



**BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA**

**FACULTAD DE CIENCIAS DE LA ELECTRÓNICA**

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

**“CONTROL DE ROBOTS COOPERATIVOS POR MEDIO DE  
VISIÓN ARTIFICIAL”**

# **T E S I S**

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

Presenta:

**Lic. Abel Alejandro Rubín Alvarado\***

Directores:

Dr. José Eligio Moisés Gutiérrez Arias (FCE-BUAP)

Dr. Fernando Reyes Cortés (FCE-BUAP)

Dra. Gregoria Corona Morales (FCE-BUAP)

**Puebla, México**

**ENERO 2021**

\* Becario CONACYT.









# Agradecimientos

---

A mis abuelitos Prospero, Candy que siempre han estado pendiente de mi y me alientan a seguir adelante, gracias por todos sus consejos, las grandes historias, por contagiarme su alegría en mis peores momentos.

Agradezco a mis padres Ignacio y María por su apoyo incondicional, muchos de mis logros se los debo a ustedes incluido este, motivándome constantemente para alcanzar mis anhelos.

Si no te tuviera, mi vida seria un desastre. Cada que te miro, me doy cuenta de lo afortunado que soy, y me das esa fuerza que requiero para continuar y seguir con el objetivo de alcanzar mis metas, eres mi principal motivación Damien. Gracias hijo.

Agradezco a mis Directores de tesis, Dr. Moisés Gutiérrez, Dr. Fernando Reyes y Dra. Gregoria Corona por sus enseñanzas tanto en lo académico como en lo personal, sin su guía, dedicación y paciencia que me brindaron no habría sido posible culminar este trabajo.

Agradezco a los miembros del jurado: Dra Aurora Vargas, Dr. Jaime Cid, Dr. César Martínez por tener la disposición y tiempo para orientar este trabajo por un buen camino, sus comentarios siempre fueron muy acertados permitiéndome realizar los cambios necesarios para alcanzar los objetivos.

Agradezco a la Coordinación del posgrado, Dra. Olga Félix, gracias por la oportunidad de pertenecer a esta gran familia, a la Facultad de Ciencias de la Electrónica y a la Universidad, por la infraestructura y formación que me han dado.

A mis amigos por estar ahí en las buenas, en las malas y peores, Pastor, Gaby, Nan... A todos ustedes con mucho cariño.

A Conacyt, por la beca otorgada.



# Índice general

---

|                                                                             |             |
|-----------------------------------------------------------------------------|-------------|
| <b>Índice de figuras</b>                                                    | <b>VII</b>  |
| <b>Índice de tablas</b>                                                     | <b>IX</b>   |
| <b>Introducción</b>                                                         | <b>XIII</b> |
| 0.1. Estado del Arte . . . . .                                              | XIII        |
| 0.2. Justificación . . . . .                                                | XVI         |
| 0.3. Objetivos . . . . .                                                    | XVII        |
| <b>1. Conceptos Involucrados.</b>                                           | <b>13</b>   |
| 1.1. Ecuaciones de Lagrange Partiendo del Principio de D'Alembert . . . . . | 13          |
| 1.2. Teorema de Tikhonov . . . . .                                          | 21          |
| 1.3. Clasificación de los sistemas. . . . .                                 | 24          |
| 1.4. Estabilidad . . . . .                                                  | 25          |
| 1.5. Controlabilidad. . . . .                                               | 31          |
| 1.6. Estabilización. . . . .                                                | 34          |
| 1.7. Programación dinámica. . . . .                                         | 37          |
| 1.8. Ecuaciones en Desviaciones y el Problema de Síntesis Lineal . . . . .  | 41          |
| 1.9. Desigualdades matriciales lineales (LMIs) . . . . .                    | 44          |
| 1.10. Control óptimo de sistemas variables en el tiempo. . . . .            | 47          |
| <b>2. Modelado Dinámico del Robot Líder</b>                                 | <b>51</b>   |
| 2.1. Ecuaciones Dinámicas del Movimiento Controlable de un Robot Móvil. .   | 51          |
| 2.2. Trayectorias deseadas y ecuaciones lineales. . . . .                   | 64          |
| 2.3. Síntesis del control óptimo mediante la programación dinámica. . . . . | 67          |
| 2.4. Síntesis del control óptimo mediante LMI's. . . . .                    | 78          |
| <b>3. Construcción Robot líder.</b>                                         | <b>85</b>   |
| 3.1. Tarjetas FPGA. . . . .                                                 | 85          |
| 3.2. Motor DC. . . . .                                                      | 87          |
| 3.3. Pulse-Width Modulation (PWM). . . . .                                  | 92          |
| 3.4. Codificadores rotatorios. . . . .                                      | 95          |
| 3.5. Módulo comunicaciones UART . . . . .                                   | 99          |

## ÍNDICE GENERAL

---

|                                                                      |            |
|----------------------------------------------------------------------|------------|
| 3.6. Implementación. . . . .                                         | 102        |
| <b>4. Construcción Robot Esclavo</b>                                 | <b>105</b> |
| 4.1. Navegación autónoma. . . . .                                    | 105        |
| 4.2. Raspberry pi 3 B . . . . .                                      | 108        |
| 4.3. Plataforma del robot . . . . .                                  | 109        |
| 4.4. Software . . . . .                                              | 110        |
| <b>5. Resultados y pruebas</b>                                       | <b>119</b> |
| 5.1. Implementación control óptimo en robot líder. . . . .           | 119        |
| 5.2. Resultado experimental de forma colaborativa . . . . .          | 124        |
| <b>6. Conclusiones</b>                                               | <b>127</b> |
| <b>A. Constancias Congreso</b>                                       | <b>131</b> |
| A.1. The 32nd International Conference on Microelectronics . . . . . | 132        |
| <b>B. Códigos Empleados</b>                                          | <b>137</b> |
| B.1. Códigos VHDL . . . . .                                          | 137        |
| B.2. Códigos Matlab . . . . .                                        | 157        |
| B.3. Códigos Python . . . . .                                        | 162        |
| <b>Bibliografía</b>                                                  | <b>167</b> |

# Índice de figuras

---

|                                                                                             |     |
|---------------------------------------------------------------------------------------------|-----|
| 1.1. Posición del centro de masas de un sistema de partículas . . . . .                     | 14  |
| 1.2. Un disco que rueda sin deslizar sobre el plano $x, y$ . . . . .                        | 16  |
| 1.3. Evolución en el tiempo de la estabilidad . . . . .                                     | 27  |
| 2.1. Prototipo inicial . . . . .                                                            | 51  |
| 2.2. Configuración del Robot desplazándose en el área de trabajo . . . . .                  | 52  |
| 2.3. Proyección del Robot en el plano XY . . . . .                                          | 54  |
| 2.4. Diagrama de un motor de DC. . . . .                                                    | 62  |
| 2.5. Trayectorias deseadas. . . . .                                                         | 65  |
| 2.6. Esquema general del sistema. . . . .                                                   | 67  |
| 2.7. Cambio de condiciones iniciales a finales de la solución de Riccati. . . . .           | 74  |
| 2.8. Cambio de condiciones iniciales a finales de la solución de Riccati. . . . .           | 76  |
| 2.9. Simulación del comportamiento de las variables de estado . . . . .                     | 77  |
| 2.10. Plano complejo S. . . . .                                                             | 81  |
| 2.11. Simulación del comportamiento de las variables de estado. . . . .                     | 84  |
| 3.1. Fpga DE0-Nano diagrama de componentes. . . . .                                         | 86  |
| 3.2. Motor con caja reductora y encoder de efecto hall. . . . .                             | 88  |
| 3.3. Selección de voltaje nominal para motores. . . . .                                     | 90  |
| 3.4. Diagrama 1 rama puente H. . . . .                                                      | 90  |
| 3.5. Implementación para el barrido en frecuencia. . . . .                                  | 91  |
| 3.6. PWM por ancho de pulso. . . . .                                                        | 92  |
| 3.7. Diagrama de flujo y Modulo en VHDL, de frecuencias y voltajes de<br>operación. . . . . | 94  |
| 3.8. Encoder con disco codificado para código binario. . . . .                              | 95  |
| 3.9. Encoder de cuadratura por efecto hall. . . . .                                         | 96  |
| 3.10. Salidas del encoder y decodificación en modos 1X, 2X, 4X. . . . .                     | 97  |
| 3.11. Decodificador de dirección 1x. . . . .                                                | 97  |
| 3.12. Verde Sentido horario y púrpura anti-horario, decodificación para 4X. . . . .         | 98  |
| 3.13. Decodificador de cuadratura. . . . .                                                  | 98  |
| 3.14. Modulo para decodificación de encoder y generación de estados. . . . .                | 99  |
| 3.15. Envío 4 paquetes de datos por ciclo. . . . .                                          | 100 |
| 3.16. Bits de identificación y datos en trama de 8 bits UART. . . . .                       | 101 |

## ÍNDICE DE FIGURAS

---

|                                                                        |     |
|------------------------------------------------------------------------|-----|
| 3.17. Vista esquemática del Modulo de comunicación . . . . .           | 101 |
| 3.18. Módulos de BT comerciales. . . . .                               | 102 |
| 3.19. Motores de DC acoplados con las ruedas y rueda loca. . . . .     | 102 |
| 3.20. Diseño de robot Líder en CAD (SolidWorks 2018). . . . .          | 103 |
| 3.21. Diagrama a bloques del robot líder. . . . .                      | 103 |
| 3.22. Vista RTL de módulos implementados en FPGA. . . . .              | 104 |
|                                                                        |     |
| 4.1. RaspiCam . . . . .                                                | 107 |
| 4.2. Tarjeta de desarrollo Raspberry pi 3 B . . . . .                  | 108 |
| 4.3. Plataforma robot móvil esclavo diseñado en 3D . . . . .           | 109 |
| 4.4. Case para Raspberry y cámara . . . . .                            | 109 |
| 4.5. Robot móvil esclavo . . . . .                                     | 110 |
| 4.6. Ambiente de trabajo Raspbian . . . . .                            | 110 |
| 4.7. Obtención de fotogramas para analizar. . . . .                    | 113 |
| 4.8. Representación cubica RGB y cilíndrica HSV. . . . .               | 113 |
| 4.9. Vista de los componentes HSV. . . . .                             | 114 |
| 4.10. Prueba de visión robot esclavo. . . . .                          | 114 |
| 4.11. Pruebas para visión de robot esclavo. . . . .                    | 116 |
| 4.12. Pruebas para identificar en movimiento y distancias. . . . .     | 117 |
| 4.13. Diagrama de operación para robot esclavo. . . . .                | 118 |
|                                                                        |     |
| 5.1. Algoritmo de ordenamiento . . . . .                               | 119 |
| 5.2. Algoritmo implementado en Matlab para Robot Líder. . . . .        | 121 |
| 5.3. Simulación Robot Líder para linea recta sentido positivo. . . . . | 122 |
| 5.4. Resultado experimental del robot líder. . . . .                   | 123 |
| 5.5. Pruebas de trayectoria robot líder. . . . .                       | 123 |
| 5.6. Robot líder y esclavo. . . . .                                    | 124 |
| 5.7. Trayectoria colaborativa . . . . .                                | 124 |
| 5.8. Acceso remoto desde VNC, visión de robot esclavo . . . . .        | 125 |



# Índice de tablas

---

|                                                         |    |
|---------------------------------------------------------|----|
| 2.1. Trayectorias linea recta. . . . .                  | 65 |
| 2.2. Trayectorias deseadas Circular. . . . .            | 66 |
| 2.3. Variables físicas. . . . .                         | 70 |
| 3.1. Información proporcionada por el vendedor. . . . . | 88 |
| 3.2. Barrido de ciclo efectivo de señal PWM. . . . .    | 92 |



# Índice de Algoritmos

---

|                                                                             |     |
|-----------------------------------------------------------------------------|-----|
| B.1. Top Entity:MatE . . . . .                                              | 137 |
| B.2. instance Name-CtrlPosModule:CtrlPos . . . . .                          | 140 |
| B.3. instance Name-Mizq:C1 . . . . .                                        | 143 |
| B.4. UARTModule:UART . . . . .                                              | 147 |
| B.5. tx:C1 . . . . .                                                        | 150 |
| B.6. rx:C2 . . . . .                                                        | 152 |
| B.7. PWMModule:PWM . . . . .                                                | 153 |
| B.8. Comunicacion con Robot Líder y Matlab . . . . .                        | 157 |
| B.9. Algoritmo de Burbuja . . . . .                                         | 162 |
| B.10. Algoritmo en Python para identificar y seguir a robot líder . . . . . | 163 |



# Introducción

---

## 0.1. Estado del Arte

La robótica cada día resuelve nuevas y fascinantes tareas para los robots, con el único afán de evitar que las personas continúen realizando actividades que las pongan en riesgo, o sea de difícil acceso como en zonas de desastres en donde, el reconocimiento, búsqueda y rescate son cruciales.

A finales del siglo pasado, la industria empleaba robots que pueden ser operados y/o supervisados por una persona logrando así mejoras en producción y reducción de costos, cumpliendo estas tareas los investigadores se enfocan en nuevas y mas complicadas actividades, al punto de ya no ser necesaria la intervención humana para su operación, al contrario hemos conceptualizando la idea de tener robots de servicio, logrando así, que muchas de las tareas que son complejas para un solo robot se logren cumplir con la operación de varios de ellos, mas simples dotándolos de un control que puede ser centralizado o descentralizado capaces de lograr un reconocimiento de su entorno y lo mejor, poder trabajar en conjunto para cumplir diversas actividades.

Un control centralizado es considerado cuando todas las funciones de planificación y decisión se realizan en un solo centro de control. Cada robot móvil contiene pocos sensores para el control guiado, tiene los actuadores para el funcionamiento y los medios de comunicación para intercambiar los datos con el centro de control. Todos los movimientos y conflictos entre los robots se resuelven en el centro de control.

Un control descentralizado es cuando cada robot móvil tiene múltiples sensores e incluyen el controlador para que reconozca su entorno y planifique las acciones de control. En cualquier situación imprevista, cada robot está capacitado para planificar un nuevo camino o encontrar una solución sin esperar ordenes del centro de control si es que lo hay.

Los robots móviles son ampliamente utilizados para realizar diversas tareas, en especial en las instalaciones industriales, transporte de mercancía, exploración, en el hogar

y lugares como hospitales, sus configuraciones ya sea empleando orugas, o ruedas, como es el caso de [1] quienes además de emplear un robot móvil, nos dan una opción de coordinar robots de forma cooperativa con la ayuda de visión artificial, bajo una trayectoria desconocida por el robot líder quien en la parte trasera tiene un imagen (patrón) que es identificada por la cámara que se encuentra instalada en un robot, configurado en modo esclavo y dedicado a mantener una distancia específica, logrando una formación de tipo convoy o triangular, considerando unicamente el modelo cinemático del robot esclavo, de igual forma en [2] llevan a cabo un control centralizado, empleando una cámara esta vez colocada de manera fija logrando capturar el área de trabajo de los robots, y mediante la transformación de coordenadas a píxeles logran el control para mantener una formación específica.

Algo que es muy importante empleando un control centralizado es contar con una comunicación adecuada entre los robots y la estación de control, como lo emplean [3] donde el uso de una FPGA se encarga de generar el control y así los robots logren desplazar una barra rígida mediante trayectorias definidas y empleando un control difuso llevan a cabo su implementación en un robot comercial llamado Robotino de la empresa FESTO, además de emplear sistemas embebidos mas robustos, para este tipo de tareas el uso de sensores llevan a desarrollar estrategias de orientación y planificaciones de rutas como lo realizan [4] quienes colocan varios sensores para la detección de obstáculos llevando al robot líder a conocer su entorno y planificar la mejor ruta mediante un control difuso.

El uso de los modelos han llevado a realizar trabajos como el de [5], donde realizan una aproximación de manera numérica al modelo dinámico llevando su implementación a un robot tipo no-holónomico, el móvil es comercial llamado PIONNER 3DX, además del empleo cinemático, realizan un estudio mas profundo para obtener un modelo dinámico nos lleva a poder aplicar más métodos como el que emplean [6] que implementan un controlador de trayectorias basados en la entrada de las velocidades lineales y angulares logrando que la simulación sea prometedora al implementarla en un robot comercial PIONNER 2DX y 3DX, de igual forma [7] quienes emplean robots holónomicos y obtienen los modelos emplean 3 algoritmos para la posición, seguimiento de trayectorias y seguimiento de caminos, esto mediante el empleo de un control cinemático, y uno para compensar la dinámica del robot, y resaltar que la manera en que obtienen el modelo dinámico es basado en el método de Newton-Euler, llevan acabo la implementación en un robot comercial PIONNER P3-AT.

Obteniendo una mejor comprensión de los sistemas gracias a los modelos que describen la dinámica de los mismos dan pauta que los controles que se emplean sean más precisos y cuando introducimos las ventajas de software especializado se logran métodos como el que presentan en [8], quienes utilizan algoritmos RRT por sus siglas en ingles (Rapidly Exploring Random Trees) para trazar rutas, logrando que un robot no-holónomico sea capaz de evitar obstáculos en un ambiente estructurado con solo

considerar punto de partida y el punto final, la implementación se llevo a cabo en un robot comercial modificado PIONNER P3-AT, el empleo de algoritmos y la posibilidad de emplear inteligencia artificial o lograr que el robot aprenda son ideas que inician, debido a que estos algoritmos tienen otros enfoques, sin embargo son consideradas como lo hacen en [9], quien basado en IA por sus siglas (Inteligencia Artificial) emplean multi-agentes logrando así, la evasión de obstáculos para robots en configuración líder-seguidor, en conjunto con una cámara instalada en el robot seguidor y aplicando un algoritmo llamado (BUG2), que en general logra llegar al punto final y en caso de encontrarse un obstáculo, el móvil rodea hasta volver a interceptarse con la línea que inicialmente lo llevaría al punto final, la implementación la realizan en un robot LEGO.

La obtención de un modelo es la base para poder diseñar, obtener o definir una ley de control que nos lleve a conseguir un desempeño óptimo, como lo presentan en [10] que obtienen un modelo cinemático y dinámico mediante el método de Euler-Lagrange, para poder implementar un control predictivo logrando una navegación autónoma de trayectorias deseadas, de esta forma poder reducir una función de costo, logrando así una optimización de dicha función, la idea de optimizar lleva a realizar mas estrategias como la que presentan en [11] para evitar colisiones en un ambiente dinámico, llevando a cabo la definición de ciertas reglas para un control difuso, obteniendo rendimientos mas eficientes por cada evento pasado, y de esta manera el móvil es capaz de tomar una decisión menos costosa de todas las consideradas en las reglas, con esto mantener una ubicación y orientación en todo momento, el móvil no holónomico es dotado de 8 sensores para poder llevar a cabo la tarea.

De igual forma se presentan enfoques donde la navegación se aborda desde un punto de vista probabilístico para su localización y que efectué el mapeo de su entorno, es el trabajo realizado por [12] quienes mediante estas técnicas con algoritmos de navegación autónoma, en conjunto con SLAM por sus siglas en ingles (Simultaneous Localization and Mapping) y empleando sensores ultrasónicos se implementa con un control tipo PID por sus siglas (Controlador Proporcional, Integral y Derivativo). Considerando solo la búsqueda de una ruta corta y bajo un algoritmo DIJKSTRA, también llamado algoritmo de caminos mínimos, es implementado por [13] para la localización y evasión de obstáculos todo de manera centralizada y empleando lógica difusa implementado en un robot comercial LEGO NXT.

Con la búsqueda de la ruta óptima mediante algoritmos genéticos y DIJKSTRA con mapas de visibilidad, en [14] se encuentra la ruta entre dos puntos en un ambiente estructurado, conociendo el mapa de visibilidad y considerando la geometría del robot, este es capaz de evitar colisiones, debido a que esta equipado con sensores para obtener la odometría.

## 0.2. Justificación

En los años 80's y 90's del siglo pasado, se diseñaron un gran número de máquinas cuyo objetivo final es cubrir más tareas, donde este en riesgo una vida humana, actividades industriales, búsqueda y rescate, con riesgo de accidentes, en condiciones peligrosas para la salud, traslado de material peligroso, inclusive la asistencia en hospitales o en nuestro propio hogar.

El área de control ha registrado un creciente interés en la investigación de robots móviles debido a la infinidad de aplicaciones que se pueden desarrollar con ellos, generando algoritmos con capacidades más robustas, adoptando comportamientos que se encuentran en la naturaleza capaces de desempeñar tareas relativamente complejas gracias a la colaboración de entre los robots.

Esta alternativa ofrece ventajas potenciales en términos de flexibilidad de funcionamiento y tolerancia a fallos debido a la redundancia de robots disponibles, y conscientes hasta cierto punto de los robots móviles que se encuentran dentro de su área de trabajo.

EL presente trabajo de tesis tiene como un punto muy importante el poder integrar la modelación matemática de la cinemática y dinámica de un robot móvil, en donde se determinaran las trayectorias deseadas para poder obtener una ley óptima de control, verificando los resultados previos mediante una simulación para poder llevarla a la implementación en un robot móvil desde cero.



## 0.3. Objetivos

### General

Controlar dos robots cooperativos por medio de visión artificial que mantengan una configuración programada.

### Específicos

- Estudio del diseño de los prototipos
- Estudio de las ecuaciones dinámicas del movimiento controlable.
- Determinación de las configuraciones programadas.
- Determinación de una ley optima de control del robot.
- Simulación de la ley de control optima.
- Selección de actuadores y sensores para robot seguidor.
- Pruebas físicas.



---

## Capítulo 1

# Conceptos Involucrados.

---

En este capítulo se dan los principales conceptos, definiciones y principios que se emplearán para el desarrollo de este trabajo de tesis.

### 1.1. Ecuaciones de Lagrange Partiendo del Principio de D'Alembert

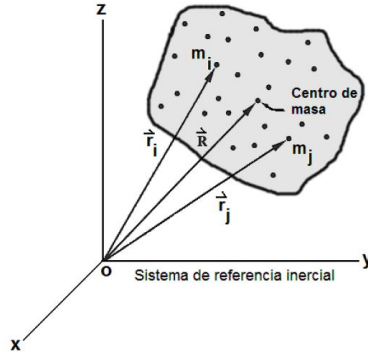
*Definición 1.1 Sistema Discreto: Este modelo de sistema de partículas se considera un cuerpo formado por un número finito de partículas. Dentro de este modelo se consideran sistemas indeformables, en los cuales la distancia relativa entre las partículas del sistema permanece inalterable en el tiempo.*

*Definición 1.2 Se denominan ligaduras a las restricciones sobre las coordenadas de un sistema siendo independientes de las fuerzas actuantes, es decir, son condiciones que restringen en movimiento de una partícula o sistema de partículas.*

#### Centro de masas

Para definir el centro de masas de un sistema de partículas, de acuerdo con [15] partimos de un sistema formado por  $n$  partículas de masas  $m_1, m_2, \dots, m_n$  cuyos vectores de posición son  $\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n$ , con respecto al origen de su sistema de referencia, como se muestra en la figura 1.1. La masa total  $m_T$  del sistema está dada por la ecuación (1.1) que se muestra a continuación:

$$m_T = \sum_{i=1}^n m_i \tag{1.1}$$



**Figura 1.1:** Posición del centro de masas de un sistema de partículas

Una vez definida la  $m_T$  podemos afirmar que:

El *centro de masa* de un sistema de partículas se define como el punto cuyo vector de posición  $\vec{r}_{cm}$  viene dado por ecuación (1.2)

$$\vec{r}_{cm} = \frac{1}{m_T} \sum_{i=1}^n m_i \vec{r}_i \quad (1.2)$$

Y de aquí obtener las correspondientes coordenadas cartesianas de 1.2 dadas por la siguiente ecuación :

$$x_{cm} = \frac{1}{m_T} \sum_{i=1}^n m_i x_i \quad y_{cm} = \frac{1}{m_T} \sum_{i=1}^n m_i y_i \quad z_{cm} = \frac{1}{m_T} \sum_{i=1}^n m_i z_i \quad (1.3)$$

Con estos resultados podemos decir que el **centro de masa** de un sistema de partículas se mueve como si fuera una partícula real, de masa igual a la masa total del sistema sobre el cual actúa la fuerza externa total e independiente de la naturaleza de las fuerza internas, siempre que se cumpla la tercera ley de Newton.

## Clasificación de ligaduras

Las ligaduras se pueden clasificar de varias formas, a continuación se mencionan algunas de ellas conforme lo realiza [15]:

**Definición 1.3** *Unilaterales:* Se dice que una ligadura es **unilateral** cuando se expresa mediante una desigualdad.

**Definición 1.4** Se dice que una ligadura es **bilateral** cuando se expresa mediante una igualdad. Este tipo de ligaduras pueden escribirse de forma general como se muestra en la ecuación (1.4):

$$f_l \left( \vec{r}_i, \dot{\vec{r}}_i, t \right) = 0, \quad \text{con } i = 1, 2, \dots, n; \quad l = 1, 2, \dots, k \quad (1.4)$$

El subíndice  $l$  indica que puede haber mas de una ligadura de este tipo en el sistema, siendo  $k$  el numero total de ellas.

**Definición 1.5 Ligaduras holónomas o geométricas:** Son clasificadas por su integrabilidad, las ligaduras bilaterales que no dependen de las velocidades, sólo dependen de las posiciones de las partículas y el tiempo, exclusivamente. Son integrables, por lo tanto, es posible emplearlas para eliminar las coordenadas dependientes debido que expresan relaciones algebraicas entre ellas.

Este tipo de ligaduras son geométricas (curvas, superficies, etc.) y se pueden escribir de la forma que se muestra en la ecuación (1.5)

$$f_l \left( \vec{r}_i, t \right) = 0, \quad \text{con } i = 1, 2, \dots, n; \quad l = 1, 2, \dots, k \quad (1.5)$$

**Definición 1.6 Ligadura no-holónomica:** Se dice que una ligadura es no-holónomica cuando no se puede escribir como ligadura holónomica, es decir, no se pueden representar como la ecuación (1.5). No son integrables, por lo tanto es imposible emplear las ecuaciones que la expresan para eliminar las coordenadas dependientes.

Basta que un sistema exista al menos una ligadura no-holónoma, para que el sistema se denomine no-holónomo.

En especial, para nuestro caso de estudio revisaremos mas a fondo la ligadura no-holónoma bilateral, como lo es un objeto que rueda (sin deslizamiento perpendicular ni traslacional) sobre una superficie. En especifico, un disco que rueda sobre el plano horizontal  $x, y$  como se muestra en la figura 1.2, obligada a moverse de tal forma que su plano permanezca siempre vertical (el disco puede ser una de las dos ruedas del eje de un carro). Si se eligen como coordenadas para describir el movimiento las coordenadas  $x, y$  del centro del disco, un ángulo de giro  $\phi$  alrededor de su eje, y el ángulo  $\theta$  formado por dicho eje y, por ejemplo, el eje  $x$  como se representa en la figura 1.2, entonces como resultado de la ligadura, la magnitud de la velocidad del centro del disco es proporcional a  $\dot{\phi}$  y así obtenemos la siguiente ecuación (1.6) que nos da esta relación:

$$v = R\dot{\phi} \quad (1.6)$$

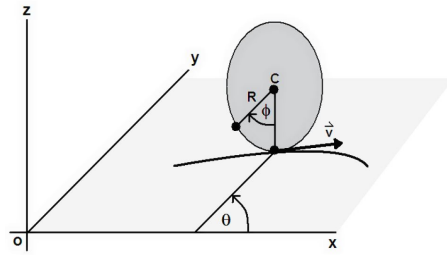
donde  $R$  es el radio del disco, y su dirección es perpendicular al eje del disco. Y con estas consideraciones las componentes están dadas por la siguiente ecuación (1.7):

$$\dot{x} = v \text{Sen} \theta, \quad \dot{y} = -v \text{Cos} \theta \quad (1.7)$$

Obteniendo las componentes y relacionándolas con la ecuación (1.6), se obtienen dos ecuaciones diferenciales de ligadura expresadas por:

$$dx - R \text{Sen} \theta d\phi = 0 \quad dy - R \text{Cos} \theta d\phi = 0 \quad (1.8)$$

que como vimos anteriormente no pueden ser integradas sin resolver. En otras palabras, no se puede encontrar un factor integrante  $f(x, y, \theta, \phi)$  que transforme las ecuaciones (1.8) en diferenciales perfectas, y por lo tanto son de naturaleza no-holónoma.



**Figura 1.2:** Un disco que rueda sin deslizar sobre el plano  $x, y$

**Definición 1.7** Se denominan **coordenadas generalizadas** a un conjunto cualesquiera de magnitudes  $\{q_i, i = 1, \dots, n\}$  que definen la configuración de un sistema de partículas dado. Sus derivadas respecto del tiempo se llaman **velocidades generalizadas**

El estado de un sistema viene determinado por el conocimiento de las coordenadas generalizadas y de sus velocidades generalizadas. Conociendo estas y mediante las ecuaciones de movimiento, junto con las ligaduras, se describe la evolución temporal del sistema.

## Grados de libertad

**Definición 1.8 Grados de libertad** es el número de coordenadas independientes (sin incluir el tiempo) que se requieren para describir completamente la posición de todas y cada una de las partículas o partes componentes del sistema. El término “parte componente” se refiere a cualquier parte del sistema, tal como una palanca, un disco, una plataforma, etc., que deben ser tratados como un cuerpo rígido y no como una partícula.

## Principio de D'Alembert

Para poder efectuar un análisis y así obtener el modelo dinámico del sistema que será objeto de estudio en esta tesis definiremos algunos conceptos los cuales son los siguientes:

*Definición 1.9* Se da el nombre de *desplazamiento real* a todo aquel que puede realizar una partícula (o un conjunto de ellas) en un determinado tiempo  $t$  y, por ende, realizado a una velocidad finita  $v_i$ .

*Definición 1.10* Se da el nombre de *desplazamiento virtual* a un desplazamiento infinitesimal de la posición de una partícula realizado instantáneamente, es decir, que es realizado a velocidad infinita, sin que transcurra el tiempo durante el desplazamiento (de aquí la condición de virtual ya que no es posible realizarlo efectivamente).

Un desplazamiento virtual además de ser instantáneo, es arbitrario no está relacionado con el movimiento real de la partícula en el instante considerado. Un desplazamiento virtual infinitesimal se representará por la diferencial  $\delta r$  en coordenadas cartesianas o por  $\delta q$  en coordenadas generalizadas.

Los desplazamientos virtuales más útiles son los denominados *compatibles con las ligaduras* (no sacan la partícula del riel que la guía, no deforman los cuerpos rígidos, no estiran los hilos, etc.); es decir, que después de realizado un desplazamiento virtual, se preservan las relaciones de ligadura del sistema, expresándolo de la siguiente manera:

$$g_l(q_i, t) = 0 \quad y \quad g_l(q_i + \delta q_i, t) = 0 \quad (1.9)$$

Es posible formular la mecánica de manera que desaparezcan las fuerzas de ligadura y de esa manera evitar muchas de las complicaciones que traen aparejadas. Se dice que un sistema está en equilibrio (estático), si las fuerzas sobre cada una de las partículas que lo componen se anulan  $F_i = 0$ . Si cada partícula sufre un desplazamiento infinitesimal  $\delta r_i$  compatible con las ligaduras, entonces el trabajo realizado por éste es nulo,  $F_i \bullet \delta r_i = 0$ , se llama **trabajo virtual**.

*Definición 1.11* A la ligadura cuya fuerza de ligadura correspondiente no realiza trabajo en los desplazamientos virtuales se le da el nombre de **ligadura ideal**.

Entonces:

$$\delta \Omega := \sum_{i=1}^n F_i \bullet \delta r_i = \sum_{i=1}^n \left( F_i^a + F_i^{lig} \right) \bullet \delta r_i = 0 \quad (1.10)$$

donde  $F_i^a$  es la fuerza aplicada y  $F_i^{lig}$  es la fuerza de ligadura.

En un gran número de sistemas las fuerzas de ligadura son normales a los desplazamientos virtuales y no realizan trabajo. Por ejemplo, si una partícula está forzada a moverse sobre una curva, la fuerza centrífuga es normal a la curva y, por lo tanto, normal al desplazamiento virtual. Los sistemas con fuerzas de deslizamiento son ejemplos donde lo anterior no se cumple: las fuerzas de contacto tienen una componente paralela al desplazamiento virtual y producirán un trabajo virtual. Si nos restringimos a sistemas libres de deslizamiento, las fuerzas de ligadura desaparecen de la ecuación (1.10), y éstas pueden escribirse como:

$$\delta\Omega := \sum_{i=1}^n F_i \bullet \delta r_i = \sum_{i=1}^n F_i^a \bullet \delta r_i = 0 \quad (1.11)$$

La ecuación (1.11) resume en una forma elegante los principios de la estática de partículas vinculadas. Su importancia es tan grande que se conoce como el siguiente principio:

**Principio de los trabajos virtuales.** *Un sistema mecánico sometido a ligaduras ideales permanece en equilibrio, si y sólo si, se anula el trabajo del conjunto de fuerzas aplicadas sobre dicho sistema, para cualquier conjunto de desplazamientos virtuales compatibles con las ligaduras.*

El principio de D'Alembert es la extensión del principio de los trabajos virtuales (que se refiere a sistemas estáticos) a sistemas dinámicos.

La segunda ley de Newton establece que:

$$F_i = \dot{\mathbf{p}}_i \quad (1.12)$$

de donde se tiene que:

$$\phi_i = F_i - \dot{\mathbf{p}}_i = 0 \quad (1.13)$$

Si cada  $i$ -ésima partícula estuviera sometida a una fuerza neta dada por  $\phi_i$  el sistema estaría en equilibrio estático instantáneamente. Desde este punto de vista, la dinámica se reduce a la estática. La fuerza  $\phi_i$  debe satisfacer el principio de los trabajos virtuales.

$$\sum_{i=1}^n \phi_i \bullet \delta r_i = \sum_{i=1}^n (F_i - \dot{\mathbf{p}}_i) \bullet \delta r_i = 0 \quad (1.14)$$

y de (1.10) se tiene  $F_i = F_i^a + F_i^{lig}$ , entonces:

$$\sum_{i=1}^n (F_i - \dot{\mathbf{p}}_i) \bullet \delta r_i = \sum_{i=1}^n (F_i^a - \dot{\mathbf{p}}_i) \bullet \delta r_i + \sum_{i=1}^n F_i^{lig} \bullet \delta r_i = 0 \quad (1.15)$$

Si se consideran sistemas en los que el trabajo virtual de las fuerzas de ligadura es nulo se obtiene:



$$\sum_{i=1}^n (F_i^a - \dot{\mathbf{p}}_i) \bullet \delta \mathbf{r}_i = \sum_{i=1}^n F_i^a \bullet \delta \mathbf{r}_i - \sum_{i=1}^n \dot{\mathbf{p}}_i \bullet \delta \mathbf{r}_i = 0 \quad (1.16)$$

que suele llamarse el Principio de D'Alembert, donde  $\dot{\mathbf{p}}_i = m_i \ddot{\mathbf{r}}_i$  son las fuerzas inerciales y las masas  $m_i$  no varían.

## Ecuaciones de Lagrange

Si aplicamos los desplazamientos virtuales  $\delta \mathbf{r}$  en terminos de las coordenadas generalizadas obtenemos la siguiente expresión:

$$\delta r_i = \sum_{j=1}^{\eta} \frac{\partial r_i}{\partial q_j} \delta q_j, \quad j = 1, 2, \dots, n \quad (1.17)$$

si sustituimos esta expresión en la ecuación (1.14) se tiene:

$$\sum_{i=1}^n (F_i^a - \dot{\mathbf{p}}_i) \bullet \delta \mathbf{r}_i = \sum_{i=1}^n \left( F_i \bullet \sum_{j=1}^{\eta} \frac{\partial r_i}{\partial q_j} \delta q_j \right) - \sum_{i=1}^n \left( m_i \ddot{\mathbf{r}}_i \bullet \sum_{j=1}^{\eta} \frac{\partial r_i}{\partial q_j} \delta q_j \right) = 0 \quad (1.18)$$

Ahora desarrollando los términos de la ecuación (1.18):

$$\sum_{j=1}^{\eta} \left\{ \frac{d}{dt} \left[ \frac{\partial r_i}{\partial q_j} \left( \sum_{i=1}^n T_i \right) \right] - \frac{\partial r_i}{\partial q_j} \left( \sum_{i=1}^n T_i \right) \right\} \delta q_j = \sum_{j=1}^{\eta} \left[ \frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} \right] \delta q_j \quad (1.19)$$

donde  $T_i$  es la energía cinética de cada una de las partículas del sistema y  $T = \sum_{i=1}^n T_i$  es la energía cinética total del sistema.

Sustituyendo el primer y segundo termino de la ecuación (1.18) obtenidos anteriormente en el **Principio de D'Alembert** la ecuación (1.16):

$$\begin{aligned} \sum_{i=1}^n (F_i^a - \dot{\mathbf{p}}_i) \bullet \delta \mathbf{r}_i &= \sum_{j=1}^{\eta} Q_j \delta q_j - \sum_{j=1}^{\eta} \left[ \frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} \right] \delta q_j = 0 \\ &= \sum_{j=1}^{\eta} \left[ \frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} - Q_j \right] \delta q_j = 0 \end{aligned} \quad (1.20)$$

Las fuerzas generalizadas  $Q_j$  se pueden descomponer como la suma de las fuerzas generalizadas de una función de energía potencial, de la siguiente forma:

$$Q_j^U(q_i, \dot{q}_i, t) = \frac{d}{dt} \left( \frac{\partial U}{\partial \dot{q}_j} \right) - \frac{\partial U}{\partial q_j} \quad (1.21)$$

y las no provenientes de una función de energía potencial como se muestra a continuación:

$$Q_j^{NU} = \sum_{i=1}^n F_i^{NU} \bullet \frac{\partial r_i}{\partial q_j} \quad (1.22)$$

Estos resultados los podemos sustituir en la ecuación (1.20) se obtiene:

$$\begin{aligned} \sum_{i=1}^{\eta} \left[ \frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} - Q_j^U - Q_j^{NU} \right] \delta q_j &= 0 \\ \sum_{i=1}^{\eta} \left[ \frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} - \frac{d}{dt} \left( \frac{\partial U}{\partial \dot{q}_j} \right) + \frac{\partial U}{\partial q_j} - Q_j^{NU} \right] \delta q_j &= 0 \\ \sum_{i=1}^{\eta} \left[ \frac{d}{dt} \left( \frac{\partial}{\partial \dot{q}_j} (T - U) \right) - \frac{\partial}{\partial q_j} (T - U) - Q_j^{NU} \right] \delta q_j &= 0 \\ \sum_{i=1}^{\eta} \left[ \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_j} \right) - \frac{\partial L}{\partial q_j} - Q_j^{NU} \right] \delta q_j &= 0 \end{aligned} \quad (1.23)$$

donde  $L = T - U$  es el lagrangiano o función lagrangiana.

Si se considera que todas las coordenadas generalizadas  $q_j$  son linealmente independientes, entonces todos los desplazamientos virtuales  $\delta q_j$  también lo son. Para que esto se cumpla se requiere que cada coeficiente de los términos  $\delta q_j$  sea cero por separado, resultando las ecuaciones de movimiento de Lagrange sin ligaduras como se muestra a continuación en la ecuación :

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_j} \right) - \frac{\partial L}{\partial q_j} = Q_j^{NU}, \quad j = 1, 2, \dots, \eta \quad (1.24)$$

Para sistemas que tengan ligaduras no holónomas, las fuerzas reactivas correspondientes a estas ligaduras  $Q_j^{lig}$  se suman al lado derecho de las ecuaciones (1.24). Por lo que, las ecuaciones de Lagrange quedan como a continuación se muestran:

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_j} \right) - \frac{\partial L}{\partial q_j} = Q_j^{lig} + Q_j^{NU}, \quad j = 1, 2, \dots, \eta \quad (1.25)$$

donde:

$$Q_j^{lig} = \sum_{l=1}^k \lambda_l A_{lj} \quad (1.26)$$

y los términos  $A_{lj}$  están relacionados con las  $l$  ligaduras no holónomas:

$$\sum_{j=1}^n A_{lj} \dot{q}_j + B_l, \quad l = 1, 2, \dots, k \quad (1.27)$$

## 1.2. Teorema de Tikhonov

La modelación matemática de los sistemas mecánicos consta de dos etapas principalmente: la primera consiste en la construcción de un modelo matemático representado con ecuaciones apropiadas, la segunda etapa es examinar dichas expresiones de forma analítica o numéricamente. Por lo general, se consideran modelos sencillos y se simplifican (o aproximan) las ecuaciones para su análisis.

Existen varios métodos de análisis aproximado para las ecuaciones diferenciales. Uno de ellos es la teoría de perturbaciones, que es una colección de métodos para obtener *soluciones analíticas aproximadas* de ecuaciones diferenciales en las que se involucra un parámetro pequeño. Otro método más amplio es el *análisis fraccional* como lo explican [16], quien introduce el parámetro pequeño y aplica herramientas matemáticas. Este método hace posible la obtención de ecuaciones aproximadas que describen los componentes rápido y lento del movimiento por separado. En un mismo problema existen dos tipos de tiempo: el tiempo adimensional lento y el rápido, que son denotados por  $t = \frac{T}{T_2}$  y  $\tau = \frac{T}{T_1}$ , respectivamente. Con  $\tau = \frac{t}{\mu}$ ,  $\mu = \frac{T_1}{T_2} \ll 1$ , donde  $\mu$  es un parámetro pequeño.

Los siguiente sistemas dependen del parámetro pequeño  $\epsilon$ :

$$\begin{aligned} \dot{x}_1 &= f_1, \\ \dot{x}_2 &= \epsilon f_2, \quad \epsilon \ll 1 \end{aligned} \tag{1.28}$$

$$\begin{aligned} \epsilon \dot{x}_1 &= f_1, \quad \epsilon \ll 1 \\ \dot{x}_2 &= f_2, \end{aligned} \tag{1.29}$$

En ambos sistemas  $x_1$  es la variable rápida y  $x_2$  la lenta. La normalización (procedimiento para transformar ecuaciones con variables dimensionales a expresiones con variables adimensionales) de las ecuaciones y la introducción de los parámetros pequeños forman parte de la primera etapa de análisis fraccional. En la segunda etapa, la pequeñez de los parámetros se usa para construir una solución aproximada.

Considerando los parámetros pequeños iguales a cero. Las expresiones iniciales que los presentan se llaman *ecuaciones perturbadas respecto a los parámetros pequeños* relativas a las no perturbadas que se obtienen cuando los parámetros pequeños son iguales a cero ( $\epsilon = 0$ ). Para las ecuaciones (1.28) y (1.29) sus correspondientes sistemas no perturbados son:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, \dots, t), \\ \dot{x}_2 &= 0 \end{aligned} \tag{1.30}$$

$$\begin{aligned} 0 &= f_1, \\ \dot{x}_2 &= f_2(x_1, x_2, \dots, t), \end{aligned} \quad (1.31)$$

Es natural esperar que las ecuaciones no perturbadas son, en primer lugar, mas simples que las perturbadas, y en segundo lugar, que determinan una solución que está cerca de la solución de las expresiones iniciales. La diferencia entre las soluciones del sistema perturbado y el no perturbado se estima por la norma Euclidiana:

$$\Delta = \sqrt{\sum_{i=1}^n [x_i(t, \epsilon) - x_i(t, 0)]^2}, \quad (1.32)$$

Donde  $n$  es el orden del sistema,  $x_i(t, \epsilon)$  y  $x_i(t, 0)$  son las componentes de la solución del sistema perturbado, respectivamente.

Para esta estimación es necesario establecer explícitamente el tamaño del intervalo de tiempo  $t \in I$  en el cual es válida. Se consideran 3 casos:

1. Intervalo de tiempo finito  $0 \leq t \leq t' < \infty$ .
2. Intervalo de tiempo asintóticamente grande  $0 \leq t \leq t'/\epsilon, \epsilon \rightarrow 0$ .
3. Intervalo de tiempo infinito  $0 \leq t < \infty$

para la estimación existen dos casos:

1. Cuando  $\epsilon \rightarrow 0$  la aproximación de las soluciones se da en todo el intervalo  $t \in I$ :

$$\sup_I \Delta \rightarrow 0 \quad \text{cuando } \epsilon \rightarrow 0. \quad (1.33)$$

Entonces el sistema con un parámetro pequeño se llama *regularmente perturbado*. El problema de este tipo está determinado por las ecuaciones (1.28) y (1.30) si la estimación se cumple en el intervalo de tiempo finito.

2. si la condición dada por la expresión (1.33) no satisface, entonces el sistema con un parámetro pequeño se llama *singularmente perturbado*.

Ahora considere un sistema singularmente perturbado de forma general:

$$\begin{aligned} \frac{dy}{dt} &= Y(y, z, t, \epsilon), \quad y(0) = y_0, \\ \epsilon \frac{dz}{dt} &= Z(y, z, t, \epsilon), \quad z(0) = z_0, \quad \epsilon \ll 1 \end{aligned} \quad (1.34)$$

donde  $y$  y  $z$  son vectores  $n, m$  – dimensional, respectivamente.

De las ecuaciones (1.34) se obtiene un sistema degenerado haciendo  $\epsilon = 0$  excluyendo  $z_0$ :

$$\begin{aligned} \frac{dy}{dt} &= Y(y, z, t, \epsilon), \quad y(0) = y_0, \\ 0 &= Z(y, z, t, \epsilon), \quad z(0) = z_0, \quad \epsilon \ll 1 \end{aligned} \quad (1.35)$$

A continuación se formulan las siguientes definiciones para posteriormente enunciar el teorema de Tikhonov.

**Definición 1.12** La raíz  $z = \varphi(y, t)$  de un sistema de ecuaciones:

$$Z(y, z, t, 0) = 0 \quad (1.36)$$

Se llama **aislada** en algún rango restringido de las variables  $y, t$  si las otras raíces del sistema (1.36) no existen para cada valor fijo  $y, t$  en ninguna vecindad pequeña de la raíz.

Se introduce un sistema de ecuaciones con aproximación de orden cero para movimientos rápidos. Este sistema es conocido como "adjunto", o sistema de capa límite. Para obtenerlo, (se hace la traslación al tiempo rápido) en las ecuaciones (1.34) se hace  $\tau = \frac{t}{\epsilon}$  y la condición  $\epsilon = 0$ :

$$\frac{dz}{d\tau} = Z(y, z, \tau, 0). \quad (1.37)$$

Aquí  $y, t$  (lentos) son considerados como parámetros. La transición a  $\tau$  en el lado derecho no es implementada por que para cualquier  $t$  finito cuando  $\epsilon \rightarrow 0$  se tiene que  $\tau \rightarrow \infty$ , y  $\epsilon\tau = t$ .

La raíz  $z = \varphi(y, t)$  de la ecuación (1.36) define el punto estacionario, esto es, la posición de equilibrio del sistema dado por (1.37).

**Definición 1.13** El dominio de influencia del punto estacionario  $z = \varphi(y, t)$  es conocido como el conjunto de puntos  $z^+$  tal que las soluciones del sistema adjunto, con puntos iniciales  $z^+$ , tiende a  $\varphi(y, t)$  cuando  $t \rightarrow \infty$ .

**Teorema 1.1 Tikhonov.** Considere las siguientes condiciones:

- a) Las funciones  $Y(y, z, t, \epsilon)$  y  $Z(y, z, t, \epsilon)$  son analíticas respecto a  $\epsilon$  en algún lado dominio del espacio de variables.
- b) La ecuación (1.36) tiene una raíz  $z = \varphi(y, t)$  en algún lado dominio acotado  $D$  de las variables  $y, t$  y además esta raíz es aislada.
- c) La función  $Y(y, \varphi(y, t), t, 0)$  es analítica respecto a  $y, t$ .
- d) Los puntos estacionarios  $z = \varphi(y, t)$  del sistema (1.37) son asintóticamente estables de acuerdo a Lyapunov para toda  $y, t$ , tales que la raíz de la ecuación (1.36) está definida.
- e) Las condiciones iniciales  $z_0$  están en el dominio de influencia de la raíz  $z = \varphi(y, t)$  del sistema (1.37).

Entonces si  $\epsilon > 0$  talque la única solución del sistema (1.34) existe para  $0 \leq \epsilon \leq \epsilon_0$ , y se satisfacen los siguientes límites:

$$\begin{aligned} \lim_{\epsilon \rightarrow 0} y(t, \epsilon) &= \underline{y}(t) \quad \text{para } 0 \leq t \leq t', \\ \lim_{\epsilon \rightarrow 0} z(t, \epsilon) &= \underline{z}(t) \quad \text{para } 0 \leq t \leq t', \end{aligned} \tag{1.38}$$

Aquí  $0 \leq t \leq t'$  es un intervalo de tiempo finito que se encuentra dentro del dominio  $D$ , donde la única solución del sistema (1.34) existe.

### 1.3. Clasificación de los sistemas.

Existen diversos tipos de sistemas descritos por los términos lineal, no lineal, invariantes y variantes en el tiempo. Y son clasificados de acuerdo a la forma de sus expresiones algebraicas de estado como lo menciona [17] y [18]. Si un sistema es clasificado como no lineal y variante en el tiempo, sus ecuaciones de estado son de la forma:

$$\dot{x}(t) = a(x(t), u(t), t). \tag{1.39}$$

Un sistema denominado no lineal invariante en el tiempo, la representación de sus ecuaciones de estado son de la forma:

$$\dot{x}(t) = a(x(t), u(t)). \tag{1.40}$$

Ahora si se trata de un sistema como lineal y variante en el tiempo, sus ecuaciones de estado son como se muestra en la ecuación donde  $A(t)$  y  $B(t)$  son matrices de  $n \times n$  y  $n \times m$  con elementos variantes en el tiempo.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t). \quad (1.41)$$

Y finalmente, las ecuaciones de estado que representan a un sistema lineal invariante en el tiempo tienen la siguiente estructura:

$$\dot{x}(t) = Ax(t) + Bu(t). \quad (1.42)$$

donde A y B son matrices constantes.

## 1.4. Estabilidad

### Estabilidad de Lyapunov

La teoría de estabilidad de Lyapunov es una herramienta indispensable para analizar esta propiedad de sistemas dinámicos. En esencia los teoremas de Lyapunov acerca de esto son la generalización del principio de Torricelli, y este análisis abarca también sistemas dinámicos lineales y no lineales, sin importar el orden del sistema.

Este análisis permite obtener información sobre la estabilidad del punto de equilibrio del sistema sin la necesidad de resolver la ecuación diferencial, que caracteriza a dicho sistema dinámico, ofrece una precisa caracterización de aquellas funciones que califican como *funciones de energía* en la vecindad del punto de equilibrio y la noción que decremantan a lo largo de las trayectorias del sistema dinámico, lo que significa que la derivada temporal de la energía (potencia del sistema) deber ser definida negativa, hasta alcanzar el punto de equilibrio.

El concepto de estabilidad tiene varias interpretaciones dependiendo del campo del conocimiento. sin embargo de todos los conceptos que existen de estabilidad del área de ingeniería, la que mayor ventaja ofrece es la teoría de estabilidad de Lyapunov debido a que proporciona varias propiedades matemáticas que facilitan el análisis y diseño de esquemas de control.

Esta teoría establece que para toda condición inicial  $\mathbf{x}(0) \in \mathbb{R}^n$  que se encuentra dentro del atractor o región de atracción, si el sistema tiene un estado de equilibrio asintóticamente estable, la energía acumulada del sistema dentro del dominio de atracción cae al evolucionar el tiempo, hasta alcanzar un valor mínimo en su punto de equilibrio.

El concepto de **atractor** o **región de atracción** significa que para cada solución de la trayectoria  $\mathbf{x}(t)$  que empieza suficientemente cercana a cero, a partir de  $t_0$  se aproxima al origen  $\mathbf{0}$  cuando el tiempo evoluciona, es decir  $t_0 + t \rightarrow \infty$ .

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{t}, \mathbf{x}) \quad \mathbf{x}(0) \in \mathbb{R}^n \quad \forall \quad t \geq 0 \quad (1.43)$$

Hipótesis sobre el sistema dinámico de la ecuación (1.43)

- El punto de equilibrio existe:  $\mathbf{f}(\mathbf{0}) = \mathbf{0} \in \mathbb{R}^n$ .
- La función  $\mathbf{f}(\mathbf{x}(t))$  es continua Lipschitz sobre  $\mathbb{R}^n$ .
- Existe la solución  $\mathbf{x}(t) \in \mathbb{R}^n$  y es única, continua en  $t$  y diferenciable con respecto al tiempo.
- La función  $\mathbf{f}(\mathbf{x}(t))$  es continua en  $\mathbf{x}$ , continua para toda condición inicial  $\mathbf{x}(0) \in \mathbb{R}^n$  y continuamente diferenciable en  $t$ .

Entre los conceptos básicos de la teoría de Lyapunov destacan los siguientes:

El origen de la ecuación (1.43) es un **punto de equilibrio estable** (en el sentido de Lyapunov) si para cada  $t_0 \geq 0$ , y  $\epsilon > 0$  se puede encontrar un número  $\gamma(t_0, \epsilon) > 0$  tal que:

$$\| \mathbf{x}(0) - \mathbf{0} \| < \gamma(t_0, \epsilon) \Rightarrow \| \mathbf{x}(0) - \mathbf{0} \| < \epsilon \quad (1.44)$$

donde  $\mathbf{x}(t)$  es la solución de (1.43), la cual empieza desde  $\mathbf{x}(0)$  en  $t_0$ .

El punto de equilibrio  $\mathbf{x} = \mathbf{0}$  es **atractivo** si  $\forall t_0 \in \mathbb{R}^+, \exists \eta(t_0) > 0$  tal que:

$$\| \mathbf{x}_0 \| < \eta(t_0) \Rightarrow \mathbf{x}(t_0 + t, t_0, \mathbf{x}(0)) \rightarrow \mathbf{0} \quad (1.45)$$

cuando  $t \rightarrow \infty$

## Equilibrio asintóticamente estable

El origen es un **equilibrio asintóticamente estable** de (1.43) si

- El origen es estable.
- El origen es atractivo, es decir, existe un número  $\gamma' > 0$  tal que:

$$\| \mathbf{x}(0) \| < \gamma' \Rightarrow \| \mathbf{x}(t) \| \quad (1.46)$$

cuando  $\mathbf{x}(t) \rightarrow \mathbf{0}$ .

En la figura 1.3 se ilustra el concepto de estabilidad asintótica para el caso de  $\mathbf{x}(0) \in \mathbb{R}^2$ .



### Equilibrio asintóticamente globalmente estable.

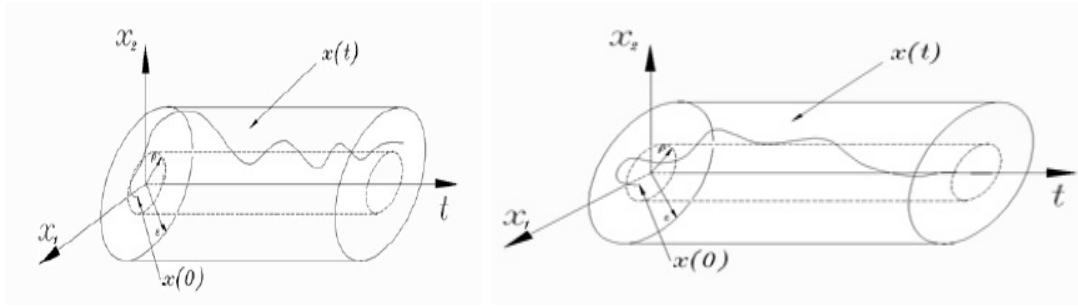
El origen es un **punto de equilibrio asintóticamente global** de la ecuación (1.43) si:

- El origen es estable.
- El origen es atractivo, para todo  $\mathbf{x}(0) \in \mathbb{R}^n$ , es decir:

$$\|\mathbf{x}(t)\| < \epsilon \forall \mathbf{x}(0) \in \mathbb{R}^n \quad (1.47)$$

cuando  $\mathbf{x}(t) \rightarrow \mathbf{0}$

En la figura 1.3 se muestra la evolución temporal de la trayectoria  $\mathbf{x}(t) \in \mathbb{R}^2$  hacia el punto de equilibrio.



(a) *Estabilidad del punto de equilibrio* (b) *Estabilidad asintótica en el punto de equilibrio*

**Figura 1.3:** Evolución en el tiempo de la estabilidad

La condición inicial se encuentra dentro de la región de atracción, entonces  $\mathbf{x}(t) \rightarrow \mathbf{0}$  cuando  $t \rightarrow \infty$ .

### Linealización de sistemas no lineales.

Considerando a  $u_0(t)$  la entrada dada para el sistema que se muestra en la siguiente ecuación diferencial:

$$\dot{x}(t) = f[x(t), u(t), t]; \quad (1.48)$$

y  $x_0(t)$  es una solución de dicha ecuación y se pueden encontrar aproximaciones alrededor de las soluciones, para pequeñas desviaciones en el estado inicial y en la entrada, con una ecuación lineal. Considerando que  $x_0(t)$  satisface que:

$$\dot{x}_0(t) = f[x_0(t), u_0(t), t], \quad t_0 \leq t \leq t_1 \quad (1.49)$$

donde  $u_0$  es una entrada nominal y  $x_0$  es una trayectoria nominal. En ocasiones se puede asumir que el sistema se encuentra operando cerca de las condiciones nominales, lo cual significa que  $u$  y  $x$  se desvían ligeramente de  $u_0$  y  $x_0$ . Por lo tanto escribiendo:

$$\begin{aligned} u(t) &= u_0(t) + \tilde{u}(t), & t_0 \leq t \leq t_1 \\ x(t_0) &= x_0(t_0) + \tilde{x}(t_0), \end{aligned} \quad (1.50)$$

donde  $\tilde{u}(t)$  y  $\tilde{x}(t)$  son pequeñas perturbaciones. Correspondientemente, introduciendo  $\tilde{x}(t)$  por la siguiente expresión

$$x(t) = x_0(t) + \tilde{x}, \quad t_0 \leq t \leq t_1, \quad (1.51)$$

Y donde sustituimos  $x$  y  $u$  en la ecuación diferencial de estado y realizando una expansión de Taylor se tiene lo siguiente:

$$\begin{aligned} \dot{x}_0(t) + \dot{\tilde{x}}(t) &= f[x_0(t), u_0(t), t] + J_x[x_0(t), u_0(t), t] \tilde{x}(t) \\ J_u[\dot{x}_0(t) + \dot{\tilde{x}}(t) &= f[x_0(t), u_0(t), t] + J_x[x_0(t), u_0(t), t] \tilde{x}(t)] \tilde{u}(t) + h(t) \\ t_0 \leq t \leq t_1 \end{aligned} \quad (1.52)$$

Aquí  $J_x$  y  $J_u$  son las matrices Jacobianas de  $f$  con respecto a  $x$  y  $u$  respectivamente, es decir,  $J_x$  es una matriz de (i-j)-ésimos elementos los cuales son:

$$(J_x)_{ij} = \frac{\partial f_i}{\partial \xi_j} \quad (1.53)$$

donde  $f_i$  es el i-ésimo componente de  $f$  y  $\xi_j$  es el j-ésimo componente de  $x$ .  $J_u$  ésta definido similarmente. El término  $h(t)$  es una expresión que se supone es "pequeña con respecto a  $\tilde{x}$  y  $\tilde{u}$ ". Despreciando  $h$ , se ve que  $\tilde{x}$  y  $\tilde{u}$  aproximadamente satisfacen la ecuación lineal:

$$\dot{\tilde{x}}(t) = A(t)\tilde{x}(t) + B(t)\tilde{u}(t) \quad t_0 \leq t \leq t_1 \quad (1.54)$$

donde  $A(t) = J_x[x_0(t), u_0(t), t]$  y  $B(t) = J_u[x_0(t), u_0(t), t]$ . Así la ecuación (1.54) es llamada ecuación diferencial de estado linealizada, con condición inicial  $\tilde{x}(t_0)$ .

El procedimiento de linealización es utilizado comúnmente en la solución de problemas de control. A menudo es mas conveniente linealizar dichas ecuaciones antes de arreglarlas en la forma diferencial de estado.

Uno de los principales propósitos en el diseño de sistemas es la estabilidad. Como ya es muy conocido, los sistemas actuales nunca son lineales, y los modelos empleados

son obtenidos por linealización. Esto significa que se diseñan sistemas cuyos modelos linealizados poseen buenas propiedades; la cuestión es: ¿Que se mantiene de estas propiedades cuando se implementa el sistema no lineal.?

**Teorema 1.2** *Considerando el sistema invariante en el tiempo con la ecuación diferencial de estado*

$$\dot{x}(t) = f[x(t)]. \quad (1.55)$$

*Se supone que el sistema tiene un estado de equilibrio  $x_c$  y que la función  $f$  posee derivadas parciales con respecto a los componentes de  $x$  en  $x_c$ . se considera que la ecuación diferencial de estado linealizada alrededor de  $x_c$  es:*

$$\dot{\tilde{x}}(t) = A\tilde{x}(t) \quad (1.56)$$

*donde la matriz constante  $A$  es el Jacobiano de  $f$  en  $x_c$ . Entonces si  $A$  es asintóticamente estable, la solución  $x(t) = x_c$  es una solución asintóticamente estable de la ecuación (1.55).*

Este teorema lleva a la siguiente conclusión, suponiendo que se está analizando un sistema inestable, y que se usan ecuaciones linealizadas para encontrar un controlador que haga al sistema linealizado de manera estable; con el teorema se puede mostrar que el sistema no lineal actual con este controlador será al menos asintóticamente estable para pequeñas desviaciones del estado de equilibrio. Y es necesario notar que el teorema solo es utilizable cuando el sistema contiene no linealidades sin problemas, si existen elementos discontinuos este teorema no ayuda.

## Estabilidad de sistemas lineales invariantes en el tiempo.

Considere el siguiente sistema, el cual ayudara para establecer las condiciones que son necesarias para que los sistemas lineales invariantes en el tiempo presenten algunas de las formas de estabilidad mencionadas.

$$\dot{x}(t) = Ax(t) \quad (1.57)$$

donde  $A$  es una matriz constante de  $n \times n$ . Si  $A$  tiene  $n$  distintos valores característicos  $\lambda_1, \lambda_2, \dots, \lambda_n$  y correspondientes vectores característicos  $e_1, e_2, \dots, e_n$ . La estabilidad del sistema es determinada por los valores característicos<sup>1</sup>  $\lambda_i, i = 1, 2, \dots, n$ .

---

<sup>1</sup>Los valores característicos de  $A$  son las raíces del polinomio característico  $\det(\lambda I - A)$ .

**Teorema 1.3** *El sistema lineal invariante en el tiempo dado por:*

$$\dot{x}(t) = Ax(t) \quad (1.58)$$

*es estable en el sentido de Lyapunov si y solo si:*

- *Todos los valores característicos de  $A$  tienen partes reales no positivas.*
- *Para cualquier valor característico sobre el eje imaginario con multiplicidad  $m$  corresponde exactamente  $m$  vectores característicos de la matriz  $A$*

Esta condición siempre es cumplida si  $A$  no tiene múltiples valores característicos sobre el eje imaginario.

**Teorema 1.4** *EL sistema invariante en el tiempo*

$$\dot{x}(t) = Ax(t) \quad (1.59)$$

*es asintóticamente estable si y solo si todos los valores característicos de  $A$  tienen estrictamente partes reales negativas.*

Si un sistema lineal invariante en el tiempo es asintóticamente estable, la convergencia del estado al estado cero es exponencial.

**Teorema 1.5** *El sistema invariante en el tiempo*

$$\dot{x}(t) = Ax(t) \quad (1.60)$$

*es estable exponencialmente si, y sólo si, es asintóticamente estable.*

Debido a que la matriz  $A$  es la que determinar si el sistema invariante en el tiempo es estable asintóticamente, es necesario recordar que:

**Definición 1.14** *La matriz constante  $A$  de  $n \times n$  es asintóticamente estable si todos los valores característicos tienen estrictamente partes reales negativas.*

Y algo que es importante mencionar a lo anterior:

**Teorema 1.6** *Si existe por lo menos un valor característico de  $A$  con parte real positiva, entonces el sistema invariante en el tiempo es inestable.*

## 1.5. Controlabilidad.

**Definición 1.15** *El sistema lineal con la ecuación diferencial de estado que se muestra en la siguiente ecuación:*

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (1.61)$$

donde  $x(t) \in \mathbb{R}^{n \times 1}$  es un vector de estados,  $u(t) \in \mathbb{R}^{r \times 1}$  es el vector de entrada,  $A \in \mathbb{R}^{n \times m}$  son matrices de coeficientes constantes; y se dice que es completamente controlable si el estado del sistema puede ser transferido desde el estado cero en cualquier tiempo inicial  $t_0$  a uno deseado cualquiera  $x(t_1) = x_1$  en un tiempo finito  $t_1 - t_0$ .

Así, cuando se diga que el sistema no puede ser transferido de un estado a otro significa que existe una entrada de control continua que por intervalos  $u(t)$  con  $t_0 \leq t \leq t_1$ , el cual lleva al sistema de un estado a otro.

La definición 1.15 parece tener algunas limitantes, debido a que el único requisito es, que el sistema pueda ser transferido desde el estado cero a cualquier otro estado. Sin embargo, esta definición implica más. La respuesta de un estado arbitrario inicial esta dado por:

$$x(t_1) = \phi(t_1, t_0)x(t_0) + \int_{t_0}^{t_1} \phi(t_1, \tau)B(\tau)u(\tau)d\tau^1 \quad (1.62)$$

por lo tanto podemos decir que:

$$x(t_1) - \phi(t_1, t_0)x(t_0) = \int_{t_0}^{t_1} \phi(t_1, \tau)B(\tau)u(\tau)d\tau \quad (1.63)$$

Esto muestra que la transferencia del sistema de un estado  $x(t_0) = x_0$  a  $x(t_1) = x_1$  es realizado por la misma entrada que transfiere a  $x(t_0) = 0$  al estado  $x(t_1) = x_1 - \phi(t_1, t_0)x_0$ , explicando lo siguiente:

**Teorema 1.7** *Dado el siguiente sistema diferencial:*

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (1.64)$$

---

<sup>1</sup>Es la solución general para el sistema diferencial (1.42) para todo  $t$ , donde  $\phi(t_1, t_0)$  denota la matriz de transición. Considerando el sistema homogéneo  $\dot{x}(t) = Ax(t)$ ; este siempre puede tener una solución que puede ser expresada como  $x(t) = \phi(t, t_0)x(t_0)$  donde la matriz de transición  $\phi(t, t_0) = e^{A(t-t_0)}$ .

*es completamente controlable si y solo si este puede ser transferido desde cualquier estado inicial  $x_0$  sin importa el tiempo  $t_0$  a algún estado deseado  $x(t_1) = x_1$  en un tiempo finito  $t_1 - t_0$ ; a través de alguna entrada de control  $u(t)$  no restringido.*

Si una de las variables de estado es independiente de control  $u(t)$ , no hay forma de dirigir esta variable de estado a cualquier otro deseado en un tiempo finito por medio del control, por lo tanto, se dice que este estado es **no controlable**, y por consecuencia el sistema es no completamente controlable, o simplemente es no controlable, mientras existe por lo menos un estado no controlable, de acuerdo con [19].

### Controlabilidad de sistemas lineales invariantes en el tiempo.

De acuerdo con [19], podemos describir la controlabilidad de los sistemas empleando lo siguiente:

**Teorema 1.8** *Sea el siguiente sistema expresado por la ecuación diferencial:*

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (1.65)$$

*para que sea **completamente controlable**, es necesario y suficiente que la **matriz de controlabilidad** de dimensiones  $n \times nr$  sea de rango  $n$ , es decir, sea de rango completo, o bien:*

$$\text{rang} \{B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B\} = n \quad (1.66)$$

La controlabilidad del sistema diferencial (1.42), es por supuesto, determinada completamente por las matrices  $A$  y  $B$ ; debido a esto es conveniente introducir la siguiente definición.

**Definición 1.16** *Sea la matriz  $A$  de dimensión  $n \times n$  y  $B$  de  $n \times m$ . Se dice que el par  $A, B$  es completamente controlable si el sistema  $\dot{x}(t) = Ax(t) + Bu(t)$  es completamente controlable.*

### Controlabilidad de sistemas lineales variantes en el tiempo.

Para sistemas lineales variantes en el tiempo la investigación del problema de control y estabilidad es más complicada que en los sistemas lineales invariantes en el tiempo. Para poder describir la controlabilidad de estos sistemas, primero se establecerá, una

necesaria y suficiente condición, y es que la matriz Gramiana asociada sea invertible o no singular. Para esto tenemos que:

**Teorema 1.9** *Considere la siguiente ecuación homogénea:*

$$\dot{x}(t) = Ax(t) \quad (1.67)$$

*Entonces si  $A(t)$  es continua para todo  $t$ , el sistema descrito por la ecuación (1.67) siempre tiene una solución la cual puede ser expresada como:*

$$x(t) = \phi(t, t_0)x(t_0), \text{ para todo } t. \quad (1.68)$$

*La **matriz de transición**  $\phi(t, t_0)$  es la solución de la ecuación diferencial matricial:*

$$\begin{aligned} \frac{d}{dt}\phi(t, t_0) &= A(t)\phi(t, t_0), \text{ para todo } t, \\ \phi(t, t_0) &= I \end{aligned} \quad (1.69)$$

*donde  $I$  es la matriz identidad, y con lo anterior podemos decir lo siguiente:*

**Teorema 1.10** *Considerando el sistema lineal variante en el tiempo con ecuación diferencial de estado:*

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (1.70)$$

*Definiendo la función matricial simétrica definida no negativa:*

$$W_c(t, t_0) = \int_{t_0}^t \phi(t, \tau)B(\tau)B^T(\tau)\phi^T(t, \tau)d\tau \quad (1.71)$$

*donde  $\phi(t, t_0)$  es la matriz de transición del sistema. Entonces el sistema es **completamente controlable** si y solo si existe para todo  $t_0$  a  $t_1$  con  $t_0 < t_1 < \infty$  tal que  $W(t_0, t_1)$  sea invertible y no singular.*

Aunque el teorema anterior es fuerte en teoría, en la practica está completamente limitado debido a el cálculo de la matriz Gramiana de controlabilidad requiere conocimiento explícito de la matriz de transición, pero para sistemas variante en el tiempo es generalmente desconocida y puede ser difícil de aproximar en algunos casos. Esta observación motiva a la siguiente condición de suficiencia para controlabilidad de acuerdo con [20].

**Teorema 1.11** *Asumiendo que las funciones matriciales  $A(t)$ ,  $B(t)$  son analíticas en  $\mathbb{R}^+$  el siguiente sistema:*

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (1.72)$$

*es completamente controlable en algún tiempo finito si:*

$$\text{rang}(M_1(t_0) \ M_2(t_0) \ \dots \ M_k(t_0)) = n, \text{ para todo } t_0 > 0 \quad (1.73)$$

*donde:*

$$\begin{aligned} M_1(t_0) &= B(t), \\ M_k(t) &= -A(t) + \frac{d}{dt}M_{k-1}(t), \ k=2, \dots, n-1 \end{aligned} \quad (1.74)$$

## 1.6. Estabilización.

Para la realización de éste trabajo, sólo es requerida la mención acerca de la estabilización de sistemas lineales invariantes en el tiempo; si lo requiere, información sobre estabilización de sistemas variantes en el tiempo puede consultarse en [20].

**Definición 1.17** *El par  $A, B$  es estabilizable si el sistema dado  $\dot{x}(t) = Ax(t) + Bu(t)$  es estabilizable.*

Con lo que obtenemos el siguiente resultado:

**Teorema 1.12** *Cualquier sistema invariante en el tiempo estable asintóticamente es estabilizable. Cualquier sistema completamente controlable es estabilizable.*

Y podemos llegar a lo siguiente:

**Teorema 1.13** *Si el par  $A, B$  del sistema diferencial  $\dot{x}(t) = Ax(t) + Bu(t)$  es controlable, entonces  $A, B$  son estabilizables.*

y por ultimo:

**Teorema 1.14** *El sistema  $\dot{x}(t) = Ax(t) + Bu(t)$  es llamado estabilizable si existe una matriz  $F$ , tal que bajo la acción de un control de la forma  $u = Fu$ , el sistema presenta un comportamiento estable.*



## Estabilidad de los sistemas de control con retroalimentación.

Un aspecto importante del diseño de sistemas retroalimentados es la estabilidad del sistema de control. Lo que se quiere obtener, es que la estabilidad sea confiable. Algunas veces, el principal objetivo de un diseño de retroalimentación es en efecto estabilizar un sistema si es inicialmente inestable, o mejorar la estabilidad si fenómenos transitorios no desaparecen lo suficientemente rápido. Ahora se investigará como se pueden mejorar las propiedades de estabilidad de sistemas lineales con retroalimentación.

Considerando el sistema lineal variante en el tiempo con ecuación diferencial de estado dado por:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (1.75)$$

Si se supone que el estado completo puede ser medido con precisión en todo tiempo, es posible implementar una **ley de control lineal** de la forma:

$$u(t) = -F(t)x(t) + u'(t) \quad (1.76)$$

donde  $F(t)$  es una *matriz de ganancias de retroalimentación* variante en el tiempo y  $u'(t)$  es una nueva entrada. Si esta ley de control es conectada al sistema (1.75) el sistema en lazo cerrado esta descrito por la ecuación diferencial de estado:

$$\dot{x}(t) = [A(t) - B(t)F(t)]x(t) + B(t)u'(t) \quad (1.77)$$

La estabilidad del sistema depende por supuesto del comportamiento de  $A(t)$  y  $B(t)$  pero además de esto, también de la matriz de ganancias  $F(t)$ . Es conveniente introducir lo siguiente:

**Definición 1.18** *La ley de control lineal :*

$$u(t) = -F(t)x(t) + u'(t) \quad (1.78)$$

*es llamada una **ley de control estable asintóticamente** para el sistema dado por la ecuación (1.75), si el sistema en lazo cerrado  $\dot{x}(t) = [A(t) - B(t)F(t)]x(t) + B(t)u'(t)$  es estable asintóticamente*

Si el sistema representado por la ecuación (1.75) es *invariante el tiempo*, y se elige una matriz constante  $F$ , la estabilidad de la ley de control descrita por la expresión (1.78), es determinada por los valores característicos de la matriz  $A - BF$ , ahora se determinara que bajo una ligera restrictiva condición, todos los valores característicos del lazo cerrado pueden ser arbitrariamente colocados en el plano complejo escogiendo una  $F$  adecuada. Si todos los polos del lazo cerrado son colocados en el semiplano izquierdo, el sistema es por supuesto asintóticamente estable.

**Teorema 1.15** *Considerando el sistema lineal invariante en el tiempo descrito por:*

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (1.79)$$

*con la ley de control invariante en el tiempo dado por:*

$$u(t) = -Fx(t) + u'(t) \quad (1.80)$$

*Entonces, los valores característicos del lazo cerrado, es decir, de  $A - BF$ , pueden ser arbitrariamente colocados en el plano complejo (con la restricción que los valores característicos complejos ocurren en pares complejos conjugados) escogiendo una  $F$  adecuada si y sólo si el sistema (1.79) es completamente controlable.*

A través del teorema 1.15 siempre es posible estabilizar un sistema completamente controlable mediante la retroalimentación, o mejorar su estabilidad por la asignación de los polos del lazo cerrado en el semiplano izquierdo complejo. El teorema sin embargo, no da orientación de en qué lugar del semiplano complejo deben ser ubicados los polos del lazo cerrado. Incluso, ocurre más incertidumbre en el caso de múltiples entradas cuando la misma configuración de polos del lazo cerrado puede ser obtenida por varias leyes de control; esta incertidumbre es eliminada con la teoría del control óptimo lineal.

Este teorema implica que siempre es posible estabilizar un sistema lineal completamente controlable. Suponiendo, sin embargo, que se está tratando con un sistema invariante en el tiempo que no es completamente controlable, se debe considerar lo siguiente.

**Teorema 1.16** *Considerando el sistema lineal invariante en el tiempo, de la siguiente expresión:*

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (1.81)$$

*con la ley de control invariante en el tiempo  $u(t) = -Fx(t) + u'(t)$ , entonces es posible encontrar una matriz constante  $F$  de tal manera que en lazo cerrado sea estable asintpticamente, si y solo si el sistema (1.81) sea estabilizable.*

## 1.7. Programación dinámica.

Uno de los principales métodos de síntesis óptima es la programación dinámica, que fue introducida por primera vez por R. Bellman. El término síntesis óptima significará: determinar el control como una función de las variables de fase, tal que minimice algún funcional especificado.

EL enfoque de la programación dinámica está destinado para sistemas dinámicos de tipo Markov, el futuro es determinado completamente por el presente y no por el pasado. Un caso especial de los sistemas de Markov son los sistemas dinámicos controlables, cuyo movimiento está determinado por algunas ecuaciones diferenciales o en diferencias sin demora en el tiempo (es decir, sin retraso).

La programación dinámica está basada en el siguiente principio de optimalidad:

*Definición 1.19 Independientemente del estado y el control inicial, el control posterior tiene que ser óptimo en relación con el estado alcanzado por el control anterior.*

En otras palabras, el control óptimo no depende de la historia, además está totalmente determinado por el estado del sistema en el momento actual y el costo de control, especificado por algún funcional adicional.

A diferencia del Principio del Máximo, la programación dinámica fue desarrollada primero para sistemas discretos. Este método permite determinar el mínimo global para un conjunto de control dado.

Para problemas de control óptimo, de sistemas dinámicos controlables, el principio de optimalidad afirma que: para cualquier instante de tiempo  $\tilde{t} \in [t_0, t_1]$ , dividiendo la trayectoria óptima en dos partes, la segunda parte determina una trayectoria óptima.

### Procedimiento de Síntesis

Dada la dinámica de un proceso controlable, descrita por la ecuación de estado no lineal siguiente:

$$\dot{x} = f(x_t, u(t), t) \quad (1.82)$$

Hallar una ley de control de retroalimentación de estado  $u(x(t), t)$ , tal que el criterio de desempeño para el comportamiento general del sistema dado:

$$J((x_t, u(t), t) = h((x_{t_1}, t_1) + \int_t^{t_1} f_0(x_\tau, u(\tau), \tau) d\tau \leq t_1 \quad (1.83)$$

donde  $x_t$ , es cualquier estado admisible. Ahora, el problema es de terminar todos los controles que minimicen la ecuación anterior, para todas las  $x(t)$  admisibles y para todo  $t \leq t_1$ .

Asumiendo que existe una función de costo mínimo  $w(x(t), t)$ , llamada función de Bellman, donde:

$$\omega(x(t), t) = \min_u \left\{ \int_t^{t_1} f_0(x(\tau), u(\tau), \tau) d\tau + h(x(t_1), t_1) \right\} \quad t \leq t_1 \quad (1.84)$$

Partiendo en dos el intervalo de tiempo se tiene

$$\omega(x(t), t) = \min_u \left\{ \int_t^{t+\Delta t} f_0 d\tau + \int_{t+\Delta t}^{t_1} f_0 d\tau + h(x(t_1), t_1) \right\}, \quad t \leq t_1 \quad (1.85)$$

Usando el principio de optimalidad se tiene

$$\omega(x(t), t) = \min_u \left\{ \int_t^{t+\Delta t} f_0 d\tau + \omega(x(t+\Delta t), t+\Delta t) \right\}, \quad t \leq t+\Delta t \quad (1.86)$$

donde  $\omega(x(t+\Delta t), t+\Delta t)$  es el costo mínimo del proceso para el intervalo de tiempo  $t+\Delta t \leq \tau \leq t_1$  con estado inicial  $x(t+\Delta t)$ .

Asumiendo que las derivadas parciales de  $\omega$  existen y son acotadas, Bellman demostró que  $\omega(t, x)$  debe satisfacer la ecuación en derivadas parciales (llamada ecuación de Bellman en derivadas parciales), como se muestran a continuación:

$$-\frac{\partial \omega(x(t), t)}{\partial t} = \min_u \left\{ f_0(x, u, t) + \left( \frac{\partial \omega}{\partial x} \right)^T f(x, u, t) \right\} \quad (1.87)$$

para hallar las condiciones de frontera de esta ecuación diferencial en derivadas parciales, se toma  $t = t_1$  en la ecuación (1.84), para así obtener:

$$\omega(t_1, x(t_1)) = h(x(t_1), t_1). \quad (1.88)$$

El procedimiento de síntesis óptima se resume en los siguientes puntos:

- Formular la ecuación de Bellman (1.87) para el problema dado.
- Determinar un control  $u^0$ , que minimice la parte derecha de la ecuación de Bellman y escribirlo de la forma:

$$u^0 = u^0 \left( x \frac{\partial \omega}{\partial x} \right). \quad (1.89)$$

- Sustituir  $u^0$  en la ecuación de Bellman para obtener, la ecuación diferencial no lineal de primer orden, en derivadas parciales en términos de  $\omega(t, x)$ , de la forma:

$$-\omega_t(x(t), t) = f_0(x, u^0(x, \omega_x)) + (\omega_x)^T f(x, u^0(x, \omega_x)) \quad (1.90)$$

con las condiciones de frontera  $\omega(t_1, x(t_1)) = h(x(t_1), t_1)$

- Determinar la solución  $\omega(t, x)$  de (1.90).
- Terminar el procedimiento de síntesis óptima sustituyendo la solución  $\omega(t, x)$  en la expresión del control óptimo  $u^0$ .

La mayor dificultad en la aplicación del procedimiento anterior surge al resolver el problema con valor en la frontera, de acuerdo con [21].

### Problema de Control Lineal Cuadrático.

En el caso especial del problema lineal cuadrático en control óptimo puede ser sintetizado. Sea el sistema dinámico controlable invariante en el tiempo, gobernado por la ecuación siguiente:

$$\dot{x}Ax + Bu \quad (1.91)$$

donde  $A$  es una matriz cuadrada,  $B$  es un vector columna,  $u$  es un control escalar de estrada. El estado inicial  $x(0) = x_0$  y el intervalo de tiempo  $[0, t_1]$  son fijos, el estado terminal  $x(t_1)$  es sin restricciones. La funcional de costo es especificada como:

$$J = \frac{1}{2}(x(t_1))^T Q x(t_1) + \frac{1}{2} \int_0^{t_1} [x^T N x + r^2 u^2] dt, \quad (1.92)$$

donde  $Q, N$  son matrices constantes simétricas no negativas,  $r$  es un escalar. El problema es sintetizar una ley de control óptima. La ecuación de Bellman puede ser escrita como:

$$\begin{aligned} -\frac{\partial \omega}{\partial t} &= \min_u \left\{ \frac{1}{2} x^T N x + \frac{1}{2} r^2 u^2 + (Ax)^T \frac{\partial \omega}{\partial x} + (Bu)^T \frac{\partial \omega}{\partial x} \right\} \\ \omega(t_1) &= \frac{1}{2} (x(t_1))^T Q x(t_1). \end{aligned} \quad (1.93)$$

De acuerdo al procedimiento anterior, primero se encuentra un control  $u^0$  que minimice el lado derecho de la ecuación (1.93). Misma puede ser transformada a la forma:

$$\frac{\partial \omega}{\partial t} + \min_u \left\{ \frac{1}{2} x^T N x + \frac{1}{2} r^2 \left[ u^2 + \frac{2}{r^2} B^T \frac{\partial \omega}{\partial x} u + \frac{1}{r^4} \left( B^T \frac{\partial \omega}{\partial x} \right)^2 \right] - \frac{1}{2r^2} \left( B^T \frac{\partial \omega}{\partial x} \right)^2 + (Ax)^T \frac{\partial \omega}{\partial x} \right\} = 0. \quad (1.94)$$

El término entre corchetes es minimizado por:

$$u^0 = -\frac{1}{r^2} B^T \frac{\partial \omega}{\partial x} \quad (1.95)$$

Si realiza la sustitución de  $u^0$ , se puede reescribir la ecuación de Bellman como se muestra a continuación:

$$\begin{aligned} \frac{\partial \omega}{\partial x} + \frac{1}{2} x^T N x + (Ax)^T \frac{\partial \omega}{\partial x} - \frac{1}{2r^2} \left( B^T \frac{\partial \omega}{\partial x} \right)^2 &= 0 \\ \omega(t_1) &= \frac{1}{2} (x(t_1))^T Q x(t_1). \end{aligned} \quad (1.96)$$

Buscando una solución para la ecuación anterior como una forma cuadrática tenemos:

$$\omega(t, x) = \frac{1}{2} x^T \mathcal{L}(t) x \quad (1.97)$$

Donde  $\mathcal{L}(t)$  es alguna matriz simétrica aún desconocida. Se puede escribir de la siguiente manera:

$$\frac{\partial \omega}{\partial t} = \frac{1}{2} x^T \dot{\mathcal{L}} x, \quad \frac{\partial \omega}{\partial x} = \mathcal{L}(t) x, \quad u^0 = -\frac{1}{r^2} B^T \mathcal{L}(t) x. \quad (1.98)$$

Sustituyendo las ecuaciones (1.97), (1.98) en (1.96) y así obtener el siguiente problema con valor inicial en  $\mathcal{L}(t)$ :

$$\frac{1}{2} x^T \dot{\mathcal{L}}(t) x + \frac{1}{2} x^T N x + (Ax)^T \mathcal{L}(t) x - \frac{1}{2r^2} (B^T \mathcal{L}(t) x)^2 = 0 \quad (1.99)$$

o equivalente si:

$$\frac{1}{2} x^T \left( \dot{\mathcal{L}} + N + \mathcal{L} A + A^T \mathcal{L} - \frac{1}{2} \mathcal{L} B B^T \mathcal{L} \right) x = 0 \quad (1.100)$$

Lo anterior es verdadero, si la matriz  $\mathcal{L}(t)$  satisface la siguiente expresión:

$$\begin{aligned} \dot{\mathcal{L}} + N + \mathcal{L} A + A^T \mathcal{L} - \frac{1}{r^2} \mathcal{L} B B^T \mathcal{L} &= 0 \\ \mathcal{L}(t_1) &= Q \end{aligned} \quad (1.101)$$

Esta expresión (1.101) se conoce como la ecuación matricial de Ricatti. En dimensiones mayores usualmente esta ecuación se resuelve numéricamente.

## 1.8. Ecuaciones en Desviaciones y el Problema de Síntesis Lineal

Suponiendo que están definido algún movimiento deseado  $y^d(t), t \in [t_0, t_1], t_0 < t_1 \leq \infty$ , y algun control deseado de salida  $u^d(t)$  que satisfacen las ecuaciones siguientes:

$$\begin{aligned} \dot{\mathbf{y}} &= f(\mathbf{y}^d(t), \mathbf{u}^d(t)), \quad t \in [t_0, t_1]. \\ \mathbf{u}^d(\cdot) &\in \mathbf{W}, \end{aligned} \quad (1.102)$$

donde  $\mathbf{W}$  es un conjunto funcional que abarca las restricciones del control (es común considerar a  $\mathbf{W}$  como un conjunto de funciones continuas a trozos.) A la triada  $\mathbf{y}^d(\cdot), \mathbf{u}^d(\cdot), [t_0, t_1]$  se le llama *proceso (programa) deseado de control*. El proceso de control deseado se puede plantear como un problema de control de retroalimentación, misma que requiere de información sobre las desviaciones del estado actual  $\mathbf{y}(t)$  con respecto al deseado  $\mathbf{y}^d(t)$ . Suponiendo que se tienen  $m$  sensores que entregan datos sobre el movimiento actual. Despues de procesar dichos datos se pueden estimar las desviaciones actuales  $\mathbf{x}(t) = \mathbf{y}(t) - \mathbf{y}^d(t)$  y se puede formar el control *actuador*. Las ecuaciones en diferencias que gobiernan las desviaciones actuales  $\mathbf{x}(t) = \mathbf{y}(t) - \mathbf{y}^d(t)$ , para  $\mathbf{u}(t) = \mathbf{u}^d(t) \quad t \in [t_0, t_1]$  pueden escribirse como:

$$\dot{\mathbf{x}} = \dot{\mathbf{y}} - \dot{\mathbf{y}}^d = f(\mathbf{y}, \mathbf{u}^d) - f(\mathbf{y}^d, \mathbf{u}^d) = f(\mathbf{y} - \mathbf{y}^d + \mathbf{y}^d, \mathbf{u}^d) - f(\mathbf{y}^d, \mathbf{u}^d) = \tilde{f}(\mathbf{x}, t) \quad (1.103)$$

donde  $\tilde{f}(0, t) \equiv 0$  para  $t \in [t_0, t_1]$  y  $\mathbf{x}(t_0) \neq 0$ . Las ecuaciones (1.103) admiten una solución trivial, correspondiente al movimiento deseado  $\mathbf{y}^d$  del sistema. Una fundamentación teórica, para el análisis del comportamiento de las desviaciones la proporciona la teoría de estabilidad, desarrollada por A. M. Lyapunov en el siglo XIX.

Restringiéndose al importante casi especial de *movimientos estacionarios deseados*  $\mathbf{y}^d(t) = \mathbf{y}^e$  y  $t_1 = \infty$ . Se define primeramente una *coordenada cíclica*  $y_i, i = 1, \dots, k; 0 \leq k < n$ , de tal forma que:

$$\frac{\partial f(\mathbf{y}, \mathbf{u})}{\partial y_i} = 0, \quad \frac{\partial f(\mathbf{y}, \mathbf{u})}{\partial u} = 0 \quad (1.104)$$

Un movimiento  $\mathbf{y}(t)$  se denomina estacionario y se denota por  $\mathbf{y}^e$ , si  $\dot{y}_i^e \equiv \text{constante}$ ,  $i = 1, 2, \dots, k$   $\dot{y}_j^e \equiv 0, j = k + 1, \dots, n$  donde  $y_1, \dots, y_k$  son las coordenadas cíclicas. Para  $K = 0$  el sistema no tiene, y como se considera en la ecuación (1.102), el movimiento deseado es factible si se cumplen las siguientes  $k$  ecuaciones diferenciales y las  $n - k$  identidades:

$$\dot{\mathbf{y}}_i^e = f_i(y_{k+1}^e, \dots, y_n^e), \quad i = 1, \dots, k \quad (1.105)$$

$$f_j(y_{k+1}^e, \dots, y_n^e, u_1, \dots, u_s) = 0, \quad j = k+1, \dots, n \quad (1.106)$$

Suponiendo que las  $n - k$  ecuaciones de la expresión (1.106) con  $s$  incógnitas  $u_1, \dots, u_s$  admiten al menos una solución  $u_1^e, \dots, u_s^e$ . Entonces el *proceso estacionario deseado de control*  $\{\mathbf{y}^e(t), \mathbf{u}^e, [t_0, \infty)\}$  está definido por la triada  $\{\mathbf{y}^d(t), \mathbf{u}^d, [t_0, t_1]\}$ , donde  $\mathbf{y}^d(t) = \mathbf{y}^e$ ,  $\mathbf{u}^d(t) = \mathbf{u}^e$  es una solución de (1.106) y  $t_1 = \infty$ .

**Observación 1.1** *Para un proceso de control estacionario el miembro derecho de la ecuación (1.103) no depende explícitamente de  $t$  y las ecuaciones en desviaciones toman la forma como se muestran a continuación:*

$$\dot{\mathbf{x}} = \tilde{f}(\mathbf{x}) \quad (1.107)$$

donde  $\tilde{f}(0) \equiv 0$  y  $\mathbf{x} \neq 0$ .

De manera que el análisis de las desviaciones de la solución estacionaria conduce al análisis de la estabilidad de la solución trivial del sistema no lineal de la ecuación (1.107)

**Definición 1.20** *La solución trivial  $\dot{\mathbf{x}} \equiv 0$  de la ecuación (1.107) se denomina estable en el sentido de Lyapunov, si para cualquier  $\epsilon$  positivo existe un  $\delta = \delta(\epsilon) > 0$ , tal que la desigualdad  $|\mathbf{x}_{t_0}| \leq \delta$  da lugar a  $|\mathbf{x}(t)| \leq \epsilon$  para cualquier  $t > t_0$ .*

**Definición 1.21** *La solución trivial se denomina asintóticamente estable si lo es también en el sentido de Lyapunov como se ha mencionada anteriormente, y además, existe una vecindad abierta de cero  $X_0 \subset \mathbb{R}^n$  tal que:*

$$\lim_{t \rightarrow \infty} \mathbf{x}(t) = 0, \text{ para todo } \mathbf{x}(t) \in X_0 \quad (1.108)$$

En este análisis de estabilidad son de importancia fundamental las ecuaciones lineales de desviaciones:

$$\dot{\mathbf{x}} = A\mathbf{x}, \quad \text{donde} \quad A_{n \times n} = \frac{\partial \tilde{f}(0)}{\partial \mathbf{x}} = \frac{\partial \tilde{f}(\mathbf{y}^e, \mathbf{u}^e)}{\partial \mathbf{y}} \quad (1.109)$$



es una matriz constante. Denotemos con  $\lambda_i$  las raíces del polinomio característico  $\det(\lambda E_n - A) = 0$  de la matriz  $A$ , como lo muestran [22].

**Teorema 1.17** *Sobre estabilidad a primera aproximación Si todos los eigenvalores  $\lambda_i$  de  $A$  están en el semiplano izquierdo del plano complejo es decir que  $\text{Re}\lambda_i < 0, i = 1, 2, \dots, n$ , entonces la solución trivial de la ecuación (1.107) es asintóticamente estable.*

**Teorema 1.18** *Si existe por lo menos un eigenvalor de  $A$  con parte real positiva, entonces la solución trivial es inestable.*

**Definición 1.22** *La solución trivial se denomina exponencialmente estable, si existen  $\alpha$  y  $\beta$ , tales que para cualquier  $x(t_0) \in X_0, t \in [t_0, t_1)$  se cumple la desigualdad:*

$$|x(t)| \leq \beta |x(t_0)| e^{-\alpha(t-t_0)}. \quad (1.110)$$

Si la velocidad  $X_0$  es igual a  $\mathbb{R}^n$  se dice que la estabilidad es global y exponencialmente estable.

Ahora se dará la formulación formal del problema de la síntesis con retroalimentación lineal, para el caso cuando dicho conjunto funcional  $W$  es de funciones continuas por tramos, cuyos valores están en un conjunto cerrado  $\Omega$  en el espacio de control  $\mathbb{R}^s$ :

$$W = \{u(\cdot) \in KC | u(t) \in \Omega \subset \mathbb{R}^s, \quad t \in [t_0, t_1)\}. \quad (1.111)$$

Suponiendo que  $u^d(t) \in \Omega$  para  $t \in [t_0, t_1)$ . El significado físico de esto último es que los actuadores tienen algunos recursos adicionales que les permiten reducir las desviaciones presentes. Considerando que se da la estrategia de control entonces:

$$u = u^d(t) + \tilde{u}(z, t), \quad (1.112)$$

donde  $\tilde{u}$  es algún control adicional. Se introducen las *ecuaciones lineales de desviaciones* dadas por:

$$\dot{x} = A(t)x + B(t)\tilde{u} \quad (1.113)$$

donde

$$A(t) = \frac{\partial f(y^d(t), u^d(t))}{\partial y}, \quad B(t) = \frac{\partial f(y^d(t), u^d(t))}{\partial u} \quad (1.114)$$

y el modelo lineal de mediación:

$$\tilde{z} = H(t)x + \gamma(t), \quad (1.115)$$

donde

$$H(t) = \frac{\partial \varphi(\mathbf{y}^d(t))}{\partial \mathbf{y}}, \quad \tilde{z} = \mathbf{z} - \varphi(\mathbf{y}^d(t)) \quad (1.116)$$

y  $\mathbf{z}$  es el vector que contiene los datos de medición. Lo que se requiere es hallar un operador homogéneo  $\tilde{\mathbf{u}} = \mathcal{P}[\mathbf{z}(t)]$ , tal que las desviaciones actuales satisfagan la expresión (1.110) mientras que  $\gamma \equiv 0$ . El problema que se plantea en (??) tal que las perturbaciones actuales cumplen (1.113) y (1.115) se denominan *problema de estabilización del movimiento deseado*. Este es uno de los problemas básicos de la síntesis lineal con retroalimentación.

En el caso de movimientos estacionarios  $\mathbf{y}^e, t \in [t_0, \infty)$  y funcionales  $\varphi(\mathbf{y})$  en coordenadas cíclicas, las matrices  $A, B$  y  $H$  son invariantes en el tiempo, y la complejidad del problema se reduce significativamente.

El problema de síntesis lineal se conoce como *problema de seguimiento*. El planteamiento anterior de síntesis, al ser los más simples posible, permiten una comprensión rápida de los problemas dinámicos de control típicos. Esto se justifica por el hecho de que la estrategia de control lineal (1.112) es semejante a la representación de un movimiento como una suma de movimiento transitorio y relativo, que se usa con frecuencia en mecánica teórica.

## 1.9. Desigualdades matriciales lineales (LMIs)

Las Desigualdades matriciales lineales (LMIs) se han convertido en una nueva y poderosa herramienta en el análisis y diseño de control. Con el desarrollo de nuevas técnicas en la optimización convexa, las LMIs se han convertido en una importante herramienta, con muchas aplicaciones actuales y futuras.

Gracias a la correcta implementación, los problemas de control pueden ser formulados en un espacio paramétrico convexo como problemas de optimización, permitiendo el uso de herramientas computacionales para la solución eficiente. En particular, los problemas convexos descritos por LMIs han atraído mas la atención, por la sencillez, elegancia y principalmente por la gran cantidad de métodos numéricos especializados disponibles.

Una gran ventaja de utilizar LMIs es la posibilidad de manejar varias restricciones y especificaciones de diseño, de forma numérica. Una vez descritos de ésta forma, los problemas pueden ser resueltos por algoritmos eficientes de optimización convexa también llamados solucionadores LMI. Muchos problemas que no tienen solución analítica

pueden ser resueltos por medios numéricos, con garantías de encontrar una solución, aún cuando ésta exista.

Muchos problemas de control y especificaciones de diseño tienen formulación LMI. Esto es especialmente cierto para el análisis y diseño de sistemas que utilizan el principio de Lyapunov, además es cierto también para los problemas de LQG óptimo (Linear Quadratic Gaussian) por sus siglas en ingles, control  $H_\infty$ , control  $H_2$ , control covarianza, etc. Las LMIs se aplican a los procedimientos de estimación, identificación, diseño óptimo, el diseño estructural, los problemas de escalonamiento de matrices y otros [23].

## Representación de LMIs

La idea Básica de una LMI, consiste en reformular problemas de control en términos de desigualdades matriciales lineales, ésto es, restricciones de desigualdades involucrando matrices simétricas afines en las variables de decisión. Por lo tanto, el objetivo es trabajar con problemas convexos que pueden ser resueltos de manera eficiente.

**Definición 1.23** Una LMI Linear Matrix Inequality es una desigualdad matricial de tipo  $F(g) > 0$ , en la cual  $F(g) : \mathbb{R}^m \rightarrow \mathbb{R}^{q \times q}$  es simétrica y afín a las variables de la búsqueda que son representadas por el vector  $g$ . Así, una LMI puede ser genéricamente representada de la forma:

$$F(g) = F_0 + \sum_{i=1}^m g_i F_i > 0, \text{ con: } g = \begin{pmatrix} g_1 \\ \vdots \\ g_m \end{pmatrix} \quad (1.117)$$

donde  $F_i = F_i^T \in \mathbb{R}^{q \times q}$  son matrices dadas y  $g_i$  son variables escalares a ser determinadas de forma que se satisfaga la desigualdad (en caso de ser posible). Cuando existe una solución  $g$  para  $F(g) > 0$  se dice que la LMI es factible.

Es importante enfatizar que una LMI se puede representar de varias formas y difícilmente aparece en un problema de la forma genérica como la representada por la ecuación (1.117). Por ejemplo, dada una matriz  $A$  y una matriz  $Q > 0$ , la función matricial  $F(P) = A^T P + P A + Q$ , que aparece en varios problemas de estabilidad, es afín a la variable  $P$  y por tanto la desigualdad  $F(P) < 0$  es una LMI que puede ser fácilmente reescrita en la forma (1.117), donde la  $g$  es el vector que contiene los elementos la matriz  $P$  que será determinada. Una ventaja de la representación genérica que se muestra en la ecuación (1.117) es que toda LMI puede ser reescrita de esta manera y todos los algoritmos de solución de LMIs desarrolladas para tal representación. Sin

embargo, la conversión de una LMI para una forma afín es hecha internamente en los algoritmos de solución y el usuario no tiene que preocuparse de eso.

## Complemento de Schur

Generalmente, una problema de control que se quiere resolver no aparece naturalmente en la forma de LMI. Algunas herramientas matemáticas son bastante útiles en la tarea de intentar reescribir el problema en cuestión a través de LMIs. Si la naturaleza del problema original es convexa, es muy probable que se encuentre una forma de reescribirlo a través de LMIs. El complemento de Schur es una herramienta básica y muy empleada en la manipulación de las desigualdades matriciales, como los menciona [24].

**Definición 1.24 Complemento de Schur:** Sea  $g \in \mathbb{R}^m$  un vector de variables de decisión y  $M_1, M_2, M_3$  funciones afines en  $g$  con  $M_1(g)$  y  $M_2(g)$  simétricas. Entonces las siguientes afirmaciones son equivalentes.

a)

$$M_1(g) - M_3(g)M_2^{-1}(g)M_3^T(g) > 0, \text{ con } M_2(g) > 0 \quad (1.118)$$

b)

$$\begin{bmatrix} M_1(g) & M_3(g) \\ M_3^T(g) & M_2(g) \end{bmatrix} > 0 \quad (1.119)$$

## Problema LMI: formulación y solución

Un problema LMI puede ser dividido en tres categorías según las necesidades de la solución:

- *Problema de Factibilidad.*- Encontrar una solución  $g$  tal que  $F(g) > 0$ .
- *Problema de minimización de una función objetivo lineal.*- Encontrar una solución  $g$  tal que:

$$\begin{array}{ll} \min_g & h(g) \\ \text{sujeto a} & \begin{cases} F(g) > 0 \\ G(g) = 0 \end{cases} \end{array} \quad (1.120)$$

- *Problema de minimización de valores característicos generalizados (problema de optimización cuasi convexa).*- Encontrar una solución  $g$  tal que:

$$\min \lambda \text{ sujeto a } \begin{cases} A(g) < \lambda B(g) \\ B(g) > 0 \\ C(g) < 0 \end{cases} \quad (1.121)$$

con  $h(g)$ ,  $F(g)$ ,  $G(g)$ ,  $A(g)$ ,  $B(g)$  y  $C(g)$  funciones afines en  $g$ .

Una búsqueda de solución numérica para estos problemas es un problema de Programación Semi Definida (SDP). Diversos paquetes computacionales han sido creados para soluciones de este tipo de problemas.

Con el surgimiento de algoritmos de puntos interiores para la solución de problemas de optimización convexa, es posible solucionar numéricamente LMIs de forma más rápida y eficiente. Desde entonces, muchas investigaciones han sido desarrolladas para la creación o mejora de paquetes computacionales para la solución de problemas de optimización convexa.

Uno de los paquetes mas conocidos para la solución de LMIs, es la herramienta de LMIlab que se encuentra presente en el software de MATLAB, que usa un Método Proyectivo creado por Nesterov y Nemirovskii para la solución de estos problemas como lo menciona [25].

## 1.10. Control óptimo de sistemas variables en el tiempo.

Vamos a considerar el siguiente sistema lineal variable en el tiempo:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (1.122)$$

donde  $t \in [t_0, T]$  denota la variación del tiempo,  $x(t) \in \mathbb{R}^n$  es el vector de estado,  $u(t) \in \mathbb{R}^m$  es el vector de control (entrada), y  $A(t)$  y  $B(t)$  son las matrices reales de dimensiones apropiadas; se estudiará el problema de traer ese sistema desde un estado inicial arbitrario al estado cero tan rápido como sea posible, Existen muchos criterios que expresan que tan rápido un estado inicial es llevado al estado cero; estos criterios son llamados *funcional objetivo* y dan una medida cuantitativa, del comportamiento del sistema en el tiempo.

Se consideran funcionales del tipo:

$$J = \int_{t_0}^t F[x(t), u(t)] dt + S[x(t_1)] \quad (1.123)$$

y consideramos a  $F$  y  $S$  que poseen derivadas parciales primeras y continuas. El primer sumando del funcional  $J$  es una integral que depende de los valores que van

tomando  $x(t)$  y  $u(t)$  a lo largo del horizonte temporal, y por lo tanto, valora el comportamiento del sistema a través del tiempo. El segundo sumando valora el estado en que queda el sistema al final del intervalo de tiempo que constituye el horizonte temporal del problema de acuerdo con [26].

Con lo anterior, se puede entonces considerar como parte de esta funcional el criterio de la integral cuadrática dada por:

$$\int_{t_0}^t x^T(t)Q(t)x(t)dt \quad (1.124)$$

en esta expresión  $Q(t)$  es una matriz simétrica definida no negativa, ( $Q(t) = Q(t)^T \geq 0$ ). La cantidad  $x^T(t)Q(t)x(t)$  es una medida de la extensión de la desviación de cada uno de los estados del estado cero en el tiempo  $t$ ; dada la matriz de ganancias  $Q(t)$  quien es la que determina cuanto peso le corresponde a cada uno de los componentes del estado. La integral (1.124) es un criterio para la desviación acumulativa de  $x(t)$  desde el estado cero durante el intervalo  $[t_0, t_1]$ .

Si ahora se intenta encontrar una entrada óptima para el sistema que minimice la cantidad del resultado de la expresión (1.124), generalmente se cae en la dificultad que resulten amplitudes indefinidamente grandes en la entrada. Para prevenir esto se incluye la entrada en el criterio; considerando así la expresión siguiente

$$\int_{t_0}^t [x^T(t)Q(t)x(t) + u^T(t)R(t)u(t)]dt \quad (1.125)$$

donde  $R(t)$  es una matriz de ganancia simétrica definida positiva ( $R(t) = R(t)^T \geq 0$ ). Esta inclusión del segundo termino en el criterio, reduce la amplitud de la entrada si se intenta hacer tan pequeño como sea posible el valor total de la expresión (1.125). La importancia relativa de estos dos términos en el criterio es determinado por las matrices  $Q$  y  $R$ .

Si es muy importante que el estado terminal  $x(t_1)$  este tan cerca como sea posible al estado cero, es útil algunas veces extender la ecuación (1.125) con un tercer termino como se muestra:

$$\int_{t_0}^t [x^T(t)Q(t)x(t) + u^T(t)R(t)u(t)]dt + x^T(t_1)Fx(t_1) \quad (1.126)$$

donde  $F$  es una matriz simétrica definida no negativa ( $F = F^T \geq 0$ ). Con esto, es posible introducir el problema del regulador o control optimo lineal.

**Definición 1.25** Considerando el siguiente sistema lineal variante en el tiempo:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (1.127)$$

donde  $x(t_0) = x_0$  y considerando también el criterio dado por:

$$\int_{t_0}^t [x^T(t)Q(t)x(t) + u^T(t)R(t)u(t)]dt + x^T(t_1)Fx(t_1) \quad (1.128)$$

donde  $F$  y  $Q(t)$  son matrices simétricas definidas no negativas ( $F = F^T \geq 0$ ;  $Q(t) = Q(t)^T \geq 0$ ) y  $R(t)$  es una matriz simétrica definida positiva ( $R(t) = R(t)^T \geq 0$ ) para  $t_0 \leq t \leq t_1$ . Entonces el problema de determinar un vector de entrada admisible  $u^0(t)$ ,  $t_0 \leq t \leq t_1$ , para el cual el criterio es **mínimo** es llamado **problema del regulador o control óptimo lineal**. Expresado en términos matemáticos se trata de:

$$\min_u J = \int_{t_0}^t [x^T(t)Q(t)x(t) + u^T(t)R(t)u(t)]dt + x^T(t_1)Fx(t_1) \quad (1.129)$$

sujeto a :  $\dot{x}(t) = A(t)x(t) + B(t)u(t)$

$$\text{con : } x(t_0) = x_0$$

Este problema especial de control óptimo tiene una solución en forma cerrada que puede ser calculada empleando el *Principio del Máximo de Pontryagin*, combinada con la ecuaciones de Euler-Lagrange; o con el *Método de la Programación Dinámica* desarrollado por R. E. Bellman de acuerdo con [19] y [27].

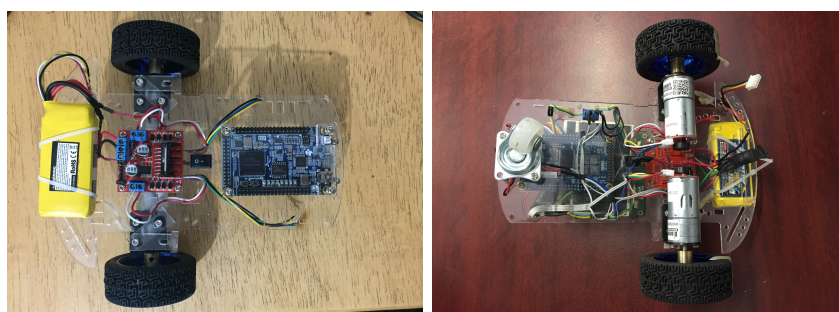




# Modelado Dinámico del Robot Líder

---

En esta sección se establecen las ecuaciones dinámicas que describen el movimiento del robot móvil que vamos a estudiar, el cual como se puede observar en la figura 2.1 cuenta con un plataforma que se mantendrá paralela al plano de trabajo, que dará soporte a dos motores que se encuentran acoplados a una rueda tanto en la parte derecha como izquierda de manera independiente, y una tercera rueda omnidireccional.



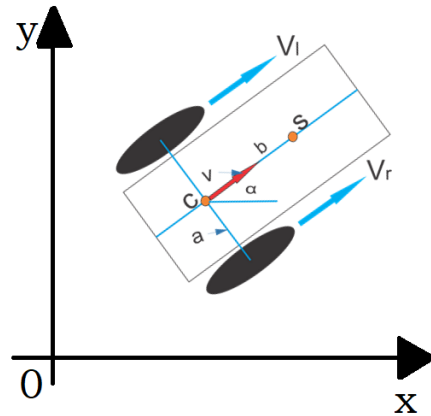
(a) *Plataforma robot móvil*

(b) *Motores independientes*

**Figura 2.1:** Prototipo inicial

## 2.1. Ecuaciones Dinámicas del Movimiento Controlable de un Robot Móvil.

Se considera la siguiente configuración de un robot móvil de acuerdo con [21] como se muestra en la figura 2.2, el cual esta constituido por dos ruedas activas alineadas por un eje, cada rueda cuenta con un motor de corriente directa y giro independiente, una tercera rueda omnidireccional que asegura que la plataforma se encuentre paralela al plano  $XY$ , para efectuar el análisis correspondiente se definen los siguientes parámetros:



**Figura 2.2:** Configuración del Robot desplazándose en el área de trabajo

- $c$  – punto medio del eje que une las dos ruedas traseras del robot.
- $s$  – centro de masa de la plataforma del robot.
- $a$  – longitud de  $c$  al centro de cualquiera de las dos ruedas activas.
- $b$  – longitud de  $c$  al centro de masa de la plataforma del robot.
- $V$  – magnitud del vector velocidad del punto  $c$ , respecto al sistema fijo.
- $\alpha$  - orientación del robot (ángulo formado por el eje  $X$  y el vector velocidad del punto  $c$ ).
- $v_l$  y  $v_r$  – velocidades del centro de masa de las ruedas izquierda y derecha.
- $R$  – radio de las ruedas activas.
- $\varphi_r, \varphi_l$  – Velocidad angular de las ruedas.

Una vez definidos estos elementos serán considerados los siguientes conceptos como los **Supuestos fundamentales del modelo**:

1. El robot se compone de partes rígidas (cada punto del robot tiene una posición y orientación fija con respecto al sistema relativo al robot).
2. No hay deslizamiento en la dirección perpendicular a la rodadura.
3. No hay deslizamiento traslacional entre la rueda y el suelo.
4. El centro de masas del robot no coincide con el centro geométrico de éste.
5. El espacio donde se mueve el robot es perfectamente plano y sin inclinaciones.

Considerando la trayectoria que realiza el robot móvil sobre el plano  $XY$ , en un instante se puede considerar como un sistema fijo, formado por el eje de las ruedas activas y el eje principal del robot, son considerados como el sistema en movimiento.

Donde la posición del robot se puede ver como la proyección del centro de masas  $s$ , obteniendo así la velocidad dando origen a un primer modelo que nos describe en su trabajo [21] por la ecuación (2.1):

$$\begin{aligned}\dot{x}_s &= V \cos \alpha - b \Omega \sin \alpha \\ \dot{y}_s &= V \sin \alpha + b \Omega \cos \alpha \\ \dot{\alpha} &= \Omega\end{aligned}\tag{2.1}$$

donde  $\Omega$  es la velocidad angular, obteniendo el modelo cinemático.

## Restricciones no Holónomas

Para el estudio y comprensión del modelo matemático obtenido por [21] es necesario considerar las restricciones no holónomas, primeramente se determinan éstas para el sistema en movimiento. Las cuales son las llamadas condiciones de rodadura que presentan las ruedas activas del robot.

Considerando el supuesto fundamental planteado al principio de este capítulo, que no existe deslizamiento traslacional ni perpendicular, con esto se presentan condiciones que relacionan la velocidad que presenta el centro de masas  $cm$  de cada una de las ruedas y, la velocidad angular de rotación de ellas, siendo representadas por la ecuación (2.2)

$$v_{cm} = R \dot{\phi}\tag{2.2}$$

Donde las velocidades de las ruedas derecha e izquierda considerando la ecuación (2.2), quedan expresadas por las siguientes ecuaciones

$$v_r = R \dot{\phi}_r, \quad v_l = R \dot{\phi}_l\tag{2.3}$$

Y de esta representación es posible obtener las componentes de dichas velocidades con respecto al sistema fijo, quedando descritas por la ecuación :

$$\begin{aligned}\dot{x}_r &= R \dot{\phi}_r \cos \alpha, & \dot{y}_r &= R \dot{\phi}_r \sin \alpha \\ \dot{x}_l &= R \dot{\phi}_l \cos \alpha, & \dot{y}_l &= R \dot{\phi}_l \sin \alpha\end{aligned}\tag{2.4}$$

donde  $\alpha$  es el ángulo formado entre el eje  $X$  y el vector de la velocidad del  $cm$  de cada una de las ruedas activas, como se muestra en la figura 2.3.

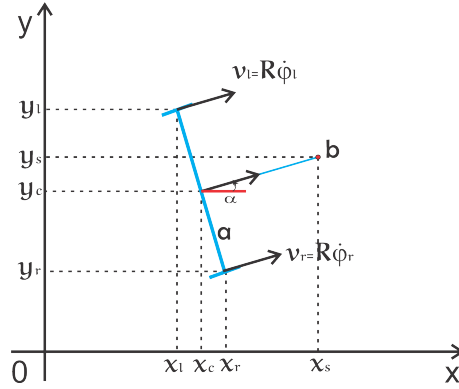
De la misma manera las coordenadas en el plano  $XY$  de la proyección del  $cm$  de las ruedas derecha e izquierda  $(x_r, y_r)$  y  $(x_l, y_l)$  respectivamente están dadas por la ecuación:

$$\begin{aligned} x_r &= x_c + a \operatorname{sen} \alpha, & y_r &= y_c - a \cos \alpha \\ x_l &= x_c - a \operatorname{sen} \alpha, & y_l &= y_c + a \cos \alpha \end{aligned} \quad (2.5)$$

y al derivar la ecuación (2.5) se obtienen las velocidades como se aprecia en la siguiente expresión :

$$\begin{aligned} \dot{x}_r &= \dot{x}_c + a \dot{\alpha} \cos \alpha, & \dot{y}_r &= \dot{y}_c + a \dot{\alpha} \operatorname{sen} \alpha \\ \dot{x}_l &= \dot{x}_c - a \dot{\alpha} \cos \alpha, & \dot{y}_l &= \dot{y}_c - a \dot{\alpha} \operatorname{sen} \alpha. \end{aligned} \quad (2.6)$$

Considerando la figura 2.3 es de ayuda para obtener las coordenadas del centro de masas del móvil, con respecto al sistema fijo, mismas que son expresadas por la ecuación:



**Figura 2.3:** Proyección del Robot en el plano XY

$$x_s = x_c + b \cos \alpha, \quad y_s = y_c + b \operatorname{sen} \alpha \quad (2.7)$$

Y que al derivar la expresión anterior obtendremos las componentes de su velocidad, que son representadas por:

$$\dot{x}_c = \dot{x}_s + b \dot{\alpha} \operatorname{sen} \alpha, \quad \dot{y}_c = \dot{y}_s - b \dot{\alpha} \cos \alpha \quad (2.8)$$

Al obtener las velocidades del centro de masas dadas por la ecuación (2.8) es posible sustituir estas en la expresión (2.6) donde las velocidades de cada rueda se encuentran en términos de la velocidad del centro de masas y así obtener:

$$\begin{aligned} \dot{x}_r &= \dot{x}_s + b \dot{\alpha} \operatorname{sen} \alpha + a \dot{\alpha} \cos \alpha, & \dot{y}_r &= \dot{y}_s - b \dot{\alpha} \cos \alpha + a \dot{\alpha} \operatorname{sen} \alpha \\ \dot{x}_l &= \dot{x}_s + b \dot{\alpha} \operatorname{sen} \alpha - a \dot{\alpha} \cos \alpha, & \dot{y}_l &= \dot{y}_s - b \dot{\alpha} \cos \alpha - a \dot{\alpha} \operatorname{sen} \alpha \end{aligned} \quad (2.9)$$

Finalmente comparando la expresión anterior con la ecuación (2.4) donde ambas ecuaciones representan las velocidades de las componentes en cada rueda, entonces al igualarlas y operar algebraicamente se obtienen 4 restricciones bilaterales:

$$\begin{aligned} \dot{x}_s + (bsen\alpha + acos\alpha)\dot{\alpha} - R\dot{\varphi}_r cos\alpha &= 0 \\ \dot{y}_s - (bcos\alpha - asen\alpha)\dot{\alpha} - R\dot{\varphi}_r sen\alpha &= 0 \\ \dot{x}_s + (bsen\alpha - acos\alpha)\dot{\alpha} - R\dot{\varphi}_l cos\alpha &= 0 \\ \dot{y}_s - (bcos\alpha + asen\alpha)\dot{\alpha} - R\dot{\varphi}_l sen\alpha &= 0 \end{aligned} \quad (2.10)$$

Del sistema de ecuaciones anterior obtenemos la matriz de coeficientes:

$$M = \begin{bmatrix} 1 & 0 & bsen\alpha + acos\alpha & -Rcos\alpha & 0 \\ 0 & 1 & -bcos\alpha + asen\alpha & -Rsen\alpha & 0 \\ 1 & 0 & bsen\alpha - acos\alpha & 0 & -Rcos\alpha \\ 0 & 1 & -bcos\alpha - asen\alpha & 0 & -Rsen\alpha \end{bmatrix} \quad (2.11)$$

Al realizar el calculo del rango de la matriz [28], se puede verificar que el determinante de cada submatriz, de  $M$  de tamaño  $4 \times 4$  es nulo, luego el rango de ésta matriz no es 4, por lo tanto las cuatro restricciones son linealmente dependientes. El determinante de la submatriz  $N$  de  $M$  de tamaño  $3 \times 3$  es  $R^2bcos\alpha$ , donde  $R^2b \neq 0$  y  $cos\alpha \neq 0$  para  $\alpha \neq (2n+1)\frac{\pi}{2}, n \in \mathbb{Z}$ , por lo que para  $\alpha(2n+1)\frac{\pi}{2}, n \in \mathbb{Z}$  el rango de  $M$  es 3.

$$N = \begin{bmatrix} bsen\alpha + acos\alpha & -Rcos\alpha & 0 \\ -bcos\alpha + asen\alpha & -Rsen\alpha & 0 \\ bsen\alpha - acos\alpha & 0 & -Rcos\alpha \end{bmatrix} \quad (2.12)$$

Para determinar las 3 restricciones independientes se lleva a  $M$  a la forma diagonal quedando el último renglón nulo y los 3 renglones restantes vienen siendo los coeficientes de las tres restricciones independientes. Así las tres restricciones linealmente independientes son:

$$\begin{aligned} \dot{x}_s + (bsen\alpha + acos\alpha)\dot{\alpha} - Rcos\alpha\dot{\varphi}_r &= 0 \\ \dot{y}_s - (bcos\alpha + asen\alpha)\dot{\alpha} - Rsen\alpha\dot{\varphi}_r &= 0 \\ 2acos\alpha\dot{\alpha} - Rcos\alpha\dot{\varphi}_r + Rcos\alpha\dot{\varphi}_l &= 0 \end{aligned} \quad (2.13)$$

## Método de Lagrange

El metodo que fue empleado para la determinación del modelo dinámico de acuerdo con [21], es mediante los multiplicadores de Lagrange, se llama así en honor a Joseph Louis Lagrange, es un procedimiento para funciones de múltiples variables sujetas a restricciones. Reduciendo el problema restringido con  $n$  variables a uno sin restricciones de  $n + k$  variables, donde  $k$  es igual al número de restricciones. Estas nuevas variables

escalares desconocidas, una para cada restricción, son llamadas multiplicadores de Lagrange.

El método dice que los puntos donde la función tiene un extremo condicionado con  $k$  restricciones, están entre los puntos estacionarios de una nueva función sin restricciones construida como una combinación lineal de la función y las funciones implicadas en las restricciones, cuyos coeficientes son los multiplicadores.

La función Lagrangiana es:

$$L = T - V \quad (2.14)$$

donde  $T$  es la energía cinética y  $V$  la energía potencial, las fuerzas externas que actúan sobre el robot son conservativas, entonces existe una función energía potencial tal que las fuerzas generalizadas  $M_j$  derivan de la función energía potencial  $M_j = \frac{\partial V}{\partial q_j}$ .

Entonces las ecuaciones de Lagrange están dadas por:

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_j} \right) - \frac{\partial L}{\partial q_j} = 0 \quad (2.15)$$

Considerando que el sistema que es objeto de estudio en este trabajo de tesis es un sistema no conservativo y a consecuencia de esta singularidad solo contamos con presencia de energía cinética  $T$ , entonces la ecuación (2.15) queda de la siguiente forma:

$$\frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} = M_j : j=1,2,\dots,n \quad (2.16)$$

Y considerando las restricciones que son colocadas del lado derecho de la ecuación (2.16), estas fuerzas generalizadas de ligadura llamadas también fuerzas de reacción  $\sum_{l=1}^k \lambda_l A_{lj}$  para  $j = 1, \dots, n$  tenemos entonces la siguiente expresión:

$$\frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} = M_j + \sum_{l=1}^k \lambda_l A_{lj} : j=1,2,\dots,n \quad (2.17)$$

y los términos  $A_{lj}$  están relacionados con las restricciones no holónomas de la siguiente manera:

$$\sum_{l=1}^3 A_{lj} \dot{q}_l + B_l = 0 : j=1,2,\dots,5 \quad (2.18)$$

Considerando la naturaleza del robot móvil que se trata de un sistema no conservativo, las ecuaciones de Lagrange a considerar son de la forma de la ecuación (1.24) y obtenemos la siguiente expresión de Lagrange que nos describe el movimiento del robot móvil:

$$\frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} = M_j + \sum_{l=1}^3 \lambda_l A_{lj} : j=1, \dots, 5. \quad (2.19)$$

Donde:

$$\begin{aligned} q_1 &= x_c & q_2 &= y_c & q_3 &= \alpha & q_4 &= \varphi_r & q_5 &= \varphi_l \\ \dot{q}_1 &= \dot{x}_c & \dot{q}_2 &= \dot{y}_c & \dot{q}_3 &= \dot{\alpha} & \dot{q}_4 &= \dot{\varphi}_r & \dot{q}_5 &= \dot{\varphi}_l \end{aligned} \quad (2.20)$$

son las coordenadas de posiciones y velocidades generalizadas respectivamente, las fuerzas generalizadas  $M_j$  son los torques de los motores correspondientes a cada rueda activa y dado que estas sólo tiene influencia directa con la velocidad de giro de cada una de ellas, se tiene:

$$M_j = 0; \quad j = 1, 2, 3 \qquad M_4 = M_r \qquad M_5 = M_l$$

siendo  $M_r$  y  $M_l$  los torques del motor de la rueda derecha e izquierda respectivamente,  $T$  es la energía cinética dada por:

$$T = \frac{1}{2} [m_s (\dot{x}_s^2 + \dot{y}_s^2) + J_s \dot{\alpha}^2 + J \dot{\varphi}_r^2 + J \dot{\varphi}_l^2] \quad (2.21)$$

Donde:

$m_s$  = Masa de la plataforma del robot.

$J_s$  = Tensor de inercia del centro de masas.

$J$  = Tensor de inercia de cada de las ruedas.

En la ecuación (2.21) es posible identificar los 5 grados de libertad de nuestro sistema, y el segundo término del lado derecho de (2.19) son las fuerzas reactivas del sistema, donde  $\lambda_1$ ,  $\lambda_2$  y  $\lambda_3$  son los multiplicadores de Lagrange.

Si se sustituye la energía cinética (2.21) en la ecuación (2.17), se obtiene el siguiente sistema de segundo orden:

$$\begin{aligned} m_s \ddot{x}_s &= \lambda_1 A_{11} + \lambda_2 A_{21} + \lambda_3 A_{31} \\ m_s \ddot{y}_s &= \lambda_1 A_{12} + \lambda_2 A_{22} + \lambda_3 A_{32} \\ J_s \ddot{\alpha} &= \lambda_1 A_{13} + \lambda_2 A_{23} + \lambda_3 A_{33} \\ J \ddot{\varphi}_r &= \lambda_1 A_{14} + \lambda_2 A_{24} + \lambda_3 A_{34} + M_r \\ J \ddot{\varphi}_l &= \lambda_1 A_{15} + \lambda_2 A_{25} + \lambda_3 A_{35} + M_l \end{aligned} \quad (2.22)$$

El sistema representado por la ecuación (2.22) se encuentran en términos de los multiplicadores de Lagrange así como de las restricciones mostradas en la ecuación (2.13), mismas que se encuentran de la forma:

$$g_l(q, \dot{q}) = \sum_{i=1}^5 A_{li} \dot{q}_i + B_l = 0, \quad B_l = 0 \text{ para } l = 1, 2, 3 \quad (2.23)$$

desarrollando la ecuación (2.23) se obtiene:

$$\begin{aligned} A_{11}\dot{x}_s + A_{12}\dot{y}_s + A_{13}\dot{\alpha} + A_{14}\dot{\varphi}_r + A_{15}\dot{\varphi}_l &= 0 \\ A_{21}\dot{x}_s + A_{22}\dot{y}_s + A_{23}\dot{\alpha} + A_{24}\dot{\varphi}_r + A_{25}\dot{\varphi}_l &= 0 \\ A_{31}\dot{x}_s + A_{32}\dot{y}_s + A_{33}\dot{\alpha} + A_{34}\dot{\varphi}_r + A_{35}\dot{\varphi}_l &= 0 \end{aligned} \quad (2.24)$$

y del sistema donde representamos las restricciones (2.13) se comparan con el sistema anterior de la ecuación (2.24) para así obtener los valores de los terminos de  $A_{lj}$  quedando de la forma:

$$\begin{aligned} A_{11} &= 1 & A_{12} &= 0 & A_{13} &= b\sin\alpha + a\cos\alpha & A_{14} &= -R\cos\alpha & A_{15} &= 0 \\ A_{21} &= 0 & A_{22} &= 1 & A_{23} &= -b\cos\alpha + a\sin\alpha & A_{24} &= -R\sin\alpha & A_{25} &= 0 \\ A_{31} &= 0 & A_{32} &= 0 & A_{33} &= b\sin\alpha + a\cos\alpha & A_{34} &= -R\cos\alpha & A_{35} &= R\cos\alpha \end{aligned} \quad (2.25)$$

Una vez obtenidos estos valores los sustituimos en la ecuación (2.22) obteniendo el siguiente modelo matemático de segundo orden:

$$\begin{aligned} m_s\ddot{x}_s &= \lambda_1 \\ m_s\ddot{y}_s &= \lambda_2 \\ J_s\ddot{\alpha} &= \lambda_1(b\sin\alpha + a\cos\alpha) + \lambda_2(-b\cos\alpha - a\sin\alpha) + \lambda_3(2a\cos\alpha) \\ J\ddot{\varphi}_r &= \lambda_1(-R\cos\alpha) + \lambda_2(-R\sin\alpha) + \lambda_3(-R\cos\alpha) + M_r \\ J\ddot{\varphi}_l &= \lambda_3(R\cos\alpha) + M_l \end{aligned} \quad (2.26)$$

con las 3 restricciones no-holónomicas ya obtenidas anteriormente y expresadas en la ecuación (2.13), este modelo matemático de 5 ecuaciones de segundo orden como se sabe puede llevarse a un sistema de primer orden pero crece a 10 ecuaciones quedando de la forma:

$$\begin{aligned} \dot{x}_s &= v_1, \\ \dot{v}_1 &= \frac{\lambda_1}{m_s}, \\ \dot{y}_s &= v_2, \\ \dot{v}_2 &= \frac{\lambda_2}{m_s}, \\ \dot{\alpha} &= v_3, \\ \dot{v}_3 &= \frac{\lambda_1}{J_s}(b\sin\alpha + a\cos\alpha) + \frac{\lambda_2}{J_s}(-b\cos\alpha + a\sin\alpha) + \frac{\lambda_3}{J_s}2a\cos\alpha, \\ \dot{\varphi}_r &= v_4 \\ \dot{v}_4 &= -\frac{\lambda_1}{J}R\cos\alpha - \frac{\lambda_2}{J}R\sin\alpha - \frac{\lambda_3}{J}R\cos\alpha + \frac{M_r}{J}, \\ \dot{\varphi}_l &= v_5, \\ \dot{v}_5 &= \frac{\lambda_3}{J}R\cos\alpha + \frac{M_l}{J} \end{aligned} \quad (2.27)$$



Ahora en este sistema de 10 ecuaciones note que las coordenadas  $v_1, \dots, v_5$  tienen una dependencia con las coordenadas  $x_s, y_s, \alpha, \varphi_r$  y  $\varphi_l$ .

Retomamos el sistema de segundo orden representado por la ecuación (2.26) y sus restricciones de la expresión (2.13), sin embargo el sistema se encuentra en este momento con los multiplicadores de Lagrange y para obtener un sistema de 5 ecuaciones con 5 incógnitas en la siguiente sección se obtendrán los valores de los multiplicadores de Lagrange y para esto se realizara una transformación lineal de las coordenadas  $\varphi_r$  y  $\varphi_l$  cambiaran por  $V$  y  $\Omega$ .

## Modelo Matemático sin Multiplicadores de Lagrange

Debido a que el sistema representado por la ecuación (2.26) es de segundo orden, para poder obtener los valores de los multiplicadores de Lagrange es necesario obtener las aceleraciones para así sustituirlas en el modelo de segundo orden y obtener los valores de  $\lambda_1, \lambda_2$  y  $\lambda_3$  como lo realizo [21].

Consideremos que la velocidad del origen del sistema en movimiento no es constante, esto es, si la velocidad  $V$  del punto  $c$  varía con el tiempo, se puede derivar (2.1) respecto de  $t$  para obtener el siguiente sistema de segundo orden:

$$\begin{aligned}\ddot{x}_s &= \cos\alpha \left( \dot{V} - b\Omega^2 \right) - \sin\alpha \left( V\Omega + b\dot{\Omega} \right) \\ \ddot{y}_s &= \sin\alpha \left( \dot{V} - b\Omega^2 \right) + \cos\alpha \left( V\Omega + b\dot{\Omega} \right) \\ \ddot{\alpha} &= \dot{\Omega}\end{aligned}\tag{2.28}$$

Retomando las coordenadas de la velocidad del punto  $c$  dadas por la ecuación (2.1) para sustituirlo en la expresión (2.6) y obtenemos:

$$\begin{aligned}\dot{x}_r &= (V + a\Omega) \cos\alpha, & \dot{y}_r &= (V + a\Omega) \sin\alpha \\ \dot{x}_l &= (V - a\Omega) \cos\alpha, & \dot{y}_l &= (V - a\Omega) \sin\alpha\end{aligned}\tag{2.29}$$

Obteniendo estas ecuaciones que corresponde a la velocidad de las coordenadas de las ruedas, y estas mismas ya se habían obtenido y expresadas en la ecuación (2.4), realizando la comparación obtenemos:

$$\dot{\varphi}_r = \frac{V}{R} + \frac{a}{R}\Omega, \quad \dot{\varphi}_l = \frac{V}{R} - \frac{a}{R}\Omega\tag{2.30}$$

y al derivar con respecto del tiempo obtenemos la aceleración:

$$\ddot{\varphi}_r = \frac{\dot{V}}{R} + \frac{a}{R}\dot{\Omega}, \quad \ddot{\varphi}_l = \frac{\dot{V}}{R} - \frac{a}{R}\dot{\Omega}\tag{2.31}$$

Se han obtenido las velocidades y estas pueden ser sustituidas en la ecuación (2.26) para obtener:

$$\begin{aligned}
 m_s \left[ \left( \dot{V} - b\Omega^2 \right) \cos\alpha - \left( V\Omega + b\dot{\Omega} \right) \sin\alpha \right] &= \lambda_1 \\
 m_s \left[ \left( \dot{V} - b\Omega^2 \right) \sin\alpha + \left( V\Omega + b\dot{\Omega} \right) \cos\alpha \right] &= \lambda_2 \\
 J_s \dot{\Omega} &= a(\lambda_1 \cos\alpha + \lambda_2 \sin\alpha) + b(\lambda_1 \sin\alpha - \lambda_2 \cos\alpha) + \lambda_3 2a \cos\alpha \\
 \frac{J}{R} \left( \dot{V} + a\dot{\Omega} \right) &= -R(\lambda_1 \cos\alpha + \lambda_2 \sin\alpha) - \lambda_3 R \cos\alpha + M_r \\
 \frac{J}{R} \left( \dot{V} - a\dot{\Omega} \right) &= \lambda_3 R \cos\alpha + M_l
 \end{aligned} \tag{2.32}$$

Lo siguiente sera operar de tal forma que podamos encontrar los valores de los multiplicadores de Lagrange, si se multiplica por  $\cos\alpha$  en la primera ecuación, por  $\sin\alpha$  en la segunda ecuación de (2.32) y sumando se tiene la primera expresión de (2.33). Luego, multiplicando por  $\sin\alpha$  en la primera ecuación, por  $\cos\alpha$  en la segunda ecuación de (2.32) y restando se tiene obtiene la segunda expresión de (2.33)

$$\begin{aligned}
 \lambda_1 \cos\alpha + \lambda_2 \sin\alpha &= m_s \left( \dot{V} - b\Omega^2 \right) \\
 \lambda_1 \sin\alpha - \lambda_2 \cos\alpha &= m_s \left( V\Omega + b\dot{\Omega} \right)
 \end{aligned} \tag{2.33}$$

Tomando las ecuaciones anteriores y las sustituimos en la tercera y cuarta ecuación de (2.32) para obtener:

$$\lambda_3 = \frac{J_s \dot{\Omega} - a m_s \left( \dot{V} - b\Omega^2 \right) + b m_s \left( V\Omega + b\dot{\Omega} \right)}{2a \cos\alpha} \tag{2.34}$$

$$\begin{aligned}
 \frac{J}{R} \left( \dot{V} + a\dot{\Omega} \right) &= -R m_s \left( \dot{V} - b\Omega^2 \right) - \lambda_3 R \cos\alpha + M_r \\
 \frac{J}{R} \left( \dot{V} - a\dot{\Omega} \right) &= \lambda_3 R \cos\alpha + M_l
 \end{aligned} \tag{2.35}$$

Así obtenemos de (2.33) y (2.34) los valores de los multiplicadores de Lagrange los cuales son:

$$\begin{aligned}
 \lambda_1 &= m_s \left[ \left( \dot{V} - b\Omega^2 \right) \cos\alpha - \left( V\Omega + b\dot{\Omega} \right) \sin\alpha \right] \\
 \lambda_2 &= m_s \left[ \left( \dot{V} - b\Omega^2 \right) \sin\alpha + \left( V\Omega + b\dot{\Omega} \right) \cos\alpha \right] \\
 \lambda_3 &= \frac{J_s \dot{\Omega} - a m_s \left( \dot{V} - b\Omega^2 \right) + b m_s \left( V\Omega + b\dot{\Omega} \right)}{2a \cos\alpha}
 \end{aligned} \tag{2.36}$$

Una vez obtenidos los valores de los multiplicadores para poder sustituirlos en nuestro modelo, primero realizaremos la suma y resta de la ecuación (2.35) y así obtener:

$$\begin{aligned}
 \frac{2J}{R}\dot{V} &= -Rm_s(\dot{V} - b\Omega^2) + (M_r + M_l), \\
 \frac{2Ja}{R}\Omega &= -Rm_s(\dot{V} - b\Omega^2) - 2\lambda_3 R \cos\alpha + (M_r - M_l)
 \end{aligned} \tag{2.37}$$

De tal forma que solo resta sustituir  $\lambda_3$  en (2.37) y así obtenemos:

$$\begin{aligned}
 \left(\frac{2}{R}J + Rm_s\right)\dot{V} &= Rm_sb\Omega^2 + (M_r + M_l) \\
 \left(\frac{2a}{R}J + \frac{R}{a}J_s + \frac{R}{a}b^2m_s\right)\dot{\Omega} &= \frac{-R}{a}m_sbV\Omega + (M_r - M_l)
 \end{aligned} \tag{2.38}$$

Finalmente multiplicamos por  $\frac{1}{R}$  a la primera ecuación de (2.38) y por  $\frac{a}{R}$  a la segunda, se obtiene así el siguiente modelo matemático sin multiplicadores de Lagrange de 5 ecuaciones de primer orden:

$$\begin{aligned}
 \dot{x}_s &= V \cos\alpha - b\Omega \sin\alpha \\
 \dot{y}_s &= V \sin\alpha + b\Omega \cos\alpha \\
 \dot{\alpha} &= \Omega \\
 M_0\dot{V} &= m_sb\Omega^2 + \frac{1}{R}(M_r + M_l) \\
 \tilde{J}\dot{\Omega} &= -m_sbV\Omega + \frac{a}{R}(M_r - M_l)
 \end{aligned} \tag{2.39}$$

Donde:

$$M_0 = \left(\frac{2}{R^2}J + m_s\right), \quad \tilde{J} = \left(\frac{2a^2}{R^2}J + J_s + b^2m_s\right)$$

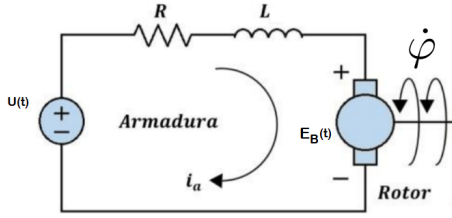
En donde podemos observar que la ecuación (2.39) es nuestro modelo matemático que describe la dinámica del robot móvil que empleamos en este trabajo, donde podemos identificar los grados de libertad que se encuentran en nuestro sistema.

### Modelo del motor e integración al modelo dinámico.

Por la ley de Kirchhoff la corriente  $I$  satisface la siguiente expresión:

$$L\dot{I} = U - E_B - ZI \tag{2.40}$$

donde  $Z$  es la resistencia,  $L$  la inductancia,  $U$  el voltaje aplicado al motor,  $E_B$  la fuerza contra-electromotriz que es proporcional a la velocidad de rotación,  $E_B = K_m\dot{\phi}$ , considerando el diagrama mostrado en la figura 2.4.



**Figura 2.4:** Diagrama de un motor de DC.

Partiendo del diagrama anterior, se tienen las siguientes ecuaciones:

$$\begin{aligned} L\dot{I}_r &= U_r - K_m\dot{\phi}_r - ZI_r \\ L\dot{I}_l &= U_l - K_m\dot{\phi}_l - ZI_l \end{aligned} \quad (2.41)$$

donde  $I_r$ ,  $I_l$  son la corriente del motor derecho e izquierdo respectivamente;  $U_r$ ,  $U_l$  son los voltajes aplicados a los motores correspondientes y por ultimo  $\dot{\phi}_r$ ,  $\dot{\phi}_l$  son las velocidades de giro para cada rueda derecha e izquierda.

Donde el torque de un motor de CD es proporcional a la corriente.

$$M_r = K_0 I_r, \quad M_l = K_0 I_l \quad (2.42)$$

Entonces podemos decir que:

$$\begin{aligned} M_r + M_l &= K_0(I_r + I_l), \\ M_r - M_l &= K_0(I_r - I_l), \end{aligned} \quad (2.43)$$

Ahora de la ecuación (2.41) y sustituimos la expresión (2.43) en el sistema (2.39) obtenemos el siguiente sistema de 7 ecuaciones de primer orden.

$$\begin{aligned} \dot{x}_s &= V \cos(\alpha) - b\Omega \sin(\alpha) \\ \dot{y}_s &= V \sin(\alpha) + b\Omega \cos(\alpha) \\ \dot{\alpha} &= \Omega \\ M_0 \dot{V} &= m_s b \Omega^2 + \frac{K_0}{R} (I_r + I_l) \\ \tilde{J} \dot{\Omega} &= -m_s b V \Omega + \frac{a K_0}{R} (I_r - I_l) \\ L \dot{I}_r &= U_r - \frac{K_m}{R} (V + a\Omega) - Z I_r \\ L \dot{I}_l &= U_l - \frac{K_m}{R} (V - a\Omega) - Z I_l \end{aligned} \quad (2.44)$$

Siendo nuestro modelo Dinámico del robot móvil considerando las ecuaciones ideales de los motores de DC.

## Reducción y modelo matemático final del sistema controlable no lineal

Ahora aplicaremos lo visto acerca del teorema de Tikhonov para reducir el modelo expresado en la ecuación (2.44) de orden siete a un sistema de orden 5.

Realizando las siguientes transformaciones y aplicándolas al modelo (2.44) tenemos que:

$$T = T_*t; \quad \Omega = \Omega_*w; \quad V = V_*v; \quad U_r = U_*u_r; \quad U_l = U_*u_l; \quad I_r = I_*i_r; \quad I_l = I_*i_l \quad (2.45)$$

Obtenemos el sistema singularmente perturbado:

$$\begin{aligned}
 \dot{x} &= T_*V_*\cos(\alpha) - T_*\Omega_*bw\sin(\alpha) \\
 \dot{y}_s &= T_*V_*\sin(\alpha) + T_*\Omega_*bw\cos(\alpha) \\
 \dot{\alpha} &= T_*\Omega_*w \\
 \dot{v} &= \frac{T_*\Omega_*^2m_sb}{V_*M_0}w^2 + \frac{T_*I_*K_0}{V_*RM_0}(i_r + i_l) \\
 \dot{w} &= \frac{-m_sbT_*V_*}{J}vw + \frac{T_*I_*aK_0}{\Omega_*R\tilde{J}}(i_r - i_l) \\
 \frac{LI_*}{T_*U_*}\dot{i}_r &= u_r - \frac{K_m}{U_*R}(V_*v + a\Omega_*w) - \frac{ZI_*}{U_*}i_r \\
 \frac{LI_*}{T_*U_*}\dot{i}_l &= u_l - \frac{K_m}{U_*R}(V_*v - a\Omega_*w) - \frac{ZI_*}{U_*}i_l
 \end{aligned} \quad (2.46)$$

donde  $\epsilon = \frac{LI_*}{T_*U_*}$  es el parámetro pequeño. Y si hacemos a  $\epsilon = 0$ , el sistema degenerado asociado a (2.46) es :

$$\begin{aligned}
 \dot{\bar{x}} &= \delta_1\bar{v}\cos(\alpha) - \delta_2b\bar{w}\sin(\alpha) \\
 \dot{\bar{y}} &= \delta_1\bar{v}\sin(\alpha) + \delta_2b\bar{w}\cos(\alpha) \\
 \dot{\bar{\alpha}} &= \delta_2\bar{w} \\
 \dot{\bar{v}} &= \delta_3b\bar{w}^2 + \delta_4(\bar{i}_r + \bar{i}_l) \\
 \dot{\bar{w}} &= -\delta_5b\bar{v}\bar{w} + \delta_6(\bar{i}_r - \bar{i}_l)
 \end{aligned} \quad (2.47)$$

y el sistema adjunto esta expresado por:

$$\begin{aligned}
 0 &= \bar{u}_r - \delta_7\bar{v} - \delta_8\bar{w} - \delta_9\bar{i}_r \\
 0 &= \bar{u}_l - \delta_7\bar{v} + \delta_8\bar{w} - \delta_9\bar{i}_l
 \end{aligned} \quad (2.48)$$

donde:  $\delta_1 = T_*V_*$ ,  $\delta_2 = T_*\Omega_*$ ,  $\delta_3 = \frac{T_*\Omega_*^2m_s}{V_*M_0}$ ,  $\delta_4 = \frac{T_*I_*K_0}{V_*RM_0}$ ,  $\delta_5 = \frac{T_*V_*m_s}{\tilde{J}}$ ,  $\delta_6 = \frac{T_*I_*aK_0}{\tilde{J}R\Omega_*}$ ,  $\delta_7 = \frac{K_mV_*}{RU_*}$ ,  $\delta_8 = \frac{K_mV_*}{U_*R}$ ,  $\delta_9 = \frac{ZI_*}{U_*}$ ,  $\tilde{J} = \frac{2a^2J}{R^2} + Js + m_sb^2$ ,  $M_0 = \frac{2J}{R^2} + m_s$ . Se puede verificar que los sistema degenerado y adjunto satisfacen el teorema de Tikhonov. Luego, despejando  $\bar{i}_r$  y  $\bar{i}_l$  del sistema adjunto dado por (2.48) y sustituyendo en (2.47) se obtiene:

$$\begin{aligned}
\dot{\bar{x}} &= \delta_1 \bar{v} \cos(\alpha) - \delta_2 b \bar{\Omega} \sin(\alpha) \\
\dot{\bar{y}} &= \delta_1 \bar{v} \sin(\alpha) + \delta_2 b \bar{\Omega} \cos(\alpha) \\
\dot{\bar{\alpha}} &= \delta_2 \bar{\Omega} \\
\dot{\bar{v}} &= \delta_3 b \bar{\Omega}^2 - 2 \frac{\delta_4 \delta_7}{\delta_9} \bar{v} + 2 \frac{\delta_4}{\delta_9} \bar{u}_1 \\
\dot{\bar{\Omega}} &= -b \delta_5 b \bar{v} \bar{\Omega} - 2 \frac{\delta_6 \delta_8}{\delta_9} \bar{\Omega} + 2 \frac{\delta_6}{\delta_9} \bar{u}_2
\end{aligned} \tag{2.49}$$

Finalmente el modelo matemático no lineal de cinco ecuaciones con dos controles:  $\bar{u}_1 = (\bar{u}_r + \bar{u}_l)/2$  y  $\bar{u}_2 = (\bar{u}_r - \bar{u}_l)/2$ .

## 2.2. Trayectorias deseadas y ecuaciones lineales.

Dada la ecuación (2.49) vamos a considerar las siguientes entradas de control:

$$\begin{aligned}
u_t &= u_i + u_o \\
u_o &= \bar{u}_1 \quad y \quad \bar{u}_2;
\end{aligned} \tag{2.50}$$

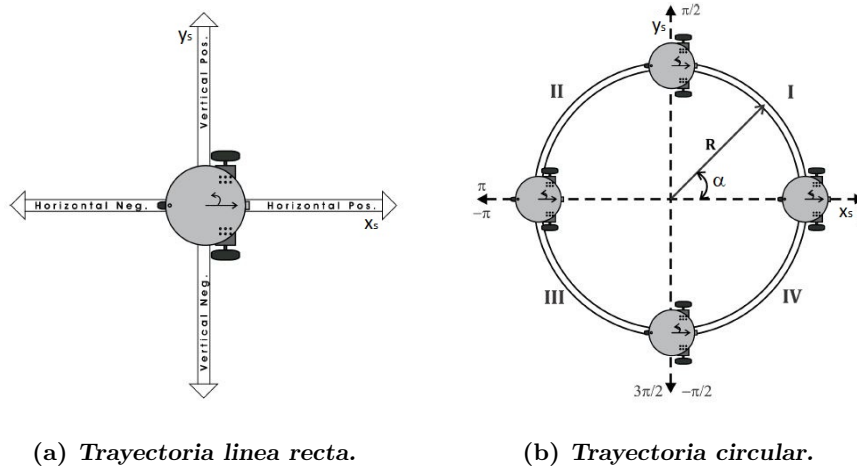
Donde  $u_i$  es un control inicial, es decir un voltaje necesario para iniciar el movimiento del robot móvil y  $u_o$  es el control óptimo. Para realizar el análisis de estabilidad correspondiente para el sistema (2.49) sobre una trayectoria es necesario linealizar el modelo alrededor de está, de acuerdo a la siguiente expresión:

$$\dot{x} = Ax; \tag{2.51}$$

donde  $A = \frac{\partial f(y^d, u^r)}{\partial y}$  y el vector de estado  $x \in \mathbb{R}^n$  representa las desviaciones del estado deseado al estado actual, es decir  $x = y^d - y$ . Por lo que aplicando el criterio (2.51) al sistema obtenemos para la linea recta:

$$A = \begin{bmatrix} 0 & 0 & -\bar{v} * \delta_1 * \sin(\alpha) & \delta_1 * \cos(\alpha) & -b * \delta_2 * \sin(\alpha) \\ 0 & 0 & \bar{v} * \delta_1 * \cos(\alpha) & \delta_1 * \sin(\alpha) & b \delta_2 * \cos(\alpha) \\ 0 & 0 & 0 & 0 & \delta_2 \\ 0 & 0 & 0 & -2 \frac{\delta_4 * \delta_7}{\delta_9} & 0 \\ 0 & 0 & 0 & 0 & -\bar{v} * b * \delta_5 - 2 \frac{\delta_6 * \delta_8}{\delta_9} \end{bmatrix} x \tag{2.52}$$

Este sistema de ecuaciones determinan a la matriz  $A$  quien sera evaluada para cada trayectoria deseada. Con los posibles movimientos básicos que un robot móvil puede realizar en un plano, como se muestran en la figura 2.5.



(a) Trayectoria linea recta.

(b) Trayectoria circular.

**Figura 2.5:** Trayectorias deseadas.

Considerando que el robot presenta movimientos horizontales y verticales paralelos a los ejes  $0, x_s$  y  $0, y_s$  se deduce la tabla dando origen a las trayectorias deseadas.

| Trayectoria                       | $x^d$         | $y^d$         | $\alpha^d$       | $V^d$ | $\Omega^d$ |
|-----------------------------------|---------------|---------------|------------------|-------|------------|
| Linea horizontal sentido positivo | $x_0 + V_0 t$ | 0             | 0                | $V_0$ | 0          |
| Linea horizontal sentido negativo | $x_0 + V_0 t$ | 0             | $\pi$            | $V_0$ | 0          |
| Linea vertical sentido positivo   | 0             | $y_0 + V_0 t$ | $\frac{\pi}{2}$  | $V_0$ | 0          |
| Linea vertical sentido negativo   | 0             | $y_0 + V_0 t$ | $-\frac{\pi}{2}$ | $V_0$ | 0          |

**Tabla 2.1:** Trayectorias linea recta.

Siendo las ecuaciones (2.39) que definen el movimiento del robot móvil, se pueden aproximar linealmente a vecindades cercanas para las trayectorias deseadas con la expresión (1.54), Así, con la trayectoria deseada:

$$y^d = \begin{bmatrix} x^d \\ y^d \\ \alpha^d \\ V^d \\ \Omega^d \end{bmatrix} = \begin{bmatrix} x_0 + v_0(t) \\ 0 \\ 0 \\ v_0 \\ 0 \end{bmatrix} \quad (2.53)$$

correspondiente a un movimiento recto horizontal en el sentido positivo, se tienen las ecuaciones lineales:

$$\begin{aligned}
\Delta \dot{\tilde{x}} &= -\delta_1 V_0 \sin(\alpha^*) \Delta \alpha + \delta_1 \cos(\alpha^*) \Delta V - \delta_2 b \sin(\alpha^*) \Delta \Omega \\
\Delta \dot{\tilde{y}} &= \delta_1 V_0 \cos(\alpha^*) \Delta \Omega + \delta_1 \sin(\alpha^*) \Delta V + \delta_2 b \cos(\alpha^*) \Delta \Omega \\
\Delta \dot{\tilde{\alpha}} &= \delta_2 \Delta \Omega \\
\Delta \dot{\tilde{V}} &= -2 \frac{\delta_4 \delta_7}{\delta_9} \Delta V + 2 \frac{\delta_4}{\delta_9} \Delta \bar{u}_1 \\
\Delta \dot{\tilde{\Omega}} &= - \left( b \delta_5 V_0 + 2 \frac{\delta_6 \delta_8}{\delta_9} \right) \Delta \Omega + 2 \frac{\delta_6}{\delta_9} \Delta \bar{u}_2
\end{aligned} \tag{2.54}$$

Con este procedimiento, es posible obtener los 4 sistemas de ecuaciones linealizados para cada una de las trayectorias, correspondientes a la línea recta.

Ahora obtenemos las ecuaciones lineales para la trayectoria deseada de una circunferencia, de acuerdo a la siguiente tabla:

| Trayectoria                         | $x^d$          | $y^d$          | $\alpha \in$             | $\beta^d$                | $V^d$   | $\Omega^d$ |
|-------------------------------------|----------------|----------------|--------------------------|--------------------------|---------|------------|
| Cuadrante I (sentido horario)       | $R \cos \beta$ | $R \sin \beta$ | $[\frac{\pi}{2}, 0]$     | $-w_0 t$                 | $w_0 R$ | $-w_0 R$   |
| Cuadrante I (Sentido antihorario)   | $R \cos \beta$ | $R \sin \beta$ | $[0, \frac{\pi}{2}]$     | $-w_0 + \frac{\pi}{2} t$ | $w_0 R$ | $w_0 R$    |
| Cuadrante II (sentido horario)      | $R \cos \beta$ | $R \sin \beta$ | $[\pi, \frac{\pi}{2}]$   | $-w_0 + \frac{\pi}{2}$   | $w_0 R$ | $-w_0 R$   |
| Cuadrante II (Sentido antihorario)  | $R \cos \beta$ | $R \sin \beta$ | $[\frac{\pi}{2}, \pi]$   | $w_0 + \pi$              | $w_0 R$ | $w_0 R$    |
| Cuadrante III (sentido horario)     | $R \cos \beta$ | $R \sin \beta$ | $[-\frac{\pi}{2}, -\pi]$ | $\pi - w_0$              | $w_0 R$ | $-w_0 R$   |
| Cuadrante III (Sentido antihorario) | $R \cos \beta$ | $R \sin \beta$ | $[-\pi, -\frac{\pi}{2}]$ | $w_0 - \frac{\pi}{2}$    | $w_0 R$ | $w_0 R$    |
| Cuadrante IV (sentido horario)      | $R \cos \beta$ | $R \sin \beta$ | $[0, -\frac{\pi}{2}]$    | $-w_0 - \frac{\pi}{2}$   | $w_0 R$ | $-w_0 R$   |
| Cuadrante IV (Sentido antihorario)  | $R \cos \beta$ | $R \sin \beta$ | $[-\frac{\pi}{2}, 0]$    | $w_0$                    | $w_0 R$ | $w_0 R$    |

**Tabla 2.2:** Trayectorias deseadas Circular.

donde la trayectoria deseada para el movimiento circular en el cuadrante I con sentido horario, o  $\beta = -w_0 t$ :

$$y^d = \begin{bmatrix} x^d \\ y^d \\ \alpha^d \\ V^d \\ \Omega^d \end{bmatrix} = \begin{bmatrix} R \cos(\alpha) \\ R \sin(\alpha) \\ -w_0 t \\ w_0 R \\ -w_0 R \end{bmatrix} \tag{2.55}$$

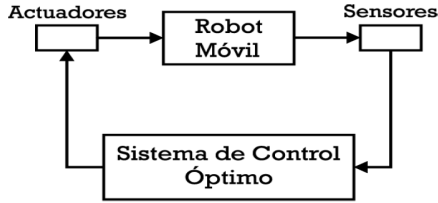
Dadas la trayectoria deseada obteniendo las siguientes ecuaciones linealizadas son:



$$\begin{aligned}
 \Delta \dot{\tilde{x}} &= \left( \Omega_0 r \sin(\Omega_0 t) - b \Omega_0 \cos(-\Omega_0 t) \right) \Delta \tilde{\alpha} + \cos(-\Omega_0 t) \Delta \tilde{V} - \sin(\Omega_0 t) \Delta \tilde{\Omega} \\
 \Delta \dot{\tilde{y}} &= \left( \Omega_0 r \cos(-\Omega_0 t) - b \Omega_0 \sin(\Omega_0 t) \right) \Delta \tilde{\alpha} - \sin(\Omega_0 t) \Delta \tilde{V} + b \cos(-\Omega_0 t) \Delta \tilde{\Omega} \\
 \Delta \dot{\tilde{\alpha}} &= \Delta \tilde{\Omega} \\
 \Delta \dot{\tilde{V}} &= \frac{-R(2m_s b \Omega_0) \Delta \tilde{\Omega} + \bar{u}_1}{2J + R^2 m_s} \\
 \Delta \dot{\tilde{\Omega}} &= \frac{R m_s b \Omega_0 \Delta \tilde{\Omega} + a \bar{u}_2}{2a^2 J + R J_s + R b^2 m_s}
 \end{aligned} \tag{2.56}$$

### 2.3. Síntesis del control óptimo mediante la programación dinámica.

En este capítulo se resuelve el problema de encontrar una ley de control óptimo para que el robot móvil pueda describir las trayectorias deseadas. Aunque es necesaria la síntesis de un control óptimo para cada una de las trayectorias, sólo se resolverá el problema para dos de estas considerando que cada una de estas pertenece a cada uno de los dos grupos de trayectorias. Se sintetizará un control óptimo a partir del sistema lineal de la trayectoria deseada.



**Figura 2.6:** Esquema general del sistema.

De esta manera el objetivo general para este trabajo de tesis se representa por el esquema de la figura 2.6, en donde se requiere del diseño de un control óptimo lineal retroalimentados para un robot móvil que consta de dos motores de DC como actuadores y dos encoders incrementales por efecto hall, como sensores.

El sistema de control óptimo recibirá información de los encoders que se encuentran montados en cada uno de los motores DC, de esta manera se conocerá la posición exacta, y así aplicar las correcciones necesarias a los motores para mantener una trayectoria continua del robot ya sea en línea recta o un segmento circular.

Con los resultados obtenidos en la sección 1.7, se puede entonces dar solución al problema de control óptimo planteado por la siguiente expresión:

$$\begin{aligned}
\min_u J &= \int_{t_0}^t [x^T(t)Q(t)x(t) + u^T(t)R(t)u(t)] dt + x^T(t_1)Fx(t_1) \\
\text{sujeto } a : \dot{x}(t) &= A(t)x(t) + B(t)u(t) \\
\text{con } : x(t_0) &= x_0
\end{aligned} \tag{2.57}$$

donde el sistema  $\dot{x}(t) = A(t)x(t) + B(t)u(t)$  representa al sistema lineal correspondiente a la trayectoria deseada. Para solucionar este problema de control óptimo, es necesario recordar la ecuación (1.87) :

$$\begin{aligned}
0 &= W_t(x, t) + \min_u \{x^T Qx + u^T R u + \nabla_x W(x, t)[Ax + Bu]\} \\
\text{con } : W(x, t_1) &= x^T Fx,
\end{aligned} \tag{2.58}$$

ahora, se establece la suposición de que la solución a la ecuación (2.58) es de la forma:

$$W(x, t) = x^T P(t)x, \tag{2.59}$$

donde  $P(t)$  es una matriz simétrica n-dimensional, obteniendo así:

$$\frac{\partial W(x, t)}{\partial t} = W_t(x, t) = x^T \frac{\partial P(t)}{\partial t} \tag{2.60}$$

$$\frac{\partial W(x, t)}{\partial x} = \nabla_x W(x, t) = 2x^T P(t) \tag{2.61}$$

Y ahora sustituyendo (2.60) y (2.61) en (2.58) de Hamilton-Jacobi-Bellman, se tiene que:

$$0 = x^T \frac{\partial P(t)}{\partial t} x + \min_u \{x^T Qx + u^T R u + 2x^T P(t)[Ax + Bu]\} \tag{2.62}$$

Derivando parcialmente con respecto a  $u$  la expresión entre llaves e igualando a cero se obtiene:

$$\begin{aligned}
\frac{\partial}{\partial u} \{x^T Qx + u^T R u + 2x^T P(t)[Ax + Bu]\} &= 0 \\
2u^T R + 2x^T P(t)B &= 0
\end{aligned} \tag{2.63}$$

de donde se deduce:

$$\begin{aligned}
2u^T R &= -2x^T P(t)B \\
u &= -R^{-1}B^T P(t)x
\end{aligned} \tag{2.64}$$

Sustituyendo (2.63) en (2.62) se tiene:

$$\begin{aligned}
 0 &= x^T \frac{\partial P(t)}{\partial t} x + x^T Q x + x^T P(t) B R^{-1} B^T P(t) x + 2x^T P(t) [A x - B R^{-1} B^T P(t) x] \\
 0 &= x^T \frac{\partial P(t)}{\partial t} x + x^T Q x - x^T P(t) B R^{-1} B^T P(t) x + 2x^T P(t) A x \\
 0 &= x^T \frac{\partial P(t)}{\partial t} x + x^T Q x - x^T P(t) B R^{-1} B^T P(t) x + x^T [P(t) A + A^T P(t)] x \\
 0 &= x^T \left\{ \frac{\partial P(t)}{\partial t} + Q - P(t) B R^{-1} B^T P(t) + P(t) A + A^T P(t) \right\} x \\
 \text{con } &: P(t_1) = 0
 \end{aligned} \tag{2.65}$$

es decir:

$$\begin{aligned}
 -\frac{\partial P(t)}{\partial t} &= Q - P(t) B R^{-1} B^T P(t) + P(t) A + A^T P(t) \\
 \text{con } &: P(t_1) = 0
 \end{aligned} \tag{2.66}$$

que es una *ecuación matricial de Riccati*, donde  $P(t)$  se encuentra resolviendo ésta ecuación.

Por lo tanto, el *control óptimo* del problema planteado por la ecuación (2.57) es:

$$u(t) = -R^{-1} B^T P(t) x(t) \tag{2.67}$$

o bien, se puede reescribir como:

$$u(t) = -K x(t) \quad K = R^{-1} B^T P(t). \tag{2.68}$$

donde ésta ley de control resuelve el problema de estabilización; la solución  $P(t) \geq 0$  de la ecuación de Riccati (2.66) es llamada *solución de estabilización*; y la matriz  $K$  definida en (2.68) es la *matriz de ganancias del control óptimo* o la *matriz de estabilización óptima*. Con una apropiada selección de peso de las matrices  $Q$  y  $R$  es posible tener un comportamiento apropiado del sistema en lazo cerrado, ya que las propiedades dinámicas y magnitudes del estado y las variables de control son de mucha influencia. Por lo último, considerando de nuevo el sistema lineal (2.57) y la entrada de control óptimo (2.68), se puede verificar el comportamiento de éste último analizando la ecuación en lazo cerrado:

$$\dot{x}(t) = [A(t) - B(t)K]x(t). \tag{2.69}$$

Ante de iniciar con la síntesis de los controles, debemos conocer los parámetros físicos del robot móvil y asignar los valores necesarios para diseñar los controles óptimos correspondientes, de acuerdo a la tabla :

| Variable | valor   | Descripción                                             |
|----------|---------|---------------------------------------------------------|
| v0       | 0.20    | Velocidad deseada [m/s].                                |
| w0       | 0.0     | Velocidad angular deseada, [rad/s].                     |
| a        | 0.075   | Distancia entre las ruedas [m].                         |
| b        | 0.08    | Distancia del centro de masa del eje de las ruedas [m]. |
| m        | 0.725   | Masa del robot móvil [kg].                              |
| R        | 0.03365 | Radio de las ruedas activas [m].                        |
| J        | 0.0018  | Tensor de inercia del cm de las ruedas.                 |
| Js       | 0.02    | Tensor de inercia del cm de la plataforma.              |

**Tabla 2.3:** Variables físicas.

Con estos valores físicos de la plataforma se tiene que el sistema lineal correspondiente a la trayectoria recta horizontal en sentido positivo a una velocidad lineal de  $v_0 = 0.15m/s$  obtenemos:

$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{y}} \\ \dot{\tilde{\alpha}} \\ \dot{\tilde{V}} \\ \dot{\tilde{\Omega}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -178.191 & 890.955 & -104878.0 \\ 0 & 0 & 178.191 & 890.955 & 104878.0 \\ 0 & 0 & 0 & 0 & 18540.0 \\ 0 & 0 & 0 & -0.000155565 & 0 \\ 0 & 0 & 0 & 0 & -31.5001 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{\alpha} \\ \tilde{V} \\ \tilde{\Omega} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.228692 & 0 \\ 0 & 0.00182135 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad (2.70)$$

Además es necesario verificar la controlabilidad completa de los sistemas lineales correspondientes a cada una de las trayectorias deseadas, ya que es la condición de existencia de un control óptimo. Para el sistema lineal invariante en el tiempo (2.70) correspondiente a una trayectoria recta, empleando el teorema 1.8 tenemos una orientación  $\alpha$  constante ( $\alpha = \alpha^*$ ) en el intervalo de tiempo  $[\tilde{t}, t_1]$ ,  $0 < \tilde{t} < t_1$  el control deseado (la velocidad angular  $\Omega^0 = 0$ ) es  $u^d = 0$  obtenemos:

$$y^d = \begin{bmatrix} a_0 + (t - t_1)V_0 \cos(\alpha^*) \\ b_0 + (t - t_1)V_0 \sin(\alpha^*) \\ \alpha^* \\ V_0 \\ 0 \end{bmatrix} \quad (2.71)$$

Siendo así, la ecuación 2.57 para la línea recta toma la siguiente forma:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (2.72)$$

donde  $x(t) = y(t) - y^d(t)$ ,  $\Delta u(t) = u(t) - u^d(t) = u(t)$ ,

$$A = \begin{bmatrix} 0 & 0 & -\delta_1 V_0 \sin(\alpha^*) & \delta_1 V_0 \cos(\alpha^*) & -\delta_2 b \sin(\alpha^*) \\ 0 & 0 & \delta_1 V_0 \cos(\alpha^*) & \delta_1 V_0 \sin(\alpha^*) & \delta_2 b \cos(\alpha^*) \\ 0 & 0 & 0 & 0 & \delta_2 \\ 0 & 0 & 0 & -\frac{2\delta_4 \delta_7}{\delta_9} & 0 \\ 0 & 0 & 0 & 0 & -\left(\delta_5 b V_0 + 2\frac{\delta_6 \delta_8}{\delta_9}\right) \end{bmatrix}, \quad (2.73)$$

$$B = \frac{\partial f}{\partial u}(y^d, u^d) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2\frac{\delta_4}{\delta_9} & 0 \\ 0 & 2\frac{\delta_6}{\delta_9} \end{bmatrix} \quad y \quad \Delta u = \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \end{bmatrix} \quad (2.74)$$

Siendo las matrices  $A, B$  del sistema lineal 2.72 constantes, la dimensión de la matriz de controlabilidad  $U = (B, AB, A^2B, A^3B, A^4B)$  es  $5 \times 10$ , entonces tenemos:

$$AB = \begin{bmatrix} 2\frac{\delta_1 \delta_4}{\delta_9} \cos(\alpha^*) & -\frac{2b\delta_2 \delta_6}{\delta_9} \sin(\alpha^*) \\ 2\frac{\delta_1 \delta_4}{\delta_9} \sin(\alpha^*) & \frac{2b\delta_2 \delta_6}{\delta_9} \cos(\alpha^*) \\ 0 & 2\frac{\delta_2 \delta_6}{\delta_9} \\ -\frac{4\delta_4^2 \delta_7}{\delta_9^2} & 0 \\ 0 & -\frac{2\delta_6}{\delta_9} \left(bV_0 \delta_5 + 2\frac{\delta_6 \delta_8}{\delta_9}\right) \end{bmatrix}$$

$$A^2B = \begin{bmatrix} -\frac{4\delta_1 \delta_4^2 \delta_7}{\delta_9^2} \cos(\alpha^*) & 2(V_0 \delta_5 \delta_9 b^2 + 2\delta_6 \delta_8 b - V_0 \delta_1 \delta_9) \frac{\delta_2 \delta_6}{\delta_9^2} \sin(\alpha^*) \\ -\frac{4\delta_1 \delta_4^2 \delta_7}{\delta_9^2} \sin(\alpha^*) & -2(V_0 \delta_5 \delta_9 b^2 + 2\delta_6 \delta_8 b - V_0 \delta_1 \delta_9) \frac{\delta_2 \delta_6}{\delta_9^2} \cos(\alpha^*) \\ 0 & -2(V_0 \delta_5 \delta_9 b + 2\delta_6 \delta_8) \frac{\delta_2 \delta_6}{\delta_9^2} \\ \frac{8\delta_4^3 \delta_7^2}{\delta_9^3} & 0 \\ 0 & \frac{2\delta_6}{\delta_9^3} (V_0 \delta_5 \delta_9 + 2\delta_6 \delta_8)^2 \end{bmatrix}$$

$$A^3B = \begin{bmatrix} \frac{8\delta_1 \delta_4^3 \delta_7^2}{\delta_9^3} \cos(\alpha^*) & -\frac{2\delta_2 \delta_6}{\delta_9^3} (2\delta_6 \delta_8 + V_0 \delta_5 \delta_9 b) (V_0 \delta_5 \delta_9 b^2 + 2\delta_6 \delta_8 b - V_0 \delta_1 \delta_9) \sin(\alpha^*) \\ \frac{8\delta_1 \delta_4^3 \delta_7^2}{\delta_9^3} \sin(\alpha^*) & \frac{2\delta_2 \delta_6}{\delta_9^3} (2\delta_6 \delta_8 + V_0 \delta_5 \delta_9 b) (V_0 \delta_5 \delta_9 b^2 + 2\delta_6 \delta_8 b - V_0 \delta_1 \delta_9) \cos(\alpha^*) \\ 0 & \frac{2\delta_2 \delta_6}{\delta_9^3} (2\delta_6 \delta_8 + V_0 \delta_5 \delta_9 b)^2 \\ -\frac{16\delta_4^4 \delta_7^3}{\delta_9^4} & 0 \\ 0 & -\frac{2\delta_6}{\delta_9^4} (2\delta_6 \delta_8 + V_0 \delta_5 \delta_9 b)^3 \end{bmatrix}$$

$$A^4 B = \begin{bmatrix} \frac{-16\delta_1\delta_4^4\delta_7^3}{\delta_9^4} \cos(\alpha^*) & \frac{2\delta_2\delta_6}{\delta_9^4} (2\delta_6\delta_8 + V_0\delta_5\delta_9b)^2 (V_0\delta_5\delta_9b^2 + 2\delta_6\delta_8b - V_0\delta_1\delta_9) \sin(\alpha^*) \\ \frac{-16\delta_1\delta_4^4\delta_7^3}{\delta_9^4} \sin(\alpha^*) & \frac{-2\delta_2\delta_6}{\delta_9^4} (2\delta_6\delta_8 + V_0\delta_5\delta_9b)^2 (V_0\delta_5\delta_9b^2 + 2\delta_6\delta_8b - V_0\delta_1\delta_9) \cos(\alpha^*) \\ 0 & \frac{-2\delta_2\delta_6}{\delta_9^4} (2\delta_6\delta_8 + V_0\delta_5\delta_9b)^3 \\ \frac{32\delta_4^5\delta_7^4}{\delta_9^5} & 0 \\ 0 & \frac{2\delta_6}{\delta_9^5} (2\delta_6\delta_8 + V_0\delta_5\delta_9b)^4 \end{bmatrix}$$

Se verificó que el rango de la matriz de controlabilidad  $U$  es 5, en virtud del Teorema 1.8 el sistema 2.72 es completamente controlable y por el Teorema 1.13 también es estabilizante.

Como se esta analizando la trayectoria en linea recta, sabemos que la orientación deseada durante un  $t$  determinado deberá ser de cero, así como la velocidad angular, y  $y(t) = 0$  para todo  $t \in [0, t_1]$ , de esta forma logramos tener  $y^d$  de la siguiente manera:

$$y^d = \begin{bmatrix} V_0 * t \\ 0 \\ 0 \\ V_0 \\ 0 \end{bmatrix}, \quad t \in [0, t_1]$$

Tomando en cuenta esto, las ecuaciones en desviaciones 2.54 quedan de la siguiente forma:

$$\begin{aligned} \Delta \dot{x} &= \delta_1 \Delta V \\ \Delta \dot{y} &= \delta_1 V_0 \Delta \Omega + b \delta_2 \Delta \Omega \\ \Delta \dot{\alpha} &= \delta_2 \Delta \Omega \\ \Delta \dot{V} &= -2 \frac{\delta_4 \delta_7}{\delta_9} \Delta V + 2 \frac{\delta_4}{\delta_9} \Delta u_1 \\ \Delta \dot{\Omega} &= - \left( b \delta_5 V_0 + 2 \frac{\delta_6 \delta_8}{\delta_9} \right) \Delta \Omega + 2 \frac{\delta_6}{\delta_9} \Delta u_2 \end{aligned} \tag{2.75}$$

En el análisis de sistemas de control óptimo convencionales, la matriz de ganancia de estabilización óptima  $K$  en (2.68), es diseñada por aproximación seleccionando las matrices de peso de acuerdo al razonamiento físico del sistema; y a causa de la complejidad de éstos, las matrices de peso  $Q$  y  $R$  son elegidas comúnmente como matrices diagonales. Estas matrices usualmente son determinadas a prueba y error hasta obtener un desempeño satisfactorio, pero en sistemas grandes no es fácil determinarlas. Para la solución del control óptimo de las dos trayectorias deseadas seleccionadas, la matriz de peso  $R$  se tomará como identidad, con lo cuál se asigna igual peso a las señales de control y se logra que cada una de ellas actúe en forma independiente; la matriz de peso  $Q$  será el problema a resolver para un buen comportamiento en la estabilidad del sistema.

## Consideraciones en la solución del control óptimo.

La solución original de un algoritmo de control óptimo, se basa en encontrar la entrada de control dada por la expresión (2.68) siendo la matriz  $P(t)$  de  $5 \times 5$  la solución desde un tiempo  $t_0 = 0$  hasta un tiempo final  $t_1$  de la ecuación diferencial de Riccati con condiciones finales como se muestra en la ecuación (2.66) donde  $(Q = Q^T) \geq 0$  es una matriz de peso de  $2 \times 2$  simétrica definida positiva ( $R = R^T \geq$ ) sujeta a la ecuación lineal correspondiente a alguna trayectoria como lo muestra la expresión:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (2.76)$$

Se hace énfasis en que (2.66) es una ecuación diferencial ordinaria matricial y por lo tanto involucra a  $n$  ecuaciones diferenciales acopladas. Partiendo la condición de controlabilidad completa cumplida, se debe encontrar la solución de estabilización la cual es el problema principal en la teoría de control óptimo. Como solo se conoce el valor o condición final de (2.66), entonces ésta debe resolverse integrando *hacia atrás* en el tiempo, para esto primero es necesario transformar la ecuación diferencial matricial de Riccati de condiciones finales a iniciales, esto con la finalidad de poder resolverla por métodos convencionales. Para lograrlo se realizara un cambio de variable para el tiempo, aplicando la nueva variable *falsa* de tiempo en retroceso<sup>1</sup>

$$\tau = t_1 - t \quad (2.77)$$

con un diferencial tenemos:

$$d\tau = -dt \quad (2.78)$$

Por lo que haciendo la sustitución en la ecuación (2.66), se tiene la siguiente expresión diferencial de Riccati con condiciones iniciales:

$$\begin{aligned} \frac{\partial P(\tau)}{\partial \tau} &= Q - P(\tau)B(\tau)R^{-1}B^T(\tau)P(\tau) + P(\tau)A(\tau) + A^T(\tau)P(\tau) \\ \text{con } P(t_0) &= 0 \end{aligned} \quad (2.79)$$

Este sistema matricial diferencial se puede resolver mediante la aplicación de métodos numéricos, por ejemplo Runge-Kutta, con los cuales se obtienen como resultados gráficas de las soluciones. La solución de la ecuación diferencial de Riccati es un sistema de 25 elementos, pero considerando de que ésta sea una matriz simétrica:

$$P(t) = P(t)^T = \begin{bmatrix} p_1(t) & p_2(t) & p_3(t) & p_4(t) & p_5(t) \\ p_2(t) & p_6(t) & p_7(t) & p_8(t) & p_9(t) \\ p_3(t) & p_7(t) & p_{10}(t) & p_{11}(t) & p_{12}(t) \\ p_4(t) & p_8(t) & p_{11}(t) & p_{13}(t) & p_{14}(t) \\ p_5(t) & p_9(t) & p_{12}(t) & p_{14}(t) & p_{15}(t) \end{bmatrix} \quad (2.80)$$

<sup>1</sup>Se considera que el tiempo evoluciona hacia atrás; por ejemplo,  $t_0 = 0$  seg a  $t_1 = 10$  seg, con ésta nueva variable se estaría realizando una integración de  $t_1$  a  $t_0$ , es decir hacia atrás en el tiempo.

con lo cuál sólo sería necesario resolver simultáneamente un sistema de 15 ecuaciones diferenciales. Una vez obtenido el conjunto de gráficas de las soluciones estas se aproximan mediante polinomio, y ya que estas soluciones dependen de la variable de tiempo en retroceso, se tienen que cambiar a la variable de tiempo en evolución normal para buscar la idea del que sería la solución de este problema.

Con la solución  $P(t)$  de la ecuación matricial diferencial de Riccati es posible obtener el control (2.68). Y finalmente aplicando este control al sistema lineal (2.76), obtenemos la expresión para el control en lazo cerrado.

$$\dot{x}(t) = [A(t) - B(t)K]x(t). \quad (2.81)$$

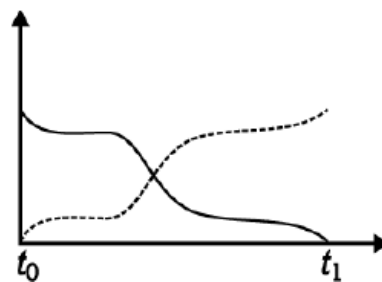
Con esto se puede verificar el comportamiento del sistema bajo la señal de control, mediante la solución de éste sistema empleando también los métodos numéricos, obteniendo gráficas de solución que muestran el comportamiento para cada variable de estado del sistema.

Como se vio anteriormente, un buen desempeño del control óptimo diseñado depende significativamente de la elección de las matrices de peso  $Q$  y  $R$ , y por cuestiones de complejidad, es común elegir a éstas matrices de la forma diagonales, para el caso de la matriz  $R$  se elegirá igual de igual forma, con esto se asigna el mismo peso a las señales de control y se logra que cada una de ellas actúe independientemente. La elección de la matriz de peso  $Q$  es mas compleja, y se puede realizar mediante un proceso iterativo a prueba y error hasta lograr un comportamiento deseado. Para esto, inicialmente se resuelve la ecuación matricial diferencial de Riccati (2.79) con la matriz  $Q$  igual a la identidad para verificar el comportamiento del control, y mediante la solución del sistema en lazo cerrado (2.81) con condiciones iniciales diferente de cero, tenemos que si el comportamiento no es el deseado se modificaran los valores para la matriz  $Q$ . Un buen comportamiento será aquel en el cual los estados del sistema lineal converjan asintóticamente a cero, o lo más próximos en un tiempo mínimo.

### Problemas en la solución del control óptimo.

Las consideraciones anteriores ponen en manifiesto los siguientes problemas que deben ser resueltos en el cálculo de la solución de estabilización del control óptimo lineal.

- La sintonía o búsqueda de los elementos de la matriz de peso  $Q$  puede ser tediosa



**Figura 2.7:** Cambio de condiciones iniciales a finales de la solución de Riccati.



e imprecisa, y se tendría que repetir éste laborioso procedimiento si se desean obtener los controles óptimos para las trayectorias restantes.

- La solución de la ecuación diferencial de Riccati, que es la parte esencial del control óptimo, es sólo para un tiempo específico, es decir, no se puede utilizar éste control para un tiempo mayor, por lo que sería necesario calcular otro para el tiempo deseado.
- Aunque es posible aumentar el grado de los polinomios de aproximación para aumentar su precisión, en ocasiones no siempre es bueno, ya que aún teniendo una mejor aproximación de la solución de la ecuación de Riccati, la solución del sistema en lazo cerrado puede presentar comportamientos caóticos. Además, al final es difícil la visualización para la solución de la matriz de Riccati o para la matriz de control óptimo, debido a la gran cantidad de términos que forman un elemento de éstas.

Estos problemas motivaron la investigación y propuesta de métodos alternativos para poder solucionar estos inconvenientes, los cuales se muestran a continuación; se presenta un método para sistemas lineales invariantes en el tiempo.

### Simulación lineal recta.

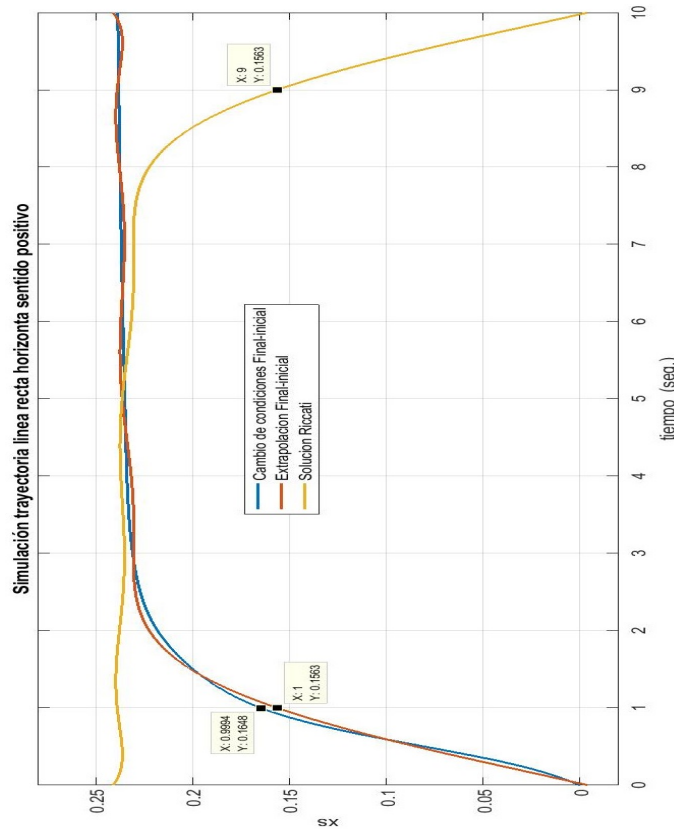
Así, con el planteamiento anterior se puede obtener un algoritmo que de solución para los sistemas lineales invariantes en el tiempo, implementado en *MATLAB* para obtener solución de estabilización  $\bar{P}$  de la ecuación algebraica de Riccati (2.66) y el método integración numérico de Runge-Kutta de 5° orden con la instrucción *ode45* para la solución de la ecuación de control en lazo cerrado. Considerado para la trayectoria recta horizontal positiva representada por la expresión (2.70).

Tomamos el par de matrices  $(A, B)$  del sistema lineal invariante en el tiempo (2.70), a una velocidad lineal deseada  $v_0 = 0.15$  m/s para el robot móvil, teniendo así:

$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{y}} \\ \dot{\tilde{\alpha}} \\ \dot{\tilde{V}} \\ \dot{\tilde{\Omega}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -178.191 & 890.955 & -104878.0 \\ 0 & 0 & 178.191 & 890.955 & 104878.0 \\ 0 & 0 & 0 & 0 & 18540.0 \\ 0 & 0 & 0 & -0.000155565 & 0 \\ 0 & 0 & 0 & 0 & -31.5001 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{\alpha} \\ \tilde{V} \\ \tilde{\Omega} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.228692 & 0 \\ 0 & 0.00182135 \end{bmatrix} \quad (2.82)$$

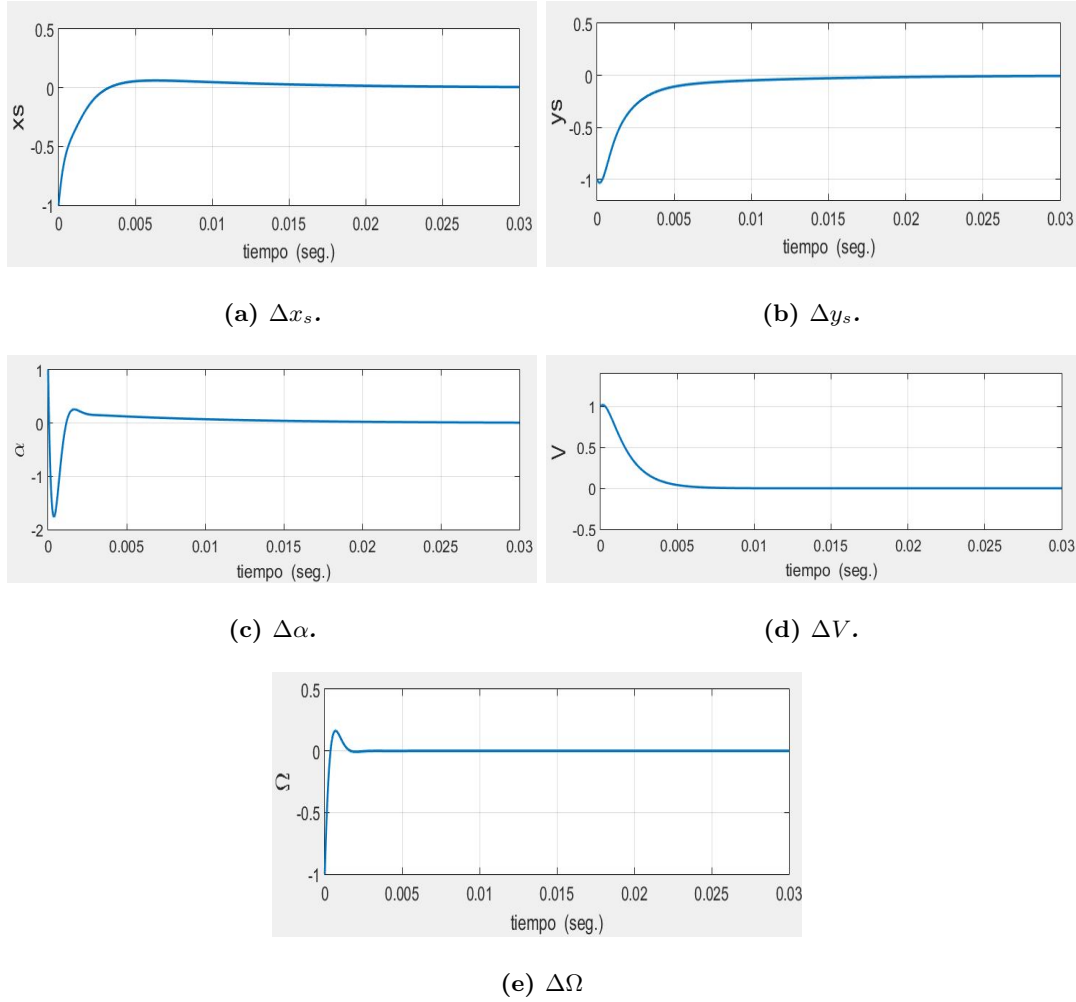
y mediante los teoremas 1.3, 1.4 y 1.6, se puede observar que el sistema lineal en lazo abierto es *inestable*. Como se menciono anteriormente, el objetivo de este método de solución es encontrar un algoritmo de control óptimo que haga que todos los valores característicos sean asignados a una región vertical en el semiplano izquierdo del plano complejo, lo cual haría que el sistema en lazo cerrado sea estable asintóticamente.

La solución de la ecuación diferencial de Riccati es un sistema de 25 elementos, por la simetría de  $P(t)$ , entonces sólo es necesario resolver un sistema de 15 ecuaciones diferenciales. Estas 15 ecuaciones se puede resolver con ayuda de MATLAB (*ode45*), obteniendo como resultado 15 conjuntos de puntos en el plano por donde pasan las curvas integrales. Uniendo estos puntos se tiene una aproximación de las gráficas, como se muestra en la figura donde se obtiene la solución y se efectúa la integración de la ecuación de Riccati. Una vez que se obtiene el conjunto de gráficas éstas se pueden aproximar, por ejemplo, mediante polinomios de grado 8. Dichas soluciones están en tiempo inverso, se invierten para buscar la solución a condiciones finales, como se muestra en 2.7, la idea de la solución para este problema. Con la aproximación de la solución  $P(t)$  de (2.68) es posible encontrar el control óptimo.



**Figura 2.8:** Cambio de condiciones iniciales a finales de la solución de Riccati.

Finalmente aplicando este control a (2.54) se obtiene la ecuación en lazo cerrado (2.69) con la cual se puede verificar el comportamiento del sistema bajo la señal de control. mediante la solución de este sistema empleando también la herramienta de MATLAB (*ode45*), se obtienen gráficas que muestran el comportamiento de cada variable del sistema (2.49).



**Figura 2.9:** Simulación del comportamiento de las variables de estado

La importancia que radica en la simulación es el hecho de verificar que efectivamente la solución y aplicación del control obtenido haga que los estados sean asintóticamente a cero, como lo podemos observar en las figuras 2.9. Donde recordemos que estas son las variables que nos dan la posición del robot móvil es decir, se les aplico una condición inicial,  $-1$  a cada estado, es como decir que se encontraban inicialmente en una posición fuera de la trayectoria deseada, y al inicial a efectuar el movimiento estas variables a cada instante que avanza el tiempo tienen un comportamiento asintótico al origen es decir a la trayectoria de la línea recta deseada. Estos comportamientos recordemos que se trata del sistema en lazo cerrado y lo que se desea es que las ecuaciones sea asintóticamente estables es decir reducir su error a cero, como se muestra en las gráficas 2.9.

## 2.4. Síntesis del control óptimo mediante LMI's.

Existen diversos tipos de sistemas descritos por los términos lineal, no lineal, invariantes y variantes en el tiempo. Y son clasificados de acuerdo a la forma de sus expresiones algebraicas de estado como lo menciona [17] y [18]. Si un sistema es clasificado como no lineal y variante en el tiempo, sus ecuaciones de estado son de la forma:

$$\dot{x}(t) = a(x(t), u(t), t). \quad (2.83)$$

Si se trata de un sistema como lineal y variante en el tiempo, sus ecuaciones de estado son como se muestra en la ecuación donde  $A(t)$  y  $B(t)$  son matrices de  $n \times n$  y  $n \times m$  con elementos variantes en el tiempo.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t). \quad (2.84)$$

### Descripción del método.

Como se mencionó anteriormente, existen sistemas lineales variantes en el tiempo, los cuales están descritos por la ecuación de estados diferencial:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (2.85)$$

y sistemas lineales invariantes en el tiempo descritos por la ecuación:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (2.86)$$

Éstos últimos, aparecen con mayor frecuencia en la literatura especializada de los sistemas de control y por tal razón existe una muy amplia teoría asociada al análisis y síntesis de controladores para este tipo de sistemas. Por el contrario para lineales variantes en el tiempo poseen una teoría muy reducida y restringida a problemas con estructuras particulares.

### Sistemas lineales con parámetros variantes

Es necesario además, introducir otro tipo de sistemas existentes, estos son denominados lineales con parámetros variantes. Dichos sistemas dependen explícitamente de un vector de parámetros variantes en el tiempo  $\theta(t)$  que puede medirse en tiempo real, éste parámetro es descrito para todo  $t \geq 0$  por la ecuación:

$$\begin{aligned} \dot{x}(t) &= A(\theta(t))x(t) + B(\theta(t))u(t), \\ \text{con} \quad &: \theta : \mathbb{R}^+ \rightarrow \mathbb{R}^r, \quad A : \mathbb{R}^r \rightarrow \mathbb{R}^{n \times n}, \quad B : \mathbb{R}^r \rightarrow \mathbb{R}^{n \times m} \end{aligned} \quad (2.87)$$

Este modelo permite tomar en cuenta las variaciones paramétricas de un sistema dinámico (temperatura, masa, concentración, velocidad, presión dinámica, ángulo de ataque en un avión, etc.), el modelo evoluciona en función de una trayectoria paramétrica admisible, ésta última es tal que cada uno de sus puntos pertenece en todo instante de tiempo al conjunto compacto  $\Omega \subset \mathbb{R}^r$ ; es decir, el vector de parámetros satisface en todo momento la condición:

$$\theta(t) \in \Omega = \{\theta(t) : \underline{\theta}_i \leq \theta_i(t) \leq \bar{\theta}_i \forall i = 1, 2, \dots, r\} \subset \mathbb{R}^r, \quad (2.88)$$

siendo  $\underline{\theta}_i$  y  $\bar{\theta}_i$  las cotas inferior y superior respectivamente del parámetro  $\theta_i(t)$ . Ahora bien, el sistema lineal con parámetros variables puede interpretarse como una generalización de un sistema lineal invariante en el tiempo cuando la trayectoria paramétrica admisible es constante  $\theta(t) = \theta_0$ .

con lo anterior, se puede entonces mencionar que el objetivo del método desarrollado en este apartado, es encontrar una ley de control óptima para un sistema lineal incierto, descrito por la ecuación:

$$\dot{x}(t) = A(\cdot)x(t) + B(\cdot)(t)u(t), \quad (2.89)$$

donde las matrices  $A(\cdot)$  y  $B(\cdot)$  son inciertas y pertenecen a los conjuntos  $D_A$  y  $D_B$  respectivamente; para esto, se considerará que un sistema de esta clase está descrito en la ecuación (2.87), es decir, las matrices  $A$  y  $B$  dependen de un vector de parámetros variante en el tiempo  $\theta(t) = [\theta_1(t) \dots \theta_r(t)]^T \in \Omega \subset \mathbb{R}^r$ , el cual representa un politopo con  $l_\theta = 2^r$  vértices.

Adicionalmente se considera un conjunto de vectores, que representan las combinaciones extremas de los parámetros en el interior del conjunto compacto definido en la ecuación (2.88) dados por:

$$\theta^{(i)}(t) = \begin{bmatrix} \theta_1^{(i)}(t) \\ \vdots \\ \theta_r^{(i)}(t) \end{bmatrix} \quad i = 1, 2, \dots, N. \quad (2.90)$$

a partir de los vectores  $\{\theta^{(i)}\}_{i=1}^N$  se construyen modelos locales lineales invariantes en el tiempo para el sistema lineal con parámetros variantes haciendo:

$$(A_i, B_i) = (A(\theta^{(i)}), B(\theta^{(i)})), \quad i = 1, \dots, N. \quad (2.91)$$

Por lo que ahora es posible escribir el sistema lineal con parámetros variantes dentro del conjunto compacto  $\Omega$  a través del siguiente sistema politópico:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad A \in D_A, \quad B \in D_B; \quad (2.92)$$

con  $D_A$ ,  $D_B$  dominios de incertidumbre politópica definidos como:

$$\begin{aligned}
A \in D_A &\hat{=} \left\{ A \in \mathbb{R}^{n \times n} : A = \sum_{i=1}^N \alpha_i A_i, \quad \alpha_i \geq 0, \quad \sum_{i=1}^N \alpha_i = 1 \right\}, \\
B \in D_B &\hat{=} \left\{ B \in \mathbb{R}^{n \times m} : B = \sum_{i=1}^N \gamma_i B_i, \quad \gamma_i \geq 0, \quad \sum_{i=1}^N \gamma_i = 1 \right\},
\end{aligned} \tag{2.93}$$

Para los dominios de incertidumbre politópica expresados en la ecuación anterior, es equivalente la siguiente notación:

$$A \in C_0 A_1, A_2, \dots, A_N \tag{2.94}$$

donde  $C_0$  denota el cascarón convexo.

Las matrices  $A_i$  de la expresión (2.93) son concebidas como vértices y los escalares  $\alpha_i$  son los elementos de combinación convexa. Una de las grandes ventajas en describir los parámetros inciertos en forma politópica es la convexidad del conjunto resultante. Por lo tanto, es posible asegurar que, si un determinado conjunto de condiciones se cumplen también en la región. Un inconveniente de este enfoque es el problema de la explosión exponencial, debido a que el número de sistemas lineales invariantes está dada por una función del tipo de  $2^r$ , donde  $r$  es el número de parámetros inciertos en el sistema. Por ejemplo, para poner a prueba las condiciones del sistema con 3 elementos de incertidumbre, debe comprobar las mismas condiciones en  $2^3$  vértices, es decir, se deben probar las condiciones para 8 valores diferentes de la matriz  $A(\theta(t))$ .

### Ecuación algebraica de Riccati como LMI

Con lo planteado anteriormente, se puede entonces construir un sistema politópico convexo conformado por sistemas lineales invariantes en el tiempo a partir del sistema lineal de parámetros variantes, teniendo éste conjunto de sistemas lineales invariantes en el tiempo, podemos utilizar el método planteado anteriormente, es decir, asignar valores característicos. Sin embargo, no podemos aplicar este método como tal, esto debido a que si es aplicado, se deberán resolver y encontrar una solución de estabilización  $\bar{P}$  para cada uno de los sistemas lineales invariantes en el tiempo, en lugar de eso, se tiene que buscar una solución  $\bar{P}$  que satisfaga a todos éstos sistemas de manera simultánea, es decir, una solución óptima para todos los sistemas lineales invariantes en el tiempo.

Esto se realizará, haciendo la transformación de la ecuación algebraica de Riccati modificada en una LMI que a la vez representará el conjunto politópico convexo del sistema lineal incierto. La transformación se realiza aplicando el *Complemento de Schur* de la definición 1.24 entonces teniendo la ecuación algebraica modificada de Riccati:

$$0 = \bar{P}\tilde{A}_i + \tilde{A}_i^T \bar{P} - \bar{P}B_i R^{-1} B_i^T \bar{P} \tag{2.95}$$

la LMI asociada a ésta es:

$$\begin{pmatrix} \bar{P}\tilde{A}_i + \tilde{A}_i^T\bar{P} & \bar{P}B_i \\ B_i^T\bar{P} & -R \end{pmatrix} < 0 \quad (2.96)$$

y tomando en cuenta que:

$$\bar{P}\tilde{A}_i + \tilde{A}_i^T\bar{P} + \bar{P}B_iR^{-1}B_i^T\bar{P} < 0 \quad (2.97)$$

Como puede observarse, las ecuaciones (2.95) y (2.97) son diferentes, la solución de la ecuación algebraica modificada de Riccati da un resultado de estabilización  $\bar{P} > 0$  mientras que la solución de la LMI asociada a esta nos dará como resultado una solución de estabilización  $\bar{P} < 0$ ; como el algoritmo de control óptimo requiere que la solución de estabilización sea definida positiva, la solución obtenida por la LMI se multiplicará por  $-1$  para obtener el resultado buscado. Considerar de manera diferente la ecuación de Riccati se hace con el fin de no perder la generalidad de la solución de una LMI mostrada en cita [25]

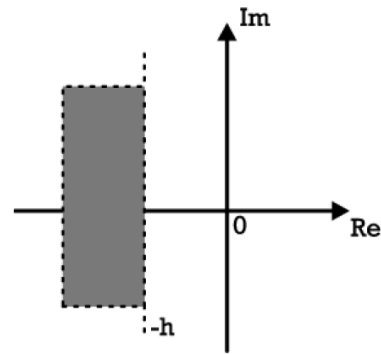
La solución de la ecuación algebraica de Riccati(2.95), es semejante a la solución de la LMI asociada a ésta, la cual se obtiene haciendo:

$$\min tr(\bar{P}) \text{ sujeto a } \begin{pmatrix} \bar{P}\tilde{A}_i + \tilde{A}_i^T\bar{P} & \bar{P}B_i \\ B_i^T\bar{P} & -R \end{pmatrix} < 0; \quad (2.98)$$

la cual se calculará mediante la ayuda del LMILab incluido en el software de MATLAB (un ejemplo de la solución de una LMI puede verse en [25]).

### Simulación de arco de circunferencia

Es preciso mencionar, que el método empleado para la obtención de un control óptimo del sistema lineal variante en el tiempo, correspondiente a una trayectoria circular en el IV cuadrante en sentido contrario del reloj, es muy similar al planteado para sistemas lineales invariantes en el tiempo, con la diferencia que su solución de estabilización  $\bar{P}$  se calcula ahora mediante una LMI para un sistema politópico descrito a partir del linealizado para la trayectoria que corresponde a el arco de circunferencia antes mencionado; además para éste caso, los valores característicos de los sistemas lineales invariantes en el tiempo resultantes, serán colocados a la izquierda de una línea vertical asignada en el semiplano complejo como se muestra en la figura y no en una región vertical.



**Figura 2.10:** Plano complejo S.

Tomando en cuenta el sistema lineal variante en el tiempo, con parámetros variantes, tenemos:

$$\begin{pmatrix} \dot{\tilde{x}}_s \\ \dot{\tilde{y}}_s \\ \dot{\tilde{\alpha}} \\ \dot{\tilde{V}} \\ \dot{\tilde{W}} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \theta_1(t) & \theta_3(t) & \theta_5(t) \\ 0 & 0 & \theta_2(t) & \theta_4(t) & \theta_6(t) \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -9.74594 & -0.0431678 \\ 0 & 0 & 0 & 3.86192 & -0.127443 \end{pmatrix} \begin{pmatrix} \tilde{x}_s \\ \tilde{y}_s \\ \tilde{\alpha} \\ \tilde{V} \\ \tilde{W} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 35.8955 & 0 \\ 0 & 13.8813 \end{pmatrix} \begin{Bmatrix} u_r \\ u_l \end{Bmatrix} \quad (2.99)$$

donde los parámetros están dados por:

$$\theta(t) = \begin{pmatrix} \theta_1(t) \\ \theta_2(t) \\ \theta_3(t) \\ \theta_4(t) \\ \theta_5(t) \\ \theta_6(t) \end{pmatrix} = \begin{pmatrix} -0.5 \sin(0.5t) - 0.0165 \cos(0.5t) \\ 0.5 \cos(0.5t) - 0.0165 \sin(0.5t) \\ \cos(0.5t) \\ \sin(0.5t) \\ -0.05 \sin(0.5t) \\ 0.05 \cos(0.5t) \end{pmatrix} \quad (2.100)$$

ahora bien teniendo identificados cada uno de los parámetros del sistema lineal variante en el tiempo, y tomando en cuenta la teoría descrita, se implementara un algoritmo que sera capaz de obtener las cotas inferiores y superiores para cada uno de los parámetros, para después construir el sistema politópico de  $2^6 = 64$  vértices para así obtener la solución de la LMI y finalizar con la estabilización y así tener el control buscado.

Si aplicamos el algoritmo para hallar el control óptimo del sistema lineal con parámetros variantes descrito en (2.99) correspondiente a una trayectoria circular de  $Radio = 1m.$ , en sentido contrario de las manecillas del reloj a una velocidad angular deseada de  $w_0 = 0.5 \frac{rad}{s}$ , primero es necesario verificar las gráficas que muestran el comportamiento de cada parámetro del sistema, y obtener las cotas tanto superior como inferior, las cuales se muestran a continuación:

$$\theta(t) = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{pmatrix} = \begin{pmatrix} [-0.5006, 0.5006] \\ [-0.5006, 0.5006] \\ [-1.0, 1.0] \\ [-1.0, 1.0] \\ [-0.05, 0.05] \\ [-0.05, 0.05] \end{pmatrix} \quad (2.101)$$

Con éste conjunto, se construye entonces un sistema politópico con  $2^6$  vértices, en los cuales existe un sistema lineal invariante en el tiempo; en otras palabras, se crea un conjunto de 64 sistemas lineales invariantes a partir de todas las combinaciones posibles de las cotas inferiores y superiores de los seis parámetros existentes en el sistema incierto, algunos de ellos son:



$$\begin{aligned}
 A_1 &= \begin{pmatrix} 0 & 0 & -0.05006 & -1 & -0.5 \\ 0 & 0 & -0.05006 & -1 & -0.5 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.9077 & -0.0470 \\ 0 & 0 & 0 & 12.8275 & 3.1167 \end{pmatrix}, & A_2 &= \begin{pmatrix} 0 & 0 & -0.05006 & -1 & -0.5 \\ 0 & 0 & -0.05006 & -1 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.9077 & -0.0470 \\ 0 & 0 & 0 & 12.8275 & 3.1167 \end{pmatrix}, \\
 A_3 &= \begin{pmatrix} 0 & 0 & -0.05006 & -1 & -0.5 \\ 0 & 0 & -0.05006 & 1 & -0.5 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.9077 & -0.0470 \\ 0 & 0 & 0 & 12.8275 & 3.1167 \end{pmatrix}, & \dots, A_{64} &= \begin{pmatrix} 0 & 0 & 0.05006 & 1 & 0.5 \\ 0 & 0 & 0.05006 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.9077 & -0.0470 \\ 0 & 0 & 0 & 12.8275 & 3.1167 \end{pmatrix}
 \end{aligned}
 \tag{2.102}$$

donde se tiene que los valores característicos en lazo abierto de cada uno de estos sistemas lineales invariantes en el tiempo son:

$$\det(I\lambda - A_i) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -0.7518 \\ 2.9609 \end{pmatrix} \tag{2.103}$$

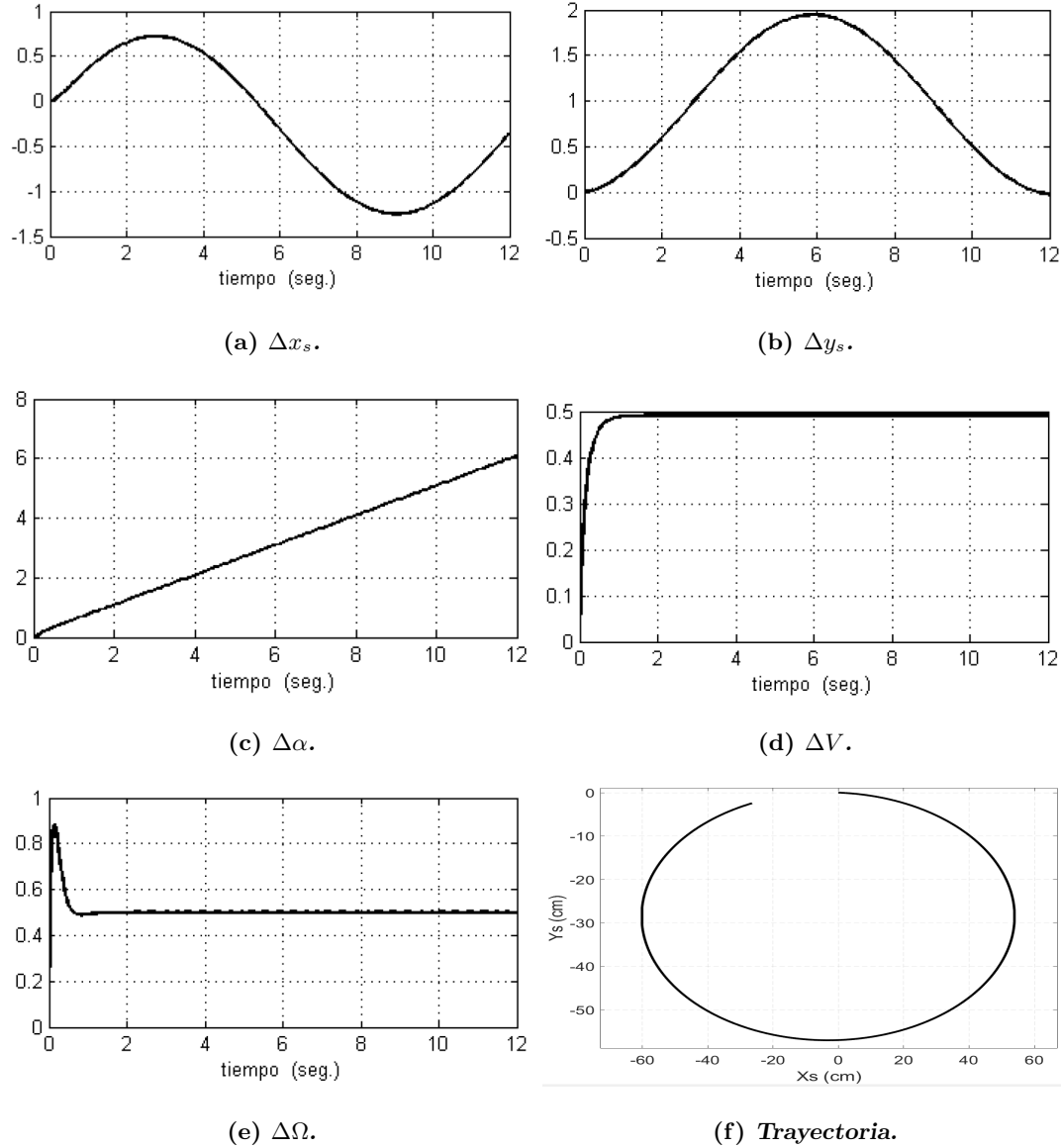
Por lo tanto, los sistemas lineales invariantes en el tiempo son *inestables*; y debido a esto, es necesario diseñar una ley de control que mejore la estabilidad de éstos sistemas para así asegurar su estabilización con parámetros variables. Escogiendo la línea vertical  $h = -6$  al lado izquierdo del plano complejo, y haciendo:

$$\min \operatorname{tr}(\bar{P}) \text{ sujeto a } \begin{pmatrix} \bar{P}\tilde{A}_i + \tilde{A}_i^T\bar{P} & \bar{P}B_i \\ B_i^T\bar{P} & -R \end{pmatrix} < 0; \tag{2.104}$$

Al obtener estos resultados será posible hallar la ley de control óptimo deseado, para realizar una trayectoria circular  $u(t) = \tilde{K}x(t)$ , apoyándonos con una herramienta de MATLAB para solucionar la LMI y así con el valor de las desviaciones está dado por:  $x(t) = y^d(t) - y(t)$

$$x(t) = \begin{pmatrix} \tilde{x}_s \\ \tilde{y}_s \\ \tilde{\alpha} \\ \tilde{V} \\ \tilde{W} \end{pmatrix} = \begin{pmatrix} \cos(0.5t) - x_s \\ \sin(0.5t) - y_s \\ 0.5(t) - \alpha \\ 0.5 - v \\ 0.5 - w \end{pmatrix}; \tag{2.105}$$

entonces podemos simular la ecuación lineal de control en lazo cerrado dada por  $\dot{x}(t) = (A - B\tilde{K})x(t)$  y proceder con la implementación del control óptimo para una trayectoria semi-circular.



**Figura 2.11:** Simulación del comportamiento de las variables de estado.

Con la utilización de las desigualdades matriciales lineales (LMIs), como solución del control óptimo para un sistema politópico convexo formado por sistemas locales lineales invariantes en el tiempo, construido a partir de un sistema lineal con parámetros variantes, se eliminó el problema que hacía imposible aplicar el método de asignación de valores característicos en una región específica estudiado con anterioridad, ya que éste sólo es aplicable a sistemas lineales invariantes en el tiempo

# Construcción Robot líder.

---

Para este capítulo se indicaran los aspectos que integran al robot líder, en una configuración diferencial como lo hemos estado abordando a lo largo de este trabajo, en esta plataforma basada en una tarjeta FPGA DE0nano se implementaran las leyes de control óptimo para la línea recta que se ha sintetizado.

Para que sea posible la construcción física y se logre cerrar el lazo de control del sistema, es necesario que al menos se cuente con dos elementos que son: un sensor y un elemento actuador.

Estos componentes tanto las señales que se emplean para que funcionen y pueda interpretarse la información que generan deben de ir interconectados a un hardware que brinde la posibilidad mediante algún lenguaje ser programado para interpretar y generar las señales requeridas para el control del sistema, en nuestro caso el robot líder.

### 3.1. Tarjetas FPGA.

El hardware contiene elementos esenciales de los cuales depende el funcionamiento fundamental del dispositivo electrónico en el cual se trabaja, para una computadora, por ejemplo son: la placa base (también conocida como Placa Madre), que es una placa de circuito impreso que aloja a la Unidad Central de Procesamiento (CPU) o microprocesador, Chipset (circuito integrado auxiliar), Memoria RAM, BIOS o Flash-ROM, etc., además de comunicarlos entre sí.

Las FPGAs son unos dispositivos nos permiten describir un circuito digital usando un lenguaje específico (los dos más comunes son VHDL y Verilog) y que tras cargarlo en el integrado, es creado físicamente en el chip. Su nombre es un acrónimo inglés que significa matriz de puertas reprogramable o Field Programmable Gate Array.

Internamente se componen principalmente de cables, puertas lógicas, biestables, y

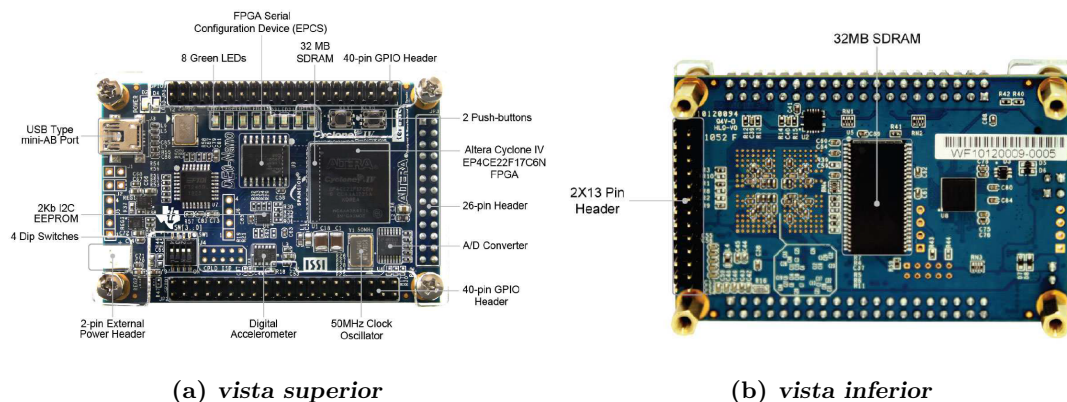
puertos de entrada y salida. Todo ello sin conectar, como una plantilla en blanco, hasta que se les carga un bitstream -un archivo generado a partir de la descripción del circuito.

## FPGA DE0-nano

La placa DE0-Nano presenta una plataforma de desarrollo de FPGA de tamaño compacto adecuada para diseños de circuitos de creación de prototipos como robots y proyectos "portátiles". La placa está diseñada para ser utilizada en la implementación más simple posible dirigida al dispositivo Cyclone IV hasta 22,320 LE.

El DE0-Nano tiene una colección de interfaces que incluyen dos cabeceras GPIO externas para extender los diseños más allá de la placa DE0-Nano, dispositivos de memoria incorporados que incluyen SDRAM y EEPROM para un mayor almacenamiento de datos y almacenamiento de cuadros, así como un periférico general de usuario con LED y Pulsadores (botones).

Las ventajas de la placa DE0-Nano incluyen su tamaño y peso, así como su capacidad de reconfiguración sin llevar hardware superfluo, lo que se distingue de otras placas de desarrollo de propósito general. Además, para los diseños móviles donde la potencia portátil es crucial, el DE0-Nano ofrece a los diseñadores tres opciones de combinación de energía que incluyen un puerto mini-AB USB, un cabezal de alimentación externa de 2 clavijas y dos clavijas de CC de 5V.



**Figura 3.1:** Fpga DE0-Nano diagrama de componentes.

Esta tarjeta fue seleccionada para la implementación del robot líder, una FPGA que nos brinda una versatilidad para generar módulos a medida que se vayan requiriendo a lo largo de las diversas pruebas, la implementación de manera modular nos permite realizar las correcciones adecuadas.

### 3.2. Motor DC.

Los motores de Corriente Directa o motor DC (correspondiente a las iniciales en inglés “direct current”) es también conocidos como motor de Corriente Continua o motor DC, son muy utilizados en diseños de ingeniería debido a las características torque-velocidad que poseen con diferentes configuraciones eléctricas o mecánicas.

Una gran ventaja de los motores de DC se debe a que es posible controlarlos con suavidad y en la mayoría de los casos son reversibles, responden rápidamente gracias a que cuentan con una gran razón de torque a la inercia del rotor.

Los detalles de los motores eléctricos han cambiado desde Faraday, sin embargo, los elementos fundamentales permanecen. El movimiento creado por un motor eléctrico obedece a la fuerza de Lorentz, de acuerdo con [29] y donde los servomotores y sensores son dispositivos fundamentales para el diseño y construcción de cualquier sistema mecatrónico y robot manipulador.

- Servomotores son sistemas electromecánicos que pertenecen a una clase particular de actuadores eléctricos encargados de transmitir energía para producir el movimiento del robot, y son los que forman las uniones o articulaciones del robot mediante las cuales éste se mueve, motivo por el cual al movimiento del robot se le denomina desplazamiento articular.
- La tecnología asociada a los servomotores es complicada, ya que su electrónica está basada en microprocesadores avanzados de procesamiento como los DSP's ó FPGA's para llevar a cabo el análisis y control de las diversas variables que determinan su modo de operación.
- Un transductor es un dispositivo que convierte un tipo de energía (es decir, energía de entrada o variable de medición) a otro tipo de energía medible (energía de salida, por lo común energía eléctrica en mili volts). Un ejemplo de transductor es la resistencia magnética que convierte el campo magnético en corriente eléctrica.

Un servomotor está compuesto principalmente por tres elementos:

- Motor eléctrico de corriente directa.
- Sensor de posición para medir el desplazamiento articular (rotacional o lineal).
- Amplificador electrónico (electronic driver), o servo amplificador constituido por un conjunto de microprocesadores que se encarga de enviar la señal de voltaje de baja potencia (comando de control) que proviene de la computadora, acopla al motor la impedancia eléctrica, y también incluye electrónica de potencia para suministrar la potencia que requiere el motor.

El motor que emplearemos es un motor de corriente directa de la marca Bringsmart como se muestra en la figura 3.2, 70 revoluciones por minuto, la operación típica del mismo es a 12V.



**Figura 3.2:** Motor con caja reductora y encoder de efecto hall.

Otorga un par de 2.17Nm en su operación de mayor eficiencia o hasta 2.42Nm en modo paro, con consumo de hasta 0.5 Amperes de acuerdo a su descripción dada en la siguiente tabla 3.1

| Modelo             | Voltaje (V)         |               | Corriente sin carga |               | Caja motorreductora |                  |
|--------------------|---------------------|---------------|---------------------|---------------|---------------------|------------------|
|                    | Rango operación     | Nominal       | Velocidad           | Corriente (A) | reducción           | Tamaño (L)       |
| Bringsmart 18-1930 | 6-12                | 12            | 70 RPS              | 0.2           | 1:150               | 6 * 2.5 cm       |
|                    | Corriente con carga |               | Maxima Eficiencia   |               |                     |                  |
|                    | Torque Kg.cm        | Corriente (A) | Velocidad           | Corriente (A) | Torque Kg.cm        | Potencia Nominal |
|                    | 0.680g.cm           | 0.650         | 40RPS               | 0.750         | 0.5                 | 2                |

**Tabla 3.1:** Información proporcionada por el vendedor.

Los servomotores tienen tres modos de operación:

*Modo posición* Permite mover al motor a una posición preestablecida, también conocida como set point. Sin embargo, no puede desplazar cargas o aplicar una fuerza determinada. En este modo se emplean reguladores simples como el proporcional derivativo (PD) y proporcional integral derivativo (PID).

*Modo velocidad.* Controla el movimiento del motor sobre una velocidad deseada. Tampoco puede ejercer fuerza como en el caso del modo de posición. Estos modos tie-

nen la característica de ser arquitectura cerrada, es decir, no permiten programar otro tipo de controladores, por lo que sus aplicaciones en robótica se encuentran limitadas.

*Modo par o torque* Esta configuración es ampliamente usada en robots industriales y sus principales características son:

Arquitectura abierta hace posible evaluar experimentalmente el desempeño y robustez de cualquier estrategia de control. De esta forma, las aplicaciones en robótica se incrementan. Permite la interacción dinámica con el sistema mecánico del robot, por lo que es posible compensar los efectos dinámicos del robot y en consecuencia controlar la posición o desplazamiento articular.

Control de la trayectoria y aplicaciones más complicadas como el control de impedancia, visual-servoing, tele-operación y control de fuerza.

### **Caracterización de motores.**

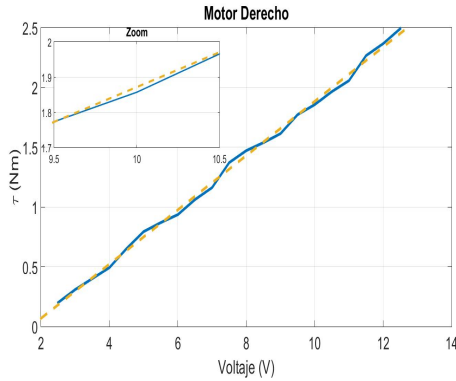
Considerando que el motor de DC, acoplado a una caja de engranes ayuda a aumentar su velocidad o torque, es necesario conocer para nuestro caso de estudio la fuerza que genera, con la aplicación del voltaje que otorgara la ley de control.

Para conocer el torque que genera el motor, se implementara el arreglo experimental conocido como freno prony, el cual consiste en fijar el motor de DC a una base y mediante una polea de radio conocido estará conectado a un dinamómetro, que al aplicar un voltaje continuo partiendo de 0.5 V hasta el máximo permitido, y se tendrá una fuerza máxima para cada valor de voltaje la idea de esta caracterización es conocer el máximo torque con el menor consumo de corriente.

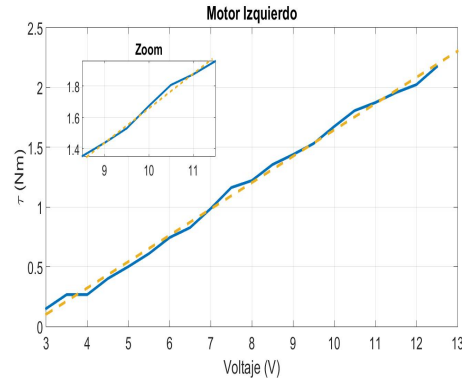
Con las mediciones además de conocer el torque que generan cada motor, es necesario identificar el voltaje nominal, y conociendo los valores de torque, es necesario obtener un índice que nos describirá cuanta potencia se esta inyectando a cada actuador y la energía mecánica entrega, esto es denominado la eficiencia del motor y es una parte fundamental para determinar el voltaje nominal, este se elige cuando la eficiencia es mayor y la mayor cantidad de torque por el voltaje aplicado.

### **Voltaje y frecuencia de operación.**

Una forma en que es posible identificar el voltaje nominal, es graficando el torque en función del voltaje, y en la región donde se note un comportamiento lineal sera alrededor de este valor la elección del voltaje nominal, de acuerdo con la gráfica, hemos seleccionado un valor de 10V.



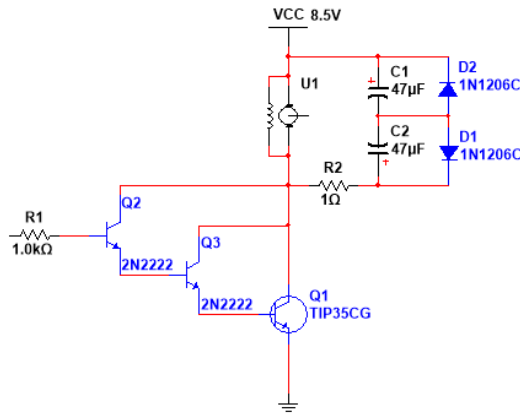
(a) Torque en función del voltaje.



(b) Torque en función del voltaje.

**Figura 3.3:** Selección de voltaje nominal para motores.

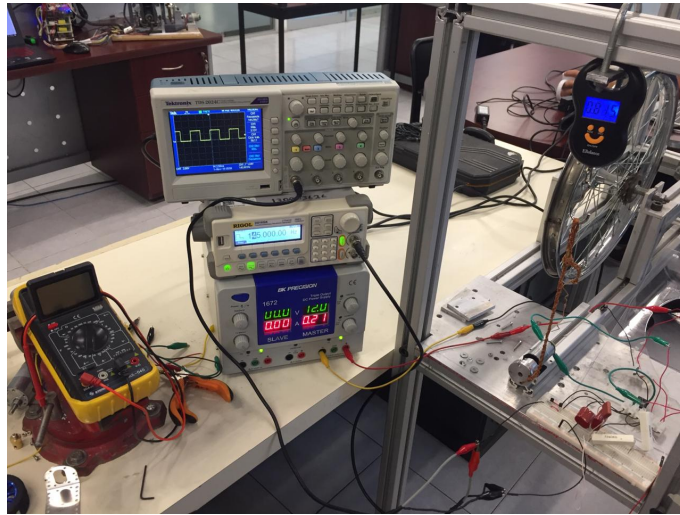
Considerando estos voltajes nominales, lo siguiente es encontrar la frecuencia de trabajo adecuada para que el motor entregue la mitad del torque máximo, considerando que la señal que se aplicara al motor es PWM, y así obtener una respuesta lineal.

**Figura 3.4:** Diagrama 1 rama puente H.

control de motores de DC.

Para esta etapa solo es necesario el giro del motor hacia un sentido, es por ello que se implementara el diagrama mostrado en la figura (3.4), el cual consiste en un arreglo darlington de transistores incorporando una red snuber, para la supresión de picos auto-inducidos por el motor, la implementación se muestra en la figura (3.5), las mediciones se muestran en la tabla , donde la frecuencia de trabajo para el motor derecho es de 195 Hz, y para el motor izquierdo tenemos un valor de 245 Hz.





**Figura 3.5:** Implementación para el barrido en frecuencia.

Para saber si la configuración Darlington es suficiente para poder proporcionar el máximo valor de corriente, por muy corto que sea el instante en que sea requerido, se calculo la corriente que deberá drenar cada uno de los transistores, considerando que la entrada de control será generada por una FPGA, que opera a 3.3V, se prevé un pico de de corriente al menos del doble del consumo nominal.

| Motor Derecho |        |            |
|---------------|--------|------------|
| % PWM         | F (Kg) | F (Nm)     |
| 1             | 0      | 0          |
| 2             | 0.03   | 0.05017815 |
| 3             | 0.06   | 0.1003563  |
| 4             | 0.06   | 0.1003563  |
| 5             | 0.065  | 0.10871933 |
| 6             | 0.07   | 0.11708235 |
| 7             | 0.08   | 0.1338084  |
| 8             | 0.095  | 0.15889748 |
| 9             | 0.1    | 0.1672605  |
| 10            | 0.11   | 0.18398655 |
| 15            | 0.165  | 0.27597983 |
| 20            | 0.255  | 0.42651428 |
| 25            | 0.345  | 0.57704873 |
| 30            | 0.44   | 0.7359462  |
| 35            | 0.5    | 0.8363025  |
| 40            | 0.565  | 0.94502183 |
| 45            | 0.65   | 1.08719325 |
| 50            | 0.74   | 1.2377277  |

| Motor Izquierdo |        |            |
|-----------------|--------|------------|
| % PWM           | F (Kg) | F (Nm)     |
| 5               | 0.03   | 0.05017815 |
| 6               | 0.03   | 0.05017815 |
| 7               | 0.04   | 0.0669042  |
| 8               | 0.05   | 0.08363025 |
| 9               | 0.055  | 0.09199328 |
| 10              | 0.075  | 0.12544538 |
| 11              | 0.085  | 0.14217143 |
| 12              | 0.1    | 0.1672605  |
| 13              | 0.12   | 0.2007126  |
| 14              | 0.125  | 0.20907563 |
| 15              | 0.13   | 0.21743865 |
| 20              | 0.2    | 0.334521   |
| 25              | 0.305  | 0.51014453 |
| 30              | 0.365  | 0.61050083 |
| 35              | 0.44   | 0.7359462  |
| 40              | 0.485  | 0.81121343 |
| 45              | 0.52   | 0.8697546  |
| 50              | 0.615  | 1.02865208 |

Tabla 3.2 (continue)

| Motor Derecho |        |            | Motor Izquierdo |        |            |
|---------------|--------|------------|-----------------|--------|------------|
| % PWM         | F (Kg) | F (Nm)     | % PWM           | F (Kg) | F (Nm)     |
| 55            | 0.81   | 1.35481005 | 55              | 0.695  | 1.16246048 |
| 60            | 0.875  | 1.46352938 | 60              | 0.75   | 1.25445375 |
| 65            | 0.9    | 1.5053445  | 65              | 0.84   | 1.4049882  |
| 70            | 0.97   | 1.62242685 | 70              | 0.925  | 1.54715963 |
| 75            | 1.03   | 1.72278315 | 75              | 1.015  | 1.69769408 |
| 80            | 1.13   | 1.89004365 | 80              | 1.215  | 2.03221508 |
| 85            | 1.225  | 2.04894113 | 85              | 1.25   | 2.09075625 |
| 90            | 1.38   | 2.3081949  | 90              | 1.35   | 2.25801675 |
| 95            | 1.415  | 2.36673608 | 95              | 1.4    | 2.341647   |
| 99            | 1.45   | 2.42527725 | 99              | 1.45   | 2.42527725 |

Tabla 3.2: Barrido de ciclo efectivo de señal PWM.

Ya conocemos el valor nominal del voltaje que emplearemos para obtener la mayor eficiencia en torque de los motores, así como la frecuencia de trabajo para cada uno de ellos.

### 3.3. Pulse-Width Modulation (PWM).

Para poder controlar la potencia destinada al motor DC, se va a utilizar la modulación por ancho de pulsos conocida como PWM (pulse-width modulation), el PWM de una señal o fuente de energía, consiste en una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal, o una cuadrada como en este caso), ya sea para transmitir información a través de un canal de comunicaciones, o bien, para controlar la cantidad de energía que se envía a una carga (motor DC como en este caso).

Puede tener varios objetivos, como tener el control de la energía que se proporciona a una carga o llevar a cabo la transmisión de datos.

Si el ciclo de trabajo es del 25 % se pasa el 25 % de su periodo arriba y el 75 % abajo. El periodo es lo que dura la onda sin repetirse. Por eso se va repitiendo con el tiempo porque el periodo se repite durante todo el tiempo. El periodo es la suma de la parte alta y baja una vez, cuando vuelve a subir ya es otro periodo y la onda vuelve a empezar otra vez, dada la siguiente expresión.

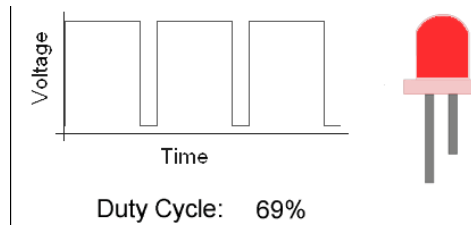


Figura 3.6: PWM por ancho de pulso.

$$D = \frac{\sigma}{T} \quad (3.1)$$

donde:  $D$  = el ciclo de trabajo,  $\sigma$  = tiempo en que la función es positiva del ancho del pulso, y  $T$  = periodo de la función.

De esta manera podremos limitar o controlar de cierta forma la energía que se suministra al motor de dc.

## Módulo PWM

Con los resultados obtenidos en la caracterización de los motores, conocemos los voltajes de operación donde obtenemos la mayor eficiencia en cuanto a torque, para ambos actuadores así como su frecuencia de operación es por ello que debemos modificar las frecuencias para la generación de la señal PWM para que cada motor opere de la manera mas eficiente posible.

Modificaremos los procesos encargados de la frecuencia base para las señales de control de cada motor, y de igual manera que se implemento en el módulo UART, utilizaremos identificadores que serán enviados desde el computador a la tarjeta de desarrollo FPGA para saber el ancho del pulso correspondiente al valor analógico que deberá ser aplicado a cada actuador, y cumplir el objetivo de la ley de control correspondiente sea para una linea recta o semicircular.

Como la tarjeta de desarrollo cuenta con un reloj de 50 MHz, utilizaremos la siguiente relación para modificar el divisor de frecuencia y aplicar el valor correspondiente de cada uno de los actuadores.

$$Num_{pulsos} = \frac{50MHz}{Hz_d}, \quad (3.2)$$

donde  $Num_{pulsos}$  = Numero de pulsos que se necesitan para cambiar de valor de 1 lógico a 0 y viceversa,  $Hz_d$  = Frecuencia deseada, entonces para las frecuencias de 245Hz y 195Hz tenemos:

$$\begin{aligned} Num_{pulsos} &= \frac{50MHz}{245Hz} = \frac{50000000}{245} = 204081 \\ Num_{pulsos} &= \frac{50MHz}{195Hz} = \frac{50000000}{195} = 256410 \end{aligned} \quad (3.3)$$

Garantizando así que las frecuencias son adecuadas para cada actuador, como hemos visto en los avances de tesis I, y II el control esta dado por:

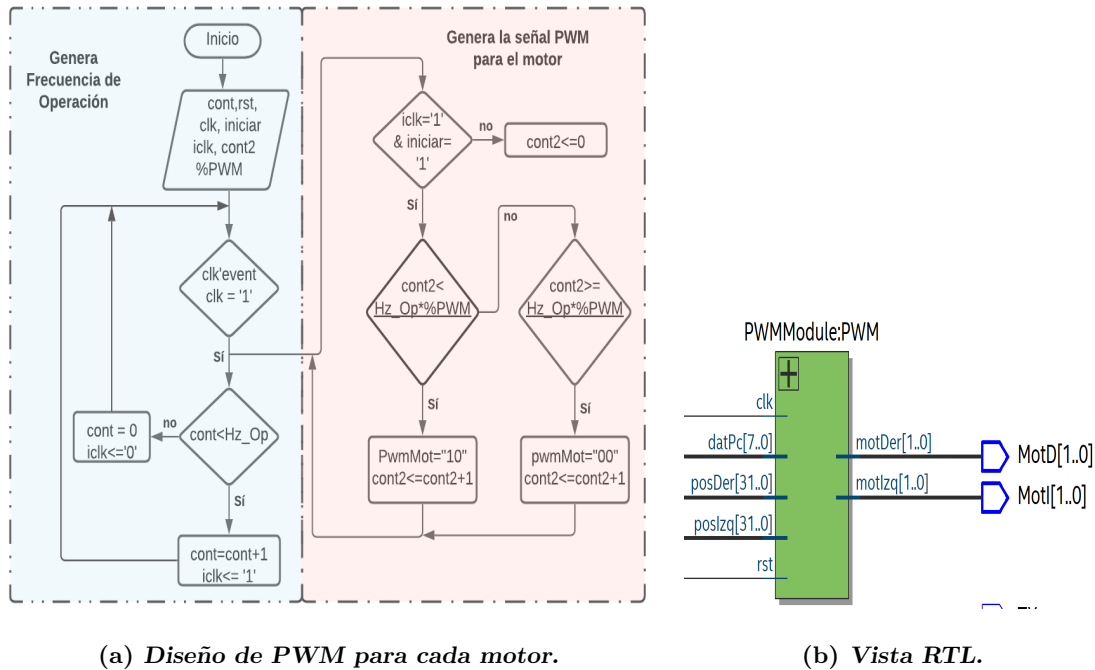
$$\tilde{u} = u_i + u_o, \quad (3.4)$$

donde  $\tilde{u}$  es la suma del control inicial  $u_i$ , el voltaje necesario para iniciar el movimiento del robot móvil, y el control óptimo  $u_o \frac{\partial f(y^d, u^r)}{\partial u}$ , el voltaje que se calculara

en cada instante de tiempo para compensar la desviación del móvil de la trayectoria deseada.

De acuerdo con esto, se le asignara un voltaje inicial correspondiente al necesario para que el robot móvil inicie su desplazamiento, el cual es de un voltaje de 1.5V, y el valor que se reciba con el identificador “1” sera el correspondiente al motor derecho y con el valor “0” como identificador sera para el motor izquierdo, es así como tendremos valores para la modulación de voltaje por ancho de pulso.

Este modulo se diseña con el siguiente diagrama de flujo para ser implementado en VHDL, como se muestra en las figuras 3.7(a) y 3.7(b) así como el código puede ser consultado en B.7 :



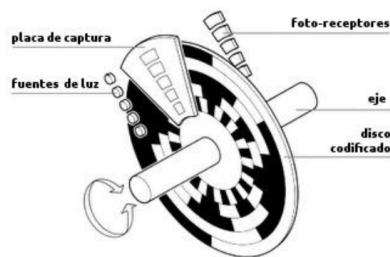
**Figura 3.7:** Diagrama de flujo y Modulo en VHDL, de frecuencias y voltajes de operación.

Es importante señalar que el tiempo que se emplea para enviar la información de los encoder's al computador y el calculado del valor asignado por el control para la trayectoria deseada, no debe ser ni mayo o menor, es decir, deberá de ser el mínimo necesario para que al mandar la información desde el dispositivo FPGA al computador este realice los cálculos requeridos para enviar la información para el ancho del PWM para cada motor, esto para evitar que se acumule información en el Bus del computador y al calcular ya no corresponda a la posición actual del móvil, si no de varios instantes de tiempo atrás, y al mandar el valor del PWM, ya no corresponda teniendo un mal desempeño en su trayectoria.

### 3.4. Codificadores rotatorios.

Los codificadores rotatorios (conocidos genéricamente como encoders) son mecanismos utilizados para entregar la posición, velocidad y aceleración del rotor de un motor, convirtiendo estas magnitudes físicas en un código digital. Sus principales aplicaciones son en robótica, lentes fotográficas, industriales que requieren medición angular, equipos electrónicos que requieran movimiento controlado de ejes, etc.

Los tipos más comunes de Encoders se clasifican en: absolutos y relativos (conocidos también como incrementales). Los encoders absolutos pueden venir codificados en binario o gray ver figura 1.25. Dentro de los encoder incrementales, se encuentran los encoder en cuadratura, ampliamente utilizados en motores de alta velocidad y en aplicaciones en las que interesa conocer la dirección del movimiento del eje además de la velocidad.



**Figura 3.8:** Encoder con disco codificado para código binario.

El tipo común de encoder incremental consiste de un disco acoplado al eje del motor que contiene un patrón de marcas o ranuras que son codificados por un interruptor óptico (par led/fotodiodo o led/ fototransistor) generando pulsos eléctricos cada vez que el patrón del disco interrumpe y luego permite el paso de luz hacia el interruptor óptico a medida que el disco gira. Esto produce una secuencia que puede ser usada para

controlar el ángulo de giro, la dirección del movimiento e incluso la velocidad.

Un parámetro importante de los encoder incrementales son los pulsos por revolución que es un parámetro que determina la resolución del encoder. Desde un encoder incremental no se puede determinar la posición angular absoluta del eje.

Para poder determinar la posición relativa a un punto de referencia (cero), el encoder incremental debe incluir una señal adicional que genera un pulso por revolución, denominada índice.

#### Encoder absoluto.

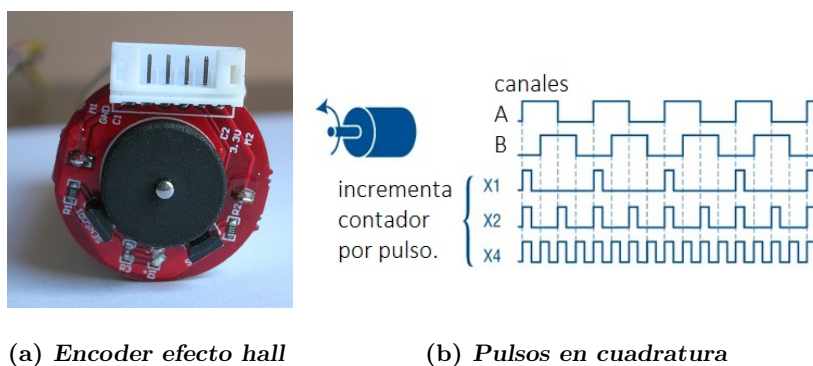
Los encoder absolutos emiten un valor numérico codificado único para cada posición del eje. En las tareas de posicionamiento particularmente, los encoder absolutos liberan al controlador de las tareas de cálculo e impiden el aumento de los costes al

no necesitarse componentes de entrada adicionales, La figura 3.8 es una muestra de un disco codificado de un encoder absoluto.

Además, ya no es necesario ejecutar el punto de referencia tras una caída de tensión o cuando la máquina se conecta, ya que se proporciona de inmediato el valor de posición actual. Los encoder absolutos en paralelo transmiten el valor de posición a la electrónica de análisis a través de varios cables en paralelo. Los encoder absolutos en serie transmiten los datos de salida por medio de protocolos e interfaces estandarizadas.

### Encoder incremental de cuadratura.

Los encoders incrementales de cuadratura se caracterizan por tener 2 canales de salida, comúnmente llamados “canal A” y “Canal B” desfasados 90 grados entre sí, ver figura 1.26b. se puede saber la velocidad de rotación y la posición mediante conteos con un solo canal el cual generalmente es el A, sin embargo es imposible saber cuál es la dirección del movimiento sino se dispone de otro canal para la comparación en este caso el canal B, de este modo el sentido de giro está determinado por el nivel de alguna de las señales de salida con respecto de la segunda, por ejemplo, en el modo 1X (modo normal), si la señal en A durante la transición de bajada (A : 100) el nivel lógico de la señal B está en Alto (B=1) implica que está girando en el sentido de las manecillas del reloj (CW), mientras que si B está en la transición de bajada y A=1 implica que está girando en sentido anti horario CCW ver figura 3.10 correspondiente al modo 1X.



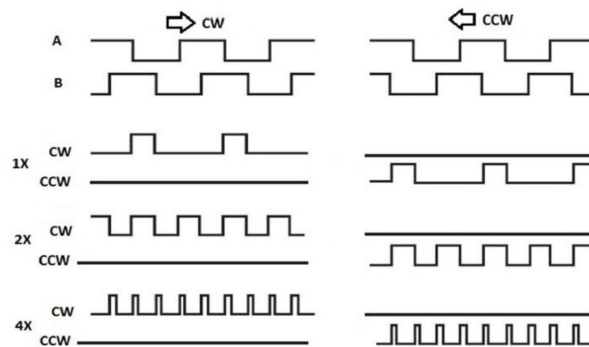
**Figura 3.9:** Encoder de cuadratura por efecto hall.

Una característica importante de los encoders incrementales es el número de pulsos por revolución, abreviado PPR, esta da lugar a la resolución del encoder que es sumamente importante a la hora de tratar con aplicaciones que requieran un posicionamiento preciso del eje. Entre más PPR se tengan se tiene una mejor resolución.

## Decodificación del encoder.

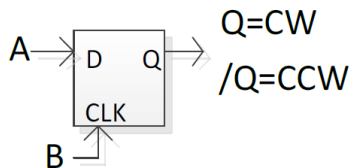
Las señales de cuadratura pueden ser decodificadas para medir desplazamiento angular y la dirección de rotación como se muestra en la figura 3.10, los pulsos de salida se muestran en columnas separadas como rotación en dirección de las manecillas de reloj (Clock Wise, CW) y rotación en sentido anti horario (Counterclockwise, CCW).

La decodificación de las transiciones A y B se realiza usando circuitos lógicos secuenciales. Estos pueden ofrecer tres diferentes resoluciones 1X, 2X y 4X. La resolución 1X proporciona un pulso de salida en cada flanco negativo (o positivo si se requiere) de la señal A o B, lo que resulta en un solo pulso para cada ciclo o periodo de la señal de cuadratura. La resolución 2X ofrece un pulso de salida en cada flanco positivo y negativo de la señal A o B, lo cual resulta en el doble de pulsos de salida. La resolución 4X ofrece un pulso de salida en cada flanco positivo y negativo de las señales A y B, dando lugar a 4 veces el número de pulsos de salida.



**Figura 3.10:** Salidas del encoder y decodificación en modos 1X, 2X, 4X.

Determinación del sentido de movimiento. La información correspondiente al desplazamiento se obtiene directamente de las señales A o B, un ciclo de la señal corresponde al mínimo avance, se puede usar como referencia el flanco de subida o bajada; para un encoder de 1600 pulsos por revolución el mínimo avance corresponde a  $0.225^\circ$ .

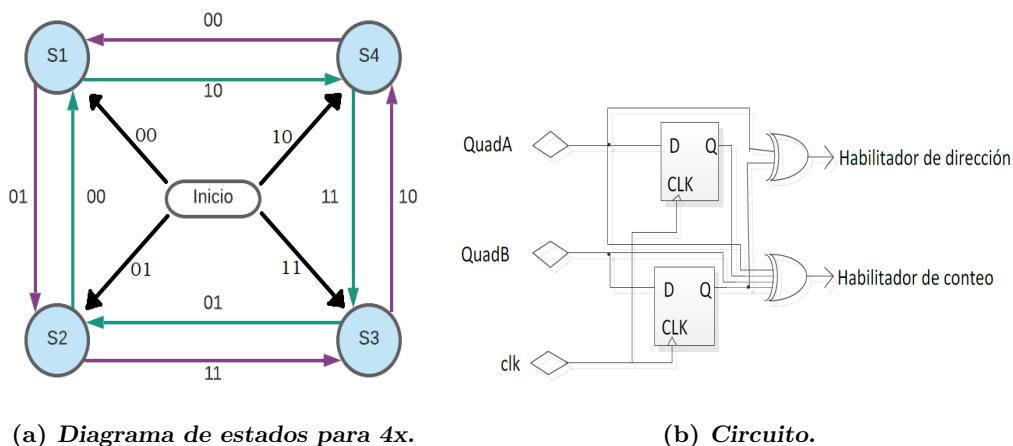


**Figura 3.11:** Decodificador de dirección 1x.

Para determinar la dirección del desplazamiento se requiere de ambas señales (A y B); en la figura 3.11 se tiene un circuito a través del cual se determina el sentido del desplazamiento. Determinación del sentido de movimiento. La información correspondiente al desplazamiento se obtiene directamente de las señales A o B, un ciclo de la señal corresponde al mínimo avance, se puede usar como referencia el flanco de subida o bajada.

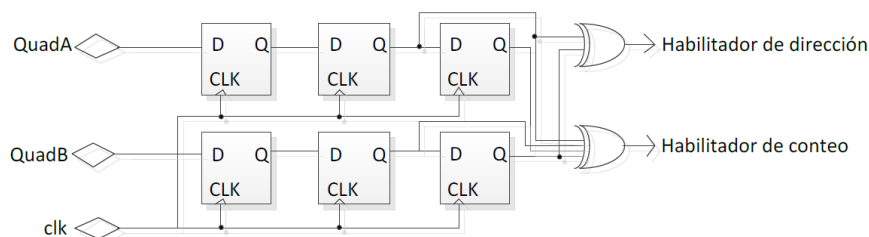
Para determinar la dirección del desplazamiento se requiere de ambas señales (A y B); en la figura 3.11 se tiene un circuito a través del cual se determina el sentido del desplazamiento.

La decodificación más básica Se implementa con un Flip-Flop “D”, la señal A se emplea como dato y B como señal de reloj, en el sentido CW (izquierda a derecha) o de color verde como se muestra en la figura 3.12(a), se captura continuamente un nivel alto, esto es porque el flanco de subida para B coincide con el nivel alto de A como se muestran las señales de cuadratura en la figura 3.12(b). Para el sentido CCW (derecha a izquierda) o de color púrpura como se muestra en la figura 3.12(a) el flanco de subida para B coincide ahora con el nivel bajo de A.



**Figura 3.12:** Verde Sentido horario y púrpura anti-horario, decodificación para 4X.

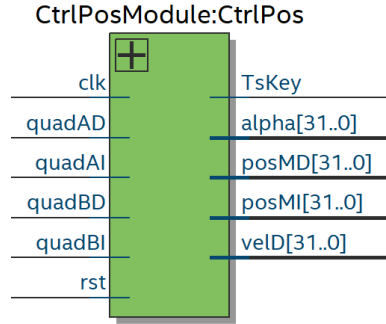
Este circuito es fácilmente programable en VHDL sin embargo en la práctica las señales en cuadratura no son sincronicas con el reloj de la FPGA, la solución clásica es la integración de 2 flip flop D extras con el fin de agregar estabilidad al circuito, con lo que se obtiene el circuito de la figura 3.13.



**Figura 3.13:** Decodificador de cuadratura.



Cada uno de los canales correspondiente a lo encoders del motor izquierdo y derecho, van conectados a la FPGA en los pines dedicados del GPIO, y son las entradas en la estructura para generar un flip-flop, como se muestra a continuación:



**Figura 3.14:** Módulo para decodificación de encoder y generación de estados.

Las salidas que nos entrega este modulo, servirán para generar los errores correspondientes de los estado del sistema del robot móvil con respecto a la trayectoria deseada, y los datos generados por los encoders, que serán enviados por el modulo de comunicación a través del dispositivo bluetooth.

Donde quadAI es la entrada del canal A de uno de los encoders y se aplica el mismo codigo pero cambiando la entrada a quadBI, correspondientes al motor izquierdo, y la salidad QI1, es procesada como entrada en un proceso idéntico generando QI2,..QI6, para poder realizar las operaciones logicas *XOR* correspondientes en la figura 3.12(b).

### 3.5. Módulo comunicaciones UART

La implementación de un modulo de comunicaciones capaz de transmitir mediante un protocolo UART (RS-232), por sus siglas en ingles (Universal Asynchronous Receiver-Transmitter), siendo transmitidos 8 bits. Estos datos contienen la información del numero de pulsos que se van acumulando conforme giran los motores, gracias a la caracterización que se realizo podemos saber con certeza el numero de pulsos para cada uno de ellos, se realizo está validación debido a que por experiencia se sabe que las especificaciones de este tipo siempre presentan variaciones, y esta ocasión no fue la excepción. Para el motor derecho tenemos que en promedio se registraron 5400 pulsos por vuelta completa para el motor derecho, y para el izquierdo tenemos un total de 5200.

Si bien es una cantidad considerable a la hora de realizar la conversión de decimal a binario para poder ser enviada por el modulo de comunicaciones, tenemos una gran limitante, esto debido a que el protocolo solo envía paquetes de datos de 8 bits cada uno, entonces el máximo numero que podría transmitirse sera  $2^8 = 256$ , es un valor por

debajo de los pulsos que son generados en una vuelta de los motores, además recordemos que fue implementado un arreglo de flip-flops para tener una resolución x4 veces el número de pulsos que arroja el encoder.

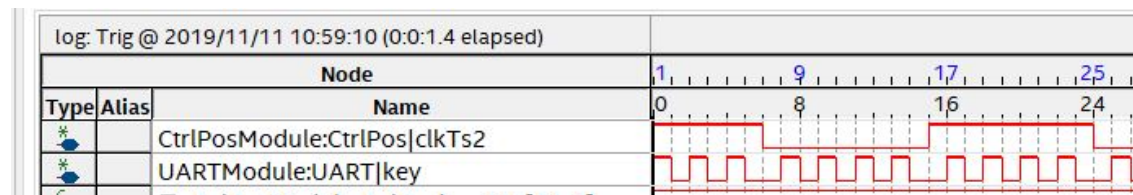
Para solucionar este inconveniente se propuso lo siguiente: como no es posible modificar el protocolo, pero si enviar el mayor numero de paquetes antes que se guarden los siguientes valores correspondientes al encoder de cada motor. Esto nos permitirá trabajar con números mucho mayores a los que se generan en una vuelta de cada actuador. Para realizar esto se debe considerar que cada paquete debe de tener un identificador para lograr ser concatenados de manera adecuada al ser procesados por el computador, y que estos valores sean los enviados por la FPGA, con esto se propone que sean 4 paquetes cada motor.

Para realizar este proceso se debe considerar una frecuencia adecuada para que sea posible enviar 8 paquetes por el modulo UART, conocemos el tiempo que tarda la transmisión de un solo bit y esta dado por:

$$T_b = \frac{1}{\text{Baudrate}} = \frac{1}{9600} = 104.2\mu S \quad (3.5)$$

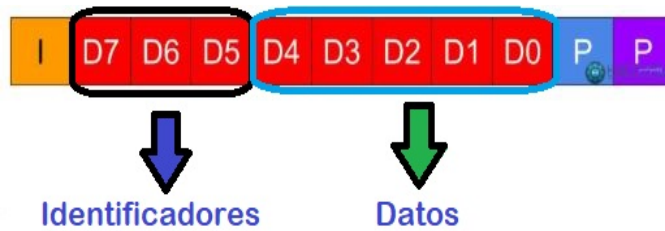
La Figura , muestra este tipo de trama, esta consta de 11 bits que se transmiten cada uno a una velocidad de 9600 bits/s. Es decir, que a cada bit le toma 1/9600 s transmitirse para cada flanco de subida del generador de BAUD. La taza de transferencia de esta configuración serial de  $9600/11 = 872.72$  Bytes/segundo.

Entonces el modulo UART debe de contar con un divisor de frecuencia a 0.1s para que en cada flanco de subida, se habilite que guarde el valor del contador que va incrementando con cada pulso del encoder, y ahí permanezca, para que este valor quede almacenado, el flanco de subida del reloj sera la señal que habilite un flip-flop y que el valor sea guardado hasta el próximo flanco de subida, como se muestra en la figura



**Figura 3.15:** Envío 4 paquetes de datos por ciclo.

Ahora ya solo queda generar un algoritmo en el modulo de comunicaciones que sea capaz de dividir en 4 paquetes el valor total del número almacenado a la salida del flip-flop, esto se logra generando un identificador para cada paquete el cual se dispondrá de 3 bits mas significativos en la cadena de 8, quedando de la siguiente forma:

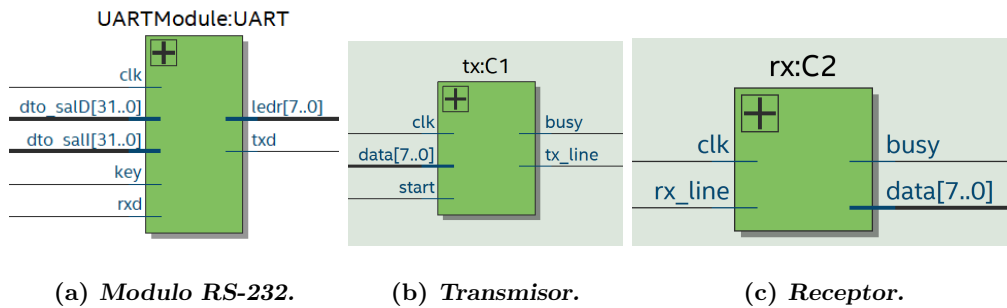


**Figura 3.16:** Bits de identificación y datos en trama de 8 bits UART.

La división del valor total será de la siguiente manera, el número que se encuentre en el registro, sera convertido en una cadena de bits, y se iniciara de los menos significativos a los mas significativos para ser enviados, y solo usaran los 5 bits menos significativos, los tres restantes serán para identificar la posición para reconstruir el valor original así y a que motor corresponde la lectura.

$$\begin{aligned} \text{del valor } 000 \text{ al } 011 &= \text{paquete con información del motor izquierdo} \\ \text{del valor } 100 \text{ al } 111 &= \text{paquete con información del motor derecho} \end{aligned} \quad (3.6)$$

Con esto aseguramos que sean enviada la información de cada encoder y que sea dentro del tiempo en que cambie el valor, en la figura 3.17 se ejemplifica en vista RTL (Vista a nivel esquemático) del modulo de comunicaciones *RS – 232* y el código se indica en B.4, para el modulo principal UART y B.5 , B.6 para los módulos rx y tx respectivamente.



**Figura 3.17:** Vista esquemática del Módulo de comunicación

Este modulo recibirá los comandos desde un ordenador, estas indicaciones están ligadas con acciones como: iniciar, detener y el porcentaje de ancho del pulso que sera aplicado a cada motor.

## Conectividad bluetooth.

BlueTooth o BT es una especificación tecnológica para transmisión de voz y datos entre distintos dispositivos a una frecuencia de 2.4GHz. Existen equipos o dispositivos

electrónicos como celulares, tabletas, computadoras, vehículos, etc., que tienen incluida esta tecnología, pero cuando no se tiene disponible, se puede agregar un módulo comercial como se muestra en la figura a un sistema.

Para nuestro trabajo se selecciono un modulo HC-06, los módulos HC-05 y HC-06 son Bluetooth V2. Pero con el tiempo han ido apareciendo módulos como los que conocemos que soportan el protocolo Bluetooth V4.0 o Low Energy al alcance de todos los bolsillos y los fabricantes chinos han empezado a suministrarlos de forma accesible, tales como los modelos HC-08 y HC-10, los pines de conexión serán conectados a la FPGA respetando los voltajes que manejan tanto la tarjeta como este dispositivo que es de 3.3 v.



**Figura 3.18:** Módulos de BT comerciales.

### 3.6. Implementación.

La estructura que donde es implementado el robot líder en configuración diferencial, posee dos ruedas activas donde cada una de ellas esta acoplada a un motor de DC independiente y una rueda pasiva.

Para la base del robot, se empleo una lamina de acrílico con un espesor de 3 mm, la cual es liviana y rígida para soportar a todos los componentes. Como puede observarse en la figura 3.19(a).

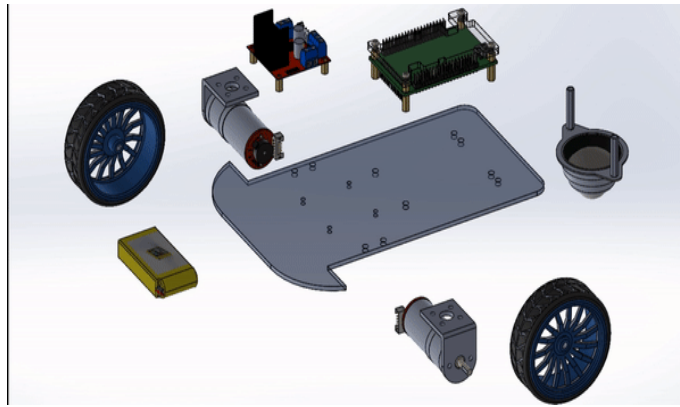


**Figura 3.19:** Motores de DC acoplados con las ruedas y rueda loca.

Cada uno de los motores que se especificaron en la tabla 3.1 tienen acoplada una rueda o llanta con un diámetro de 6.6 cm, los motores son acoplados a la placa de acrílico con unos soportes en forma de L, siendo fijados con tornillos y tuercas como se muestra en la figura 3.19(c).

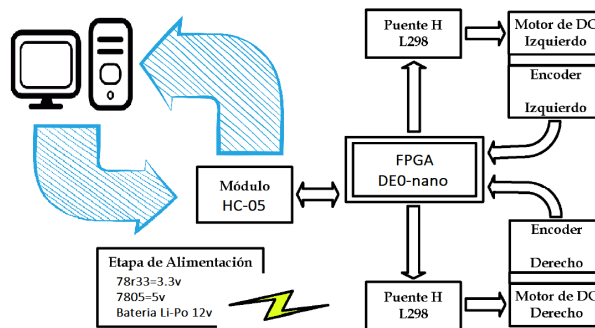
Con la rueda pasiva se colocó una rueda loca comercial como se muestra en la figura 3.19(d) siendo ajustada a la altura necesaria con postes hexagonales y tornillos para lograr que la plataforma se encuentre siempre en paralelo al piso.

Finalmente se colocaron las tarjetas electrónicas y componentes necesarios para realizar la interconexión de todos ellos, para la etapa de potencia de los motores se empleó un módulo basado en un L298N el cual contiene un regulador de voltaje a 5v, y para el voltaje que emplean tanto la tarjeta de desarrollo FPGA DE0nano, los encoders y el módulo Bluetooth se emplea un regulador de voltaje 78r33 que provee 3.3v y tenemos listo nuestro robot líder como se muestra en la figura 3.20.



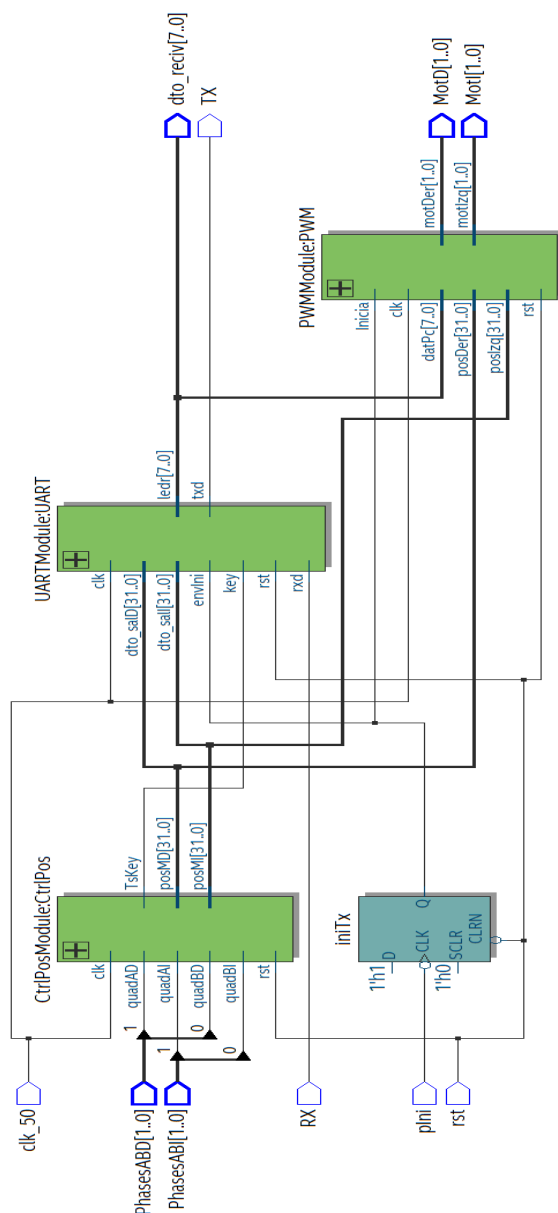
**Figura 3.20:** Diseño de robot Líder en CAD (SolidWorks 2018).

En el siguiente diagrama se muestra la interconexión de los diversos módulos y componentes de manera general.



**Figura 3.21:** Diagrama a bloques del robot líder.

Considerando la complejidad para su implementación de las leyes obtenidas de control óptimo que han sido diseñados y el cálculo de las variables involucradas de estado, el robot líder será tele-operado por un sistema de control centralizado, implementado en una PC, como se muestra en la figura 3.21.



**Figura 3.22:** Vista RTL de módulos implementados en FPGA.

Los bloques implementados en la tarjeta DE0nano se muestran en la figura 3.22, cada modulo diseñado en un lenguaje denominado descripción de hardware, cada uno interconectado de acuerdo al diagrama a bloques presentado en la figura 3.21, permitiendo el procesamiento de las señales que genera cada uno de los sensores acoplados al eje de cada motor (encoder), para ser enviada tal información vía inalámbrica por el modulo Bluetooth y recibir los valores que serán aplicados a los módulos PWM que controlan a los actuadores de DC.

## Construcción Robot Esclavo

---

En este capítulo se desarrolla el sistema de conducción autónomo mediante la visión artificial. Este robot está integrado por una cámara que proveerá las imágenes que al ser procesadas por un algoritmo, se detectará una figura de color definido indicando distancia y ubicación para guiar al robot esclavo y mantenerse a una distancia definida.

El robot esclavo estará compuesto por un chasis con dos motores de DC independientes y una rueda loca para darle estabilidad, una cámara, la etapa de potencia para los motores y reguladores de voltaje necesarios para el hardware requerido, todo interconectado a una tarjeta de desarrollo Raspberry Pi 3, quien opera bajo un sistema operativo que nos permitirá por medio de un lenguaje de comunicación crear un algoritmo capaz de integrar la visión artificial para generar las señales de control del robot esclavo requiere para seguir al robot líder en su trayectoria.

### 4.1. Navegación autónoma.

Un robot móvil el poder contar con tecnología que le permita desplazarse por diversos entornos es muy necesaria e imprescindible en estos días, los diversos escenarios donde se desarrollan las tareas con o sin supervisión, requieren de un sistema de navegación muy completo.

La navegación puede definirse como la combinación de tres tecnologías fundamentales:

- Autolocalización.
- Mapeo e interpretación del mapa.
- Planificación y ejecución de trayectoria y movimiento.

Las dos primeras competencias son necesarias de la tercera. No ser capaz de localizarse en un entorno conocido o desconocido de una manera fiable inhibe la generación de

planes para una trayectoria y movimientos exitosos. La autonomía de un robot depende de su capacidad para navegar de forma independiente en el entorno y en cooperación con otros elementos implicados en su función como humanos u otros robots.

Para sustituir el elemento humano en la conducción se deben utilizar diferentes tecnologías desarrolladas de forma simultánea, para poder sustituir funciones humanas sensoriales realmente importantes.

Una de las principales funciones es la capacidad para tomar decisiones mientras se conduce, que tienen los seres humanos, la cual se debería sustituir por una máquina capaz de aprender multitud de distintos algoritmos. Este ordenador central será el cerebro del coche autónomo, actuará en tiempo real gracias a un software que procesará la información y tomará las decisiones oportunas. En el proyecto, este ordenador central será el micro-controlador elegido: la Raspberry Pi 3 Model B.

Las cámaras sustituyen al sentido de la vista y es posible reconocer a otros vehículos, peatones, colores, luces, señales de tráfico, etc. Por si solas, no ofrecen grandes funcionalidades pero combinadas con algoritmos y técnicas de visión artificial, se dota al vehículo del sentido de la visión.

## Visión Artificial

Es un campo de la inteligencia artificial enfocado a que las computadoras puedan extraer información a partir de imágenes, ofreciendo soluciones a problemas del mundo real a partir de ellas. Un ejemplo que se puede ver en el día a día, es la detección de caras que realizan al hacerse una foto los Smartphones.

La visión por computador tiene como principal objetivo obtener la información explícita y el significado de la realidad lo mas aproximado posible de como lo haría una persona.

Hay muchas tecnologías que utilizan la visión por computador, entre las cuales están: el reconocimiento y seguimiento de objetos, detección de eventos, reconstrucción de una escena (mapping), restauración de imágenes, por lo que puede aplicarse a numerosos campos como pueden ser el procesamiento de señales, reconocimiento de patrones y la robótica, entre otros.

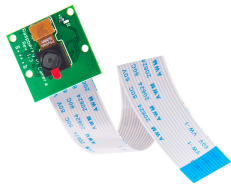
El objetivo de la visión por computadora es, como ya se ha mencionado anteriormente, extraer información a partir de imágenes se debe realizar un proceso que se puede dividir en seis áreas principales:

1. Adquisición de la imagen: proceso que nos lleva a la obtención de una imagen visual.



2. Preprocesamiento: técnicas de reducción de ruido y enriquecimiento de detalles en la imagen.
3. Segmentación: proceso que divide una imagen en objetos de interés.
4. Extracción de características: se diferencian unos objetos de otros a partir de un conjunto de características útiles.
5. Reconocimiento: proceso que identifica esos objetos.
6. Interpretación: asigna un significado a un conjunto de objetos reconocidos.

## Cámara



**Figura 4.1:** RaspiCam

También se ha decidido incorporar a este robot una pequeña cámara, como la que se muestra en la Figura 4.1, que se conecta directamente a la Raspberry. Con ella no solo se pretende conseguir utilidades como el seguimiento de líneas o circuitos, sino que también se podrá utilizar para la visión remota de zonas o lugares que pueden ser peligrosos para las personas. Y no solo eso, ya que gracias a la Visión Artificial se abre un gran abanico de posibilidades, con aplicaciones como detección de códigos,

logotipos, señales, etc.

Características más importantes:

- El sensor de imagen utilizado es un Omnivision 6547 CMOS.
- Resolución de 5 Megapíxeles.
- Resolución de imagen fija de 2592x1944 píxeles.
- Tasa máxima de transferencia de imágenes a 1080 píxeles de 30 frames por segundo.
- Tasa máxima de transferencia de imágenes a 720 píxeles de 60 frames por segundo.
- Funciones de control de imagen
  - Control de exposición automático
  - Balance de contraste automático
  - Filtro de banda automático
  - Detección de luminancia 50/60 Hz automática
  - Calibración de nivel de oscuridad automático
- Rango de temperatura de -20°C a 60°C

- Tamaño de la lente de 1,4 ''
- Dimensiones: 23.86 x 25 x 9 milímetros
- Peso: 3 gramos

## 4.2. Raspberry pi 3 B

Es una placa microcontrolador originalmente desarrollado con el objetivo de estimular la enseñanza informática en los colegios y diseñada para tener un bajo coste. El hardware tiene un tamaño algo más pequeño al de la placa Arduino, siendo de 8,6 centímetros de largo, 5,4 centímetros de ancho y 1,7 centímetros de grosor. Tiene como gran ventaja la variedad de sistemas operativos que puede soportar, al igual que los lenguajes de programación (C, C++, Python).

Esta placa es la elegida para realizar el proyecto por la combinación de la integración para el uso de la cámara y realizar el procesamiento de las imágenes mediante un lenguaje de programación, gracias a que dispone de un procesador ARMv8, un procesador de cuatro núcleos a 1,2 GHz de 64 bits, un 1 GB de RAM, 40 I/O digitales, 2 de las cuales tienen salida PWM, 2 UARTS (puertos de comunicación Serie Tx Rx), 4 conexiones USB, puerto Ethernet, conexión HDMI, conexión audio jack, CSI (Camera Interface) y DSI (Display Interface). Además, lleva incorporado un módulo WIFI 802.11n Wireless LAN y un módulo Bluetooth 4.1 Low Energy y tiene entrada para tarjeta micro SD.



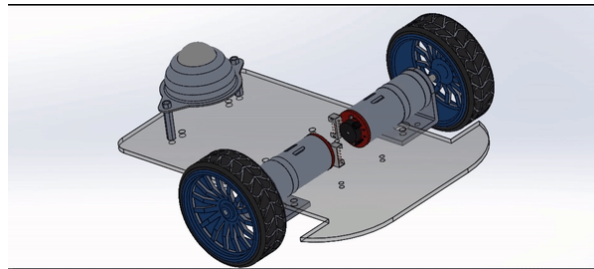
**Figura 4.2:** Tarjeta de desarrollo Raspberry pi 3 B

Para la comunicación usando pines no definidos como UARTS es necesario el uso de la librería WiringPi y tener especial cuidado con la diferencia de voltajes de las conexiones.

La fundación da soporte para las descargas de las distribuciones para arquitectura ARM, Raspbian (derivada de Debian), RISC OS, Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora); y promueve principalmente el aprendizaje del lenguaje de programación Python, y otros lenguajes como Tiny BASIC, C y Perl.

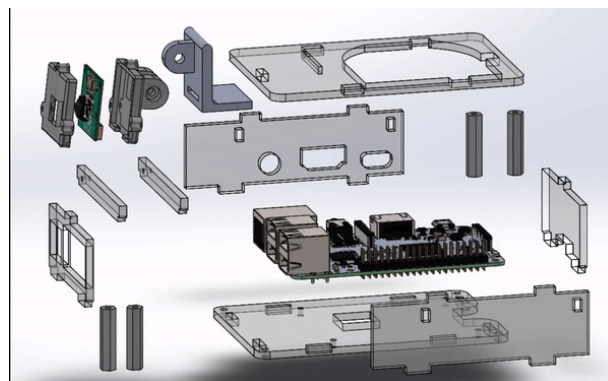
### 4.3. Plataforma del robot

El robot esclavo se establece en una configuración diferencial el cual cuenta con dos motores de DC independientes acoplados a ruedas y una tercera rueda loca para dar estabilidad como se muestra en la figura 4.3, estos están sujetos a una plataforma que sera la estructura donde se colocaran los módulos de potencia y la misma tarjeta de desarrollo Raspberry.



**Figura 4.3:** Plataforma robot móvil esclavo diseñado en 3D

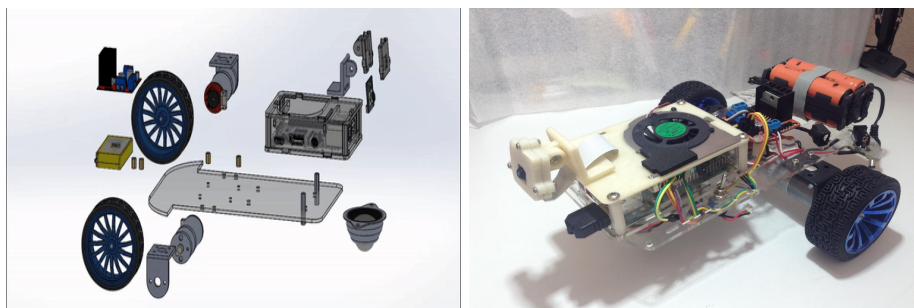
Para esta plataforma se diseñaron piezas como se muestra en la figura para poder fijar tanto la tarjeta Raspberry a la plataforma como una cubierta para la cámara, estas piezas se diseñan en SolidWorks 2018 y generando los archivos STL necesarios para poder ser impresos en 3D, estas piezas mantendrán fija la tarjeta de desarrollo Raspberry pi 3B añadiendo un espacio para colocar un ventilador fan para mantener la temperatura de la misma en optimas condiciones, debido a las diversas operaciones que realizara debe mantenerse por debajo de los 60 grados, para mantener la velocidad del CPU a 1200MHz, las piezas diseñadas para la cámara son para mantenerla orientada hacia el frente del robot esclavo y poder localizar al robot líder



**Figura 4.4:** Case para Raspberry y cámara

Una vez impresas las piezas que fueron diseñadas para dar soporte a la tarjeta

Raspberry pi y a la cámara son colocadas en la plataforma mostrada en la figura 4.3, además de colocar un puente H basado en un L298N como etapa de potencia para los motores de DC que son de las mismas características que se emplearon para el robot líder.

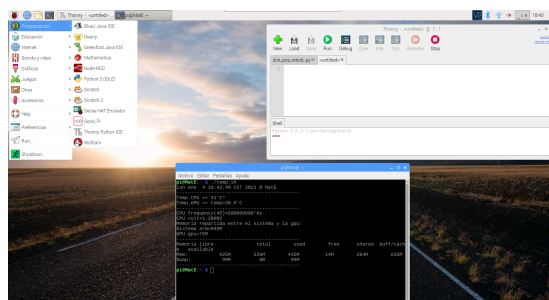


**Figura 4.5:** Robot móvil esclavo

De acuerdo a los requerimiento de energía que requiere la tarjeta que son 5v y 2A mínimo, el modulo L298N solo suministra un máximo de 1A, es por eso que se incluye un modulo LM2596 regulado a 5v con la corriente necesaria para que la tarjeta Raspberry opere de forma adecuada. Finalmente se integran todas las partes del robot esclavo como se muestra en la figura 4.5.

## 4.4. Software

El sistema operativo Raspbian es una distribución de GNU/Linux y por lo tanto está basado en Debian Wheezy, optimizado para la placa Raspberry Pi. Este sistema operativo tiene los programas y utilidades básicas para poder arrancar la Raspberry Pi. Además, Raspbian proporciona más de 35.000 paquetes y software pre compilados de fácil instalación para una mejora del rendimiento de la placa.



**Figura 4.6:** Ambiente de trabajo Raspbian

Tras instalar el sistema operativo se deben de realizar diversas configuraciones, para habilitar el uso de la cámara, resolución de la imagen mostrada como escritorio remoto, la conexión WiFi, habilitar los puertos SSH, y VNC (por sus siglas en ingles, Virtual Network Computing) para establecer una conexión remota con la placa desde el ordenador sin necesidad de un monitor, para poder descargar sus paquetes y actualizaciones correspondientes. Una vez activada se instalan los programas que donde se diseñara el algoritmo en lenguaje Python empleando librerías de OpenCV que nos ayudara a procesar las imágenes captadas por la cámara. Por último, se deben instalar las librerías GPIO para poder comunicar y controlar los motores de DC.

## Python y OpenCV

El lenguaje de programación Python, es el que se ha utilizado para realizar el software de este proyecto. Es un lenguaje poderoso, fácil de aprender, capaz de soportar multitud de formas de programar y compatible con la librería OpenCV

Es una biblioteca libre de visión artificial, fácil de utilizar y altamente eficiente. Tiene licencia BSD, que permite ser usada libremente para propósitos comerciales y de investigación, con las condiciones en ella expresada

Contiene más de 500 funciones, algunas de las cuales se han utilizado en el proyecto. Esta librería nos permitirá leer una imagen, pasarla a distintos escalas de colores (RGB, HSV, blanco y negro), eliminar contornos no deseados o encontrar líneas rectas, y analizar sus ángulos y sus puntos de fuga.

## Localización y seguimiento

El tema de reconstrucción de imágenes en 3 dimensiones a partir de diferentes tomas, es hoy en día una tarea muy compleja, sin importar que tan sofisticadas sean los ordenadores, poder realizar una identificación adecuada es una tarea a resolver por la gran diversidad de formas, colores, posiciones, tamaños etc... Es por eso que se ha invertido un gran cantidad de técnicas y algoritmos para el reconocimiento e identificación de objetos.

Para nuestra tarea a resolver es una linea de algoritmos desarrollados para la detección de un color determinado, simplificando la programación y el tiempo de procesamiento. Existen técnicas muy diversas para la detección de objetos, para ser breves se clasifican en 4 metodologías:

- Detectores de regiones: Para la detección de regiones se utilizan técnicas basadas en el hecho de la similitud: binarización, apoyada en el uso de umbrales; crecimiento de regiones, mediante la adición de píxeles; división de regiones y similitud de textura, color o niveles de gris. En general, el proceso de la segmentación suele

resultar complejo, debido, por un lado, a que no se tiene una información adecuada de los objetos a extraer y, por otro, a que en la escena a segmentar aparece normalmente ruido. Es por esto que el uso de conocimiento sobre el tipo de imagen a segmentar o alguna otra información de alto nivel puede resultar muy útil para conseguir la segmentación de la imagen. Cuando la calidad de la imagen no es buena y no es posible extraer la información de forma adecuada, es necesario aplicar técnicas de mejora en la calidad de la imagen original.

- **Sustracción del fondo (background subtraction):** La sustracción de fondo es una técnica utilizada para la detección de los objetos en movimiento, cuya aplicación consiste en comparar cada uno de los frames (cada una de las imágenes que componen un vídeo) de una secuencia de imágenes con el frame inicial u otro que se escoja a conveniencia, de tal forma que en el vídeo resultante, los elementos que permanezcan constantes se vean de color negro, y los que han cambiado, de blanco o viceversa. Este proceso corresponde a la etapa de segmentación y está basado en técnicas de detección de regiones que hacen uso de la binarización basada en umbrales y funciones estadísticas de densidad para establecer el modelo de fondo. El resultado (manchas) de este proceso determina cuáles son los objetos de interés en la imagen.
- **Machine Learning:** También denominado aprendizaje automático. Es el campo que da a las computadoras la habilidad para aprender sin haber sido explícitamente programada. Esto es, mostrando una serie de ejemplos al algoritmo éste es capaz de identificar objetos similares a los ejemplos, pero existe un problema, este método necesita que sean aportados las muestras a detectar, además de la potencia de procesamiento requerida para el análisis es muy grande.
- **Segmentación:** Es el proceso de particionar una imagen digital en múltiples regiones, conjuntos de píxeles también llamados superpíxeles, y tiene como objetivo simplificar la imagen para hacer mucho más fácil su análisis. Dando como resultado de la aplicación función segmentación a una imagen regiones con características similares. Las cuales son adyacentes y significativamente diferentes en cuanto a sus propiedades características, como puede ser el color, intensidad, textura, etc. Los diversos algoritmos de segmentación proporcionan cada uno de ellos información diferente, esto es debido a que no existe una solución general para segmentar una imagen, y en función de los resultados se emplean diferentes técnicas o agrupar varias de ellas, de acuerdo a la tarea a resolver, algunos de estas son: Métodos de agrupamiento (Clustering), Métodos basados en el histograma, detección de bordes, método del valor umbral, segmentación basada en modelos, etc.

Se empleara el método del valor umbral para resolver la tarea de identificar un color en específico, ya que el costo computacional es menor comparado con los demás, y permite ser implementado en entornos dinámicos, para comenzar iniciamos importando las librerías **cv2** y **numpy** y podemos iniciar el proceso de captura del vídeo en streaming. como se muestra en la figura 4.7.



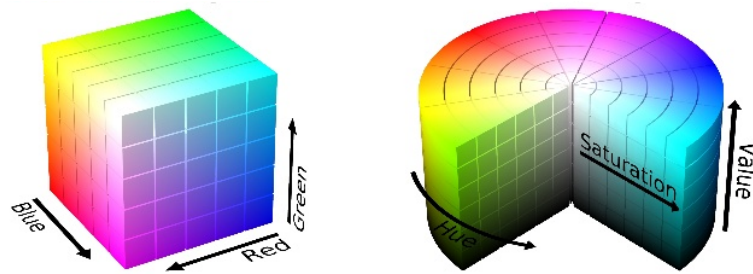
**Figura 4.7:** Obtención de fotogramas para analizar.

Es posible guardar el vídeo para su análisis post-grabación, para la tarea que requiere desarrollar nuestro robot esclavo es necesario procesar estos fotogramas en tiempo real, aunque requiere mas procesamiento, la tarjeta Raspberry pi puede realizar esta tarea.

### Conversión de RGB a HSV

Para comenzar se realiza una conversión de la imagen del espacio de color BGR (Blue, Green, Red) a HSV (Hue, Saturation y Value; traducido al español serían: Matíz, Saturación y Valor). En el espacio HSV, Hue indica el color, Saturation indica la oscuridad, la escala de gris, el valor más alto de saturación dará el color que se haya indicado en el matiz y un valor pequeño de saturación siempre será gris. Por último, el valor indica el brillo, siendo el extremo inferior negro y el extremo superior blanco.

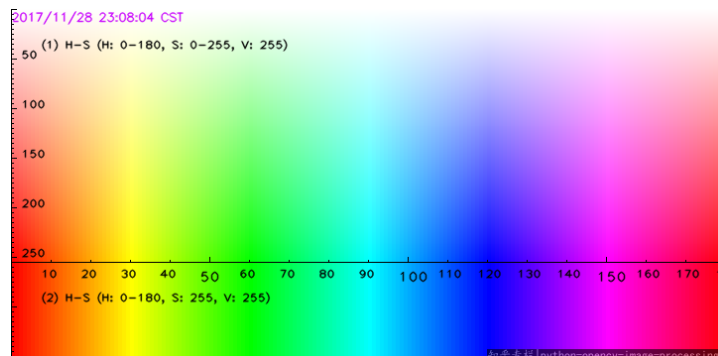
En la siguiente imagen se puede apreciar como varían los colores dependiendo de las distintas variables:



**Figura 4.8:** Representación cubica RGB y cilíndrica HSV.

Se realiza esta conversión porque es más fácil buscar un color en el espacio HSV que en el BGR, ya que en BGR no se tiene en cuenta la luz, el brillo y las sombras, y mientras una sombra en RGB cambia el color, en HSV solo habría que ajustar el brillo.





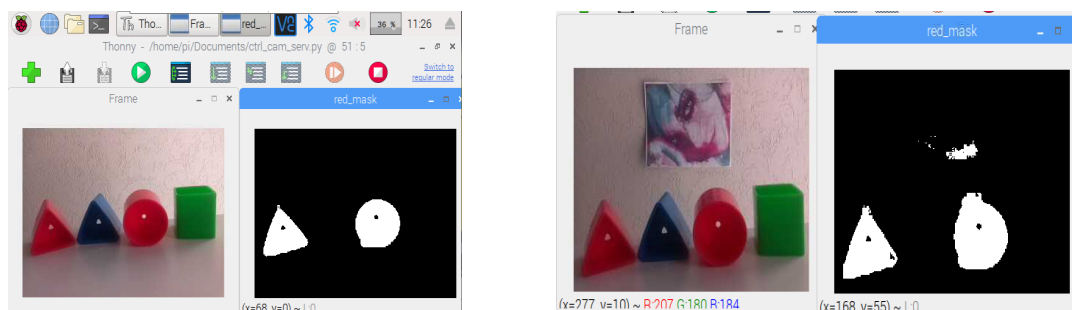
**Figura 4.9:** Vista de los componentes HSV.

Ahora de estos componentes vamos a escoger un rango para el color rojo como vemos en la figura 4.9 la componente H va de 0 a 179, y en este se encuentran todos los colores, el color rojo se encuentra en dos regiones, vamos a elegir solo una de ellas la que se encuentra en 161 a 179 en H, para delimitar con un color rojo mínimo y un máximo, este rango se encuentra en el código del apartado B.10 en las líneas 41, 42.

Como se muestra en la figura 4.10(b) se colocó una imagen impresa en la parte posterior de la escena para introducir ruido en la gama detectable, donde podemos confirmar que se identifican algunas áreas de ella además de las figuras del color deseado.

### Operaciones morfológicas

Para este caso vamos a realizar la segmentación del frame o imagen capturada mediante la conversión a binaria (blanco y negro), en función de ciertos parámetros asignados, de tal forma que si el valor del píxel a analizar es mayor a un umbral definido será asignado un color blanco de lo contrario será negro, facilitando la separación del objeto con el resto del fondo, con las siguientes líneas de código creando un rango para el color rojo.



(a) *Rango de color rojo detectable.*

(b) *Aplicando conversión a blanco y negro.*

**Figura 4.10:** Prueba de visión robot esclavo.



Con este algoritmo sencillo es posible localizar a un objeto de color rojo, pero además de eso requerimos que reconozca el centro de la figura y que mientras se desplaza el objeto de color se vaya actualizando la información, por lo tanto es necesario aplicar una operación intermedia.

## Identificación de contornos

Esta aplicación reducirá la cantidad de información que hay que considerar para llevar a cabo el seguimiento del objeto en tiempo real lo más sencillo posible para que el cálculo sea más rápido.

OpenCV incorpora una funcionalidad que nos permitirá realizar dicha tarea mediante la reducción de la imagen original a otra nueva formada solo por contornos de la misma, al ser una imagen binaria proporciona resultados considerables que reducen de forma cualitativa la cantidad de datos a procesar en el seguimiento del objeto. Esta función que se emplea es *findContours*, y esta basada en un algoritmo desarrollado por Satoshi Suzuki, obteniendo los bordes que se encuentran en una imagen binaria y cuenta con los siguientes parámetros:

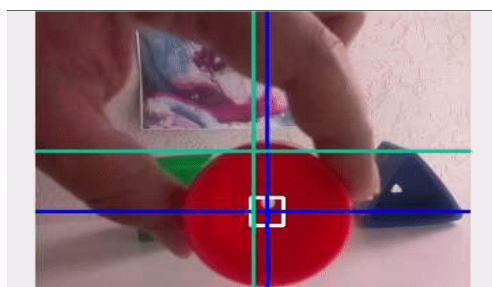
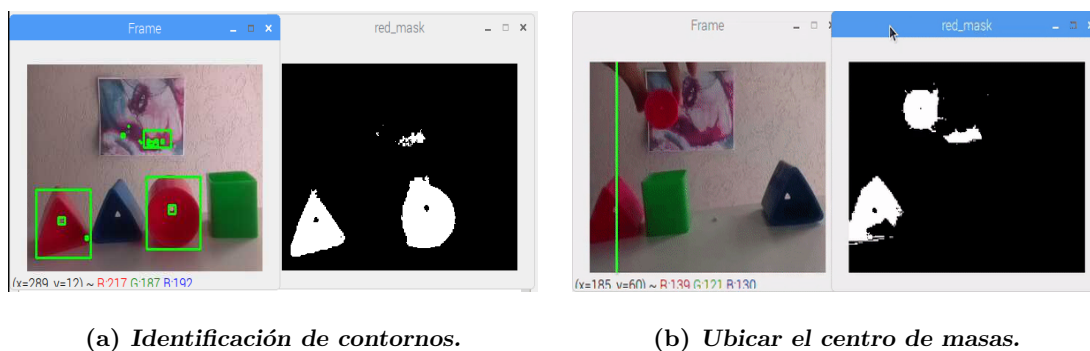
- **Image:** Imagen de entrada (binaria) a aplicar la detección de contornos.
- **Contours:** Es un vector de vectores, un vector de contornos en el que cada contorno es un vector de puntos.
- **Hierarchy:** Contiene información sobre su topología de la imagen, es decir, como se relaciona entre sí los contornos detectados. Contiene tantos elementos como el número de contornos detectados.
- **Mode:** Permite definir un modo de recuperación para los contornos, existen modos que nos dan la opción de eliminar las relaciones de jerarquía, o establecer nuevas relaciones que identifiquen solo los contornos externos, etc. En el caso para el seguimiento de un objeto aplicaremos la operación *RETR\_TREE* que recupera todos los contornos con sus jerarquías.
- **Method:** El método de aproximación de contornos. Dispone de varios modos de trabajo, como por ejemplo: sin aproximación y almacena todos los puntos del contorno, realiza una aproximación simple al hacer una compresión horizontal y verticalmente para reducir los puntos del contorno. Para nuestro caso aplicaremos el modo *CHAIN\_APPROX\_SIMPLE* el cual ofrece resultados aceptables con un mínimo de datos del contorno, siendo posible reducir en gran medida los cálculos que el procesador debe de realizar.

Ahora es necesario poder definir el seguimiento de objetos como un problema de estimación de la trayectoria, en la imagen a analizar, para ello es necesario un algoritmo

cuyo fin sea seguir el objeto y mostrar cual es la situación del objeto que esta siguiendo en cada frame del vídeo.

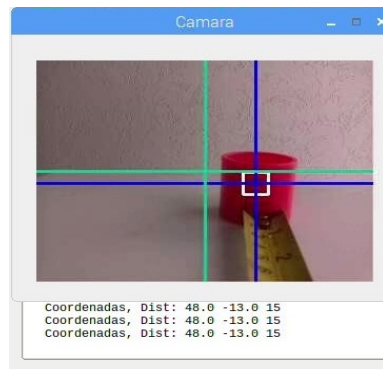
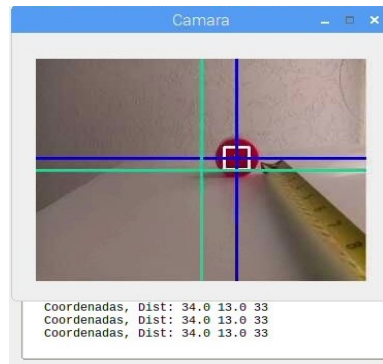
Para esto se necesita utilizar el centro de masas de los objetos como un punto principal a seguir, y definir la manera de calcular de forma sencilla el tamaño del objeto dentro del campo de visión principalmente esto nos ayudara a saber que tan alejado se encuentra dicho objeto y así poder tomar acciones de control necesarias para que el robot móvil una vez que localiza el objeto que para este caso es nuestro robot líder, se mantenga a una distancia de el. Por lo tanto el objeto identificado tendrá su centro de masas y su tamaño en forma cuadrada, ancho y alto.

Para obtener el centro de masas se emplea la función **moments** de OpenCV, dicha operación devuelve los momentos de área hasta tercer orden de un polígono en una estructura de tipo “Moments” el momento espacial es el área encerrada por el contorno debido a que es la suma de todos los contornos que componen el objeto, definiendo así el centro de masas por sus coordenadas x,y como se muestra en el siguiente código y el resultado de las pruebas se muestran en la figura 4.11



**Figura 4.11:** Pruebas para visión de robot esclavo.

Para calcular el tamaño del objeto con relación a la escena se implementa un modulo que dibuja un rectángulo circunscrito a la figura identificada para proporcionar una mejor y mas clara visualización.

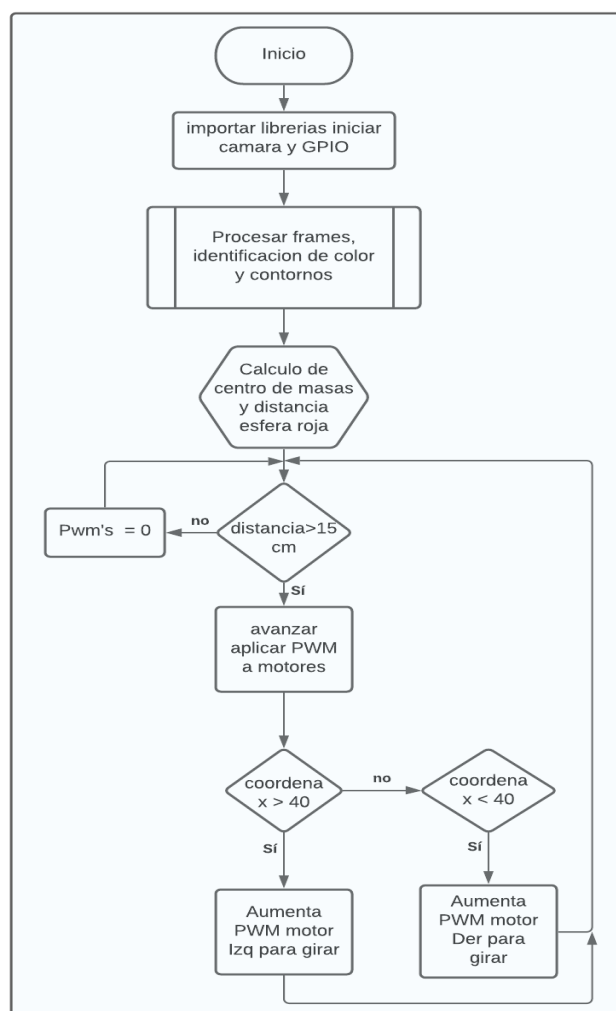
(a) *Ubicación por coordenadas.*(b) *Distancia medida.*(c) *calculo de distancia.*(d) *objeto a 34 cm.***Figura 4.12:** Pruebas para identificar en movimiento y distancias.

Esto es de gran utilidad para restringir el tamaño de los objetos que se desean identificar y descartar falsos positivos, además que con la información del área es posible generar las acciones de control es decir, al obtener un área mayor el objeto se encuentra a una distancia cercana, conforme se aleja el robot líder el área disminuye y así deberá de avanzar el robot esclavo para mantenerse a una distancia definida siempre, sin importar la trayectoria que el robot líder realice.

### Control de dirección.

Una vez realizados todos los procesos anteriores, se obtiene toda la información necesaria para poder decidir sobre la dirección a tomar por el robot, como bien se ha explicado anteriormente. Para esto se emplea la librería GPIO.BCM así como “time” en python. Una vez importadas se debe indicar qué salidas de GPIO están conectadas a nuestro controlador de los motores. Tras haberlo indicado, ya sólo se deben gestionar dos señales: de acuerdo a la caracterización de los motores, igual forma que los

empleados en el robot líder, se obtienen las frecuencias de operación para cada motor, 195 Hz para el motor derecho y 245 Hz para el motor izquierdo, el algoritmo se diseño de acuerdo al diagrama de flujo que se muestra en la figura 4.13 y el código se puede consultar en B.10.



**Figura 4.13:** Diagrama de operación para robot esclavo.

Para indicar el valor del PWM a aplicar en los motores se utiliza la función “ChangeDutyCycle” y se especifica un valor entre 0-100. De esta manera, cada vez que el código encargado de calcular el área del objeto se podrá aumentar o disminuir el valor para cada motor ademas de saber hacia que dirección se tiene que compensar para mantenerse centrado el objeto.

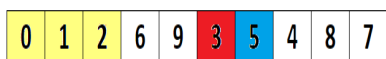
## Resultados y pruebas

---

### 5.1. Implementación control óptimo en robot líder.

Es momento de procesar la información que manda la tarjeta de desarrollo FPGA, los valores de cada uno de los encoder's, para esto emplearemos una comunicación por medio de Bluetooth desde el software de MATLAB, para recibir los 8 paquetes de datos, y codificar de acuerdo al identificador que cuenta cada paquete donde emplearemos un algoritmo capaz de identificar el valor correspondiente a la posición y el dato, colocando esta ultima de una forma ordenada para conocer el valor que fue mandado originalmente.

Existen diversos algoritmos de ordenamiento, para nuestro caso emplearemos en denominado: Ordenación de burbuja (Bubble Sort en inglés) es un sencillo algoritmo de ordenamiento como se muestra en la figura 5.1. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. También es conocido como el método del intercambio directo.



**Figura 5.1:** Algoritmo de ordenamiento

Con este algoritmo estamos garantizando un ordenamiento, primero debemos leer los 8 paquetes y que se concatenen para así poder aplicar el algoritmo *burbuja*, el cual leerá de 8 en 8 posiciones enfocándose solo en ordenar los 3 bits mas significativos que son los bits identificadores, logrando así agrupar los 5 bits que son de nuestro interés, para así convertir el dato binario a decimal correspondiente al valor de cada encoder, como se muestra en el código B.9.

Hasta este punto con la decodificación de los encoders, conocemos el numero de pulsos que hay en cada intervalo de tiempo (muestreo), así como la dirección que toma cada una de los motores, podemos efectuar la conversión para saber cual ha sido la

distancia que recorrió cada una de las ruedas, esto lo logramos con la siguiente ecuación:

$$L = Np * \text{resolución} \quad (5.1)$$

donde  $Np$  =numero de pulsos generados por el circuito implementado, representado por la figura 3.13 multiplicado por la resolución, este valor lo obtenemos de saber la longitud recorrida en una vuelta de las ruedas, donde el radio es  $3.4\text{cm}$  entonces, en una vuelta la longitud lineal recorrida es de:

$$L = 2\pi R = 2 * \pi * 3.4\text{cm} = 21.3628 \text{ cm} \quad (5.2)$$

conociendo la distancia que recorre una rueda, lo dividimos entre el numero de pulsos generados en una vuelta, que son 5200 la rueda izquierda y 5400 en la rueda derecha, entonces obtenemos la siguiente resolución:

$$\begin{aligned} \text{Resolución motor izquierdo} &= \frac{21.3628}{5200} = 0.0041082 \text{ cm} * \text{pulso} \\ \text{Resolución motor derecho} &= \frac{21.3628}{5400} = 0.0039560 \text{ cm} * \text{pulso} \end{aligned} \quad (5.3)$$

Sustituimos en la ecuación (5.1) para conocer la distancia recorrida en cada muestreo de los encoders. La velocidad así como la aceleración pueden ser estimadas, esto debido a la falta de un sensor que nos pueda entregar este valor, es necesario que a partir de la posición recorrida en cada intervalo, estimar su velocidad, existen diversas formas de implementar un algoritmo que nos genere este parámetro.

Una forma ampliamente usada es el método de Euler (diferenciación numérica) la cual esta dada por la siguiente expresión:

$$\dot{f}(tk) \approx \frac{f(tk) - f(tk-1)}{tk - tk_1} = \frac{f(tk) - f(tk-1)}{h} \quad (5.4)$$

Particularmente esta aproximación es conocida como diferenciación con un paso atrás (*backward difference*), de acuerdo con [29], tomando este método es posible implementarlo para aproximar la derivada de la posición y obtener la velocidad, pero antes es necesario determinar  $h$ , que es el intervalo de tiempo que le daremos al sistema para saber cuantos pulsos se han generado por los encoders, este valor es de 0.1 s.

Obteniendo las velocidades correspondientes a cada una de las ruedas se puede obtener la velocidad lineal y angular del robot móvil, así como la orientación del mismo con las siguientes expresiones:

$$V_{RM} = \frac{V_{Mr} + V_{Ml}}{2} \quad (5.5)$$

$$\Omega_{RM} = \frac{V_{Mr} - V_{Ml}}{2a} \quad (5.6)$$

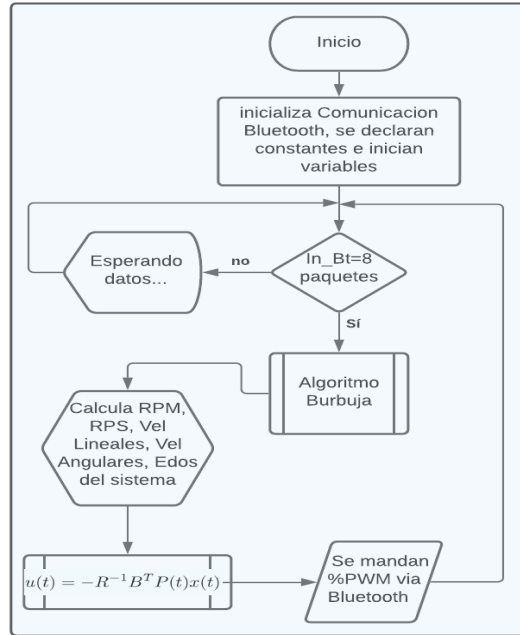
$$\alpha_{RM} = \int_0^t \omega = \nabla t \sum_{i=0}^N \quad (5.7)$$

Generando así el cálculo de las variables de estado: velocidad lineal, la velocidad angular y la orientación del robot móvil, respectivamente (a es la distancia media entre las dos ruedas).

Para la estimación de los estados  $x_s$  y  $y_s$  se realizara mediante un proceso de integración numérica denominado trapezoidal debido a que las expresiones (5.8) nos dan información de sus velocidades, y efectuando la integración de estas obtendremos la posición.

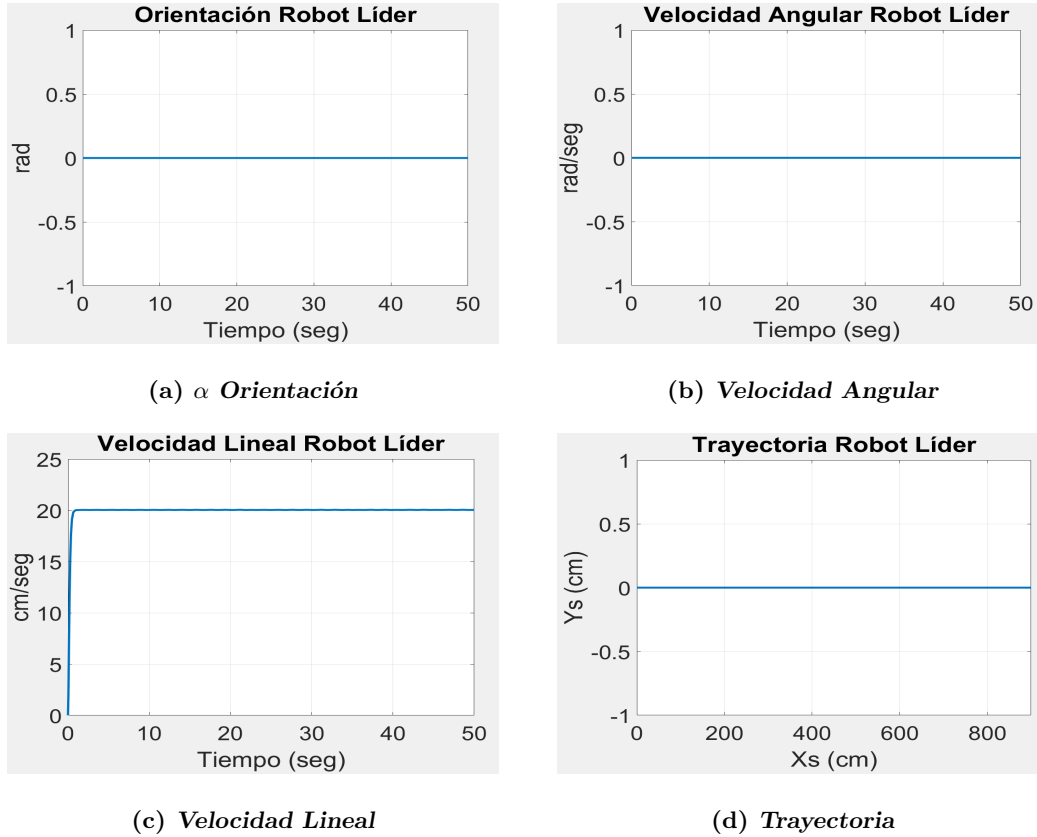
$$\begin{aligned} xsp(1, n) &= vMovil(1, n) * \cos(phi(1, n)) - b * wMovil(1, n) * \sin(phi(1, n)) \\ ysp(1, n) &= vMovil(1, n) * \sin(phi(1, n)) + b * wMovil(1, n) * \cos(phi(1, n)); \end{aligned} \quad (5.8)$$

donde: b= distancia que hay del centro del eje de las ruedas al centro del móvil (s). Con esto ya tenemos los estados de nuestro sistema, con esta información se calcula  $\tilde{x}_s$  que corresponde a  $\tilde{x}_s = x_{deseada} - x_{actual}$ . Como encontramos la solución de estabilización  $\underline{P}$  de la ecuación algebraica de Riccati, retomaremos esta matriz para poder operar adecuadamente de acuerdo a las ecuaciones (2.67) y (2.68) para obtener la ley de control, estos algoritmos se han implementado en MATLAB y se muestran en el apéndice B.8 de acuerdo con el siguiente diagrama en la figura 5.2.



**Figura 5.2:** Algoritmo implementado en Matlab para Robot Líder.

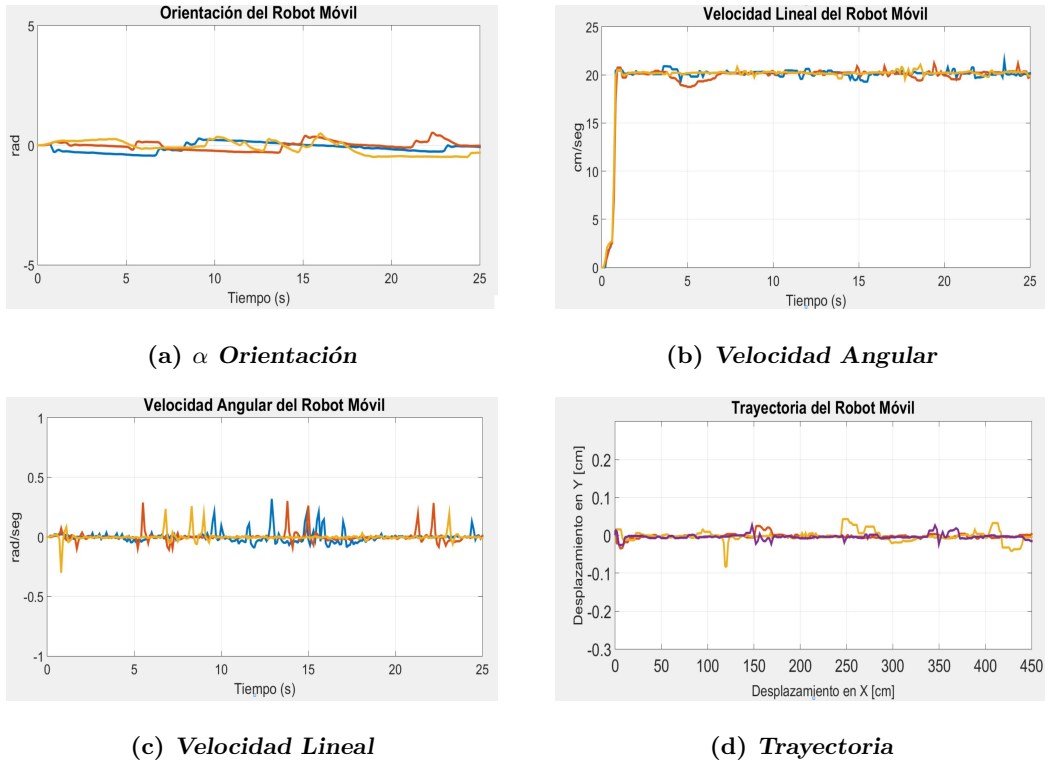
Considerando la ley de control obtenida para que recorra una línea recta se realiza la simulación para esta primer trayectoria deseada a una velocidad de  $v_0 = 20\text{cm/seg}$  se muestra el comportamiento de las variables en la siguiente figura 5.3:



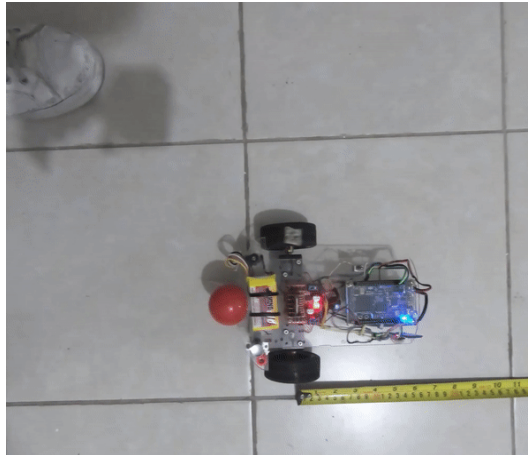
**Figura 5.3:** Simulación Robot Líder para línea recta sentido positivo.

Recordemos que los valores calculados en cada iteración corresponden a  $u_r = M_r + M_l$  y  $u_l = M_r - M_l$  pero son magnitudes correspondientes a torque, debido a esto se da solución al sistema de 2 ecuaciones con 2 incógnitas, y obtenemos un valor dentro del rango de voltaje para los motores, y así calcular el ancho del pulso de acuerdo a la ley de control obtenida.





**Figura 5.4:** Resultado experimental del robot líder.

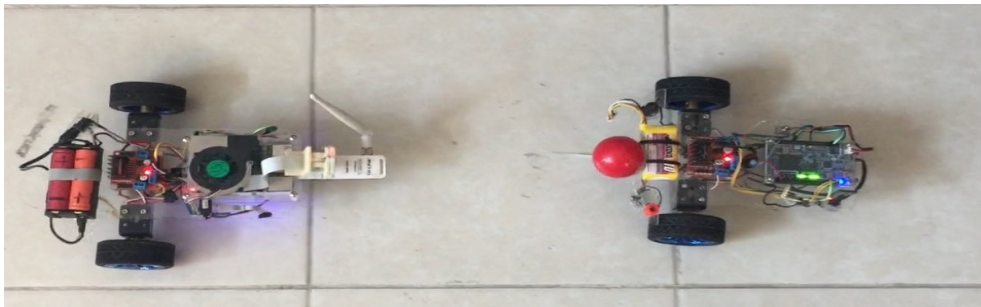


**Figura 5.5:** Pruebas de trayectoria robot líder.

Se muestra en la figura 5.4 los resultados experimentales para la línea recta que realizó el robot líder, con una velocidad deseada de 20 cm/seg se observa en estas gráficas los resultados de las acciones que genera el control óptimo implementado.

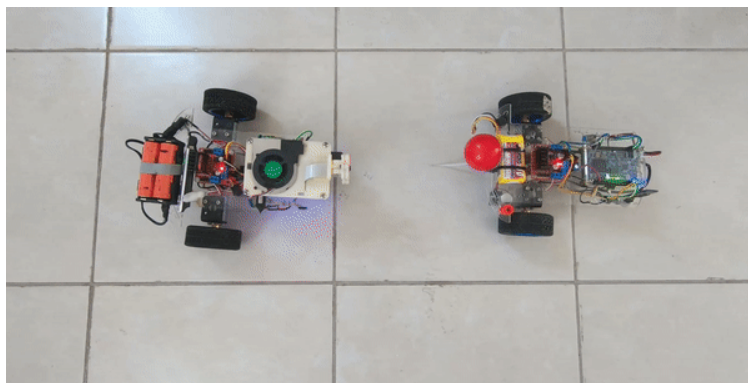
## 5.2. Resultado experimental de forma colaborativa

Aplicado el sistema de control óptimo en el robot líder para que recorra una línea recta, será colocado el robot esclavo justo detrás y mediante el acceso remoto que se implemento con VNC, vamos a inicializar el algoritmo de control por medio de visión artificial generado en la tarjeta de desarrollo Raspberry Pi 3B, también nos servirá esta conexión remota para monitorear que está captando la cámara, y como se va ejecutando en tiempo real la localización del objeto colocado en la parte posterior de nuestro robot líder, la esfera de color rojo será el punto a mantener centrado el robot esclavo y mientras recorre la trayectoria deberá de mantenerse a una distancia predefinida, durante todo el recorrido.



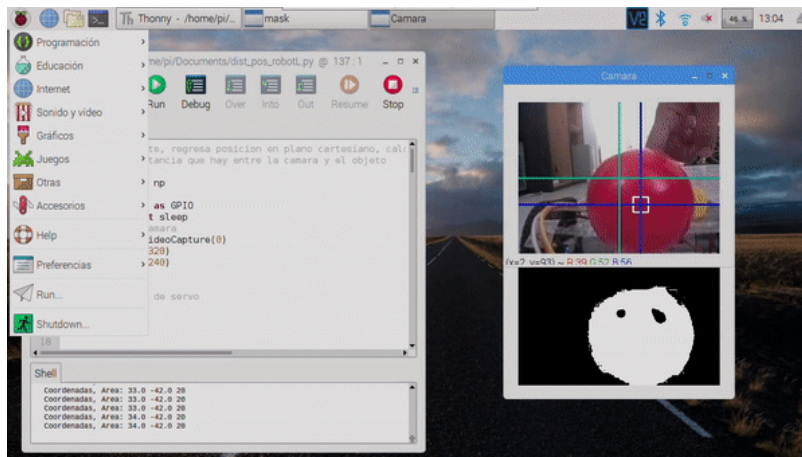
**Figura 5.6:** Robot líder y esclavo.

El sistema de visión implementado en el algoritmo entrega la información siguiente: abre dos ventanas emergentes, una de ellas tiene lo que llamaremos máscara, se trata de la imagen o frame que captura la cámara aplicando un procesamiento a blanco y negro identificando un color en específico, una gamma de dicho color será detectado omitiendo cualquier otro, como se muestra en la figura 5.8 localizada en la parte inferior derecha.



**Figura 5.7:** Trayectoria colaborativa

La segunda ventana que genera el algoritmo de control mediante la visión artificial, se muestra la imagen dividida en 4 cuadrantes, sobre este mismo frame se localiza el centro de masas del área identificada de manera aproximada, detectando este punto y dibujando un rectángulo circunscrito en cada instante, se conoce la ubicación en alguna de las divisiones del área de visión disponible así como la distancia aproximada a la que se encuentra, con las coordenadas disponibles se generan las acciones de control para ir hacia la derecha o izquierda, y mantener el centro de masas del objeto de color rojo al origen del eje x, y de esta manera recorrer la misma trayectoria que el robot líder.



**Figura 5.8:** Acceso remoto desde VNC, visión de robot esclavo

De esta forma se logra que una trayectoria definida sea recorrida por dos robots de manera colaborativa, un robot líder donde se implementa un control óptimo para seguir una línea recta únicamente con la información que generan los encoder's en cada uno de los motores independientes en la plataforma, y un segundo robot el cual desconoce la trayectoria y su única tarea es identificar y seguir un objeto de color en específico, gracias a una cámara que lo provee de una visión artificial, como se muestra en la figura 5.7 ambos robots recorren una trayectoria de manera colaborativa.



# Conclusiones

---

La implementación para que un robot móvil del tipo diferencial, siga una trayectoria definida es posible hasta cierto punto con ayuda de los sensores adecuados, y que esta trayectoria recorrida sea la adecuada para una tarea determinada, es indispensable que el sistema tenga una ley de control, asegurando así que en todo momento sea la trayectoria correcta, la teoría de control óptimo no solo lo hace posible, además nos asegura su estabilización en el menor tiempo posible, ayudando a aprovechar el consumo de energía requerido para cumplir su tarea.

Es necesario conocer el modelo dinámico para establecer y comprender los alcances que tendrá el robot móvil una vez implementado físicamente, la obtención de estas ecuaciones dinámicas, se realizaron mediante conceptos de mecánica clásica y aplicando el principio de D'Alembert; se decidió emplear este método debido a la sencillez para relacionar las fuerzas implicadas, en la dinámica del sistema con el movimiento que las ruedas ejercen en él. Y con el fin de mejorar la estabilización del sistema, se estudió y logró la inclusión del modelo electromecánico de los motores de DC, mismo estudio no se encuentra en la bibliografía; debido a que es uno de los aportes que se lograron al desarrollar este trabajo de tesis, mejorando los resultados obtenidos en la implementación.

Obteniendo un modelo no lineal de 5 ecuaciones con 2 controles, que al someter a una simulación se desea que el robot móvil recorra una trayectoria en línea recta, se puede confirmar una mejora en su estabilidad, en comparación con el modelo antes de incluir las ecuaciones de los motores, es por esta razón que se continuo trabajando con el modelo final aplicando el Teorema de Tikhonov para obtener la solución para una trayectoria en línea recta y semi-circular.

Se presenta hasta cierto punto una forma muy natural, para caracterizar las posibles trayectorias que el robot móvil puede lograr, es decir: que con líneas rectas y arcos de circunferencia es posible construir una trayectoria con la unión de las mismas, gracias a la programación dinámica obtenemos una ley de control óptimo, que llevara al robot móvil a describir una trayectoria en línea recta para cualquiera de sus variantes. Con la

utilización de las desigualdades matriciales lineales (LMIs), como solución del control óptimo para el sistema, considerando que deberá recorrer un arco de circunferencia, nos lleva a plantear a dicho sistema como variante en el tiempo. Considerando estas metodologías se obtienen las leyes de control óptimo para que el robot móvil describa una trayectoria deseada.

Aunque es un resultado importante, aun quedan grandes problemas por resolver; ya que se evita sintonizar la matriz de peso  $Q$ , y las leyes de control diseñadas solo se comprueba la obtenida para una la linea recta.

La implementación de la electrónica para el robot líder, que solo cuenta con los encoder's como sensores, es suficiente para generar un algoritmo en la tarjeta FPGA que analice dicha información y así obtener las variables necesarias para aplicar la ley de control óptimo, para una trayectoria en linea recta. Además que gracias a la versatilidad que tiene la tarjeta de desarrollo, nos permitió modificar y adaptar el módulo de comunicaciones en varios puntos de las pruebas que se realizaron, así como la resolución de los PWM para operar en los voltajes y frecuencias mas óptimos, que nos llevaron a los resultados presentados en este trabajo, corroborando los obtenidos analíticamente en las simulaciones

La implementación de un sistema de visión artificial para un segundo robot móvil (esclavo), nos sumergió en un mundo muy interesante, que gracias al uso de tarjetas como la empleada para este robot esclavo, una unidad de procesamiento Raspberry pi, nos dio la viabilidad de construir un sistema capaz de analizar los fotogramas e identificar a un objeto de color definido, gracias al lenguaje de programación Python y a las librerías que fueron implementadas de OpenCV, el robot esclavo genera un plano en el cual sabe a cada instante de tiempo donde se ubica el robot líder, y así mantenerse a una distancia definida durante todo el recorrido.

---

## Trabajo a Futuro

Con la implementación del robot líder en una tarjeta de desarrollo FPGA las posibilidades son innumerables, y hay un abanico de opciones, algunas de ellas es la implementación de un sistema de validación del movimiento, mediante el uso del acelerómetro que maneja la FPGA de manera interna, podríamos monitorear cuando inicia y cada intersección de las trayectorias a recorrer, así como el uso del ADC para monitorear el nivel de la batería.

La implementación del control óptimo para una trayectoria semicircular, además de analizar una estrategia para sintonizar las matrices de peso que son empleadas en la síntesis para las leyes de control, considerando además el cambio de la tecnología inalámbrica Bluetooth por un módulo WiFi.

Con el robot esclavo y el uso de la tarjeta Raspberry pi, podríamos extender la idea a una plataforma IoT, así como la detección de varios objetos al mismo tiempo, de diferentes formas y colores, considerando una navegación basada en inteligencia artificial.







## Apéndice A

# Constancias Congreso

### A.1. The 32nd International Conference on Microelectronics



# Trajectory planning in cooperative robots using artificial vision.

Abel A. Rubín-Alvarado\*, José E. M. Gutiérrez-Arias<sup>†</sup>, Fernando Reyes-Cortés<sup>‡</sup>, and Gregoria Corona-Morales<sup>§</sup>

Facultad de Ciencias de la Electrónica, Benemérita Universidad Autónoma de Puebla. CU, Boulevard 18 Sur,

Av. San Claudio y Jardines de San Manuel, 72570 Puebla, Pue., México.

\*abel.rubin@alumno.buap.mx, <sup>†</sup>arigutmses5@gmail.com, <sup>‡</sup>fernando.reyes@correo.buap.mx, <sup>§</sup>gcoronamorales@gmail.com

**Abstract**—When obtaining a mathematical model of any physical system, it is very desirable to be able to verify it before carrying out its implementation, for it is resorted to several methods, one of them is the simulation. In this work we present a methodology denominated SimMechanics, for the system of a mobile robot differential type, carrying out a simulation more approximated to the reality, validating this way the model. The planning of the trajectory it is implemented only in the robot leader with a controller PID applied in the speed of each one of the motors, a second robot endowed with artificial vision denominated slave, that it will maintain identified to the robot leader in all moment and maintaining a determined distance robot leader, to achieve that the trajectory it is traveled by both robots, leader-slave of form cooperative.

**Index Terms**—Mathematical Model, Simulation, Validation of the mathematical model, PID control, Artificial Vision, Collaborative work.

## I. INTRODUCTION

Robotics every day solves new and fascinating tasks for diverse systems, with the only aim of preventing people from continuing to carry out activities; high-risk, difficult to access as in disaster areas where recognition, search and rescue are crucial.

At the end of last century, the industry employs robots that can be operated and / or supervised by a person thus achieving improvements in production and cost reduction, fulfilling these tasks researchers are focused on new and more complicated activities to solve, to the point of no longer being necessary human intervention for its operation, on the contrary we have been conceptualizing the idea of having service robots, thus achieving that many of the tasks that are complex for a single robot are achieved with the operation of several of them.

The mobile robots differential type have been studied for several years, due to its configuration of two independent wheels active and a third passive, being very recurrent in industrial facilities, for the transport of goods, exploration, at home and places like hospitals [1]. Its research considering the kinematic model in conjunction with another robot to achieve a collaborative task, many of the works before its implementation performs a computer simulation with the help of specialized software, one of the objectives of this work is to present a method called SimMechanics which provides

a 3D simulation environment where they are included all the masses, inertia, joints, restrictions and 3D geometry. An automatically generated 3D animation allows the visualization of the system dynamics.

The study and use of the models [2] where their research makes an approximation of numerical way to the dynamic model taking its implementation in a robot type non-holonomist, the mobile is commercial called PIONNER 3DX, and improve the control algorithms that are used to make them more accurate, and introduce the advantages of specialized software [3], who use RRT (Rapidly Exploring Random Trees) algorithms to trace routes, making a non-holonomical robot capable of avoiding obstacles in a structured environment with only considering the starting point and the end point, the implementation was carried out in a commercial robot modified PIONNER P3-AT, the employment of algorithms and the possibility of artificial intelligence or to achieve that the robot learn are ideas that initiate, because these algorithms have other approaches [4], based on AI for their initials (Artificial Intelligence) employ multi agents achieving, this way, the evasion of obstacles for robots in configuration leader-follower, using a camera installed in the follower robot and applying an algorithm called (BUG2), that in general achieves to reach the end point and in case of finding an obstacle, the mobile surrounds until return to be intercepted with the line that initially would take it to the end point, the implementation is done in a LEGO robot.

It is shown in II-A a mathematical model that describes the movement of the mobile robot in a differential configuration corresponding to the implementation of the leader robot, in II-B is exported from SolidWorks to MATLAB obtaining a block model within a 3D environment for a simulation that allows you to visualize their behavior, the implementation of the leader robot only has the sensors type encoders in quadrature of DC motors, in II-C it is shown a strategy of control PID for the speeds achieving that a defined trajectory is programmed only in the robot called leader, in III a second mobile robot called slave, will have a system of artificial vision implemented in a card of development of raspberry pi 3, its purpose will be to identify the leader robot and to remain to a predetermined distance while the leader robot travels the trajectory that was programmed as it is shown in IV.

## II. ROBOT LEADER.

### A. Mathematical model of the controllable movement of a Mobile Robot

It is considered the following configuration of a mobile robot according to [5] as shown in Figure 1, which consists of two active wheels aligned by an axis, each wheel has a direct current motor and independent rotation, a third omnidirectional wheel that ensures that the platform is parallel to the XY plane, to make the corresponding analysis are defined the following parameters:

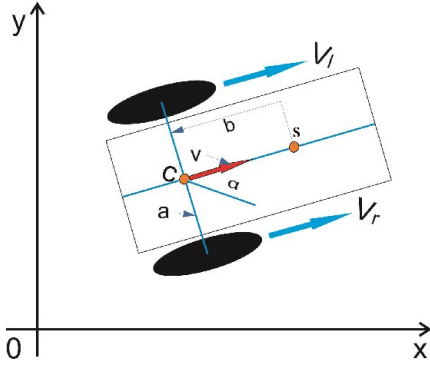


Fig. 1. Robot configuration moving in the work area.

- $c$  - midpoint of the axis connecting the two rear wheels of the robot.
- $s$  - center of mass of the robot platform.
- $a$  - length of  $c$  at the center of either of the two active wheels; left and right.
- $b$  - length of  $c$  to the center of mass of the robot platform.
- $V$  - magnitude of the vector speed of point  $c$ , with respect to the system
- $\alpha$  - robot orientation (angle formed by the X axis and the speed vector of the point  $c$ ).
- $\varphi_l$  and  $\varphi_r$  - left and right wheel  $c$

Once these elements are defined, the following concepts will be considered as the Fundamental Assumptions of the model:

- 1) The robot is composed of rigid parts (each point of the robot has a fixed position and orientation with respect to the system related to the robot).
- 2) There is no sliding in the direction perpendicular to the rolling.
- 3) There is no translational sliding between the wheel and the ground.
- 4) The center of masses of the robot may not coincide with the geometric center of it.
- 5) The space where the robot moves is perfectly flat and without inclination.

Therefore, the position of the robot can be seen as the projection of the point  $s$  in the X,Y plane, thus obtaining the speed giving rise to the model that describes its movement, as

it is stated [5]–[7], the kinematic model is given by equation 1:

$$\begin{aligned}\dot{x}_s &= V \cos \alpha - b \Omega \sin \alpha \\ \dot{y}_s &= V \sin \alpha + b \Omega \cos \alpha \\ \dot{\alpha} &= \Omega\end{aligned}\quad (1)$$

Considering the fundamental assumption above, where there is no translational or perpendicular slip, with this there are conditions that relate the speed presented by the center of masses  $cm$  of each of the wheels and the angular speed of rotation of them, being represented by equation 2:

$$v_{cm} = R\dot{\varphi} \quad (2)$$

Therefore, considering equation 2, the speeds of the right and left wheels are expressed by:

$$v_r = R\dot{\varphi}_r, \quad v_l = R\dot{\varphi}_l \quad (3)$$

So, by obtaining the speeds of each of the wheels, it is possible to calculate the linear speed of the mobile robot.

### B. SimMechanics.

The CAD design by its acronym (Computer Aided Design) is of great help due to all the characteristics and properties that can be included in the design, such as; the masses of each piece, the inertias, the restrictions of each joint, etc. As well as having a faithful reproduction of the system in a simulation that shows us its behavior, ie its position, orientation and displacement itself.

It is designed the robot leader in the tool SolidWorks 2018, where they are assembled all the pieces that it involve in their physical implementation, as it is shown in the figure 2.

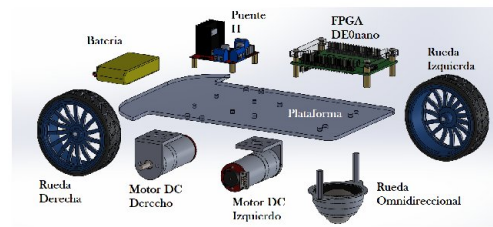


Fig. 2. CAD designed parts for assembly.

With the SimMechanics tool, the design is exported and viewed from the Simulink tool as a block model shown in Figure 3 and an animation of each of the assembled parts interacting with the features that comprise it. The mathematical model obtained and represented by the equation 1, is found inside the block Kinematics Subsystem shown in figure 4 and the Planar Joint, necessary to indicate the variable to census that in our case are the angular velocities of the DC motors, grouping the blocks that correspond to the right and left wheels and the omnidirectional wheel to have them well identified.

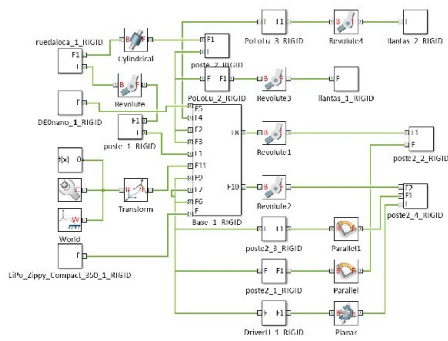


Fig. 3. Model generated by importing the system from SolidWorks with SimMechanics.

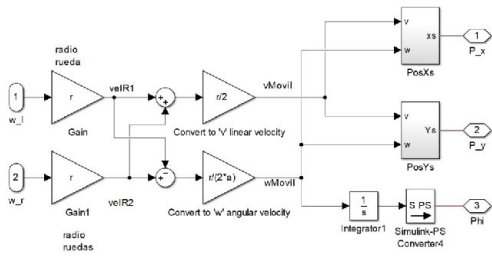


Fig. 4. Kinematic model incorporated into the system in the Kinematics Subsystem block

Now we can vary the inputs of the rIzq and rDer blocks which are the angular velocities corresponding to the DC motors connected to the wheels, and validate their behavior which is governed by the kinematics model in the Kinematics Subsystem block.

### C. PID control design

The PID controller has as input, regulate the angular speed of the DC motors, in each of the blocks that correspond to the wheels there is a sensor that provides information for the feedback of the system. As shown in the figure 5, the PID controller blocks incorporated into the system to travel a previously planned path formed by the union of straight lines and arcs of circumference.

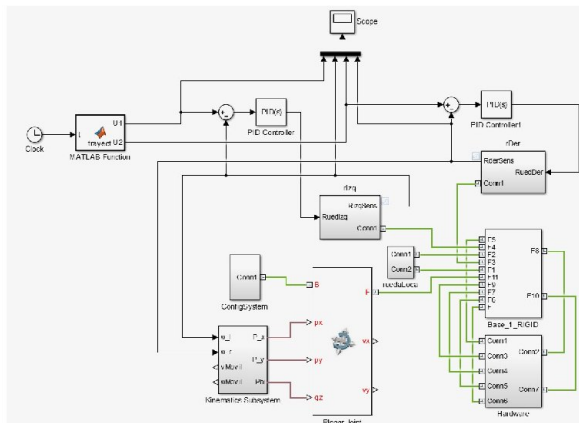


Fig. 5. PID controllers for motor angular speed

The tuning of the PID controller is done with its toolbox, for both wheels with the information of the planned path, while it is a straight line the angular speeds  $\varphi_l, \varphi_r$ . With the trajectory desired programmed in the robot leader, and the controls PID acting, it is observed how it travel the trajectory.

### III. FOLLOWER ROBOT

For the second robot we will use the same differential configuration adding a vision sensor, the development card where it is implemented is a Raspberry Pi 3, with a camera which was programmed in Python under the Raspbian environment, to follow a defined color object, which in our case will be the leader robot, who will have in its back a certain figure as well as the color for its red detection, that in our case will be a sphere of red color, that will have in its later part the leading robot.

The following algorithm is implemented to locate the object.

```

1 while True:
2     _, frame = cap.read()
3     hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
4     # red color
5     low_red = np.array([161, 155, 84])
6     high_red = np.array([179, 255, 255])
7     red_mask = cv2.inRange(hsv_frame, low_red,
8                             high_red)
9     _, contours, _ = cv2.findContours(red_mask, cv2.
10    RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
11    contours = sorted(contours, key=lambda x:cv2.
12    contourArea(x), reverse=True)
13    for cnt in contours:
14        (x, y, w, h) = cv2.boundingRect(cnt)
15        x_medium = int((x + x + w) / 2)
16        break
17    cv2.line(frame, (x_medium, 0), (x_medium, 480),
18              (0, 255, 0), 2)

```

Listing 1. Detecting the object by color and calculating its position in the center of the figure

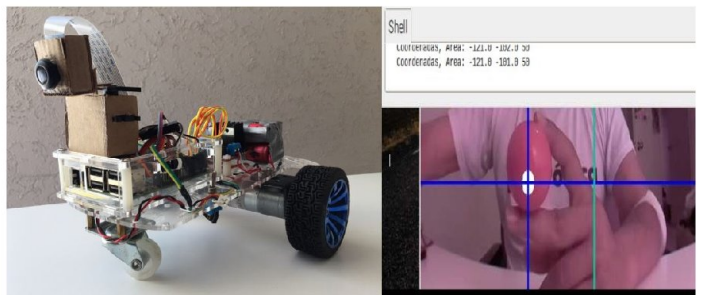


Fig. 6. Robot tracker implementation on raspberry pi 3

The information obtained of the algorithm as it is seen in the 6 indicate us the coordinate of the objective for thus to take the actions of control, depending on the area calculated of the sphere, the robot follower will advance or will owe of to rotate according to the position in where it is locate the robot leader.

## IV. RESULTS

If we consider that having the same speed in both motors, that is  $\varphi_l = \varphi_r = n$ , where  $n=1\text{rad/s}$ , it is being described



that the robot traces a straight line, as it is observed in the graph corresponding to the simulation in figure 7, with some initial conditions  $(x_0, y_0, \alpha) = (0, 0, 0)$  with  $t=30$ . For that the mobile robot go in a trajectory semicircular or a circle, already be in sense clockwise or anti-clockwise the angular velocities of the motors, they should of be different, in the same sense, for example: for a trajectory semicircular in sense clockwise  $\varphi_l > \varphi_r$ , as it is it demonstrates its simulation in the figure 9.

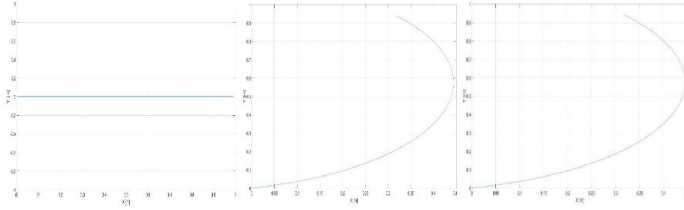


Fig. 7. Trajectory with speeds  $\varphi_l = \varphi_r$ ,  $\varphi_l > \varphi_r$  and  $\varphi_l < \varphi_r$

And in the same way, for the mobile robot to follow a semicircular trajectory counterclockwise, the angular velocities of the motors must be  $\varphi_l < \varphi_r$  as shown in the figure 7 corresponding to the simulation. A PID controller was implemented for the speeds of the direct current motors, with the objective of regulating the voltage in each one of the motors, so that the leading robot turns, or goes in straight line, the path that the leading robot travels as it is shown in the figure reffig12.

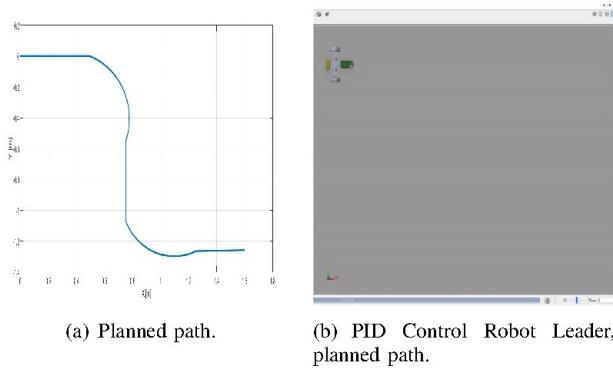


Fig. 8. Trajectory covered by the leading robot.

While the leading robot travels the defined path, the slave robot keeps the leading robot identified by means of artificial vision provided by the camera as shown in the figure 8, the raspbian operating system is accessed remotely, by means of VNC, Start the image capture algorithm with the camera so that the slave robot locates the red object, and by calculating the area of the object it can advance, when a rotation according to the quadrant and how far from the origin it is, either right or left respectively, as shown below.

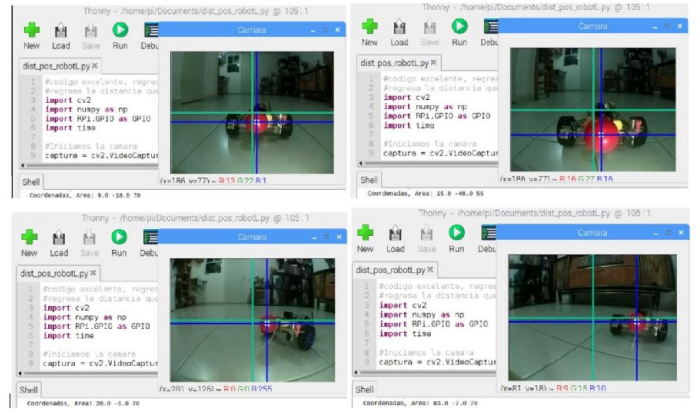


Fig. 9. Robot Vision Tracking System.

By identifying the red sphere on the back of the leading robot, the coordinates as well as the estimated area are obtained by the algorithm in the slave robot to apply the respective control actions and keep the target at the origin of the camera's field of view.

## V. CONCLUSIONS

This work allowed us to validate the option of an artificial vision system implemented in a differential type mobile robot, which only by identifying an object without the need to program the trajectory or know it, is able to travel the same route as the leading robot.

Artificial vision allows us to control the movement of the DC motors according to the position in which the object to be identified is located, it can identify different shapes, colors, even more than one at the same time, being very versatile and large. Help more slave robots join a formation following the leader's path.

With the results obtained it is possible to propose and implement an optimal control or some other strategy for the leading robot, it will be necessary to obtain the dynamic model.

## REFERENCES

- [1] Soria, C., Carelli, R., Kelly, R., and Zannatha, J. M. I. (2004). Control de Robots Cooperativos por Medio de Vision Artificial. In XVI Congreso de La Asociación Chilena de Control Automático (Vol. 223).
- [2] Rosales, A., Scaglia, G., Mut, V., and Di Sciascio, F. (2008). Control dinámico mediante métodos numéricos para robots móviles tipo unicycle. Universidad Nacional de San Juan, Argentina.
- [3] Asqui Paguay, L. M. (2017). Planificación y seguimiento de caminos de manera autónoma para robots móviles tipo unicycle. Master's thesis, Escuela Superior Politécnica de Chimborazo, 2017.
- [4] Cervera Andes, A. (2011). Coordinación y control de robots móviles basado en agentes. Universidad Politécnica de Valencia. PhD Thesis.
- [5] Corona Morales, G. (2016). Modelación matemática de un robot móvil y la síntesis de control óptimo. Benemérita Universidad Autónoma de Puebla. PhD Thesis.
- [6] Moisés, G. A. J. E., María, R. E. R., Castillo, M., and Montserrat, M. (2013). Modelado y Planificación de las Trayectorias de un Robot Limpiador. Personal de la Revista, 21.
- [7] Andaluz, G., Andaluz, V., and Rosales, A. (2011). Modelación, Identificación y Control de robots móviles. Escuela Politécnica Nacional, Quito, Tesis de Ingeniería en Electrónica y Control.

---

## Apéndice B

# Códigos Empleados

---

### B.1. Códigos VHDL

```
1 library IEEE;
2 use IEEE.std_logic_1164.ALL;
3 use IEEE.NUMERIC_STD.ALL;
4 use IEEE.std_logic_unsigned.all;
5
6 entity MatE is
7 port(
8   -- Conexiones Fisicas
9   clk_50      : in  std_logic;
10  rst         : in  std_logic;
11  pIni        : in  std_logic;
12  -- Conexiones UART
13  TX          : out std_logic;
14  RX          : in  std_logic;
15  dto_reciv   : out std_logic_vector(7 downto 0);
16  -- Conexiones Ctrl Posicion
17  PhasesABI   : in  std_logic_vector(1 downto 0);
18  PhasesABD   : in  std_logic_vector(1 downto 0);
19  -- Conexiones Motores ----
20  MotI        : out std_logic_vector(1 downto 0);
21  MotD        : out std_logic_vector(1 downto 0));
```

```

22 end entity;
23 architecture MatE_arc of MatE is
24
25 component UARTModule
26 port(
27   clk      : in  std_logic;
28   rst      : in  std_logic;
29   envIni   : in  std_logic;
30   dto_salI : in  integer;    -- dato a enviar de FPGA a la PC
31   dto_salD : in  integer;    -- dato a enviar de FPGA a la PC
32   key      : in  std_logic;  -- al pulsar el pushboton, envia ...
      dto_sal
33   ledr      : out std_logic_vector(7 downto 0);
34   ledg      : out std_logic_vector(7 downto 0);
35   txd       : out std_logic;  -- transmisor del bluetooH
36   rxd       : in  std_logic); -- receptor   del bluetooH
37 end component;
38
39 component CtrlPosModule
40 port(
41   clk      : in  STD_LOGIC;
42   TsKey    : out std_logic;
43   rst      : in  std_logic;
44   quadAI   : in  std_logic;    -- phaseA encoder motor Izquierdo
45   quadBI   : in  std_logic;    -- phaseB encoder motor Izquierdo
46   gradI    : out integer;      -- dato en distancia angular Motor...
      Izq
47   posMI    : out integer;      -- datos de la distancia lineal ...
      Motor Izq
48   velI     : out Natural;      -- velocidad motor Izquierdo
49   quadAD   : in  std_logic;    -- phaseA encoder motor Derecho
50   quadBD   : in  std_logic;    -- phaseB encoder motor Derecho
51   gradD    : out integer;      -- dato en distancia angular Motor...
      Der
52   posMD    : out integer;      -- datos de la distancia lineal ...
      Motor Der

```



```
53 velD      : out Natural);
54 end component;
55
56 component PWMModule
57 Port (
58   clk      : in  std_logic;           -- reloj de 50MHz
59   rst      : in  std_logic;           -- rst
60   Inicia   : in  std_logic;
61   posIzq   : in  integer;             -- posicion actual ...
      Izq
62   posDer   : in  integer;             -- posicion actual ...
      Derecha
63   datPc    : in  std_logic_vector(7 downto 0); -- Dato Recibido PC
64   motIzq   : out std_logic_vector(1 downto 0); -- Motor Izq
65   motDer   : out std_logic_vector(1 downto 0) -- Motor Der
66 );
67 end component;
68
69 signal fll_envDTO    : std_logic;
70 signal dto_salida    : std_logic_vector(7 downto 0);
71 signal dto_inPc      : std_logic_vector(7 downto 0);
72 signal dto_pEnv      : std_logic_vector(7 downto 0);
73 signal corr_DI       : std_logic_vector(1 downto 0):="00";
74 signal fll_blcd      : std_logic_vector(7 downto 0);
75 signal PosGradosI    : integer;
76 signal PosGradosD    : integer;
77 signal posI          : integer;
78 signal posD          : integer;
79 signal velMizq       : natural;
80 signal velMDer       : natural;
81 signal iniTx         : std_logic;
82
83 begin
84
85 process(pIni)        -- Bttm Iniciar
86 begin
```

```

87  if rst = '0' then
88      iniTx ≤ '0';
89  elsif pIni'event and pIni='0' then
90      iniTx ≤ '1';
91  end if;
92  end process;
93
94  dto_reciv≤dto_inPc;
95
96  UART : UARTModule port map(clk_50, rst, iniTx, posI, posD, ...
      fll_envDT0, dto_inPc, dto_salida, TX, RX);
97
98  CtrlPos : CtrlPosModule port map(clk_50, fll_envDT0, rst, ...
      PhasesABI(1), PhasesABI(0), PosGradosI, posI, velMizq, ...
      PhasesABD(1), PhasesABD(0), PosGradosD, posD, velMDer);
99
100 PWM : PWMModule port map(clk_50, rst, iniTx, posI, posD, dto_inPc,...
      MotI, MotD);
101
102 end architecture MatE_arc;

```

Algoritmo B.1: Top Entity:MatE

```

1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  use IEEE.std_logic_unsigned.all;
5
6  ENTITY CtrlPosModule IS
7  PORT(
8  clk      : in  STD_LOGIC;
9  TsKey    : out std_logic;
10 rst      : in  std_logic;
11 quadAI   : in  std_logic;      -- phaseA encoder motor Izquierdo
12 quadBI   : in  std_logic;      -- phaseB encoder motor Izquierdo
13 gradI    : out integer;        -- dato en distancia angular Motor...

```

```

    Izq
14 posMI    : out integer;           -- datos de la distancia lineal ...
    Motor Izq
15 velI     : out Natural;          -- velocidad motor Izquierdo
16 quadAD   : in  std_logic;        -- phaseA encoder motor Derecho
17 quadBD   : in  std_logic;        -- phaseB encoder motor Derecho
18 gradD    : out integer;          -- dato en distancia angular Motor...
    Der
19 posMD    : out integer;          -- datos de la distancia lineal ...
    Motor Der
20 velD     : out Natural);         -- velocidad motor Izquierdo
21
22 END entity;
23
24 ARCHITECTURE main OF CtrlPosModule IS
25
26 signal count3,count4              : integer ;
27 signal i_clk3, clkTs3             : std_logic;
28 signal encdrIzq, encdrDer         : std_logic_vector(1 downto 0);
29 constant tsDist                   : integer := 5000000;  -- ts h=100 ...
    ms
30
31 component Mder
32 port(
33   clk      : in  std_logic;
34   rst      : in  std_logic;
35   Dncodr   : in  std_logic_vector(1 downto 0);
36   posDer   : out integer);
37 end component Mder;
38
39 component Mizq
40 port(
41   clk      : in  std_logic;
42   rst      : in  std_logic;
43   Incodr   : in  std_logic_vector(1 downto 0);
44   posIzq   : out integer);
```

```
45 end component Mizq;
46
47 BEGIN
48 --Divisor de frecuencia para TsKey UART mandar dato de encoders--
49 process (rst, clk)
50 begin
51 if (rst = '0') then
52     count3 <= 0;
53 elsif (clk'event and clk = '1') then
54     if(count3 < tsDist/4) then
55         count3 <= count3+1;
56         i_clk3 <= '1';
57     else
58         count3 <= 0;
59         i_clk3 <= '0';
60     end if;
61     if (i_clk3 = '1') then
62         if count4 < (tsDist/1000000) then
63             clkTs3 <= '1';
64             count4 <= count4 + 1;
65         elsif count4 >=(tsDist/1000000) then
66             clkTs3 <= '0';
67             count4 <= count4 + 1;
68         end if;
69     else
70         count4 <= 0;
71     end if;
72 end if;
73 end process;
74 TsKey<= clkTs3;
75
76 encdrIzq(0)<=quadAI;
77 encdrIzq(1)<=quadBI;
78 encdrDer(0)<=quadAD;
79 encdrDer(1)<=quadBD;
80
```

```
81 C1: Mizq port map (clk, rst, encdrIzq, posMI);
82 C2: Mder port map (clk, rst, encdrDer, posMD);
83
84 end architecture main;
```

#### Algoritmo B.2: instance Name-CtrlPosModule:CtrlPos

```
1  LIBRARY IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  use IEEE.std_logic_unsigned.all;
5
6  ENTITY Mizq IS
7  port(
8  clk      : in  std_logic;
9  rst      : in  std_logic;
10 Incodr   : in  std_logic_vector(1 downto 0);
11 posIzq   : out integer);
12 end entity;
13
14 ARCHITECTURE main OF Mizq IS
15
16 signal QI1, QI2, QI3, QI4, QI5, QI6      : std_logic := '0' ;
17 signal contI, cont, cont2                 : integer;
18 signal count, count2                     : integer ;
19 -- fllContL, fllContD = habilita contadores, dirL, dirD = ...
   direccion
20 signal dirI, fllContI, i_clk, i_clk2      : std_logic;
21 signal i_clk3, i_clk4, clkTsMI, clkTs2    : std_logic;
22
23 begin
24 -----Divisor de frecuencia para encoders-----
25 process (rst, clk)
26 begin
27     if (rst = '0') then
28         count <= 0;
```

```

29     elsif (clk'event and clk = '1') then
30         if(count = 500) then          -----numero de pulsos para...
100 khz
31             count <= 0;
32             i_clk <= '1';
33         else
34             count <= count+1;
35             i_clk <= '0';
36         end if;
37     end if;
38 end process;
39
40 process (rst, i_clk)
41 begin
42     if(rst = '0') then
43         clkTsMI <= '0';
44     elsif (rising_edge(i_clk)) then
45         clkTsMI <= not clkTsMI;
46     end if;
47 end process;
48 ----Divisor de frecuencia para guardar valores de encoders Ts...
49 -----
49 process (rst, clk)
50 begin
51     if (rst = '0') then
52         count2 <= 0;
53     elsif (clk'event and clk = '1') then
54         if(count2 = 5000000) then
55             count2 <= 0;
56             i_clk2 <= '1';
57         else
58             count2 <= count2+1;
59             i_clk2 <= '0';
60         end if;
61     end if;
62 end process;

```

```
63
64 process (rst, i_clk2)
65 begin
66     if(rst = '0') then
67         clkTs2 <= '0';
68     elsif (rising_edge(i_clk2)) then
69         clkTs2 <= not clkTs2;
70     end if;
71 end process;
72 -- arreglo de FlipFlops tipo D contador y sentido de giro MIzq...
73 ---
74 PROCESS (clkTsMI, Incodr(0))
75 BEGIN
76     IF (clkTsMI'event and clkTsMI='1') THEN
77         QI1<=Incodr(0);
78     END IF;
79 END PROCESS;
80 PROCESS (clkTsMI, Incodr(1))
81 BEGIN
82     IF (clkTsMI'event AND clkTsMI='1') THEN
83         QI2<=Incodr(1);
84     END IF;
85 END PROCESS;
86
87 PROCESS (clkTsMI,QI1)
88 BEGIN
89     IF (clkTsMI'event AND clkTsMI='1') THEN
90         QI3<=QI1;
91     END IF;
92 END PROCESS;
93
94 PROCESS (clkTsMI,QI2)
95 BEGIN
96     IF (clkTsMI'event AND clkTsMI='1') THEN
97         QI4<=QI2;
```

```

98     END IF;
99     END PROCESS;
100
101     PROCESS (clkTsMI,QI3)
102     BEGIN
103         IF (clkTsMI'event AND clkTsMI='1') THEN
104             QI5< =QI3;
105         END IF;
106     END PROCESS;
107
108     PROCESS (clkTsMI,QI4)
109     BEGIN
110         IF (clkTsMI'event AND clkTsMI='1') THEN
111             QI6< =QI4;
112         END IF;
113     END PROCESS;
114
115     dirI      < = QI3 XOR QI6;
116     fllContI < = QI3 XOR QI4 XOR QI5 XOR QI6;
117     -----contador pulsos sentido motor Izquierdo-----
118     PROCESS(clkTsMI)
119     BEGIN
120         IF rst = '0' THEN
121             contI< = 0;
122             cont < =0;
123         ELSif(clkTsMI'event and clkTsMI='1') then
124             IF (fllContI='1') then
125                 if (dirI = '1') THEN
126                     contI< =contI+1;
127                 end if;
128             END IF;
129             if cont = 9999 then
130                 cont2 < = contI;
131                 contI < = 0;
132                 cont < = 0;
133             else

```



```
134         cont < = cont+1;
135     end if;
136 END IF;
137 END PROCESS;
138
139 PROCESS (clkTs2)
140 BEGIN
141     IF (clkTs2'event and clkTs2='1') THEN
142         posIzq < = cont2;    --(retroI*57)/10000;
143     END IF;
144 END PROCESS;
145 end architecture main;
```

#### Algoritmo B.3: instance Name-Mizq:C1

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity UARTModule is
6     port (
7         clk      : in  std_logic;
8         rst      : in  std_logic;
9         envIni   : in  std_logic;
10        dto_salI : in  integer;    -- dato FPGA a la PC MIZq
11        dto_salD : in  integer;    -- dato FPGA a la PC MDer
12        key      : in  std_logic;
13        ledr     : out std_logic_vector(7 downto 0);
14        ledg     : out std_logic_vector(7 downto 0);
15        txd      : out std_logic;
16        rxd      : in  std_logic);
17 end UARTModule;
18
19 architecture main of UARTModule is
20     signal tx_data : std_logic_vector(7 downto 0);
21     signal tx_start : std_logic;
```

```
22 signal tx_busy      : std_logic;
23 signal dto_MI       : std_logic_vector(19 downto 0);
24 signal dto_MD       : std_logic_vector(19 downto 0);
25 signal rx_data      : std_logic_vector(7  downto 0);
26 signal rx_busy      : std_logic;
27 signal enviar       : std_logic:='0';
28 signal muxDI        : integer range 0 to 7:= 0;
29
30 component tx
31   port(
32     clk      : in  std_logic;
33     start    : in  std_logic;
34     busy     : out std_logic;
35     data     : in  std_logic_vector(7  downto 0);
36     tx_line  : out std_logic
37   );
38 end component tx;
39
40 component rx
41   port(
42     clk      : in  std_logic;
43     rx_line  : in  std_logic;
44     data     : out std_logic_vector(7  downto 0);
45     busy     : out std_logic
46   );
47 end component rx;
48
49 begin
50
51   process(rx_busy)
52   begin
53     if(rx_busy'event and rx_busy='0') then
54       ledr(7 downto 0)≤rx_data;
55     end if;
56   end process;
57
```

```
58 process(clk)
59 begin
60     if(clk'event and clk='1') then
61         if envIni = '1' then
62             enviar ≤ '1';
63             tx_start ≤ key;
64         else
65             enviar ≤ '0';
66             tx_start ≤ '0';
67         end if;
68     end if;
69 end process;
70
71 process(rst,key)
72 begin
73     if rst = '0' then
74         muxDI ≤ 0;
75     elsif rising_edge(key) then
76         if enviar = '1' and tx_busy = '0' then
77             case (muxDI) is
78                 when 0 => tx_data(7 downto 5) ≤ "000";
79                     tx_data(4 downto 0) ≤ dto_MI(4 downto 0);
80                     muxDI ≤ muxDI+1;
81                     ledg ≤ tx_data;
82                 when 1 => tx_data(7 downto 5) ≤ "001";
83                     tx_data(4 downto 0) ≤ dto_MI(9 downto 5);
84                     muxDI ≤ muxDI+1;
85                     ledg ≤ tx_data;
86                 when 2 => tx_data(7 downto 5) ≤ "010";
87                     tx_data(4 downto 0) ≤ dto_MI(14 downto 10);
88                     muxDI ≤ muxDI+1;
89                     ledg ≤ tx_data;
90                 when 3 => tx_data(7 downto 5) ≤ "011";
91                     tx_data(4 downto 0) ≤ dto_MI(19 downto 15);
92                     muxDI ≤ muxDI+1;
93                     ledg ≤ tx_data;
```

```

94         when 4 => tx_data(7 downto 5) ≤ "100";
95             tx_data(4 downto 0) ≤ dto_MD(4 downto 0);
96             muxDI ≤ muxDI+1;
97             ledg ≤ tx_data;
98         when 5 => tx_data(7 downto 5) ≤ "101";
99             tx_data(4 downto 0) ≤ dto_MD(9 downto 5);
100            muxDI ≤ muxDI+1;
101            ledg ≤ tx_data;
102         when 6 => tx_data(7 downto 5) ≤ "110";
103            tx_data(4 downto 0) ≤ dto_MD(14 downto 10);
104            muxDI ≤ muxDI+1;
105            ledg ≤ tx_data;
106         when 7 => tx_data(7 downto 5) ≤ "111";
107            tx_data(4 downto 0) ≤ dto_MD(19 downto 15);
108            ledg ≤ tx_data;
109            muxDI ≤ 0;
110         when others => muxDI ≤ 0;
111     end case;
112 end if;
113 end if;
114 end process;
115
116 dto_MI≤std_logic_vector(to_unsigned(dto_salI, 20));
117 dto_MD≤std_logic_vector(to_unsigned(dto_salD, 20));
118
119 C1: tx port map (clk, tx_start, tx_busy, tx_data, txd);
120 C2: rx port map (clk, rxd, rx_data, rx_busy);
121
122 end main;

```

#### Algoritmo B.4: UARTModule:UART

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std;
4

```

```
5 entity tx is
6   port(
7       clk      : in  std_logic;
8       start    : in  std_logic;
9       busy     : out std_logic;
10      data     : in  std_logic_vector(7 downto 0);
11      tx_line  : out std_logic);
12 end tx;
13
14 architecture main of tx is
15   signal prscl      : integer range 0 to 5208 := 0;
16   signal index      : integer range 0 to 9 := 0;
17   signal datafl1    : std_logic_vector(9 downto 0);
18   signal tx_flg     : std_logic := '0';
19
20 begin
21   process(clk)
22   begin
23     if(clk'event and clk='1') then
24       if(tx_flg='0' and start='1') then
25         tx_flg<='1';
26         busy<='1';
27         datafl1(0)<='0';
28         datafl1(9)<='1';
29         datafl1(8 downto 1)<=data;
30       end if;
31       if(tx_flg='1') then
32         if(prscl<5207) then
33           prscl<=prscl+1;
34         else
35           prscl<=0;
36         end if;
37         if(prscl=2607) then
38           tx_line<=datafl1(index);
39           if(index<9) then
40             index<=index+1;
```

```
41         else
42             tx_flg≤'0';
43             busy≤'0';
44             index≤0;
45         end if;
46     end if;
47 end if;
48 end if;
49 end process;
50
51 end main;
```

## Algoritmo B.5: tx:C1

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity rx is
6      port(
7          clk      : in std_logic;
8          rx_line  : in std_logic;
9          data     : out std_logic_vector(7 downto 0);
10         busy     : out std_logic);
11 end rx;
12
13 architecture main of rx is
14
15     signal datafll : std_logic_vector(9 downto 0);
16     signal rx_flg  : std_logic := '0';
17     signal prscl   : integer range 0 to 5208 := 0;
18     signal index   : integer range 0 to 9 := 0;
19
20 begin
21     process(clk)
22     begin
```

```
23     if(clk'event and clk='1') then
24         if(rx_flg='0' and rx_line='0') then
25             index≤0;
26             prscl≤0;
27             busy≤'1';
28             rx_flg≤'1';
29         end if;
30         if(rx_flg='1') then
31             datafll(index)≤rx_line;
32             if(prscl≤5207) then
33                 prscl≤prscl+1;
34             else
35                 prscl≤0;
36             end if;
37             if(prscl=2500) then
38                 if(index<9) then
39                     index≤index+1;
40                 else
41                     if(datafll(0)='0' and datafll(9)='1') then
42                         data≤datafll(8 downto 1);
43                     else
44                         data≤(others=>'0');
45                     end if;
46                     rx_flg≤'0';
47                     busy≤'0';
48                 end if;
49             end if;
50         end if;
51     end if;
52 end process;
53 end main;
```

#### Algoritmo B.6: rx:C2

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
```

```

3 use IEEE.numeric_std.all;
4 use IEEE.std_logic_unsigned.all;
5
6
7 ENTITY PWMModule is
8
9 Port (
10     clk      : in  std_logic;    -- reloj de 50MHz
11     rst      : in  std_logic;    -- rst
12     Inicia   : in  std_logic;
13     posIzq   : in  integer;      -- posicion actual Izquierda
14     posDer   : in  integer;      -- posicion actual Derecha
15     datPc    : in  std_logic_vector(7 downto 0);
16     motIzq   : out std_logic_vector(1 downto 0);-- salida PWM Motor...
17     Izq      : out std_logic_vector(1 downto 0);
18     motDer   : out std_logic_vector(1 downto 0);-- Salida PWM ...
19     Motor Der
20 end PWMModule ;
21
22 ARCHITECTURE behavioral of PWMModule is
23
24     signal count, count1      : integer:=0;
25     signal count2, count3     : integer:=0;
26     signal count4, count5     : integer range 0 to 100:=0;
27     signal i_clk1, i_clk2     : std_logic;
28     signal iniT1              : std_logic:= '0';
29
30 begin
31     ----- contador para ajustar el % efectivo del PWM -----
32     process(rst, datPc, clk)
33     begin
34         if rst = '0' then
35             iniT1 ≤ '0';
36         elsif rising_edge(clk) then
37             if Inicia = '1' then
38                 iniT1 ≤ '1';

```



```
37     if datPc(7)='0' then
38         count4 ≤ to_integer(unsigned(datPc(6 downto 0)));
39     else
40         count5 ≤ to_integer(unsigned(datPc(6 downto 0)));
41     end if;
42     elsif count4 > 90 then
43         count4 ≤ 90;
44     elsif count5 > 90 then
45         count5 ≤ 90;
46     else
47         count4 ≤ 0;
48         count5 ≤ 0;
49     end if;
50 end if;
51 end process;
52 --- contador y generador de PWM a frecuencia 245Hz rueda Izq ...
53 -----
54 process (rst, clk)
55 begin
56     if (rst = '0') then
57         count ≤ 0;
58     elsif (clk'event and clk = '1') then
59         if(count < 204081) then
60             count ≤ count + 1;
61             i_clk1 ≤ '1';
62         else
63             count ≤ 0;
64             i_clk1 ≤ '0';
65         end if;
66         if i_clk1 = '1' and iniT1 = '1' then
67             if (count1 < ((204081 * (count4+10)) / 100)) then
68                 motIzq ≤ "10";
69                 count1 ≤ count1 + 1;
70             elsif (count1 ≥ ((204081 * (count4+10)) / 100)) then
71                 motIzq ≤ "00";
72                 count1 ≤ count1 + 1;
```

```

72         end if;
73     else
74         count1 ≤ 0;
75     end if;
76 end if;
77 end process;
78 --- contador y generador de PWM a frecuencia 195Hz rueda Der ---
79 process (rst, clk)
80 begin
81     if (rst = '0') then
82         count2 ≤ 0;
83     elsif (clk'event and clk = '1') then
84         if(count2 < 256410) then
85             count2 ≤ count2 + 1;
86             i_clk2 ≤ '1';
87         else
88             count2 ≤ 0;
89             i_clk2 ≤ '0';
90         end if;
91         if i_clk2 = '1' and iniT1 = '1' then
92             if (count3 < ((256410 * (count5+10)) / 100)) then
93                 motDer ≤ "10";
94                 count3 ≤ count3 + 1;
95             elsif (count3 ≥ ((256410 * (count5+10)) / 100)) then
96                 motDer ≤ "00";
97                 count3 ≤ count3 + 1;
98             end if;
99         else
100             count3 ≤ 0;
101         end if;
102     end if;
103 end process;
104 end behavioral;

```

Algoritmo B.7: PWMModule:PWM

## B.2. Códigos Matlab

```

1 t=0.1;
2 LSol= [ - 2.45952e-7*t^8 + 0.00000916291*t^7 - 0.000138751*t^6 + ...
          0.00109649*t^5 - 0.00482655*t^4 + 0.0116754*t^3 - 0.0142437*t^2...
          + 0.00709457*t + 0.0142724, 3.75177e-9*t^8 - 1.38438e-7*t^7 + ...
          0.00000207673*t^6 - 0.0000162615*t^5 + 0.0000709415*t^4 - ...
          0.00017012*t^3 + 0.000205818*t^2 - 0.000101732*t - 0.000234011,...
          0.00000381945*t^8 - 0.000142254*t^7 + ...
          0.00215342*t^6 - 0.0170116*t^5 + 0.0748533*t^4 - 0.180995*t^3 +...
          0.22071*t^2 - 0.10988*t - 0.196381,
          0.0000179535*t^8 + 0.00066889*t^7 - 0.0101288*t^6 + 0.0800414*t...
          ^5 - 0.352301*t^4 + 0.852116*t^3 - 1.03939*t^2 + 0.517594*t + ...
          0.920284,
          0.00224824*t^8 - ...
          0.0837355*t^7 + 1.26759*t^6 - 10.0139*t^5 + 44.0633*t^4 - ...
          106.548*t^3 + 129.934*t^2 - 64.694*t - 115.583;
3
  3.75177e-9*t^8 - 1.38438e-7*t^7 + 0.00000207673*t^6 - ...
  0.0000162615*t^5 + 0.0000709415*t^4 - 0.00017012*t^3 + ...
  0.000205818*t^2 - 0.000101732*t - 0.000234011,
  - 2.45952...
  e-7*t^8 + 0.00000916291*t^7 - 0.000138751*t^6 + 0.00109649*t^5 ...
  - 0.00482655*t^4 + 0.0116754*t^3 - 0.0142437*t^2 + 0.00709457*t...
  + 0.0142724,
  - 0.00000381945*t^8 + 0.000142254*...
  t^7 - 0.00215342*t^6 + 0.0170116*t^5 - 0.0748533*t^4 + ...
  0.180995*t^3 - 0.22071*t^2 + 0.10988*t + 0.196381,
  - 0.0000179535*t^8 + 0.00066889*t^7 - 0.0101288*t^6 + ...
  0.0800414*t^5 - 0.352301*t^4 + 0.852116*t^3 - 1.03939*t^2 + ...
  0.517594*t + 0.920284,
  - ...
  0.00224824*t^8 + 0.0837355*t^7 - 1.26759*t^6 + 10.0139*t^5 - ...
  44.0633*t^4 + 106.548*t^3 - 129.934*t^2 + 64.694*t + 115.583;
4
  0.00000381945*t^8 - 0.000142254*t^7 + 0.00215342*t^6 - ...
  0.0170116*t^5 + 0.0748533*t^4 - 0.180995*t^3 + 0.22071*t^2 - ...
  0.10988*t - 0.196381,
  - 0.00000381945*t^8 + ...
  0.000142254*t^7 - 0.00215342*t^6 + 0.0170116*t^5 - 0.0748533*t...
  ^4 + 0.180995*t^3 - 0.22071*t^2 + 0.10988*t + 0.196381,
  ...

```

```

- 0.000203766*t^8 + 0.0075594*t^7 - ...
0.113984*t^6 + 0.89691*t^5 - 3.93097*t^4 + 9.46748*t^3 - ...
11.499*t^2 + 5.70155*t + 10.7504, 8.23937e-20*t^8 - 3.23963e...
-18*t^7 + 5.14857e-17*t^6 - 4.24302e-16*t^5 + 1.93771e-15*t^4 -...
4.86812e-15*t^3 + 6.25491e-15*t^2 - 3.30753e-15*t + 3.98462e...
-16,
- 0.119917*t^8 + 4.44873*t...
^7 - 67.0803*t^6 + 527.844*t^5 - 2313.46*t^4 + 5571.95*t^3 - ...
6767.84*t^2 + 3355.99*t + 6325.62;
5 - 0.0000179535*t^8 + 0.00066889*t^7 - 0.0101288*t^6 + ...
0.0800414*t^5 - 0.352301*t^4 + 0.852116*t^3 - 1.03939*t^2 + ...
0.517594*t + 0.920284,
- 0.0000179535*t^8 + ...
0.00066889*t^7 - 0.0101288*t^6 + 0.0800414*t^5 - 0.352301*t^4 +...
0.852116*t^3 - 1.03939*t^2 + 0.517594*t + 0.920284, 8.23937e...
-20*t^8 - 3.23963e-18*t^7 + 5.14857e-17*t^6 - 4.24302e-16*t^5 +...
1.93771e-15*t^4 - 4.86812e-15*t^3 + 6.25491e-15*t^2 - 3.30753e...
-15*t + 3.98462e-16,
- ...
0.00466364*t^8 + 0.173113*t^7 - 2.61175*t^6 + 20.5628*t^5 - ...
90.1731*t^4 + 217.296*t^3 - 264.067*t^2 + 131.002*t + 243.634, ...
4.3565e-17*t^8 - 1.75258e-15*t^7 + 2.86713e-14*t^6 - 2.45103e...
-13*t^5 + 1.17127e-12*t^4 - 3.10207e-12*t^3 + 4.21154e-12*t^2 -...
2.38096e-12*t + 3.12079e-13;
6 0.00224824*t^8 - 0.0837355*t^7 + 1.26759*t^6 - 10.0139*t^5 + ...
44.0633*t^4 - 106.548*t^3 + 129.934*t^2 - 64.694*t - 115.583, ...
- 0.00224824*t^8 + 0.0837355*t^7 ...
- 1.26759*t^6 + 10.0139*t^5 - 44.0633*t^4 + 106.548*t^3 - ...
129.934*t^2 + 64.694*t + 115.583,
- 0.119917*t^8 + 4.44873*t^7 - 67.0803*t^6 + 527.844*t^5 - ...
2313.46*t^4 + 5571.95*t^3 - 6767.84*t^2 + 3355.99*t + 6325.62, ...
4.3565e-17*t^8 - 1.75258e-15*t^7 + 2.86713e-14*t^6 - 2.45103e...
-13*t^5 + 1.17127e-12*t^4 - 3.10207e-12*t^3 + 4.21154e-12*t^2 -...
2.38096e-12*t + 3.12079e-13,
- 70.5844*t...
^8 + 2618.63*t^7 - 39486.1*t^6 + 310721.0*t^5 - 1361911.0*t^4 +...
3280400.0*t^3 - 3984988.0*t^2 + 1976588.0*t + 3722311.0
7 ];
8
9 R=1*eye(2);

```

```
10 B =[      0,      0;
11      0,      0;
12      0,      0;
13      0.228692,      0;
14      0,      0.00182135];
15
16 der='1';
17 izq='0';
18 n=2;
19 a=7.5;
20 b=8;
21 diametro=6.73;
22 radio=diámetro/2;
23 x0=[0,0];
24 xp0=0;
25 v0=0.18;
26 Ts=0.1;
27 phi2=0;
28 NumD=8;
29 VD=20;
30 global Mat
31 j=2;
32 while 1
33     if Mat.BytesAvailable == NumD
34         t1(1,n)=t;
35         dato(:,n)=fread(Mat);
36         distMot=burbuja2(dato(:,n));
37         pMovil(:,n)=distMot;
38         %----- RPS c/rueda---Velocidad-----
39         RPSL(1,n)=distMot(1)*0.0021014232; %5310 pulsosx vuelta
40         RPSR(1,n)=distMot(2)*0.001894761904762; %5250 px v
41         % RPM
42         RPML(1,n)=RPSL(1,n)*60;
43         RPMR(1,n)=RPSR(1,n)*60;
44         %-----velocidades lin cm*s = Vi,Vd-----
45         DL(1,n)=pi*diámetro*RPSL(1,n);
```

```

46     DR(1,n)=pi*diametro*RPSR(1,n); %veloc lin cm*s = Vi,Vd
47     %-----Distancia recorrida-----ok-----
48     DLT(n)=distMot(1)*0.00406594230769230769230769230769;
49     DLT1(1,n)=trapz(DLT);
50     DRT(n)=distMot(2)*0.00391535185185185185185185185185;
51     DRT1(1,n)=trapz(DRT);
52     DMovil(1,n)=DRT1(1,n)+DLT1(1,n)/2; %distancia punto c
53     %-----Velocidad Ang Movil-----
54     Wi(1,n) = 2*pi*RPSL(1,n);
55     Wd(1,n) = 2*pi*RPSR(1,n); % velocidad cm/seg
56     %-----Velocidades lineales-----ok-----
57     Vi(1,n) = Wi(1,n) * radio;
58     Vd(1,n) = Wd(1,n) * radio;
59     %-----Velocidades lin (ok) y ang Movil (ok)-
60     VMovil(1,n) = (Vd(1,n)+Vi(1,n))/2;
61     WMovil(1,n) = (Wd(1,n)-Wi(1,n))/(2*a);
62     %-----PHI error de desviacion-----ok-----
63     PHI(1,n)=(trapz(WMovil));
64     PHI(1,n)=atan2(sin(PHI(1,n)),cos(PHI(1,n)));
65     %----- Xs-----
66     xsp(1,n)=VMovil(1,n)*cos(PHI(1,n)) - b*WMovil(1,n)*sin(PHI...
(1,n));
67     xs1(1,n)=trapz(xsp)*t1(1,n);
68     %----- Ys -----
69     ysp(1,n)=VMovil(1,n)*sin(PHI(1,n)) + b*WMovil(1,n)*cos(PHI...
(1,n));
70     ys1(1,n)=trapz(ysp)*t1(1,n);
71     %----- trayectoria deseada-----
72     yd(:,n)=[v0*t1(1,n);0;pi/4;v0;0];
73     %----- var edos-----
74     Xt(:,n)=[xs1(1,n);ys1(1,n);PHI(1,n);VMovil(1,n);WMovil(1,n)...
)];
75
76     %Xt(:,n)=[xs1(1,n);ys1(1,n);PHI(1,n);VMovil(1,n);WMovil(1,...
n)];
77     Xtilde(n,:)=yd(:,n)-Xt(:,n);

```

```
78      %   Controles
79      SolP=double(subs(LSol));
80      K=inv(R)*B'*SolP;
81      U=-K*Xtilde(n,:)' ;
82      Tau1(1,n)=(U(1,1)-U(2,1))/2;      % tau Ml
83      Tau2(1,n)=U(1,1)-Tau1(1,n);      % tau Mr
84      TAUL(1,n)=round(Tau1(1,n));
85      TAUR(1,n)=round(Tau2(1,n));
86
87      if TAUL(1,n) > 126
88          TAUL(1,n) = 127;
89      elseif TAUL(1,n) < 0
90          TAUL(1,n) = 0;
91      end
92      if TAUR(1,n) > 126
93          TAUR(1,n) = 127;
94      elseif TAUR(1,n) < 0
95          TAUR(1,n) = 0;
96      end
97      PWML=dec2bin(TAUL(1,n),7);
98      PWML1=strcat(izq,PWML);
99      PWML2(1,n)=bin2dec(PWML1);
100     PWMR=dec2bin(TAUR(1,n),7);
101     PWMR1=strcat(der,PWMR);
102     PWMR2(1,n)=bin2dec(PWMR1);
103     if n > 5
104         fwrite(Mat,PWMR2(1,n));
105         fwrite(Mat,PWML2(1,n));
106     end
107     n=n+1;
108     t=t+0.1;
109 else
110     disp('Esperando datos...')
111 end
112 end
```

## Algoritmo B.8: Comunicacion con Robot Líder y Matlab

```

1  function aux1=burbuja2(dato)
2  d1=dec2bin(dato(1),8);
3  d2=dec2bin(dato(2),8);
4  d3=dec2bin(dato(3),8);
5  d4=dec2bin(dato(4),8);
6  d5=dec2bin(dato(5),8);
7  d6=dec2bin(dato(6),8);
8  d7=dec2bin(dato(7),8);
9  d8=dec2bin(dato(8),8);
10 A=strcat(d1,d2,d3,d4,d5,d6,d7,d8);
11 M='...
    0002222200133333010444440115555510066666101777771108888811199999...
    ';
12 for p=1:8:64
13     for q=1:8:64
14         if eval(A(p:p+2)) == eval(M(q:q+2))
15             aux(q:q+7)=A(p:p+7);
16         end
17     end
18 end
19
20 distMot=[strcat(aux(28:32),aux(20:24),aux(12:16),aux(4:8));
21          strcat(aux(60:64),aux(52:56),aux(44:48),aux(36:40))];
22 aux1=bin2dec(distMot);
23
24 end

```

## Algoritmo B.9: Algoritmo de Burbuja

## B.3. Códigos Python



```
1 import cv2
2 import numpy as np
3 import RPi.GPIO as GPIO
4 from time import sleep
5
6 #Iniciamos la camara
7 captura = cv2.VideoCapture(0)
8
9 #resolucion
10 captura.set(3, 320)
11 captura.set(4, 240)
12
13 #inicializacion de ctrl Motor
14 #servoPIN = 13
15 MdA = 16
16 MiA = 26
17
18 GPIO.setmode(GPIO.BCM)
19
20 GPIO.setup(MdA, GPIO.OUT)
21 GPIO.setup(MiA, GPIO.OUT)
22
23 # Frecuencias de operacion
24 Dhz = 195
25 Ihz = 245
26
27 rder = GPIO.PWM(MdA,195)
28 rizq = GPIO.PWM(MiA,245)
29
30 while(captura.isOpened()):
31     rder.start(0)
32     rizq.start(0)
33     #Capturamos una imagen y la convertimos de RGB -> HSV
34     ret, imagen = captura.read()
35     h,w,m=imagen.shape
```

```
36     M = cv2.getRotationMatrix2D((w//2,h//2),180,1)
37     imagenR=cv2.warpAffine(imagen,M,(w,h))
38
39     if (ret):
40         hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
41         #Pelota roja a detectar
42         bajos = np.array([161, 155, 84], dtype=np.uint8)
43         altos = np.array([179, 255, 255], dtype=np.uint8)
44         #Mascara con solo los pixeles dentro del rango
45         mask = cv2.inRange(hsv,bajos,altos)
46         #Encontrar el area de los objetos que detecta la camara
47         moments = cv2.moments(mask)
48         area =int( moments['m00'])
49         x=0
50         y=0
51         dist=0
52
53         if(area > 100000):
54             #Buscamos el centro x, y del objeto
55             x = int(moments['m10']/moments['m00'])
56             y = int(moments['m01']/moments['m00'])
57
58             #se calcula la distancia respecto al area
59             if area≥845695.7:
60                 if area≥2301333.5:
61                     dist=-((area/307781.175)+37.4771743)
62             if area<2301333.5:
63                 if area≥859737.4:
64                     dist=-((area/72079.83)+61.9275725)
65             if area<859737.4:
66                 dist=-((area/19652.8425)+93.74622211)
67             if int(dist)≤15:
68                 dist=15
69             if int(dist)≥100:
70                 dist=100
71             # Acciones de control
```

```
72         if int(dist) > 15:
73             rder.ChangeDutyCycle(78)
74             rizq.ChangeDutyCycle(70)
75         if int(dist) > 65:
76             rder.ChangeDutyCycle(100)
77             rizq.ChangeDutyCycle(100)
78         if ((x-w/2) > 40):
79             rder.ChangeDutyCycle(80)
80             rizq.ChangeDutyCycle(50)
81         if ((x-w/2) < -40):
82             rder.ChangeDutyCycle(50)
83             rizq.ChangeDutyCycle(80)
84         if int(dist) < 15:
85             rder.ChangeDutyCycle(0)
86             rizq.ChangeDutyCycle(0)
87
88         #Mostramos sus coordenadas y distancia de la pelota
89         print ("Coordenadas, Dist:", (x-w/2),(-(y-h/2)),int(...
dist))
90
91         #Dibujamos un rectangulo en el centro del objeto
92         cv2.rectangle(imagen, (x-12, y-12), (x+12, y+12)...
,(255,255,255), 2)
93
94         #Marca el centro del area encontranda
95         cv2.line(imagen,(0,y),(w,y),(255,0,0),2)
96         cv2.line(imagen,(x,0),(x,h),(255,0,0),2)
97
98         #Divide la imagen en cuatro cuadrantes
99         cv2.line(imagen,(0,int(h/2)),(w,int(h/2)),(150,200,0),2)
100         cv2.line(imagen,(int(w/2),0),(int(w/2),h),(150,200,0),2)
101
102         #Mostramos la imagenes obtenidas
103         cv2.imshow('mask', mask)
104         cv2.imshow('Camara', imagen)
105
106         tecla = cv2.waitKey(5) & 0xFF
```

```
106         if tecla == ord('q'):  
107             break  
108  
109 cv2.destroyAllWindows()  
110 p.stop()  
111 GPIO.cleanup()
```

Algoritmo B.10: Algoritmo en Python para identificar y seguir a robot líder

# Bibliografía

---

- [1] Carlos Soria, Ricardo Carelli, Rafael Kelly, and Juan M Ibarra Zannatha. Control de robots cooperativos por medio de visión artificial. In *XVI Congreso de La Asociación Chilena de Control Automático*, volume 223, 2004. [XIV](#)
- [2] Bugarin-Carlos Eusebio and Aguilar-Bustos Ana Yaveni. Control visual para la formación de robots móviles tipo unicycle bajo el esquema líder-seguidor. *Ingeniería, investigación y tecnología*, 15(4):593–602, 2014. [XIV](#)
- [3] Cano Usiña, Grace Paola, Palaquibay Inga, and Diego José. Control cooperativo de robots utilizando fpga's. B.S. thesis, QUITO/EPN/2014, 2014. [XIV](#)
- [4] Giovanni Rodrigo Bermudez-Bohorquez, Christian Anderson Mancipe-Bernal, and Joan Emmanuel Ortiz-Velasquez. Control de un convoy robótico mediante planificación de rutas y estrategias de orientación. *Revista científica*, 3(30):237–251, 2017. [XIV](#)
- [5] Andrés Rosales, Gustavo Scaglia, Vicente Mut, and Fernando Di Sciascio. Control dinámico mediante métodos numéricos para robots móviles tipo unicycle. *Universidad Nacional de San Juan, Argentina*, 2008. [XIV](#)
- [6] Felipe Nascimento Martins, Ricardo Carelli, Mário Sarcinelli-Filho, and TF Bastos-Filho. Modelo dinámico y controlador de seguimiento para robots móviles tipo unicycle. *V Jornadas Argentinas de Robótica-JAR08*, 2008. [XIV](#)
- [7] Gabriela Andaluz. Modelación, identificación y control de robots móviles. *Escuela Politécnica Nacional, Quito, Tesis de Ingeniería en Electrónica y Control*, 2011. [XIV](#)
- [8] Asqui Paguay and Leonardo Manuel. Planificación y seguimiento de caminos de manera autónoma para robots móviles tipo unicycle. Master's thesis, Escuela Superior Politécnica de Chimborazo, 2017. [XIV](#)
- [9] Adrián Cervera Andrés. *Coordinación y control de robots móviles basado en agentes*. PhD thesis, 2011. [XV](#)

- [10] Gabriela Magdalena Andaluz Ortiz. Modelación y control predictivo de un robot móvil con centro de masa desplazado. Master's thesis, Escuela Superior Politécnica de Chimborazo, 2018. [xv](#)
- [11] Jacobo Sandoval Gutiérrez. *Trayectorias para robots móviles de navegación autónoma*. PhD thesis. [xv](#)
- [12] Ivan Arturo Calle Flores. Navegación autónoma de un robot móvil usando técnicas probabilísticas de localización y mapeo basadas en métodos monte carlo secuenciales. 2014. [xv](#)
- [13] MN Ibarra Bonilla, FJ Quiñones Novelo, IJ García Enríquez, and JM Ramírez Cortés. Desplazamiento de un robot con localización y evasión de obstáculos por visión y ultrasonido. In *Congreso Nacional de Ingeniería Electrónica del Golfo-CONAGOLFO*, pages 3–7, 2009. [xv](#)
- [14] Rafael Esteban Arango Sanchez, Leidy Diana Jiménez Pinzón, and Jaime Alberto Guzmán Luna. Búsqueda de la ruta óptima mediante los algoritmos: genético y dijkstra utilizando mapas de visibilidad. *Scientia et technica*, 2(51):107–112, 2012. [xv](#)
- [15] Terenzio Soldovieri. Introducción a la mecánica la lagrange y hamilton, 1. *Universidad del Zulia. Preprint*, 2010. [13](#), [14](#)
- [16] Igor V Novozhilov. *Fractional analysis: Methods of motion decomposition*. Springer Science & Business Media, 2012. [21](#)
- [17] Benjamin C Kuo. *Sistemas de control automático*. Pearson Educación, 1996. [24](#), [78](#)
- [18] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002. [24](#), [78](#)
- [19] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*, volume 1. Wiley-interscience New York, 1972. [32](#), [49](#)
- [20] John M Davis, Ian A Gravagne, Billy J Jackson, II Marks, and J Robert. Controllability, observability, realizability, and stability of dynamic linear systems. *arXiv preprint arXiv:0901.3764*, 2009. [33](#), [34](#)
- [21] Gregoria Corona. *Modelación Matemática de un Robot Móvil y la Síntesis de Control Óptimo*. PhD thesis, Facultad de Ciencias Físico Matemáticas, Postgrado en Ciencias Matemáticas, feb 2016. [39](#), [51](#), [53](#), [55](#), [59](#)
- [22] VV Alexandrov, Y Bolotin, SS Lemak, NA Parusnikov, SI Zlochevsky, and SWF Guerrero. Introduction to control of dynamic systems. *Ediciones BUAP*, page 239, 2009. [43](#)

- [23] Paulo Henrique Tozoni Palma. Identificação experimental e controle ativo de vibrações aplicadas em estruturas inteligentes. 2007. [45](#)
- [24] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994. [46](#)
- [25] Gary Balas, Richard Chiang, Andy Packard, and Michael Safonov. Robust control toolbox. *For Use with Matlab. User's Guide, Version, 3*, 2005. [47](#), [81](#)
- [26] Emilio Cerdá Tena. *Optimización dinámica*. Pearson Educación, 2001. [48](#)
- [27] Vasile Sima. *Algorithms for linear-quadratic optimization*, volume 200. CRC Press, 1996. [49](#)
- [28] Serge Lang. *Introducción al álgebra lineal*. Addison-Wesley Iberoamericana,, 1990. [55](#)
- [29] Fernando Reyes Cortés. *Robótica, control de robots manipuladores*. Alfa Omega, Benemérita Universidad Autónoma de Puebla, segunda edition, 2011. [87](#), [120](#)