**CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL**

UNIDAD ZACATENCO
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE MECATRÓNICA

# Compensación de retardos para teleoperación bilateral de robots móviles diferenciales

Tesis que presenta:
**Julio Alejandro Vences Jiménez**

Para obtener el grado de:
**Maestro en Ciencias**

En la especialidad de:
**Ingeniería Eléctrica**

Director de la Tesis:
**Dr. Alejandro Rodríguez Ángeles**
**Dr. Jaime Álvarez Gallegos**

México, Distrito Federal. Febrero 2018

II

# Agradecimentos

*A Dios por permitirme continuar en este trayecto llamado vida y por darme la hermosa familia y amigos que tengo y poder así una vez más lograr otro sueño.*

*A mi Madre por siempre estar ahí para mi, apoyándome, aconsejándome y amándome tanto cómo solo mi madre lo puede hacer. Por ser además un ejemplo de fuerza y voluntad que siempre me motiva a continuar con mis sueños.*

*A mi Padre por ser parte de mi vida y apoyarme siempre en mis decisiones y por guiarme en el camino de la ingeniería siendo un ejemplo de un gran ingeniero.*

*A mis Hermanos por motivarme y apoyarme a alcanzar este objetivo, ya que sin ellos habría sido imposible concluir mis estudios.*

*A mi Abuelita por sus bendiciones y velar por mi en todo momento y por su amor incondicional.*

*A mis Tíos que me han apoyando toda mi vida desde que era pequeño y que siempre estarán ahí para mi.*

*A mis Primos mis también hermanos que me animan y apoyan en todo momento y contagian de su alegría, que también me han permitido ser un ejemplo más para ellos.*

*A mi Ben mi compañero que ha sido una mano extendida durante este camino juntos, ayudándome a no caer después de tanto estrés, trabajo y malos momentos.*

*A mis Asesores por guiarme a través de su experiencia y conocimientos que me han permitido terminar satisfactoriamente este trabajo de investigación.*

*A mis Amigos y Compañeros del CINVESTAV que durante muchas horas de buenos y a veces malos ratos hemos compartido la experiencia de estar en esta institución y esta etapa de nuestras vidas.*

*Al Auxiliar M.en C. Igor Morett Valenzulea por el préstamo de equipo especializado en este proyecto y por compartir sus conocimientos en el área experimental.*

*Al CINVESTAV y CONACYT por su apoyo económico a través del proyecto CB 254329 y por la beca para realizar mis estudios de Maestría a través del cvu 702709.*

# Resumen

Este trabajo de tesis presenta la teleoperación bilateral tipo maestro-esclavo de robots móviles diferenciales por medio de la implementación de la estrategia de control por modelo interno. El sistema de teleoperación considera los retardos inherentes en la comunicación, por lo cual, con el diseño del controlador se pretende compensar de forma estimada los efectos de los retardos presentes en el sistema.

El esquema de teleoperación consta de dos robot móviles diferenciales, maestro y esclavo. El robot maestro presente en el sitio local, es el encargado de realizar el seguimiento de la trayectoria de referencia asignada por un operador, así mismo, envía su posición al robot esclavo el cual sigue la trayectoria del robot maestro en un sitio remoto y envía como retroalimentación su posición al robot maestro. Los retardos considerados en el sistema se encuentran presentes en el envío de datos del maestro al esclavo y viceversa.

El diseño del controlador por modelo interno o IMC por sus siglas en inglés, requiere del modelo de la planta, para este caso en particular el modelo del robot móvil diferencial. El modelo cinemático clásico del robot presenta no linealidades, por lo que se realiza una linealización del modelo por medio de una retroalimentación dinámica con el objetivo de obtener de forma directa el controlador, debido a que se requiere la inversión de la función de transferencia de la planta. En el diseño del IMC se requiere una retroalimentación del error entre la planta y su modelo, por tal motivo se utiliza un controlador PID que permita reducir este error entre el robot y su modelo. El diseño de un filtro para perturbaciones tipo aditivas es presentado así como el diseño de un predictor de Smith que compense los efectos de los retardos en la comunicación.

La plataforma experimental utilizada en este trabajo se efectuó con dos robots Pioneer 3DX de la marca Mobile Robots. Las mediciones de las posiciones de ambos robots se realizaron por medio de odometría, es decir calculando la distancia recorrida del robot de acuerdo con las velocidades en cada llanta. La comunicación con ambos robots se hizo de forma inalámbrica sobre una red local WLAN. Los retardos presentes en los experimentos se llevaron a cabo por medio de un buffer, para simular retardos aproximados en una comunicación por internet. El control fue realizado en el lenguaje de programación c++ con el programa Visual Studio 2010.

# Abstract

This thesis presents bilateral master-slave teleoperation of differential mobile robots through the strategy of internal model control or IMC by its acronym. The teleoperation system considers the inherent delays in the communication, therefore, the design of the controller is intended to compensate through estimations the effects of the delays present in the system.

The teleoperation scheme consists of two differential mobile robots, master and slave. The master robot present in the local site, is responsible for tracking the reference trajectory assigned by an operator, likewise, it sends its position to the slave robot which follows the path of the master robot at a remote site and sends as feedback its position to the master robot. The delays considered in the system are present in the transmission of data from the master to the slave and vice versa.

The IMC requires the model of the plant, for this case in particular the model of the differential mobile robot. The classic kinematic model of the robot presents non-linearities, therefore a linearization of the model is performed by a dynamic feedback with the objective of directly obtaining the controller, because of the inversion of the transfer function of the plant. In the design of the IMC a feedback of the error between the plant and its model is required, for this reason a PID controller is used to reduce the error between the robot and its model. The design of a filter for additive disturbances is presented as well as the design of a smith predictor that compensates the effects of delays in communication.

The experimental platform used in this work was carried out with two Pioneer 3DX robots of the Mobile Robots brand. The measurements of the positions of both robots were made with odometry, this means calculating the distance traveled of the robot according to the speeds of each wheel. Communication with both robots was implemented wirelessly over a local WLAN network. The delays present in the experiments were carried out by a buffer, to simulate delays in an internet communication. The control was made in the programming language c ++ with the program Visual Studio 2010.

# Contenido

# Lista de figuras

# Lista de tablas

# Capítulo 1

# Introducción

## 1.1. Generalidades.

Hoy en día la robótica móvil es una de las áreas de investigación y aplicación con más desarrollo dentro de la ingeniería, debido a que es posible realizar una amplia gama de actividades que brindan mayor flexibilidad, autonomía y seguridad. La robótica móvil es el área de la robótica que se encarga del estudio de los robots móviles. Se entiende por robot móvil a una máquina o servomecanismo capaz de desplazarse de un lugar a otro en un determinado espacio de trabajo sin la intervención física del hombre. Los robots móviles se pueden clasificar en tres tipos dependiendo del ambiente donde realicen las actividades, estos pueden ser terrestres, acuáticos y aéreos. Este trabajo de tesis fue desarrollado con la utilización de robots móviles terrestres por lo que por lo consecuente se referirá a los robots móviles terrestres como robots móviles de forma indistinta.

Existen múltiples aplicaciones donde los robots móviles juegan un papel importante y que hoy en día no es posible concebir la idea de realizar este tipo de tareas sin la ayuda de éstos. Tal es el caso de búsqueda y rescate dónde el robot es utilizado para acceder a lugares peligrosos y difíciles de llegar para una persona, este tipo de actividad es utilizada en siniestros como terremotos, incendios, derrumbes en minas etc. Otra actividad conocida en la cual los robots móviles facilitan las tareas, es la exploración espacial [3], en este tipo de tareas el robot es enviado para reconocer el terreno y recolectar material e información necesaria para ayudar a futuras expediciones. Así también el uso de estos robots en el sector industrial ha sido ampliamente empleado en tareas como almacenamiento de paquetería, transporte de materia prima, limpieza y transporte de residuos peligrosos entre otros. Otro sector que también se ha beneficiado del uso de estos robots es el militar, utilizando vehículos capaces de detectar y eliminar un blanco determinado, así como detectar minas y desactivar bombas; estas tareas dónde el usuario se encuentra constantemente en ambientes peligrosos son evitadas al utilizar robots, por lo cual este sector es uno de los que más desarrollo tiene en la robótica móvil.

En las tareas mencionadas anteriormente se pretende contar con robots que sean capaces de realizar actividades de forma remota, ésto debido a que podría existir peligro para el operador de encontrarse en el mismo sitio donde se encuentra el robot o porque es imposible acceder al lugar dónde el robot sí es capaz de llegar. Un ejemplo de este tipo de operaciones remotas es el caso de actividades espaciales donde el robot explorador puede realizar tareas en Marte siendo controlado desde la Tierra, eliminando el posible peligro para astronautas al explorar terrenos desconocidos. Así mismo, la practicidad en la utilización de robots de forma remota coadyuva en la realización de múltiples tareas al mismo tiempo. Por éstas y varias razones más, el uso de robots a distancia es vital en casi cualquier área de aplicación. A este tipo de tareas en las cuales el robot realiza una operación de forma remota o a distancia se le conoce como sistemas teleoperados. La teleoperación viene de la etimología griega «*tele* »que significa "lejos" o "a distancia" lo que literalmente significa realizar una operación o actividad a distancia o en un sito lejano.

Un sistema de teleoperación clásico consta de un sitio local y un sitio remoto. En el sitio local se encuentra el operador el cual envía los comandos necesarios al robot para que realice una tarea específica. El operador puede ser una persona que indique directamente desde una computadora o desde una interfaz Hombre-Máquina, o HMI por sus siglas en inglés, la tarea que el robot debe efectuar, o una unidad de procesamiento que contenga las tareas que el robot debe realizar. Por otro lado, en el sitio remoto se encuentra el robot el cual recibe la información que llega del sitio local y ejecuta las instrucciones necesarias para la realización de la tarea encomendada. La teleoperación puede clasificarse en unilateral o bilateral. En la teleoperación unilateral, existe una comunicación en una sola dirección del sitio local al sitio remoto, por otro lado en la teleoperación bilateral existe una comunicación en ambos sentidos, es decir existe tanto envío de información del sitio local al sitio remoto y viceversa. Este tipo de información compartida en ambos sitios puede ser, posiciones, velocidades, aceleraciones, fuerzas, dependiendo de la aplicación.

Una de las características principales en un esquema de teleoperación, es la presencia de retardos. Los retardos en un sistema teleoperado se presentan por diferentes circunstancias, una de ellas y la más evidente es el retardo de comunicación debido a la distancia de separación entre el operador y el robot, es decir entre el sitio local y el sitio remoto. Otro de los factores que provocan la existencia de retardos en los sistemas teleoperados es la baja velocidad de sensado en los sistemas de adquisición del robot, si un sensor detecta con una velocidad menor a la velocidad de procesamiento y envío de datos, éste provocará un retardo en el sistema. Por último, existen retardos debido al tiempo de procesamiento del controlador, ya que estos procesamientos pueden llegar a ser mayores al periodo de muestreo utilizado en el sistema. Los retardos afectan de forma negativa a los sistemas provocando un bajo desempeño del controlador o incluso provocando inestabilidad en el sistema [4]. Es por eso la importancia de estrategias de control que permitan mitigar los efectos de los retardos, y a su vez permitan realizar un adecuado control de los estados del robot.

## 1.2. Antecedentes.

En esta sección se presentan los trabajos relacionados al control de sistemas teleoperados y su aplicación con robots móviles diferenciales, mostrando un panorama general y posicionamiento en el desarrollo de este tema. Los sistemas teleoperados representan un tema constante de interés debido a sus múltiples aplicaciones en la vida diaria del ser humano, como se mencionó en la sección anterior las aplicaciones para este tipo de sistemas son incontables y con aportaciones importantes como reducción de tiempos de trabajo, disminución de riesgos para el operador, reducción de costos, entre muchos otros beneficios. El problema de teleoperación bilateral fue abordado por primera vez a mediados de la década de los 40's por Goertz de acuerdo con [5] con un esquema maestro-esclavo. En 1954 una versión mejorada [6, 7] del esquema anterior es propuesta además de la utilización de una fuerza eléctrica como reflejo de la posición de un servomecanismo es agregada. En la década de los 60 Ferrelll y Sheridan [8], realizan experimentos para obtener un mayor entendimiento de los efectos de los retardos en los sistemas teleoperados. En estos experimentos encontraron que cuando existen retardos de tiempo en el lazo de comunicación el operador adopta una estrategia de control de mover y esperar, en donde se deduce que el tiempo para completar la tarea es lineal respecto a los retardos inducidos en el sistema.

En [4] el problema es atacado utilizando un enfoque de pasividad y empleando teoría de dispersión para el diseño de controladores que garanticen estabilidad haciendo al canal de comunicación libre de retardos. En [9], un esquema de control continuo es propuesto de forma estable para la realización de una teleoperación de un robot móvil. El modelo utilizado en la estructura de control está basado en coordenadas polares con la cual describe el movimiento del robot así mismo el diseño de la estructura de control contempla un compensador de retardo variable tanto en el sitio del operador como en el lugar donde el robot realiza la tarea.En [10] se emplea un algoritmo de filtro de partículas para compensar los efectos negativos de los retardos en el sistema, por otro lado en [11] se utiliza un control basado en un observador para la compensación de retardos. Una solución común, dentro de los múltiples enfoques en los sistemas teleoperados con retardos en la comunicación, es la utilización del predictor de Smith en las estructuras de control. La principal ventaja del predictor de Smith es hacer al sistema de control libre de retardos. En el capítulo 2 se explica más a detalle esta estrategia de control. En [12] se propone una estructura de control basada en modelos internos IMPACT (Internal Model Principle and Control Together) la cual junto con el predictor de Smith logra estabilidad robusta y el rechazo a perturbaciones externas. Esta estrategia de control incluye control por modelo interno y principio de modelo interno como adición al predictor de Smith. En [13], se realiza un trabajo de teleoperación, por medio de la estrategia IMPACT, de un sistema bilateral maestro-esclavo. En [14] se considera un esquema de teleoperación bilateral maestro-esclavo con retardos de comunicación constantes, en esta implementación se hace uso de robots manipuladores.

En este trabajo de tesis, se emplea un estrategia de control IMPACT para compensar los efectos de los retardos en el sistema de teleoperación haciendo uso del control por modelo interno, (o IMC por sus siglas en inglés), este esquema consiste de forma general, en diseñar un controlador a partir de la inversa del modelo de la planta; así como una comparación entre el modelo del sistema y la planta, en el capítulo 2 de este trabajo se explica a detalle esta estrategia. Debido al buen desempeño de esta estrategia de control, en este trabajo de tesis se propone una extensión de estás técnicas de control para su aplicación a un esquema de teleoperación bilateral maestro-esclavo en robots móviles diferenciales.

## 1.3. Planteamiento del problema.

El problema a resolver en este trabajo es teleoperar un robot móvil diferencial a través de un sistema bilateral tipo maestro-esclavo implementando la estrategia de control por modelo interno como parte del esquema de control IMPACT que permita compensar los efectos de los retardos de tiempo presentes en la comunicación. Se presentan los aspectos considerados en la realización de este trabajo:

1. Retardos constantes en canal de comunicación, envío maestro a esclavo y envío esclavo a maestro. El envío de la posición del robot maestro y robot esclavo respectivamente.

2. Problema de seguimiento de trayectoria suave de un robot móvil tipo diferencial.

3. Controlador robusto respecto a perturbaciones aditivas al sistema, bías en sensores, ruido en señales de sensores.

4. Problema de error entre retardos reales y retardos nominales en esquema de control.

## 1.4. Objetivos.

### 1.4.1. Objetivo General.

El objetivo general de este trabajo de investigación es el diseño y la implementación de un control para aplicaciones de teleoperación bilateral, es decir, teniendo una comunicación en ambos sentidos entre el sitio local y el sitio remoto, realizando el robot esclavo, en el sitio remoto, el seguimiento de trayectorias enviadas desde el sitio local por el robot maestro. Así mismo, se busca que el controlador sea robusto respecto a los retardos en el tiempo presentes en la comunicación entre un robot maestro y un robot esclavo así como robustez respecto a perturbaciones externas de tipo aditivo.

### 1.4.2. Objetivos Específicos.

1. Realizar un esquema de control capaz de seguir trayectorias variables en el tiempo.

2. Realizar una estructura de control capaz de soportar retardos y perturbaciones no medibles o desconocidos.

3. Realización de comunicación remota y simultánea en tiempo real de robots móviles diferenciales.

4. Implementación de control propuesto en plataforma experimental y validación de resultados en simulación numérica.

## 1.5. Trabajo desarrollado.

Se presenta a grandes rasgos el proceso cronológico del trabajo realizado en esta tesis con lo cual se pretende dar un panorama general de las actividades y el desarrollo realizado durante este trabajo de investigación. Como primer punto a atacar, fue necesario realizar la investigación de los antecedentes referentes al tema de retardos en teleoperación así como el desarrollo teórico de la estrategia de control empleada. Así mismo se consultó la literatura de sistemas no lineales para el desarrollo de la linealización del modelo cinemático del robot móvil diferencial, con lo cual nos permitió realizar de forma directa el IMC.

Una vez efectuada la parte teórica del trabajo, se verificaron los resultados obtenidos en la teoría con simulaciones hechas en simulink matlab como primera aproximación y después en el lenguaje de programación C++ por medio del programa Visual Studio 2010, el cual también sería utilizado para la parte experimental. Las simulaciones se realizaron, primeramente tomando en cuenta un esquema de teleoperación unilateral operador-esclavo seguido del esquema completo de teleoperación bilateral maestro-esclavo. Por último el trabajo experimental fue llevado a cabo con dos robots Pioneer P3-DX de la marca mobile robots, en una red inalámbrica local y en un ambiente controlado, utilizando odometría para la obtención de las posiciones de los robots.

## 1.6. Aportaciones de la tesis.

La aportación de este trabajo contempla el uso de una estrategia de control como el IMC para tareas de teleoperación bilateral. Anteriormente el IMC se había aplicado en actividades de teleoperación bilateral sólo para robots manipuladores, obteniendo alto desempeño del controlador respecto a retardos y perturbaciones, es el caso de [14], sin embargo, el uso de esta estrategia de control para un robot móvil diferencial no se había considerado anteriormente, debido a que representa un mayor reto de diseño al ser un sistema no lineal con restricciones no holonómicas. Es así que la

contribución hecha por este trabajo implica una nueva forma de atacar el problema de teleoperación con resultados prometedores.

## 1.7. Organización de la tesis.

El presente trabajo se encuentra organizado en 6 capítulos. En el capítulo 2 se presenta el marco teórico, la primer parte de este capítulo muestra el modelado cinemático de los robots móviles diferenciales, seguido de una estrategia de linealización utilizada para modelos no lineales con características específicas, esta linealización permite de forma directa la realización del diseño de la estrategia de control. Por último la obtención de la matriz de transferencia del sistema linealizado del robot.

El en capítulo 3 se muestra el esquema de teleoperación bilateral para robots móviles diferenciales. En la primera sección se presenta la arquitectura del sistema de teleoperación bilateral propuesto en el trabajo, así como la retroalimentación de posición esclavo-maestro presentada en la sección error de posición. El diseño del controlador es presentado en esta sección aplicando estrategias de control robusto como es el caso del IMC. De igual forma se muestra la construcción del la inversa del modelo lineal obtenido en el capítulo anterior y la utilización de un controlador PID para reducir el error planta-modelo presentado en el esquema de control. Así mismo, la realización de un filtro utilizado para perturbaciones externas al sistema, finalmente se muestran resultados en simulación numérica elaborados en el software Matlab 2015.

El el capítulo 4 se presenta la plataforma experimental, los robots móviles diferenciales Pioneer P3-DX de la marca mobile robots utilizados en los experimentos, así como el protocolo de comunicación implementado para el envío de datos. En el capítulo 5 se presentan los resultados experimentales realizados con los robots, las trayectorias propuestas en los experimentos y los tiempos de retardos utilizados para las pruebas. Finalmente en el capítulo 6 se presentan las conclusiones del trabajo de tesis y trabajo a futuro.

# Capítulo 2

# Marco Teórico

## 2.1. Robot móvil

Un robot móvil, como se mencionó en el capítulo anterior, se define como un sistema electromecánico capaz de desplazarse de un punto a otro en un determinado espacio de trabajo. Es posible clasificar a los robots móviles de acuerdo al tipo de mecanismo de locomoción que cada robot emplea para desplazarse en un ambiente de trabajo. Es así que existen robots móviles de patas, orugas o ruedas [15]. En la mayoría de los casos el robot móvil de ruedas representa una alternativa más atractiva con respecto a los otros tipos de locomoción debido a que el control de las ruedas es menos complejo, dado el hecho que el diseño de este tipo de mecanismos es más sencillo al no requerir de un gran número de partes, lo que lo hace un robot de bajo costo además que el consumo de energía es bajo debido a que las ruedas son ideales para desplazarse en superficies firmes y lisas. Por estas razones el robot móvil de ruedas representa una opción viable para varios trabajos de investigación así como de aplicación.

### 2.1.1. Tipos de ruedas

Existen diferentes tipos de ruedas empleadas en los robots móviles. En general es posible clasificar las ruedas en cinco tipos, las ruedas fijas, ruedas céntricas orientables, ruedas excéntricas orientables, ruedas suecas y ruedas esféricas. En la figura 2.1 se muestran las ruedas empleadas en los robots móviles. En todos los casos se considera que existe un único punto de contacto entre la rueda y el piso.
Las ruedas fijas dan al robot un movimiento hacia adelante y hacia atrás, estas ruedas cuentan con un sólo grado de libertad, solo pueden girar sobre su eje horizontal. La orientación de estas ruedas está fija con respecto al chasis del robot, es decir no pueden girar sobre su eje vertical. En la figura 2.2 (a) se muestra el esquema de una rueda fija o también llamada rueda estándar fija, el punto $A$ es el punto central de la rueda, la posición del punto $A$ se determina usando coordenadas polares, donde $l$ representa la distancia del punto $P$ del robot al punto $A$ de la rueda con un ángulo $\alpha$. La orientación del plano de la rueda con respecto al vector $\vec{PA}$ se determina por el ángulo fijo $\beta$ y $\varphi$ representa el ángulo de rotación de la rueda respecto a su eje

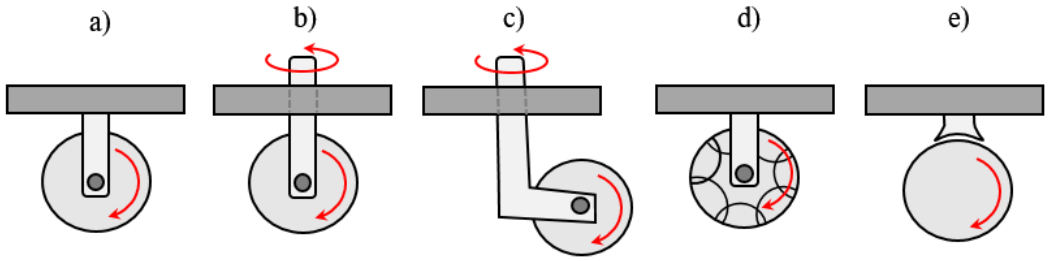
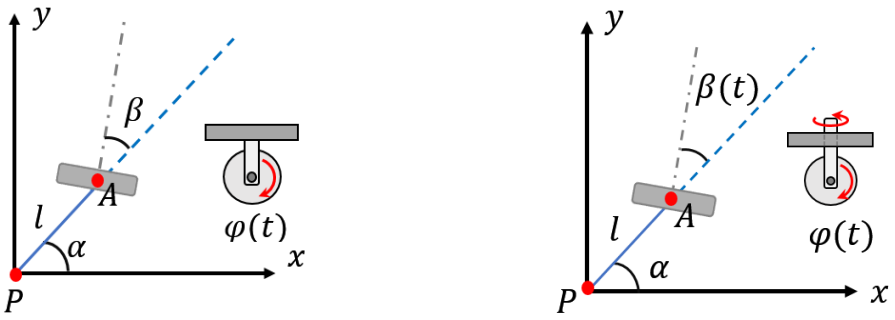



Figura 2.1: Tipos de ruedas: a)Rueda fija b)Rueda céntrica orientable c)Rueda excéntrica orientable (Castor) d)Rueda sueca e)Rueda esférica

horizontal. En el caso de las ruedas céntricas orientables 2.2 (b), éstas difieren de las ruedas fijas por la adición de un grado de libertad sobre el eje vertical de la ruedas, es decir el ángulo $\beta$ es variable en el tiempo permitiendo modificar la dirección de la rueda con respecto al chasis del robot. Ambas ruedas cuentan con restricciones, donde la velocidad en el punto de contacto entre la rueda y el piso debe ser igual a cero, esto implica que la componente de velocidad ortogonal y paralela al plano de la llanta deben ser iguales a cero.



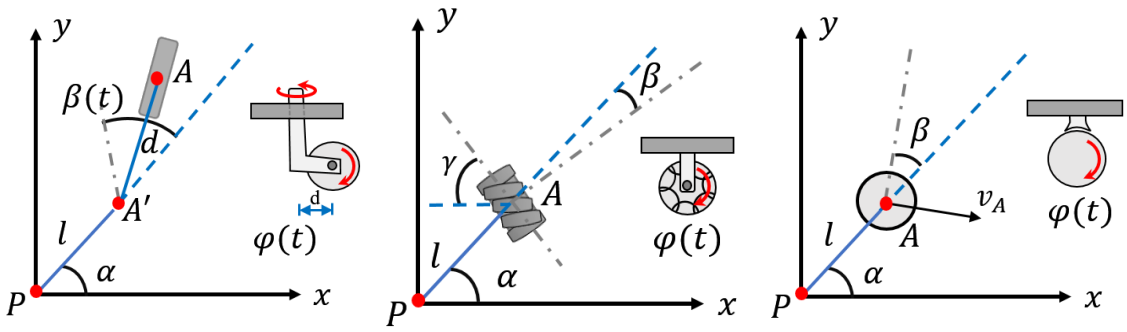

(a) Rueda fija

(b) Rueda céntrica orientable

Figura 2.2: Esquema de ruedas convencionales

Las ruedas excéntricas orientables son capaces de cambiar su dirección con respecto a un eje vertical sin embargo a diferencia de las ruedas céntricas orientables, como su nombre lo indica, el eje vertical de giro no pasa por el punto de contacto entre la rueda y el piso, como se puede apreciar en la figura 2.3 (a), el punto medio de la rueda denotado por $A$ se encuentra separado del eje vertical $A'$ una distancia $d$. Estas ruedas también son llamadas ruedas tipo castor. La diferencia entre una rueda céntrica y excéntrica radica en que la acción de direccionamiento de una rueda céntrica no causa por si misma un movimiento en el chasis del robot sin embargo en una rueda excéntrica esto si sucede debido a la distancia $d$ entre el punto de contacto del piso y el eje vertical de rotación. Las ruedas castor se consideran omnidireccionales debido a que dado cualquier movimiento en el chasis del robot, existe un valor en la velocidad de rotación $\dot{\varphi}$ y en la velocidad de direccionamiento $\dot{\beta}$ en las cuales las restricciones se cumplen. Las ruedas suecas, figura 2.3 (b), se consideran ruedas omnidireccionales

a pesar de que no existe un eje vertical definido, esto debido a que es agregado un grado de libertad extra a una rueda fija, el grado de libertad extra se obtiene gracias a los rodamientos con los cuales estas ruedas cuentan. Los rodamientos se encuentran sujetos alrededor del perímetro de la rueda orientados un ángulo $\gamma$ con respecto al eje principal de la rueda. Esto permite que la rueda sueca pueda moverse en cualquier dirección de forma omnidireccional. La rueda esférica, figura 2.3 (c), es otra rueda omnidireccional que no cuenta con ejes principales de rotación, por lo cual no existen restricciones propias de rodadura o deslizamiento, al igual que las ruedas suecas y castor está rueda puede realizar sus movimientos de forma omnidireccional.




(a) Castor (b) Sueca (c) Esférica

Figura 2.3: Ruedas omnidireccionales

Los diferentes tipos de ruedas mostrados anteriormente cuentan, como se ha mencionado, con restricciones de rodadura y deslizamiento las cuales pueden expresarse de forma matemática, en [15], se muestran estas ecuaciones para cada una de las ruedas mencionadas en este apartado.

### 2.1.2. Restricciones de movilidad

Un robot móvil cuenta con restricciones en su movimiento dependiendo del tipo de llantas empleadas en su diseño, por tal motivo es importante definir dos tipos de grados de libertad que nos ayudarán a clasificar a los robots móviles. Los grados de movilidad $S_m$ y los grados de direccionabilidad $S_s$. Como se había visto anteriormente las ruedas fijas y las ruedas céntricas orientables añaden al robot móvil restricciones en su movimiento, a diferencia de las ruedas suecas, castor y esféricas que no restringen el movimiento del robot. Un robot móvil cuenta con 3 grados de libertad en su movimiento sobre un plano, por lo cual al tener ruedas fijas y/o céntricas orientables independientes restarán movilidad al robot. Por lo general las ruedas fijas cuando son más de una se encuentran sobre el mismo eje de rotación, a estas ruedas se les consideran dependientes pues únicamente restringen un movimiento del robot, de igual forma las ruedas céntricas orientables que siempre tengan la misma dirección se considerarán dependientes y restringirán un movimiento del robot. Los grados de movilidad se pueden definir como $S_m = 3 - numero\ de\ ruedas\ independientes\ (fijas\ y\ centricas\ orientables)$. Los

grados de movilidad se refieren al número de grados de libertad con los que el robot cuenta con la configuración dada en un instante de tiempo sin cambiar la dirección de sus ruedas [15], es decir los grados de libertad que el robot tendrá únicamente al manipular las velocidades de sus ruedas. Es así que una bicicleta cuenta con $S_m = 1$ debido a que cuenta con una rueda fija y una céntrica orientable y un robot con todas sus ruedas tipo castor tendrá $S_m = 3$ al no contar con ruedas que restrinjan su movimiento.

Los grados de direccionabilidad $S_s$ se pueden definir como el número de ruedas céntricas orientables que puedan ser direccionadas independientemente con el objetivo de darle orientación al robot. Un robot con 3 ruedas tipo castor tendrá $S_s = 0$, sin embargo al agregar una rueda céntrica orientable y dejar 2 ruedas tipo castor su grado de direccionabilidad cambiará a $S_s = 1$. Los grados de maniobrabilidad $S_M$ se definen como la suma de los grados de movilidad $S_m$ y direccionabilidad $S_s$, $S_M = S_m + S_s$. $S_M$ se define como todos lo grados de libertad de un robot móvil que pueden ser manipulados. De acuerdo a los grados de libertad descritos anteriormente se puede clasificar cinco tipos de robots con una configuración que consta de 3 ruedas. En la figura 2.4 se muestran los 5 tipos de robots móviles. Los robots (3,0) u omnidireccionales, los robots (2,0) o tipo uniciclo o diferencial, los robots (2,1), los robot (1,1), carro convencional de tracción delantera o trasera, bicicleta o triciclo, y los robot (1,2).

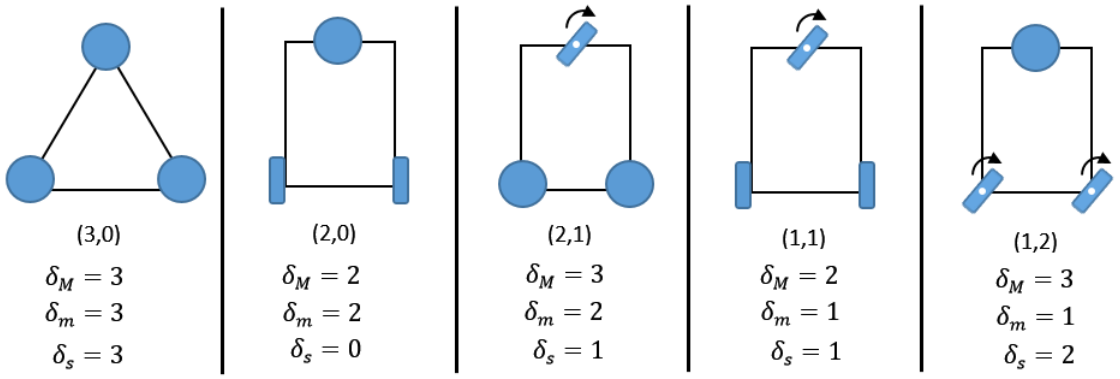



Figura 2.4: Configuración de robots móviles

Como se aprecia en la figura 2.4 existen robots con el mismo grado de maniobrabilidad sin embargo como se puede apreciar la configuración con respecto a las llantas empleadas es diferente, tal es el caso del robot (2,0) y (1,1) y los robots (3,0), (2,1) y (1,2). En este trabajo de investigación se trabajará con robots tipo (2,0) o también llamados robots diferenciales los cuales cuentan con 2 llantas fijas y una llanta omnidireccional.

## 2.2. Modelo cinemático del robot móvil diferencial.

La localización de un robot móvil en un marco de referencia inercial (global) $\{X, Y\}$, está descrita por las coordenadas $x$ y $y$. Estas coordenadas designan la posición del punto $p$ localizado a la mitad del eje de las ruedas del robot.
El marco de referencia del robot $\{x_r, y_r\}$ es usado para determinar la orientación del robot respecto del marco de referencia inercial $\{X, Y\}$, por medio de la variable $\theta$, ver [15]. Es decir, $\theta$ representa la rotación, alrededor eje $Z$ en el punto $p$, del marco de referencia del robot con respecto al marco de referencia inercial. En la figura 2.5 se muestra la posición de un robot móvil diferencial.

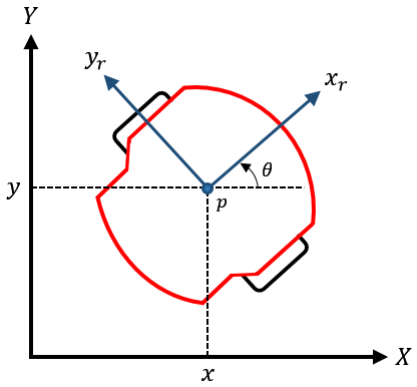



Figura 2.5: Posicionamiento del robot móvil diferencial.

En la figura 2.6 se presentan las velocidades del robot, donde $v$ representa la velocidad traslacional en dirección perpendicular al eje de las ruedas y $w$ la velocidad rotacional sobre el eje $Z$ en el punto $p$. Es posible observar también, que la orientación de la velocidad traslacional $v$ está dada por la variable $\theta$ y las componentes horizontal y vertical son $\dot{x}$ y $\dot{y}$ respectivamente.

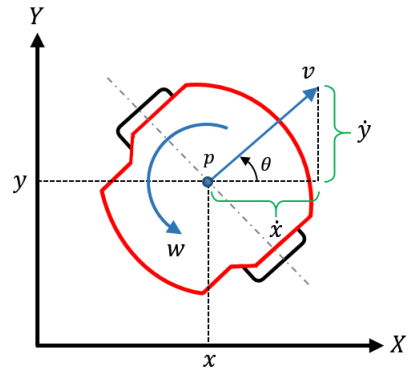



Figura 2.6: Velocidades del robot móvil diferencial.

El modelo cinemático del robot móvil está dado por las siguientes ecuaciones,

$$\dot{x} = v\cos\theta \tag{2.1}$$

$$\dot{y} = v\sin\theta \tag{2.2}$$

$$\dot{\theta} = w \tag{2.3}$$

Se observa en las ecuaciones del modelo 2.1, 2.2 y 2.3 que las entradas de control para el sistema del robot diferencial móvil son las velocidades $v$ y $w$. Así mismo se aprecia que el sistema es subactuado, es decir se cuenta con 2 entradas de control $(v, w)$ y 3 estados $(x, y, \theta)$. Por último se observa que el modelo del robot diferencial es un modelo no lineal. En el modelo clásico cinemático del robot diferencial mostrado anteriormente las velocidades traslacional y rotacional ($v$ y $w$ respectivamente) representan las entradas de control, sin embargo en la práctica en la mayoría de los casos el uso de estas variables de control complica la implementación en una plataforma experimental. Por lo cual, frecuentemente se utilizan como variables de control las velocidades rotacionales en cada una de las ruedas, considerando la conexión directa de actuadores en los ejes de las ruedas que facilita el control de giro en cada una de las ruedas. $w_i$ y $w_d$ representan las velocidades rotacionales en la rueda izquierda y rueda derecha respectivamente. En la figura 2.7 se presentan el esquema de relación de velocidades [16], es decir una relación entre las velocidades de cuerpo rígido del robot $v$ y $w$ sobre el punto $p$ y las velocidades en las ruedas $w_i$ y $w_d$. La distancia entre los centros de cada rueda $A_i$ y $A_d$ es $2l$, el radio de las ruedas es $r$, así como las velocidades lineales del centro de cada rueda $v_{A_d}$ y $v_{A_i}$ derecha e izquierda respectivamente. Por último los puntos de contacto de cada rueda sobre el piso son $o_i$ y $o_d$

Nótese que las velocidades lineales de cada rueda $v_{A_d}$ y $v_{A_i}$ derecha e izquierda respectivamente, se encuentran dadas por las ecuaciones 2.4 y 2.5, donde $\vec{v}_{o_d}$ y $\vec{v}_{o_i}$ representan las velocidades en los puntos de contacto de cada rueda

$$\vec{v}_{A_d} = \vec{v}_{o_d} + \vec{w}_d \times \vec{r} \tag{2.4}$$

$$\vec{v}_{A_i} = \vec{v}_{o_i} + \vec{w}_i \times \vec{r} \tag{2.5}$$

Debido a que no existe deslizamiento en las ruedas las velocidades en los puntos de contacto son cero, es decir $\vec{v}_{o_d} = 0$ y $\vec{v}_{o_i} = 0$

$$\vec{v}_{A_d} = \left(-w_d\hat{k}\right) \times \left(r\hat{j}\right) = w_d r\hat{i} \tag{2.6}$$

$$\vec{v}_{A_i} = \left(-w_i\hat{k}\right) \times \left(r\hat{j}\right) = w_i r\hat{i} \tag{2.7}$$

Las velocidades lineales de cada rueda también están relacionadas con las velocidades $v$ y $w$ del robot como se muestra en las ecuaciones 2.8 y 2.9

$$\vec{v}_{A_d} = \left(v\hat{i}\right) + \left(w\hat{j}\right) \times \left(l\hat{k}\right) = (v + lw)\,\hat{i} \tag{2.8}$$

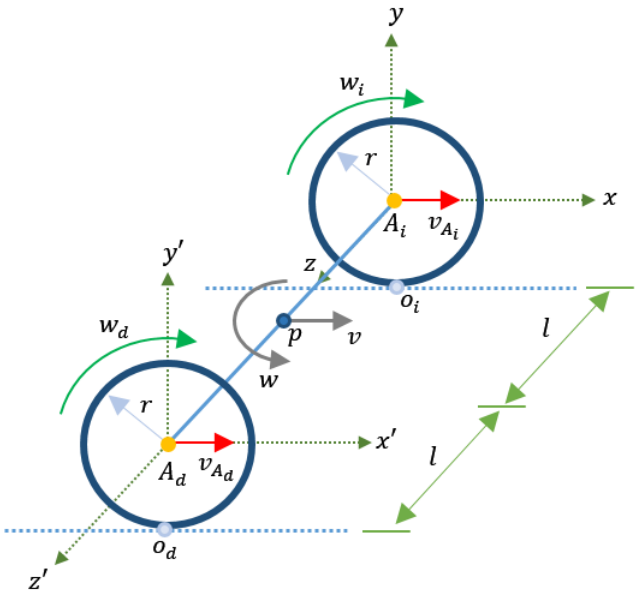



Figura 2.7: Relación de velocidades

$$\vec{v}_{A_i} = \left(v\hat{i}\right) + \left(w\hat{j}\right) \times \left(-l\hat{k}\right) = (v - lw)\,\hat{i} \tag{2.9}$$

Con las ecuaciones 2.6 a la 2.9 es posible obtener las ecuaciones 2.10 y 2.11, las cuales muestran la relación entre las velocidades en las ruedas ($w_i$ y $w_d$) y del cuerpo rígido del robot ($v$ y $w$).

$$v = \frac{r}{2}\left(w_d + w_i\right) \tag{2.10}$$

$$w = \frac{r}{2l}\left(w_d - w_i\right) \tag{2.11}$$

De forma matricial el sistema quedaría de la siguiente forma

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \begin{bmatrix} w_d \\ w_i \end{bmatrix} \tag{2.12}$$

Donde

$$\begin{bmatrix} w_d \\ w_i \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix}^{-1} \begin{bmatrix} v \\ w \end{bmatrix} \tag{2.13}$$

Finalmente obtenemos las variables de control nuevas $w_i$ y $w_d$ en función de las velocidades $v$ y $w$.

$$\begin{bmatrix} w_d \\ w_i \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & l \\ 1 & -l \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{2.14}$$

Se puede apreciar que la matriz de transformación es no singular si y solo sí $r > 0$ y $l > 0$, lo que implica que es posible obtener cualquier valor en las velocidades del robot $v$ y $w$ por medio de las velocidades de rotación de cada rueda.

### 2.2.1. Restricciones del modelo cinemático

La utilización del modelo cinemático del robot diferencial facilita su control, ya que no se contempla información adicional a las ecuaciones de movimiento del robot. Sin embargo, al no contemplar esta información como dinámicas inherentes al modelo, es posible que el controlador no desempeñe un buen funcionamiento. Por lo tanto, es importante realizar tareas de seguimiento de trayectorias a baja velocidad que no existen estas dinámicas no modeladas, en este trabajo todo el desarrollo se hace con la suposición que el robot realizará trayectorias a bajas velocidades. Además, el sistema per se, cuenta con restricciones no holónomas. Esta restricción implica que no es posible realizar cualquier trayectoria para llevar al robot (2,0) de un punto a otro debido a que las ruedas fijas de este tipo de robots no pueden girar en cualquier dirección, es decir no se pueden desplazar de forma perpendicular al sentido de giro de las ruedas. Tomando el modelo del sistema del robot (2,0), tenemos que

$$\dot{x}_1 = u_1 cos\theta \Rightarrow u_1 = \frac{\dot{x}_1}{cos\theta}$$

$$\dot{x}_2 = u_1 sin\theta \Rightarrow u_1 = \frac{\dot{x}_2}{sin\theta}$$

$$\dot{x}_3 = \theta$$

Lo que implica que

$$\frac{\dot{x}_1}{cos\theta} = \frac{\dot{x}_2}{sin\theta}$$

$$\Rightarrow \dot{x}_1 sin\theta - \dot{x}_2 cos\theta = 0 \tag{2.15}$$

En la ecuación 2.15 se muestra la restricción no holónoma del robot diferencial móvil.

## 2.3. Linealización del modelo.

Dado que el modelo del robot móvil diferencial mostrado en las ecuaciones 2.1, 2.2 y 2.3 es de carácter no lineal, se realiza una linealización que permita de forma directa la obtención de la matriz función de transferencia del sistema utilizada en el diseño del controlador. Se propone una linealización por retroalimentación entrada salida del sistema representado en espacio de estados descrito a continuación:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w \tag{2.16}$$

Donde $x$, $y$ y $\theta$ son los estados del sistema, y las variables de control están representadas por las velocidades $v$ y $w$. Sea el sistema no lineal de la forma:

$$\dot{x} = f(x) + g_1(x)\, u_1 + g_2(x)\, u_2 \tag{2.17}$$

$$y_1 = h_1(x) \tag{2.18}$$

$$y_2 = h_2(x) \tag{2.19}$$

Donde:

$$f(x) = 0,\ g_1(x) = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix},\ g_2(x) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$u_1 = v,\ u_2 = w$$

$$h_1(x) = x$$

$$h_2(x) = y$$

Para linealizar el sistema MIMO, es necesario que el vector de grado relativo se encuentre bien definido [17], es decir la matriz de desacoplamiento $A(x)$ debe ser no singular. El sistema cuenta con 2 entradas y 2 salidas, por lo que la matriz de desacoplamiento se expresa de la siguiente manera:

$$A(x) = \begin{bmatrix} L_{g_1}h_1 & L_{g_2}h_1 \\ L_{g_1}h_2 & L_{g_2}h_2 \end{bmatrix}$$

Calculando las derivadas de Lie

$$L_{g_1}h_1 = \frac{\partial h_1}{\partial x} g_1(x) = \cos\theta$$

$$L_{g_2}h_1 = \frac{\partial h_1}{\partial x} g_2(x) = 0$$

$$L_{g_1}h_2 = \frac{\partial h_2}{\partial x} g_1(x) = \sin\theta$$

$$L_{g_2}h_1 = \frac{\partial h_1}{\partial x} g_2(x) = 0$$

Se obtiene la siguiente matriz de desacoplamiento

$$A(x) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \tag{2.20}$$

Se aprecia que la matriz de desacoplamiento en 2.20 es singular, lo que implica que el vector de grado relativo para el sistema MIMO no lineal, no está definido. Para resolver este problema se plantea una extensión dinámica del sistema [18]. Se propone

un nuevo estado $\xi = u_1 = v$, así mismo se agrega una nueva entrada de control $\breve{u}_1 = \dot{\xi}$, obteniendo el sistema en 2.21

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} \xi \cos\theta \\ \xi \sin\theta \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \breve{u}_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_2 \tag{2.21}$$

Donde

$$f(x) = \begin{bmatrix} \xi \cos\theta \\ \xi \sin\theta \\ 0 \\ 0 \end{bmatrix}, \; g_1(x) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \; g_2(x) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\breve{u}_1 = \dot{\xi}, \; u_2 = w$$

$$h_1(x) = x$$

$$h_2(x) = y$$

Calculo de la matriz de desacoplamiento

$$A(x) = \begin{bmatrix} L_{g_1} L_f h_1 & L_{g_2} L_f h_1 \\ L_{g_1} L_f h_2 & L_{g_2} L_f h_2 \end{bmatrix}$$

Calculando las derivadas de Lie

$$L_f h_1 = \frac{\partial h_1}{\partial x} f(x) = \xi \cos\theta$$

$$L_{g_1} L_f h_1 = \frac{\partial (L_f h_1)}{\partial x} g_1(x) = \cos\theta$$

$$L_{g_2} L_f h_1 = \frac{\partial (L_f h_1)}{\partial x} g_2(x) = -\xi \sin\theta$$

$$L_f h_2 = \frac{\partial h_2}{\partial x} f(x) = \xi \sin\theta$$

$$L_{g_1} L_f h_2 = \frac{\partial (L_f h_2)}{\partial x} g_1(x) = \sin\theta$$

$$L_{g_2} L_f h_2 = \frac{\partial (L_f h_2)}{\partial x} g_2(x) = \xi \cos\theta$$

Se obtiene la matriz de desacoplamiento

$$A(x) = \begin{bmatrix} \cos\theta & -\xi \sin\theta \\ \sin\theta & \xi \cos\theta \end{bmatrix}$$

El determinante de $A(x)$

$$det(A(x)) = \xi \cos^2\theta + \xi \sin^2\theta = \xi$$

implica que para que la matriz sea no singular el estado $\xi$ debe ser diferente de 0, es decir la matriz $A(x)$ pierde rango si el robot no se mueve en ninguna dirección o se encuentra simplemente rotando sobre su propio eje. Se realiza un cambio de variables para el sistema mostrado en la ecuación 2.21 con el objetivo de obtener un sistema lineal entrada-salida. Donde

$$z_1 = x \tag{2.22}$$

$$z_2 = y \tag{2.23}$$

$$z_3 = \xi cos\theta \tag{2.24}$$

$$z_4 = \xi sen\theta \tag{2.25}$$

Al derivar los estados nuevos obtenemos lo siguiente

$$\dot{z}_1 = \dot{x} = \xi cos\theta = z_3$$

$$\dot{z}_2 = \dot{y} = \xi sen\theta = z_4$$

$$\dot{z}_3 = \dot{\xi} cos\theta - \xi sin\theta\dot{\theta} = cos\theta\check{u}_1 - z_4 u_2$$

$$\dot{z}_4 = \dot{\xi} sen\theta + \xi cos\theta\dot{\theta} = sen\theta\check{u}_1 + z_3 u_2$$

Sabemos que

$$\xi^2 = \xi^2 \cos^2\theta + \xi^2 \sin^2\theta$$

$$\Rightarrow \xi = \sqrt{(\xi\cos\theta)^2 + (\xi\sin\theta)^2}$$

$$= \sqrt{z_3^2 + z_4^2}$$

Redefiniendo

$$\cos\theta = \frac{\xi\cos\theta}{\xi} = \frac{z_3}{\sqrt{z_3^2 + z_4^2}}$$

$$\sin\theta = \frac{\xi\sin\theta}{\xi} = \frac{z_4}{\sqrt{z_3^2 + z_4^2}}$$

El sistema con el cambio de variables queda de la siguiente forma

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{z_3}{\sqrt{z_3^2+z_4^2}} \\ \frac{z_4}{\sqrt{z_3^2+z_4^2}} \end{bmatrix} \check{u}_1 + \begin{bmatrix} 0 \\ 0 \\ -z_4 \\ z_3 \end{bmatrix} u_2 \tag{2.26}$$

$$\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

Derivamos las salidas hasta encontrar una de las entradas

$$\dot{h}_1 = \dot{z}_1 = z_3$$

$$\ddot{h}_1 = \dot{z}_3 = \frac{z_3}{\sqrt{z_3^2 + z_4^2}}\check{u}_1 - z_4 u_2$$

$$\dot{h}_2 = \dot{z}_2 = z_4$$

$$\ddot{h}_2 = \dot{z}_4 = \frac{z_4}{\sqrt{z_3^2 + z_4^2}}\check{u}_1 + z_3 u_2$$

$$\begin{bmatrix}\ddot{h}_1 \\ \ddot{h}_2\end{bmatrix} = \begin{bmatrix}\frac{z_3}{\sqrt{z_3^2+z_4^2}} & -z_4 \\ \frac{z_4}{\sqrt{z_3^2+z_4^2}} & z_3\end{bmatrix}\begin{bmatrix}\check{u}_1 \\ u_2\end{bmatrix}$$

$$\Rightarrow \begin{bmatrix}\check{u}_1 \\ u_2\end{bmatrix} = \begin{bmatrix}\frac{z_3}{\sqrt{z_3^2+z_4^2}} & -z_4 \\ \frac{z_4}{\sqrt{z_3^2+z_4^2}} & z_3\end{bmatrix}^{-1}\begin{bmatrix}\ddot{h}_1 \\ \ddot{h}_2\end{bmatrix}$$

$$\begin{bmatrix}\vartheta_1 \\ \vartheta_2\end{bmatrix} = \begin{bmatrix}\ddot{h}_1 \\ \ddot{h}_2\end{bmatrix} \Rightarrow \begin{bmatrix}\check{u}_1 \\ u_2\end{bmatrix} = \begin{bmatrix}\frac{z_3}{\sqrt{z_3^2+z_4^2}} & \frac{z_4}{\sqrt{z_3^2+z_4^2}} \\ -\frac{z_4}{z_3^2+z_4^2} & \frac{z_3}{z_3^2+z_4^2}\end{bmatrix}\begin{bmatrix}\vartheta_1 \\ \vartheta_2\end{bmatrix}$$

$$\check{u}_1 = \frac{z_3}{\sqrt{z_3^2 + z_4^2}}\vartheta_1 + \frac{z_4}{\sqrt{z_3^2 + z_4^2}}\vartheta_2 \tag{2.27}$$

$$u_2 = \frac{z_3}{z_3^2 + z_4^2}\vartheta_2 - \frac{z_4}{z_3^2 + z_4^2}\vartheta_1 \tag{2.28}$$

Con las variables de control en las ecuaciones 2.27 y 2.28 es posible obtener el sistema linealizado mostrado en 2.29 en función de las nuevas variables de control $\vartheta_1$ y $\vartheta_2$

$$\begin{bmatrix}\dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}z_1 \\ z_2 \\ z_3 \\ z_4\end{bmatrix} + \begin{bmatrix}0 \\ 0 \\ 1 \\ 0\end{bmatrix}\vartheta_1 + \begin{bmatrix}0 \\ 0 \\ 0 \\ 1\end{bmatrix}\vartheta_2 \tag{2.29}$$

En la figura 2.8 se muestra el sistema linealizado en coordenadas z, donde $\varphi_1$ y $\varphi_2$ son funciones representadas por las ecuaciones 2.27 y 2.28 respectivamente. Así mismo las funciones $\varphi_3$ y $\varphi_4$ son funciones representadas por las ecuaciones 2.24 y 2.25. Como se puede observar en las entradas de control 2.27 y 2.28 existe singularidades cuando $\xi = 0$ por lo tanto es importante evitar que el robot permanezca sin velocidad traslacional. El sistema mostrado en la ecuación 2.29 será utilizado para obtención de la función de transferencia en el apartado siguiente.

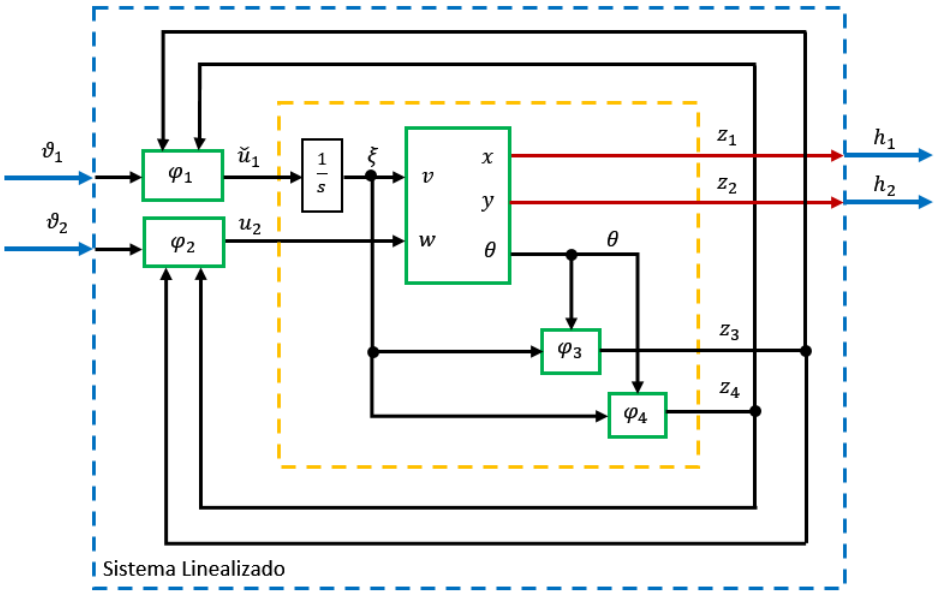



Figura 2.8: Esquema de linealización robot móvil

## 2.4. Matriz de transferencia.

El sistema linealizado en la ecuación 2.29 es un sistema con múltiples entradas y múltiples salidas es decir un sistema MIMO. Específicamente el sistema cuenta con 2 entradas de control $(\vartheta_1, \vartheta_2)$ y 2 salidas $(h_1, h_2)$. Se busca encontrar la función de transferencia del sistema para después trabajar con ella en el diseño del controlador por modelo interno propuesto en el capítulo 3.

Del sistema 2.29 obtenemos,

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Las matrices $A$, $B$ y $C$ se utilizarán para realizar el cálculo de la matriz de transferencia $H(s)$, ver [19]

$$Y(s) = \left[C(sI - A)^{-1} B + D\right] U(s)$$

$$\Rightarrow H(s) = C(sI - A)^{-1} B + D$$

Donde $H(s) = \frac{Y(s)}{U(s)}$

$$H(s) = \begin{bmatrix} \frac{1}{s^2} & 0 \\ 0 & \frac{1}{s^2} \end{bmatrix} = \begin{bmatrix} \hat{G}(s) & 0 \\ 0 & \hat{G}(s) \end{bmatrix} \tag{2.30}$$

Donde $\hat{G}(s)$ representa la función de transferencia del modelo linealizado del robot diferencial por cada canal de salida. Note que la matriz de transferencia 2.30 implica un sistema desacoplado, donde las funciones de transferencia de cada salida con su entrada, corresponden a dos integradores. Por esta razón y por simplicidad, y sin pérdida de generalidad, en lo subsecuente se presenta la metodología de control propuesta considerando un sistema de una entrada una salida, (o SISO por sus siglas en inglés), con función de transferencia.

$$\hat{G}(s) = \frac{1}{s^2}$$

## 2.5. Control por modelo interno.

En este trabajo de tesis se propone la utilización de la estrategia de control por modelo interno (IMC) para la compensación de los efectos de los retardos presentados en el esquemea de teleoperación bilateral explicado en el capítulo anterior, por lo cual en este apartado se pretede explicar de forma general en que consiste el IMC y su características principales, facilitando el entendimiento del diseño de este controlador realizado en el capítulo 3. El control por modelo interno o IMC por sus siglas en inglés, es un esquema de control robusto presentado por primera vez en 1989 por Manfred Morari [20]. El IMC contempla el diseño de un controlador por medio de la utilización del modelo de la planta, este esquema permite estimar perturbaciones tipo aditivas cuando la diferencia entre la salida de la planta y la salida de su modelo es casi cero, además que la obtención del controlador puede ser más directa y sencilla que los controladores convencionales. En la figura 2.9 se muestra el esquema clásico del IMC. En este esquema se presenta una retroalimentación negativa $\hat{D}(s)$ la cuál representa la diferencia entre la salida $Y(s)$ de la planta $G(s)$ y la salida $\hat{Y}(s)$ del modelo $\hat{G}(s)$. Esta diferencia puede ser considerada como una estimación de la perturbación $D(s)$ siempre que $G(s) = \hat{G}(s)$, es decir si no existen diferencias entre la planta y el modelo. Por otro lado, $\hat{R}(s)$ representa la diferencia entre la referencia $R(s)$ y la perturbación estimada $\hat{D}(s)$, y $C(s)$ representa el controlador del sistema con salida $U(s)$ como señal de control.
En las ecuaciones 2.31, 2.32, 2.33, 2.34 y 2.35 se muestran las relaciones entre los elementos del esquema IMC,

$$Y(s) = G(s)\,U(s) + D(s) \tag{2.31}$$

$$\hat{Y}(s) = \hat{G}(s)\,U(s) \tag{2.32}$$

$$\hat{D}(s) = Y(s) - \hat{Y}(s) \tag{2.33}$$

$$\hat{R}(s) = R(s) - \hat{D}(s) \tag{2.34}$$

$$Y(s) = G(s)\,C(s)\,\hat{R}(s) + D(s) \tag{2.35}$$

En el caso cuando $\hat{D}(s) = 0$ , es decir $G(s) = \hat{G}(s)$ y la perturbación externa en el sistema $D(s) = 0$, es posible seguir la referencia $R(s)$ escogiendo el controlador

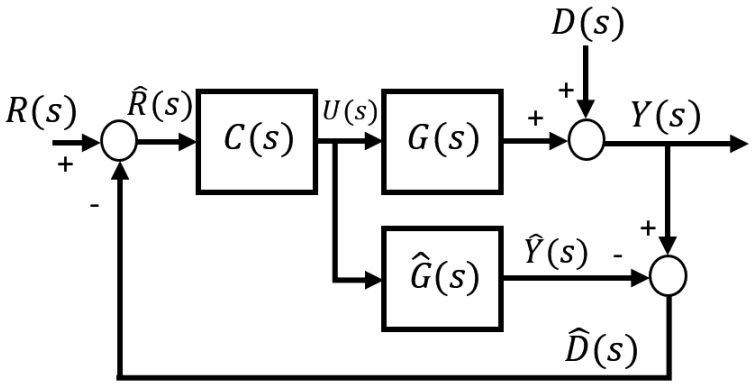



Figura 2.9: Esquema de control IMC

como la inversa del modelo, mostrado en las ecuaciones 2.36 y 2.37. Es posible observar también que este esquema de control actúa en la compensación de errores de modelado y de perturbaciones tipo aditivas al sistema.

$$Y(s) = G(s)\,C(s)\,R(s) = G(s)\,\hat{G}(s)^{-1}\,R(s) \tag{2.36}$$

$$Y(s) = R(s) \tag{2.37}$$

Para diseñar el controlador $C(s)$, como se mencionó anteriormente es necesario invertir el modelo de de la planta $C(s) = \hat{G}(s)^{-1}$ sin embargo, en la mayoría de los casos al invertir el modelo el controlador, la función $C(s)$ se convierte en una función no realizable o impropia, esto significa que el grado del numerador es mayor al grado del denominador. Sea

$$\hat{G}(s) = \frac{1}{s+1}$$

$$\Rightarrow C(s) = s + 1$$

En el ejemplo anterior se muestra que al invertir el modelo el controlador es impropio, por lo cual para resolver este problema se utiliza un filtro $f(s)$ mostrado en la ecuación 2.38 que convierta al controlador en una función propia. Donde $\lambda$ es un parámetro de ajuste que determina la velocidad de respuesta, al ser $\lambda$ pequeño las respuestas serán más rápidas y si $\lambda$ es grande las respuestas serán más lentas.

$$f(s) = \frac{1}{\lambda s + 1} \tag{2.38}$$

La ecuación 2.38 muestra un filtro de primer orden el cual ayuda a solucionar el problema del ejemplo dado

$$C(s) = \hat{G}(s)\,f(s) = \frac{s+1}{\lambda s + 1}$$

Sin embargo para un modelo con un denominador de orden mayor a 1, es necesario utilizar un filtro de orden superior. Se propone en la ecuación 2.39 un filtro de orden

superior, donde $n >$ orden del denominador de $\hat{G}(s)$

$$f(s) = \frac{n\lambda s + 1}{(\lambda s + 1)^n} \tag{2.39}$$

Con los filtros propuestos en las ecuaciones 2.38 y 2.39 es posible obtener un controlador IMC de un modelo con polos y ceros no positivos.

## 2.6. Predictor de Smith.

De igual forma que en la sección anterior, en esta sección se pretende dar una explicación y un panorama general de lo que consiste el predictor de Smith, el cual será utilizado en el esquema de teleoperación bilateral propuesto en el capítulo siguiente. El predictor de Smith se utiliza en esquemas de control; es un predictor de variables del sistema que fue desarrollado en los años 50's, el cuál sigue siendo ampliamente utilizado como solución a los problemas de sistemas con retardos. Es una estrategia mediante la cual los estados de un sistema con retardo pueden ser predichos para un cierto instante de tiempo en el futuro limitado por la magnitud del retardo, por medio de la utilización del modelo de la planta o proceso. Este predictor es utilizado para ayudar en la compensación de los efectos de los retardos en el sistema teleoperado propuesto en esta tesis, puesto que como se explicó en el capítulo 1 en los sistemas teleoperados existen retardos que afectan el desempeño del sistema incluso provocando inestabilidad, en este trabajo de tesis se consideran estos retardos presentes en el canal de comunicación entre el sitio remoto y el sitio local.

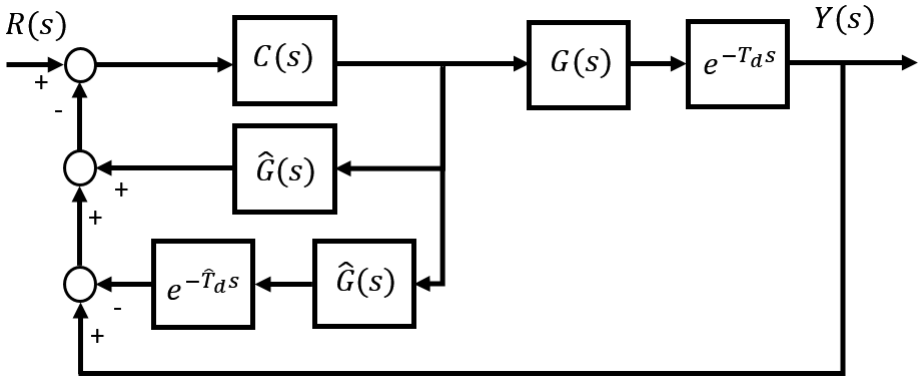



Figura 2.10: Predictor de Smith

En la figura 2.10 se muestra el predictor de Smith, donde $G(s)$ representa la planta, $C(s)$ es el controlador del esquema, el cual puede ser un controlador $PID$, $\hat{G}(s)$ representa el modelo de la planta, $T_d$ es el retardo de tiempo constante en el sistema y $\hat{T}_d$ es el retardo nominal de tiempo, $R(s)$ y $Y(s)$ representan la entrada y la salida del sistema respectivamente. Se puede observar que si $\hat{G}(s) = G(s)$ y $\hat{T}_d = T_d$ el sistema se comporta cono un esquema libre de retardos.

# Capítulo 3

# Esquema de teleoperación bilateral

## 3.1. Arquitectura de teleoperación bilateral maestro-esclavo

Un esquema de teleoperación como se mencionó anteriormente consta de la realización de actividades de forma remota. En el sitio local se encuentra el maestro el cual envía los comandos al sitio remoto en donde el esclavo realiza las tareas. En un sistema de teleoperación unilateral, es considerado únicamente un sentido en el flujo de la información, se considera un sistema en cascada dónde solo los comandos del operador son enviados al esclavo para que se sigan. En un sistema de teleoperación bilateral, la información que proviene de los sensores del esclavo se envía al operador, esta información puede ser enviada en el mismo canal de operación o en otro distinto, formando un lazo cerrado interno. En el caso de la arquitectura de teleoperación bilateral, ésta representa un mayor reto debido al hecho que el robot maestro y el robot esclavo se encuentran acoplados por un algoritmo de control, es decir, existe un compromiso entre el seguimiento de trayectoria del robot esclavo al maestro y el seguimiento del robot maestro a la trayectoria de referencia. Un problema importante en los sistemas de teleoperación bilateral es la presencia de retardos en los canales de comunicación puesto que pueden ocasionar un deterioro en el desempeño del sistema e incluso provocar inestabilidad. Estos retardos existen por distintas razones, debido a comunicaciones a largas distancias, comunicación a través de Internet, retardos en los sensores del esclavo y retardos en los tiempos de procesamiento del controlador.

Las arquitecturas de teleoperación pueden ser también clasificadas de acuerdo al tipo de información adquirida a través de los sensores que interactúan con el ambiente, esta información es intercambiada entre el maestro y el esclavo. Es así que las arquitecturas más comunes dentro de esta clasificación son de error de posición o PERR por sus siglas en inglés, retroalimentación de fuerza kinestésica y la arquitectura de control de 4 canales. La arquitectura de teleoperación PERR se basa en el envío de la información de la posición entre el maestro y el esclavo. Su principal ventaja de acuerdo con [14] es que no requiere de sensores de fuerza. Una desventaja de esta arquitectura es que no es posible ofrece una gran fuerza de reflexión y maniobrabilidad al mismo tiempo. La arquitectura de retroalimentación de fuerza kinestésica además de los

sensores de posición utiliza sensores de fuerza para retroalimentar fuerzas presentes en la interacción esclavo-ambiente, por lo general en esta arquitectura la ganancia de retroalimentación de fuerza es atenuada para evitar problemas de inestabilidad y bajo desempeño. En la arquitectura de cuatro canales se presenta un intercambio de información maestro-esclavo tanto de posiciones como de fuerzas, en esta teleoperación el maestro cuenta también con sensores de fuerza, enviando esta información al robot esclavo. En este tipo de arquitectura existe una mejor estimación de la interacción del esclavo con el sitio remoto, proporcionando al operador un panorama más real del ambiente en el que el esclavo ejecuta las tareas. Esta arquitectura puede presentar problemas de desempeño y estabilidad debido a incertidumbres en el ambiente y el humano, así como por retardos en el sistema. En este trabajo de investigación la arquitectura de error de posición (PERR) es utilizada en la teleoperación maestro-esclavo, debido a que se consideran únicamente las posiciones del robot maestro y esclavo como información utilizada en el canal de comunicación.

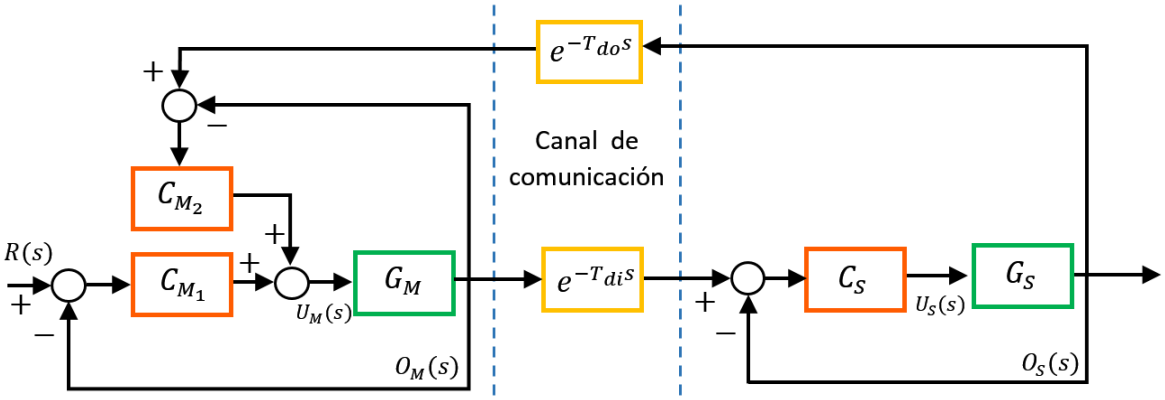



Figura 3.1: Esquema de teleoperación PERR

En la figura 3.1 se muestra el esquema de teleoperación de error de posición (PERR), en el cual se consideran retardos en el canal de comunicación representados por los bloques de color amarillo, donde $T_{di}$ y $T_{do}$ son los tiempos de retardo de recepción y envío de información respectivamente por parte del esclavo. Los bloques verdes representan la planta del maestro $G_M$ y del esclavo $G_S$ con salidas $O_M(s)$ y $O_S(s)$ respectivamente. Los bloques naranjas representan los controladores, donde $C_{M_1}$ actúa en el seguimiento de la referencia por parte del maestro y $C_{M_2}$ actúa en el seguimiento maestro y esclavo. El controlador $C_S$ actúa sobre el esclavo para realizar un seguimiento de la referencia enviada por el maestro. Las señales de control $U_M(s)$ y $U_S(s)$ están descritas por las ecuaciones 3.1 y 3.2

$$U_M(s) = C_{M_1}(s)\left(R(s) - O_M(s)\right) + C_{M_2}(s)\left(O_S(s)\, e^{-T_{do}s} - O_M(s)\right) \tag{3.1}$$

$$U_S(s) = C_S(s)\left(O_M(s)\, e^{-T_{di}s} - O_S(s)\right) \tag{3.2}$$

En el esquema PERR de la figura 3.1 se considera que el controlador $C_{M_1}$ ejecuta la tarea requerida en vez del operador humano, por lo que $C_{M_1}$ representa un controlador local. Dado el hecho que al tener dos controladores en el sitio del maestro $C_{M_1}$ y $C_{M_2}$, aumenta un grado de libertad en el ajuste de ganancias, complicando su elección, se consideran iguales los controladores locales $C_{M_1} = C_{M_2} = C_M$. En la figura 3.2 se ajusta el esquema PERR con esta consideración.

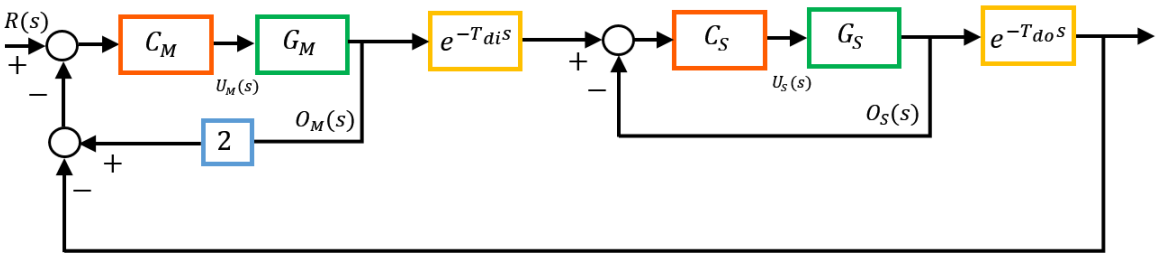



Figura 3.2: Esquema de teleoperación PERR ajustado

## 3.2. Diseño del controlador IMC.

El esquema de control propuesto para este sistema de teleoperación bilateral, consta de la implementación de un control por modelo interno que permita además de controlar remotamente al robot esclavo, estimar los efectos de los retardos presentes en la comunicación del sistema. Como se mencionó anteriormente, el sistema del robot diferencial linealizado se encuentra desacoplado, es decir cada salida depende respecto de una entrada. Al ser las dos funciones de transferencia iguales, se tratará en el diseño controlador únicamente como un sistema SISO. El controlador $C_S$, se propone de acuerdo al IMC en la ecuaciones 3.4, donde $\hat{G}_S(s)$ representa el modelo linealizado del robot esclavo

$$\hat{G}_S(s) = \hat{G}_M(s) = \frac{1}{s^2} \tag{3.3}$$

$$C_S(s) = f(s)\,\hat{G}_S(s)^{-1} \tag{3.4}$$

Nótese que $f(s)$ representa un filtro que permite hacer realizables el controlador del esclavo, así mismo el filtro elegido es un filtro de orden superior debido a que los modelos son de segundo orden mostrado en la ecuación 2.39. En la ecuación 3.5 se muestra el controlador

$$C_S(s) = \frac{(n\lambda s + 1)\,s^2}{(\lambda s + 1)^n} = \frac{n\lambda s^3 + s^2}{(\lambda s + 1)^n} \tag{3.5}$$

Donde $n$ es un entero positivo de valor mayor al orden del sistema, y $\lambda$ es un parámetro de ajuste de respuesta del sistema. En la figura 3.3 se muestra el esquema de teloperación PERR con la estructura IMC, los bloques morados representan los retardos de tiempo nominales $e^{-L_i s}$ y $e^{-L_o s}$ retardo de recepción y envío respectivamente. Las diferencia entre la salida de la planta real y nominal está dada por $\hat{D}$, este

error representa una perturbación estimada, compuesta por la perturbación externa $D$, diferencias entre la planta y el modelo, y diferencias entre los retardos de tiempo reales y nominales.

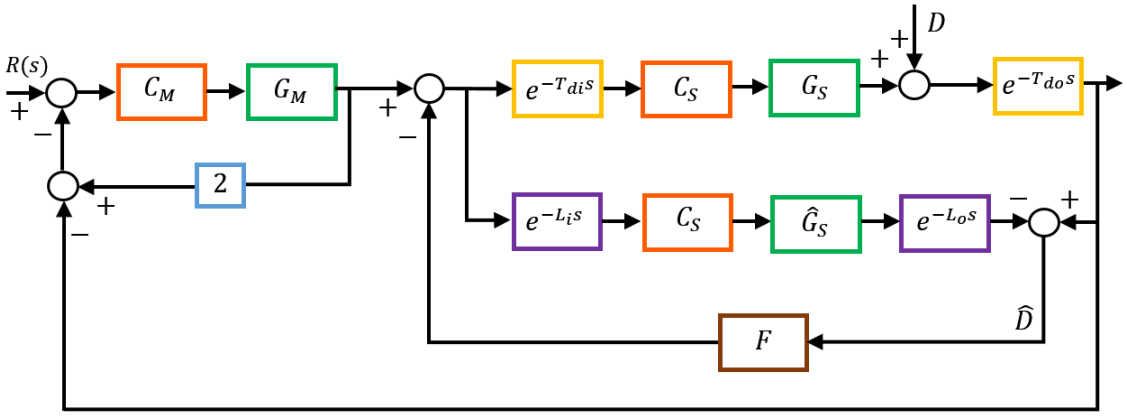



Figura 3.3: Esquema de teleoperación con IMC

### 3.2.1. Filtro de rechazo de perturbaciones

El bloque café $F(s)$ es un filtro que sirve en el rechazo de perturbaciones ($D$) de tipo aditivas. El diseño de este filtro mostrado en la ecuación 3.6 se propone de acuerdo con [21].

$$F(s) = \frac{A(s)}{Q(s)\,P(s)} \tag{3.6}$$

La función de transferencia $A(s)/P(s)$ representa un modelo interno de perturbaciones [14], y $Q(s)$ una función polinomial. $Q(s)$ y $P(s)$ deben ser funciones con ceros estables para garantizar estabilidad en el filtro, una forma simple de seleccionar estas funciones es decrecentar el número de parámetros ajustables, propuesto en [12],

$$Q(s) = 10, \quad P(s) = (T_0 s + 1)^r$$

Donde $T_0$ es un tiempo constante y $r$ el es orden del filtro, estos parámetros de diseño determinan la velocidad en el proceso de absorción de perturbaciones. La perturbación es absorbida de forma rápida si se elijen valores pequeños para $T_0$ y $r$. El polinomio $A(s)$ puede ser seleccionado como cualquier polinomio que sea estable. Se elige

$$A(s) = (\epsilon s + 1)^2$$

$$F(s) = \frac{(\epsilon s + 1)^2}{10\,(T_0 s + 1)^r} \tag{3.7}$$

### 3.2.2. Controlador PID.

Además del controlador por modelo interno, es agregado un controlador $PID$ que estabiliza el modelo de la planta y permite la reducción del error entre la planta y su modelo, este controlador es agregado a la entrada de la planta y del modelo, $G_S(s)$ y $\hat{G}_S(s)$ respectivamente. En la ecuación 3.8 se define el controlador $PID$ con ganancias proporcional, integral y derivativa $(k_p, k_i, k_d)$, este mismo tipo de controlador es utilizado como controlador local $C_M$ del robot maestro. En la figura 3.4 se muestra el bloque azul del controlador PID agregado al esquema de teleoperación.

$$PID = k_p\left(1 + \frac{k_i}{s} + k_d s\right) \tag{3.8}$$

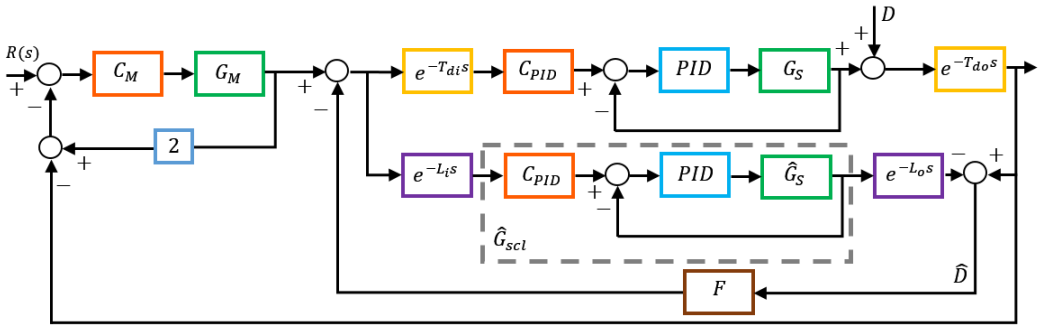



Figura 3.4: Esquema de teleoperación con PID

En las ecuaciones 3.9 y 3.10 se muestra lazo cerrado interno del controlador $PID$ con la planta y con el modelo del robot diferencial respectivamente, donde $G_{PID}$ representa el lazo cerrado del controlador $PID$ con la planta $G_S(s)$ y $\hat{G}_{PID}$ representa el lazo cerrado del controlador $PID$ con el modelo de la planta $\hat{G}_S(s)$.

$$G_{PID}(s) = \frac{(PID)\ G_S(s)}{1 + (PID)\ G_S(s)} \tag{3.9}$$

$$\hat{G}_{PID}(s) = \frac{(PID)\ \hat{G}_S(s)}{1 + (PID)\ \hat{G}_S(s)} = \frac{(k_p k_d)\, s^2 + (k_p)\, s + k_p k_i}{s^3 + (k_p k_d)\, s^2 + (k_p)\, s + k_p k_i} \tag{3.10}$$

Con el lazo cerrado del modelo $\hat{G}_{PID}(s)$ se obtiene el nuevo controlador $C_{PID}$, utilizando el mismo filtro propuesto en 2.39 y la inversa de $\hat{G}_{PID}$, mostrado en la siguiente ecuación

$$C_{PID}(s) = f\ \hat{G}_{PID}(s)^{-1}$$

Con el filtro utilizado en la ecuación 3.4 y $n = 3$, este valor es utilizado debido a que experimentalmente el filtro presenta un mejor desempeño, el controlador $C_{PID}$ queda de la siguiente manera

$$C_{PID}(s) = \frac{3\lambda s + 1}{(\lambda s + 1)^3}\left(\frac{s^3 + (k_p k_d)\, s^2 + (k_p)\, s + k_p k_i}{(k_p k_d)\, s^2 + (k_p)\, s + k_p k_i}\right) \tag{3.11}$$

$$= \frac{b_0 s^4 + b_1 s^3 + b_2 s^2 + b_3 s + b_4}{a_1 s^5 + a_2 s^4 + a_3 s^3 + a_4 s^2 + a_5 s + a_6}$$

$$a_1 = \lambda^3 k_p k_d \qquad b_0 = 3\lambda$$
$$a_2 = \lambda^3 k_p + 3\lambda^2 k_p k_d \qquad b_1 = 3\lambda k_p k_d + 1$$
$$a_3 = \lambda^3 k_p k_i + 3\lambda^2 k_p + 3\lambda k_p k_d \qquad b_2 = k_p k_d + 3\lambda k_p$$
$$a_4 = 3\lambda^2 k_p k_i + 3\lambda k_p + k_p k_d \qquad b_3 = 3\lambda k_p k_i + k_p$$
$$a_5 = 3\lambda k_p k_i + k_p \qquad b_4 = k_p k_i$$
$$a_6 = k_p k_i$$

En la figura 3.4 se muestra un recuadro en linea punteada gris a $\hat{G}_{scl}$, el cual representa el lazo cerrado interno $\hat{G}_{PID}(s)$ por el controlador $C_{PID}(s)$

$$\hat{G}_{scl} = C_{PID}(s)\,\hat{G}_{PID}(s)$$

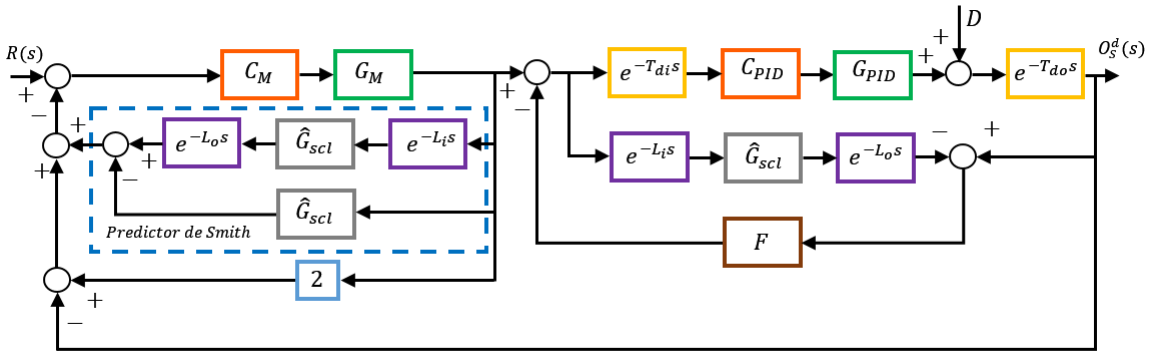



Figura 3.5: Esquema de teleoperación completo

En la figura 3.5 se presenta el esquema de teloperación IMPACT, el cual contempla la implementación de un predictor de Smith mostrado en la figura 3.5 con línea punteada en azul en el sitio local del robot maestro. Este predictor es formado con base en el modelo y controlador del robot esclavo, es decir, $\hat{G}_{scl}$, utilizando los retardos nominales $L_o$ y $L_i$, el predictor añade un mayor desempeño al sistema teleoperado frente a los retardos presentes en la comunicación. Como se mencionó anteriormente, $T_{di}$ y $T_{do}$ son los retardos de tiempo reales los cuales se consideran constantes. $T_{di}$ es el retardo de comunicación del maestro al esclavo y $T_{do}$ es el retardo del esclavo al

maestro. De la misma forma, $L_o$ y $L_i$ los retardos de tiempo constantes nominales. En este esquema de teleoperación, los retardos tanto en el envío como en la recepción de datos pueden ser considerados iguales, es decir un esquema de comunicación simétrico. Así también, se puede considerar un esquema asimétrico, donde la recepción y el envío de datos cuentan con tiempos de retardo diferentes. Por otro lado, la existencia de incertidumbre entre los retardos nominales y los retardos reales es también contemplada en el esquema de control.

## 3.3. Análisis de estabilidad.

Es posible determinar la función de transferencia del lazo cerrado de la figura 3.5, de la salida $O_S^d(s)$ con respecto a la entrada de referencia $R(s)$, en la figura 3.6 se muestra el esquema simplificado de teleoperación, al reducir el número de bloques del esquema IMPACT de la figura 3.5.

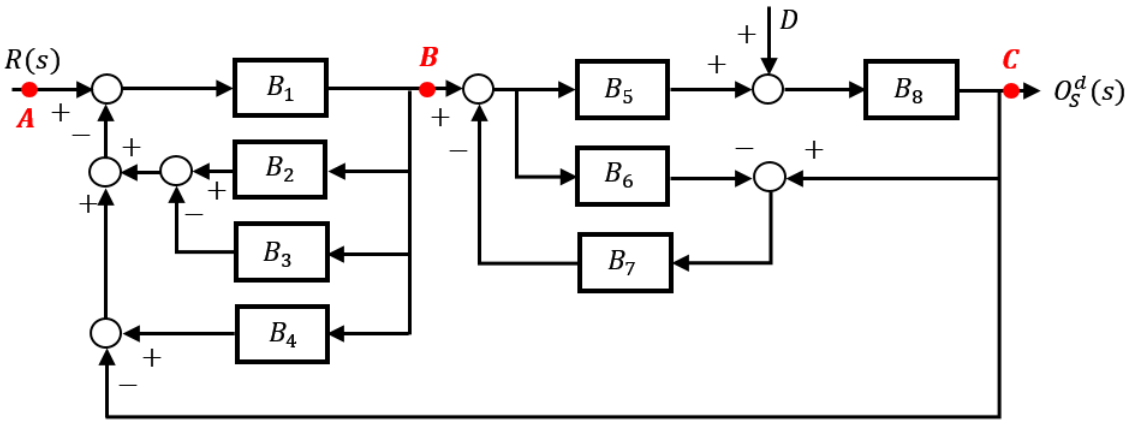



Figura 3.6: Esquema de teleoperación simplificado

Donde

$$B_1 = C_M \hat{G}_M$$
$$B_2 = \hat{G}_{scl}\, e^{-2Ls}$$
$$B_3 = \hat{G}_{scl}$$
$$B_4 = 2$$
$$B_5 = \hat{G}_{scl}\, e^{-Ls}$$
$$B_6 = \hat{G}_{scl}\, e^{-2Ls}$$
$$B_7 = \frac{A(s)}{Q(s)\,P(s)}$$
$$B_8 = e^{-Ls}$$

El lazo cerrado del esquema está basado en la planta nominal, es decir $G_M = \hat{G}_M$ y $G_S = \hat{G}_S$, del mismo modo los retardos del sistema de teleoperación se consideran nominales es decir $T_{di} = T_{do} = L_i = L_o = L$. De igual forma la función de transferencia $\hat{G}_{scl} = C_{PID}\hat{G}_{PID}$, así mismo se considera que $G_{PID} = \hat{G}_{PID}$. En la figura 3.7 se muestra la subsección $B - C$ del diagrama completo $A - C$.

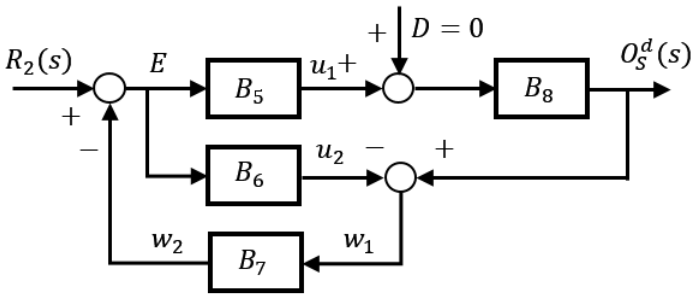



Figura 3.7: Diagrama bloques B-C

Se obtiene la función de transferencia $\frac{O_S^d(s)}{R_2(s)}$

$$\begin{aligned} E &= R_2 - w_2 \\ &= R_2 - B_7 w_1 \\ &= R_2 - B_7\left(O_S^d - u_2\right) \\ &= R_2 - B_7\left(O_S^d - B_6 E\right) \\ E &= R_2 + B_7 B_6 E - B_7 O_S^d \end{aligned} \tag{3.12}$$

$$\begin{aligned} O_S^d &= B_8 B_5 E \\ \Rightarrow E &= \frac{O_S^d}{B_8 B_5} \end{aligned} \tag{3.13}$$

Sustituyendo la ecuación 3.13 en 3.12, y despejando la salida y la referencia deseada se obtiene la función de transferencia

$$\frac{O_S^d}{R_2\left(s\right)} = \frac{B_8 B_5}{1 + B_8 B_7 B_5 - B_7 B_6} \tag{3.14}$$

En la figura 3.8 se muestra el diagrama a bloques $A - B$, donde el bloque $B_9$ representa la función de transferencia $\frac{O_S^d}{R_2(s)}$
Se obtiene la función de transferencia $\frac{O_S^d(s)}{R(s)}$

$$\begin{aligned} E &= R - w_1 \\ &= R - (w_2 + w_3) \end{aligned}$$

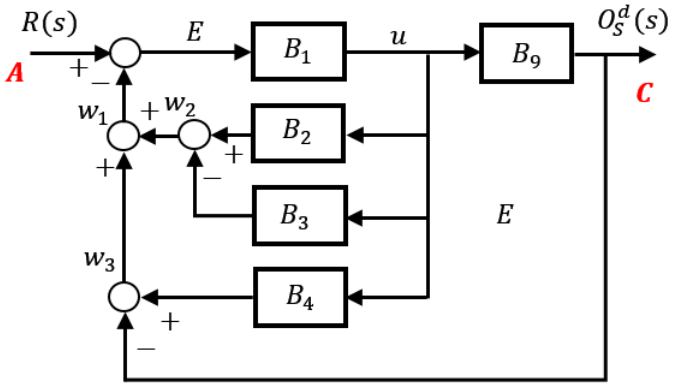



Figura 3.8: Diagrama bloques A-C

$$= R - ((\, B_2 u - B_3 u\,) + (B_4 u - O_S^d))$$
$$= R - ((\, B_2 B_1 E - B_3 B_1 E\,) + (B_4 B_1 E - O_S^d))$$
$$E = R + O_S^d + (B_3 - B_4 - B_2)\, B_1 E \tag{3.15}$$

$$O_S^d = B_9 B_1 E$$
$$\Rightarrow E = \frac{O_S^d}{B_9 B_1} \tag{3.16}$$

Sustituyendo la ecuación 3.16 en 3.15, y despejando la salida y la referencia deseada se obtiene la función de transferencia

$$\frac{O_S^d\,(s)}{R\,(s)} = \frac{B_9 B_1}{1 - B_9 B_1 - (B_3 - B_4 - B_2)\, B1} \tag{3.17}$$

Donde $N_R$ es es numerador de la función de transferencia de la ecuación anterior, y $D_R$ el denominador de la función. Se obtienen ambas partes

$$N_R = B_9 B_1 = \frac{B_8 B_5 B_1}{1 + B_8 B_7 B_5 - B_7 B_6}$$

$$D_R = 1 - B_9 B_1 - (B_3 - B_4 - B_2)\, B1$$
$$= \frac{1 + \alpha + \beta}{1 + B_8 B_7 B_5 - B_7 B_6}$$

Donde

$$\alpha = B_8 B_7 B_5 - B_7 B_6 - B_3 B_1 + B_4 B_1 + B_2 B_1 - B_8 B_7 B_5 B_3 B_1$$

$$\beta = B_8 B_7 B_5 B_4 B_1 + B_8 B_7 B_5 B_2 B_1 + B_7 B_6 B_3 B_1 - B_7 B_6 B_4 B_1 - B_7 B_6 B_2 B_1 - B_8 B_5 B_1$$

Se puede observar al sustituir los valores de los bloques que

$$B_8B_7B_5 - B_7B_6 = 0$$
$$B_2B_1 - B_8B_5B_1 = 0$$
$$B_7B_6B_3B_1 - B_8B_7B_5B_3B_1 = 0$$
$$B_8B_7B_5B_4B_1 - B_7B_6B_4B_1 = 0$$
$$B_8B_7B_5B_2B_1 - B_7B_6B_2B_1 = 0$$

Por lo que la función de transferencia $\frac{N_R}{D_R}$ queda de la siguiente forma

$$\frac{N_R}{D_R} = \frac{B_8B_5B_1}{1 + B_4B_1 - B_3B_1}$$

Obteniendo la función de transferencia al sustituir los valores de los bloques $B_1,\ B_3,\ B_4,\ B_5\ y\ B_8$

$$\frac{O_S^d(s)}{R(s)} = \frac{N_R}{D_R} = \frac{C_M\hat{G}_M\hat{G}_{scl}}{1 + 2C_M\hat{G}_M - C_M\hat{G}_M\hat{G}_{scl}}\ e^{-2Ls}$$

$N(s)$ es la función de transferencia de la salida $O_S^d(s)$ y la referencia retardada $R^d(s)$, donde $R^d(s) = R(s)\,e^{-2Ls}$

$$N(s) = \frac{O_S^d(s)}{R^d(s)} = \frac{C_M\hat{G}_MC_{PID}\hat{G}_{PID}}{1 + 2C_M\hat{G}_M - C_M\hat{G}_MC_{PID}\hat{G}_{PID}} \tag{3.18}$$

Al sustituir las ecuaciones 3.3, 3.9 y 3.11 en 3.18

$$N(s) = \frac{O_S^d(s)}{R^d(s)} = \frac{(3k_pk_d\lambda)\,s^3 + (3k_p\lambda + k_pk_d)\,s^2 + (3k_pk_i\lambda + k_p)\,s + k_pk_i}{a_0s^6 + a_1s^5 + a_2s^4 + a_3s^3 + a_4s^2 + a_5s + a_6} \tag{3.19}$$

$$a_0 = \lambda^3$$
$$a_1 = 2k_pk_d\lambda^3 + 3\lambda^2$$
$$a_2 = 2k_p\lambda^3 + 6k_pk_d\lambda^2 + 3\lambda$$
$$a_3 = 2k_pk_i\lambda^3 + 6k_p\lambda^2 + 3k_pk_d\lambda + 1$$
$$a_4 = 6k_pk_i\lambda^2 + 3k_p\lambda + k_pk_d$$
$$a_5 = 3k_pk_i\lambda + k_p$$
$$a_6 = k_pk_i$$

La estabilidad en la función de transferencia de la ecuación 3.19 se puede evaluar utilizando el criterio de estabilidad de Routh Hurwitz [19]. Si en la primer fila de la tabla de Routh todos lo elementos son positivos, de acuerdo con el criterio del mismo los polos del lazo cerrado se encuentran en el semiplano izquierdo del plano complejo siendo el sistema estable. Para que todos lo elementos de la primer columna de la tabla de Routh sean positivos para la función de transferencia 3.19, es necesario cumplir con $\lambda, k_p, k_i, k_d > 0$ y $k_pk_d > \frac{1}{2}k_i$.

## 3.4. Simulación numérica.

Se realiza en simunlink Matlab un diagrama a bloques del esquema de control de la figura 3.5. La simulación se realizó con un paso de integración $h = 0.01$ segundos con una rutina de integración Runge Kutta de orden 4, con un tiempo de simulación de 120 segundos. Las trayectorias de referencia son un círculo de radio $r = 1\,m$, con frecuencia $w = 0.2\ rad/s$ y una lemniscata de ancho de $1\,m$ y de largo $1\,m$, ambas trayectorias con centro en el origen. En la tabla 3.1 se muestran los valores empleados para la simulación.

| $\lambda$ | $T_0$ | $n$ | $r$ | $k_p$ | $k_i$ | $k_d$ |
|---|---|---|---|---|---|---|
| 0.25 | 0.25 | 3 | 3 | 10 | 0.01 | 25 |

Tabla 3.1: Valores de simulación

La sintonización de las ganancias del PID se hicieron de manera heurística, estos valores son utilizados en todos los controladores PID del sistema, tanto en el controlador del robot esclavo como del maestro. Se realiza la primer simulación con una trayectoria circular con los siguientes tiempos de retardo

| $T_{di}$ | $T_{do}$ | $L_i$ | $L_o$ |
|---|---|---|---|
| 100 $ms$ | 100 $ms$ | 100 $ms$ | 100 $ms$ |

Tabla 3.2: Retardos simulación 1

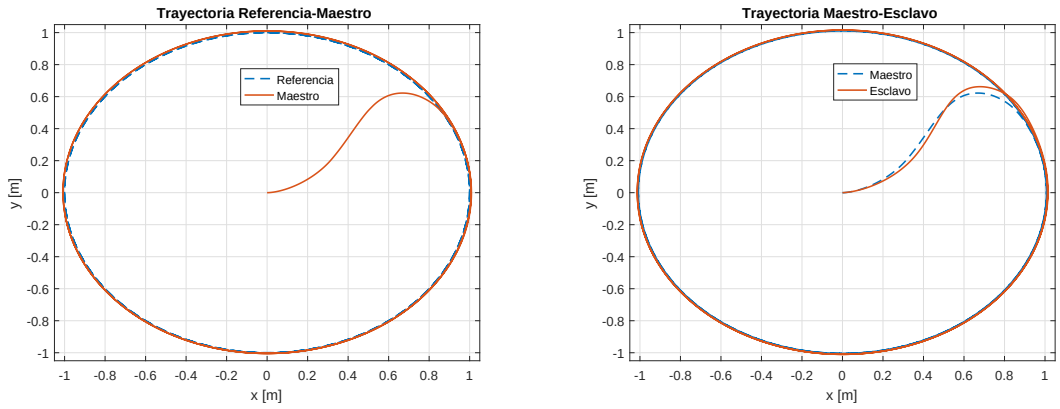



Figura 3.9: Trayectorias simulación 1 círculo

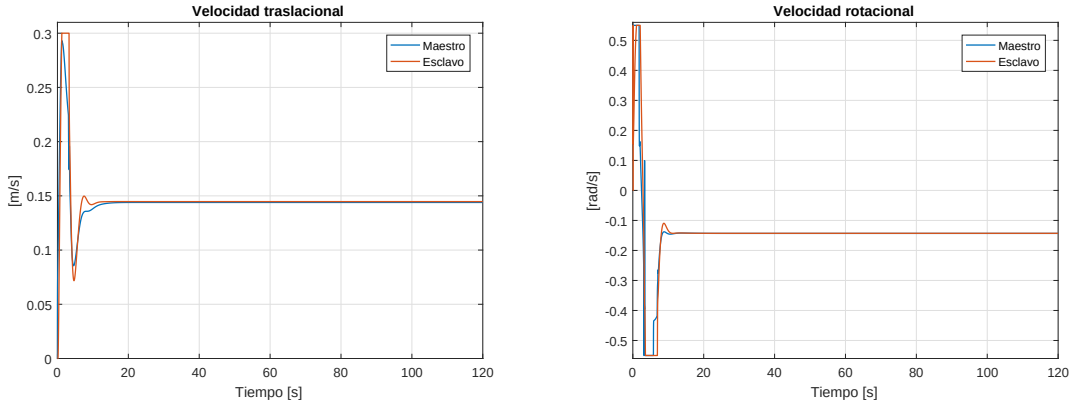



Figura 3.10: Variables de control simulación 1 círculo

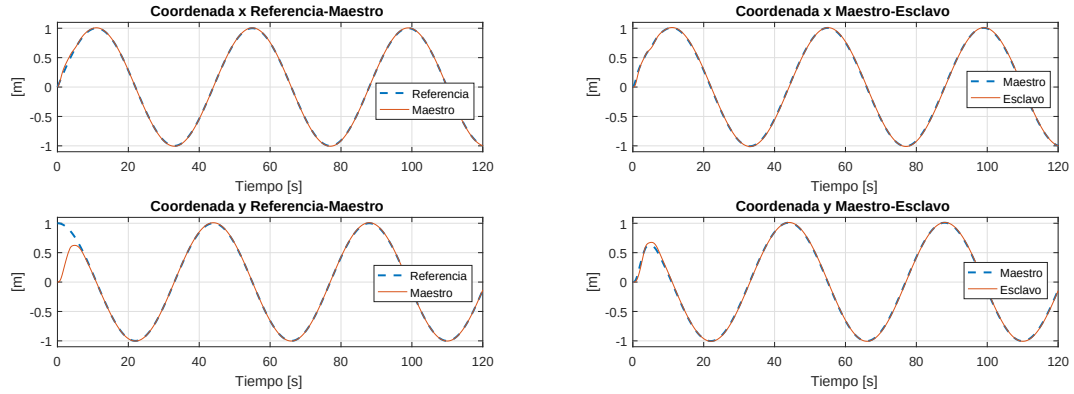



Figura 3.11: Coordenadas x-y simulación 1 círculo

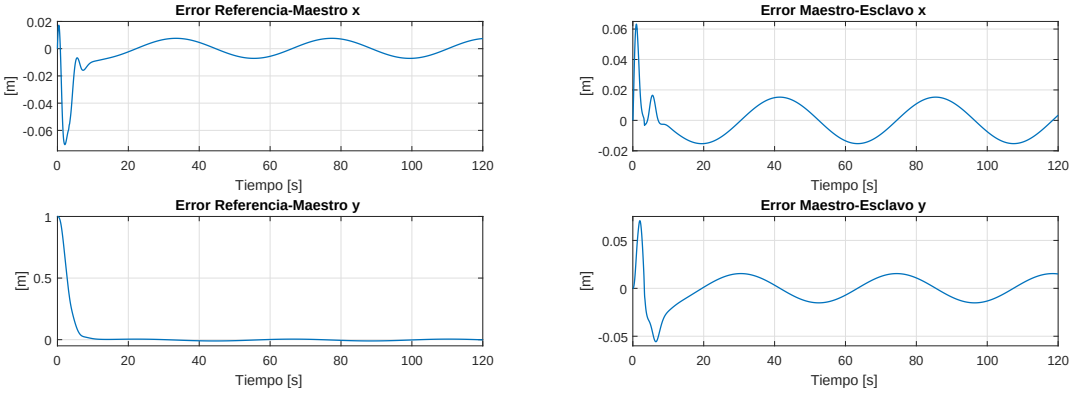



Figura 3.12: Errores de seguimiento simulación 1 círculo

En la figura 3.9 se muestra que el seguimiento de trayectoria del robot maestro a la referencia y el seguimiento entre del robot esclavo y el robot maestro. En la figura 3.10 se observan que las variables de control $v$ y $w$ de ambos robots, se muestra que después de 10 segundos las dos variables permanecen con un valor fijo. En la figura 3.11 se muestran las coordenadas x y ya de la referencia, el robot maestro y el robot esclavo, mientras que los errores mostrados en la figura 3.12 permanecen acotados con un valor muy pequeño alrededor de +/- 2 centímetros. Las 3.13, 3.14 y 3.16 muestran el desempeño del controlador frente a una trayectoria tipo lemnicasta.

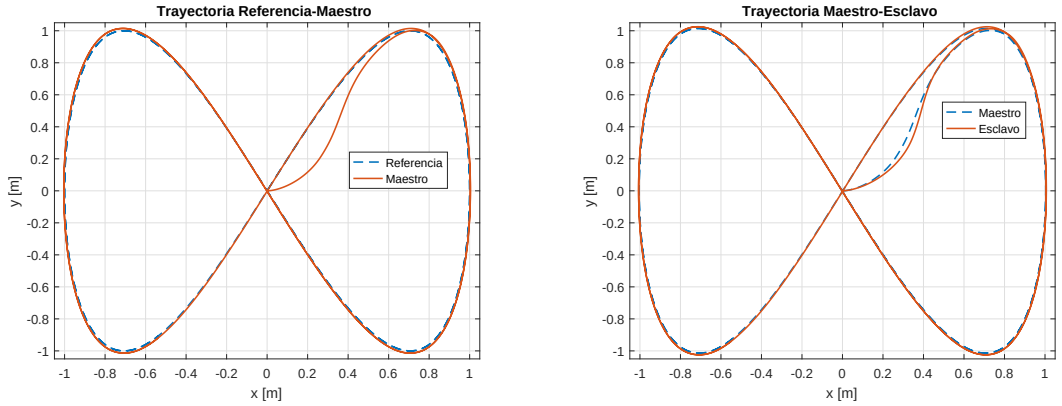



Figura 3.13: Trayectorias simulación 1 lemniscata

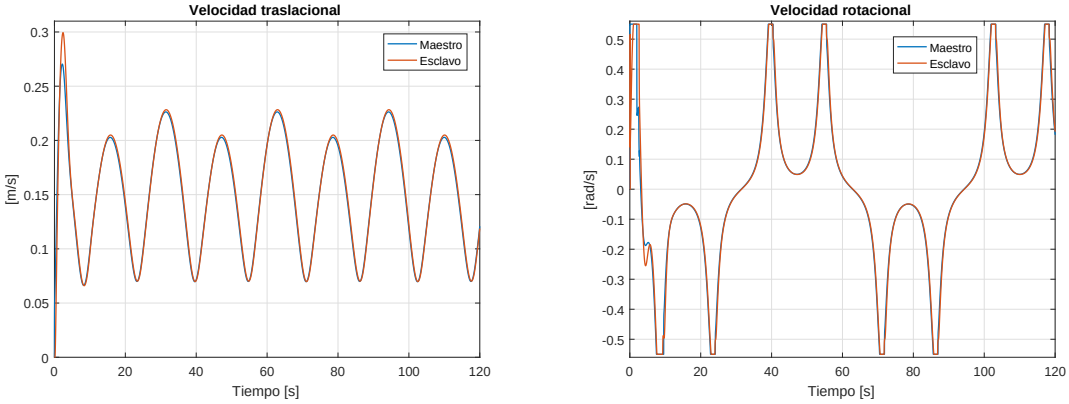



Figura 3.14: Variables de control simulación 1 lemniscata

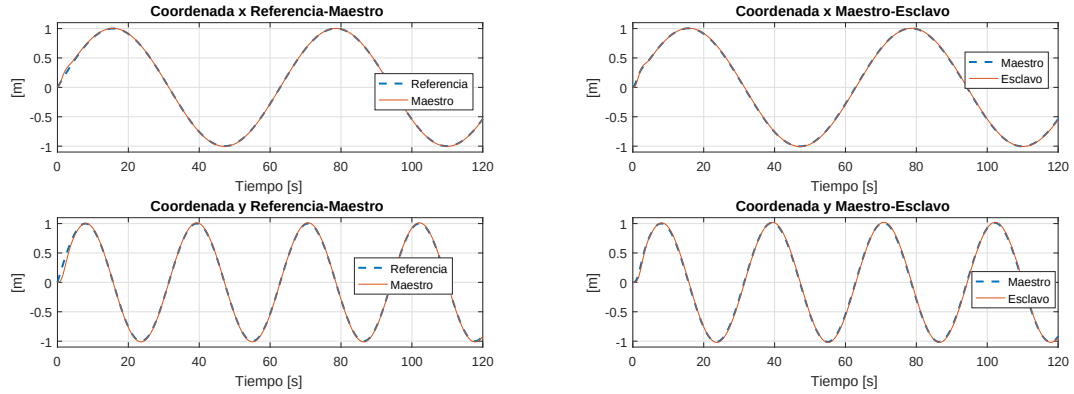



Figura 3.15: Coordenadas x-y simulación 1 lemniscata

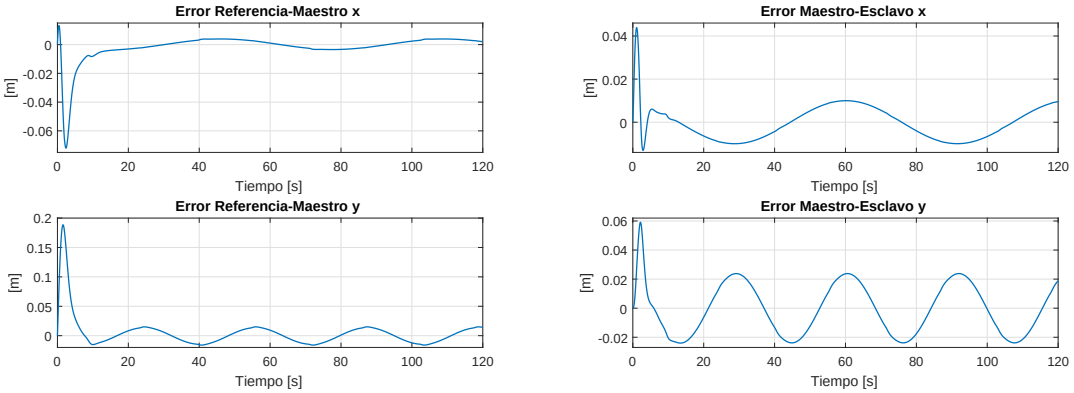



Figura 3.16: Errores de seguimiento simulación 1 lemniscata

De la misma forma que en la trayectoria circular, los retardos reales y nominales son iguales a 100 $ms$, así mismo se aprecia que las variables de control 3.14 no tienden a un valor constante, en cambio éstas varían de forma acotada. Esto debido a que la trayectoria lemniscata cuenta con tramos de curvas cerradas y tramos rectos que provocan que las variables de control cambien continuamente. En la figura 3.15 se muestran las coordenadas $x$ y $y$ de ambos robots y la referencia, los errores mostrados en la figura 3.16 son de alrededor de+/- 2 centímetros, mostrando un buen desempeño frente a los retardos de tiempo inducidos. Se muestra una segunda simulación con errores entre los valores de los retardos reales y nominales mostrado en la tabla 3.3

| $T_{di}$ | $T_{do}$ | $L_i$ | $L_o$ |
|---|---|---|---|
| 200 $ms$ | 200 $ms$ | 100 $ms$ | 100 $ms$ |

Tabla 3.3: Retardos simulación 2

El valor del retardo real de 200 $ms$ es utilizado como un valor promedio en los retardos presentes en sistemas teleoperados a través de internet. En este caso el valor real del retardo es de dos veces orden de magnitud del valor nominal de 100 $ms$. En las figuras 3.17, 3.18 y 3.20 se muestra el desempeño del controlador. En la figura 3.17 se muestran las gráficas del seguimiento de trayectoria, donde se pude apreciar un sobre impulso, esto debido al aumento del retardo en el sistema y el aumento del porcentaje de error de estimación. En la figura 3.18 se aprecia que le toma más tiempo a las variables de control llegar al valor estacionario, alrededor de 20 segundos.

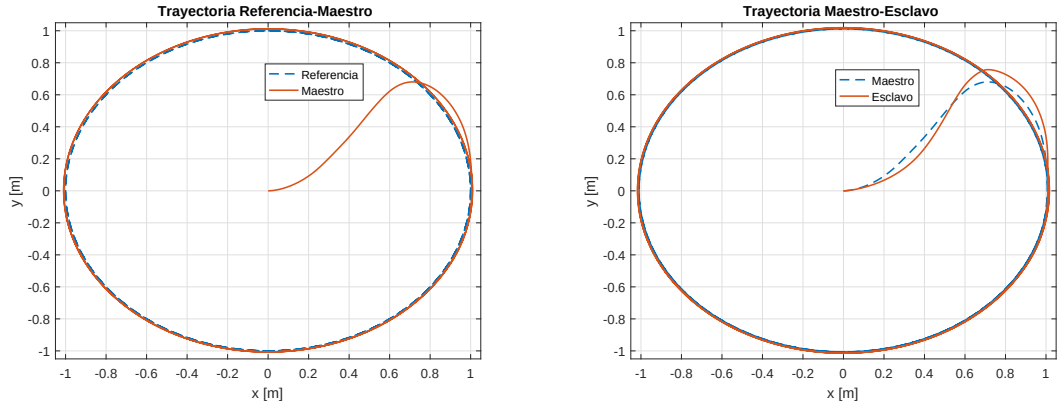



Figura 3.17: Trayectorias simulación 2 círculo

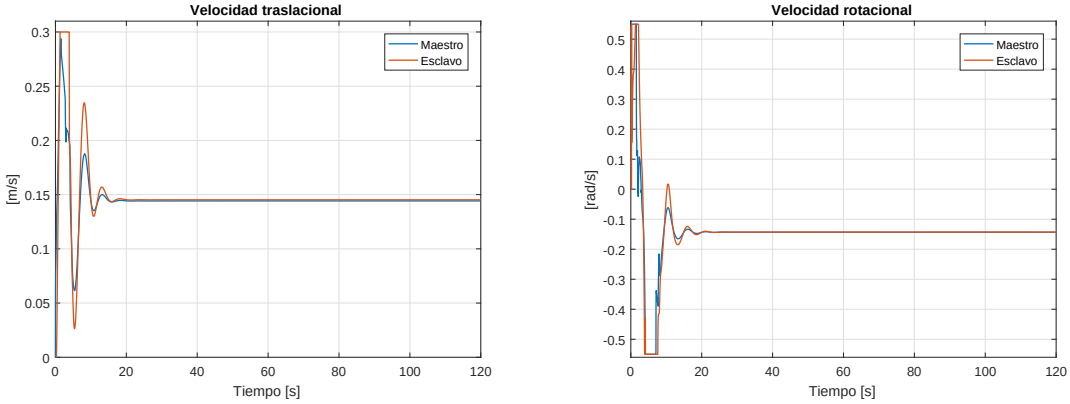



Figura 3.18: Variables de control simulación 2 círculo

En la figura 3.19 se observan las coordendas $x$ y $y$ de los robots y la referencia, así mismo en la figura 3.20 es posible ver que también los errores de seguimiento aumentaron en los primeros segundos de simulación provocando mayores oscilaciones antes de llegar al estado estacionario.

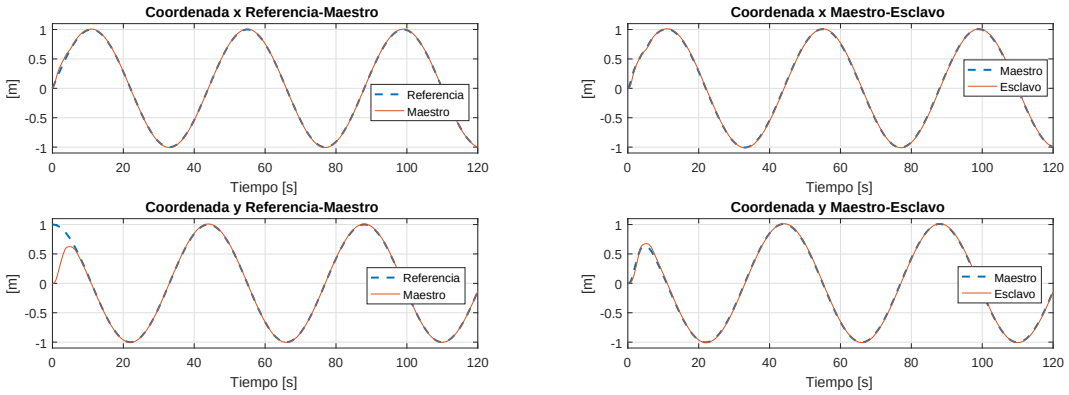



Figura 3.19: Coordenadas x-y simulación 2 círculo

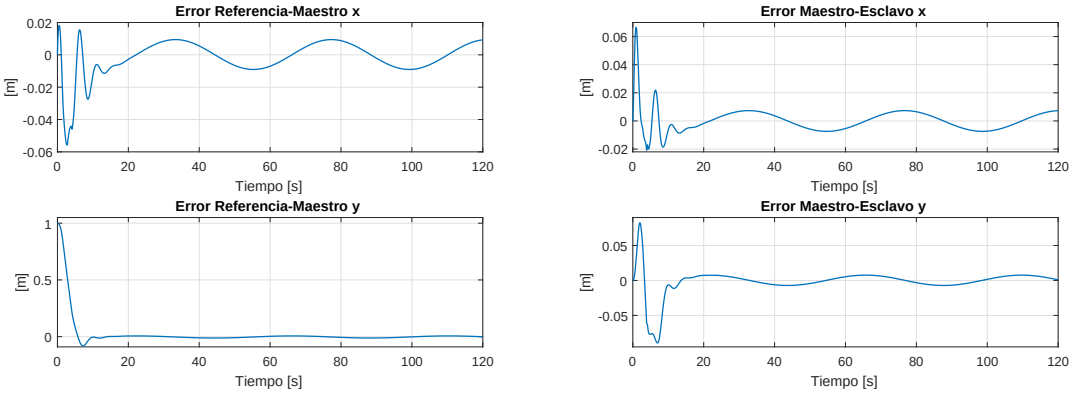



Figura 3.20: Errores de seguimiento simulación 2 círculo

En las figuras 3.21, 3.22, 3.23 y 3.24 se muestra la reducción del desempeño del controlador para la trayectoria lemniscata, esto al igual que la trayectoria circular se debe al aumento del retardo real y el porcentaje de error entre retardos reales y nominales. En las simulaciones mostradas en esta sección se tomo en consideración un sistema simétrico donde los valores de retardo tanto reales como nominales de envío y recepción al robot esclavo son idénticos. El desempeño del controlador para un sistema asimétrico es similar al simétrico, mostrando que la importancia en el desempeño del controlador radica en los valores de los retardos así como en el porcentaje de error entre los retardos reales y nominales. Siendo así, entre menor porcentaje de error mejor desempeño del controlador.

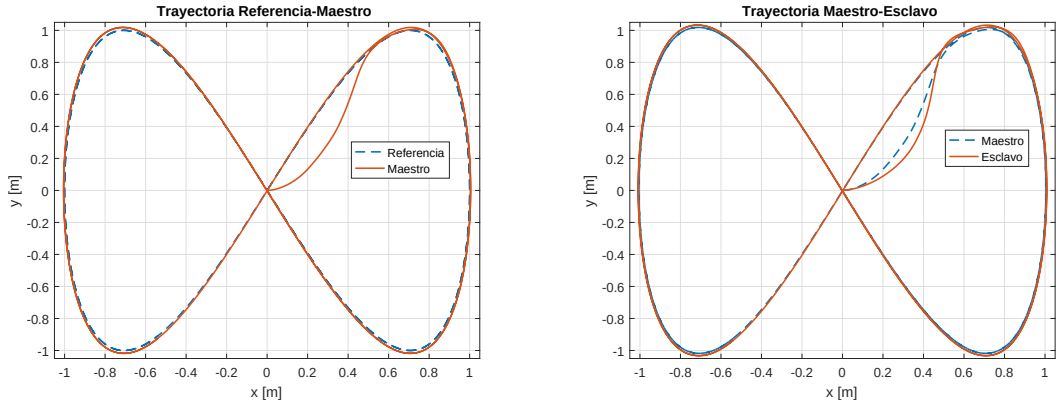



Figura 3.21: Trayectorias simulación 2 lemniscata

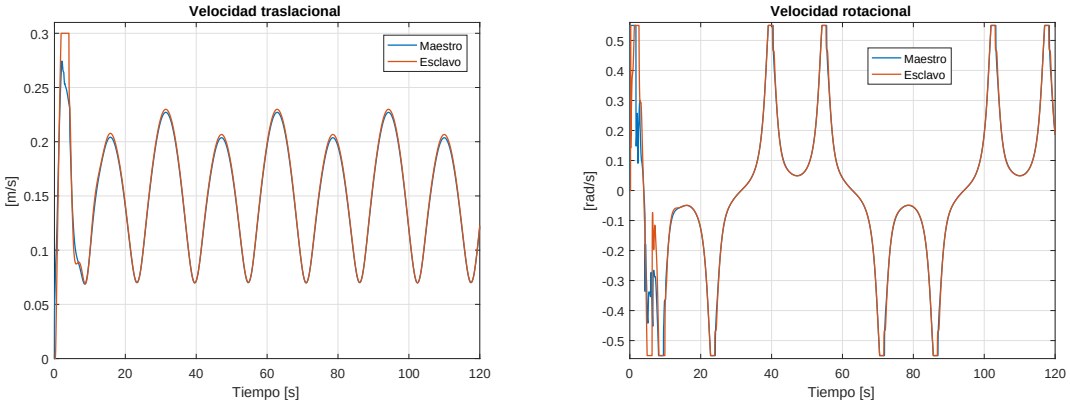



Figura 3.22: Variables de control simulación 2 lemniscata

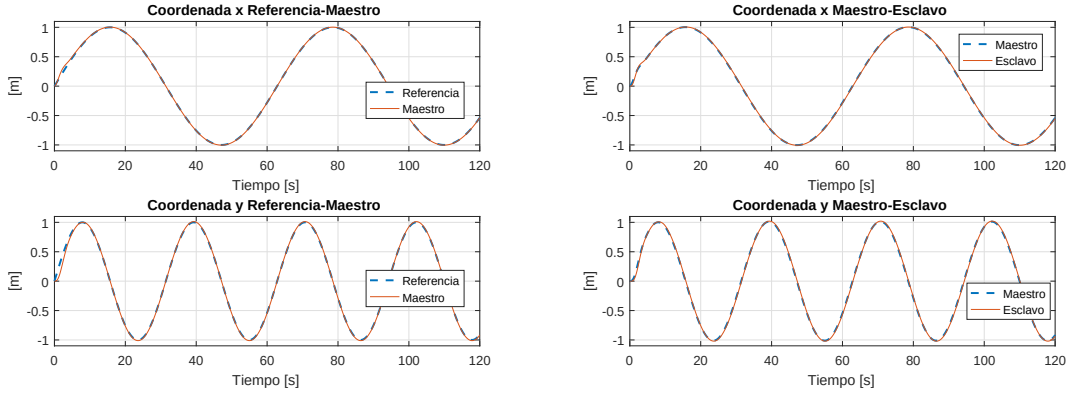



Figura 3.23: Coordenadas x-y simulación 2 lemniscata

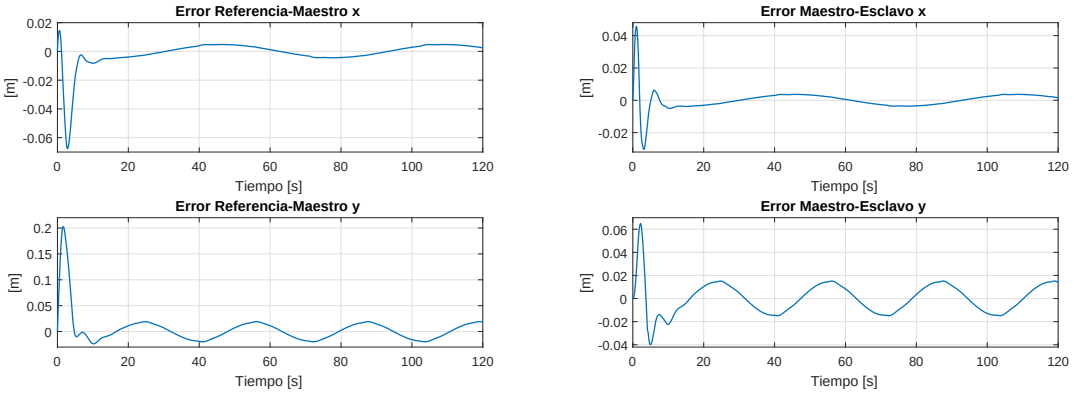



Figura 3.24: Errores de seguimiento simulación 2 lemniscata

Con las simulaciones hechas se puede apreciar que el sistema responde adecuadamente para retardos hasta de 200 milisegundos, sin embargo se hicieron pruebas con valores de retardo mayores mostrando que con un retardo de medio segundo el desempeño del controlador es bajo con un mal seguimiento de la referencia y seguimiento entre los robots, por otro lado con un valor de retardo de 0.6 segundos o mayor el sistema se inestabiliza, sin embargo estos valores de retardos son casos extremos en una comunicación por internet puesto que el valor nominal del retardo a través de internet oscila en 200 milisegundos.

# Capítulo 4

# Plataforma experimental

## 4.1. Robot Pioneer P3-DX.

Los experimentos realizados en este trabajo se efectuaron con un robot tipo 2.0 Pioneer 3DX de la marca Mobile Robots.

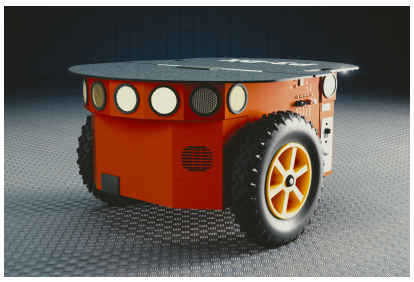


Figura 4.1: Robot Pioneer P3-DX. [1]

Este robot cuenta con un peso de 9kg, con capacidad de soportar una carga hasta de 17 kg, cuenta con una altura de 237 mm y largo de 455 mm. En la figura 4.2 se muestran las dimensiones del robot Pionner [2]. La medición de la posición del robot es realizada por medio de odometría, es decir, el robot cuenta con un encoder de 500 pulsos de resolución con el cual calcula la distancia recorrida del robot de acuerdo con las velocidades en cada llanta. El Pionneer P3-DX es capaz de alcanzar una velocidad traslacional $v$ de 1.2 $m/s$ y una velocidad rotacional de hasta 300 $deg/s$. Este robot cuenta con un controlador PID interno que envía una señal interna de control a cada llanta para moverlo, es decir después de enviarle el usuario la señal de control, esta señal pasa por el controlador interno para mejorar su desempeño.

El robot Pioneer P3-DX es de arquitectura cerrada por lo cual no fue posible obtener un modelado dinámico del robot, y por esta y otras razones se optó por un diseño de un controlador por medio del modelo cinemático de un robot móvil. Sin embargo, las características que el robot ofrece son suficientes para alcanzar un buen desempeño en el controlador. Además del controlador interno PID, mencionado anteriormente, que permite por medio de las variables de control enviadas al robot, seguir de manera precisa la trayectoria deseada, el robot cuenta con 8 sensores sónicos en la parte frontal del robot, lo que le ayudaría en un trabajo futuro detectar obstáculos u objetos en su trayectoria. Por último el robot es capaz de soportar un dispositivo de video, es decir, es posible conectar al robot una cámara que permita de manera más precisa la identificación de objetos u obstáculos que se pudieran presentar.

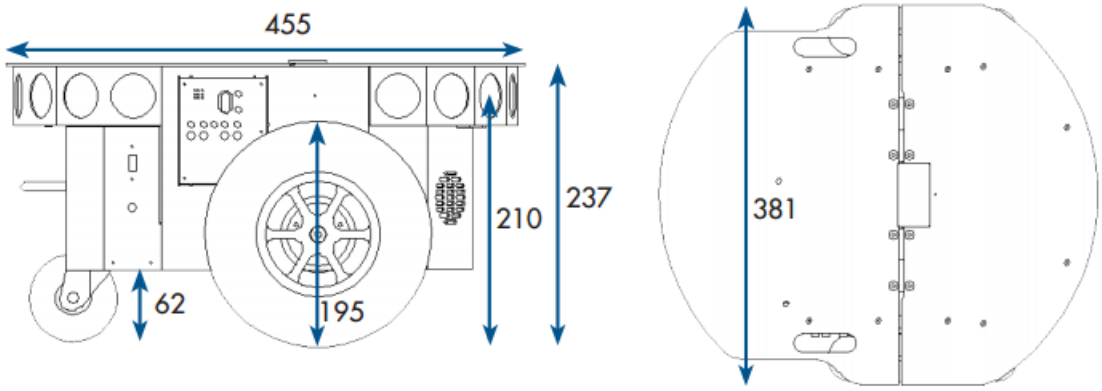



Figura 4.2: Dimensiones del Robot Pioneer P3-DX. [2]

### 4.1.1. Aria.

Los robots pioneer P3-DX y en general los robots de la marca mobile robots, cuentan con un software propio llamdado ARIA por sus siglas en inglés (Advanced Robotics Interface for Aplications). ARIA es un ambiente de desarrollo libre con base en el lenguaje de programación C++ que permite la utilización de todas las características o sistemas del robot, como la utilización de odometría para obtener la posiciónd el robot en cada instante, así como la utilización de sensores sonicos e infrarrojos para detectar objetos u obstáculos[22]. Otra característica importante es la utilización MobileSim, el cual es un programa que permite generar un ambiente virtual donde se encuentra el robot y así probar así las diferentes estrategias de control a implementarse, este programa se conecta con las librerías de Aria, ofreciendo un robot virtual que realiza las tareas de la misma forma que un robot real.

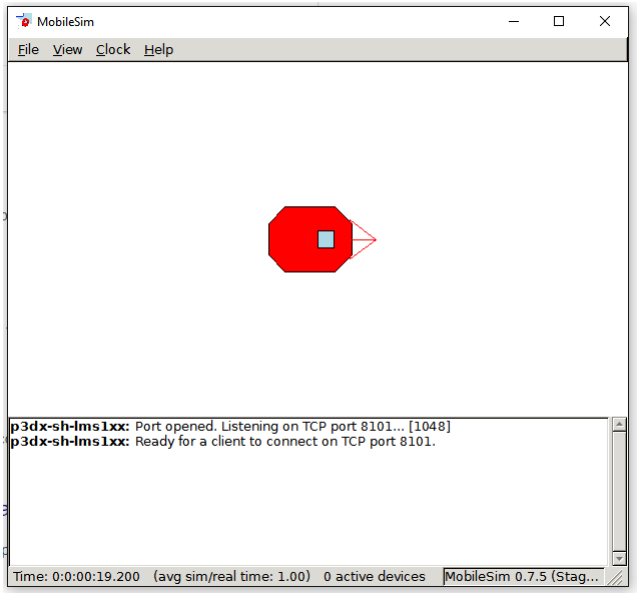



Figura 4.3: Plataforma virtual Aria

## 4.2. Visual Studio.

La programación como se había comentado se realizó en el lenguaje de programación C++ en el software Visual Studio 2010, este software se utilizó en un sistema operativo Windows 10. Así mismo, en este lenguaje de programación se realizaron subrutinas para desarrollar el esquema completo de control, como son una integración numérica Runge Kuta de orden 4, así mismo la función de transferencia hecha en simulink se implementó en código utilizando la rutina de integración. Dentro de sus librerías, Aria cuenta con ArSocket y ArNetWorking, estas librerías permiten realizar una comunicación en red con otras computadoras por medio del protocolo de comunicación TC/IP. Estas librerías fueron utilizadas para la programación de sockets y permitir así la comunicación dentro de una red LAN con dos computadoras. En la figura 4.4 se muestra el esquema de trabajo de Visual Studio 2010.

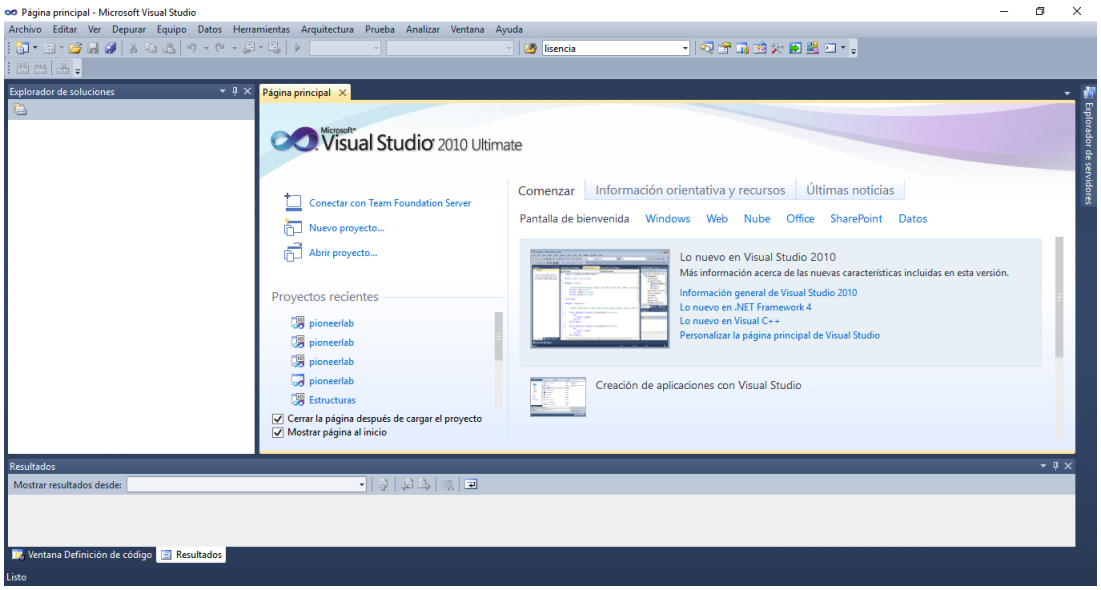



Figura 4.4: Ambiente de trabajo Visual Studio 2010

## 4.3. Comunicación.

En la figura 4.5 se muestra el esquema de comunicación del sistema de teleoperación. La computadora A se comunica con el robot maestro de forma inalámbrica utilizando el protocolo de comunicación TCP/IP de la librería de Aria, con un periodo de muestreo de 10 milisegundos. La computadora y el robot envían y reciben datos sobre una red local inalámbrica WLAN generada por el mismo robot. De forma equivalente la comunicación entre la computadora B y el robot esclavo es efectuada, trabajando con el mismo periodo de muestreo y sobre una red inalámbrica local generada por el robot esclavo. Las computadoras A y B se encuentran conectadas por medio de una red alámbrica local (LAN) las cuales se envían la información de cada robot con un periodo de muestreo de 10 ms. En la figura 4.6 se muestra el esquema de control con la división de que procesos realiza cada una de las computadoras, donde la computadora A realiza el control principal local en el sitio del robot maestro, y la computadora B es la encargada de realizar el control en el sitio remoto B del robot esclavo. Los retardos presentes en los experimentos se llevaron a cabo por medio de un buffer, para simular retardos aproximados en una comunicación por internet.

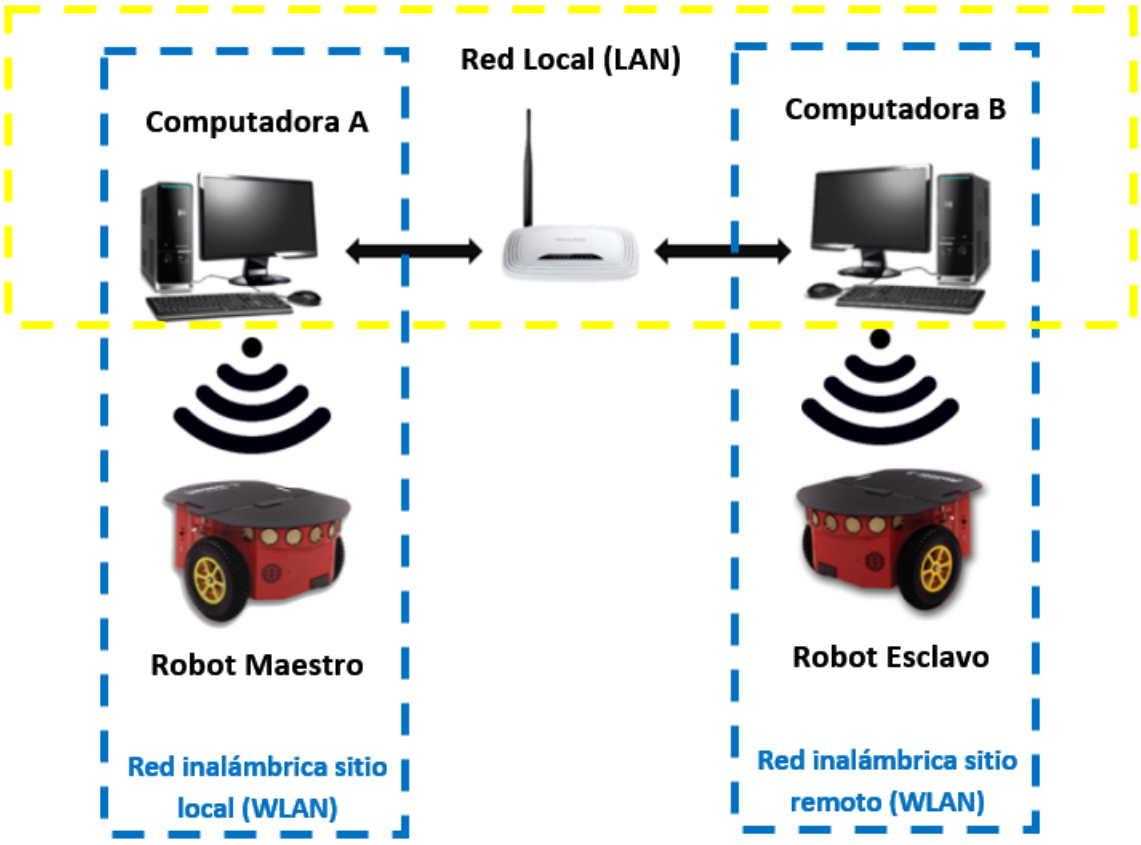



Figura 4.5: Esquema de comunicación.

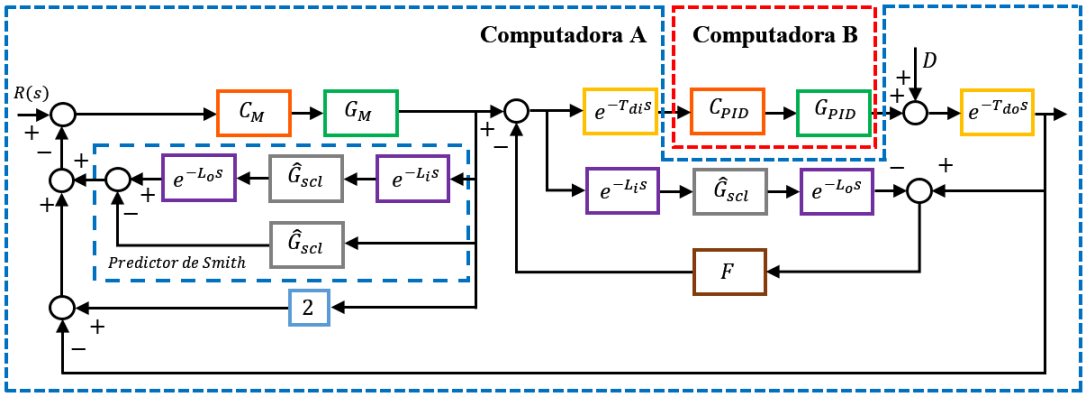



Figura 4.6: División del esquema de control por computadora

En la figura 4.7 se muestran los diferentes tipos de conexiones posibles que Aria nos permite realizar para comunicarnos con el robot Pioneer P3-DX. Estas son de forma alámbrica, inalámbrica, con una laptop a bordo o de forma autónoma con una computadora dentro del robot.

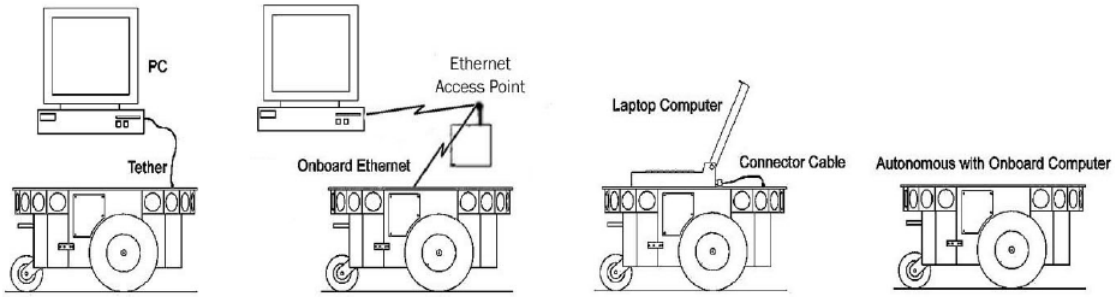



Figura 4.7: Esquemas de comunicación Aria

# Capítulo 5

# Experimentación

## 5.1. Valores de experimentación.

En esta sección se realizan las pruebas experimentales del controlador propuesto para el esquema de teleoperación bilateral de robots móviles diferenciales. Las pruebas se realizaron en un laboratorio sobre una superficie plana en un ambiente controlado. Con estos experimentos se verifica la compensación de los efectos de los retardos presentes en el sistema teleoperado. Los retardos agregados al sistema son generados por medio de un búffer en el algoritmo de control, los cuales se suponen constantes. En las pruebas experimentales se considera una trayectoria circular con un radio $r = 1\ m$, y con frecuencia $w = 0.2\ rad/s$, el tiempo de duración del experimento es de 60 segundos, con un paso de integración de 0.01 segundos. La tabla 5.1 muestra los valores utilizados para los experimentos del esquema de teleoperación, donde los valores de las ganancias del controlador $PID$ se obtuvieron de forma heurística.

| $\lambda$ | $T_0$ | $n$ | $r$ | $k_p$ | $k_i$ | $k_d$ |
|---|---|---|---|---|---|---|
| 0.25 | 0.25 | 3 | 3 | 1.3 | 0.001 | 3 |

Tabla 5.1: Valores de simulación

Las pruebas experimentales se clasifican de acuerdo al valor del retardo real en el sistema teleoperado. Como primer experimento se muestra el desempeño del esquema de control con respecto a un valor de 100 milisegundos y un retardo nominal de 100 milisegundos. Las gráficas de las trayectorias de los robots maestro y esclavo son presentadas así como los errores de seguimiento del robot maestro a la referencia y el robot esclavo al robot maestro. Así mismo se muestran las variables de control traslacional y rotacional.

## 5.2. Retardo real 100 ms

Como primer experimento se propone un retardo de 100 $ms$ en ambas direcciones del canal de comunicación, así como un retardo nominal de 100 $ms$. Se considera en este experimento que no existe incertidumbre en el valor del retardo real.

| $T_{di}$ | $T_{do}$ | $L_i$ | $L_o$ |
|---|---|---|---|
| 100 $ms$ | 100 $ms$ | 100 $ms$ | 100 $ms$ |

Tabla 5.2: Retardos experimento 1

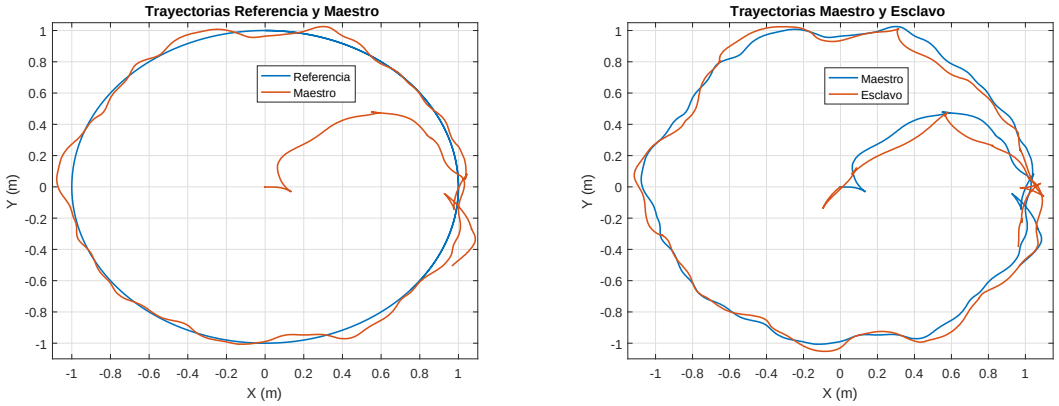



Figura 5.1: Trayectorias experimento 1

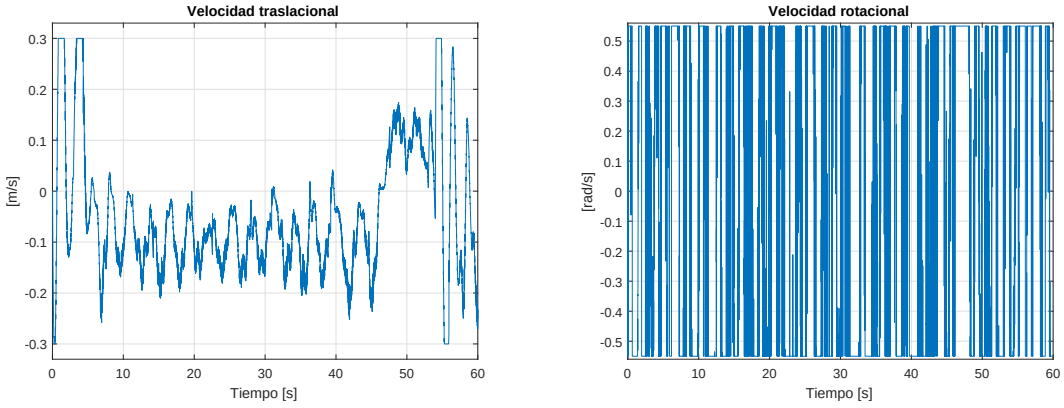



Figura 5.2: Variables de control experimento 1

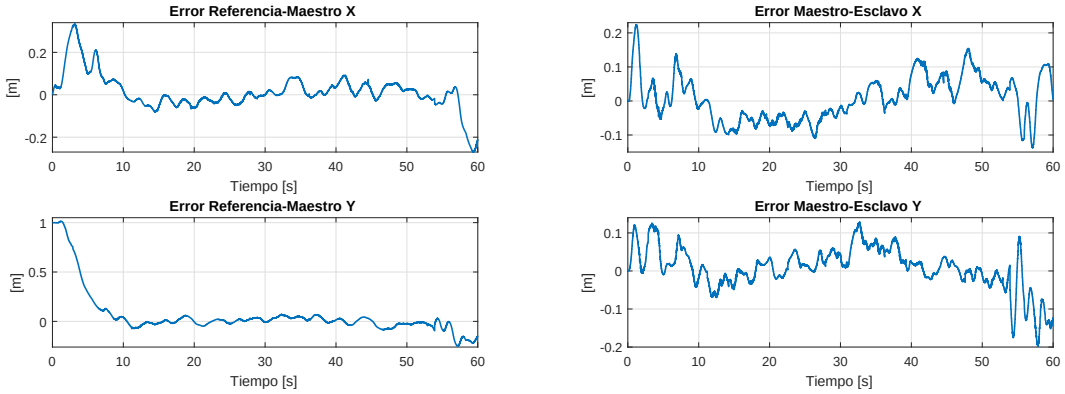



Figura 5.3: Errores de seguimiento experimento 1

Se muestra en la figura 5.1 las trayectorias de seguimiento por parte de los robots, tanto del robot maestro a la referencia y el robot esclavo al robot maestro. Se aprecia que a pesar de que las trayectoria del robot maestro no es tan suave, el robot esclavo es capaz de seguir al robot maestro sin desestabilizarse. En la figura 5.2 se muestran las variables de control traslacional $v$ y rotacional $w$ en el sitio remoto, es decir en el robot esclavo en la cual se aprecia que las variables de control se encuentran acotadas dentro de un rango, es así que la velocidad traslacional trabaja dentro de un valor $+/-$ 0.3 $m/s$ y la velocidad rotacional en un valor de $+/-$ 0.55 $rad/s$, esto debido a cuestiones de seguridad además de garantizar un mejor desempeño del robot debido que a mayores velocidades el robot excita dinámicas no modeladas por el modelo cinemático. En la figura 5.3 se muestran los errores de seguimiento tanto del robot maestro a la referencia como del robot esclavo al robot maestro. Se observa que los errores referencia-maestro y maestro-esclavo permanecen pequeños con valores dentro de un rango promedio de +/- 12 cm. En el seguimiento de la trayectoria del robot maestro a la referencia se aprecia en los primeros segundos un error de 1 m en la coordenada $y$ puesto que el robot maestro comienza el seguimiento en el centro del círculo mientras que la trayectoria de referencia comienza en la coordenada $(0, 1)$ lo que provoca este error y una acción de control que lo compense, haciendo mover al robot maestro de forma acelerada provocando el mismo efecto en el robot esclavo.

## 5.3. Retardo real 200 ms

En esta prueba se considera que existe incertidumbre en el retardo real, puesto que se considera un retardo nominal de 100 $ms$ y un retardo real de 200 $ms$. La incertidumbre se considera con un valor del 2 veces el valor del retardo nominal. El valor de retardo de 200 $ms$ es propuesto como un valor promedio en los retardos a través de internet.

| $T_{di}$ | $T_{do}$ | $L_i$ | $L_o$ |
|---|---|---|---|
| 200 $ms$ | 200 $ms$ | 100 $ms$ | 100 $ms$ |

Tabla 5.3: Retardos experimento 2

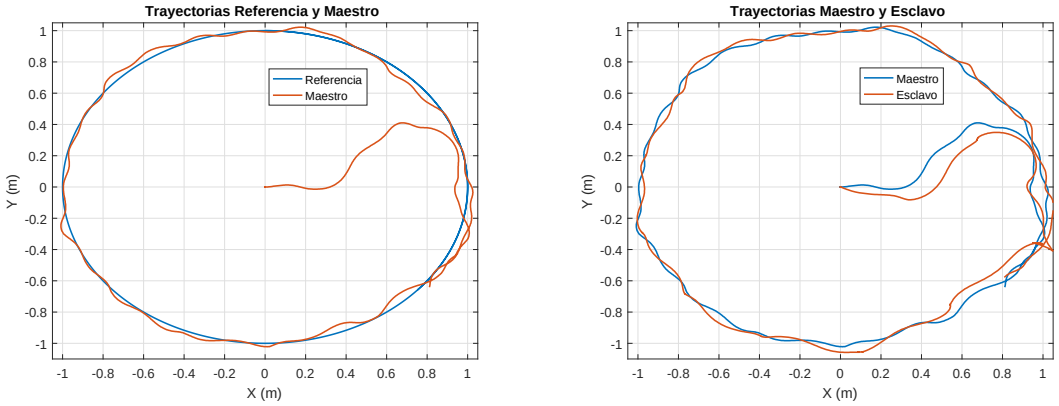



Figura 5.4: Trayectorias experimento 2

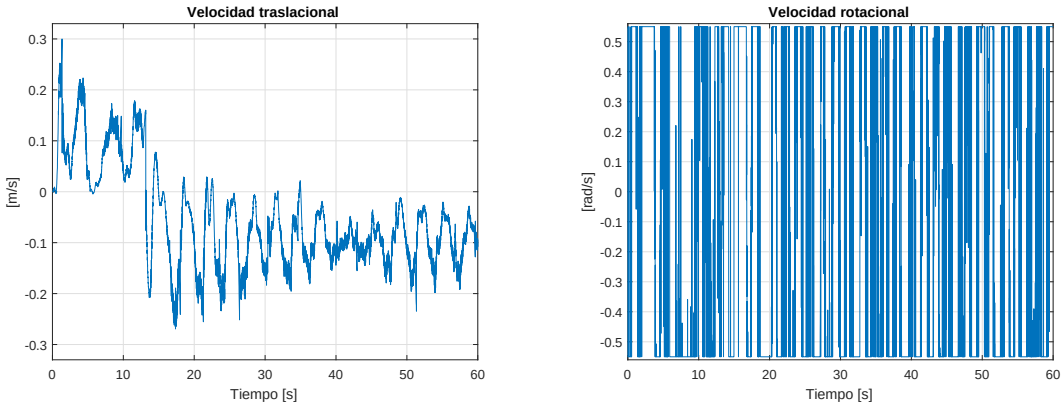



Figura 5.5: Variables de control experimento 2

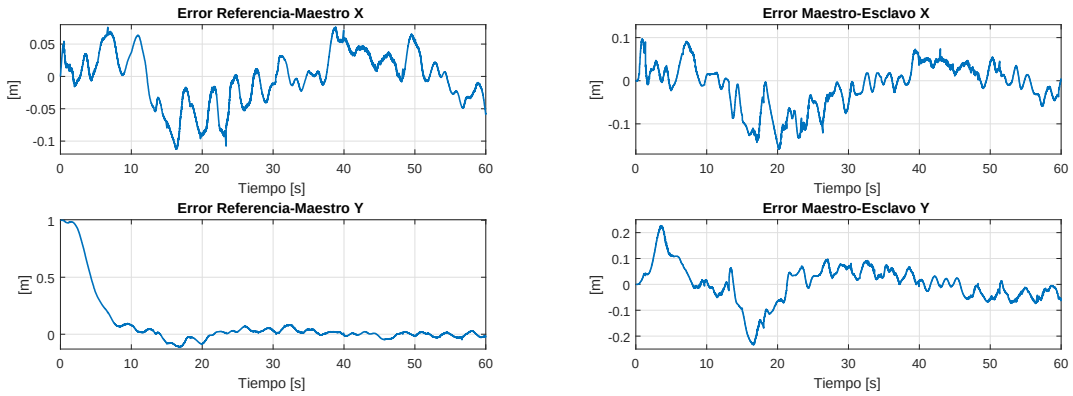



Figura 5.6: Errores de seguimiento experimento 2

En la figura 5.4 se muestran las trayectorias obtenidas tanto del robot maestro como el robot esclavo. La trayectoria de seguimiento entre la referencia y el robot maestro parece ser seguida con ruido sin embargo a diferencia de los resultados obtenidos en simulación, existen diferentes factores que afectan el desempeño del controlador provocando que tanto las trayectorias del robot maestro y del robot esclavo parezcan ruidosas. Sin embargo, el seguimiento es adecuado al considerar una incertidumbre del orden de dos veces la magnitud del retardo nominal. En la figura 5.5 se muestran las variables de control del robot $v$ y $w$ y en la figura 5.3 se muestran los errores de seguimiento entre la referencia y el robot maestro, así como del robot maestro y el robot esclavo. Ambos errores se encuentran acotados con rangos aproximados de $+/-$ 10 $cm$ con una tendencia a ir disminuyendo con el tiempo.

## 5.4. Retardo real 300 ms

En el siguiente experimento se aumenta el tiempo de retardo a 300 ms para observar la robustez del controlador bajo retardos de tiempo grandes, implicando una incertidumbre de tres veces el valor nominal del retardo de 100 $ms$.

| $T_{di}$ | $T_{do}$ | $L_i$ | $L_o$ |
|---|---|---|---|
| 300 $ms$ | 300 $ms$ | 100 $ms$ | 100 $ms$ |

Tabla 5.4: Retardos experimento 3

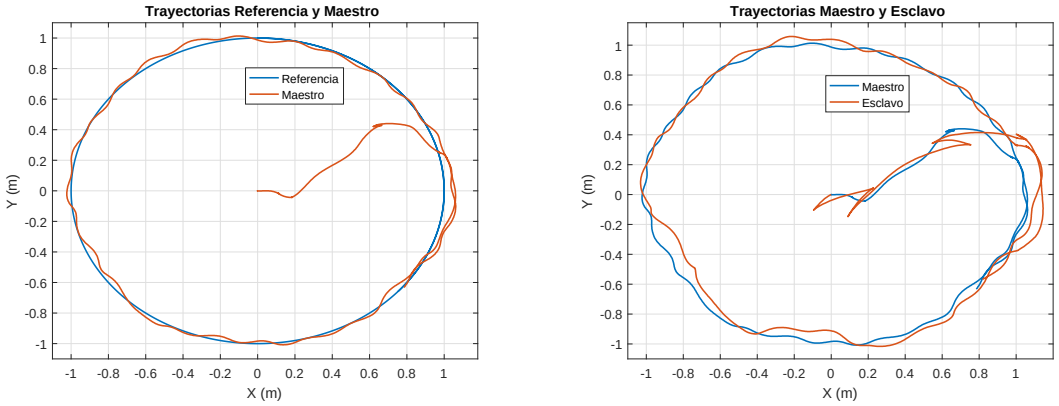



Figura 5.7: Trayectorias experimento 3

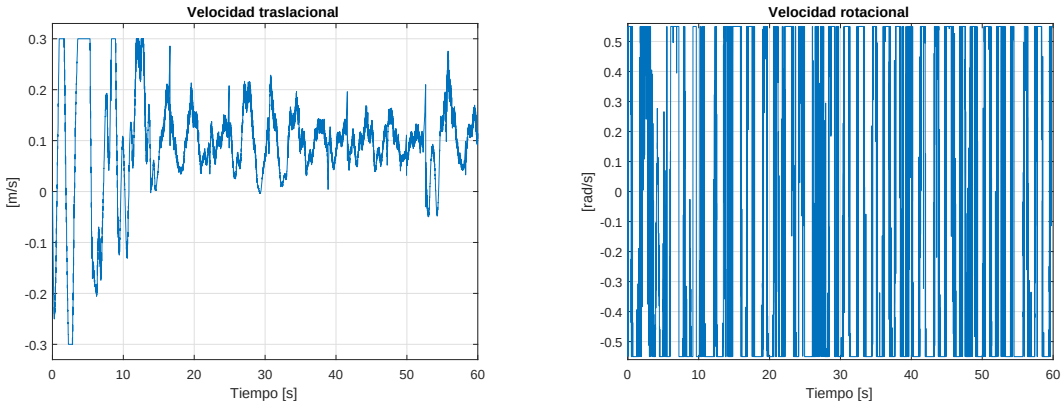



Figura 5.8: Variables de control experimento 3

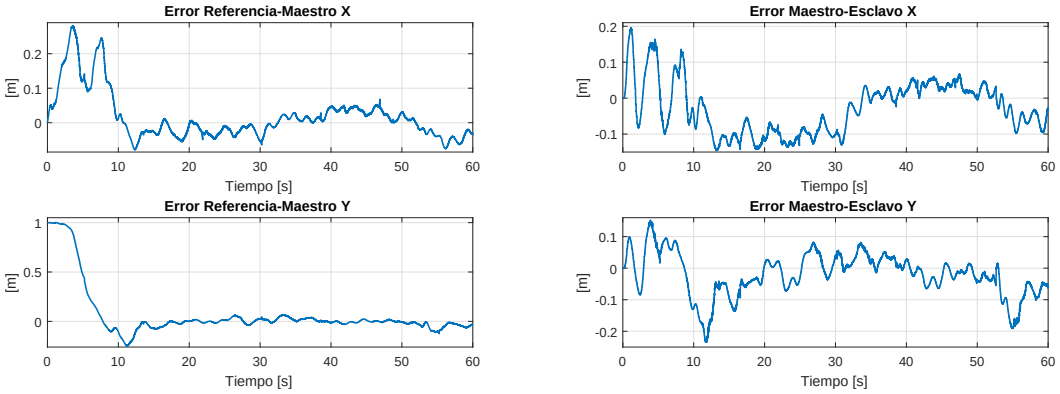



Figura 5.9: Errores de seguimiento experimento 3

Se puede apreciar en la figura 5.7 que el desempeño del sistema empeora a medida que el retardo va en aumento, esto quiere decir que el error entre el robot maestro y esclavo aumenta al incrementar el retardo. Así mismo, se observa el tiempo en el cual el robot maestro alcanza la trayectoria de referencia aumenta, mostrándose un sobreimpulso en los primeros segundos del experimento, esto a su vez afecta el seguimiento del robot esclavo al robot maestro, provocando un sobreimpulso aún mayor que el maestro con respecto a la referencia. Es evidente el deterioro del desempeño del controlador, al observar las variables de control en la figura 5.8 que la velocidad traslacional toma un tiempo mayor al asentarse sobre un valor constante, y como es de esperarse en la figura 5.9 los errores aumentan con respecto al experimento anterior, con errores hasta de 25 cm. Existen también factores que afectan el sistema, tal es el caso de los retardos presentados en la comunicación robot-computadora, ya que existen instantes en los cuales el robot no lee la información en el momento preciso. Esto provoca que el robot no reciba la información de control necesaria para ese instante de tiempo específico retardando la respuesta del sistema. Por otro lado en el tiempo de procesamiento de la rutina de control en ocasiones excede el tiempo de muestreo lo que provoca un retardo extra en el sistema.

-

# Capítulo 6

# Conclusiones y trabajo futuro

## 6.1. Conclusiones.

La implementación de un esquema de control por modelo interno en teleoperación sirve como compensador en los efectos de retardos en la comunicación. Se presenta a continuación un listado de puntos importantes por destacar en la realización de este trabajo.

1. El esquema de control propuesto compensa los efectos de los retardos en la comunicación, mejorando el desempeño de los sistemas teleoperados.

2. Los retardos pueden tener incertidumbre en el retardo de hasta 2 órdenes de magnitud o más entre el retardo real y el retardos nominal sin causar una evidente inestabilidad, este resultado implica que en cierta medida que el esquema de control propuesto es robusto frente a incertidumbres considerables en los retardos de comunicación.

3. El control por modelo interno representa una estrategia factible de control de robots móviles. Anteriormente en la literatura no se había abordado en robótica móvil el problema de retardos aplicando la estrategia de control por modelo interno. Lo cual lo hace una solución innovadora y atractiva respecto a los controles bastante utilizados.

4. Es posible realizar control por medio del modelo cinemático. El desempeño del controlador utilizando un modelo cinemático fue adecuado, incluso si existiesen dinámicas no modeladas.

5. Es posible compensar errores en modelado por medio de un controlador interno local PID, ayudando al esquema de control por modelo interno reduciendo el error entre la salida de la planta y el modelo.

## 6.2. Trabajo Futuro.

Como trabajo a futuro, queda un gran número de problemas que por tiempo no pudieron ser abordados sin embargo no por eso resta importancia a dichas actividades. A continuación se enumeran los puntos importantes a retomar por parte del autor y/o gente interesada en el tema.

1. La implementación de retardos variables en el tiempo, con lo cual se probaría más aún la eficacia del esquema de control contra retardos en la comunicación.

2. Experimentación a través del internet. Debido a cuestiones de seguridad de la institución no fue posible realizar este tipo de pruebas, sin embargo queda como un trabajo pendiente, lo que daría mayor realismo a las tareas.

3. La implementación de un sistema de posicionamiento absoluto es primordial para la verificación y validación del esquema de control, debido a que en este proyecto se utilizó odometría para determinar la posición del robot sin embargo existían errores por deslizamiento de las llantas.

4. Evasión de obstáculos, un tema interesante a abordar comprende la evasión de obstáculos tanto estáticos como dinámicos, en el cual la información de sensores de presencia es retroalimentada al sitio local.

5. Utilización del modelo dinámico. En este trabajo se realizó el diseño del control por medio del modelo cinemático es decir simplemente con ecuaciones de movimiento, dejando a tras dinámicas presentes en el sistema. Por lo cual es interesante realizar un control con mas valores e información respecto a los robots.

6. Incluir perturbaciones en el sistema teleoperado, debido a que obtención de la posición del robot era determinada de forma relativa al movimiento de las ruedas del robot, no fue posible incluir perturbaciones aditivas como una función escalón en el robot esclavo.

# Bibliografía


[1] Aynur Zakirov. Pioneer 3-dx mobile robot. Website.

[2] Adept Technology, Inc, 10 Columbia Drive, Amherst, NH 03031. *Pioneer 3-DX*, 2011. www.mobilerobots.com.

[3] G. Bermúdez. Robots móviles. teoría, aplicaciones y experiencias. In *Tecnura*, volume 5, pages 6–17. Universidad Distrital Francisco José de Caldas, June 2002.

[4] R. J. Anderson and M. W. Spong. Bilateral control of teleoperators with time delay. *IEEE Transactions on Automatic Control*, 34(5):494–501, May 1989.

[5] T.B. Sheridan. Telerobotics. *Automatica*, 25(4):487–507, 1989.

[6] R.C. Goertz. Electronically controlled manipulator. *Nucleonics*, 12(11):46–47, 1954.

[7] R.C. Goertz. Mechanical master?slave manipulator. *Nucleonics*, 12(11):45–46, 1954.

[8] T.B. Sheridan and W.R. Ferrell. Remote manipulative control with transmission delay. *IEEE Transactions on Human Factors in Electronics*, (4):25–29, 1963.

[9] E. Slawiñski, V. A. Mut, and J. F. Postigo. Bilateral teleoperation of mobile robots with delay. In *IEEE International Conference Mechatronics and Automation*, volume 3, pages 1672–1677, July 2005.

[10] H. Pérez-Villeda, F. Mirelez-Delgado, and A. Morales-Díaz. Reducción del efecto del retardo en el control de un robot móvil diferencial. In *Congreso Nacional de Control Automático 2015*, pages 376–381, October 2015.

[11] H. Sira-Ramírez, C. López-Uribe, and M. Velasco-Villa. Trajectory-tracking control of an input delayed omnidirectional mobile robot. In *2010 7th International Conference on Electrical Engineering Computing Science and Automatic Control*, pages 470–475, September 2010.

[12] M.R. Stojić, M.S. Matijević, and L.S. Draganović. A robust smith predictor modified by internal models for integrating process with dead time. *IEEE Transactions on Automatic Control*, 46(8):1293–1298, 2001.

[13] A. Denasi, D. Kostic, and H. Nijmeijer. Time delay compensation in bilateral teleoperations using impact. *IEEE Transactions on Control Systems Technology*, 21(3):704–715, May 2013.

[14] A. Denasi. *Teleoperated and cooperative robotics: a performance oriented control design*. PhD thesis, Eindhoven University of Technology, Eindhoven, Netherlands, 2015.

[15] R. Siegwart, I.R. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*, chapter 3. Massachusetts: The MIT Press, 2004.

[16] R. Orosco. *Modelado, análisis y control de vehículos articulados y robots móviles: teoría y experimentos*. PhD thesis, CINVESTAV, Ciudad de México, México, 2003.

[17] S. Sastry. *Nonlinear Systems: Analasys, Stability and Control*. New York: Springer-Verlag, 1999.

[18] E. Aranda, T. Salgado, and M. Velasco. Control no lineal discontinuo de un robot móvil. In *Computación y Sistemas*, pages 42–49. CIC-IPN, March 2002.

[19] G. Goodwin, S. Graebe, and M. Salgado. *Control System Design*. Upper Saddle River, 2001.

[20] M. Morari and Z. Evanghelos. *Robust process control*. Prentice-Hall, 1989.

[21] A. Denasi, D. Kostic, and H. Nijmeijer. An application of impact structure to bilateral teleoperations. In *49th IEEE Conference on Decision and Control (CDC)*, pages 1985–1990, December 2010.

[22] MobileRobots Inc. *Pioneer 3 Operations Manual*, January 2006.

# Apéndice A

# Artículos Publicados

A continuación se presentan los artículos realizados durante el desarrollo del presente trabajo de tesis.

- J. Vences Jiménez, A. Rodríguez-Ángeles y J. Álvarez Gallegos, *Control por modelo interno con compensación de retardos para robots móviles diferenciales en teleoperación*. MXIX Congreso Mexicano de Robótica COMRob 2017, que se llevó a cabo del 8 al 10 de Noviembre de 2017 en Mazatlán, Sinaloa. Nominación a premio Rafael Kelly categoría postgrado.

- Julio A. Vences-Jiménez, A. Rodríguez-Ángeles y J. Álvarez Gallegos, *Bilateral Time Delay Compensation in Bilateral Master Slave Teleoperation of Differential Mobile Robots Using IMC*. En The 13th IEEE Conference on Industrial Electronics and Applications (ICIEA 2018 ), que se llevará a cabo en Wuhan China, del día 31 de Mayo a 1 de Junio de 2018. (Estatus: enviado)

# Apéndice B

# Código control

Se muestra el código usado en el esquema de control teleoperado, el cual se realizó en Visual Studio 2010 en lenguaje estructurado C++.

## B.1. Computadora A

```
#include "Aria.h"
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <sstream>
#include <iomanip>
//#include "Control.h"
#include "Control_maestro.h"

using namespace std;

// Ganancias del controlador PID
//*
   double kpxM=0.7;
   double kixM=0.0001;
   double kdxM=2.2;
   double kpyM=0.7;
   double kiyM=0.0001;
   double kdyM=2.2;
   double lda=0.25;

   //*/

// Retardos en el sistema

    double r1=10;
```

```
double r2=10;
double r3=10;
double r4=10;
double r5=10;
double r6=10;

double repx;
double repy;
double remx;
double remy;
double xmp_r;
double ymp_r;

double xrep;
double yrep;
double xrem;
double yrem;

double xdrp[50000]={0};
double ydrp[50000]={0};
double xdrm[50000]={0};
double ydrm[50000]={0};
double exrp[50000]={0};
double eyrp[50000]={0};
double exrm[50000]={0};
double eyrm[50000]={0};
double xeps[50000]={0};
double yeps[50000]={0};
double xsps[50000]={0};
double ysps[50000]={0};

// Valores de simulación

    double t=0;
double tc=0;
    double tf=120;
    double h=0.010;
   int i=1;
  long msi,msf;

// Valores iniciales coordenadas Robot

    double x=0;
    double y=0;
```

```
    double th=0;
double Vt, Vr;

double xs=0;
    double ys=0;
    double ths=0;

double xm=0;
    double ym=0;
    double thm=0;

// Valores iniciales para función control_maestro

double s1M=0;
double s2M=0;
    double ixaM=0;
    double iyaM=0;
    double epxM=0;
    double epyM=0;

double upxM=0;
    double upyM=0;
    double duxM=0;
    double duyM=0;

    double dw1M=0;
    double dvM=0;
    double dwM=0;
    double vbM=0;
    double wbM=0;
    double zbM=0.00001;
    double w1bM=0;
double vmm=0;
double wmm=0;
double W=0;
double L=190;
double wrmm=0;
double wlmm=0;

double xbM[5]={0};
double ybM[5]={0};

// Filtro principio de modelo interno
```

```
double xb2[3]={0};
double yb2[3]={0};
double upxf=0;
    double upyf=0;
    double duxf=0;
    double duyf=0;
double efx=0;
double efy=0;

// Trayectoria deseada

double xr;
double yr;

double cx=0;
double cy=0;
double ax=1;
double ay=2;

// Errores del sistema

double ex;
double ey;
double ex_m;
double ey_m;
double epmx=0;
double epmy=0;

double Emex=0;
double Emey=0;
double Ep1x=0;
double Ep1y=0;
double xmpr=0;
double ympr=0;
double xmp=0;
double ymp=0;
double Ep2x=0;
double Ep2y=0;
double Egx=0;
double Egy=0;

// Envío y recepción de datos

char *ptrToken;
```

```
int comu=3;
////////////////////////////////////////////////
            /*     Clase modelo       */
////////////////////////////////////////////////
class Modelo{
public:

// Funciones miembro
Modelo();
void funcion_modelo();
void imprime();
void control(double xr,double yr, int i, double h);
void inversax(double x, double h);
void inversay(double y, double h);
double saturacion(double a, double b);
void integracion(double h);

// Datos miembro
double x, y, th;
double v, w;

private:

// Datos miembro
double s1, s2;
double ex, ey;
double ixa, iya;
double epx, epy;
double upx, upy;
double dux, duy;
double dw1;
double dv, dw;
double vb, wb;
double zb;
double w1b;
double xb[5], yb[5];
double kpxm, kixm, kdxm;
double kpym, kiym, kdym;

};
Modelo::Modelo(){
x=0; y=0; th=0; s1=0; s2=0; ixa=0; iya=0; epx=0;
 epy=0; upx=0; upy=0; dux=0; duy=0; dw1=0; v=0;
 w=0; dv=0; dw=0; vb=0; wb=0;
```

```
zb=0.00001; w1b=0; xb[0]=0; xb[1]=0; xb[2]=0;
 xb[3]=0; xb[4]=0; yb[0]=0; yb[1]=0; yb[2]=0;
 yb[3]=0; yb[4]=0;
kpxm=0.9; kixm=0.0001; kdxm=20; kpym=0.9;
 kiym=0.0001; kdym=20;}

// Funcion_modelo
void Modelo::funcion_modelo(){
x=x+1;
}

// Imprime x
void Modelo::imprime(){
cout<<"x= "<<x<<endl;
cout<<"y= "<<y<<endl;
cout<<"th= "<<th<<endl;
cout<<endl;
}

// Control Modelo
void Modelo::control(double xr,double yr, int i, double h){
// Error referencia
ex=xr;
ey=yr;

// Delta error
if (i>=2){
dux=ex-upx;
duy=ey-upy;
}


        // Inversa Modelo
inversax(ex, h); // 1.21
inversay(ey, h); // 1.216

    // Errores de seguimiento PID 2
double ex2=s1-x;
double ey2=s2-y;

    // Integral del error
double iex=ixa+h*(ex2+epx)/2;
double iey=iya+h*(ey2+epy)/2;
```

```
// Derivada del error
double dex=(ex2-epx)/h;
double dey=(ey2-epy)/h;

// Control PID 2
double ux=kpxm*ex2+kixm*iex+kdxm*dex;
double uy=kpym*ey2+kiym*iey+kdym*dey;

// Linealizador
double xl3=th;
double w1=cos(xl3)*ux+sin(xl3)*uy;
double w2=-(sin(xl3)/zb)*ux+(cos(xl3)/zb)*uy;

            if (i>=2){
                dw1=w1-w1b;
}

// Integración de z
double k1=h*w1;
double k2=h*(w1+dw1/2);
double k3=h*(w1+dw1/2);
double k4=h*(w1+dw1);
double z=zb+(k1+2*k2+2*k3+k4)/6;

// No cruce por cero
if(zb>z && z<0.00001 && z>-0.00001){
z=-0.00001;}
else if(zb<z && z<0.00001 && z>-0.00001){
z=0.00001;}

// Saturación del control
if (i%1==0){
v=saturacion(z,0.3); // 0.3
w=saturacion(w2,0.55);} // 0.3
else{
v=vb;
w=wb;}

// Integración de las velocidades del sistema
            if (i>=2){
                dw=w-wb;
dv=v-vb;}

// Integración de coordenadas del modelo
```

```
           integracion(h);

// Acturalización de datos

upx=ex;
upy=ey;
ixa=iex;
iya=iey;
epx=ex2;
epy=ey2;
w1b=w1;
wb=w;
vb=v;
zb=z;
   }

// Inversa en x
void Modelo::inversax(double a, double h){

double xb11=xb[0];
double xb12=xb[1];
double xb13=xb[2];
double xb14=xb[3];
double xb15=xb[4];

   // Constantes para x
double a1x=((lda*lda*lda)*kpxm+3*(lda*lda)*kpxm*kdxm)
/((lda*lda*lda)*kpxm*kdxm);
double a2x=((lda*lda*lda)*kpxm*kixm
+3*(lda*lda)*kpxm+3*lda*kpxm*kdxm)/((lda*lda*lda)*kpxm*kdxm);
double a3x=(3*(lda*lda)*kpxm*kixm
+3*lda*kpxm+kpxm*kdxm)/((lda*lda*lda)*kpxm*kdxm);
double a4x=(3*lda*kpxm*kixm+kpxm)/((lda*lda*lda)*kpxm*kdxm);
double a5x=kixm/((lda*lda*lda)*kdxm);

double b1x=3/((lda*lda)*kpxm*kdxm);
double b2x=(3*lda*kpxm*kdxm+1)/((lda*lda*lda)*kpxm*kdxm);
double b3x=(kpxm*kdxm+3*lda*kpxm)/((lda*lda*lda)*kpxm*kdxm);
double b4x=(3*lda*kpxm*kixm+kpxm)/((lda*lda*lda)*kpxm*kdxm);
double b5x=kixm/((lda*lda*lda)*kdxm);

double  ux=a;

//  Integración x por método Runge Kutta 4 orden
```

```
double k1x1=h*xb12;
double k1x2=h*xb13;
double k1x3=h*xb14;
double k1x4=h*xb15;
double k1x5=h*(-a5x*xb11-a4x*xb12-a3x*xb13
-a2x*xb14-a1x*xb15+ux);

double k2x1=h*(xb12+k1x2/2);
double k2x2=h*(xb13+k1x3/2);
double k2x3=h*(xb14+k1x4/2);
double k2x4=h*(xb15+k1x5/2);
double k2x5=h*(-a5x*(xb11+k1x1/2)-a4x*(xb12+k1x2/2)
-a3x*(xb13+k1x3/2)-a2x*(xb14+k1x4/2)
-a1x*(xb15+k1x5/2)+(ux+dux/2));

double k3x1=h*(xb12+k2x2/2);
double k3x2=h*(xb13+k2x3/2);
double k3x3=h*(xb14+k2x4/2);
double k3x4=h*(xb15+k2x5/2);
double k3x5=h*(-a5x*(xb11+k2x1/2)-a4x*(xb12+k2x2/2
)-a3x*(xb13+k2x3/2)-a2x*(xb14+k2x4/2)-a1x*(xb15+k2x5/2)+(ux+dux/2));

double k4x1=h*(xb12+k3x2);
double k4x2=h*(xb13+k3x3);
double k4x3=h*(xb14+k3x4);
double k4x4=h*(xb15+k3x5);
double k4x5=h*(-a5x*(xb11+k3x1)-a4x*(xb12+k3x2)
-a3x*(xb13+k3x3)-a2x*(xb14+k3x4)-a1x*(xb15+k3x5)+(ux+dux));

double x1=xb11+(k1x1+2*k2x1+2*k3x1+k4x1)/6;
double x2=xb12+(k1x2+2*k2x2+2*k3x2+k4x2)/6;
double x3=xb13+(k1x3+2*k2x3+2*k3x3+k4x3)/6;
double x4=xb14+(k1x4+2*k2x4+2*k3x4+k4x4)/6;
double x5=xb15+(k1x5+2*k2x5+2*k3x5+k4x5)/6;

s1=(b5x)*x1+(b4x)*x2+(b3x)*x3+(b2x)*x4+(b1x)*x5;

xb[0]=x1;
xb[1]=x2;
xb[2]=x3;
xb[3]=x4;
xb[4]=x5;

}
```

```
// Inversa en y
void Modelo::inversay(double a, double h){

double yb11=yb[0];
double yb12=yb[1];
double yb13=yb[2];
double yb14=yb[3];
double yb15=yb[4];

// Constantes para y
double a1y=((lda*lda*lda)*kpym+3*(lda*lda)*kpym*kdym)
/((lda*lda*lda)*kpym*kdym);
double a2y=((lda*lda*lda)*kpym*kiym
+3*(lda*lda)*kpym+3*lda*kpym*kdym)/((lda*lda*lda)*kpym*kdym);
double a3y=(3*(lda*lda)*kpym*kiym
+3*lda*kpym+kpym*kdym)/((lda*lda*lda)*kpym*kdym);
double a4y=(3*lda*kpym*kiym+kpym)/((lda*lda*lda)*kpym*kdym);
double a5y=kiym/((lda*lda*lda)*kdym);

double b1y=3/((lda*lda)*kpym*kdym);
double b2y=(3*lda*kpym*kdym+1)/((lda*lda*lda)*kpym*kdym);
double b3y=(kpym*kdym+3*lda*kpym)/((lda*lda*lda)*kpym*kdym);
double b4y=(3*lda*kpym*kiym+kpym)/((lda*lda*lda)*kpym*kdym);
double b5y=kiym/((lda*lda*lda)*kdym);

double  uy=a;
//  Integración y por método Runge Kutta 4 orden
double g1x1=h*yb12;
double g1x2=h*yb13;
double g1x3=h*yb14;
double g1x4=h*yb15;
double g1x5=h*(-a5y*yb11-a4y*yb12-a3y*yb13
-a2y*yb14-a1y*yb15+uy);

double g2x1=h*(yb12+g1x2/2);
double g2x2=h*(yb13+g1x3/2);
double g2x3=h*(yb14+g1x4/2);
double g2x4=h*(yb15+g1x5/2);
double g2x5=h*(-a5y*(yb11+g1x1/2)-a4y*(yb12+g1x2/2)
-a3y*(yb13+g1x3/2)-a2y*(yb14+g1x4/2)-a1y*(yb15+g1x5/2)
+(uy+duy/2));

double g3x1=h*(yb12+g2x2/2);
```

```
double g3x2=h*(yb13+g2x3/2);
double g3x3=h*(yb14+g2x4/2);
double g3x4=h*(yb15+g2x5/2);
double g3x5=h*(-a5y*(yb11+g2x1/2)-a4y*(yb12+g2x2/2)
-a3y*(yb13+g2x3/2)-a2y*(yb14+g2x4/2)-a1y*(yb15+g2x5/2)+(uy+duy/2));

double g4x1=h*(yb12+g3x2);
double g4x2=h*(yb13+g3x3);
double g4x3=h*(yb14+g3x4);
double g4x4=h*(yb15+g3x5);
double g4x5=h*(-a5y*(yb11+g3x1)-a4y*(yb12+g3x2
)-a3y*(yb13+g3x3)-a2y*(yb14+g3x4)-a1y*(yb15+g3x5)+(uy+duy));


double f1=yb11+(g1x1+2*g2x1+2*g3x1+g4x1)/6;
double f2=yb12+(g1x2+2*g2x2+2*g3x2+g4x2)/6;
double f3=yb13+(g1x3+2*g2x3+2*g3x3+g4x3)/6;
double f4=yb14+(g1x4+2*g2x4+2*g3x4+g4x4)/6;
double f5=yb15+(g1x5+2*g2x5+2*g3x5+g4x5)/6;

s2=(b5y)*f1+(b4y)*f2+(b3y)*f3+(b2y)*f4+(b1y)*f5;


yb[0]=f1;
yb[1]=f2;
yb[2]=f3;
yb[3]=f4;
yb[4]=f5;


}

//Saturación
double Modelo::saturacion(double a, double b){
    if (a>b)
{return b; }
else if (a<b)
{return -b;}
    else
{ return a;}
}

//Integración
void Modelo::integracion(double h){
```

```
/*  Esta función integra las velocidades del
robot móvil 2.0 xdot, ydot y thdot, por el
método Runge Kutta de
orden 4, obteniendo x, y  y th */
     double xb=x;
     double yb=y;
     double thb=th;

     double k1t=h*w;
     double k2t=h*(w+dw/2);
     double k3t=h*(w+dw/2);
     double k4t=h*(w+dw);
     th=thb+(k1t+2*k2t+2*k3t+k4t)/6;

     double k1x=h*(v*cos(thb));
     double k2x=h*((v+dv/2)*cos(thb+k1t/2));
     double k3x=h*((v+dv/2)*cos(thb+k2t/2));
     double k4x=h*((v+dv)*cos(thb+k3t));
     x=xb+(k1x+2*k2x+2*k3x+k4x)/6;

     double k1y=h*(v*sin(thb));
     double k2y=h*((v+dv/2)*sin(thb+k1t/2));
     double k3y=h*((v+dv/2)*sin(thb+k2t/2));
     double k4y=h*((v+dv)*sin(thb+k3t));
     y=yb+(k1y+2*k2y+2*k3y+k4y)/6;

}

// Definición de clases
Modelo Mod1, Mod2, Mod3;

////////////////////////////////////////////////////
/*   Clase Robot    */
////////////////////////////////////////////////////
class Robot{
public:

// Funciones miembro
Robot();
void control(double xr,double yr, int i, double h);
void inversax(double x, double h);
void inversay(double y, double h);
double saturacion(double a, double b, double c);
```

```
// Datos miembro
//double x, y, th;
double v, w;
double W, wmm, vmm;
private:

// Datos miembro
double s1, s2;
double ex, ey;
double ixa, iya;
double epx, epy;
double upx, upy;
double dux, duy;
double dw1;
double dv, dw;
double vb, wb;
double zb;
double w1b;
double xb[5], yb[5];
double kpxm, kixm, kdxm;
double kpym, kiym, kdym;

};
Robot::Robot(){
x=0; y=0; th=0; s1=0; s2=0; ixa=0; iya=0;
epx=0; epy=0; upx=0; upy=0; dux=0; duy=0;
dw1=0; v=0; w=0; dv=0; dw=0; vb=0; wb=0;
zb=0.00001; w1b=0; xb[0]=0; xb[1]=0; xb[2]=0;
 xb[3]=0; xb[4]=0; yb[0]=0; yb[1]=0; yb[2]=0;
  yb[3]=0; yb[4]=0;
kpxm=1; kixm=0.0001; kdxm=2.9; kpym=1;
 kiym=0.0001; kdym=2.9;}

// Control Modelo
void Robot::control(double xr,double yr, int i, double h){
// Error referencia
ex=xr;
ey=yr;

// Delta error
if (i>=2){
dux=ex-upx;
duy=ey-upy;
}
```

```
        // Inversa Modelo
inversax(1.01*ex, h); // 1.21
inversay(1.01*ey, h); // 1.216

    // Errores de seguimiento PID 2
double ex2=s1-x;
double ey2=s2-y;

    // Integral del error
double iex=ixa+h*(ex2+epx)/2;
double iey=iya+h*(ey2+epy)/2;

// Derivada del error
double dex=(ex2-epx)/h;
double dey=(ey2-epy)/h;

// Control PID 2
double ux=kpxm*ex2+kixm*iex+kdxm*dex;
double uy=kpym*ey2+kiym*iey+kdym*dey;

// Linealizador
double xl3=th;
double w1=cos(xl3)*ux+sin(xl3)*uy;
double w2=-(sin(xl3)/zb)*ux+(cos(xl3)/zb)*uy;

            if (i>=2){
                dw1=w1-w1b;
}

// Integración de z
double k1=h*w1;
double k2=h*(w1+dw1/2);
double k3=h*(w1+dw1/2);
double k4=h*(w1+dw1);
double z=zb+(k1+2*k2+2*k3+k4)/6;

// No cruce por cero
if(zb>z && z<0.00001 && z>-0.00001){
z=-0.00001;}
else if(zb<z && z<0.00001 && z>-0.00001){
z=0.00001;}
```

```
// Saturación del control
if (i%1==0){
v=saturacion(z,0.3,-0.3); // 0.3
w=saturacion(w2,0.55,-0.55);} // 0.3
else{
v=vb;
w=wb;}

 W=w;
 wmm=w*180/3.141516;
     vmm=v*double (1000.0);


// Acturalización de datos

upx=ex;
upy=ey;
ixa=iex;
iya=iey;
epx=ex2;
epy=ey2;
w1b=w1;
wb=w;
vb=v;
zb=z;
   }

// Inversa en x
void Robot::inversax(double a, double h){

double xb11=xb[0];
double xb12=xb[1];
double xb13=xb[2];
double xb14=xb[3];
double xb15=xb[4];

   // Constantes para x
double a1x=((lda*lda*lda)*kpxm+3*(lda*lda)*kpxm*kdxm)
/((lda*lda*lda)*kpxm*kdxm);
double a2x=((lda*lda*lda)*kpxm*kixm+3*(lda*lda)*kpxm
+3*lda*kpxm*kdxm)/((lda*lda*lda)*kpxm*kdxm);
double a3x=(3*(lda*lda)*kpxm*kixm
+3*lda*kpxm+kpxm*kdxm)/((lda*lda*lda)*kpxm*kdxm);
double a4x=(3*lda*kpxm*kixm+kpxm)
```

```
//((lda*lda*lda)*kpxm*kdxm);
double a5x=kixm/((lda*lda*lda)*kdxm);

double b1x=3/((lda*lda)*kpxm*kdxm);
double b2x=(3*lda*kpxm*kdxm+1)
//((lda*lda*lda)*kpxm*kdxm);
double b3x=(kpxm*kdxm+3*lda*kpxm)
//((lda*lda*lda)*kpxm*kdxm);
double b4x=(3*lda*kpxm*kixm+kpxm)
//((lda*lda*lda)*kpxm*kdxm);
double b5x=kixm/((lda*lda*lda)*kdxm);

double  ux=a;

//  Integración x por método Runge Kutta 4 orden
double k1x1=h*xb12;
double k1x2=h*xb13;
double k1x3=h*xb14;
double k1x4=h*xb15;
double k1x5=h*(-a5x*xb11-a4x*xb12
-a3x*xb13-a2x*xb14-a1x*xb15+ux);

double k2x1=h*(xb12+k1x2/2);
double k2x2=h*(xb13+k1x3/2);
double k2x3=h*(xb14+k1x4/2);
double k2x4=h*(xb15+k1x5/2);
double k2x5=h*(-a5x*(xb11+k1x1/2)-a4x*(xb12+k1x2/2)
-a3x*(xb13+k1x3/2)-a2x*(xb14+k1x4/2)-a1x*(xb15+k1x5/2)+(ux+dux/2));

double k3x1=h*(xb12+k2x2/2);
double k3x2=h*(xb13+k2x3/2);
double k3x3=h*(xb14+k2x4/2);
double k3x4=h*(xb15+k2x5/2);
double k3x5=h*(-a5x*(xb11+k2x1/2)-a4x*(xb12+k2x2/2)
-a3x*(xb13+k2x3/2)-a2x*(xb14+k2x4/2)-a1x*(xb15+k2x5/2)+(ux+dux/2));

double k4x1=h*(xb12+k3x2);
double k4x2=h*(xb13+k3x3);
double k4x3=h*(xb14+k3x4);
double k4x4=h*(xb15+k3x5);
double k4x5=h*(-a5x*(xb11+k3x1)-a4x*(xb12+k3x2)
-a3x*(xb13+k3x3)-a2x*(xb14+k3x4)-a1x*(xb15+k3x5)
+(ux+dux));
```

```
double x1=xb11+(k1x1+2*k2x1+2*k3x1+k4x1)/6;
double x2=xb12+(k1x2+2*k2x2+2*k3x2+k4x2)/6;
double x3=xb13+(k1x3+2*k2x3+2*k3x3+k4x3)/6;
double x4=xb14+(k1x4+2*k2x4+2*k3x4+k4x4)/6;
double x5=xb15+(k1x5+2*k2x5+2*k3x5+k4x5)/6;

s1=(b5x)*x1+(b4x)*x2+(b3x)*x3+(b2x)*x4
+(b1x)*x5;

xb[0]=x1;
xb[1]=x2;
xb[2]=x3;
xb[3]=x4;
xb[4]=x5;

}

// Inversa en y
void Robot::inversay(double a, double h){

double yb11=yb[0];
double yb12=yb[1];
double yb13=yb[2];
double yb14=yb[3];
double yb15=yb[4];

// Constantes para y
double a1y=((lda*lda*lda)*kpym
+3*(lda*lda)*kpym*kdym)/((lda*lda*lda)*kpym*kdym);
double a2y=((lda*lda*lda)*kpym*kiym
+3*(lda*lda)*kpym+3*lda*kpym*kdym)/((lda*lda*lda)*kpym*kdym);
double a3y=(3*(lda*lda)*kpym*kiym
+3*lda*kpym+kpym*kdym)/((lda*lda*lda)*kpym*kdym);
double a4y=(3*lda*kpym*kiym+kpym)/((lda*lda*lda)*kpym*kdym);
double a5y=kiym/((lda*lda*lda)*kdym);

double b1y=3/((lda*lda)*kpym*kdym);
double b2y=(3*lda*kpym*kdym+1)
/((lda*lda*lda)*kpym*kdym);
double b3y=(kpym*kdym+3*lda*kpym)
/((lda*lda*lda)*kpym*kdym);
double b4y=(3*lda*kpym*kiym+kpym)
/((lda*lda*lda)*kpym*kdym);
double b5y=kiym/((lda*lda*lda)*kdym);
```

```
double  uy=a;
//  Integración y por método Runge Kutta 4 orden
double g1x1=h*yb12;
double g1x2=h*yb13;
double g1x3=h*yb14;
double g1x4=h*yb15;
double g1x5=h*(-a5y*yb11-a4y*yb12-a3y*yb13
-a2y*yb14-a1y*yb15+uy);

double g2x1=h*(yb12+g1x2/2);
double g2x2=h*(yb13+g1x3/2);
double g2x3=h*(yb14+g1x4/2);
double g2x4=h*(yb15+g1x5/2);
double g2x5=h*(-a5y*(yb11+g1x1/2)-a4y*(yb12+g1x2/2
)-a3y*(yb13+g1x3/2)-a2y*(yb14+g1x4/2)-a1y*(yb15+g1x5/2)+(uy+duy/2));

double g3x1=h*(yb12+g2x2/2);
double g3x2=h*(yb13+g2x3/2);
double g3x3=h*(yb14+g2x4/2);
double g3x4=h*(yb15+g2x5/2);
double g3x5=h*(-a5y*(yb11+g2x1/2)-a4y*(yb12+g2x2/2)
-a3y*(yb13+g2x3/2)-a2y*(yb14+g2x4/2)-a1y*(yb15+g2x5/2)+(uy+duy/2));

double g4x1=h*(yb12+g3x2);
double g4x2=h*(yb13+g3x3);
double g4x3=h*(yb14+g3x4);
double g4x4=h*(yb15+g3x5);
double g4x5=h*(-a5y*(yb11+g3x1)-a4y*(yb12+g3x2)
-a3y*(yb13+g3x3)-a2y*(yb14+g3x4)-a1y*(yb15+g3x5)+(uy+duy));


double f1=yb11+(g1x1+2*g2x1+2*g3x1+g4x1)/6;
double f2=yb12+(g1x2+2*g2x2+2*g3x2+g4x2)/6;
double f3=yb13+(g1x3+2*g2x3+2*g3x3+g4x3)/6;
double f4=yb14+(g1x4+2*g2x4+2*g3x4+g4x4)/6;
double f5=yb15+(g1x5+2*g2x5+2*g3x5+g4x5)/6;

s2=(b5y)*f1+(b4y)*f2+(b3y)*f3+(b2y)*f4+(b1y)*f5;


yb[0]=f1;        // Relación si no está es negativa
yb[1]=f2;   // Relación
yb[2]=f3;   // Relación
```

```
yb[3]=f4;   // Relación
yb[4]=f5;   // Relación


}

//Saturación
double Robot::saturacion(double a, double b, double c){
    if (a>b)
{return b; }
else if (a<c)
{return c;}
    else
{ return a;}
}

// Definición de clases
Robot R1;
////////////////////////////////////////////////
                      /*  Inicio main  */
////////////////////////////////////////////////

int main (int argc, char** argv){

ofstream rtraydata1("Referencia.txt");
ofstream rtraydata2("Robot_Maestro.txt");
ofstream rtraydata3("Robot_Esclavo.txt");
ofstream rtraydata4("Modelo_Esclavo.txt");
ofstream rtraydata5("Retardo_Envio.txt");
ofstream rtraydata6("Retardo_Recepcion.txt");
ofstream rtraydata7("Error_general.txt");
ofstream rtraydata8("Error_planta_modelo.txt");
ofstream rtraydata9("Filtro.txt");
ofstream rtraydata10("Control_Maestro.txt");
ofstream rtraydata11("Salida_Inversa_Maestro.txt");
ofstream rtraydata12("Modelo_PS_retardo.txt");
ofstream rtraydata13("Modelo_PS_sin_retardo.txt");
ofstream rtraydata14("Errores.txt");
ofstream rtraydata15("Modelo_3.txt");


//ofstream rtraydata9("Control_modelo.txt");
/*
ofstream rtraydata8("error.txt");
```

```
ofstream rtraydata9("error_planta.txt");
ofstream rtraydata10("referencia.txt");
//*/



////////////////////////////////////////////
           /*  Comunicación TPC/IP Sockets en Aria  */
////////////////////////////////////////////

// The string to send to the server.
const char *strToSend="Hello Server";
// The buffer in which to recieve the hello from the server
char buff[2];
char *bufer;
// The size of the string the server sent
size_t strSize;
// The socket object
ArSocket sock;
// Initialize Aria.  It is especially important to do
// this on Windows, because it will initialize Window's
// sockets system.
Aria::init();
// Connect to the server
ArLog::log(ArLog::Normal, "socketClientExample:
Connecting to Server TCP port 7777...");
if (sock.connect("192.168.0.103", 7777, ArSocket::TCP))
ArLog::log(ArLog::Normal, "socketClientExample:
Connected to server at localhost TCP port 7777.");
else
{
ArLog::log(ArLog::Terse, "socketClientExample:
Error connecting to server at localhost TCP port 7777:
 %s", sock.getErrorStr().c_str());
return(-1);
}
// Sleep(3000);

////////////////////////////////////////////////
              /*  Conexión con Robot Pioneer (Maestro)  */
////////////////////////////////////////////////

//*
//Aria::init();
```

```
ArArgumentBuilder arg_nut(10);
ArArgumentParser parser_nut(&arg_nut);
ArRobot robot_nut;
arg_nut.add("-rh 10.0.126.11");

ArRobotConnector robotConnector_nut(&parser_nut, &robot_nut);
if(!robotConnector_nut.connectRobot())
    {
ArLog::log(ArLog::Terse, "simpleConnect:Could not connect to the robot.");
if(parser_nut.checkHelpAndWarnUnparsed())
{
Aria::logOptions();
Aria::exit(1);
}
}
if (!Aria::parseArgs() || !parser_nut.checkHelpAndWarnUnparsed())
{
Aria::logOptions();
Aria::exit(1);
}

ArLog::log(ArLog::Normal, "simpleConnect: Connected to robot");
robot_nut.runAsync(true);
robot_nut.comInt(ArCommands::SONAR, 0);
ArTime start;
ArSerialConnection robcon;
ArLog::log(ArLog::Normal, "Programa prueba: Se iniciara en 3 segundos...");
ArUtil::sleep(3000);
ArLog::log(ArLog::Normal, "Programa prueba: Iniciado.");  //*/


////////////////////////////////////////////
        /*  Inicio proceso */
////////////////////////////////////////////

ArTime tmsi,tmsf;
tmsi.setToNow();
tmsf.setToNow();

//*  // Se habilitan los motores del robot maestro //
robot_nut.lock();
robot_nut.enableMotors();
robot_nut.unlock(); //*/
```

```
while(t<=130){
msi=tmsi.mSecSince();
tc=double (msi)*double(0.001);

////////////////////////////////////////////
        /*  Trayectoria de referencia */
////////////////////////////////////////////

//* Círculo *//
//*
xr=sin(t/7);
yr=cos(t/7);  //*/
//printf("xr=%f   yr=%f\n",xr,yr);

//* Lemniscata *//
/*
xr=cx+ax*sin(t/10);
yr=cy+ay*sin(t/10)*cos(t/10);  //*/

////////////////////////////////////////////
    /*  Errores */
////////////////////////////////////////////

Emex=(x-xs)/2;
Emey=(y-ys)/2;

Ep1x=xmpr-Mod2.x;
Ep1y=ympr-Mod2.y;

Ep2x=Ep1x+Emex;//Emex;
Ep2y=Ep1y+Emey;//Emey;

Egx=xr-Ep2x;
Egy=yr-Ep2y;

////////////////////////////////////////////
    /*  Control maestro */
////////////////////////////////////////////

//control_maestro(xr, yr); //Egx, Egy
    R1.control(Egx,Egy,i,h);
////////////////////////////////////////////
       /* Envío de datos al Robot Pioneer */
////////////////////////////////////////////
```

```
//*
wlmm=R1.vmm-(R1.W*L);
wrmm=R1.vmm+(R1.W*L);
//printf("wlmm=%f   wrmm=%f\n",wlmm,wrmm);

robot_nut.lock();
robot_nut.setVel2(wlmm,wrmm);
robot_nut.unlock();  //*/

/*
robot_nut.lock();
robot_nut.setVel(100); //vmm
robot_nut.unlock();

    robot_nut.lock();
robot_nut.setRotVel(20); //wmm
robot_nut.unlock();
//*/

////////////////////////////////////////////////////////////
                /* Obtención de datos del Robot Pioneer (Odometría) */
////////////////////////////////////////////////////////////

//*
x=robot_nut.getX()* double(0.001);
y=robot_nut.getY()* double(0.001);
th=ArMath::degToRad(robot_nut.getTh());

Vt=robot_nut.getVel();
Vr=robot_nut.getRotVel();
//printf("x=%f   y=%f\n",x,y);
//*/

////////////////////////////////////////////////
/*  Predictor de Smith */
////////////////////////////////////////////////

xmp_r = retardo(x,r5,8);
ymp_r = retardo(y,r5,9);

Mod1.control(xmp_r,ymp_r,i,h);
xmpr = retardo(Mod1.x,r6,10);
ympr = retardo(Mod1.y,r6,11);
```

```
Mod2.control(x,y,i,h);

////////////////////////////////////////////////
/*  Error Referencia */
////////////////////////////////////////////////

ex=x-efx/5.0;
ey=y-efy/5.0;

////////////////////////////////////////////////
/*  Retardos Envío */
////////////////////////////////////////////////

repx = retardo(ex,r1,4);
repy = retardo(ey,r1,5);
remx = retardo(ex,r2,6);
remy = retardo(ey,r2,7);

////////////////////////////////////////////////
       /*  Envío de datos al esclavo */
////////////////////////////////////////////////

if (i%comu==0){
stringstream ss, ss1;
ss  << setprecision(6) << repx;
ss1 << setprecision(6) << repy;
string sw1 = ss.str();
string sw2 = ss1.str();
std::string sw3="|";
std::string sw=sw1+sw3+sw2;
strToSend = sw.c_str();
cout<<strlen(strToSend)<<endl;
sock.writeString(strToSend);
}

////////////////////////////////////////////////
  /*  Error de seguimiento del modelo */
////////////////////////////////////////////////

//ex_m=ex-xm;
//ey_m=ey-ym;

////////////////////////////////////////////////
```

```
/*  Modelo */
////////////////////////////////////////////
/*
kpx=2;
kix=0.1;
kdx=4;
kpy=2;
kiy=0.1;
kdy=4;
//*/
//Mod3.control(remx,remy,i,h);
Mod3.control(ex,ey,i,h);
//modelo_bloque(remx,remy);

////////////////////////////////////////////
      /*  Lectura de datos del maestro */
////////////////////////////////////////////

//*
if (i%comu==0){

//if(sock.readString()==NULL){
bufer=sock.readString();
ptrToken=strtok(strdup(bufer),"|");
xs=atof(ptrToken);
  ptrToken=strtok(NULL,"|");
ys=atof(ptrToken);
}//}
printf("xs=%f   ys=%f\n",xs,ys);
//*/

////////////////////////////////////////////
   /*  Retardos Recepción */
////////////////////////////////////////////

xrep = retardo(xs,r3,1);
yrep = retardo(ys,r3,0);
xrem=retardo(Mod3.x,r4,2);
yrem=retardo(Mod3.y,r4,3);

////////////////////////////////////////////
   /*  Error robot-modelo */
////////////////////////////////////////////
```

```
epmx=(xrep-xrem);
epmy=(yrep-yrem);

//epmx=(xs-Mod3.x);
//epmy=(ys-Mod3.y);

////////////////////////////////////////////////
 /*  Filtro */
////////////////////////////////////////////////

efx=filtro_imp(epmx,0);
efy=filtro_imp(epmy,1);

////////////////////////////////////////////////
/*  Guardar datos */
////////////////////////////////////////////////

rtraydata1<<xr<<"    "<<yr<<endl;
rtraydata2<<x<<"    "<<y<<"    "<<th<<endl;
rtraydata3<<xs<<"    "<<ys<<endl;
rtraydata4<<xm<<"    "<<ym<<"    "<<thm<<endl;
rtraydata5<<repx<<"    "<<repy<<"    "<<remx<<"    "<<remy<<endl;
rtraydata6<<xrep<<"    "<<yrep<<"    "<<xrem<<"    "<<yrem<<endl;
rtraydata7<<ex<<"    "<<ey<<endl;
rtraydata8<<epmx<<"    "<<epmy<<endl;
rtraydata9<<efx<<"    "<<efy<<endl;
rtraydata10<< vmm/double(1000.0) <<"    "<< W <<endl;
rtraydata11<< s1M <<"    "<< s2M <<endl;
rtraydata12<< Mod1.x <<"    "<< Mod1.y <<"    "<<Mod1.th<<endl;
rtraydata13<< Mod2.x <<"    "<< Mod2.y <<"    "<<Mod2.th<<endl;
rtraydata14<< Egx <<"    "<< Egy <<"    "<< Ep1x <<"    "
<< Ep1y<<"    "<< Ep2x <<"    "<< Ep2y<<"    "<<Emex <<"    "
<< Emey<<"    "<<endl; // Errores Eg, Ep1, Ep2 y Eme
rtraydata15<< Mod3.x <<"    "<< Mod3.y <<"    "<<Mod3.th<<endl;
//rtraydata9<<efx<<"    "<<efy<<endl;

////////////////////////////////////////////////
/*  Tiempo */
////////////////////////////////////////////////
msf=tmsf.mSecSince();
double tp=msf-msi;
if(tp<h*1000){
ArUtil::sleep(h*1000-(tp));}
printf("t= %.8lf\t tc= %.8lf\n",t,tc);
```

```
t=t+h;
i++;

if(t==60){ return(1);}
}

robot_nut.lock();
robot_nut.stop();
robot_nut.unlock();
ArUtil::sleep(1000);

////////////////////////////////////////////////////////
            /* Fin de la comunicación con el Robot Pioneer */
////////////////////////////////////////////////////////

sock.close();
ArLog::log(ArLog::Normal,
"socketServerExample: Server socket closed.");
ArLog::log(ArLog::Normal,
"simpleConnect: Ending robot thread...");
robot_nut.stopRunning();
// wait for the thread to stop
robot_nut.waitForRunExit();
// exit
ArLog::log(ArLog::Normal, "simpleConnect: Exiting.");
Aria::shutdown();
Aria::exit();

    cin.get();
return 0;
}
```

## B.2. Computadora B

```
#include "Aria.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <stdlib.h>
#include "control.h"
```

```
using namespace std;

// Ganancias del controlador PID
/*
   double kpx=1;
   double kix=0.01;
   double kdx=10;
   double kpy=1;
   double kiy=0.01;
   double kdy=10;
   double lda=0.25;
   //*/

//*
   double kpx=0.9;//2;
   double kix=0.0001;
   double kdx=2.8;//10;
   double kpy=0.9;//8;
   double kiy=0.0001;
   double kdy=2.8;//40;
   double lda=0.25;
   //*/


// Valores de simulación

  long msi,msf;
double tc=0;
double t=0;
    double tf=60;
    double h=0.010;
   int i=1;

// Valores iniciales coordenadas

    double x=0;
    double y=0;
    double th=0;
double Vt, Vr;

// Valores iniciales para bloque planta

    double ixa2=0;
    double iya2=0;
```

```
    double epx2=0;
    double epy2=0;

double upx=0;
    double upy=0;
    double dux=0;
    double duy=0;

    double dw1=0;
    double dv=0;
    double dw=0;
    double vb=0;
    double wb=0;
    double zb=0.00001;
    double w1b=0;

double xb[5]={0};
double yb[5]={0};

// Trayectoria deseada

double xr=0;
double yr=1;

// Variables de control

double vmm=0;
double wmm=0;
double wrmm=0;
double wlmm=0;
double W=0;
double L=190;

double s1=0;
double s2=0;

const char * q;
char *ptrToken;
int comu=3;


////////////////////////////////////////////////
                      /*  Inicio main  */
////////////////////////////////////////////////
```

```
int main (int argc, char** argv){

ofstream rtraydata1("Recepcion.txt");
ofstream rtraydata2("Envio.txt");
ofstream rtraydata3("Control.txt");
ofstream rtraydata4("Proceso.txt");
ofstream rtraydata5("Velocidades_esclavo.txt");
ofstream rtraydata6("Salida_Inversa_Esclavo.txt");

////////////////////////////////////////////
           /*  Comunicación TPC/IP Sockets en Aria  */
////////////////////////////////////////////

const char *strToSend="S";
char *bufer;
Aria::init();
ArSocket serverSock, clientSock;

if (serverSock.open(7777, ArSocket::TCP))
ArLog::log(ArLog::Normal, "socketServerExample: Opened the server port.");
else
 {
ArLog::log(ArLog::Normal, "socketServerExample: Failed to open the server port: %s.",
serverSock.getErrorStr().c_str());
Aria::exit(-1);
return(-1);
 }
int clientNo = 0;

// Wait for a client to connect to us.
ArLog::log(ArLog::Normal, "socketServerExample: Waiting for a client to connect. Press C
if (serverSock.accept(&clientSock))
ArLog::log(ArLog::Normal, "socketServerExample: Client %d has connected.", clientNo);
else
ArLog::log(ArLog::Terse, "socketServerExample: Error in accepting a connection from the
serverSock.getErrorStr().c_str());

////////////////////////////////////////////////
             /*  Conexión con Robot Pioneer (Esclavo) */
////////////////////////////////////////////////

//*
//Aria::init();
```

```
ArArgumentBuilder arg_nut(10);
ArArgumentParser parser_nut(&arg_nut);
ArRobot robot_nut;
arg_nut.add("-rh 10.0.126.12");

ArRobotConnector robotConnector_nut(&parser_nut, &robot_nut);
if(!robotConnector_nut.connectRobot())
    {
ArLog::log(ArLog::Terse, "simpleConnect:Could not connect to the robot.");
if(parser_nut.checkHelpAndWarnUnparsed())
{
Aria::logOptions();
Aria::exit(1);
}
}
if (!Aria::parseArgs() || !parser_nut.checkHelpAndWarnUnparsed())
{
Aria::logOptions();
Aria::exit(1);
}

ArLog::log(ArLog::Normal, "simpleConnect: Connected to robot");
robot_nut.runAsync(true);
robot_nut.comInt(ArCommands::SONAR, 0);
ArTime start;
ArSerialConnection robcon;
ArLog::log(ArLog::Normal, "Programa prueba: Se iniciara en 3 segundos...");
ArUtil::sleep(3000);
ArLog::log(ArLog::Normal, "Programa prueba: Iniciado.");

//*/
//robot_nut.setVel(300);
//robot_nut.setRotVel(97.4828);
/*if (clientSock.write(strToSend, strlen(strToSend)) == (int) strlen(strToSend))
ArLog::log(ArLog::Normal, "Se envió la cadena 'comienza' al cliente.");
else
{
ArLog::log(ArLog::Normal, "Error al enviar la cadena 'comienza' al cliente.");
Aria::exit(-1);
return(-1);
}
//*/

////////////////////////////////////////////////
```

```
                    /*  Programa principal */
////////////////////////////////////////////////

ArTime tmsi,tmsf;
tmsi.setToNow();
tmsf.setToNow();

//*    // Se habilitan los motores del robot esclavo //
robot_nut.lock();
robot_nut.enableMotors();
robot_nut.unlock(); //*/

while (t<tf+h){
msi = tmsi.mSecSince();
tc = double(msi)*double(0.001);

////////////////////////////////////////////////
      /*  Lectura de datos del maestro */
////////////////////////////////////////////////

if (i%comu==0){

bufer=clientSock.readString();
ptrToken=strtok(strdup(bufer),"|");
xr=atof(ptrToken);
  ptrToken=strtok(NULL,"|");
yr=atof(ptrToken);
}
printf("xr=%f   yr=%f\n",xr,yr);

////////////////////////////////////////////////
/*  Control */
////////////////////////////////////////////////

control(xr, yr);

////////////////////////////////////////////
       /* Envío de datos al Robot Pioneer */
////////////////////////////////////////////

//*
wlmm=vmm-(W*L);
wrmm=vmm+(W*L);
```

```
robot_nut.lock();
robot_nut.setVel2(wlmm,wrmm);
robot_nut.unlock();
//*/

/*
robot_nut.lock();
robot_nut.setVel(vmm); //vmm
robot_nut.unlock();

    robot_nut.lock();
robot_nut.setRotVel(wmm); //wmm
robot_nut.unlock();
//*/

////////////////////////////////////////////////////////////
                /* Obtención de datos del Robot Pioneer (Odometría) */
////////////////////////////////////////////////////////////

x=robot_nut.getX()* double(0.001);
y=robot_nut.getY()* double(0.001);
th=ArMath::degToRad(robot_nut.getTh());

Vt=robot_nut.getVel();
Vr=robot_nut.getRotVel();

//printf("x=%f   y=%f\n",x,y);

//x=x+0.2*cos(th);
//y=y+0.2*sin(th);

////////////////////////////////////////////////
      /*  Envío de datos al maestro */
////////////////////////////////////////////////

//*
if (i%comu==0){
stringstream ss, ss1;
ss  << setprecision(6) << x;
ss1 << setprecision(6) << y;
string sw1 = ss.str();
string sw2 = ss1.str();
std::string sw3="|";
std::string sw=sw1+sw3+sw2;
```

```
strToSend = sw.c_str();
clientSock.writeString(strToSend);
}
//*/

////////////////////////////////////////////////
    /*  Tiempo de muestreo a 10 ms */
////////////////////////////////////////////////

msf = tmsf.mSecSince();
double tp=msf-msi;
if (tp<10){
ArUtil::sleep(10-(tp));}
    printf("t=%.3lf\t tc=%.3lf\n",t,tc);
printf("tp=%.2lf\n",tp);
cout<<endl;

////////////////////////////////////////////////
/*  Guardar datos */
////////////////////////////////////////////////

rtraydata1<<xr<<"    "<<yr<<endl;
rtraydata2<<x<<"    "<<y<<"    "<<th<<endl;
rtraydata3<<vmm/double (1000.0) <<"    "<< W <<endl;
rtraydata4<<tp<<endl;
rtraydata5<<Vt/double (1000.0) <<"    "<< Vr <<endl;
rtraydata6<<s1 <<"    "<< s2 <<endl;

if (GetAsyncKeyState(27)){t=tf+h;}
t=t+h;
i=i+1;
}

robot_nut.lock();
robot_nut.stop();
robot_nut.unlock();
ArUtil::sleep(1000);

////////////////////////////////////////////////////////
            /* Fin de la comunicación con el Robot Pioneer */
////////////////////////////////////////////////////////

serverSock.close();
ArLog::log(ArLog::Normal, "socketServerExample: Server socket closed.");
```

```
ArLog::log(ArLog::Normal, "simpleConnect: Ending robot thread...");
robot_nut.stopRunning();
// wait for the thread to stop
robot_nut.waitForRunExit();
// exit
ArLog::log(ArLog::Normal, "simpleConnect: Exiting.");
Aria::shutdown();
Aria::exit();

    cin.get();
return 0;
}
```