (5.39).

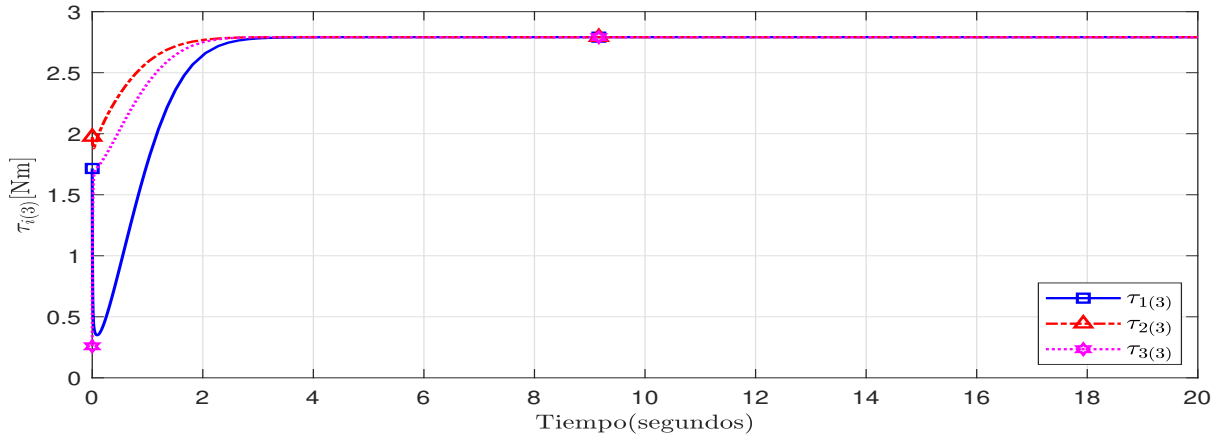


Figura 5.25: Energía suministrada para $\tau_{i(3)}$ empleando el algoritmo (5.39).

Capítulo 6

Conclusiones

En este trabajo se realizó la implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores de transmisión directa de tres grados de libertad, mediante el método de Hardware in the Loop. La implementación del sistema hardware in the loop es pertinente, debido a que se pudo mostrar que tanto el robot físico que está en el laboratorio de robótica y control de la FCE-BUAP y el robot que se propone con el método Hardware In the Loop comparten un comportamiento similar, empleando el mismo algoritmo de control, las mismas condiciones iniciales y ganancias propuestas.

Para mostrar la efectividad de la instrumentación propuesta se empleó el algoritmo de control de trayectoria backstepping, el cual por su estructura matemática supone un alto grado de complejidad ya que este involucra la dinámica completa del robot manipulador más compensación de gravedad, por lo tanto, el microcontrolador debe tener la capacidad de poder calcular una acción de control ante una señal de referencia dada y más aún si esta es variable en el tiempo.

Por lo anterior se deja una planta física experimental completamente funcional la cual permite poder evaluar diferentes algoritmos de control, así como diferentes aplicaciones, con instrumentación de bajo costo y de fácil acceso. De la misma forma es posible mediante el método Hardware in the Loop.

Los algoritmos de consenso que son caso de estudio en este trabajo y que fueron implementados de forma exitosa en una red de robots manipuladores, nos permiten ampliar hacia un panorama donde es posible evaluar experimentalmente la respuesta de múltiples robots, así como, la evaluación del desempeño de algoritmos de control de manera simultánea, y más aún hacia el trabajo colaborativo donde la red de robots es capaz de resolver tareas complejas como lo es el caso de seguimiento de trayectoria y sincronización entre otras.

Como trabajo futuro en primer lugar se propone validar mediante algún método de identificación paramétrica los valores dados para los componentes de la matriz de inercia de Coriolis y fuerzas centrípetas ya que en este trabajo se abordó mediante el método de mediciones directas, siendo este un tema complejo que incluso ha sido objeto de estudio en otros trabajos de tesis para otras plantas experimentales similares. También se propone sincronizar al robot de transmisión directa físico con los robots implementados en Hardware in the Loop para así tener grupo de robots más extenso de robots manipuladores.

Apéndice A

Códigos realizados, Capitulo 3

En este capítulo se encuentran los códigos realizados para la implementación del sistema Hardware In the Loop. Primero se muestra la implementación del algoritmo de control Backstepping en la tarjeta Arduino DUE y la recepción de posiciones y velocidades articulares del robot manipulador de transmisión directa virtual así como la señal de control que se envía. Posteriormente se muestra el código de la función de Simulink para la implementación del robot virtual.

A.1. Unión implementada en Arduino DUE

En el siguiente recuadro se encuentra el código realizado para la declaración de una unión la cual se emplea para el envío de señal de control y recepción de posición y velocidad articular.

“Implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores”

capítulo 3: Instrumentación.

Sección: Hardware In the Loop.

Script: Arduino union.

```
1 //Declaración de una unión
2 typedef union {
3   float floatV;
```

```

4  byte  binaryV[4];
5  }BinaryFloat;

```

A.2. Control de trayectoria: Implementación en Arduino DUE

En el recuadro se muestra la implementación del algoritmo de control de trayectoria Backstepping para el control del robot virtual.

“Implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores”

capítulo 3: Instrumentación.

Sección: Hardware In the Loop.

Script: Arduino TrackingHIL.

```

1 #include "HIL.h"
2 #define pi 3.141592
3
4 BinaryFloat q1,q2,qp1,qp2,t,tau1,tau2,qd1,qd2;
5
6 int a1 = 0, a2 = 0;
7 float m1=6.94, l1=0.35, lc1=0.057;
8 float m2=1.19, l2=0.35, lc2=0.041, g=9.81;
9 float qt1=0,qt2=0,Kp1=0,Kp2=0,Kv1=0,Kv2=0,Lambda1=0,Lambda2=0;
10 float xd = 0, yd = 0, xdp = 0, ydp = 0, xdpp = 0, ydpp = 0;
11 float Jinv11 = 0, Jinv12 = 0, Jinv21 = 0, Jinv22 = 0, qpd1 = 0, qpd2 = 0;
12 float Jp11 = 0, Jp12 = 0, Jp21 = 0, Jp22 = 0, qppd1 = 0, qppd2 = 0;
13 float qpt1 = 0, qpt2 = 0, tant = 0, qpf1=0, qpf2=0;
14
15 void setup() {
16   Serial.begin(115200);
17   Serial1.begin(115200);}
18 void loop() {
19   if (Serial1.available() > 0){
20     a1 = Serial1.read();
21     if (a1 == 0x7E) {
22       a2 = Serial1.read();
23       if (a2 == 0x7E) {
24         q1.binaryV[0] = Serial1.read();
25         q1.binaryV[1] = Serial1.read();
26         q1.binaryV[2] = Serial1.read();
27         q1.binaryV[3] = Serial1.read();
28         qp1.binaryV[0] = Serial1.read();
29         qp1.binaryV[1] = Serial1.read();
30         qp1.binaryV[2] = Serial1.read();

```



```

31     qp1.binaryV[3] = Serial1.read();
32     q2.binaryV[0] = Serial1.read();
33     q2.binaryV[1] = Serial1.read();
34     q2.binaryV[2] = Serial1.read();
35     q2.binaryV[3] = Serial1.read();
36     qp2.binaryV[0] = Serial1.read();
37     qp2.binaryV[1] = Serial1.read();
38     qp2.binaryV[2] = Serial1.read();
39     qp2.binaryV[3] = Serial1.read();
40     t.binaryV[0] = Serial1.read();
41     t.binaryV[1] = Serial1.read();
42     t.binaryV[2] = Serial1.read();
43     t.binaryV[3] = Serial1.read();
44     a1 = Serial1.read();
45     a2 = Serial1.read();
46     if (a1 == 0x0D && a2 == 0x0A) {
47         // Trayectoria Deseada Cartesiana
48         xd = 0.3 + 0.2 * sin(0.5*t.floatV);
49         yd = -0.3 + 0.2 * cos(0.5*t.floatV);
50         // primera derivada
51         xdp = 0.01 * cos(0.5*t.floatV);
52         ydp = -0.01 * sin(0.5*t.floatV);
53         // segunda derivada
54         xdpp = -0.05 * sin(0.5*t.floatV);
55         ydpp = -0.05 * cos(0.5*t.floatV);
56
57         // Trayectoria deseada articular
58         qd2.floatV = acos(((xd * xd) + (yd * yd) - (0.35 * 0.35) - (0.35 *
59             0.35)) / (2 * 0.35 * 0.35));
60         qd1.floatV = pi/2 + atan(yd / xd) - atan((0.35 * sin(qd2.floatV)) /
61             (0.35 + 0.35 * cos(qd2.floatV)));
62
63         // Cineatica Inversa
64         // Velocidad articular deseada
65
66         // Derivada Jacobiano
67         Jp11 = -l1*sin(qd1.floatV)*qpd1-l2*sin(qd1.floatV+qd2.floatV)*(qpd1+
68             qpd2);
69         Jp12 = -l2*sin(qd1.floatV+qd2.floatV)*(qpd1+qpd2);
70         Jp21 = l1*cos(qd1.floatV)*qpd1+l2*cos(qd1.floatV+qd2.floatV)*(qpd1+qpd2
71             );
72         Jp22 = l2*cos(qd1.floatV+qd2.floatV)*(qpd1+qpd2);
73
74         // Aceleración Deseada
75         qppd1 = (- Jinv11*(Jp11*(Jinv11*xdp + Jinv12*ydp) - xdpp + Jp12*(Jinv21
76             *xdp + Jinv22*ydp)) - Jinv12*(Jp21*(Jinv11*xdp + Jinv12*ydp) - ydpp
77             + Jp22*(Jinv21*xdp + Jinv22*ydp)));
78         qppd2 = (- Jinv21*(Jp11*(Jinv11*xdp + Jinv12*ydp) - xdpp + Jp12*(Jinv21
79             *xdp + Jinv22*ydp)) - Jinv22*(Jp21*(Jinv11*xdp + Jinv12*ydp) - ydpp
80             + Jp22*(Jinv21*xdp + Jinv22*ydp)));
81
82         // Errores
83         qt1 = qd1.floatV-q1.floatV;
84         qt2 = qd2.floatV-q2.floatV;
85         qpt1 = qpd1-qp1.floatV;
86         qpt2 = qpd2-qp2.floatV;
87
88     }
89 }

```

```

80     qpr1 = - qpd1 + Lambda1*qt1;
81     qpr2 = - qpd2 + Lambda2*qt2;
82     qppr1 = - qppd1 + Lambda1*qpt1;
83     qppr2 = - qppd2 + Lambda2*qpt2;
84     s1 = qpt1 + Lambda1*qt1;
85     s2 = qpt2 + Lambda2*qt2;
86
87     //Ganancias
88     Kp1 = 128; Kp2 = 28;
89     Kv1 = 0.5*Kp1; Kv2 = 0.5*Kp2;
90     Lambda1 = 1.5; Lambda2 = 2;
91
92     //Algoritmo de control Backstepping
93     tau1.floatV = - Kp1*qt1 - Kv1*s1 + (0.03444*cos(q2.floatV)+1.53056526)
        *qppr1 + (0.01722*cos(q2.floatV)+0.0950172)*qppr2 - 0.03444*qp2*
        sin(q2.floatV)*qp2.floatV*qpr1 - 0.01722*qp2.floatV*sin(q2.floatV)
        *qp2.floatV*qpr2 + 0.057 * 6.94 * 9.81 * sin(q1.floatV) + 1.18 *
        0.35 * 9.81 * sin(q1.floatV) + (0.041 * 1.18 * 9.81 * sin(q1.
        floatV + q2.floatV));
94     tau2.floatV = - Kp2*qt2 - Kv2*s2 + (0.01722*cos(q2.floatV)+0.0950172)
        *qppr1 + 0.0950172*qppr2 + 0.01722*qp1.floatV*sin(q2.floatV)*qp1.
        floatV*qpr1+ (0.041 * 1.18 * 9.81 * sin(q1.floatV + q2.floatV));
95
96     if (tau1.floatV >= 40.0) {tau1.floatV = 40.0;}
97     if (tau1.floatV <= -40.0) {tau1.floatV = -40.0;}
98     if (tau2.floatV >= 3.7) {tau2.floatV = 3.7;}
99     if (tau2.floatV <= -3.7) {tau2.floatV = -3.7;}
100     Serial.write(0x7E);
101     Serial.write(tau1.binaryV,4);
102     Serial.write(tau2.binaryV,4);
103     Serial.write(qd1.binaryV,4);
104     Serial.write(qd2.binaryV,4);
105     Serial.write(0x0D);
106 }
107 }
108 }
109 }
110 }

```

A.3. Control de trayectoria: Modelo dinámico Robot de transmisión directa HIL

En el recuadro se muestra la función de matlab para la implemmentación del robot de transmisión directa virtual para el sistema hardware In the Loop como se muestra en la Figura 4.19.

“Implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores”

capítulo 3: Instrumentación.

Sección: Hardware In the Loop.

Función Matlab: Robot2gdl.

```

1 function qpp = robot2gdl(q1,q2,qp1,qp2,tau1,tau2)
2
3 %vector de posición articular
4 q = [q1;q2];
5 %vector de velocidad articular
6 qp = [qp1;qp2];
7 tau = [tau1;tau2];
8
9 m1=6.94; l1=0.35; lc1=0.057; I1=1.422; b1=2.155;
10 m2=1.18; l2=0.35; lc2=0.041; g=9.81; I2=0.097; b2=0.215;
11 M11=m1*lc1*lc1+m2*l1*l1+m2*lc2*lc2+I1+I2+2*m2*l1*lc2*cos(q2);
12 M12=m2*lc2*lc2+m2*l1*lc2*cos(q2)+I2;
13 M21=m2*lc2*lc2+m2*l1*lc2*cos(q2)+I2;
14 M22=m2*lc2*lc2+I2;
15 C11=-m2*l1*lc2*sin(q2)*qp2;
16 C12=-m2*l1*lc2*sin(q2)*(qp1+qp2);
17 C21=m2*l1*lc2*sin(q2)*qp1;
18 g1=lc1*m1*sin(q1)+m2*l1*sin(q1)+m2*lc2*sin(q1+q2);
19 g2=lc2*m2*sin(q1+q2);
20
21 %matriz de inercia
22 M=[M11 M12;M21 M22];
23 %matriz de coriolis
24 C = [C11, C12; C21, 0];
25 %vector de pares gravitacionales
26 gq = g*[g1; g2];
27
28 qpp=M^(-1)*(tau-C*qp-gq);

```

Apéndice B

Códigos realizados, capítulo 5

En esta sección se presentan los programas implementados en el bloque *function* de simulink, para el robot manipulador 1 B.1.1, para el robot manipulador B.1.2 y para el robot manipulador B.1.3. También, se presentan los códigos empleados en Arduino para la implementación de los algoritmos de consenso para cada robot indicando sus correspondientes posiciones articulares q_1 , q_2 y q_3 . En los siguientes recuadros se muestra la implementación de los algoritmos de consenso (5.35) en el sistema arduino. En el ide de arduino primero se declaran variables, en void setup() se inicializan los puertos seriales con el comando *Serial.begin()* y *Serial1.begin()* a 115200 baudios, en void loop se realiza la obtención de las posiciones y velocidades articulares de cada agente posteriormente se implementa el algoritmo de consenso y la señal de control se envía a simulink a través de la instrucción *Serial.write()*.

B.1. Implementación en simulink

B.1.1. Script: Robot de transmisión directa 1

“Implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores”

capítulo 5: Resultados.

Sección: Implementación de algoritmos de consenso.

Script: robot3gdl1.

```
1 function qpp = robot3gdl1 (tau1 ,tau2 ,tau3 ,q1 ,q2 ,q3 ,qp1 ,qp2 ,qp3)
```

```

2  %vector de posición articular
3  q = [q1;q2;q3];
4  %vector de velocidad articular
5  qp = [qp1;qp2;qp3];
6  %vector de pares
7  tau=[tau1;tau2;tau3];
8
9  Iz1=1.26; Iz2=0.084; Iz3=0.056; Iy1=0.089; Iy2=0.003; Iy3=0.0012;
    Ix1=0.03; Ix2=0.05; Ix3=0.009;    m1=11.523; l1=1.075; b1
    =2.288;
10 fc1=7.17; fe1=8.8; m2=6.94; l2=0.35; lc2=0.057; b2=2.155;
11 fc2=1.9; fe2=2.1; m3=1.18; l3=0.35; lc3=0.041; b3=0.215; fc3
    =1.734; fe3=1.87; %B.29
12
13 m11=Iy2*sin(q2)*sin(q2)+Iy3*sin(q2+q3)*sin(q2+q3)+Iz1+Iz2*cos(q2)*
    cos(q2)+Iz3*cos(q2+q3)*cos(q2+q3)+m2*lc2*lc2*cos(q2)*cos(q2)+
    m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*cos(q2)+lc3*cos(q2+q3));
14 m12=0; m13=0; m21=0; m22=Ix2+Ix3+m3*l2*l2+m2*lc2*lc2+m3*lc3*lc3+
    2*m3*l2*lc3*cos(q3);
15 m23=Ix3+m3*lc3*lc3+m3*l2*lc3*cos(q3);
16 m31=0; m32=Ix3+m3*lc3*lc3+m3*l2*lc3*cos(q3); m33=Ix3+m3*lc3*lc3;
17
18 gammal12=(Iy2-Ix2-m2*lc2*lc2)*cos(q2)*sin(q2)+(Iy3-Iz3)*cos(q2+q3)
    *sin(q2+q3)-m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*sin
    (q2+q3));
19 gammal13=(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)-m3*lc3*sin(q2+q3)*(l2*cos
    (q2)+lc3*cos(q2+q3));
20 gammal21=(Iy2-Iz2-m2*lc2*lc2)*cos(q2)*sin(q2)+(Iy3-Iz3)*cos(q2+q3)
    *sin(q2+q3)-m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*sin
    (q2+q3));
21 gamma131=(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)-m3*lc3*sin(q2+q3)*(l2*cos
    (q2)+lc3*cos(q2+q3));
22 gamma211=(Ix2-Iy2+m2*lc2*lc2)*cos(q2)*sin(q2)+ (Iz3-Iy3)*cos(q2+q3)
    *sin(q2+q3)+m3*(l2*cos(q2)+lc3*cos(q2+q3))* (l2*sin(q2)+lc3*
    sin(q2+q3));
23 gamma223=-l2*m3*lc3*sin(q3);
24 gamma232=-l2*m3*lc3*sin(q3);
25 gamma233=-l2*m3*lc3*sin(q3);
26 gamma311=(Iz3-Iy3)*cos(q2+q3)*sin(q2+q3)+m3*lc3*sin(q2+q3)*(l2*cos
    (q2)+lc3*cos(q2+q3));
27 gamma322=l2*m3*lc3*sin(q3);
28
29 c11=gammal12*qp2+gammal13*qp3; c12=gammal21*qp1; c13=gammal31*qp1;
30 c21=gamma211*qp1; c22=gamma223*qp3; c23=gamma232*qp2+gamma233*qp3;
31 c31=gamma311*qp1; c32=gamma322*qp2;

```

```

32 c33=0;
33 %matriz de inercia
34 M=[m11, m12, m13; m21, m22, m23; m31, m32, m33];
35 %matriz de fuerzas centripetas y de coriolis
36 C=[c11, c12, c13; c21, c22, c23; c31, c32, c33];
37 %Vector de pares gravitacionales
38 gq11=0;
39 gq21=0.8086*sin(q1)+0.0484*sin(q1+q2);
40 gq31=0.0484*sin(q1+q2);
41 g=9.81;
42 gq=g*[gq11; gq21; gq31];
43 fr=[b1*qp1; b2*qp2; b3*qp3];
44
45 qpp = M^(-1)*(tau-C*qp-gq-fr); %aceleración articular
46 end

```

B.1.2. Script: Robot de transmisión directa 2

“Implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores”

capítulo 5: Resultados.

Sección: Implementación de algoritmos de consenso.

Script: robot3gdl2.

```

1 function qpp = robot3gdl2(tau1,tau2,tau3,q1,q2,q3,qp1,qp2,qp3)
2 %vector de posición articular
3 q = [q1;q2;q3];
4 %vector de velocidad articular
5 qp = [qp1;qp2;qp3];
6 %vector de pares
7 tau=[tau1;tau2;tau3];
8
9 Iz1=1.26; Iz2=0.084; Iz3=0.056; Iy1=0.089; Iy2=0.003; Iy3=0.0012;
   Ix1=0.03; Ix2=0.05; Ix3=0.009; m1=11.523; l1=1.075; b1
   =2.288;
10 fc1=7.17; fe1=8.8; m2=6.94; l2=0.35; lc2=0.057; b2=2.155;
11 fc2=1.9; fe2=2.1; m3=1.18; l3=0.35; lc3=0.041; b3=0.215; fc3
   =1.734; fe3=1.87; %B.29
12

```

```

13 m11=Iy2*sin(q2)*sin(q2)+Iy3*sin(q2+q3)*sin(q2+q3)+Iz1+Iz2*cos(q2)*
    cos(q2)+Iz3*cos(q2+q3)*cos(q2+q3)+m2*lc2*lc2*cos(q2)*cos(q2)+
    m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*cos(q2)+lc3*cos(q2+q3));
14 m12=0; m13=0; m21=0; m22=Ix2+Ix3+m3*l2*l2+m2*lc2*lc2+m3*lc3*lc3+
    2*m3*l2*lc3*cos(q3);
15 m23=Ix3+m3*lc3*lc3+m3*l2*lc3*cos(q3);
16 m31=0; m32=Ix3+m3*lc3*lc3+m3*l2*lc3*cos(q3); m33=Ix3+m3*lc3*lc3;
17
18 gammal12=(Iy2-Ix2-m2*lc2*lc2)*cos(q2)*sin(q2)+(Iy3-Iz3)*cos(q2+q3)
    *sin(q2+q3)-m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*sin
    (q2+q3));
19 gammal13=(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)-m3*lc3*sin(q2+q3)*(l2*cos
    (q2)+lc3*cos(q2+q3));
20 gammal21=(Iy2-Iz2-m2*lc2*lc2)*cos(q2)*sin(q2)+(Iy3-Iz3)*cos(q2+q3)
    *sin(q2+q3)-m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*sin
    (q2+q3));
21 gammal31=(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)-m3*lc3*sin(q2+q3)*(l2*cos
    (q2)+lc3*cos(q2+q3));
22 gamma211=(Ix2-Iy2+m2*lc2*lc2)*cos(q2)*sin(q2)+(Iz3-Iy3)*cos(q2+q3)
    *sin(q2+q3)+m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*
    sin(q2+q3));
23 gamma223=-l2*m3*lc3*sin(q3);
24 gamma232=-l2*m3*lc3*sin(q3);
25 gamma233=-l2*m3*lc3*sin(q3);
26 gamma311=(Iz3-Iy3)*cos(q2+q3)*sin(q2+q3)+m3*lc3*sin(q2+q3)*(l2*cos
    (q2)+lc3*cos(q2+q3));
27 gamma322=l2*m3*lc3*sin(q3);
28
29 c11=gammal12*qp2+gammal13*qp3; c12=gammal21*qp1; c13=gammal31*qp1;
30 c21=gamma211*qp1; c22=gamma223*qp3; c23=gamma232*qp2+gamma233*qp3;
31 c31=gamma311*qp1; c32=gamma322*qp2;
32 c33=0;
33 Matriz de inercia
34 M=[m11, m12, m13; m21, m22, m23; m31, m32, m33];
35 Matriz de fuerzas centripetas y de coriolis
36 C=[c11, c12, c13; c21, c22, c23; c31, c32, c33];
37 Vector de pares gravitacionales
38 gq11=0;
39 gq21=0.8086*sin(q1)+0.0484*sin(q1+q2);
40 gq31=0.0484*sin(q1+q2);
41 g=9.81;
42 gq=g*[gq11; gq21; gq31];
43 fr=[b1*qp1; b2*qp2; b3*qp3];
44
45 qpp = M^(-1)*(tau-C*qp-gq-fr); Aceleración articular

```

46 **end**

B.1.3. Script: Robot de transmisión directa 3

“Implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores”

capítulo 5: Resultados.

Sección: Implementación de algoritmos de consenso.

Script: robot3gdl3.

```

1 function qpp = robot3gdl3(tau1,tau2,tau3,q1,q2,q3,qp1,qp2,qp3)
2 %vector de posición articular
3 q = [q1;q2;q3];
4 %vector de velocidad articular
5 qp = [qp1;qp2;qp3];
6 %vector de pares
7 tau=[tau1;tau2;tau3];
8
9 Iz1=1.26; Iz2=0.084; Iz3=0.056; Iy1=0.089; Iy2=0.003; Iy3=0.0012;
   Ix1=0.03; Ix2=0.05; Ix3=0.009;   m1=11.523; l1=1.075; b1
   =2.288;
10 fc1=7.17; fe1=8.8; m2=6.94; l2=0.35; lc2=0.057; b2=2.155;
11 fc2=1.9; fe2=2.1; m3=1.18; l3=0.35; lc3=0.041; b3=0.215; fc3
   =1.734; fe3=1.87; %2.29
12
13 m11=Iy2*sin(q2)*sin(q2)+Iy3*sin(q2+q3)*sin(q2+q3)+Iz1+Iz2*cos(q2)*
   cos(q2)+Iz3*cos(q2+q3)*cos(q2+q3)+m2*lc2*lc2*cos(q2)*cos(q2)+
   m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*cos(q2)+lc3*cos(q2+q3));
14 m12=0; m13=0; m21=0; m22=Ix2+Ix3+m3*l2*l2+m2*lc2*lc2+m3*lc3*lc3+
   2*m3*l2*lc3*cos(q3);
15 m23=Ix3+m3*lc3*lc3+m3*l2*lc3*cos(q3);
16 m31=0; m32=Ix3+m3*lc3*lc3+m3*l2*lc3*cos(q3); m33=Ix3+m3*lc3*lc3;
17
18 gamma112=(Iy2-Ix2-m2*lc2*lc2)*cos(q2)*sin(q2)+(Iy3-Iz3)*cos(q2+q3)
   *sin(q2+q3)-m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*sin
   (q2+q3));
19 gamma113=(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)-m3*lc3*sin(q2+q3)*(l2*cos
   (q2)+lc3*cos(q2+q3));
20 gamma121=(Iy2-Iz2-m2*lc2*lc2)*cos(q2)*sin(q2)+(Iy3-Iz3)*cos(q2+q3)

```



```

    *sin(q2+q3)-m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*sin
    (q2+q3));
21 gamma131=(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)-m3*lc3*sin(q2+q3)*(l2*cos
    (q2)+lc3*cos(q2+q3));
22 gamma211=(Ix2-Iy2+m2*lc2*lc2)*cos(q2)*sin(q2)+ (Iz3-Iy3)*cos(q2+q3
    )*sin(q2+q3)+m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*
    sin(q2+q3));
23 gamma223=-l2*m3*lc3*sin(q3);
24 gamma232=-l2*m3*lc3*sin(q3);
25 gamma233=-l2*m3*lc3*sin(q3);
26 gamma311=(Iz3-Iy3)*cos(q2+q3)*sin(q2+q3)+m3*lc3*sin(q2+q3)*(l2*cos
    (q2)+lc3*cos(q2+q3));
27 gamma322=l2*m3*lc3*sin(q3);
28
29 c11=gamma112*qp2+gamma113*qp3; c12=gamma121*qp1; c13=gamma131*qp1;
30 c21=gamma211*qp1; c22=gamma223*qp3; c23=gamma232*qp2+gamma233*qp3;
31 c31=gamma311*qp1; c32=gamma322*qp2;
32 c33=0;
33 %matriz de inercia
34 M=[m11, m12, m13; m21, m22, m23; m31, m32, m33];
35 %matriz de fuerzas centripetas y de coriolis
36 C=[c11, c12, c13; c21, c22, c23; c31, c32, c33];
37 %Vector de pares gravitacionales
38 gq11=0;
39 gq21=0.8086*sin(q1)+0.0484*sin(q1+q2);
40 gq31=0.0484*sin(q1+q2);
41 g=9.81;
42 gq=g*[gq11; gq21; gq31];
43 fr=[b1*qp1; b2*qp2; b3*qp3];
44
45 qpp = M^(-1)*(tau-C*qp-gq-fr); %aceleración articular
46 end

```

B.2. Implementación en Arduino Due

B.2.1. Sketch: Obtención, envío de datos y algoritmo de consenso para el robot

“Implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores”
capítulo 5: Resultados.

Sección: Implementación de algoritmos de consenso.

Sketch: ConsensuscontrolHILrobot1.

```

1 // Robot 1: Send Serial COM3, Receive Serial1 COM6
2 #include "HIL.h"
3 #define pi 3.141592
4
5 BinaryFloat q11,q12,q13,qp11,qp12,qp13,q21,q22,q23,qp21,qp22,qp23,q31,q32,q33,
    qp31,qp32,qp33,tau11,tau12,tau13;
6 int a1 = 0, a2 = 0;
7 float g=9.81,a12=0,b12=0,a13=0,b13=0,Kq=0,Kqp=0,Kl=0,Kd1=0,gq11=0,gq12=0,gq13
    =0;
8
9 void setup() { Serial.begin(115200); Serial1.begin(115200);}
10 void loop() {
11   if (Serial1.available() > 0){
12     a1 = Serial1.read();
13     if (a1 == 0x7E) {
14       //Serial.println("First header ok");
15       a2 = Serial1.read();
16       if (a2 == 0x7E ) {
17         q11.binaryV[0] = Serial1.read();q11.binaryV[1] = Serial1.read();
18         q11.binaryV[2] = Serial1.read();q11.binaryV[3] = Serial1.read();
19         q12.binaryV[0] = Serial1.read();q12.binaryV[1] = Serial1.read();
20         q12.binaryV[2] = Serial1.read();q12.binaryV[3] = Serial1.read();
21         q13.binaryV[0] = Serial1.read();q13.binaryV[1] = Serial1.read();
22         q13.binaryV[2] = Serial1.read();q13.binaryV[3] = Serial1.read();
23         qp11.binaryV[0] = Serial1.read();qp11.binaryV[1] = Serial1.read();
24         qp11.binaryV[2] = Serial1.read();qp11.binaryV[3] = Serial1.read();
25         qp12.binaryV[0] = Serial1.read();qp12.binaryV[1] = Serial1.read();
26         qp12.binaryV[2] = Serial1.read();qp12.binaryV[3] = Serial1.read();
27         qp13.binaryV[0] = Serial1.read();qp13.binaryV[1] = Serial1.read();
28         qp13.binaryV[2] = Serial1.read();qp13.binaryV[3] = Serial1.read();
29         q21.binaryV[0] = Serial1.read();q21.binaryV[1] = Serial1.read();
30         q21.binaryV[2] = Serial1.read();q21.binaryV[3] = Serial1.read();
31         q22.binaryV[0] = Serial1.read();q22.binaryV[1] = Serial1.read();
32         q22.binaryV[2] = Serial1.read();q22.binaryV[3] = Serial1.read();
33         q23.binaryV[0] = Serial1.read();q23.binaryV[1] = Serial1.read();
34         q23.binaryV[2] = Serial1.read();q23.binaryV[3] = Serial1.read();
35         qp21.binaryV[0] = Serial1.read();qp21.binaryV[1] = Serial1.read();
36         qp21.binaryV[2] = Serial1.read();qp21.binaryV[3] = Serial1.read();
37         qp22.binaryV[0] = Serial1.read();qp22.binaryV[1] = Serial1.read();
38         qp22.binaryV[2] = Serial1.read();qp22.binaryV[3] = Serial1.read();
39         qp23.binaryV[0] = Serial1.read();qp23.binaryV[1] = Serial1.read();
40         qp23.binaryV[2] = Serial1.read();qp23.binaryV[3] = Serial1.read();
41         q31.binaryV[0] = Serial1.read();q31.binaryV[1] = Serial1.read();
42         q31.binaryV[2] = Serial1.read();q31.binaryV[3] = Serial1.read();
43         q32.binaryV[0] = Serial1.read();q32.binaryV[1] = Serial1.read();
44         q32.binaryV[2] = Serial1.read();q32.binaryV[3] = Serial1.read();
45         q33.binaryV[0] = Serial1.read();q33.binaryV[1] = Serial1.read();
46         q33.binaryV[2] = Serial1.read();q33.binaryV[3] = Serial1.read();
47         qp31.binaryV[0] = Serial1.read();qp31.binaryV[1] = Serial1.read();
48         qp31.binaryV[2] = Serial1.read();qp31.binaryV[3] = Serial1.read();

```

```

49     qp32.binaryV[0] = Serial1.read(); qp32.binaryV[1] = Serial1.read();
50     qp32.binaryV[2] = Serial1.read(); qp32.binaryV[3] = Serial1.read();
51     qp33.binaryV[0] = Serial1.read(); qp33.binaryV[1] = Serial1.read();
52     qp33.binaryV[2] = Serial1.read(); qp33.binaryV[3] = Serial1.read();
53     a1 = Serial1.read();
54     a2 = Serial1.read();
55     if (a1 == 0x0D && a2 == 0x0A) {
56         a12 = 1; b12 = 1;
57         a13 = 1; b13 = 1;
58         Kq = 0.25;
59         Kqp = 0.25;
60         K1 = 0.25;
61         Kd1 = 0.025;
62 ///////////////////////////////////////////////////Algoritmo de consenso NO lineal: Robot 1
63         gq11 = 0;
64         gq12 = g*(0.8086*sin(q11.floatV)+0.0484*sin(q11.floatV+q12.floatV));
65         gq13 = g*(0.0484*sin(q11.floatV+q12.floatV));
66         //tau11.floatV = - K1*sin(Kd1*qp11.floatV) - a12*sin(Kq*(q11.floatV-
           q21.floatV)) - b12*sin(Kqp*(qp11.floatV-qp21.floatV)) - a13*sin(Kq
           *(q11.floatV-q31.floatV)) - b13*sin(Kqp*(qp11.floatV-qp31.floatV))
           + gq11;
67         //tau12.floatV = - K1*sin(Kd1*qp12.floatV) - a12*sin(Kq*(q12.floatV-
           q22.floatV)) - b12*sin(Kqp*(qp12.floatV-qp22.floatV)) - a13*sin(Kq
           *(q12.floatV-q32.floatV)) - b13*sin(Kqp*(qp12.floatV-qp32.floatV))
           + gq12;
68         tau11.floatV = - K1*tanh(Kd1*qp11.floatV) - a12*tanh(Kq*(q11.floatV-
           q21.floatV)) - b12*tanh(Kqp*(qp11.floatV-qp21.floatV)) - a13*tanh(
           Kq*(q11.floatV-q31.floatV)) - b13*tanh(Kqp*(qp11.floatV-qp31.
           floatV)) + gq11;
69         tau12.floatV = - K1*tanh(Kd1*qp12.floatV) - a12*tanh(Kq*(q12.floatV-
           q22.floatV)) - b12*tanh(Kqp*(qp12.floatV-qp22.floatV)) - a13*tanh(
           Kq*(q12.floatV-q32.floatV)) - b13*tanh(Kqp*(qp12.floatV-qp32.
           floatV)) + gq12;
70         tau13.floatV = - K1*tanh(Kd1*qp13.floatV) - a12*tanh(Kq*(q13.floatV-
           q23.floatV)) - b12*tanh(Kqp*(qp13.floatV-qp23.floatV)) - a13*tanh(
           Kq*(q13.floatV-q33.floatV)) - b13*tanh(Kqp*(qp13.floatV-qp33.
           floatV)) + gq13;
71         Serial.write(0x7E);
72         Serial.write(tau11.binaryV,4);
73         Serial.write(tau12.binaryV,4);
74         Serial.write(tau13.binaryV,4);
75         Serial.write(0x0D);
76     } //Footer 1
77 } //Header 2
78 } //Header 1
79 } //Serial available
80 delay(1);
81 }

```

B.2.2. Sketch: Obtención, envío de datos y algoritmo de consenso para el robot

“Implementación de algoritmos de consenso en un sistema multiagente de
robots manipuladores”

capítulo 5: Resultados.

Sección: Implementación de algoritmos de consenso.

Sketch: ConsensuscontrolHILrobot2.

```

1 //Robot 2: Send Serial COM11, Receive Serial1 COM10
2 #include "HIL.h"
3 #define pi 3.141592
4
5 BinaryFloat q21, q22, q23, qp21, qp22, qp23, q11, q12, q13, qp11, qp12, qp13, q31, q32, q33,
   qp31, qp32, qp33, tau21, tau22, tau23;
6 int a1 = 0, a2 = 0;
7 float g = 9.81, a21=0, b21=0, a23=0, b23=0, Kq=0, Kqp=0, K2=0, Kd2=0, gq21=0, gq22=0,
   gq23=0;
8
9 void setup() {
10 Serial.begin(115200); Serial1.begin(115200);}
11 void loop() {
12 if (Serial1.available() > 0) {
13   a1 = Serial1.read();
14   if (a1 == 0x7E) {
15     //Serial.println("First header ok");
16     a2 = Serial1.read();
17     if (a2 == 0x7E) {
18       q21.binaryV[0] = Serial1.read(); q21.binaryV[1] = Serial1.read();
19       q21.binaryV[2] = Serial1.read(); q21.binaryV[3] = Serial1.read();
20       q22.binaryV[0] = Serial1.read(); q22.binaryV[1] = Serial1.read();
21       q22.binaryV[2] = Serial1.read(); q22.binaryV[3] = Serial1.read();
22       q23.binaryV[0] = Serial1.read(); q23.binaryV[1] = Serial1.read();
23       q23.binaryV[2] = Serial1.read(); q23.binaryV[3] = Serial1.read();
24       qp21.binaryV[0] = Serial1.read(); qp21.binaryV[1] = Serial1.read();
25       qp21.binaryV[2] = Serial1.read(); qp21.binaryV[3] = Serial1.read();
26       qp22.binaryV[0] = Serial1.read(); qp22.binaryV[1] = Serial1.read();
27       qp22.binaryV[2] = Serial1.read(); qp22.binaryV[3] = Serial1.read();
28       qp23.binaryV[0] = Serial1.read(); qp23.binaryV[1] = Serial1.read();
29       qp23.binaryV[2] = Serial1.read(); qp23.binaryV[3] = Serial1.read();
30       q11.binaryV[0] = Serial1.read(); q11.binaryV[1] = Serial1.read();
31       q11.binaryV[2] = Serial1.read(); q11.binaryV[3] = Serial1.read();
32       q12.binaryV[0] = Serial1.read(); q12.binaryV[1] = Serial1.read();
33       q12.binaryV[2] = Serial1.read(); q12.binaryV[3] = Serial1.read();
34       q13.binaryV[0] = Serial1.read(); q13.binaryV[1] = Serial1.read();
35       q13.binaryV[2] = Serial1.read(); q13.binaryV[3] = Serial1.read();
36       qp11.binaryV[0] = Serial1.read(); qp11.binaryV[1] = Serial1.read();
37       qp11.binaryV[2] = Serial1.read(); qp11.binaryV[3] = Serial1.read();
38       qp12.binaryV[0] = Serial1.read(); qp12.binaryV[1] = Serial1.read();
39       qp12.binaryV[2] = Serial1.read(); qp12.binaryV[3] = Serial1.read();
40       qp13.binaryV[0] = Serial1.read(); qp13.binaryV[1] = Serial1.read();
41       qp13.binaryV[2] = Serial1.read(); qp13.binaryV[3] = Serial1.read();
42       q31.binaryV[0] = Serial1.read(); q31.binaryV[1] = Serial1.read();

```

```

43     q31.binaryV[2] = Serial1.read(); q31.binaryV[3] = Serial1.read();
44     q32.binaryV[0] = Serial1.read(); q32.binaryV[1] = Serial1.read();
45     q32.binaryV[2] = Serial1.read(); q32.binaryV[3] = Serial1.read();
46     q33.binaryV[0] = Serial1.read(); q33.binaryV[1] = Serial1.read();
47     q33.binaryV[2] = Serial1.read(); q33.binaryV[3] = Serial1.read();
48     qp31.binaryV[0] = Serial1.read(); qp31.binaryV[1] = Serial1.read();
49     qp31.binaryV[2] = Serial1.read(); qp31.binaryV[3] = Serial1.read();
50     qp32.binaryV[0] = Serial1.read(); qp32.binaryV[1] = Serial1.read();
51     qp32.binaryV[2] = Serial1.read(); qp32.binaryV[3] = Serial1.read();
52     qp33.binaryV[0] = Serial1.read(); qp33.binaryV[1] = Serial1.read();
53     qp33.binaryV[2] = Serial1.read(); qp33.binaryV[3] = Serial1.read();
54     a1 = Serial1.read();
55     a2 = Serial1.read();
56     if (a1 == 0x0D && a2 == 0x0A) {
57         a21 = 1;
58         b21 = 1;
59         a23 = 1;
60         b23 = 1;
61         Kq = 0.25;
62         Kqp = 0.25;
63         K2 = 0.25;
64         Kd2 = 0.025;
65         ///////////Algoritmo de consenso NO lineal: Robot 2
66         gq21 = 0;
67         gq22 = g*(0.8086*sin(q21.floatV)+0.0484*sin(q21.floatV+q22.floatV));
68         gq23 = g*(0.0484*sin(q21.floatV+q22.floatV));
69         //tau21.floatV = - K2*sin(Kd2*qp21.floatV) - a21*sin(Kq*(q21.floatV-
          q11.floatV)) - b21*sin(Kqp*(qp21.floatV-qp11.floatV)) - a23*sin(Kq
          *(q21.floatV-q31.floatV)) - b23*sin(Kqp*(qp21.floatV-qp31.floatV))
          + gq21;
70         //tau22.floatV = - K2*sin(Kd2*qp22.floatV) - a21*sin(Kq*(q22.floatV-
          q12.floatV)) - b21*sin(Kqp*(qp22.floatV-qp12.floatV)) - a23*sin(Kq
          *(q22.floatV-q32.floatV)) - b23*sin(Kqp*(qp22.floatV-qp32.floatV))
          + gq22;
71         tau21.floatV = - K2*tanh(Kd2*qp21.floatV) - a21*tanh(Kq*(q21.floatV-
          q11.floatV)) - b21*tanh(Kqp*(qp21.floatV-qp11.floatV)) - a23*tanh(
          Kq*(q21.floatV-q31.floatV)) - b23*tanh(Kqp*(qp21.floatV-qp31.
          floatV)) + gq21;
72         tau22.floatV = - K2*tanh(Kd2*qp22.floatV) - a21*tanh(Kq*(q22.floatV-
          q12.floatV)) - b21*tanh(Kqp*(qp22.floatV-qp12.floatV)) - a23*tanh(
          Kq*(q22.floatV-q32.floatV)) - b23*tanh(Kqp*(qp22.floatV-qp32.
          floatV)) + gq22;
73         tau23.floatV = - K2*tanh(Kd2*qp23.floatV) - a21*tanh(Kq*(q23.floatV-
          q13.floatV)) - b21*tanh(Kqp*(qp23.floatV-qp13.floatV)) - a23*tanh(
          Kq*(q23.floatV-q33.floatV)) - b23*tanh(Kqp*(qp23.floatV-qp33.
          floatV)) + gq23;
74         Serial.write(0x7E);
75         Serial.write(tau21.binaryV, 4);
76         Serial.write(tau22.binaryV, 4);
77         Serial.write(tau23.binaryV, 4);
78         Serial.write(0x0D);
79     } //Footer 1
80 } //Header 2
81 } //Header 1
82 } //Serial available
83 delay(1);
84 }

```

B.2.3. Sketch: Obtención, envío de datos y algoritmo de consenso para el robot

“Implementación de algoritmos de consenso en un sistema multiagente de robots manipuladores”

capítulo 5: Resultados.

Sección: Implementación de algoritmos de consenso.

Sketch: ConsensuscontrolHILrobot3.

```

1 //Robot 3: Send Serial COM5, Receive Serial1 COM12
2 #include "HIL.h"
3 #define pi 3.141592
4
5 BinaryFloat      q31 , q32 , q33 , qp31 , qp32 , qp33 , q11 , q12 , q13 , qp11 , qp12 , qp13 , q21 , q22 ,
                  q23 , qp21 , qp22 , qp23 , tau31 , tau32 , tau33 ;
6 int a1 = 0 , a2 = 0 ;
7 float g=9.81 , a31=0 , b31=0 , a32=0 , b32=0 , Kq=0 , Kqp=0 , K3=0 , Kd3=0 , gq31=0 , gq32=0 , gq33
   =0 ;
8
9 void setup() { Serial.begin(115200) ; Serial1.begin(115200) ; }
10 void loop() {
11   if ( Serial1.available() > 0 ) {
12     a1 = Serial1.read() ;
13     if ( a1 == 0x7E ) {
14       //Serial.println("First header ok") ;
15       a2 = Serial1.read() ;
16       if ( a2 == 0x7E ) {
17         q31.binaryV[0] = Serial1.read() ; q31.binaryV[1] = Serial1.read() ;
18         q31.binaryV[2] = Serial1.read() ; q31.binaryV[3] = Serial1.read() ;
19         q32.binaryV[0] = Serial1.read() ; q32.binaryV[1] = Serial1.read() ;
20         q32.binaryV[2] = Serial1.read() ; q32.binaryV[3] = Serial1.read() ;
21         q33.binaryV[0] = Serial1.read() ; q33.binaryV[1] = Serial1.read() ;
22         q33.binaryV[2] = Serial1.read() ; q33.binaryV[3] = Serial1.read() ;
23         qp31.binaryV[0] = Serial1.read() ; qp31.binaryV[1] = Serial1.read() ;
24         qp31.binaryV[2] = Serial1.read() ; qp31.binaryV[3] = Serial1.read() ;
25         qp32.binaryV[0] = Serial1.read() ; qp32.binaryV[1] = Serial1.read() ;
26         qp32.binaryV[2] = Serial1.read() ; qp32.binaryV[3] = Serial1.read() ;
27         qp33.binaryV[0] = Serial1.read() ; qp33.binaryV[1] = Serial1.read() ;
28         qp33.binaryV[2] = Serial1.read() ; qp33.binaryV[3] = Serial1.read() ;
29         q11.binaryV[0] = Serial1.read() ; q11.binaryV[1] = Serial1.read() ;
30         q11.binaryV[2] = Serial1.read() ; q11.binaryV[3] = Serial1.read() ;
31         q12.binaryV[0] = Serial1.read() ; q12.binaryV[1] = Serial1.read() ;
32         q12.binaryV[2] = Serial1.read() ; q12.binaryV[3] = Serial1.read() ;
33         q13.binaryV[0] = Serial1.read() ; q13.binaryV[1] = Serial1.read() ;
34         q13.binaryV[2] = Serial1.read() ; q13.binaryV[3] = Serial1.read() ;

```

```

35     qp11.binaryV[0] = Serial1.read(); qp11.binaryV[1] = Serial1.read();
36     qp11.binaryV[2] = Serial1.read(); qp11.binaryV[3] = Serial1.read();
37     qp12.binaryV[0] = Serial1.read(); qp12.binaryV[1] = Serial1.read();
38     qp12.binaryV[2] = Serial1.read(); qp12.binaryV[3] = Serial1.read();
39     qp13.binaryV[0] = Serial1.read(); qp13.binaryV[1] = Serial1.read();
40     qp13.binaryV[2] = Serial1.read(); qp13.binaryV[3] = Serial1.read();
41     q21.binaryV[0] = Serial1.read(); q21.binaryV[1] = Serial1.read();
42     q21.binaryV[2] = Serial1.read(); q21.binaryV[3] = Serial1.read();
43     q22.binaryV[0] = Serial1.read(); q22.binaryV[1] = Serial1.read();
44     q22.binaryV[2] = Serial1.read(); q22.binaryV[3] = Serial1.read();
45     q23.binaryV[0] = Serial1.read(); q23.binaryV[1] = Serial1.read();
46     q23.binaryV[2] = Serial1.read(); q23.binaryV[3] = Serial1.read();
47     qp21.binaryV[0] = Serial1.read(); qp21.binaryV[1] = Serial1.read();
48     qp21.binaryV[2] = Serial1.read(); qp21.binaryV[3] = Serial1.read();
49     qp22.binaryV[0] = Serial1.read(); qp22.binaryV[1] = Serial1.read();
50     qp22.binaryV[2] = Serial1.read(); qp22.binaryV[3] = Serial1.read();
51     qp23.binaryV[0] = Serial1.read(); qp23.binaryV[1] = Serial1.read();
52     qp23.binaryV[2] = Serial1.read(); qp23.binaryV[3] = Serial1.read();
53     a1 = Serial1.read();
54     a2 = Serial1.read();
55     if (a1 == 0x0D && a2 == 0x0A) {
56         a31 = 1;
57         b31 = 1;
58         a32 = 1;
59         b32 = 1;
60         Kq = 0.25;
61         Kqp = 0.25;
62         K2 = 0.25;
63         Kd2 = 0.025;
64         ////////// Algoritmo de consenso NO lineal: Robot 3
65         gq31 = 0;
66         gq32 = g*(0.8086*sin(q31.floatV)+0.0484*sin(q31.floatV+q32.floatV));
67         gq33 = g*(0.0484*sin(q31.floatV+q32.floatV));
68         //tau31.floatV = - K3*sin(Kd3*qp31.floatV) - a31*sin(Kq*(q31.floatV-
q11.floatV)) - b31*sin(Kqp*(qp31.floatV-qp11.floatV)) - a32*sin(Kq
*(q31.floatV-q21.floatV)) - b32*sin(Kqp*(qp31.floatV-qp21.floatV))
+ gq31;
69         //tau32.floatV = - K3*sin(Kd3*qp32.floatV) - a31*sin(Kq*(q32.floatV-
q12.floatV)) - b31*sin(Kqp*(qp32.floatV-qp12.floatV)) - a32*sin(Kq
*(q32.floatV-q22.floatV)) - b32*sin(Kqp*(qp32.floatV-qp22.floatV))
+ gq32;
70         tau31.floatV = - K31*tanh(Kd31*qp31.floatV) - a31*tanh(Kq31*(q31.
floatV-q11.floatV)) - b31*tanh(Kqp31*(qp31.floatV-qp11.floatV)) -
a32*tanh(Kq31*(q31.floatV-q21.floatV)) - b32*tanh(Kqp31*(qp31.
floatV-qp21.floatV)) + gq31;
71         tau32.floatV = - K32*tanh(Kd32*qp32.floatV) - a31*tanh(Kq32*(q32.
floatV-q12.floatV)) - b31*tanh(Kqp32*(qp32.floatV-qp12.floatV)) -
a32*tanh(Kq32*(q32.floatV-q22.floatV)) - b32*tanh(Kqp32*(qp32.
floatV-qp22.floatV)) + gq32;
72         tau33.floatV = - K33*tanh(Kd33*qp33.floatV) - a31*tanh(Kq33*(q33.
floatV-q13.floatV)) - b31*tanh(Kqp33*(qp33.floatV-qp13.floatV)) -
a32*tanh(Kq33*(q33.floatV-q23.floatV)) - b32*tanh(Kqp33*(qp33.
floatV-qp23.floatV)) + gq33;
73         Serial.write(0x7E);
74         Serial.write(tau31.binaryV,4);
75         Serial.write(tau32.binaryV,4);
76         Serial.write(tau33.binaryV,4);

```

```
77     Serial.write(0x0D);  
78     }//Footer 1  
79     }//Header 2  
80     }//Header 1  
81 }//Serial available  
82 delay(1);  
83 }
```


Apéndice C

Publicaciones y participaciones en congreso

En esta sección se encuentran los reconocimientos obtenidos de las diferentes publicaciones presentados durante el desarrollo de este trabajo de tesis, en el orden en que fueron presentados. También se anexan los artículos publicados en revista, el primero se presentó en el *18º Congreso Nacional de Mecatrónica* y lleva por título “ Implementación de un algoritmo de control de trayectoria en un sistema de arquitectura abierta ” y se publicó en el libro *Tecnificación Mecatrónica*. El segundo artículo se presentó en el *Congreso Nacional de Control Automático CNCA* y lleva por título “ Evaluación experimental del desempeño de un controlador basado en pasividad de un robot manipulador ” y se publicó en la revista *Memorias del Congreso Nacional de Control Automático*.

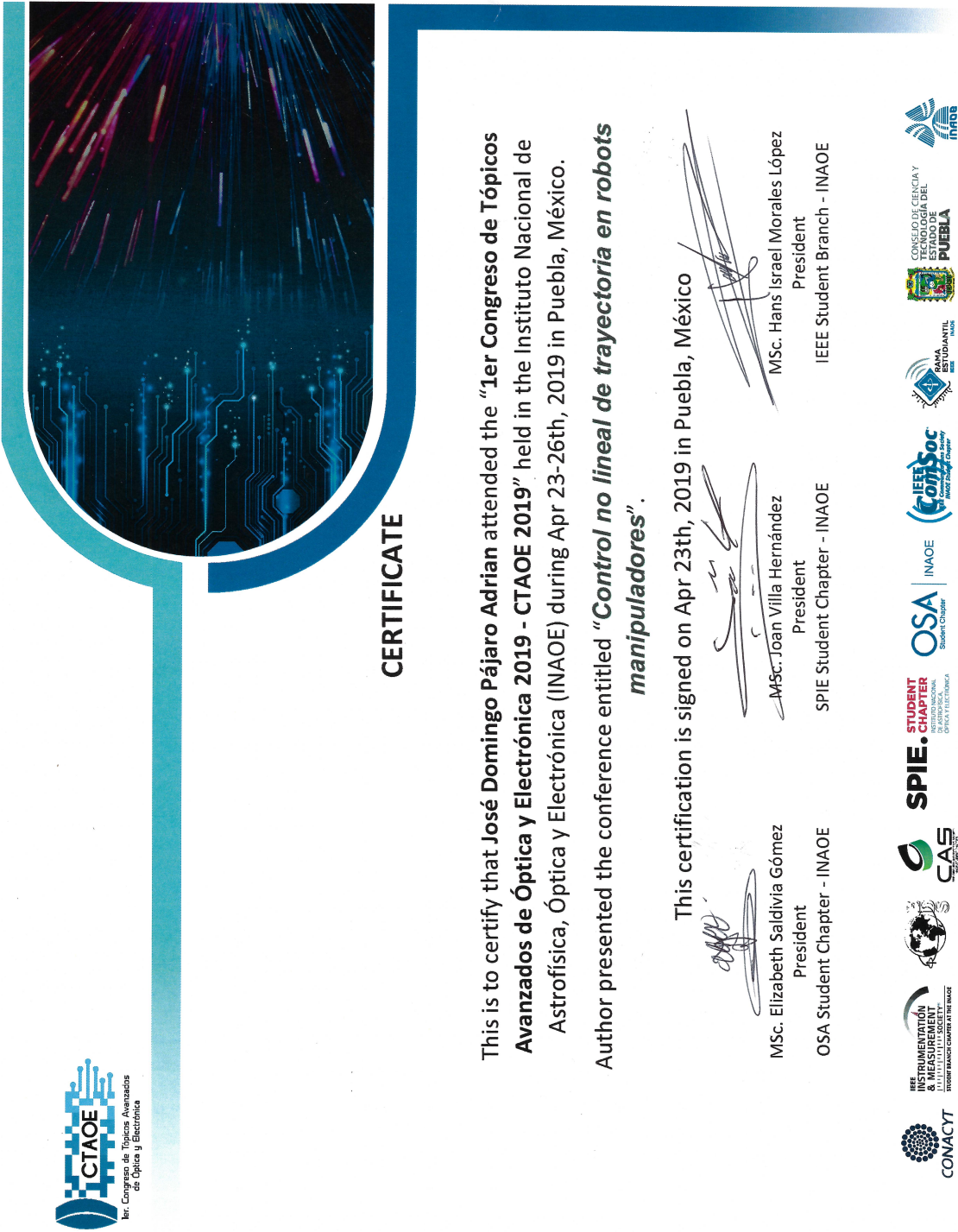


Figura C.1: Constancia de participación en el primer congreso de tópicos avanzados de óptica y electrónica.



Figura C.2: Constancia de participación en el seminario de la maestría en ciencias de la electrónica, opción automatización.



Figura C.3: Constancia de participación en el 18º congreso nacional de mecatrónica.



Figura C.4: Constancia de participación en el congreso nacional de control automático.



Figura C.5: Constancia de participación en el primer foro en investigación y desarrollo tecnológico.



Implementación de un Algoritmo de Control de Trayectoria en un Sistema de Arquitectura Abierta

Pájaro Adrián J. Domingo¹, Reyes Báez Rodolfo²,
Cid Monjaraz Jaime J.¹ y Reyes Cortés Fernando¹

¹Facultad de Ciencias de la Electrónica, Benemérita Universidad Autónoma de Puebla.

²Jan C. Willems Center for Systems and Control, Faculty of Science and Engineering,
University of Groningen.

Resumen

En este trabajo se presenta el diseño y puesta a marcha de un sistema electrónico de arquitectura abierta para la implementación experimental de algoritmos de control de trayectoria en robots manipuladores de transmisión directa. El sistema electrónico de arquitectura abierta que se emplea en este trabajo es una tarjeta Arduino Due a la que se le realiza el acoplamiento de las señales con las que se obtiene la posición actual del robot adaptando un sencillo circuito electrónico para este propósito, también se realiza un ajuste a los convertidores digital-analógico de la tarjeta empleada a través de la adaptación de un circuito electrónico que permita enviar de forma correcta la señal de control al robot manipulador de transmisión directa. Para mostrar que las adaptaciones y ajustes realizados a la tarjeta Arduino Due funcionan de forma correcta se realiza la implementación del algoritmo de control de trayectoria backstepping en un robot manipulador de transmisión directa de dos grados de libertad.

Palabras clave: Arduino, Backstepping, Control de trayectoria, Robot manipulador.

1. Introducción

En la actualidad existe una gran variedad de robots manipuladores que pueden realizar numerosas tareas complejas, sin embargo, la mayoría de estos posee una arquitectura cerrada, es decir, que no es accesible su sistema mecánico o electrónico lo que impide desarrollar investigación en temas como el diseño de controles de posición, de trayectoria, de fuerza, entre otros temas de interés para la comunidad científica, por eso resulta necesario contar con sistemas que tengan una arquitectura abierta que permita desarrollar investigación.

A los robots antropomórficos que por diseño poseen eslabones rígidos unidos por articulaciones giratorias compuestas por servomotores de transmisión directa se les denomina robot de transmisión directa como en (Reyes, 2011). Transmisión directa (Direct drive) significa que el servomotor funciona como una fuente ideal de par o torque, no tiene pérdidas de energía como sucede en motores provistos de engranaje. Direct drive elimina el sistema de engranaje tradicional, por lo que el rotor del motor está directamente acoplado al eslabón, además, con esta tecnología se elimina el juego mecánico y se reduce significativamente el fenómeno de fricción.

Un servomotor está compuesto por motor eléctrico, sensor de posición y el amplificador electrónico (electronic driver), los servomotores de transmisión directa tienen tres modos de operación: posición, velocidad y par. El modo posición permite mover al motor a una posición preestablecida con la desventaja de que en este modo de operación no es posible desplazar objetos o realizar control de fuerza. El modo velocidad permite controlar el movimiento del robot a una velocidad deseada, tanto en modo posición y modo velocidad no es posible implementar otro tipo de controladores conservando las características de sistemas con arquitectura cerrada, por otra parte, el modo de operación por par

Tecnificación Mecatrónica, Capítulo 22, pp. 294-309.
ISBN: 978-607-9394-20-2.



posee características de sistemas con arquitectura abierta, haciendo posible implementar algoritmos de control, por ejemplo, el control PD+ que se presenta en (Kelly, 2006) el algoritmo de control backstepping que se presenta en (Fossen, 1997) o el algoritmo Slotine-Li (Slotine, 1987) entre otros.

Además de la arquitectura mecánica del robot manipulador es necesario contar con arquitectura electrónica que permita realizar la implementación de algoritmos de control. Arduino es un sistema electrónico de arquitectura abierta con una amplia gama de aplicaciones en robótica y mecatrónica como se presenta en (Cortés, 2015), en robots móviles como en (Martínez Guzmán Sonia, 2017) y diversos autores. El Arduino Due, figura 3, está basado en un microcontrolador ARM Cortex-M3. Tiene 54 pines de entrada-salida digital (de los cuales 12 se pueden usar como salidas PWM), 12 entradas analógicas, 4 UART (puertos seriales), un reloj de 84 MHz, conexión compatible con USB, 2 DAC (convertidor digital a analógico), 2 TWI, un conector de alimentación, un conector SPI, un botón de reinicio y un botón de borrado, además de otras características que pueden ser consultadas más a detalle en (Arduino.cc, 2019). Sin embargo, para su correcto funcionamiento requiere de sistemas que permitan acoplar las señales de entrada a la tarjeta y al mismo tiempo ajustar las salidas que por lo general son las señales que se requiere para el control de un sistema físico.

En este trabajo se realiza la implementación de un algoritmo de control de trayectoria para robots manipuladores en una plataforma experimental de arquitectura abierta tanto en la estructura mecánica del robot y en la electrónica necesaria para la implementación del control, por lo que se realiza el acoplamiento de las señales que proporcionan la posición actual del robot a las entradas de la tarjeta Arduino Due, además, se realiza el ajuste de los DAC que están disponibles en la tarjeta, con lo que se controlan los dos grados de libertad del robot manipulador. Para validar que el sistema completo funciona adecuadamente y que puede ser utilizado a nivel investigación y enseñanza de control de robots manipuladores se implementa el control de trayectoria backstepping.

2. CASO DE ESTUDIO

Considere un robot manipulador con dos eslabones rígidos unidos por articulaciones giratorias compuestas por servomotores de transmisión directa como el que se presenta en la figura 1.



Figura 1. Robot manipulador de transmisión directa de dos grados de libertad



además, considere un encoder incremental y un servoamplificador, a continuación, se presentan los modelos de los servomotores que conforman la estructura del robot manipulador:

Tabla 1. Servomotores que conforman la estructura del robot manipulador

Eslabón	Modelo	Resolución	Torque
1	DMG3-1050A-115B	10260 $\frac{\text{pulsos}}{\text{rev}}$	50 Nm
2	DMG3-1004C-115B	10260 $\frac{\text{pulsos}}{\text{rev}}$	4 Nm

Los servomotores están configurados para operar en modo par, en esta configuración se requiere enviar al servomotor una señal que puede estar en el rango de -5V a 5V para su correcto funcionamiento. Los encoder proporcionan dos señales A y B de 5V las cuales están desfazadas 90° como se muestra en la figura 2.

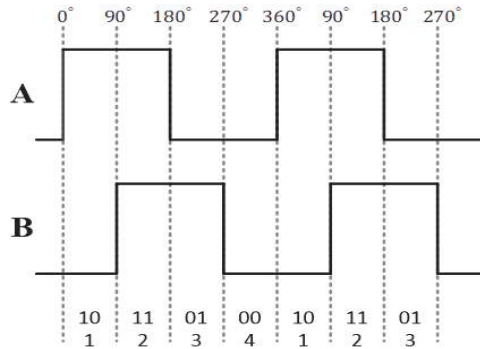


Figura 2. Forma de las señales del encoder.

La implementación de algoritmos de control de trayectoria requiere del conocimiento del modelo dinámico del robot manipulador de transmisión directa de 2 grados de libertad, el cual está expresado como:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + g \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \quad (1)$$

donde

$$M_{11} = m_1 l_{c1}^2 + m_2 l_{c2}^2 + m_2 l_1^2 + 2m_2 l_1 l_{c2} \cos(q_2) + I_1 + I_2, \quad (2)$$

$$M_{12} = m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2, \quad (3)$$

$$M_{21} = m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2, \quad (4)$$

$$M_{22} = m_2 l_{c2}^2 + I_2, \quad (5)$$

$$C_{11} = -m_2 l_1 l_{c2} \sin(q_2) \dot{q}_2, \quad (6)$$

$$C_{12} = -m_2 l_1 l_{c2} \sin(q_2) (\dot{q}_1 + \dot{q}_2), \quad (7)$$

Tecnificación Mecatrónica, Capítulo 22, pp. 294-309.
ISBN: 978-607-9394-20-2.



$$C_{21} = m_2 l_1 l_{c2} \sin(q_2) \dot{q}_1, \quad (8)$$

$$C_{22} = 0, \quad (9)$$

$$g_1 = l_{c1} m_1 \sin(q_1) + m_2 l_1 \sin(q_1) + m_2 l_{c2} \sin(q_1 + q_2), \quad (10)$$

$$g_2 = l_{c2} m_2 \sin(q_1 + q_2). \quad (11)$$

A continuación, se presenta en tabla 2, los valores de los parámetros del robot manipulador de transmisión directa de dos grados de libertad, para los componentes de la matriz de inercia, la matriz de Coriolis y el vector de pares gravitacionales del modelo dinámico de la ecuación (6).

Tabla 2. Parámetros del robot manipulador de dos grados de libertad

Eslabón	Notación	Unidades
Eslabón 1	m_1	6.94 kg
	l_1	0.35 m
	l_{c1}	0.057 m
	I_1	$1.422 \text{ Nm } \frac{s^2}{rad}$
Eslabón 2	m_2	1.18 kg
	l_2	0.35 m
	l_{c2}	0.041 m
	I_2	$0.097 \text{ Nm } \frac{s^2}{rad}$
	g	$9.81 \frac{Nm}{s^2}$

2.1 Tarjeta Arduino Due

El Arduino Due funciona a 3.3V por lo que para la implementación de algoritmos de control se debe considerar que voltajes superiores podrían dañar la tarjeta, también se debe considerar que las salidas de los DAC's incorporados no son ideales ya que estos presentan un offset de aproximadamente 0.500mV por lo que es necesario realizar una etapa de ajuste a 0V, además estos DAC's solo proporcionan voltaje positivo haciendo necesaria la etapa electrónica para la obtención de voltajes negativos y voltajes positivos.

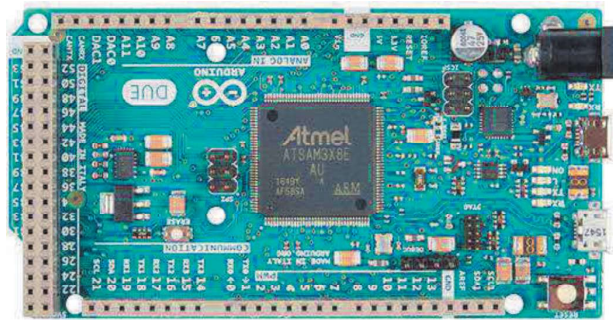


Figura 3. Tarjeta Arduino Due.

3. PLANTEAMIENTO DEL PROBLEMA DE CONTROL DE TRAYECTORIA

Considere un robot manipulador compuesto de n eslabones rígidos unidos a través de articulaciones giratorias con servomotores de transmisión directa y el modelo dinámico que se presenta en la siguiente ecuación:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau, \quad (12)$$

donde $M(q) \in \mathbb{R}^{n \times n}$ es la matriz de inercia, $C(q, \dot{q})\dot{q} \in \mathbb{R}^{n \times n}$ es la matriz de fuerzas centrífugas y de Coriolis, $g(q) \in \mathbb{R}^n$ es el vector de pares gravitacionales y $\tau \in \mathbb{R}^n$ es el vector de pares aplicados. Los vectores $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$ denotan la posición, velocidad y aceleración articular respectivamente. La representación en espacio de estados del modelo dinámico de la ecuación (12) toma la siguiente forma:

$$\frac{d}{dt} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}[\tau - C(q, \dot{q})\dot{q} - g(q)] \end{bmatrix}. \quad (13)$$

Algunas propiedades de interés que se presentan en (Kelly, 2006) del modelo dinámico de la ecuación (12) para diseño de controladores y análisis de estabilidad por Lyapunov se muestran a continuación:

Propiedad 1. La matriz de inercia $M(q) \in \mathbb{R}^{n \times n}$ es simétrica $M(q) = M(q)^T$ y definida positiva $M(q) > 0$.

Propiedad 2. La matriz de fuerzas centrípetas y de Coriolis $C(q, \dot{q})\dot{q}$ y la derivada con respecto al tiempo de la matriz de inercia $\dot{M}(q)$ satisfacen:

$$x^T \left[\frac{1}{2} \dot{M}(q) - C(q, \dot{q}) \right] x = 0 \quad \forall x, \quad (14)$$

es decir, la matriz $\dot{M}(q) - 2C(q, \dot{q})$ es antisimétrica.

El problema de control de trayectoria es formulado en los siguientes términos: Considere el modelo dinámico de la ecuación (12) y una trayectoria de posición deseada, la cual se considera que es suave, continua y al menos dos veces diferenciable. El problema de seguimiento de trayectoria consiste en diseñar una ley de control τ , de tal forma que el vector de posición q siga con precisión a la trayectoria deseada q_d , es decir,

$$\lim_{t \rightarrow \infty} \begin{bmatrix} \tilde{q}(t) \\ \dot{\tilde{q}}(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{2n}, \quad (15)$$

Tecnificación Mecatrónica, Capítulo 22, pp. 294-309.
ISBN: 978-607-9394-20-2.



donde $\tilde{q} \in \mathbb{R}^n$ representa el error de posición y se define como $\tilde{q} := q - q_d$ y el error de velocidad $\dot{\tilde{q}} \in \mathbb{R}^n$ se define como $\dot{\tilde{q}} := \dot{q} - \dot{q}_d$.

4. ACOPLAMIENTO Y AJUSTE DE SEÑALES

En esta sección se presenta los ajustes realizados a la tarjeta Arduino Due para poder realizar la implementación de algoritmos de control de trayectoria en un robot manipulador de transmisión directa de dos grados de libertad, primero se presenta la electrónica que permite acoplar las salidas de los encoder's a la tarjeta Arduino Due y después se presenta la electrónica que permite ajustar y amplificar la salida de los convertidores digital-analógico.

4.1 Acoplamiento de los encoders

Para realizar una correcta lectura de la posición actual del robot manipulador a través de la tarjeta Arduino Due se requiere que la señal se conserve lo más posible sin ruido, es decir, libre de señales no deseadas que alteren la señal del encoder para una correcta lectura. En la figura 4, se presentan las señales que salen del encoder antes de la etapa de acoplamiento.



Figura 4. Señales antes del circuito de acoplamiento.

Para ajustar la señal de salida de los encoder se realiza la implementación de un circuito basado en un transistor tipo npn el cual es operado en modo saturación, en el colector del transistor se coloca el voltaje de referencia en este caso corresponde a los 3.3 V que se requieren a la entrada del Arduino Due, en la base del transistor se inserta la señal del encoder lo que nos permitirá tener en el emisor la señal del encoder acondicionada para la correcta lectura del encoder, el circuito se muestra en la figura 5.

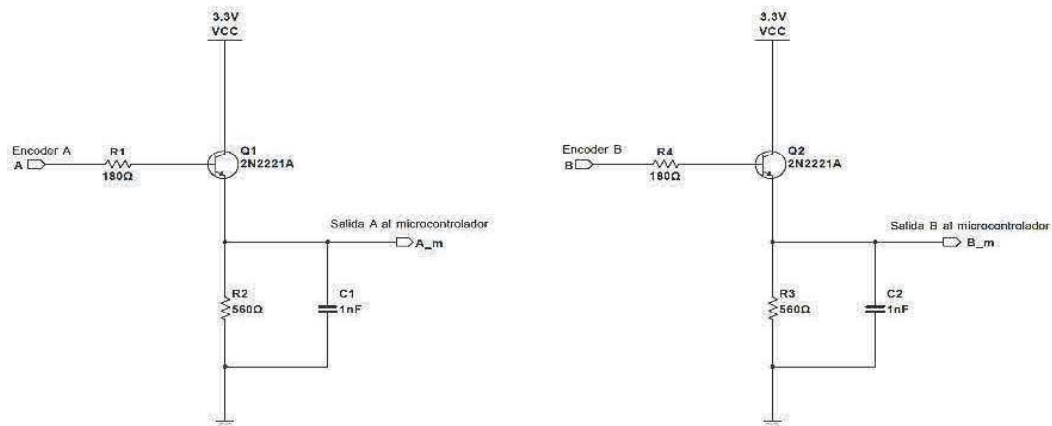


Figura 5. Circuito de acoplamiento del encoder del robot a la tarjeta arduino due.

Después de la etapa de acoplamiento con el circuito propuesto se obtienen las siguientes señales a la salida del encoder como se muestra en la figura 6.

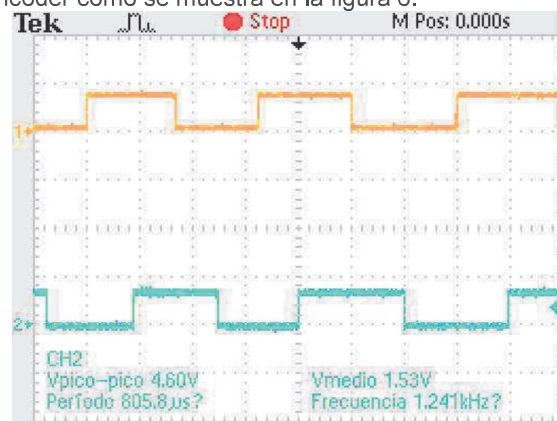


Figura 6. Señales despues del circuito de acoplamiento

4.2 Adaptación de los convertidores digital-analógico de la tarjeta Arduino Due

Los convertidores digital-analógico con los que cuenta la tarjeta Arduino Due pueden ser configurados para operar con resoluciones de 10 o 12 bits, por conveniencia en este trabajo se configuran para operar a 12 bits con lo que el voltaje de salida puede tomar 4096 valores distintos. Considerando las necesidades de adaptación para los DAC del Arduino Due se realiza un circuito electrónico que adapte los DAC's del Arduino a los requerimientos necesarios y se presenta en dos partes.

En la primera etapa se realiza el ajuste del offset del DAC interno del Arduino DUE, a través del diseño de un divisor de voltaje con el cual se pueda obtener 500mV los cuales se emplean para ajustar el offset de los DAC's del Arduino DUE, los valores de las resistencias se obtienen haciendo uso de la siguiente ecuación:

Tecnificación Mecatrónica, Capítulo 22, pp. 294-309.
ISBN: 978-607-9394-20-2.



$$V_S = \frac{R_2 \times V_E}{R_1 + R_2}. \quad (16)$$

La salida del divisor de voltaje se conecta a un amplificador operacional configurado como seguidor esto con la finalidad de desacoplar impedancias entre el divisor de voltaje y el resto del circuito, a continuación la salida del amplificador seguidor se conecta a la terminal negativa del siguiente amplificador operacional que esta configurado como restador inversor el cual se encarga de realizar la resta del offset que tiene el DAC de arduino DUE, los valores de las resistencias para esta parte se obtienen como se muestra en la siguiente ecuación:

$$V_{S1} = V_{ADAC} \left[\frac{(R_3 + R_1)R_4}{(R_4 + R_2)R_1} \right] - V_S \left(\frac{R_3}{R_1} \right). \quad (17)$$

En la figura 7, se muestra el circuito de la primera etapa.

En la segunda etapa, el voltaje obtenido a la salida del amplificador operacional U4B de la figura 7, es de aproximadamente 2.35V, por lo que es necesario implementar un siguiente amplificador operacional que nos permita ajustar a 5V, esto se realiza con el amplificador operacional U4C el cual tiene un arreglo de resistencias R8 y R7 para obtener una ganancia de 2.15 con la que podremos ajustar al voltaje que se requiere esto a través de la siguiente ecuación:

$$V_{S2} = V_{S1} \left(1 + \frac{R_7}{R_8} \right). \quad (18)$$

El amplificador operacional U4D nos permite ajustar el voltaje del DAC de tal forma que nos permita obtener un voltaje que este entre -5V y 5V, como se muestra en la figura 8, con lo que es posible que los servomotores del robot manipulador puedan trabajar en ambos sentidos de giro.

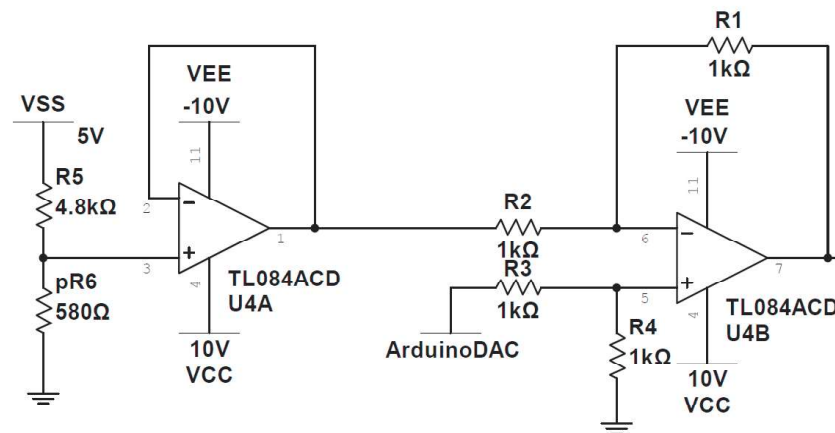


Figura 7. Ajuste de offset del Arduino Due.

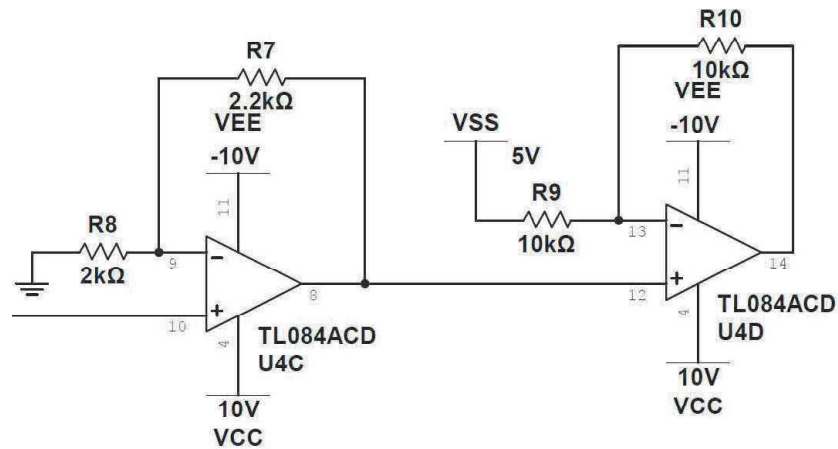


Figura 8. Amplificación de voltaje del DAC de Arduino Due.

Se deben realizar dos circuitos, uno para cada DAC del Arduino Due. Para verificar el correcto funcionamiento del circuito se le realiza un barrido de voltaje, esto con la finalidad de observar la respuesta y asegurar que su comportamiento es lineal. El barrido de voltaje se realiza enviando al DAC del Arduino Due valores en intervalos de 50 con lo que se obtienen 82 muestras de valores analógicos en el rango de -5V a 5V, con un multímetro se toman los valores analógicos obtenidos de cada muestra, este procedimiento se realiza a ambos circuitos con lo que se obtienen las siguientes graficas:

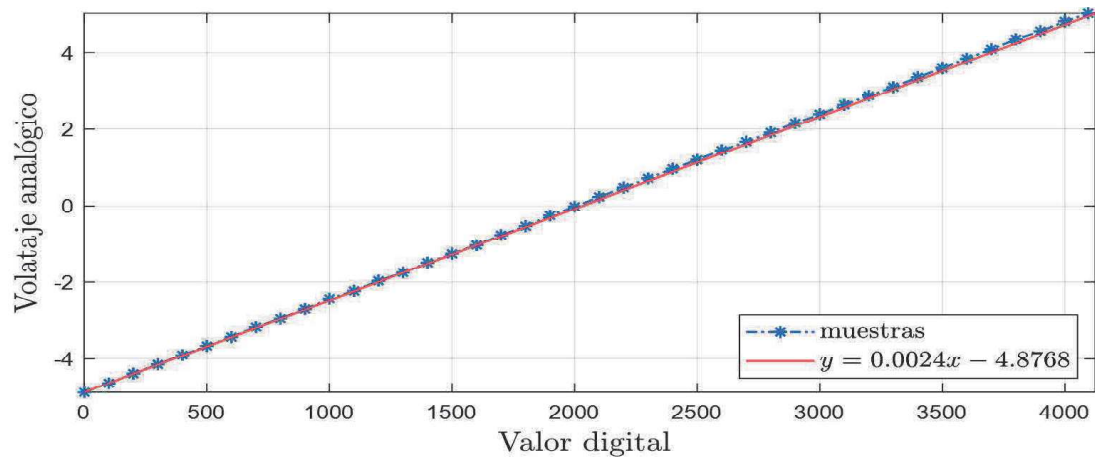


Figura 9. Respuesta del DAC1 de Arduino.Due

Tecnificación Mecatrónica, Capítulo 22, pp. 294-309.
ISBN: 978-607-9394-20-2.

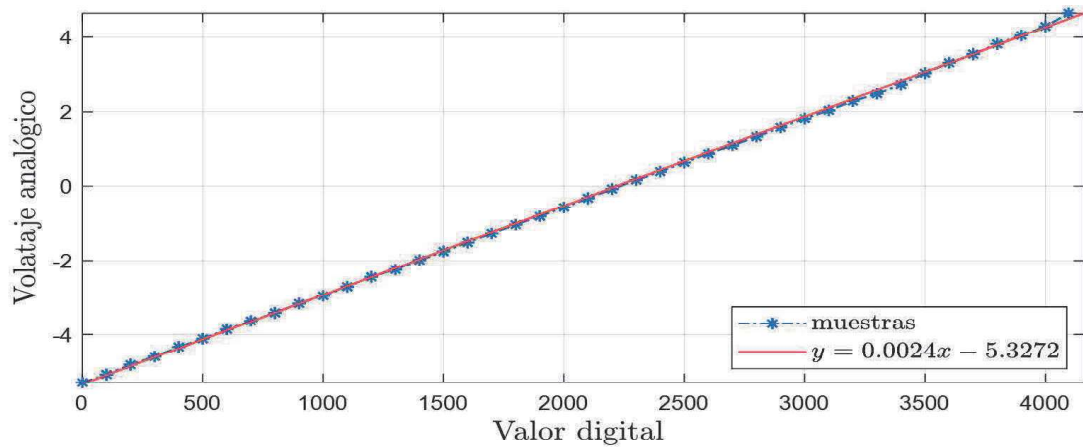


Figura 10. Respuesta del DAC2 de Arduino.Due

En la figura 9, se presenta la respuesta obtenida del circuito implementado al DAC1 del Arduino due, se observa en la línea azul que su comportamiento es lineal, por lo que se realiza la obtención de la ecuación que caracteriza a la recta del DAC1 ya que con esta se realiza el envío de valores con lo que se realiza el control del robot manipulador de 2 grados de libertad, de la misma forma se observa un comportamiento semejante para el DAC2 del Arduino DUE con el circuito implementado por lo que también se obtiene la ecuación de la recta que caracteriza su comportamiento.

5. IMPLEMENTACIÓN Y EVALUACIÓN EXPERIMENTAL

En esta sección se realiza la implementación experimental de un algoritmo de control de trayectoria en el robot manipulador de transmisión directa de dos grados de libertad, en la figura 11, se presenta el diagrama general del sistema, donde se observa la lectura de posición actual del robot y el envío de la señal de control τ , también se presenta la adquisición de datos, la cual se realiza mediante un cable USB conectado del Arduino Due a una computadora en la que se realizan las gráficas con los datos obtenidos de la tarjeta Arduino Due.

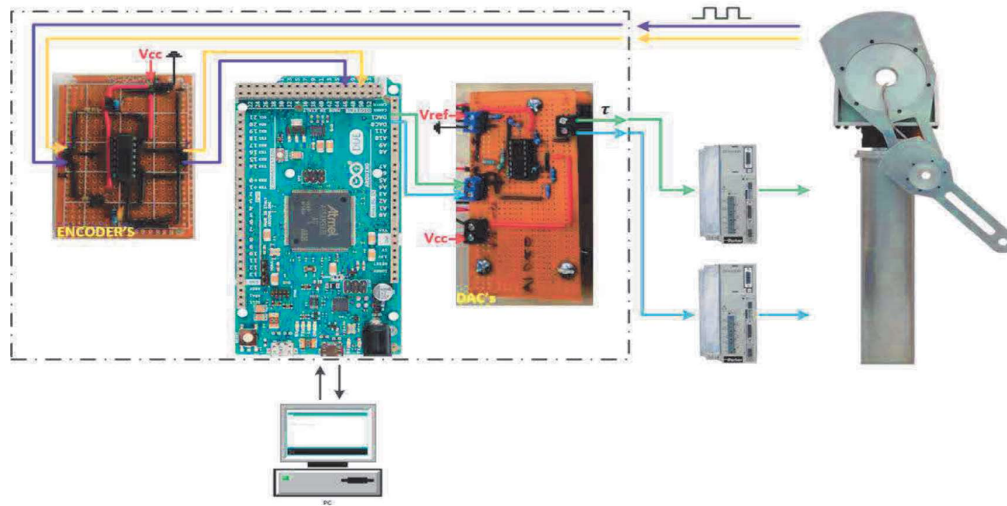


Figura 11. Diagrama general de la planta experimental.

5.1 Trayectoria deseada

Para la implementación del algoritmo de control de trayectoria considere una trayectoria deseada q_d , la cual consiste en un círculo ubicado en el cuarto cuadrante del plano cartesiano y la ecuación que lo describe como se muestra a continuación:

$$x_d = x_c + r \sin(t), \quad (19)$$

$$y_d = -y_c + r \cos(t), \quad (20)$$

donde x_d y y_d representan a la trayectoria deseada en coordenadas cartesianas, r es el radio del círculo con longitud $r = 0.2m$, el centro del círculo en el plano cartesiano se ubica en $x_c = 0.3m$ y $y_c = 0.3m$. Para el diseño e implementación se requiere la primera derivada de las ecuaciones (19) y (20), por lo que se tienen las siguientes expresiones:

$$\dot{x}_d = r \cos(t), \quad (21)$$

$$\dot{y}_d = -r \sin(t). \quad (22)$$

La segunda derivada de las ecuaciones (19) y (20) queda como se muestra enseguida:

$$\ddot{x}_d = -r \sin(t), \quad (23)$$

$$\ddot{y}_d = -r \cos(t). \quad (24)$$

Debido a que es necesario conocer las posiciones deseadas en coordenadas articulares el procedimiento para obtenerlas se realiza a partir de la cinemática inversa del robot manipulador de 2 grados de libertad asociada al cuarto cuadrante como se realiza en (Kelly, 2006), entonces se tiene:

Tecnificación Mecatrónica, Capítulo 22, pp. 294-309.
ISBN: 978-607-9394-20-2.



$$q_{d1} = \frac{\pi}{2} + \tan^{-1}\left(\frac{y_d}{x_d}\right) - \tan^{-1}\left(\frac{l_2 \sin(q_{d2})}{l_1 + l_2 \cos(q_{d2})}\right), \quad (25)$$

$$q_{d2} = \left(\frac{\cos^{-1}(x_d^2 + y_d^2 - l_1^2 - l_2^2)}{2l_1 l_2} \right). \quad (26)$$

5.2 Algoritmo de control de trayectoria

Considere el algoritmo de control de trayectoria backstepping que se presenta en (Fossen, 1997) el cual está dado en la siguiente ecuación:

$$\tau = M(q) \ddot{q}_r + C(q, \dot{q}) \dot{q}_r + g(q) - K_v s - K_p \tilde{q}, \quad (27)$$

donde $K_v \in \mathbb{R}^{n \times n}$ y $K_p \in \mathbb{R}^{n \times n}$ son matrices diagonales de ganancias definidas positivas y q_r es la referencia virtual de velocidad y se define como

$$q_r := -q_d - \Lambda \tilde{q}, \quad (28)$$

donde $\Lambda \in \mathbb{R}^{n \times n}$ es una matriz diagonal de ganancias definida positiva y s es la entrada virtual de control que se define como

$$s := \dot{\tilde{q}} + \Lambda \tilde{q}. \quad (29)$$

Las ganancias que se emplean para la implementación del algoritmo de control backstepping se presentan en la siguiente tabla

Tabla 3. Ganancias para el algoritmo de control backstepping.

Ganancias	Valor
K_{p1}	128
K_{p2}	28
K_{v1}	64
K_{v2}	14
Λ_1	1.5
Λ_2	2

La implementación del algoritmo de control backstepping en el ambiente de programación de Arduino (IDE) se realiza siguiendo el diagrama de flujo de la figura 12.

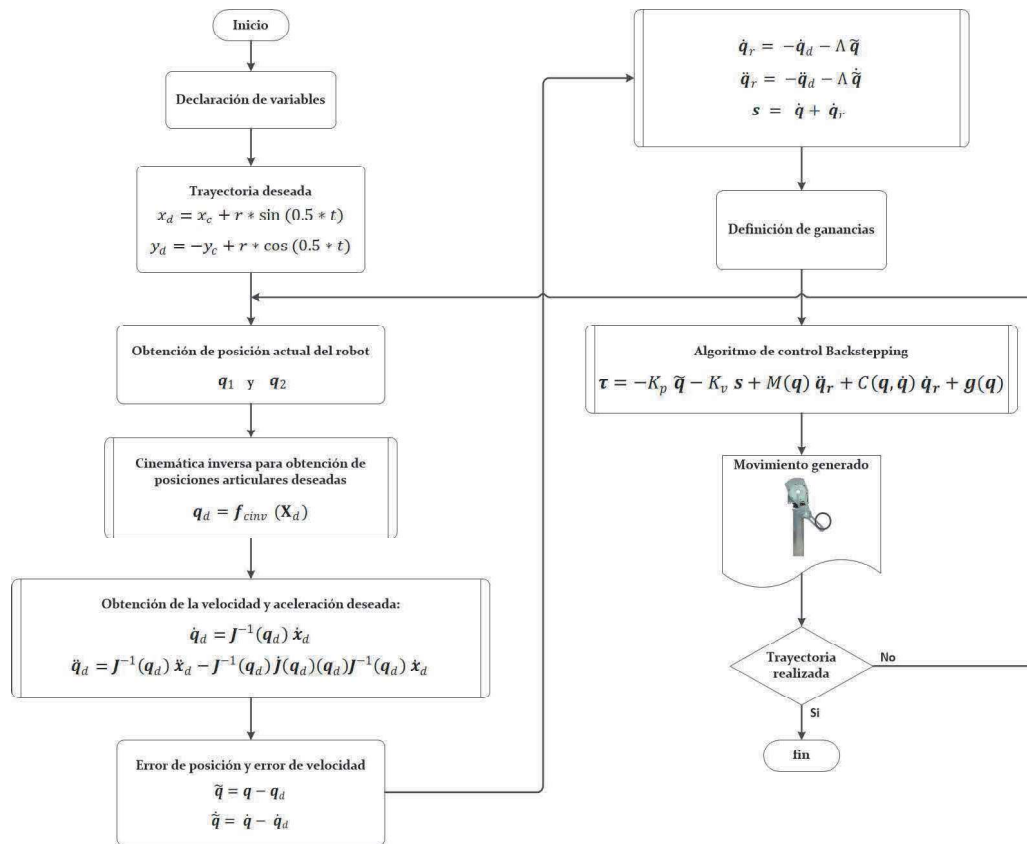


Figura 12. Diagrama de flujo para la implementación del algoritmo de control backstepping.

6. Resultados

En esta sección se presenta los resultados obtenidos de la implementación del algoritmo de control de trayectoria backstepping en la planta experimental de arquitectura abierta para robots manipuladores de transmisión directa de dos grados de libertad propuesta.

Para mostrar que las etapas de adquisición de posición actual del robot a través de la tarjeta Arduino Due con el circuito electrónico de la figura 5 y el envío de la señal de control τ mediante el uso de los DAC's de la tarjeta Arduino Due con la etapa de ajuste y amplificación electrónica de la figura 7 y la figura 8 operan de forma correcta en configuración como se muestra en el diagrama general del sistema de la figura 11, con la trayectoria deseada en el plano cartesiano que se propone y se muestra en las ecuaciones (19) y (20), y el algoritmo de control backstepping de la ecuación (27) y las ganancias propuestas de la tabla 3, se presentan las siguientes gráficas de trayectoria deseada, posiciones articulares y errores de posición y torques empleados.

En la figura 13, se muestra la gráfica de trayectoria deseada en línea azul y la trayectoria realizada por el robot manipulador de transmisión directa de dos grados de libertad en la línea roja, se observa que con las ganancias propuestas para realizar la trayectoria deseada el robot manipulador no alcanza a la posición deseada $q_d(t)$ teniendo un error de posición en su punto máximo de

Tecnificación Mecatrónica, Capítulo 22, pp. 294-309.
ISBN: 978-607-9394-20-2.



aproximadamente 5 grados. Aunque la trayectoria deseada no fue alcanzada por el robot manipulador de transmisión directa no significa que para el sistema sea difícil alcanzar a la posición deseada, lo que significa es que se necesita realizar una correcta estrategia matemática de sintonización de ganancias con las que a través del empleo de cotas máximas y mínimas en pares máximos permitidos así como las cotas en velocidad máxima permitida por los servo actuadores que influyen directamente en la estructura matemática de los algoritmos de control de trayectoria se obtendría una mejora en los resultados obtenidos ya que en este caso la elección de las ganancias empleadas se realizó de forma empírica.

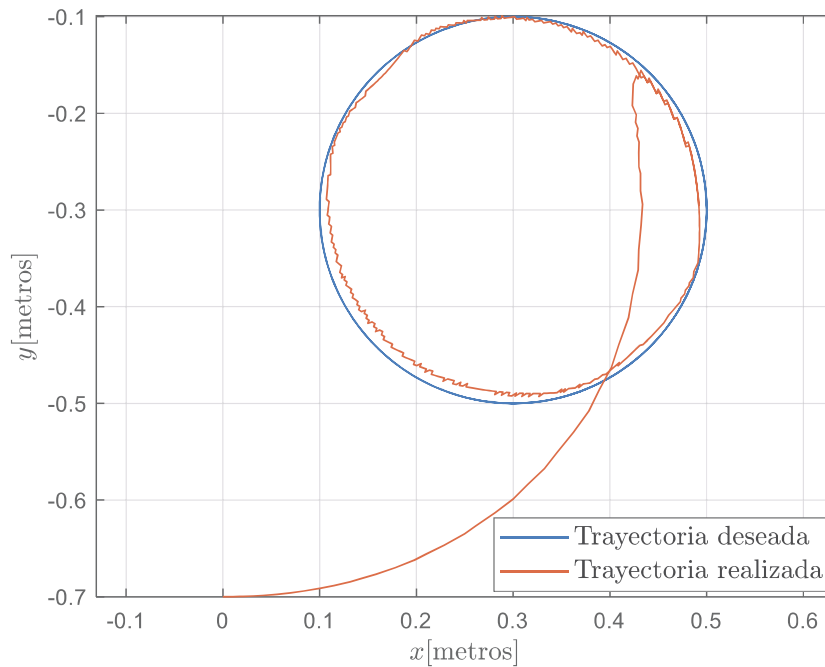


Figura 13. Gráfica comparativa, trayectoria realizada y trayectoria deseada.

En la figura 14, se muestra primero la gráfica de las posiciones articulares que realiza el robot manipulador de transmisión directa con las que se obtiene la trayectoria deseada haciendo uso de la cinemática directa del robot manipulador como se muestra en las siguientes ecuaciones

$$x_r = 0.35 * \text{sen}(q_1) + 0.35 * \text{sen}(q_1 + q_2), \quad (30)$$

$$y_r = -0.35 * \cos(q_1) - 0.35 * \cos(q_1 + q_2), \quad (31)$$

con las que es posible visualizar en el cuarto cuadrante del plano cartesiano la trayectoria realizada por el robot manipulador, después, se presenta la gráfica de los errores de posición para el eslabón 1 (q_1) y el eslabón 2 (q_2), donde se visualiza lo que ya se había evidenciado en la gráfica de la figura 13, que el robot manipulador tiene un considerable error de posición, también se puede visualizar que al robot manipulador le toma 2 segundos en aproximarse a las posiciones deseadas partiendo desde su posición de reposo la cual está ubicado sobre el eje negativo y del plano cartesiano.

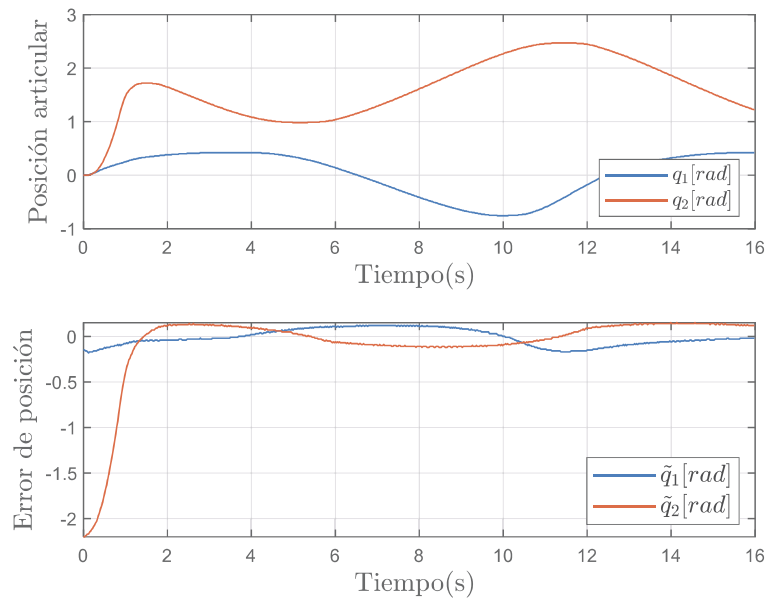


Figura 14. a) Posiciones articulares. b) Error de posición.

En la figura 15, se presentan las gráficas de torques o pares que se envían a los servomotores del robot manipulador como señal de control, en la primera parte se presenta el torque empleado para el servomotor de la articulación 1 se observa que el torque máximo no sobrepasa los 50 Nm permitidos por el servomotor, de igual forma se observa la gráfica de torque para el servomotor de la articulación 2, aunque el torque máximo que puede proporcionar el servomotor es de apenas 4 Nm es suficiente para realizar la trayectoria.

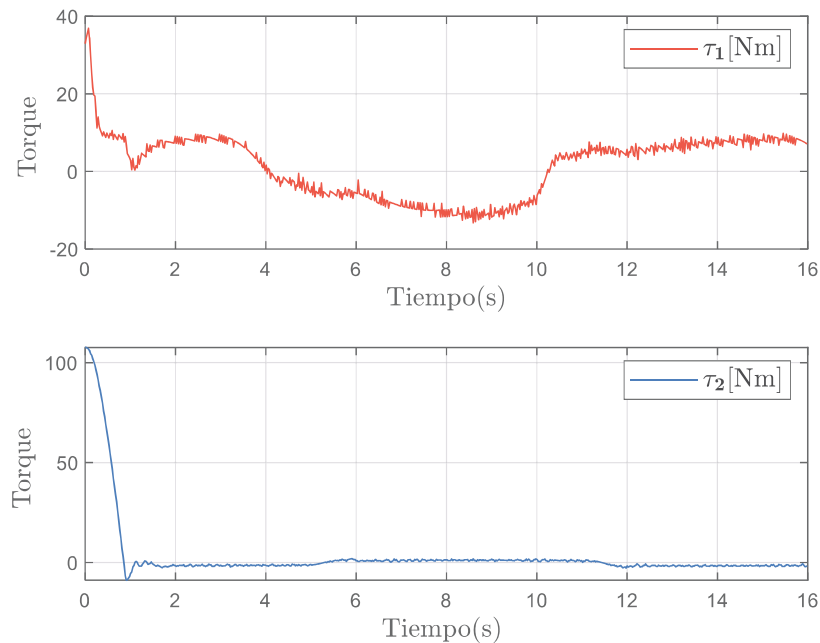


Figura 15. Torques empleados para realizar la trayectoria.

Tecnificación Mecatrónica, Capítulo 22, pp. 294-309.
ISBN: 978-607-9394-20-2.



7. Conclusiones

En este trabajo se realizó la implementación de un algoritmo de control de trayectoria en un sistema de arquitectura abierta, se le denomina de esa forma ya que los servomotores que conforman la estructura mecánica del robot manipulador pueden ser configurados para operar como una fuente de par lo que facilita la implementación de algoritmos de control de trayectoria y otros tipos de controladores, además, se hizo uso de la conocida plataforma Arduino, famosa por su arquitectura abierta que permite a diseñadores, estudiantes e investigadores diseñar diversas aplicaciones mediante sus librerías o modificaciones y adaptaciones que es posible realizar a su arquitectura electrónica con lo que se abre la posibilidad de realizar diseños a bajo costo.

En este trabajo se hizo énfase a utilizar solo dos grados de libertad de los tres disponibles en el robot manipulador de transmisión directa ya que el Arduino Due cuenta con solo dos DAC's, esto no significa que no sea posible implementar el tercer grado de libertad, es posible agregando el un convertidor digital-analógico que se pueda adaptar a la arquitectura Arduino.

Aunque en este trabajo se hizo uso del Arduino Due, actualmente existen tarjetas con mayores capacidades, por ejemplo, el Arduino mkr vidor 4000 el cual incluye dentro de su estructura un FPGA con 16000 elementos lógicos con un reloj que puede ser configurado a 150MHz, el FPGA integrado también puede ser utilizado para operaciones DSP de alta velocidad en procesamiento de audio y video.
y todas las capacidades que incluye la arquitectura Arduino.

El control de trayectoria en robots manipuladores es un tema que ha sido estudiado por diversos investigadores y centros de investigación por lo que hoy en día sigue siendo de interés contar con sistemas que permitan implementar de forma correcta y robusta estos controladores.

Referencias

- [1] Arduino.cc. (15 del 08 del 2019). *Arduino Due*. Obtenido de Arduino Due: <https://store.arduino.cc/usa/due>
- [2] Cortés, F. R. (2015). *Arduino: aplicaciones en robótica, mecatrónica e ingenierías*. Alfaomega.
- [3] Fossen, T. I. (1997). Nonlinear vectorial backstepping design for global exponential tracking of marine vessels in the presence of actuator dynamics. *Proceedings of the 36th IEEE Conference on Decision and Control*, 4237-4242.
- [4] Kelly, R. a. (2006). *Control of robot manipulators in joint space*. Springer Science & Business Media.
- [5] Martínez Guzmán Sonia, C. D. (2017). Aplicación de Arduino para Controlar un Robot Móvil Seguidor de Línea en la Materia de Inteligencia Artificial. En J. E. José Emilio Vargas Soto, *Modelado y Control de Sistemas Mecatrónicos* (págs. 156-166). Asociación Mexicana de Mecatrónica A.C.
- [6] Reyes, F. (2011). *Robótica-Control de robots manipuladores*. Alfaomega grupo editor.
- [7] Slotine, J.-J. E. (1987). On the adaptive control of robot manipulators. *The international journal of robotics research*, 49-59.

Evaluación experimental del desempeño de un controlador basado en pasividad de un robot manipulador

J. Domingo Pájaro-Adrián * Rodolfo Reyes-Báez **
Jaime J. Cid-Monjaraz * Fernando Reyes-Cortés *

* Facultad de Ciencias de la Electrónica, Benemérita Universidad Autónoma de Puebla (e-mail: joc_dom@hotmail.com, [@correo.buap.mx](mailto:jaime.cid,fernando.reyes)).

** Jan C. Willems Center for Systems and Control, Faculty of Science and Engineering, University of Groningen (e-mail: r.reyes-baez@ieee.org)

Resumen: En este trabajo se presenta la evaluación experimental del rediseño de un control de trayectoria basado en pasividad combinado con el procedimiento backstepping para robots manipuladores completamente actuados. El rediseño consiste en modificar el esquema de control backstepping de tal forma que la dinámica del error de posición en lazo cerrado pueda escribirse como la suma de una acción proporcional y una de saturación, esta modificación proporciona una ganancia extra que agrega un grado de libertad en el proceso de sintonización. También se presenta el análisis de estabilidad del sistema en lazo cerrado por pasividad. El desempeño del rediseño propuesto se compara experimentalmente con los algoritmos de control PD+ y backstepping utilizando como indicadores de desempeño el error de posición y el error de velocidad.

Keywords: Pasividad, Backstepping, Control de trayectoria, Robot manipulador.

1. INTRODUCCIÓN

El problema de control de trayectoria para robots manipuladores es un tema que ha sido ampliamente tratado en la literatura especializada con diferentes enfoques para el diseño de controladores que permitan seguir asintóticamente a la trayectoria deseada. Algunos controladores de trayectoria que podemos encontrar en la literatura son, el control por par calculado en Kelly (1990) y el control PD+ en Kelly et al. (2006), de forma similar en Siciliano et al. (2010) se presentan diseños de controladores que se abordan mediante el enfoque de moldeo de energía y el control feedforward (que se usa para evitar errores o perturbaciones al sistema en lazo cerrado). Otros autores han mostrado su interés por el control basado en pasividad, esta es una propiedad de entrada-salida presente en sistemas dinámicos y que puede ser aplicada para análisis de estabilidad y el diseño de controladores de trayectoria para robots manipuladores como se presenta en Lozano et al. (1992), Ortega et al. (1997), Siciliano and Villani (1996), Spong and Vidyasagar (2008) y Canudas et al. (2012).

Además de los controladores mencionados anteriormente en la literatura se muestran diseños de controladores de trayectoria en sistemas con estructura triangular como

lo muestran Ortega et al. (1997), Ortega et al. (2013), Canudas et al. (2012) y Van Der Schaft (2017), cuyos diseños están basados en pasividad para la estabilización de sistemas. Uno de los enfoques que ha mostrado grandes ventajas para el diseño de controladores en sistemas con estructura triangular es el conocido método de backstepping y cuya relación con pasividad es mostrada en Lozano et al. (1992), backstepping es un procedimiento recursivo que consiste en extender una función de Lyapunov de un sistema simple a sistemas que incluyan variables de estado adicionales y al mismo tiempo diseñar el control por retroalimentación para garantizar estabilidad como se presenta en Krstic et al. (1995) y Sepulchre et al. (2012).

En este trabajo se hace uso de la técnica backstepping que se presenta en Fossen and Berge (1997) con lo que se realiza la propuesta de un rediseño, este consiste en modificar al control agregando una estructura que por diseño incorpora dos ganancias estabilizadoras que permiten seguir asintóticamente la trayectoria deseada implementada en un robot manipulador de transmisión directa de 2 grados de libertad. El análisis de estabilidad del rediseño propuesto y del sistema completo se realiza por el enfoque de pasividad debido a que el análisis puede ser interpretado en términos de entrada-salida de la energía del sistema. Con el fin de comparar el

desempeño del rediseño propuesto se implementan otros controladores, el control backstepping que se presenta en Fossen and Berge (1997) y el control PD+ que se presenta en Kelly et al. (2006).

2. PRELIMINARES

Considere un sistema dinámico en espacio de estados con entrada y salida de la forma

$$\Sigma : \begin{cases} \dot{x} = f(x, u), \\ y = h(x, u), \end{cases} \quad (1)$$

donde $x \in \mathbb{R}^n$ es el estado, $u \in \mathbb{R}^m$ es la entrada y $y \in \mathbb{R}^m$ es la salida. Tome en cuenta que Σ , es un sistema físico con la propiedad de que su energía se puede aumentar solo a través del suministro de una fuente externa. Consideremos las siguientes definiciones que pueden ser consultadas de forma más detallada en Van Der Schaft (2017) y en Sepulchre et al. (2012):

Definición 1. (Disipatividad) El sistema Σ es disipativo con respecto a la tasa de suministro $s(u, y) : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ si y solo si existe una función de almacenamiento $S : \mathbb{R}^n \rightarrow \mathbb{R}$, talque

$$S(x(T)) \leq S(x(0)) + \int_0^T s(u(t), y(t)) dt, \quad (2)$$

para todo $u \in U$ y todo $x(t) \in X$ para todo $t \in [0, T]$. La función $S(x)$ se denomina función de almacenamiento.

Definición 2. (Pasividad) El sistema Σ es pasivo si es disipativo con respecto a la tasa de suministro $s(u, y) = u^\top y$. Σ es de entrada estrictamente pasiva si existe $\delta > 0$ tal que Σ es disipativo con tasa de suministro $s(u, y) = u^\top y - \delta \|u\|^2$. Σ es de salida estrictamente pasiva si existe $\varepsilon > 0$ tal que Σ es disipativo con respecto a $s(u, y) = u^\top y - \varepsilon \|y\|^2$. Σ es sin pérdida si es conservativo con respecto a $s(u, y) = u^\top y$.

La propiedad de pasividad en sistemas dinámicos puede ser usada como una herramienta de estabilización en el sentido de Lyapunov tomando a la función $S(x)$ como una función candidata de Lyapunov, consideramos el siguiente teorema:

Teorema 1. (Pasividad y estabilidad) Consideramos que el sistema Σ es pasivo con función de almacenamiento $S(x)$, entonces se cumplen las siguientes propiedades:

- (i) Si $S(x)$ es definida positiva, entonces el equilibrio $x = 0$ de Σ con $u = 0$ es estable.
- (ii) Si Σ es pasivo estricto de salida, entonces es $\mathcal{L}_2$ estable con ganancia finita.
- (iii) Si Σ es pasivo estricto de estado con $S(x)$ definida positiva, entonces el equilibrio $x = 0$ de Σ es asintóticamente estable.

Cuando la función de almacenamiento $S(x)$ es una función radialmente desacotada, estas propiedades son globales, es decir, el equilibrio del sistema es globalmente-asintóticamente estable.

3. PLANTEAMIENTO DEL PROBLEMA

Considere un robot manipulador compuesto de n eslabones rígidos unidos a través de articulaciones giratorias con servomotores de transmisión directa y el modelo dinámico que se presenta en la siguiente ecuación:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau, \quad (3)$$

donde $M(q) \in \mathbb{R}^{n \times n}$ es la matriz de inercia, $C(q, \dot{q})\dot{q} \in \mathbb{R}^{n \times n}$ es la matriz de fuerzas centrífugas y de Coriolis, $g(q) \in \mathbb{R}^n$ es el vector de pares gravitacionales y $\tau \in \mathbb{R}^n$ es el vector de pares aplicados. Los vectores $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$ denotan la posición, velocidad y aceleración articular respectivamente. La representación en espacio de estados del modelo dinámico de la ecuación (3) toma la siguiente forma

$$\frac{d}{dt} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M(q)^{-1} [\tau - C(q, \dot{q})\dot{q} - g(q)] \end{bmatrix}. \quad (4)$$

Algunas propiedades de interés que se presentan en Kelly et al. (2006) del modelo dinámico de la ecuación (3) para diseño de controladores y análisis de estabilidad por Lyapunov se muestran a continuación:

Propiedad 1: La matriz de inercia $M(q) \in \mathbb{R}^{n \times n}$ es simétrica $M(q) = M(q)^\top$ y definida positiva $M(q) > 0$.

Propiedad 2: La matriz de fuerzas centrífugas y de Coriolis $C(q, \dot{q})$ y la derivada con respecto al tiempo de la matriz de inercia $\dot{M}(q)$ satisfacen:

$$x^\top \left[\frac{1}{2} \dot{M}(q) - C(q, \dot{q}) \right] x = 0 \quad \forall x,$$

es decir, la matriz $\dot{M}(q) - 2C(q, \dot{q})$ es antisimétrica.

El problema de control de trayectoria es formulado en los siguientes términos: Considere el modelo dinámico de la ecuación (3) y una trayectoria de posición deseada $q_d \in \mathbb{R}^n$, la cual se considera que es suave, continua y al menos dos veces diferenciable. El problema de seguimiento de trayectoria consiste en diseñar una ley de control τ , de tal forma que el vector de posición q siga con precisión a la trayectoria deseada q_d , es decir,

$$\lim_{t \rightarrow \infty} \begin{bmatrix} \tilde{q}(t) \\ \dot{\tilde{q}}(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{2n}, \quad (5)$$

donde $\tilde{q} \in \mathbb{R}^n$ representa el error de posición y se define como $\tilde{q} := q - q_d$ y el error de velocidad $\dot{\tilde{q}} \in \mathbb{R}^n$ se define como $\dot{\tilde{q}} := \dot{q} - \dot{q}_d$.

4. REDISEÑO DEL CONTROLADOR BACKSTEPPING PARA ROBOTS MANIPULADORES

El rediseño propuesto del controlador de trayectoria para robots manipuladores presentado en Fossen and Berge (1997) se enuncia en la siguiente proposición:

Proposición 1. Considere una trayectoria deseada $q_d(t)$ que es suave, continua y al menos dos veces diferenciable, considere también la ley de control dada por

$\tau = M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + g(q) - K_v s - \phi(\tilde{q})$, (6)
donde $K_v \in \mathbb{R}^{n \times n}$ es una matriz de ganancias simétrica y definida positiva; $\tilde{q} = q - q_d$ es el error de posición y

$$s = \dot{q} - \dot{q}_r, \quad (7)$$

es el error de velocidad, donde la referencia de velocidad artificial $\dot{q}_r$ es

$$\dot{q}_r = \dot{q}_d - \phi(\tilde{q}), \quad (8)$$

con $\tilde{q}^\top \phi(\tilde{q}) > 0$. Entonces, el modelo dinámico de la ecuación (3) en lazo cerrado con el controlador de la ecuación (6) satisface que el error de posición y el error de velocidad sigan a la trayectoria deseada como se enuncia en la ecuación (5).

Proof. La demostración de estabilidad mediante el enfoque de pasividad se realiza considerando la siguiente función $S(x)$ como propuesta de una función candidata de Lyapunov,

$$S = \frac{1}{2} s^\top M(q) s + \int_0^{\tilde{q}} \phi(\bar{q}) d\bar{q} \quad (9)$$

La derivada temporal de la función candidata de Lyapunov (9) adquiere la siguiente forma:

$$\dot{S} = \frac{1}{2} s^\top \dot{M}(q) s + s^\top M(q) \dot{s} + \phi^\top(\tilde{q}) \dot{\tilde{q}} \quad (10)$$

$$= \frac{1}{2} s^\top \dot{M}(q) s + s^\top M(q) \dot{s} + \phi^\top(\tilde{q}) (-\phi(\tilde{q}) + s) \quad (11)$$

$$= \frac{1}{2} s^\top \dot{M}(q) s + s^\top M(q) \dot{s} - \phi^\top(\tilde{q}) \phi(\tilde{q}) + \phi^\top s \quad (12)$$

Combinando y realizando operaciones entre la ecuación (7) y la ecuación (8) se tiene

$$M(q)\dot{s} = M(q)\ddot{q}_r - M(q)\ddot{q}_r \quad (13)$$

$$= \tau - M(q)\ddot{q}_r - C(q, \dot{q})\dot{q}_r - g(q) - C(q, \dot{q})s \quad (14)$$

Sustituyendo la ecuación (14) en la ecuación (12)

$$\begin{aligned} \dot{S} &= s^\top (\tau - M(q)\ddot{q}_r - C(q, \dot{q})\dot{q}_r - g(q) + \phi(\tilde{q})) \\ &\quad + s^\top \left(\frac{1}{2} \dot{M}(q) - C(q, \dot{q}) \right) s - \phi^\top(\tilde{q}) \phi(\tilde{q}) \end{aligned} \quad (15)$$

Haciendo uso de la *Propiedad 2*, y considerando la ecuación (6) se tiene

$$\begin{aligned} \dot{S} &= s^\top (M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + g(q) - K_v s - \phi(\tilde{q})) \\ &\quad - M(q)\ddot{q}_r - C(q, \dot{q})\dot{q}_r - g(q) + \phi(\tilde{q}) - \phi^\top(\tilde{q}) \phi(\tilde{q}) \\ &= -s^\top K_v s - \phi^\top(\tilde{q}) \phi(\tilde{q}) < 0 \end{aligned} \quad (16)$$

Remark 1. Este trabajo también puede ser interpretado mediante la técnica de inmersión e invarianza para sistemas mecánicos, donde la posición es variable de estado de la dinámica objetivo y la velocidad es la coordenada fuera de la variedad (superficie), Astolfi and Ortega (2003).

5. CASO DE ESTUDIO

5.1 Plataforma experimental

La implementación y evaluación experimental del control propuesto se realiza en un robot manipulador de

transmisión directa de 2 grados de libertad, cuyo modelo dinámico está dado de la siguiente forma:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + g \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \quad (17)$$

donde

$$M_{11} = m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2} \cos(q_2) + I_1 + I_2, \quad (18)$$

$$M_{12} = m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2, \quad (19)$$

$$M_{21} = m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(q_2) + I_2, \quad (20)$$

$$M_{22} = m_2 l_{c2}^2 + I_2, \quad (21)$$

$$C_{11} = -m_2 l_1 l_{c2} \sin(q_2) \dot{q}_2, \quad (22)$$

$$C_{12} = -m_2 l_1 l_{c2} \sin(q_2) (\dot{q}_1 + \dot{q}_2), \quad (23)$$

$$C_{21} = m_2 l_1 l_{c2} \sin(q_2) \dot{q}_1, \quad (24)$$

$$C_{22} = 0, \quad (25)$$

$$g_1 = l_{c1} m_1 \sin(q_1) + m_2 l_1 \sin(q_1) + m_2 l_{c2} \sin(q_1 + q_2), \quad (26)$$

$$g_2 = m_2 l_{c2} \sin(q_1 + q_2). \quad (27)$$

Para la implementación experimental del rediseño propuesto y los algoritmos de control de trayectoria que se emplean en este trabajo, considere los valores numéricos de la siguiente tabla, para los parámetros de los componentes de la matriz de inercia, la matriz de Coriolis y el vector de pares gravitacionales del modelo dinámico de la ecuación (17).

Tabla 1. Parámetros del robot manipulador de 2 grados de libertad.

Articulación	Notación	Unidades
Hombro	m_1	6.94kg
	l_1	0.35m
	l_{c1}	0.057m
	I_1	1.422Nm $\frac{s^2}{rad}$
	b_1	2.155Nm $\frac{s}{rad}$
Codo	m_2	1.18kg
	l_2	0.35m
	l_{c2}	0.041m
	I_2	0.097Nm $\frac{s^2}{rad}$
	b_2	0.215Nm $\frac{s}{rad}$
	g	9.81 $\frac{Nm}{s^2}$

5.2 Diseño de la trayectoria deseada

La trayectoria deseada consiste en un círculo ubicado en el cuarto cuadrante del plano cartesiano y su ecuación se muestra a continuación:

$$x_d = x_c + r \sin(t), \quad (28)$$

$$y_d = -y_c + r \cos(t), \quad (29)$$

donde x_d y y_d representan a la trayectoria deseada en coordenadas cartesianas, r es el radio del círculo con longitud $r = 0.2m$, el centro del círculo en el plano cartesiano se ubica en $x_c = 0.3m$ y $y_c = 0.3m$. Para el implementación y evaluación experimental del rediseño

propuesto, también se requiere la primera derivada y segunda derivada de las ecuaciones (28) y (29) por lo que se obtienen las siguientes expresiones

$$\dot{x}_d = r \cos(t), \quad (30)$$

$$\dot{y}_d = -r \sin(t), \quad (31)$$

$$\ddot{x}_d = -r \sin(t), \quad (32)$$

$$\ddot{y}_d = -r \cos(t), \quad (33)$$

con lo que se obtiene velocidad y aceleración deseada en coordenadas cartesianas. Además, la implementación requiere de conocer la posición articular deseada está se obtiene de la cinemática inversa del robot manipulador de 2 grados de libertad asociada al cuarto cuadrante empleando las ecuaciones (28) y (29) y las longitudes de los eslabones, hombro y codo de la Tabla 1, entonces

$$q_{d1} = \frac{\pi}{2} + \tan^{-1} \left(\frac{y_d}{x_d} \right) - \tan^{-1} \left(\frac{l_2 \sin(q_{d2})}{l_1 + l_2 \cos(q_{d2})} \right), \quad (34)$$

$$q_{d2} = \frac{\cos^{-1}(x_d^2 + y_d^2 - l_1^2 - l_2^2)}{2l_1 l_2}. \quad (35)$$

5.3 Implementación experimental del controlador propuesto

Considere la ley de control (6) con las siguientes especificaciones para ϕ dadas por

$$\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} \kappa_1 \tilde{q}_1 + \frac{\theta_2}{\theta_1} \tanh(\tilde{q}_1) \\ \kappa_2 \tilde{q}_2 + \frac{\theta_1}{\theta_2} \tanh(\tilde{q}_2) \end{bmatrix}, \quad (36)$$

donde $\kappa_1, \kappa_2, \theta_1, \theta_2 \in \mathbb{R}$ son constantes definidas positivas. Observe que $\phi(\tilde{q})$ en cada componente tiene dos ganancias, $\kappa_{1,2}$ permite una rápida convergencia a la zona de saturación de $\tanh(\tilde{q})$ y $\theta_{1,2}$ permite ajustar la zona lineal de $\tanh(\tilde{q})$.

Considere el control de la ecuación (6), la ecuación (36) y la trayectoria deseada en coordenadas articulares de las ecuaciones (34) y (35). Los valores de las ganancias κ , θ y las ganancias K_v se presentan en la Tabla 2.

Tabla 2. Ganancias elegidas para la implementación del control de la ecuación (6).

Ganancia	Valor
K_{v1}	6.65
K_{v2}	16.8
κ_1	15
κ_2	0.15
θ_1	95
θ_2	48

Con los valores de ganancias elegidos se obtienen las siguientes gráficas que muestran los resultados obtenidos de la implementación experimental:

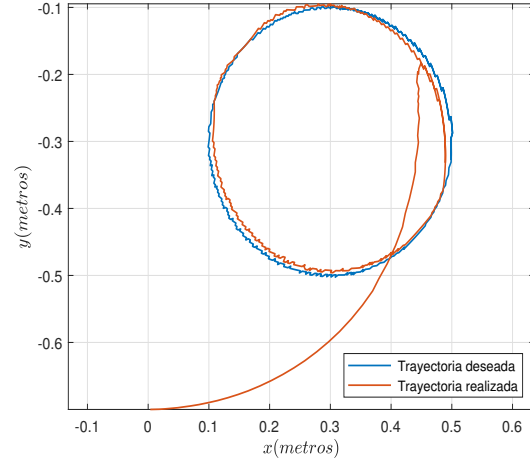


Fig. 1. Trayectoria realizada por el robot manipulador de 2 grados de libertad.

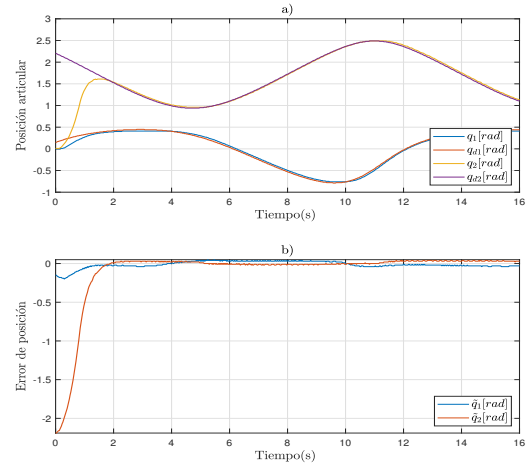


Fig. 2. a) Posiciones articulares deseadas. b) Error de posición.

La Figura 1, muestra la gráfica de la trayectoria que realiza el robot manipulador con el rediseño propuesto (6) en la gráfica se observa que el robot sigue de cerca a la trayectoria deseada, en la gráfica de la Figura 2, se muestran las posiciones articulares deseadas y realizadas por el robot y los errores de posición, también se observa que las articulaciones del robot se aproximan a la trayectoria deseada en un tiempo menor a 2 segundos, después de este tiempo el controlador hace que el error de posición este cerca del cero, este comportamiento es igualmente visible en la gráfica de la Figura 3, donde se muestra que las velocidades articulares siguen de cerca a las velocidades deseadas aproximándose en un tiempo menor a 2 segundos por el robot, el error de velocidad

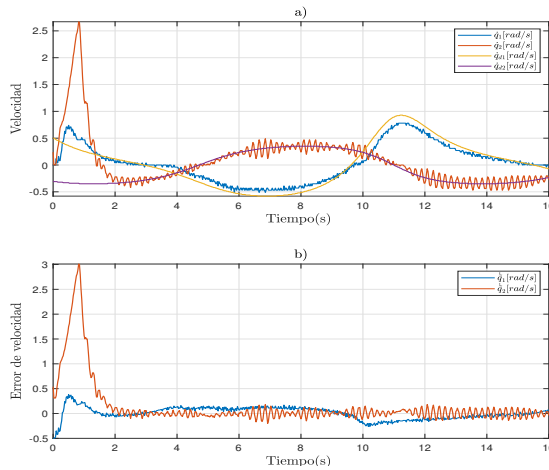


Fig. 3. a) Velocidad articular deseada. b) Error de velocidad.

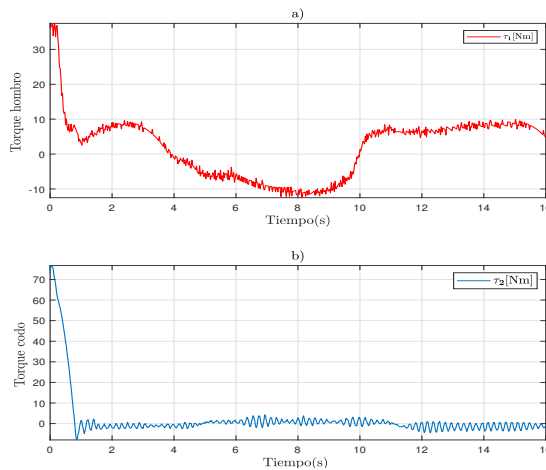


Fig. 4. a) Torque para el hombro. b) Torque para el codo.

muestra que este se mantiene cerca del cero. En la Figura 4, se muestran los torques aplicados al robot para poder realizar la trayectoria, los valores de los torques se encuentran entre $-10Nm$ y $10Nm$ para el hombro y entre $-2Nm$ y $2Nm$ para el codo.

6. IMPLEMENTACIÓN EXPERIMENTAL DE LOS CONTROLES PD+ Y BACKSTEPPING

6.1 Implementación algoritmo PD+

La implementación y evaluación experimental del control de trayectoria PD+ que se presenta en (Kelly et al., 2006), está dado por la siguiente ecuación

$$\tau = M(q)\ddot{q}_d + C(q, \dot{q})\dot{q}_d + g(q) + K_v\dot{\tilde{q}} + K_p\tilde{q}, \quad (37)$$

donde $K_v \in \mathbb{R}^{n \times n}$ y $K_p \in \mathbb{R}^{n \times n}$ son matrices diagonales de ganancias definidas positivas. Para propósitos de comparación de desempeño se consideran las siguientes ganancias:

Tabla 3. Ganancias para el control PD+.

Ganancia	Valor
K_{p1}	95
K_{p2}	48
K_{v1}	6.65
K_{v2}	16.8

Observe que K_p toma los valores de θ_1 y θ_2 del rediseño de control propuesto.

6.2 Implementación algoritmo backstepping

Se realiza la implementación y evaluación experimental del control de trayectoria backstepping (Fossen and Berge, 1997), el cual está dado por la siguiente ecuación:

$$\tau = M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + g(q) - K_v s - K_p \tilde{q} \quad (38)$$

donde $K_v \in \mathbb{R}^{n \times n}$ y $K_p \in \mathbb{R}^{n \times n}$ son matrices diagonales de ganancias definidas positivas y q_r es la referencia virtual de velocidad y se define como

$$q_r := -\dot{q}_d - \Lambda \tilde{q}, \quad (39)$$

además, s es la entrada virtual de control que se define como

$$s := \dot{\tilde{q}} + \Lambda \tilde{q}, \quad (40)$$

donde $\Lambda \in \mathbb{R}^{n \times n}$ es una matriz diagonal de ganancias definida positiva. Para propósitos de comparación de desempeño se consideran las siguientes ganancias:

Tabla 4. Ganancias para el control backstepping.

Ganancia	Valor
K_{p1}	95
K_{p2}	48
K_{v1}	6.65
K_{v2}	16.8
Λ_1	3
Λ_2	2

Donde K_p toma los valores de θ_1 y θ_2 del control propuesto y se agregan las ganancias Λ_1 y Λ_2 .

7. COMPARACIÓN DE DESEMPEÑO

Para poder realizar un procedimiento que nos permita cuantificar el rendimiento de los algoritmos de control de trayectoria que en este trabajo se presentan, se determina el índice de desempeño de cada uno con la norma $\mathcal{L}_2$, tomando como medida de desempeño el error de posición

$\tilde{q}$ y el error de velocidad $\dot{\tilde{q}}$ como se muestra en las siguientes ecuaciones:

$$\mathcal{L}_2[\tilde{q}] = \sqrt{\frac{1}{T} \int_0^T \|\tilde{q}(t)\|^2 dt}, \quad (41)$$

$$\mathcal{L}_2[\dot{\tilde{q}}] = \sqrt{\frac{1}{T} \int_0^T \|\dot{\tilde{q}}(t)\|^2 dt}. \quad (42)$$

Tabla 5. Comparación de desempeño del error de posición y del error de velocidad.

Controlador	Desempeño error de posición	Desempeño error de velocidad
Rediseño propuesto	0.4237	0.5192
PD+	0.4292	0.5147
Backstepping	0.4271	0.6221

En la Tabla 5, se muestran los valores de desempeño obtenidos de la implementación para los tres algoritmos de control, observe que el algoritmo de control propuesto (6) presenta un mejor desempeño tomando como medida el error de posición respecto al control backstepping (38) y al control PD+ (37). Además, se observa que el desempeño del error de la velocidad del control propuesto presenta un mejor desempeño que el desempeño del control backstepping y se observa que el control PD+ presenta un mejor desempeño que el control propuesto, sin embargo, aunque los índices de desempeño se encuentran muy de cerca al control propuesto, backstepping y PD+ tienen un error de posición mayor lo que hace pensar que si se realiza una mejor sintonización de ganancias a los algoritmos de control de trayectoria (37) y (38) el índice de desempeño mostraría menor calidad lejos del desempeño que muestra el rediseño propuesto en este trabajo.

8. CONCLUSIONES

En este trabajo se presentó el rediseño de un control de trayectoria basado en el algoritmo backstepping para robots manipuladores. Se mostró que el rediseño propuesto es asintóticamente estable realizando el análisis de estabilidad por pasividad. Para poder realizar un estudio experimental que nos permita comparar el desempeño del rediseño propuesto se realizó su implementación en un robot manipulador de dos grados de libertad y se consideraron otros algoritmos para su implementación, estos son backstepping y PD+ ya que estos tienen una estructura matemática similar. El estudio del índice de desempeño de los tres algoritmos mostró que el rediseño propuesto presenta mejoría sobre los controladores comparados, por otra parte, el diseño de la trayectoria deseada se presentó en coordenadas cartesianas debido a que es posible presentar el diseño de forma más simple lo que facilita su interpretación. Finalmente, y como trabajo futuro se propone diseñar estrategias de sintonización de ganancias que garanticen que los servomotores no se

saturen y así evitar vibraciones mecánicas y realizar mejor seguimiento de la trayectoria deseada.

REFERENCES

- Astolfi, A. and Ortega, R. (2003). Immersion and invariance: A new tool for stabilization and adaptive control of nonlinear systems. *IEEE Transactions on Automatic control*, 48(4), 590–606.
- Canudas, C., Siciliano, B., and Bastin, G. (2012). *Theory of robot control*. Springer Science & Business Media.
- Fossen, T.I. and Berge, S.P. (1997). Nonlinear vectorial backstepping design for global exponential tracking of marine vessels in the presence of actuator dynamics. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 5, 4237–4242. IEEE.
- Kelly, R. (1990). Adaptive computed torque plus compensation control for robot manipulators. *Mechanism and Machine Theory*, 25(2), 161 – 165.
- Kelly, R., Davila, V.S., and Perez, J.A.L. (2006). *Control of robot manipulators in joint space*. Springer Science & Business Media.
- Krstic, M., Kanellakopoulos, I., Kokotovic, P.V., et al. (1995). *Nonlinear and adaptive control design*. Wiley New York.
- Lozano, R., Brogliato, B., and Landau, I. (1992). Passivity and global stabilization of cascaded nonlinear systems. *IEEE Transactions on Automatic Control*, 37(9), 1386–1388.
- Ortega, R., Jiang, Z.P., and Hill, D.J. (1997). Passivity-based control of nonlinear systems: A tutorial. In *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, volume 5, 2633–2637. IEEE.
- Ortega, R., Perez, J.A.L., Nicklasson, P.J., and Sira-Ramirez, H.J. (2013). *Passivity-based control of Euler-Lagrange systems: mechanical, electrical and electromechanical applications*. Springer Science & Business Media.
- Sepulchre, R., Jankovic, M., and Kokotovic, P.V. (2012). *Constructive nonlinear control*. Springer Science & Business Media.
- Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010). *Robotics: modelling, planning and control*. Springer Science & Business Media.
- Siciliano, B. and Villani, L. (1996). A passivity-based approach to force regulation and motion control of robot manipulators. *Automatica*, 32(3), 443–447.
- Spong, M.W. and Vidyasagar, M. (2008). *Robot dynamics and control*. John Wiley & Sons.
- Van Der Schaft, A. (2017). *L2-gain and passivity techniques in nonlinear control*, volume 3. Springer International Publishing.

Bibliografía

- [1] He Bai, Murat Arcak, and John Wen. *Cooperative control design: a systematic, passivity-based approach*. Springer Science & Business Media, 2011. 1, 4, 22, 23
- [2] Frank L Lewis, Hongwei Zhang, Kristian Hengster-Movric, and Abhijit Das. *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013. 2, 16, 22, 26
- [3] Wei Ren and Randal W Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on automatic control*, 50(5):655–661, 2005. 2
- [4] Xin Chen and Yangmin Li. Smooth formation navigation of multiple mobile robots for avoiding moving obstacles. *International Journal of Control, Automation, and Systems*, 4(4):466–479, 2006. 2
- [5] Reza Olfati-Saber and Richard M Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9):1520–1533, 2004. 2
- [6] Takeshi Hatanaka, Nikhil Chopra, Masayuki Fujita, and Mark W Spong. *Passivity-based control and estimation in networked robotics*. Springer, 2015. 2, 4, 22
- [7] HF McCreery. A comparative approach to cooperative transport in ants: individual persistence correlates with group coordination. *Insectes sociaux*, 64(4):535–547, 2017. 4
- [8] Craig W Reynolds. *Flocks, herds and schools: A distributed behavioral model*, volume 21. ACM, 1987. 4
- [9] Mo Jamshidi et al. Consensus-based and network control of uavs. In *2010 5th International Conference on System of Systems Engineering*, pages 1–6. IEEE, 2010. 4

- [10] Joongbo Seo, Youdan Kim, Seungkeun Kim, and Antonios Tsourdos. Consensus-based reconfigurable controller design for unmanned aerial vehicle formation flight. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 226(7):817–829, 2012. 4
- [11] Ziyang Zhen, Gang Tao, Yue Xu, and Ge Song. Multivariable adaptive control based consensus flight control system for uavs formation. *Aerospace Science and Technology*, page 105336, 2019. 4
- [12] Patrik Kolaric, Ci Chen, Ankur Dalal, and Frank L Lewis. Consensus controller for multi-uav navigation. *Control Theory and Technology*, 16(2):110–121, 2018. 4
- [13] Jin-Bo Hu, Zhi-Wei Liu, and Ming-Feng Ge. Cascade-based control for distributed consensus tracking of multi-usv systems under sampled-data interaction. In *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 1018–1023. IEEE, 2018. 4
- [14] Yassine Bouteraa and Jawhar Ghommam. Synchronization control of multiple robots manipulators. In *2009 6th International Multi-Conference on Systems, Signals and Devices*, pages 1–6. IEEE, 2009. 4
- [15] Shun-Yan Ren, Rui Mao, and Jigang Wu. Passivity-based leader-following consensus control for nonlinear multi-agent systems with fixed and switching topologies. *IEEE Transactions on Network Science and Engineering*, 2018. 4
- [16] Kim D Listmann, Craig A Woolsey, and Jürgen Adamy. Passivity-based coordination of multi-agent systems: a backstepping approach. In *2009 European Control Conference (ECC)*, pages 2450–2455. IEEE, 2009. 4
- [17] Yu Zhao, Zhongkui Li, and Zhisheng Duan. Distributed consensus tracking of multi-agent systems with nonlinear dynamics under a reference leader. *International Journal of Control*, 86(10):1859–1869, 2013. 4
- [18] Long Cheng, Zeng-Guang Hou, and Min Tan. Decentralized adaptive consensus control for multi-manipulator system with uncertain dynamics. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 2712–2717. IEEE, 2008. 4
- [19] Wei Ren and Randal W Beard. *Distributed consensus in multi-vehicle cooperative control*. Springer, 2008. 4, 30

-
- [20] Wei Ren and Yongcan Cao. *Distributed coordination of multi-agent networks: emergent problems, models, and issues*. Springer Science & Business Media, 2010. 4, 28, 35, 36
 - [21] Xudong Chen and Roger W Brockett. Centralized and decentralized formation control with controllable interaction laws. In *53rd IEEE Conference on Decision and Control*, pages 601–606. IEEE, 2014. 4
 - [22] Zhiyong Sun. Distributed stabilization control of rigid formations with prescribed orientations. In *Cooperative Coordination and Formation Control for Multi-agent Systems*, pages 81–99. Springer, 2018. 4
 - [23] Fernando Reyes. *Robótica-Control de robots manipuladores*. Alfaomega grupo editor, 2011. 6, 7, 37
 - [24] Romeo Ortega, Julio Antonio Loría Perez, Per Johan Nicklasson, and Hebertt J Sira-Ramirez. *Passivity-based control of Euler-Lagrange systems: mechanical, electrical and electromechanical applications*. Springer Science & Business Media, 2013. 13, 26, 35
 - [25] Rafael Kelly, Victor Santibáñez Davila, and Julio Antonio Loría Perez. *Control of robot manipulators in joint space*. Springer Science & Business Media, 2006. 13, 34, 47
 - [26] Lorenzo Sciavicco and Bruno Siciliano. *Robotics: Modelling and control of robot manipulators*. Springer Science & Business Media, 2012. 13, 34
 - [27] Frank L Lewis, Darren M Dawson, and Chaouki T Abdallah. *Robot manipulator control: theory and practice*. CRC Press, 2003. 13
 - [28] Carlos Canudas de Wit, Bruno Siciliano, and Georges Bastin. *Theory of robot control*. Springer Science & Business Media, 2012. 13
 - [29] FRANCESCO. BULLO. *LECTURES ON NETWORK SYSTEMS*. CreateSpace, 2018. 16, 17, 18, 21
 - [30] Ravindra B Bapat. *Graphs and matrices*, volume 27. Springer, 2010. 17, 19, 20
 - [31] Rodolphe Sepulchre, Mrdjan Jankovic, and Petar V Kokotovic. *Constructive non-linear control*. Springer Science & Business Media, 2012. 23
 - [32] Hassan Khalil. *Nonlinear Systems*. Prentice Hall, 2002. 24, 26, 35

- [33] Nikhil Chopra and Mark W Spong. Passivity-based control of multi-agent systems. In *Advances in robot control*, pages 107–134. Springer, 2006. 26, 27
- [34] Emmanuel Nuño, Romeo Ortega, Luis Basañez, and David Hill. Trajectory tracking and consensus of networks of euler—lagrange systems. *IFAC Proceedings Volumes*, 44(1):938–943, 2011. 35
- [35] Fernando Reyes Cortés, Jaime Cid Monjaraz, et al. *Mecatrónica: control y automatización*. Alfaomega grupo editor, 2013. 38
- [36] Parker Automation. Dynaserv g3 drive user guide. Technical report, Yokogawa Electric Corporation, 2004. 38, 39
- [37] Edgar Mayoral. Automatización de una celda robótica con retroalimentación visual. Master’s thesis, Benemérita Universidad Autónoma De Puebla, 2017. 38
- [38] Fernando Reyes Cortés and Jaime Cid Monjaraz. *Arduino: aplicaciones en robótica, mecatrónica e ingenierías*. Alfaomega grupo editor, 2015. 40, 53
- [39] Arduino.cc. Tech specs. url: <https://store.arduino.cc/usa/duemilanove>, 2019. 40
- [40] J. Domingo Pájaro Adrián, Rodolfo Reyes Báez, and Jaime Cid Monjaraz. Implementación de un algoritmo de control de trayectoria en un sistema de arquitectura abierta. In *Tecnificación Mecatrónica*, pages 294–309. Asociación Mexicana de Mecatrónica A.C., 2019. 42
- [41] Thor I Fossen and Svein P Berge. Nonlinear vectorial backstepping design for global exponential tracking of marine vessels in the presence of actuator dynamics. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 5, pages 4237–4242. IEEE, 1997. 46