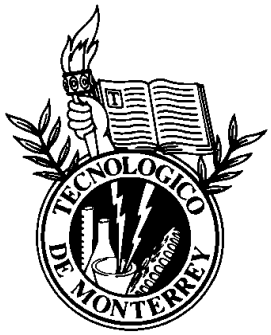
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS DE LA DIVISIÓN DE ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES



MODELO DE COMUNICACIÓN PARA LA TELEOPERACIÓN Y EJECUCIÓN AUTÓNOMA DE TAREAS EN UN VEHÍCULO

MAESTRIA EN CIENCIAS

CON ESPECIALIDAD EN AUTOMATIZACIÓN

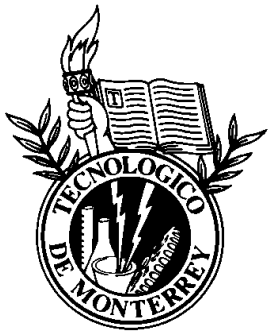
SISTEMAS INTELIGENTES

POR

ADRIANA ARACELI BENERANDA CANTÚ GONZÁLEZ

DICIEMBRE DEL 2001

MODELO DE COMUNICACIÓN PARA LA TELEOPERACIÓN Y EJECUCIÓN AUTÓNOMA DE TAREAS EN UN VEHÍCULO



Maestría en Automatización

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Por

Adriana Araceli Beneranda Cantú González

Diciembre del 2001

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Programa de Graduados de la división en Electrónica, Computación, Información y Comunicaciones

Los miembros del comité de tesis recomendamos que la presente tesis del Ing. Adriana Araceli Beneranda Cantú González sea aceptada como requisito parcial para obtener el grado académico de **Maestro en Ciencias con especialidad en Automatización: Sistemas Inteligentes**.

Comité de Tesis

Dr. José Luis Gordillo Moscoso

Asesor

Dr. Ramón Rodríguez Dagnino

Sinodal

Dr. Horacio Martínez Alfaro

Sinodal

Dr. David Garza Salazar

Director del programa de Posgrado en Electrónica, Computación, Información y Comunicaciones

Diciembre del 2001

MODELO DE COMUNICACIÓN PARA LA TELEOPERACIÓN Y EJECUCIÓN AUTÓNOMA DE TAREAS EN UN VEHÍCULO

por

Adriana Araceli Beneranda Cantú González

Tesis

Presentada al Programa de Graduados en Electrónica, Computación, Información y Comunicaciones

del

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

como requisito parcial para obtener el grado académico de

Maestro en Ciencias

Monterrey, N.L., Diciembre del 2001

A Dios.

Por todo lo que me ha concedido y de quien todo lo espero.

A mis padres Juan y Adriana.

Por su amor y apoyo incondicional.

A mis amigos.

Por todo su apoyo.

A mi asesor, el Dr. Gordillo.

Por su guía y amistad durante todo este tiempo.

Agradecimientos

Deseo expresar mi más profundo agradecimiento a las personas que hicieron posible mi trabajo de investigación, en especial, a mi asesor, el Dr. José Luis Gordillo por confiar en mí y por su paciencia y guía para la realización de esta tesis.

A mis sinodales, el Dr. Ramón Rodríguez y el Dr. Horacio Martínez por su labor de revisión, sus sugerencias y comentarios.

Al Centro de Inteligencia Artificial, en especial al Dr. Francisco Cantú por darme la oportunidad de ser asistente de investigación y poder realizar mis estudios de maestría.

A mis compañeros que pertenecen o han pertenecido al Grupo Visión por compartir sus experiencias y conocimientos conmigo.

A Gerardo Palacios, Alfredo Cruz, Humberto Martínez y a todas las personas que de alguna forma me ayudaron a realizar esta tesis.

Adriana Araceli Beneranda Cantú González

Instituto Tecnológico y de Estudios Superiores de Monterrey

Diciembre del 2001

Resumen

Esta tesis propone la operación a distancia de un vehículo, a través de Internet. Se define una arquitectura con dos formas de operación: en la primera el vehículo cuenta con cierta autonomía; mientras que en la segunda, el vehículo es teleoperado. En ambos casos se requiere de la transmisión de datos e imágenes, así como el desarrollo de un protocolo de comunicación.

Para el comportamiento autónomo, los algoritmos de control utilizan visión, para que el vehículo siga una consigna específica, tal como mantenerse a una distancia fija de una marca artificial. En la segunda forma de operación, se procede mediante la teleoperación con imágenes, donde un usuario conectado a Internet controla los movimientos del vehículo.

La arquitectura se compone de tres módulos: el *Anfitrión* que se compone del vehículo previamente dicho y una computadora para su control, el *Invitado* que es la computadora que interactúa con el usuario y el *Medio* de comunicación utilizado para la transmisión de datos e imágenes.

Contenido

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

En la industria militar o en el área civil, es cada vez mayor la necesidad de trabajar con materiales peligrosos, tal como ácidos, explosivos, tóxicos o radioactivos; lo cual evidencia la conveniencia de contar con máquinas controladas a distancia para el manejo y transportación de este tipo de materiales.

Por su parte, los robots realizan algunas operaciones más eficientemente y con más precisión que los humanos, sobre todo cuando trabajan en ambientes inaccesibles, hostiles o peligrosos para la vida humana. Algunos ejemplos de estos ambientes son en la investigación submarina, en la exploración planetaria o en la minería, donde se requieren tareas de exploración y excavación que son riesgosas para un humano las cuales pueden ser ejecutadas por un vehículo cuya operación no requiera la intervención directa del conductor.

Actualmente algunos laboratorios como el Centro de Maquinas Inteligentes y Robótica [Cimar 98] y el Centro de Inteligencia Artificial del ITESM [Gordillo 98], orientan sus actividades a automatizar vehículos pesados, para realizar una tarea específica dentro de una mina. Debido a las dificultades que se tienen al trabajar con este tipo de vehículos, otros laboratorios han desarrollado vehículos a escala [Fernández 97] que son guiados visualmente por un usuario remoto.

Aunque algunas tareas pueden ser automatizadas, para que las ejecute un vehículo con poca o nula intervención humana, existen operaciones tales como llevar al vehículo al sitio de la ejecución de la tarea, o desbloquear una situación inesperada que debido a su complejidad, requieren una destreza muy alta para ser ejecutadas por una máquina de forma confiable. Estas tareas necesitan de la ayuda humana, ya que al trabajar en ambientes reales se tienen altos niveles de operación con incertidumbre.

Actualmente se plantean dos modos de operación que se consideran complementarios: teleoperación y operación autónoma. En ambos casos se requiere que el vehículo sea controlado a distancia y tenga cierto nivel de autonomía para ejecutar la tarea. Para que su actuación sea de beneficio para el humano, se requiere que el vehículo permita al conductor permanecer fuera de los ambientes peligrosos y extremos de trabajo.

La teleoperación se refiere a la operación a distancia de un aparato mecánico o dispositivo para la realización de un trabajo. En este modo de operación, el usuario requiere de excesiva concentración debido a que controla directamente los movimientos del vehículo, usando la información recibida del espacio de trabajo. Cuando las tareas son repetitivas, el usuario debe repetirlas manualmente, cada vez, lo cual a la larga, le provoca cansancio, perdida de concentración y errores. Además, considerando el retraso por la comunicación, si la tarea requiere retroalimentación inmediata y el tiempo de respuesta es crítico, el sistema se vuelve inoperable.

Al permitir cierta autonomía, para que al vehículo realice las tareas repetitivas sin que el usuario tenga que intervenir, se descarga al usuario del control de los detalles, permitiéndole concentrarse en el alto nivel de la tarea. En la operación autónoma, el usuario establece una meta al vehículo, la cual es ejecutada por un sistema de control y supervisada por el usuario; por ello el vehículo debe tener un grado de control mientras desarrolla la tarea de forma autónoma. De acuerdo a lo anterior, se puede considerar autonomía como la capacidad de realizar una tarea específica con mínima o nula intervención humana utilizando la percepción de los sensores y el conocimiento previamente adquirido. Un vehículo autónomo basado en visión utiliza elementos del medio ambiente, dentro de su espacio de trabajo, para conocer su posición.

El objetivo de esta tesis es definir un modelo de comunicación para controlar un vehículo a distancia, a través de Internet, que integre la teleoperación y la operación autónoma. Se usará una forma teleoperada para ubicar el vehículo, reconocer el entorno de trabajo e iniciar la tarea; en cambio el vehículo, estará dotado de autonomía durante la ejecución de la tarea. En la operación autónoma, el ejercicio puesto en funcionamiento busca mantener al vehículo a una distancia establecida de la marca; se considera que la autonomía del vehículo empieza cuando el usuario libera el control para que el vehículo ejecute la tarea y termina cuando, una vez finalizada, el vehículo devuelve el control al usuario. Durante la operación, el usuario estará en constante comunicación con el vehículo.

El desarrollo de la comunicación conmutada entre las modalidades de teleoperación y comportamiento autónomo requiere de un módulo para la teleoperaración del vehículo, un módulo que incluya los algoritmos de visión necesarios para realizar la tarea en el comportamiento autónomo y un mecanismo de conmutación entre ambos modos de operación. Debido a que este trabajo forma parte de un proyecto del Grupo Visión del ITESM, para el control de la velocidad y dirección del vehículo se utilizan los algoritmos desarrollados por [Cruz 01] y [Palacios 00].

La arquitectura propuesta, para que un usuario pueda controlar remotamente un vehículo, mostrada en la figura 1.1, está compuesta por un módulo *Anfitrión* y un módulo *Invitado* conectados a través de un *Medio.* Por lo tanto, esta arquitectura puede ser utilizada con cualquier clase de vehículo sin importar su tamaño.

El módulo *Anfitrión* esta compuesto por el *Vehículo* y por el *Controlador Anfitrión*. Para que el usuario tenga un mayor conocimiento del ambiente remoto, se cuenta con una cámara montada sobre el *Vehículo*. Para la comunicación entre el *Vehículo* y el *Controlador Anfitrión* se utilizan canales independientes para la transmisión de datos e imágenes.

El módulo *Invitado* se encarga de interactuar con el usuario; lo compone cualquier computadora que se encuentre del lado del usuario y que cuente con la maquina virtual de Java y una conexión a *Internet*.

El *Medio* utilizado para la transmisión de datos e imágenes del *Anfitrión* al *Invitado* es Internet ya que permite que el vehículo sea operado remotamente casi desde cualquier lugar.

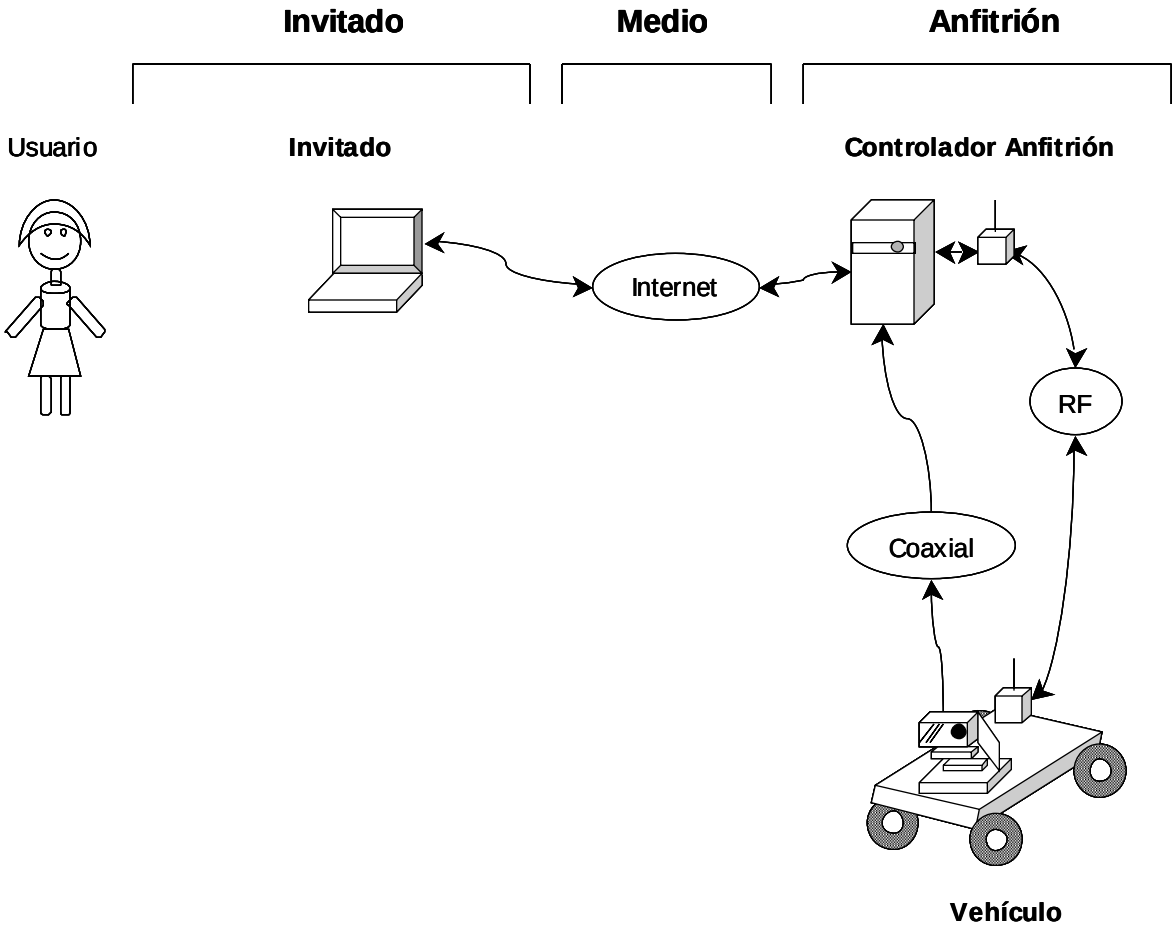


Figura 1.1: Arquitectura general. El sistema está compuesto por un Módulo *Anfitrión* y un Módulo *Invitado* conectados a través de un *Medio*

El *Controlador Anfitrión* se encarga de la comunicación entre el *Vehículo* y el *Invitado* y del procesamiento de las imágenes recibidas del *Vehículo*.

Durante la teleoperación, el *Invitado* recibe la retroalimentación de datos e imágenes; de acuerdo a la información recibida, el *Invitado* envía las instrucciones que el usuario desea sean ejecutadas por el *Vehículo*.

En la operación autónoma, el *Invitado* sólo recibe la retroalimentación enviada por el *Controlador Anfitrión,* para que el usuario supervise la ejecución de la tarea. En este modo de operación, el *Controlador Anfitrión* tiene el control del *Vehículo*.

Para la teleoperación, la transmisión de datos entre el *Controlador Anfitrión* y el *Vehículo* utiliza un canal bidireccional; mientras que para la operación autónoma, la transmisión de datos es en una dirección. La transmisión de imágenes del *Vehículo* al *Controlador Anfitrión* se realiza a través de un canal unidireccional, para ambos modos de operación.

Para describir el planteamiento de esta tesis, el Capítulo 2 describe en detalle los componentes del sistema bajo control: *Invitado*, *Anfitrión* y el *Medio* así como otros medios de comunicación utilizados en cada parte del sistema. Además, se explica el protocolo de comunicaciones y el control a distancia, del *Vehículo,* utilizado en el modo teleoperado. El Capítulo 3 detalla la operación del *Vehículo* en modo autónomo, describiendo el comportamiento del sistema cuando la marca deseada se encuentra presente en la imagen. El Capítulo 4 describe la implementación, incluyendo las pruebas realizadas y los resultados. En el Capítulo 5 se presentan las conclusiones y se proponen mejoras, basadas en los principales problemas que se presentaron durante la experimentación.

El Apéndice A contiene la especificación y el procedimiento de elaboración de los módems. El Apéndice B es un manual de operación del sistema, donde se describe la interfaz desarrollada para su operación y se detallan los programas que residen en el *Anfitrión* y el *Invitado*. El Apéndice C contiene las especificaciones de los controladores utilizados.

Capítulo 2

Modelo para el control a distancia de un *Vehículo*

Se busca desarrollar una arquitectura que permita que un *Vehículo* sea controlado a distancia por un usuario, dentro de un ambiente conocido. Por lo tanto, el sistema de control a distancia del *Vehículo* cuenta con los siguientes componentes [Sayers 98]:

1. *Invitado*: Una interfaz de operación, que incorpora un dispositivo de entrada utilizado por el usuario para controlar el sistema.

2. *Anfitrión*: Un dispositivo de salida que realiza las acciones que el usuario ordenó desde el sitio remoto; el vehículo en este caso.

3. *Medio*: Un esquema de comunicación entre los dos sitios.

Para compartir recursos e intercambiar información sin que el usuario conozca en detalle donde se localiza cada uno de los recursos, se decidió utilizar Internet como medio de comunicación entre el *Invitado* y el *Anfitrión*. Internet es una colección internacional de redes de computadoras, unidas para compartir información y tener acceso a servicios remotos. Todos los servicios que se encuentran en Internet funcionan con base a la arquitectura Cliente-Servidor.

La arquitectura general propuesta para controlar remotamente un *Vehículo* desde cualquier lugar mostrada en la figura 2.1, hace uso de Internet, bajo la arquitectura Cliente-Servidor. Esta arquitectura se compone por un módulo *Invitado* y un módulo *Anfitrión,* descritos en las secciones 2.1 y 2.2 respectivamente. Como se muestra en la figura 2.1, el usuario se conecta a través del *Invitado* para controlar al *Vehículo*.

2.1 *Invitado* y *Medio*

El *Invitado* es el componente del sistema que interactúa con el usuario; por lo tanto, se encarga de establecer la comunicación con el *Anfitrión* y de mostrar la información recibida del área de trabajo del *Vehículo*.

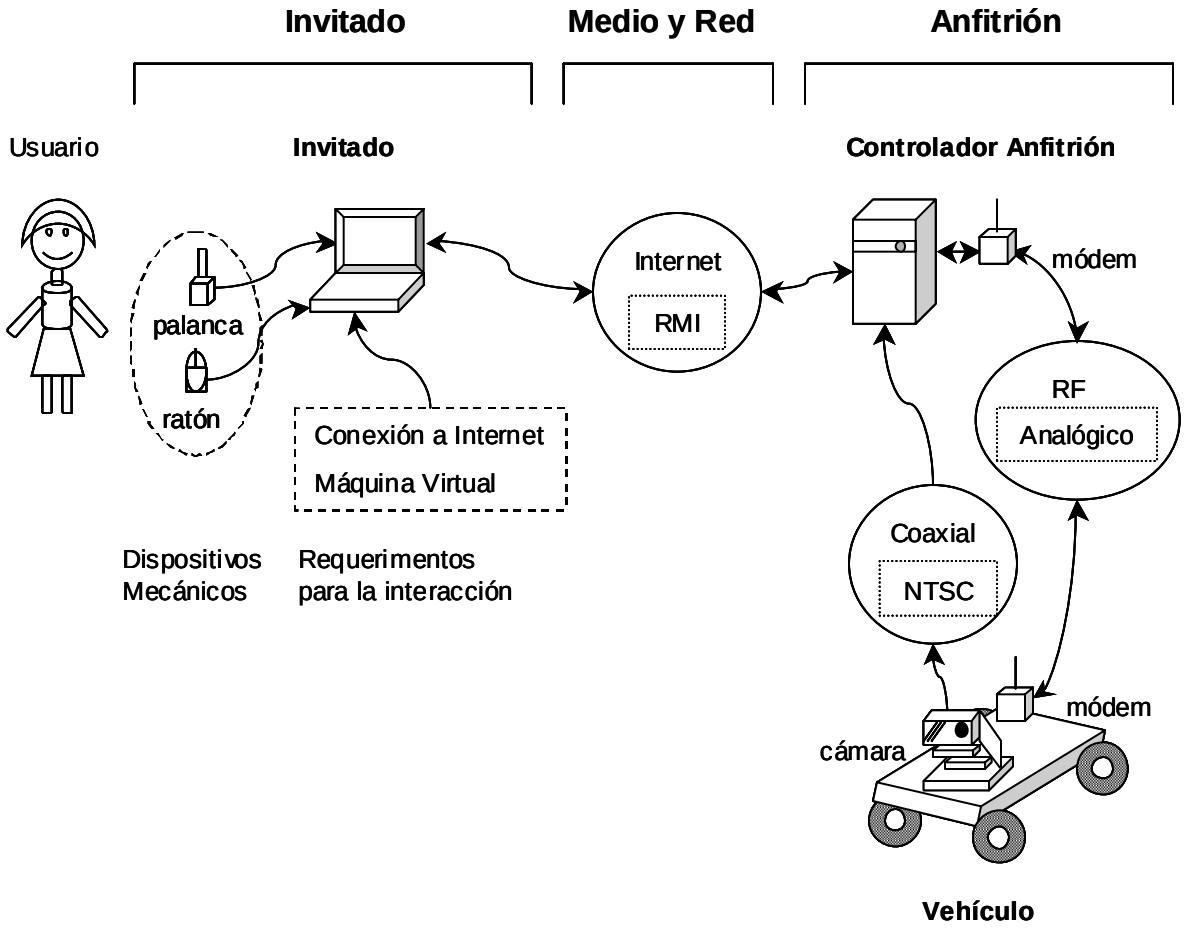


Figura 2.1: Esquema general para el control a distancia: el usuario se conecta a través del *Invitado*, para controlar al *Vehículo*

En la teleoperación, el *Invitado* se encarga de enviar las instrucciones establecidas por el usuario para ser ejecutadas por el *Vehículo* y recibe como retroalimentación las imágenes del área de trabajo y los datos de los sensores. En el modo autónomo, el usuario sólo supervisa la operación del *Vehículo*, por lo que el *Invitado* recibe como retroalimentación las instrucciones que el *Controlador Anfitrión* envió al *Vehículo* y la imagen que muestra la localización de la marca.

El *Invitado* puede ser cualquier computadora, del lado del usuario, que cuente con la maquina virtual de Java instalada y con una conexión a Internet. Para facilitar el control de un

Vehículo remoto, el *Invitado*, además del teclado y del ratón, puede contar con una palanca conectada al puerto de juegos.

Por su parte, el propósito del *Medio* es establecer una conexión entre el *Invitado* y el *Anfitrión* a través de la cual se compartan datos mediante un protocolo. Para la comunicación entre el *Invitado* y el *Anfitrión,* Internet utiliza el protocolo TCP/IP (*Transmission Control Protocol* / *Internet Protocol*).

El protocolo de control de transmisión TCP, utilizado para conectar al *Invitado* con el *Anfitrión*, es un protocolo orientado a conexión, que se encarga de asegurar que la información es enviada en el orden original. Un protocolo orientado a conexión establece un circuito virtual entre las maquinas; el *Invitado* solicita al *Anfitrión* el establecimiento de la comunicación, y solamente cuando se ha creado un canal de comunicación y ambas maquinas están preparadas para la transmisión, empieza la transferencia de datos en los dos sentidos.

El protocolo de Internet IP, define las direcciones de las computadoras conectadas a la red. Las direcciones de IP son asignadas a las computadoras según la red a la que estén conectadas. La dirección IP es un número entero de 4 bytes (32 bits) escrito en una notación que agrupa el número en cuatro conjuntos de 8 bits cada uno separado por un punto. La representación se hace en notación decimal; un ejemplo de una dirección es 131.178.38.80. A pesar de que las computadoras en Internet tienen asignadas una dirección IP única, se les puede asignar un nombre ya que la IP es difícil de recordar, debido a su formato numérico. El sistema de nombres de dominios (DNS, *Domain Name Server*) es el encargado de mapear la dirección de IP a los nombres de las computadoras. El *Invitado* utiliza como parámetro el valor del DNS para establecer la conexión.

El protocolo utilizado para la transmisión de datos e imágenes del *Invitado* al *Anfitrión* y viceversa, es la Invocación Remota de Métodos de Java (RMI) descrita en el Apéndice B, el cual funciona sobre el protocolo TCP/IP. RMI es un protocolo de alto nivel basado en los "*Sockets*" que funciona según la filosofía Cliente-Servidor a través del cual se pueden comunicar objetos remotos, de forma invisible, sin tener que manejar los detalles de bajo nivel requeridos por los "*Sockets*".

2.2 *Anfitrión*

El *Anfitrión*, mostrado en la figura 2.2, está compuesto por el *Vehículo* y por el *Controlador Anfitrión,* que es la computadora del lado del *Vehículo* encargada de mantener la comunicación entre el *Vehículo* y el *Invitado*.

El *Controlador Anfitrión* se encarga de manejar las instrucciones recibidas del *Invitado* y mandarlas al *Vehículo* para que se ejecuten; también se encarga de recibir y procesar las imágenes recibidas del *Vehícul*o. Este controlador cuenta con una tarjeta de captura de video que recibe las imágenes enviadas por la cámara del *Vehículo* para digitalizarlas y procesarlas.

La transmisión de imágenes de la cámara localizada sobre el *Vehículo* al *Controlador Anfitrión* se realiza a través de un cable coaxial. El medio de transmisión de datos entre el

Controlador Anfitrión y el *Vehículo* es Radio Frecuencia (RF). Se cuenta con un módem Transmisor-Receptor de datos de un solo canal, como se describe en el Apéndice A, conectado al puerto serial del *Controlador Anfitrión* y otro sobre el *Vehículo*.

Además de la plataforma móvil, el *Vehículo* contiene la unidad de procesamiento o *Controlador del Vehículo*, el módulo de control de velocidad y el módulo de control de dirección como se muestran en la figura 2.2. Los módulos de velocidad y dirección son descritos en las secciones 2.2.2 y 2.2.3, respectivamente.

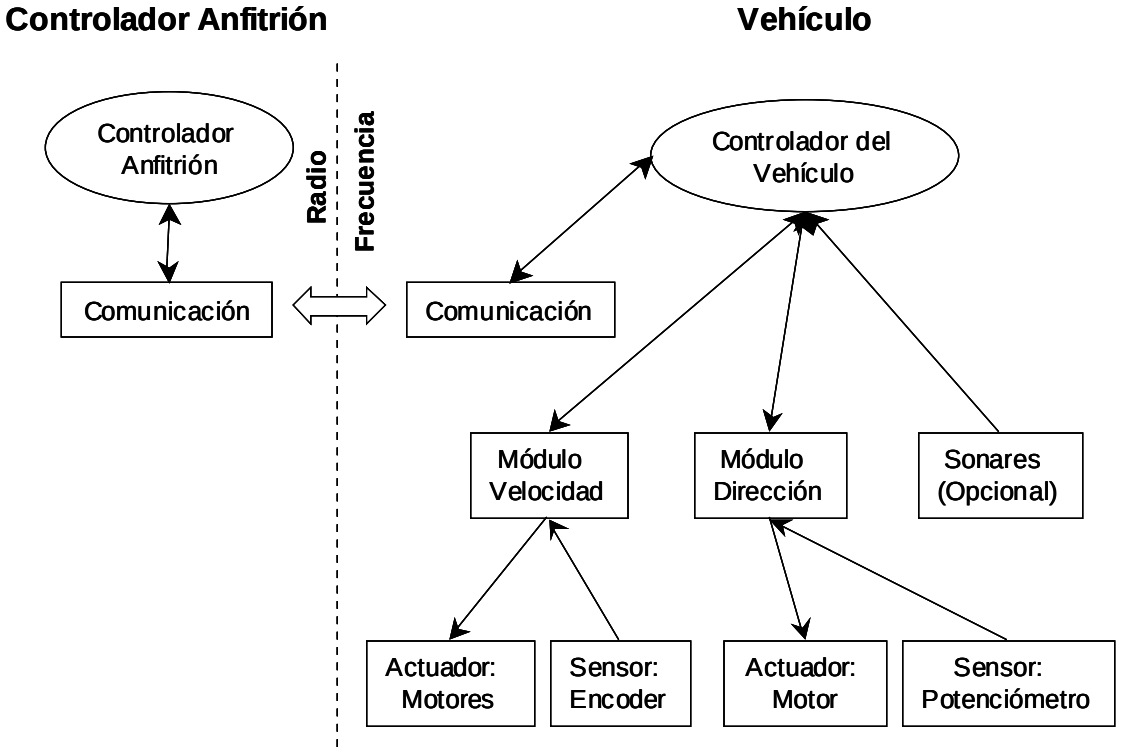


Figura 2.2: Componentes del *Anfitrión*. Además de la plataforma del carro, el *Vehículo* cuenta con el *Controlador* para coordinar el módulo de control de velocidad y el módulo de control de dirección

El *Vehículo* puede contar opcionalmente por un módulo que cuenta con uno o más sonares, los cuales producen una interrupción general para detener el *Vehículo* cuando se detecte algún obstáculo. Un sonar generalmente consta de dos elementos: un transductor ultrasónico y un circuito controlador en donde se genera la señal que se transmite y también se detecta el eco recibido por el mismo transductor.

El *Controlador del Vehículo* originalmente es un controlador en cascada; esto consiste en una tarjeta con el microprocesador principal que cuenta con el programa de instrucciones cargado en la memoria. Se utilizan dos lazos de control independientes para velocidad y dirección.

2.2.1 Modelo del *Vehículo*

El *Vehículo* que se automatizó, cuyo modelo se muestra en la figura 2.3, es un carro a escala de control remoto. Por sus características, se trata de un *Vehículo* no holonómico que se mueve hacia adelante y hacia atrás, además cambia de dirección a la derecha e izquierda, aunque por sus restricciones cinemáticas no puede moverse de lado.

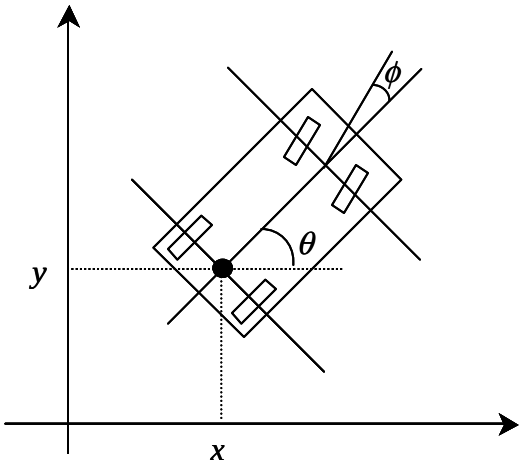


Figura 2.3: Modelo del *Vehículo* localizado en el plano (x, y) con una orientación θ. El ángulo de dirección es ϕ

El espacio de trabajo del *Vehículo* se define por un plano con coordenadas (x, y). El ángulo de orientación del *Vehículo* con respecto al plano es θ. El ángulo de dirección del *Vehículo* es ϕ, que corresponde a la inclinación de las llantas delanteras con respecto a la orientación del *Vehículo* en el plano (x, y).

El *Vehículo* cuenta con tres motores: uno para controlar el avance o retroceso en el eje de las llantas traseras, otro para las llantas delanteras y el último para dar vuelta a la derecha o izquierda. Para cambiar de dirección sólo se utilizan las dos llantas delanteras cuyo cambio de dirección tiene un límite mecánico. El *Vehículo* cuenta con un potenciómetro para medir el ángulo de dirección; se agregó un codificador óptico para medir la velocidad y distancia recorrida.

2.2.2 Módulo de velocidad

El módulo de velocidad está compuesto por la tarjeta controladora de velocidad, dos motores de corriente directa conectados en paralelo: uno para las llantas delanteras y otro para las llantas traseras. Para medir la velocidad y desplazamiento se cuenta con un sensor de velocidad que consiste en un codificador óptico diferencial.

En la figura 2.4 se muestran las variables controladas en el módulo de velocidad. V_d es la velocidad deseada, $\mathcal{E}_v$ es el error velocidad, m_v es la señal PWM de manipulación de velocidad, y V es la velocidad real. El algoritmo de control de velocidad es un controlador de

tipo PI (Proporcional-Integral) utilizado para llevar al *Vehículo* a una velocidad constante a lo largo de la trayectoria.

El *Vehículo* cuenta con un motor de corriente directa para las llantas del eje delantero y otro para las del eje trasero, conectados en paralelo. El sistema se compone de una entrada que es el voltaje aplicado al motor , mientras que la salida es la velocidad, la cual se mide con el sensor diferencial.

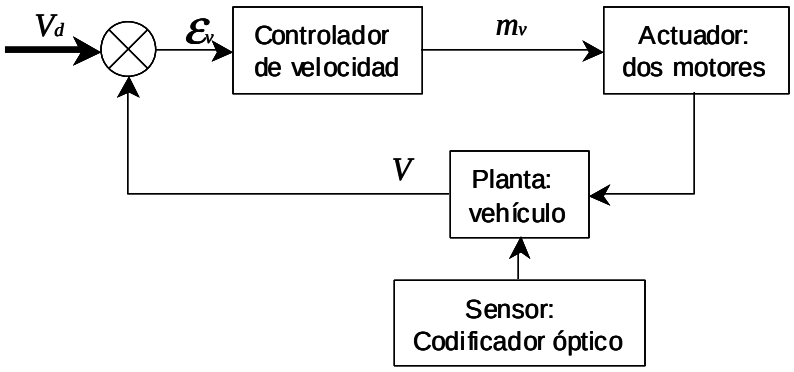


Figura 2.4: Diagrama de control de velocidad donde V_d es la velocidad deseada, $\mathcal{E}_v$ es el error velocidad, m_v es la señal PWM de manipulación de velocidad y V es la velocidad real

Como se muestra en la figura 2.4, la variable de entrada del sistema de control es la velocidad deseada V_d, el controlador calcula el error $\mathcal{E}_v$ utilizando la velocidad real V calculada por el vehículo. La salida del controlador es la manipulación m_v que es enviada a los motores.

Para cambiar la velocidad del *Vehículo*, se utiliza una técnica de Modulación de Ancho de Pulso (PWM) mostrada en la figura 2.5, la cual consiste en variar la razón de alimentación de corriente de los motores.

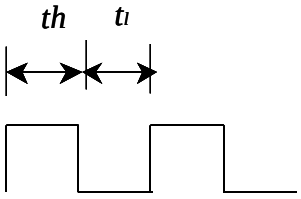


Figura 2.5: Porcentaje de anchura del pulso. La señal de voltaje PWM tiene un período constante

El porcentaje de anchura del pulso esta definido por:

$$PWM = \frac{t_h}{t_h + t_i} \tag{2.1}$$

donde t_h es el tiempo que dura el pulso en estado alto (1) y t_l es el tiempo que dura en estado bajo (0). Las señales PWM son señales de voltaje con un período constante, donde se cambia el ciclo de trabajo. El ciclo de trabajo (DC) se calcula con la siguiente ecuación:

$$\%DC = 100\left(\frac{t_h}{t_h + t_i}\right) \tag{2.2}$$

En la figura 2.6 se muestran algunos ejemplos de señales PWM con diferentes ciclos de trabajo. La potencia de entrada que recibe el motor es directamente proporcional al ciclo de trabajo de la señal.

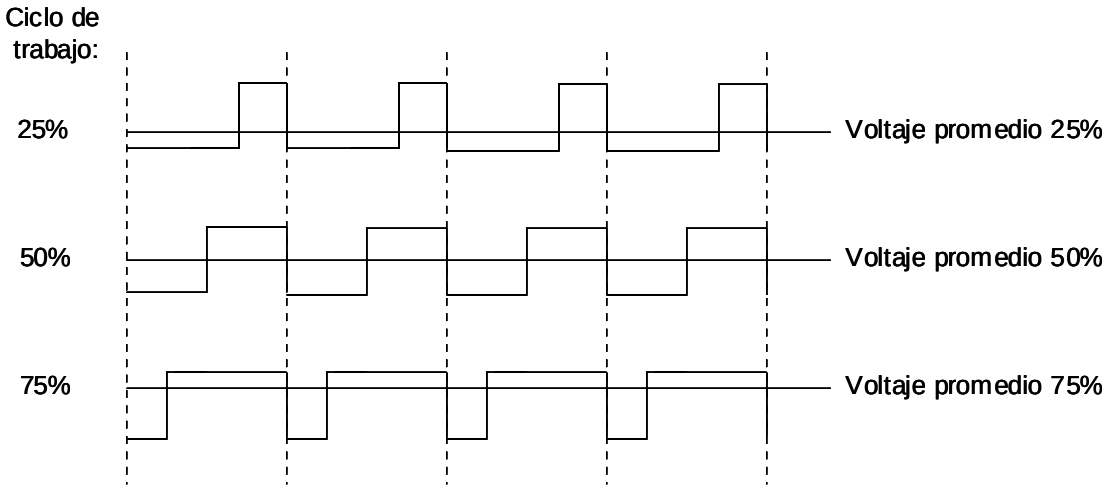


Figura 2.6: Ejemplo de señales PWM con diferentes ciclos de trabajo. La potencia de entrada que recibe el motor es directamente proporcional al ciclo de trabajo de la señal

La información que recibe el *Vehículo* de los sensores es su orientación y posición así como la distancia que ha avanzado medida con un codificador.

Hay dos tipos básicos de codificadores ópticos: el incremental y el absoluto. La versión incremental mide la velocidad rotacional y puede inferir la posición relativa mientras que los modelos absolutos directamente miden la posición angular y infieren la velocidad. El tipo más simple de codificadores incrementales es un codificador de un solo canal, que es un instrumento mecánico que produce un cierto número de pulsos de onda cuadrada por cada revolución del eje; estos dispositivos son incapaces de detectar la dirección de rotación por lo que no pueden ser usados como sensores de posición.

Por sus propiedades el *Vehículo* utilizó un codificador óptico del tipo incremental de cuadratura de fase; este codificador añade un segundo canal desplazado del primero, lo cual produce una sucesión de pulsos desfasados 90°. Por lo tanto, para determinar la dirección del movimiento, se tienen dos señales de salida cuadradas desfasadas 90° como se muestra en la figura 2.7. La naturaleza incremental de las señales de salida permiten que cualquier resolución

de posición angular sea relativa a alguna referencia específica [Borenstein 96]. Cada pulso se convierte a distancia recorrida y el sentido del movimiento se mide con el desfasamiento de las señales.

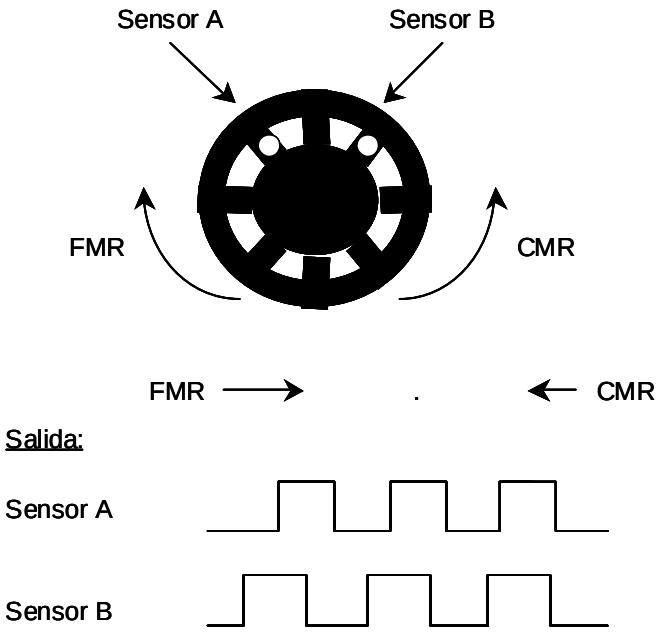


Figura 2.7: Codificador óptico incremental con dos señales desfasadas utilizadas para medir la distancia recorrida y el sentido del movimiento

En la figura 2.7 se obtienen dos secuencias distintas en los canales del codificador óptico para medir el sentido de avance del *Vehículo*. En la tabla 2.1 se muestra la secuencia de la señal mostrada en la figura 2.7 cuando el codificador gira en el sentido contrario a las manecillas del reloj. Se puede ver que como la señal del sensor A se atrasa a la del sensor B.

Tabla 2.1: Secuencia en contra de las manecillas del reloj

A	B
0	0
0	1
1	1
1	0

Para calcular la distancia recorrida, se utiliza el número de pulsos que se generan por unidad de tiempo en relación a las vueltas de la llanta,

$$D = \frac{P_{llanta}}{N_{pulsos}} \tag{2.3}$$

donde D es la distancia recorrida en cada pulso, P_{llanta} es el perímetro de la llanta, N_{pulsos} son los números de pulsos por vuelta de llanta. El perímetro de la llanta es un valor conocido.

Para calcular la velocidad se utiliza el mismo sensor pero ahora se cuenta el número de pulsos detectados en el codificador dentro de un período de tiempo,

$$V = \frac{P_{contados}}{T_{transcurrido}} D \tag{2.4}$$

donde V es la velocidad del *Vehículo*, $P_{contados}$ es el número de pulsos contados, $T_{transcurrido}$ es el período de tiempo y D es la distancia que se recorre por cada pulso. La distancia recorrida por cada pulso y el período de tiempo son valores conocidos.

Para determinar el sentido del *Vehículo* se utiliza la segunda señal del codificador; de acuerdo a la secuencia de pulsos se determina si el *Vehículo* avanza o si retrocede. Cuando se detecta la entrada al pulso positivo en la señal A, se mide el valor de la señal B, si se encuentra en alto, significa que el *Vehículo* avanza, y si está en bajo significa que el *Vehículo* retrocede.

2.2.3 Módulo de dirección

El módulo de dirección está compuesto por la tarjeta controladora de dirección, por un motor de corriente directa que permite girar las llantas delanteras y por un sensor. El controlador de dirección se encarga de calcular las manipulaciones, en el motor de dirección, para colocar las llantas en la posición deseada.

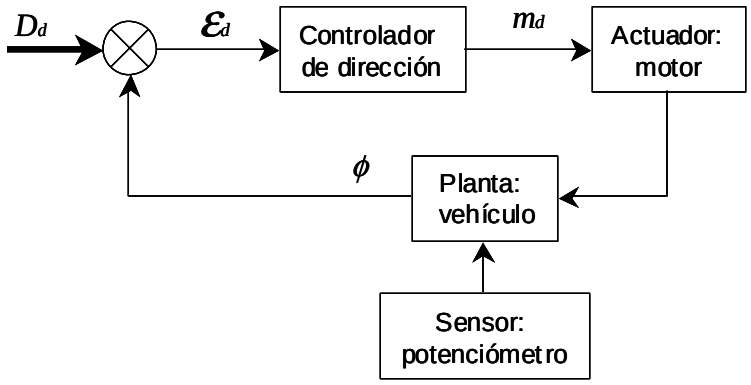


Figura 2.8: Diagrama de control de dirección donde D_d es la dirección deseada, ε_d es el error dirección, m_d es la señal de manipulación dirección, ϕ es el ángulo real de inclinación de las llantas con respecto al plano (x, y)

En la figura 2.8 se muestran las variables controladas en el módulo de dirección. El sistema se compone de una entrada y una salida: la entrada a la planta es un voltaje y la salida es la inclinación de las llantas, la cual se mide con un potenciómetro. La inclinación de las llantas

depende directamente de la posición de la flecha del motor. El controlador utilizado es de tipo PD debido a que debe mandar una manipulación de cero al obtener cero error [Palacios 00].

2.3 Protocolo de comunicación

Para el manejo de datos, se definió un protocolo de comunicación que permite que los datos enviados por *Controlador Anfitrión* sean interpretados por el *Vehículo*.

El *Controlador Anfitrión* y el *Vehículo* se comunican por Radio Frecuencia (RF), a través de dos canales independientes para la transmisión de datos e imágenes. La transmisión de imágenes de la cámara que se encuentra sobre el *Vehículo* al *Controlador Anfitrión*, se realiza en una dirección tanto para la teleoperación como para la operación autónoma.

Durante la operación autónoma, la transmisión de datos del *Controlador Anfitrión* al *Vehículo* es en un sentido. En la teleoperación, se utiliza un canal bidireccional "*half-duplex*" para la transmisión de datos.

La transmisión de datos "*half-duplex*" significa que los datos fluyen en ambas direcciones pero no al mismo tiempo; por lo tanto, se requiere optimizar el uso del canal de comunicación, minimizando los tiempos de ocio en el canal donde no hay transferencia de información. Además se requiere evitar que ambos lados de comunicación soliciten enviar la información al mismo tiempo, ya que si esto ocurriera, se perdería toda la información. Para resolver este problema, se le dio prioridad a la transmisión de instrucciones del *Controlador Anfitrión* al *Vehículo* por lo que el *Vehículo* sólo manda datos en modo síncrono cuando se le solicite.

2.3.1 Descripción del protocolo para la transmisión de datos

Para la transmisión de las instrucciones del *Controlador Anfitrión* al *Vehículo*, el formato con el que envía el paquete de datos es mostrado en la figura 2.9.

Header1	Comando	Valor1	Tail1

Figura 2.9: Formato de envío de instrucciones del *Controlador Anfitrión* al *Vehículo*

donde "*Header1*" y "*Tail1*" son constantes previamente establecidas utilizadas para filtrar los errores. Si estos valores son correctos se considera que el comando es válido. Actualmente, todas las instrucciones sólo requieren de un "*byte*" de comando y otro "*byte*" de valor.

En la tabla 2.2 se muestra el código de identificación de los principales comandos utilizados y los valores que pueden tomar. Todos los datos que se manejan son convertidos al tipo "*byte*" antes de ser enviados.

El comando "Pedir-Dato", utilizado para solicitar al *Vehículo* que mande los datos de retroalimentación, no tiene valor, por lo que se envía un cero.

El comando "Cambiar-Sentido" puede tomar 3 valores: reversa, adelante o detenido. Dicho comando se envía siempre antes del comando de "Cambiar-Velocidad", para establecer la dirección de avance del *Vehículo*.

El comando "Cambiar-Velocidad" envía el PWM utilizado para controlar la velocidad. Aunque este comando puede tomar valores de 0 a 255, para uso práctico, sólo se utilizan valores de 0 a 125, ya que entre mayor sea la velocidad, es más difícil controlar al *Vehículo*.

El comando "Cambiar-Dirección" envía la dirección que se desea tome el *Vehículo*. El rango de valores que toma el *Vehículo* es de 80 hasta 180; sin embargo, el centro de la dirección es 128, por lo que tiene mayor rango de valores hacia la derecha.

El paquete de datos mostrado en la figura 2.9 y los comandos de la tabla 2.2, con excepción del "Pedir-Dato" se utilizan tanto para la teleoperación como para la operación autónoma.

Tabla 2.2: Principales comandos utilizados

Comando	**Id**	**Valor**
Pedir-Dato	0x4C	0x00
Cambiar-Sentido	0x21	0 Reversa 1 Adelante 2 Detenido
Cambiar-Velocidad	0x27	0 a 255 (dividido entre 3.5)
Cambiar-Dirección	0x05	Desde 80 (izquierda) a 127 128 (centro) de 129 hasta 180 (derecha)

El protocolo implementado para asegurar que los caracteres enviados por el *Vehículo* sean recibidos correctamente por el *Controlador Anfitrión* es el de CRC (*Cyclic Redundancy Codes*). Para determinar si el mensaje está corrupto, el transmisor construye un valor llamado "*ckecksum*" que es una función de los valores de los caracteres del mensaje y lo agrega al mensaje. El receptor usa la misma función para calcular el valor del mensaje recibido y lo compara para ver si fue recibido correctamente. En este caso, se utilizo como CRC la suma de los datos.

El formato del paquete de envío de los datos obtenidos por el *Controlador Anfitrión*, cuando los envía el *Vehículo*, se muestra en la figura 2.10. Este paquete de datos sólo se utiliza en el modo teleoperado. El Dato1 es la distancia que el *Vehículo* ha avanzado en el eje *X*, el Dato2 es la distancia avanzada en el eje *Y* y el Dato3 es el ángulo de inclinación θ.

Header2	Dato1	Dato2	Dato3	Acknowledged (ACK)	CRC	Tail2

Figura 2.10: Formato de los datos enviados por el *Vehículo* al *Controlador Anfitrión*

donde: CRC = Dato1 + Dato2 + Dato3.

El ACK, que es la última instrucción recibida por el *Vehículo,* no se incluye en el CRC para que el cálculo del CRC no se vea afectado en el caso de que se haya perdido la última instrucción.

2.3.2 Funcionamiento del protocolo

Las tres partes principales de las que esta compuesto el protocolo utilizado para la comunicación del *Controlador Anfitrión* al *Vehículo* son: el envío de instrucciones por parte del *Controlador Anfitrión*, la ejecución de instrucciones por parte del *Vehículo* y la recepción de la retroalimentación en el *Invitado*. El algoritmo de control que se ejecuta en el *Anfitrión*, es una maquina de estados mostrada en la figura 2.11; conforme se cumplen ciertas condiciones durante la operación, se ejecutan tareas diferentes de acuerdo al estado del *Vehículo*.

En la figura 2.11 se muestran los diferentes estados de los algoritmos de control. Cada ovalo contiene las tareas ejecutadas en un determinado estado, mientras que junto a las flechas se especifican las condiciones de falso (F) o verdadero (T) que deben cumplirse para efectuar una transición y cambiar de estado.

El *Controlador Anfitrión* inicia la comunicación con el *Vehículo*, después de un tiempo determinado (500 ms) verifica si hay una instrucción que mandar; de ser así, envía la instrucción para que el *Vehículo* la ejecute; si no la hay, manda la instrucción "Pide dato" para recibir la retroalimentación de los sensores. El tiempo de espera antes del envío de la instrucción fue determinado experimentalmente.

El *Vehículo* verifica que la instrucción tenga el formato deseado y se encuentre dentro de su grupo de instrucciones; de ser así, cambia el valor del ACK con la última instrucción recibida, ejecuta la instrucción y espera que el *Controlador Anfitrión* le envíe la siguiente instrucción. Si el dato no tiene el formato correcto se considera como ruido, se descarta y espera la siguiente instrucción.

Cuando la instrucción fue "Pedir-Dato", el *Vehículo* envía como datos el avance en "X", en "Y" y el valor del ángulo "θ". Junto con los datos se envía un ACK que es el último comando recibido por el *Vehículo*, para verificar si perdida de la última instrucción y el CRC que es igual a la suma de los datos.

Cuando el *Controlador Anfitrión* recibe los datos del *Vehículo* verifica que el CRC sea correcto y que el ACK sea la última instrucción que se mandó, antes del "Pide dato". En caso de que la información no estuviera correcta, o no se haya recibido la última instrucción, se le manda un mensaje al *Invitado* para que decida si desea repetir el último comando.

No se utilizó el envío del último comando de forma automática, debido a que el *Vehículo* ya no se encuentra en la misma posición que cuando se envió y posiblemente ya no se desee que reciba ese comando sino algún otro.

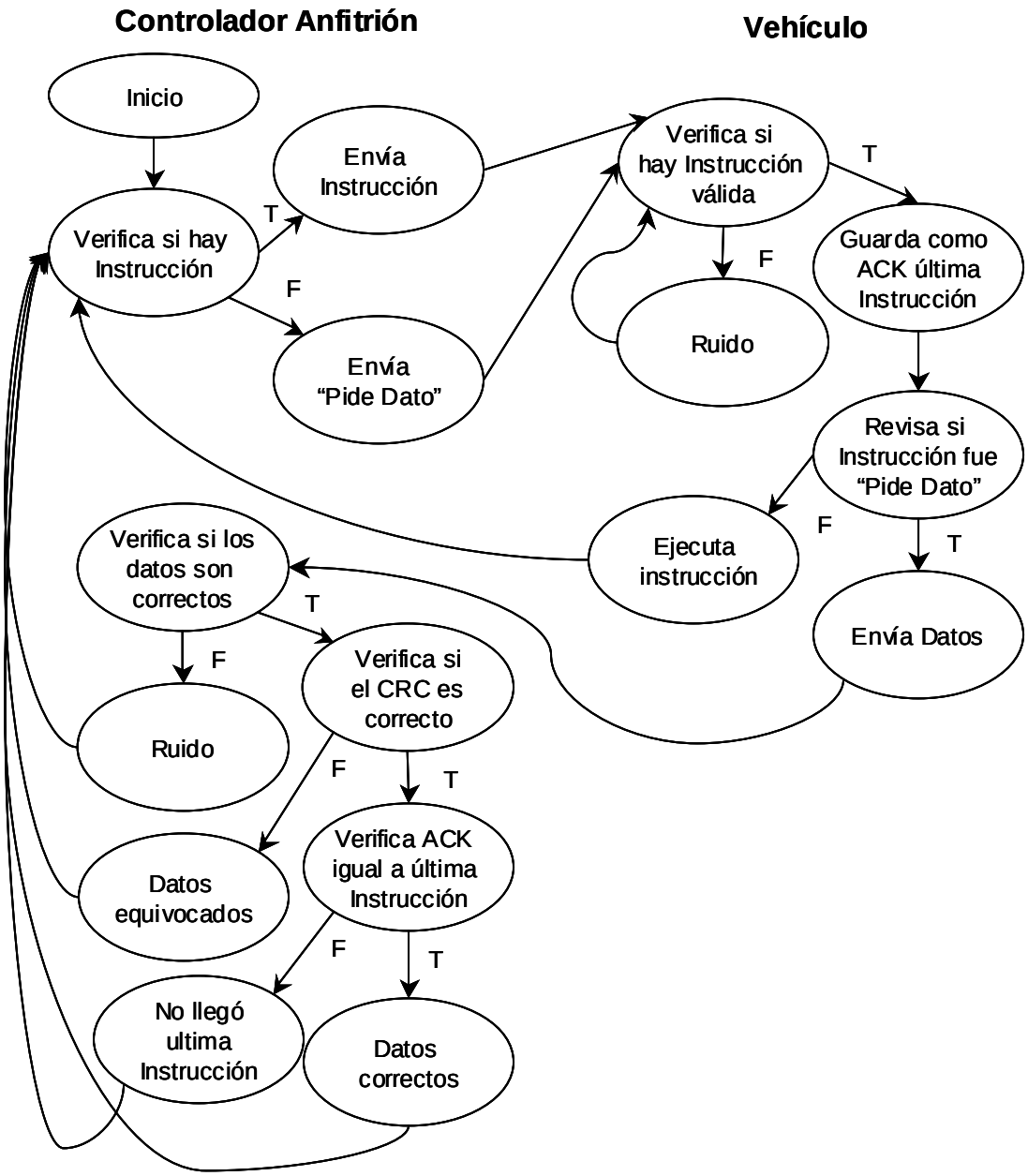


Figura 2.11: Funcionamiento del protocolo utilizado para la transmisión de datos utilizando una máquina de estados.

2.4 Funcionamiento general del *Vehículo*

El *Vehículo* cuenta con dos modos de operación: la teleoperación y la operación autónoma. En esta sección se describe de forma general el funcionamiento del sistema para que un usuario controle al *Vehículo*, ya sea teleoperándolo o solicitando alguna actuación autónoma según se muestra en la figura 2.12.

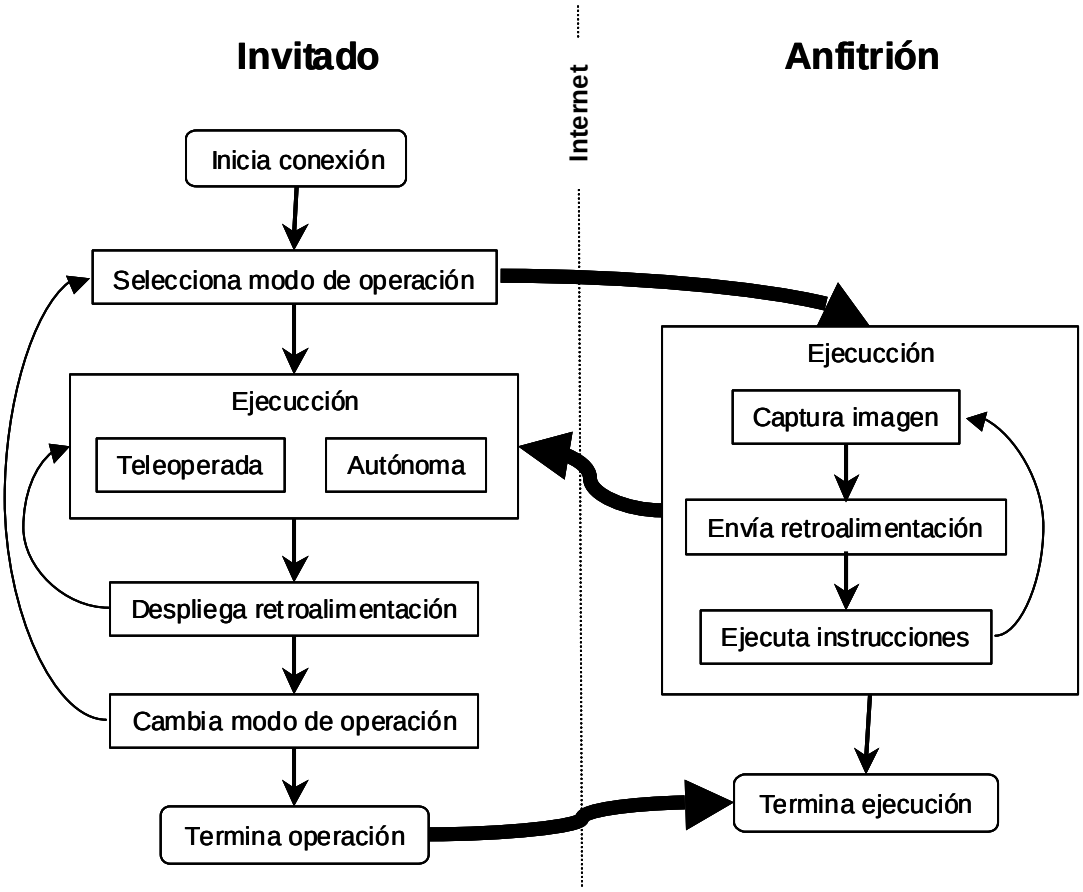


Figura 2.12: Funcionamiento general del sistema utilizado para controlar un *Vehículo* a distancia. El *Invitado* establece la conexión con el *Anfitrión*

El *Invitado* establece la conexión con el *Anfitrión*, selecciona el modo de operación y lo envía al *Anfitrión* para iniciar el ciclo de ejecución de acuerdo al modo de operación seleccionado. En la sección 2.4 se describe detalladamente el comportamiento teleoperado; mientras que el comportamiento autónomo es el tema del Capítulo 3.

En el caso mas general, el ciclo de ejecución del *Anfitrión*, consiste en la captura de la imagen de la cámara, que se encuentra sobre el *Vehículo*, el envío de retroalimentación al *Invitado* y la ejecución de las instrucciones recibidas.

Durante el ciclo de ejecución, el *Invitado* recibe la retroalimentación enviada por el *Anfitrión*. El ciclo de ejecución del *Invitado* y del *Anfitrión* se termina cuando el usuario, a través del *Invitado,* cambia el modo de operación o termina de operar al *Vehículo*.

Una vez terminada la ejecución, el *Invitado* se desconecta y el *Anfitrión* espera por una conexión con un nuevo usuario.

La nomenclatura utilizada en las graficas presentadas en los siguientes capítulos para mostrar el flujo de la información se describe en la figura 2.13.

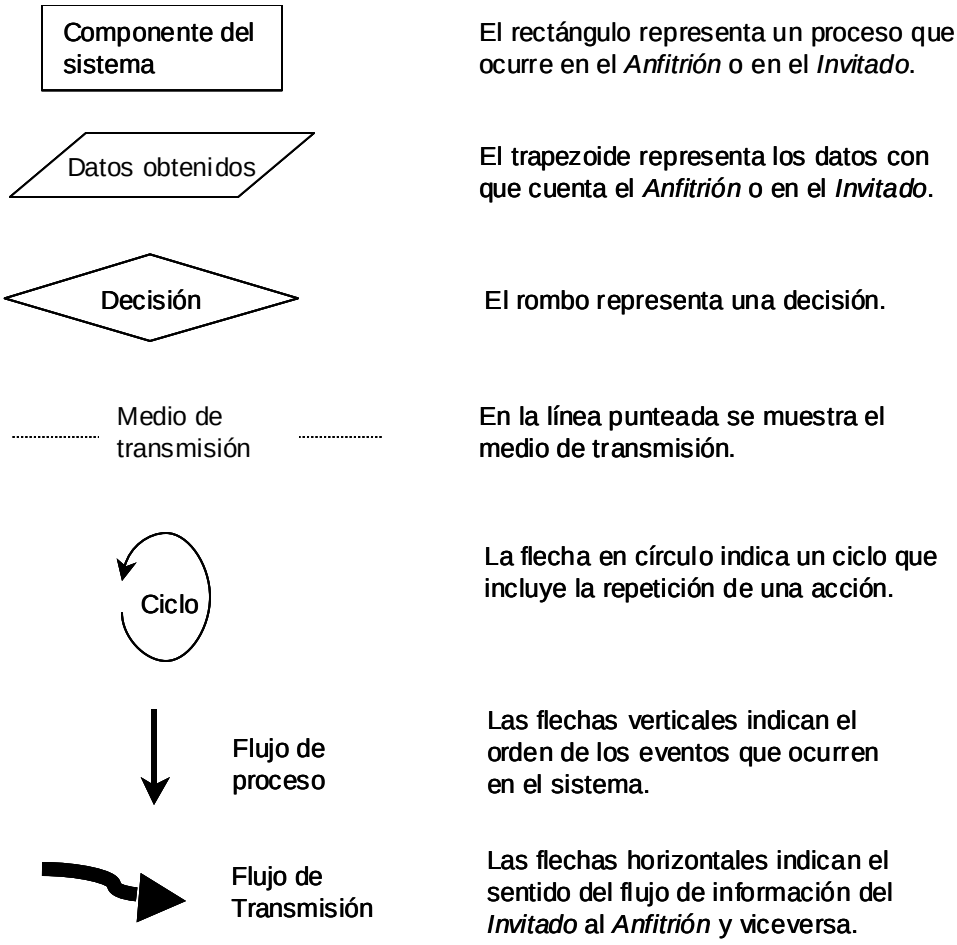


Figura 2.13: Formato del flujo de la información

2.5 Comportamiento Teleoperado

La teleoperación se refiere a la operación a distancia de un aparato mecánico o dispositivo para la realización de una tarea. En la minería se puede consideran las tareas de exploración y excavación que resultan riesgosas para la vida humana y pueden ser ejecutadas usando un *Vehículo* cuya operación no requiera la intervención directa del conductor.

La teleoperación manual es el modo de operación básico de un *Vehículo* automatizado, ya que el usuario controla los movimientos del *Vehículo*, con un ratón o una palanca. Para ello, el usuario utiliza la información recibida del espacio de trabajo, que consiste en un ambiente no estructurado. En este tipo de ambiente no se cuenta con conocimiento previo sobre su estructura por lo que es imposible modelarlos con precisión.

Para la teleoperación se usa un lazo de control cerrado mostrado en la figura 2.14, con retroalimentación visual, donde físicamente el usuario cierra el lazo de control. El control de la velocidad y de la dirección se encuentran en lazos cerrados independientes en cascada.

Las instrucciones enviadas por el usuario consisten en consignas de velocidad y dirección simultáneamente. Además de las imágenes, el usuario recibe como retroalimentación, la distancia que el *Vehículo* ha avanzado en X, Y y el ángulo de inclinación θ del *Vehículo* con respecto al plano.

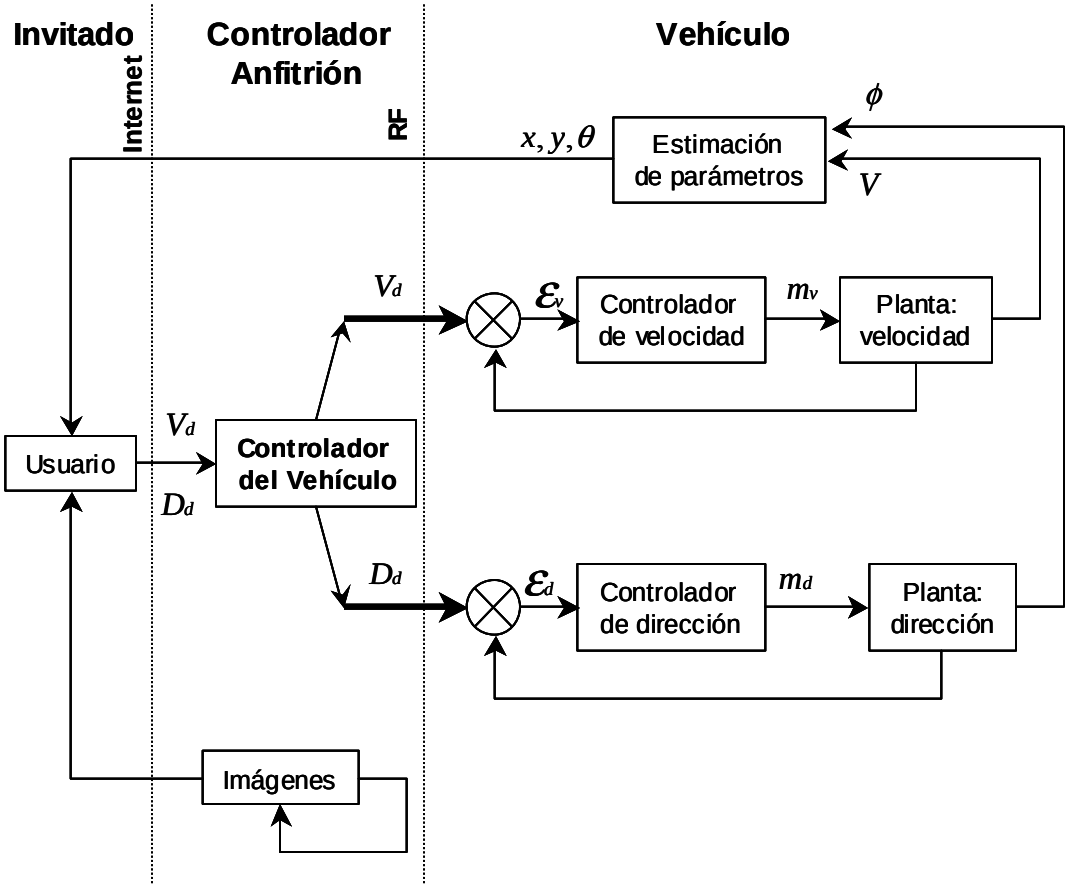


Figura 2.14: Diagrama de control de la teleoperación. El usuario controla al *Vehículo* utilizando la información recibida del espacio de trabajo

Los parámetros que se controlan en la teleoperación son la velocidad deseada V_d y la dirección deseada D_d. $\mathcal{E}_v$ es el error velocidad, m_v es la señal PWM de manipulación de velocidad, V es la velocidad real, $\mathcal{E}_d$ es el error dirección, m_d es la señal de manipulación dirección, ϕ es el ángulo real de inclinación de las llantas con respecto al plano (x, y). De acuerdo a la instrucción recibida, el algoritmo de control envía la señal de voltaje PWM a los motores para controlar la velocidad o la dirección del *Vehículo*. Cada controlador, de velocidad y de dirección , actúa separadamente según se describió en las secciones 2.2.2 y 2.2.3.

Una vez que el usuario ha enviado la instrucción a través del *Invitado* y que el *Controlador Anfitrión* la envía al *Vehículo*, el *Controlador del Vehículo* calcula el error entre el valor deseado y el actual y envía la manipulación (PWM) a los actuadores para corregir la velocidad o dirección del *Vehículo*.

Además de los controladores de velocidad y dirección, en la figura 2.14 se encuentra un bloque donde se estiman los parámetros X, Y, θ que son enviados al usuario en el caso que solicite la retroalimentación de datos. Sin embargo, los datos no son necesarios para teleoperar al *Vehículo* ya que la información visual es la mayor fuente de información entre todos los sensores utilizados hasta la fecha [Borestein 96], debido a que provee una gran cantidad de información del medio ambiente del *Vehículo*.

2.5.1 Funcionamiento del sistema teleoperado

En la figura 2.15 se muestra la descripción general del funcionamiento del sistema teleoperado. En esta forma de operación, el usuario tiene el control total del *Vehículo*.

Una vez iniciada la teleoperación, el usuario recibe como retroalimentación las imágenes de la cámara que se encuentra sobre el *Vehículo*. Como se observa en la figura 2.15, el *Invitado* inicia la teleoperación. La cámara que se encuentra sobre el *Vehículo* toma una secuencia de imágenes que son transmitidas al *Anfitrión*, el cual se encuentra en un ciclo esperando esta secuencia.

Al recibir la secuencia de imágenes, el *Invitado* las muestra en la pantalla del usuario. De acuerdo a la imágenes recibidas, el *Invitado* toma una decisión sobre la instrucción que debe ser enviada al *Vehículo*. Después de un determinado tiempo, si hay alguna instrucción, en el *Controlador Anfitrión*, se envía al *Anfitrión*, en caso contrario, manda pedir los datos de los sensores del *Vehículo* como retroalimentación de datos para el usuario.

Por su parte el *Controlador Anfitrión* se encuentra en un ciclo de espera; en el momento en que recibe la instrucción, la manda ejecutar al *Vehículo*; si se pidió el envío de la retroalimentación, se envían los datos al *Invitado* para que los muestre al usuario.

Debido a las limitantes que se tienen en los transmisores, se optimizó el uso del canal estableciendo un intervalo de tiempo fijo entre el envío de instrucciones al *Vehículo*. Recordemos que las instrucciones son consignas que guían al *Vehículo*, referentes a su velocidad y dirección.

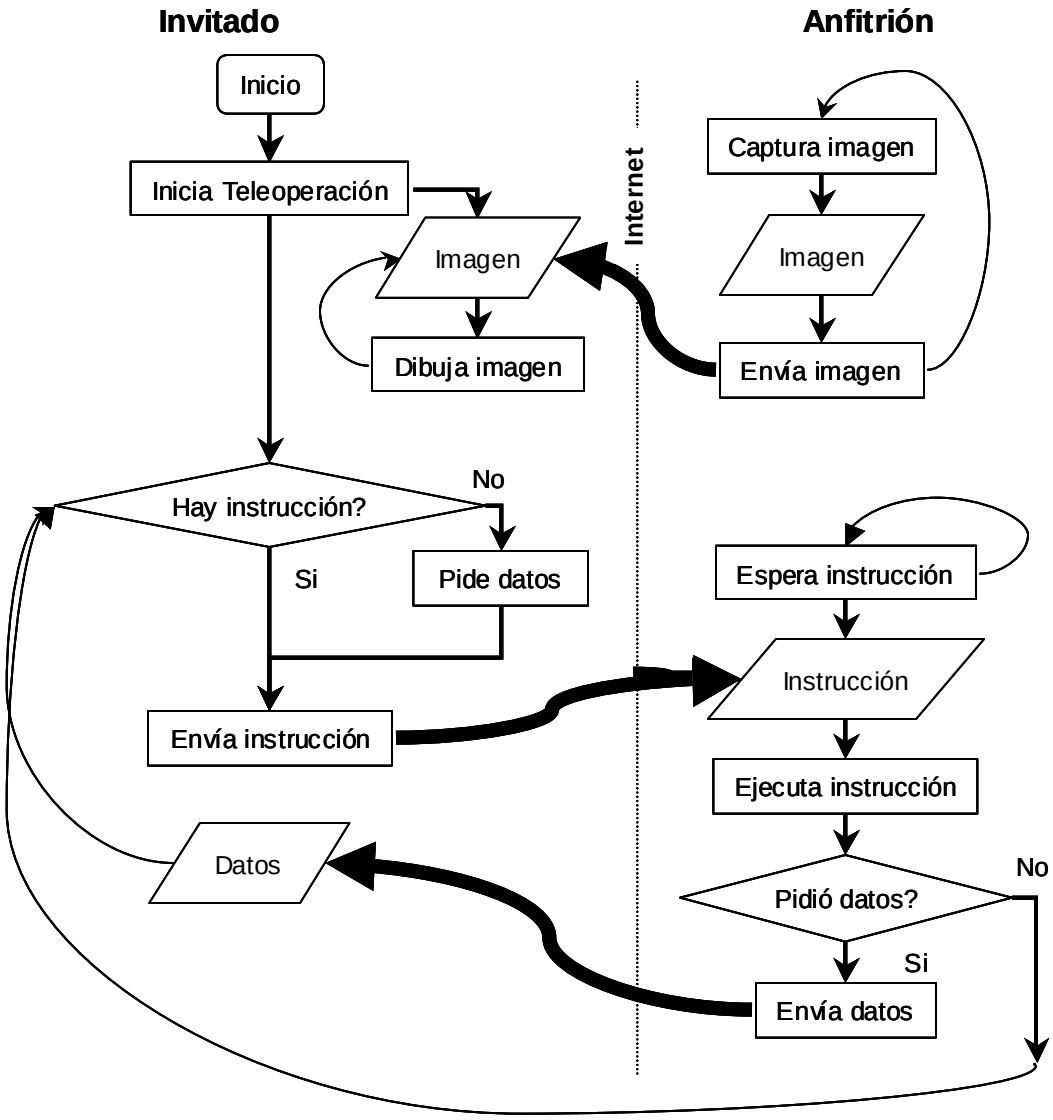


Figura 2.15: Descripción del funcionamiento del Sistema Teleoperado

Si el usuario no tiene ninguna instrucción que enviar, se solicitan los datos de los sensores para que cuente con otra retroalimentación. Sin embargo la prioridad la tiene el envío de instrucciones.

Capítulo 3

Autonomía

Cuando las tareas son repetitivas, durante la teleoperación, el usuario debe repetirlas manualmente; lo cual a la larga, provoca cansancio, perdida de concentración y errores. Además, considerando el retraso por la comunicación, si la tarea requiere retroalimentación inmediata y el tiempo de respuesta es crítico, el sistema se vuelve inoperable.

Se puede considerar la autonomía, como la capacidad de realizar una tarea específica con mínima o nula intervención humana utilizando la percepción de los sensores y el conocimiento previamente adquirido. Un sistema autónomo combina la percepción y el control con la capacidad de acción. La autonomía descarga al usuario del control de los detalles, permitiéndole concentrarse en el alto nivel de la tarea.

Combinar la teleoperación con la operación autónoma, permite al *Invitado* teleoperar al *Vehículo* para llevarlo a la posición deseada; una vez ahí, manda el inicio de la operación autónoma para que el *Controlador Anfitrión* tome el control del *Vehículo* e inicie la ejecución de la tarea hasta que el *Invitado* decida terminar este modo de operación. Se considera que la autonomía del *Vehículo* empieza cuando el *Invitado* libera el control del *Vehículo* y termina cuando, una vez finalizada la tarea, el *Controlador Anfitrión* devuelve el control al *Invitado*.

Para la operación autónoma, se realizó un ejercicio que consiste en el seguimiento de una marca artificial con consignas de velocidad y sentido descrito enseguida; en este ejercicio no se modificó la dirección. Se utilizó la visión como la principal forma de percepción.

3.1 Modelo General del Sistema Autónomo

En la figura 3.1 se muestra el modelo general del sistema autónomo. Para ejecutar el experimento, primero se selecciona la marca que se desea utilizar y se entrena al reconocedor fuera de línea.

Las marcas son distintos rasgos que un robot puede reconocer por sus sensores de entrada, por ejemplo esquinas o figuras geométricas. Las marcas son de dos tipos: naturales, las cuales

se encuentran en el medio ambiente, o artificiales las cuales son especialmente diseñadas y necesitan ser colocadas en el medio ambiente con el propósito de ayudar a la navegación del robot. Para este experimento se utilizan marcas artificiales, debido a que se conoce la forma y tamaño exacto y son diseñadas para contrastar con el medio ambiente facilitando su detección.

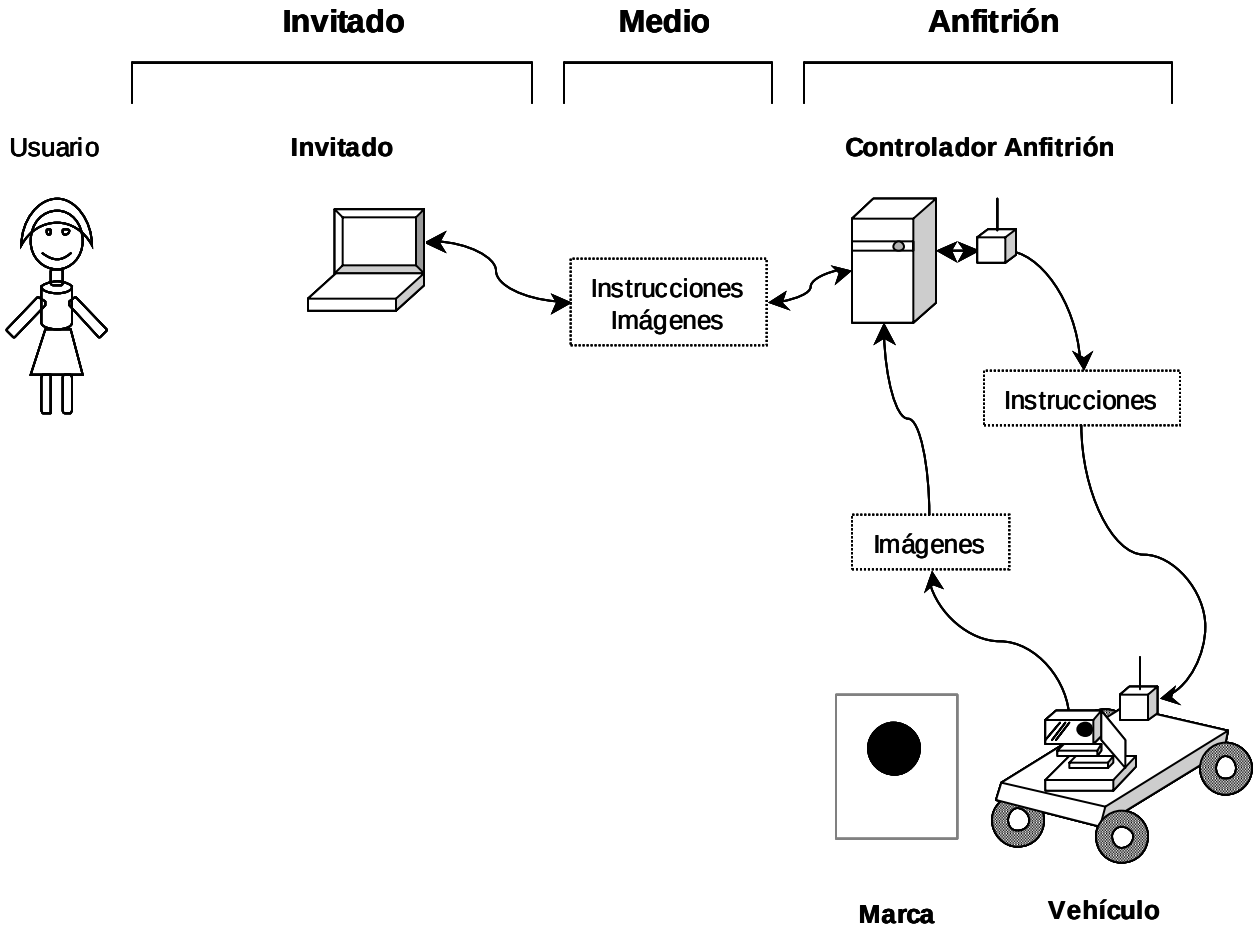


Figura 3.1: Modelo general del comportamiento autónomo para el seguimiento de una marca artificial con consignas de velocidad y sentido. El *Controlador Anfitrión* controla el avance y retroceso del *Vehículo* para que siga una marca artificial.

La cámara que se encuentra sobre el *Vehículo* envía las imágenes del área de trabajo al *Controlador Anfitrión*, para que dicho controlador decida que acción se debe tomar de acuerdo a la posición de la marca. Las acciones de control son convertidas a instrucciones del protocolo de comunicación que utiliza el *Vehículo*.

Las instrucciones obtenidas por el *Controlador Anfitrión* son enviadas al *Vehículo* para ser ejecutadas y al *Invitado* como retroalimentación. El *Invitado* también recibe las imágenes en tiempo real del área de trabajo.

Los algoritmos de control de velocidad y sentido descritos en el Capítulo 2, utilizados para la teleoperación, también son utilizados en el comportamiento autónomo. La dirección no se controla en este modo de operación.

El control del *Vehículo* en la operación autónoma utilizado para efectuar la tarea propuesta de seguimiento de una marca con consignas de velocidad y sentido, utiliza un lazo de control cerrado mostrado en la figura 3.2. El *Controlador Anfitrión* cierra el lazo de control y el usuario sólo actúa como supervisor. La retroalimentación recibida por el usuario es la imagen procesada por el *Controlador Anfitrión* y las instrucciones enviadas al *Vehículo*.

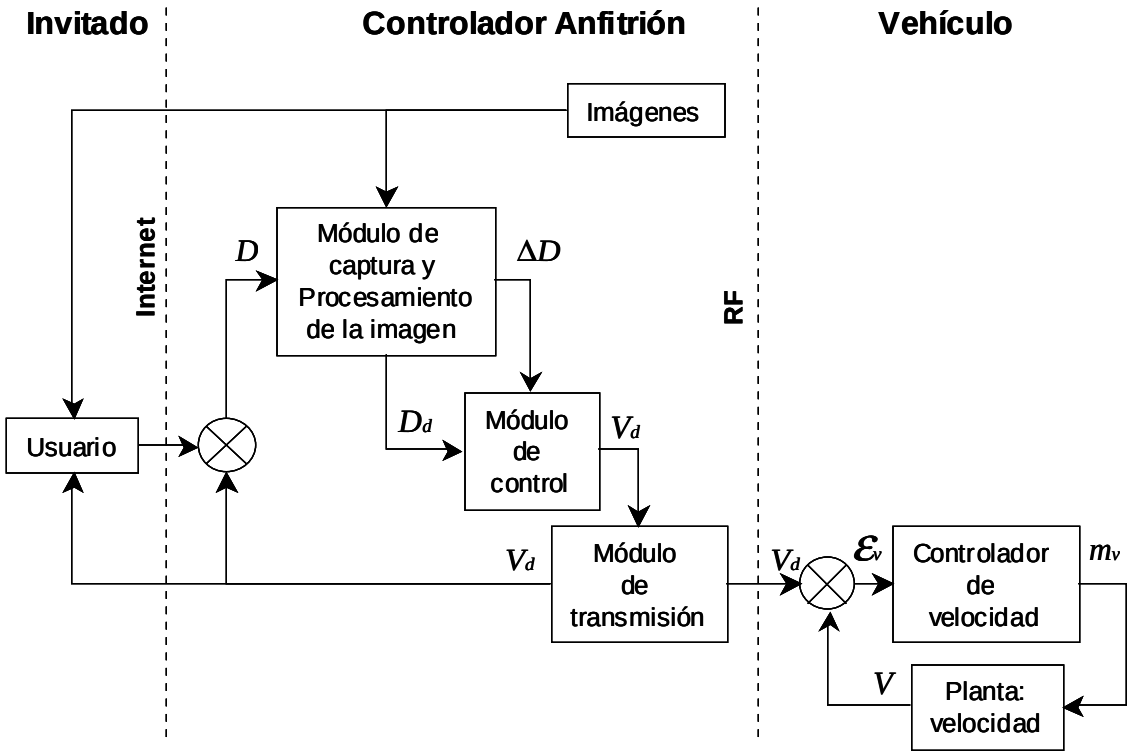


Figura 3.2: Diagrama de las variables controladas del *Vehículo* en la operación autónoma. En este modo de operación el *Controlador Anfitrión* tiene el control del *Vehículo* y el usuario solo actúa como supervisor.

El sistema de control que reside en el *Controlador Anfitrión* está compuesto de tres módulos.

Módulo de Captura y reconocimiento: La cámara que se encuentra sobre el *Vehículo* captura la imagen y la envía al *Controlador Anfitrión*, quien la procesa para buscar la marca deseada (previo entrenamiento); al encontrarla calcula el área de la marca y arroja como

resultado un número que representa la distancia del carro a la marca D_d. El sistema tiene almacenada la distancia del estado anterior y la distancia a la que debe permanecer el Vehículo de la marca D.

Módulo de Control: El sistema analiza el comportamiento para determinar los cambios necesarios en la velocidad y sentido del *Vehículo*, para no perder la marca. Teniendo como entradas la distancia actual D_d así como la diferencia entre la distancia actual y la distancia anterior ΔD, se obtiene la velocidad V_d y sentido (avance o retroceso) que debe tomar el *Vehículo* para seguir la marca.

Módulo de Transmisión: Los datos de velocidad V_d y sentido, obtenidos por el controlador, son convertidos a instrucciones del protocolo de comunicación. Dichas instrucciones son enviadas al *Vehículo* para su ejecución.

El parámetro que se controla en la operación autónoma es la velocidad deseada V_d donde ε_v es el error velocidad, m_v es la señal PWM de manipulación de velocidad, V es la velocidad real.

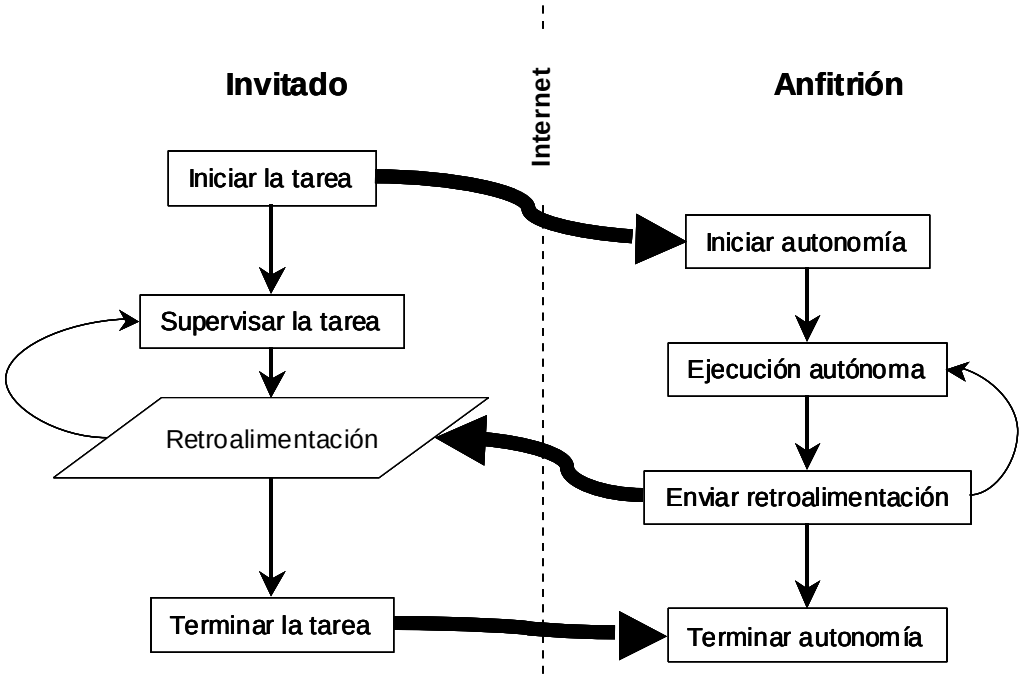


Figura 3.3: Funcionamiento general del comportamiento autónomo. El *Invitado* es el encargado de iniciar o terminar la operación.

En la figura 3.3 se observa el funcionamiento general del comportamiento autónomo. El *Invitado* interviene para iniciar o terminar la operación autónoma, ya que la función del *Invitado* en este modo de operación, solamente es supervisar la ejecución de la tarea basándose en la retroalimentación recibida y no inteviene en el control.

Para iniciar la operación autónoma, el *Invitado* envía la instrucción de inicio de la ejecución autónoma, para que el *Vehículo* la ejecute. El *Anfitrión* entra un ciclo de envío de instrucciones al *Vehículo* y de retroalimentación al *Invitado*. El *Invitado* recibe como retroalimentación la imagen en tiempo real transitada por la cámara que se encuentra sobre el *Vehículo* y los datos obtenidos por el controlador después del procesamiento de la imagen.

En la figura 3.4 se describen las acciones ocurridas en el *Anfitrión* durante el ciclo de ejecución autónoma y envío de retroalimentación.

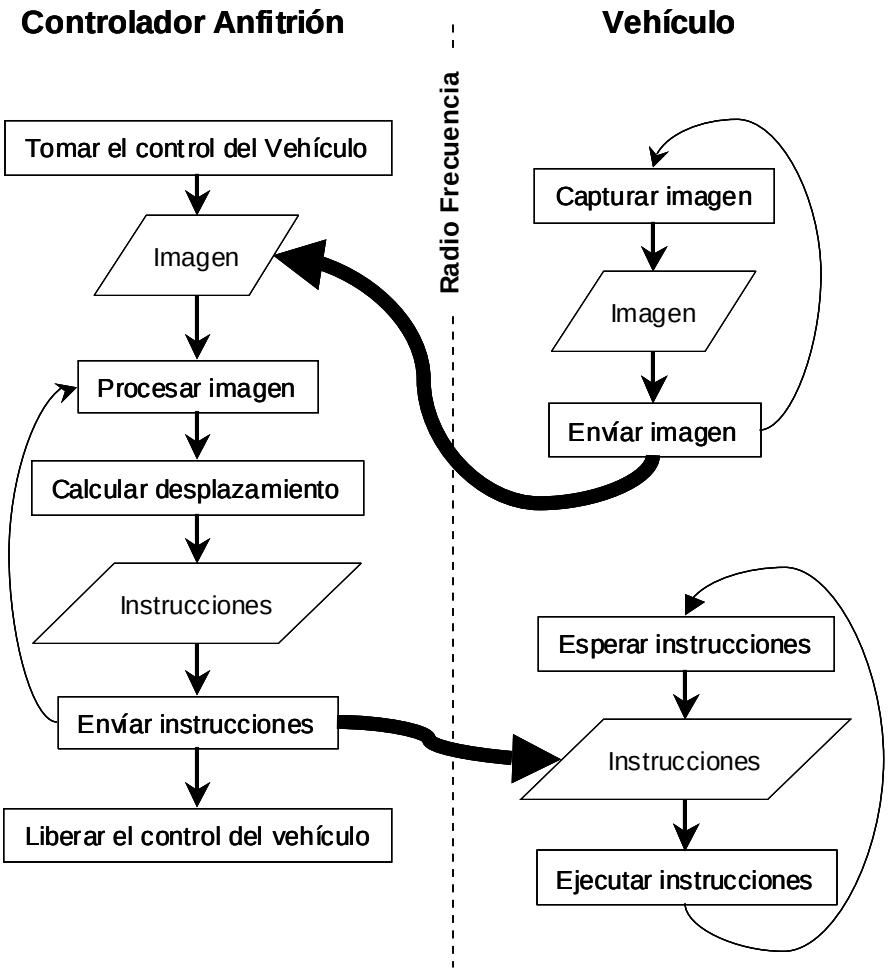


Figura 3.4: Ciclo de ejecución autónoma y de envío de retroalimentación realizado en el *Anfitrión*

En la ejecución autónoma el *Controlador Anfitrión* tiene el control total del *Vehículo*. La cámara, que se encuentra sobre el *Vehículo*, captura la imagen y la envía al *Controlador Anfitrión* para que la procese y determine si contiene la marca deseada. De ser así, el *Controlador Anfitrión* calcula la distancia a la que se encuentra el *Vehículo* de la marca. Se guarda el valor de la distancia anterior para ver como varía la posición de la marca en el tiempo.

Utilizando estas distancias, el controlador obtiene la velocidad y sentido que el *Vehículo* necesita moverse para mantenerse a la distancia establecida de la marca. Las instrucciones son enviadas al *Vehículo* para ser ejecutadas.

Una vez que el *Vehículo* ejecuta la instrucción, espera la siguiente. La operación autónoma finaliza cuando el *Invitado* solicita retomar el control del *Vehículo*, ya sea para pasar a modo teleoperado o para finalizar.

Durante la operación autónoma, la comunicación de datos entre el *Controlador Anfitrión* y el *Vehículo* es en una sola dirección.

3.2 Captura y procesamiento de la imagen

Se desarrolló un programa para reconocer figuras geométricas. El programa tiene como entrada la imagen de la cámara que está sobre el *Vehículo* y proporciona la posición y la escala de la figura dentro de la imagen.

Las características con las que cuenta el programa reconocedor son las siguientes:

- El sistema reconocedor sólo se utiliza para reconocer círculos, sin embargo puede ser entrenado para reconocer cualquier figura geométrica.
- La cámara debe permanecer perpendicular al plano de la figura a seguir.
- Sólo se controla el avance y el retroceso del *Vehículo*.

El *Controlador Anfitrión* se encarga de ejecutar todas las etapas de captura y procesamiento de la imagen mostradas en la figura 3.5, aunque la cámara se encuentra sobre el *Vehículo*.

Adquisición de la imagen: La primera etapa del proceso es la captura de una imagen digital, para lo cual se necesita un sensor de imágenes y la posibilidad de digitalizar la señal producida por el sensor. Digitalizar es el proceso de convertir una imagen de su forma original a la forma digital.

La cámara que se encuentra sobre el *Vehículo* toma una imagen que es transmitida al *Controlador Anfitrión* de forma analógica utilizando un cable coaxial.

El *Controlador Anfitrión* recibe la imagen y por medio de una tarjeta de video convierte la imagen recibida en formato NTSC en una imagen digital, para procesarla. El tipo de imagen utilizada es "*raw data*" debido a que no tiene compresión. La imagen es capturada en niveles de gris de tamaño 320x240 *píxeles*.

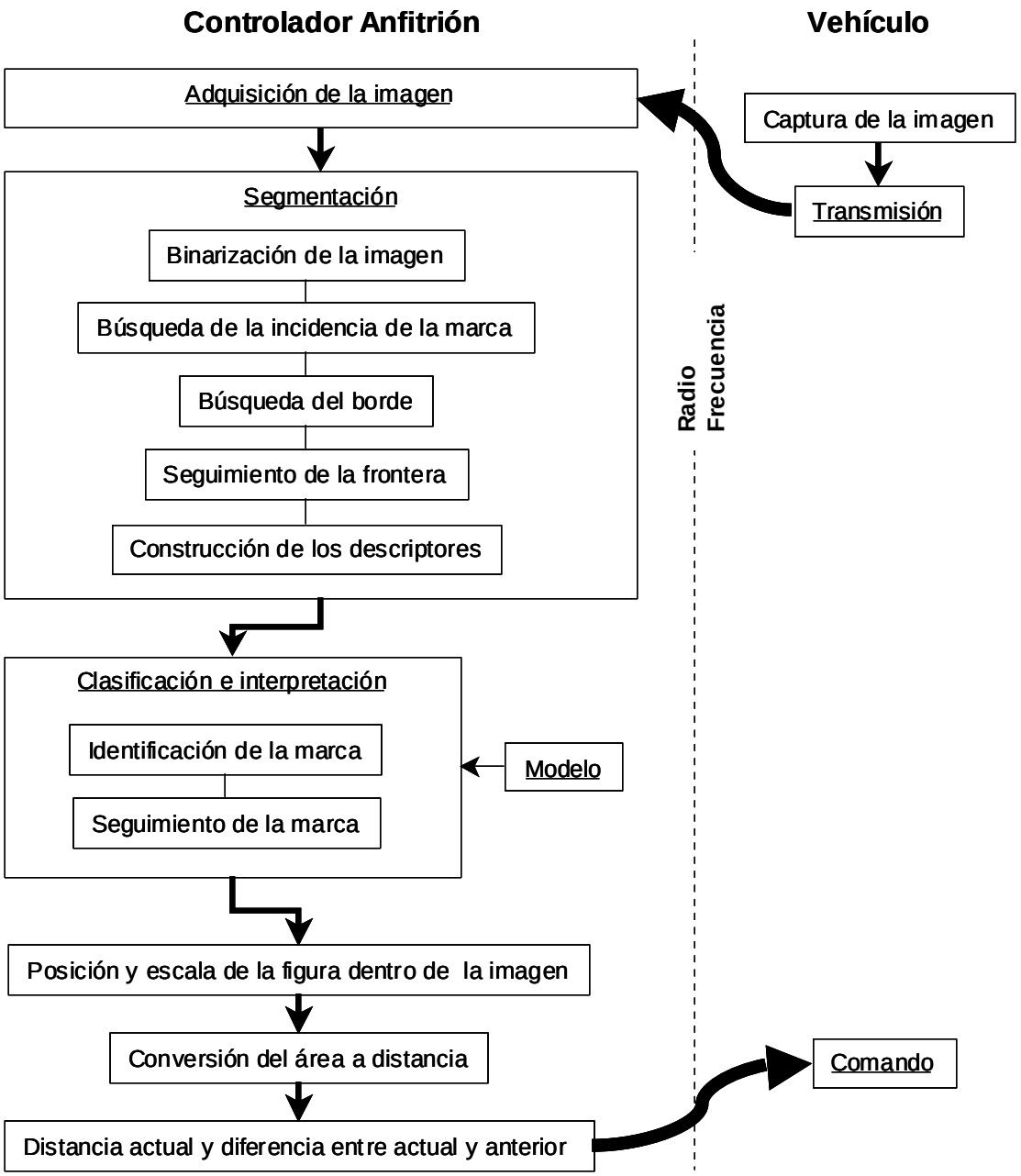


Figura 3.5: Módulo de captura y procesamiento de la imagen. Este módulo nos permite obtener la distancia actual de la marca a partir de la imagen recibida.

Segmentación: La segmentación consiste en partir una imagen de entrada en sus partes constituyentes u objetos. La descripción o selección de rasgos, consiste en extraer rasgos con alguna información cuantitativa de interés o que sean fundamentales para diferenciar una clase de objetos de otra.

La segmentación es una de las etapas más importantes, del procesamiento de la imagen pues permite obtener los descriptores que determinan si se encuentra presente la marca

buscada dentro de la imagen. Uno de los descriptores utilizados es el área de marca, el otro de los descriptores es la compacidad.

Clasificación e interpretación: El reconocimiento es el proceso que asigna una etiqueta a un objeto basándose en la información proporcionada por sus descriptores. La interpretación implica asignar significado a un conjunto de objetos reconocidos. El conocimiento sobre un dominio está codificado como una base de datos de modelos donde radican las formas conocidas.

En esta etapa, se utiliza una base de datos de los objetos conocidos para compararlos con el objeto que se va a reconocer, por lo tanto, un paso necesario antes de realizar la clasificación e interpretación consiste en entrenar al sistema para obtener el modelo de la marca deseada.

El entrenamiento es el proceso mediante el cual el sistema memoriza los objetos que podrá reconocer. Primeramente se verifica si el objeto ya existe, si es así se actualizan los valores de los descriptores mediante un promedio ponderado entre el valor ya existente y en nuevo valor obtenido. Si el objeto no existe, se crea una nueva clase a la que se asignan los valores de los descriptores del objeto. En nuestro caso, el sistema está entrenado para reconocer un círculo.

Finalmente, durante la operación de reconocimiento, el sistema compara los valores de los descriptores obtenidos de la imagen con los que se tenían del entrenamiento y determina si la marca deseada se encuentra dentro de la imagen

Conversión del área a distancia: es la última etapa del módulo de captura y procesamiento. Si el sistema encontró que dentro de la imagen se encuentra la marca deseada, se obtiene la distancia que hay entre la marca y el *Vehículo*, la cual es utilizada por el controlador. Si no encontró la marca, el sistema ignora la imagen actual y procede a analizar la siguiente imagen.

3.3 Segmentación de la imagen

El primer paso para analizar la una imagen, consiste en segmentar la imagen [González 92]. El objetivo de la segmentación es extraer un conjunto de características que faciliten el reconocimiento de los objetos segmentados, así como la información relacionada con su ubicación (posición y orientación) dentro de la imagen.

Una imagen se define como la proyección de una escena sobre un plano. La intensidad luminosa I de un punto (x, y) como función de respuesta a una longitud de onda del espectro luminoso λ, se representa por:

$$I = f(x, y) \tag{3.1}$$

donde $Z^2 \rightarrow Z$

A la intensidad de una imagen monocromática f en las coordenadas (x, y) se le denomina nivel de gris de la imagen en este punto. Las imágenes que se manejan en este trabajo cuentan con 256 niveles de gris.

Una imagen digital es una imagen (x, y) que se ha discretizado tanto en las coordenadas espaciales como en la intensidad. Una imagen digital puede considerarse como una matriz cuyos índices de fila y columna identifican un punto de la imagen y el valor del correspondiente elemento de la matriz indica el nivel de gris en ese punto. Cada punto de la imagen se denomina elemento de la imagen o *píxel* lo cual es una abreviatura de "*picture elements*".

Cuando se haga referencia a un píxel en particular se emplearan letras minúsculas, como p y q. Un subconjunto de *pixeles* de $f(x, y)$ se indicara como S. Por lo tanto:

$$p = (x_p, y_p),\ q = (x_q, y_q) \tag{3.2}$$

En este capítulo se va a utilizar terminología probabilística en lugar de terminología estadística debido a que en el área de visión computacional es lo usual.

3.3.1 Binarización de la imagen y búsqueda de la incidencia de la marca

La segmentación comienza con la binarización, cuyo objetivo es establecer el umbral de corte T que divida en dos grupos los píxeles que son totalmente blancos de los que son totalmente negros, de esa forma se separan los píxeles del fondo de los píxeles del objeto.

Se usará el histograma, como la herramienta para calcular el umbral ideal T.

Suponiendo una imagen $B(x, y)$ en escala de grises, a los píxeles que tienen altos valores de gris se les da el valor binario de 1.

$$B(x, y) = \begin{cases} 1 & si \quad f(x) \geq T \\ 0 & de \quad otra \quad forma \end{cases} \tag{3.3}$$

En la figura 3.6 se muestra la imagen de la marca utilizada.

El histograma de niveles de gris es una gráfica que representa, para cada nivel de gris, el número de píxeles relativos al total que tienen ese nivel. El eje X representa los valores de los niveles de gris desde lo oscuro (0) hasta lo claro (255); el eje Y representa la frecuencia (el número total de píxeles con un valor determinado).

Si se supone que el histograma de nivel de gris de la figura 3.7 corresponde a la imagen $f(x, y)$ de la figura 3.6, una forma de extraer el objeto del fondo es elegir un umbral T que separe la figura del fondo; así cualquier punto (x, y) para el que $f(x, y) > T$ corresponde al objeto, en caso contrario, se trata de un punto del fondo. La parte de la izquierda del

histograma son los píxeles antes del umbral (objeto), y la parte de la derecha son los píxeles después del umbral (fondo).

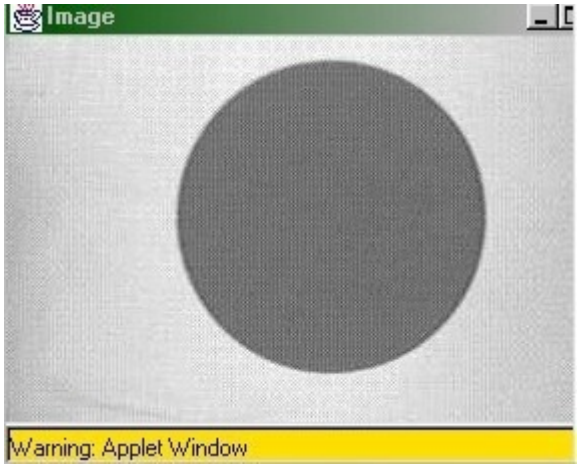


Figura 3.6: Imagen original de la marca utilizada

El algoritmo de Otsu es un método para encontrar el umbral de binarización y depende solamente de parámetros estadísticos contenidos dentro del histograma [Otsu 79]. El primer paso para aplicar el algoritmo de Otsu consiste en obtener el histograma de la imagen. El histograma de una imagen digital se define como la función $f(r_k) = a_k$ que existe para cada uno de los L valores discretos de gris r_0, r_1 … r_{L-1}, donde a_k es el espacio de acumulación con $k \in [0, L]$, que describe el número de elementos de la imagen (píxeles) con valor r_k.

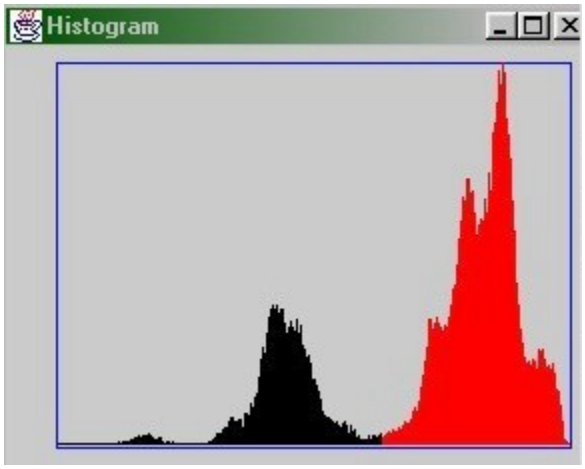


Figura 3.7: Obtención del histograma de la imagen mostrada en la figura 3.7. La parte izquierda (negra) del histograma nos muestra los píxeles del objeto y la de la derecha (roja) los del fondo.

El histograma absoluto se define por:

$$N = \sum_{k=0}^{L-1} a_k \tag{3.4}$$

donde N representa el número de píxeles en la imagen, L es el número de niveles y a_k es el número de veces que este nivel aparece en la imagen.

En el histograma probabilístico para los niveles de gris que constituyen los valores discretos, se tienen las probabilidades

$$f(r_k) = P_r(r_k) = \frac{a_k}{N} \quad 0 \le r_k \le 1, \quad k \in [0, L] \tag{3.5}$$

donde $P_r(r_k)$ es la probabilidad del k-ésimo nivel de gris.

Una vez que se tiene el histograma, se calcula la masa, la media y la varianza del histograma completo y se obtiene $\omega(t), \mu(t)$ y ${\sigma_B}^2(t)$ respectivamente para toda $1 \le t \le L$

Considerando que $\omega(t)$ son los píxeles que representan al fondo se tiene la siguiente ecuación:

$$\omega(t) = \sum_{i=1}^{t} P_r(r_k) \tag{3.6}$$

se puede obtener los píxeles del objeto de acuerdo al complemento de la ecuación 3.4:

$$1 - \omega(t) = 1 - \sum_{k=t+1}^{L} P_r(r_k) \tag{3.7}$$

Una vez que se tienen estos datos, se calculan las medias:

$$\mu_f = \frac{\sum_{k=1}^{t} kP_r(r_k)}{\sum_{k=1}^{k} P_r(r_k)} = \frac{\mu(t)}{\omega(t)} \tag{3.8}$$

$$\mu_o = \frac{\sum_{k=t+1}^{L} kP_r(r_k)}{\sum_{k=t+1}^{L} P_r(r_k)} = \frac{\sum_{k=1}^{L} kP_r(r_k) - \sum_{k=1}^{t} kP_r(r_k)}{1 - \omega(t)} = \frac{\mu_T - \mu(t)}{1 - \omega(t)} \tag{3.9}$$

donde μ_f es la media del fondo, μ_o es la media del objeto y μ_T es la media de la imagen completa, y es igual a:

$$\mu_T = \sum_{k=1}^{L} kP_r(r_k) \tag{3.10}$$

Para obtener las varianzas se tienen las siguientes ecuaciones:

$$\sigma_f^2 = \frac{\sum_{k=1}^{t}(k-\mu_f)^2 P_r(r_k)}{\sum_{k=1}^{t} P_r(r_k)} = \frac{1}{\omega(t)}\sum_{k=1}^{t}(k-\mu_f)^2 P_r(r_k) \tag{3.11}$$

$$\sigma_o^2 = \frac{\sum_{k=t+1}^{L}(k-\mu_f)^2 P_r(r_k)}{\sum_{k=t+1}^{L} P_r(r_k)} = \frac{1}{1-\omega(t)}\sum_{k=t+1}^{L}(k-\mu_f)^2 P_r(r_k) \tag{3.12}$$

donde σ_f^2 es la variancia del fondo y σ_o^2 es la varianza del objeto. De igual forma se tiene la varianza total definida como:

$$\sigma_T^2 = \sum_{k=1}^{L}(k-\mu_T)^2 P_r(r_k) \tag{3.13}$$

que puede ser descompuesta en dos sumatorias:

$$\sigma_T^2 = \sum_{k=1}^{t}(k-\mu)^2 P_r(r_k) + \sum_{k=t+1}^{L}(k-\mu)^2 P_r(r_k) \tag{3.14}$$

La expresión final se reduce a:

$$\sigma_T^2 = \underbrace{\omega(t)\sigma_f^2 + (1-\omega(t))\sigma_o^2}_{\sigma_\omega^2} + \underbrace{(\mu_f-\mu_T)^2\omega(t) + (\mu_o-\mu_T)^2(1-\omega(t))}_{\sigma_B^2} \tag{3.15}$$

donde σ_ω^2 representa las varianzas dentro de las clases (*within*), mientras que σ_B^2 representa la varianza entre clases (*between*). Como σ_T^2 es constante, y deseamos que la varianza entre las clases sea lo más pequeña posible, entonces la varianza dentro de las clases debe ser maximizada. Por lo tanto:

$$\sigma_B^2 = (\mu_f-\mu_T)^2\omega(t) + (\mu_o-\mu_T)^2(1-\omega(t)) \tag{3.16}$$

a partir de ahí y utilizando las ecuaciones anteriores, se obtiene obtiene la varianza $\sigma_B^2(t)$ que es la función a evaluar utilizada directamente sobre un histograma previamente calculado.

$$\sigma_B{}^2(t) = \frac{[\mu_T \omega(t) - \mu(t)]^2}{\omega(t)[1-\omega(t)]} \tag{3.17}$$

Una vez que se tiene la varianza, se selecciona el mayor resultado para *k*, y el valor de *t* cuyo $\sigma_B{}^2$ es máximo, debe ser escogido como el umbral. La imagen de la figura 3.8 nos muestra la imagen binarizada después de haber aplicado el algoritmo de Otsu.

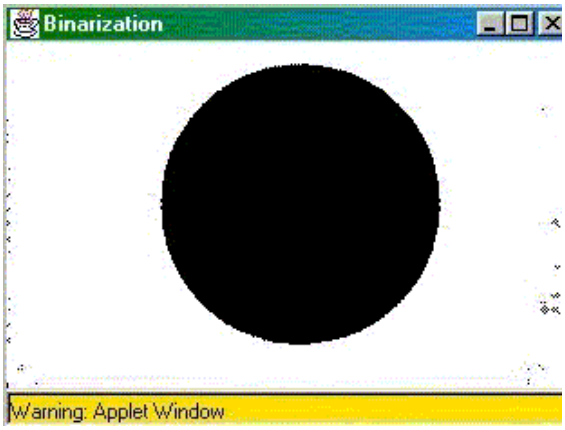


Figura 3.8: Obtención de la imagen binarizada al aplicar el algoritmo de Otsu.

3.3.2 Búsqueda del borde, seguimiento de frontera y cálculo de los momentos

El propósito de esta sección es limitar la figura mediante el seguimiento de su contorno; una vez que se tiene la imagen en blanco y negro, el siguiente paso es buscar la incidencia de la marca, para lo que se escoge de forma aleatoria un píxel dentro de la marca y si no es negro, se escoge otro hasta que se encuentre un píxel negro. Una vez que se tiene un píxel del objeto, el siguiente paso es avanzar hasta llegar a un punto de la frontera del objeto.

El teorema de Green descrito en la fórmula 3.18 convierte dobles integrales a integrales de área sobre el plano, que calculan propiedades de la región englobada por la curva al recorrer la frontera; supongamos el área

$$A = \frac{1}{2}\sum_{i=1}^{N_b}[x_i(y_{i+1} - y_i) - y_i(x_{i+1} - x_i)] \tag{3.18}$$

donde A es el área de la figura y N_b es el número de puntos en la frontera.

De acuerdo a la figura 3.9, si se considera el punto *C* como punto de inicio del recorrido de la frontera, al avanzar de *C* a *B* se consideran valores positivos, y en el recorrido de *B* a *C* se consideran valores negativos, lo cual nos permite conocer el área de la región englobada por la curva.

Una vez que se tiene un píxel del borde del objeto, la cadena de Freeman se utiliza para describir el contorno de un objeto conforme al siguiente formato:

$$S = \{s,(a_1, a_2,...a_n)\} \tag{3.19}$$

donde $s = (x, y)$ es un punto arbitrario de inicio y, a_i (i = 0....n) es una secuencia de direcciones.

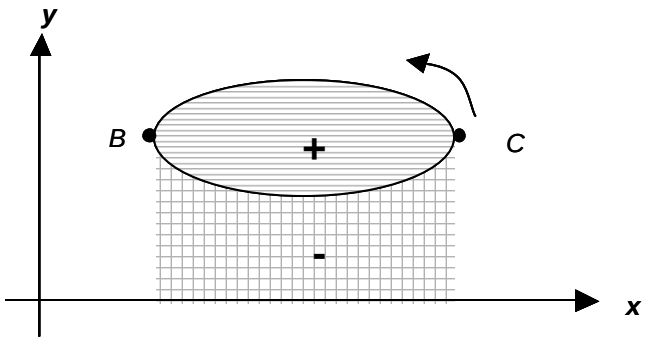


Figura 3.9: Teorema de Green utilizado para obtener el área de la región englobada por la curva

Un píxel p de coordenadas (x, y) tiene cuatro vecinos horizontales y verticales cuyas coordenadas vienen dadas por

$$(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1) \tag{3.20}$$

este conjunto de píxeles es denominado los 4-vecinos de p. Cada píxel está a una distancia unitaria de (x, y). Los cuatro vecinos en diagonal de p tienen coordenadas

$$(x + 1, y + 1), (x + 1, y - 1), (x - 1, y + 1), (x - 1, y - 1) \tag{3.21}$$

estos puntos, junto a los 4-vecinos se denominan los 8-vecinos de p. La distancia d_8 también llamada distancia de tablero de ajedrez entre p y q se define como

$$d_8(p,q) = \max\{\ |x_p - x_q|, |y_p - y_q|\ \} \tag{3.22}$$

los píxeles con una distancia $d_8 = 1$ son los 8-vecinos de (x, y).

En este caso se utiliza vecindad 8, por lo que se consideran todos los vecinos como parte del vecindario. Los códigos de cadena son usados para representar un borde por una secuencia conectada de segmentos de línea recta de longitud y dirección específica. En la figura 3.10, se muestra el arreglo con las direcciones que toma la cadena de Freeman en vecindad 8. El contorno del objeto es seguido en el sentido de las manecillas del reloj.

3	2	1
4	P	0
5	6	7

Figura 3.10: Código de Freeman utilizado para seguir el contorno de un objeto. El contorno es seguido en el sentido de las manecillas del reloj.

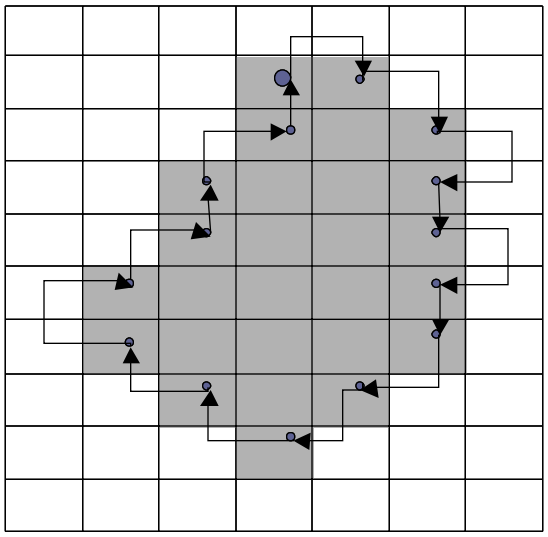


Figura 3.11: Seguimiento de la frontera de un objeto en el sentido de las manecillas del reloj utilizando la cadena de Freeman

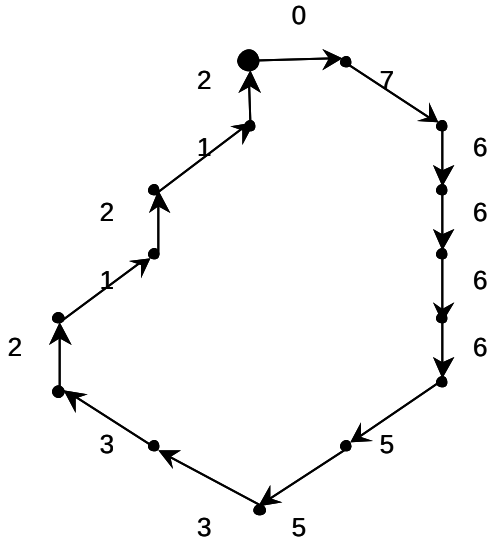


Figura 3.12: Construcción de la cadena de Freeman. Si iniciamos en el punto el código es 0-7-6-6-6-6-5-5-3-3-2-1-2-1-2

La figura 3.11 muestra como la cadena de Freeman describe el contorno del objeto. Suponiendo que a partir del píxel negro aleatorio obtenido llegamos al punto que forma parte de la frontera del objeto. Si al recorrer la frontera cruzamos por un píxel negro, se da una vuelta a la izquierda y se sigue con el siguiente píxel. Si ese píxel es negro de nuevo da vuelta a la izquierda y si ese píxel es blanco da vuelta a la derecha. El procedimiento continúa hasta que se regresa al punto de inicio [Pratt 97].

El código generado del borde mostrado en la figura 3.12, depende del punto de inicio, si iniciamos en el punto, al describir el contorno del objeto obtenemos que el código del borde es: 0-7-6-6-6-6-5-5-3-3-2-1-2-1-2.

3.3.3 Caracterización

Se busca una conversión de la figura a una descripción para su clasificación. Como la imagen analizada está en un plano, el conjunto de descripción, debe presentar invarianza respecto a cambios de posición, rotación y escalamiento de los objetos analizados.

Los momentos estadísticos nos ayudan a conocer el centro, la extensión, y la forma del objeto que estamos midiendo, en este caso, nuestras medidas son valores de píxel y posiciones.

Estos momentos de Hu describen mediante la manipulación algebraica de las muestras estadísticas una forma de reconocer patrones y tienen la característica de ser invariantes en rotación, translación y escala [Hu 62]. Los momentos de primer orden dicen que existe una mancha, los momentos de segundo orden indican la forma general la dispersión que tiene la figura bajo condiciones.

Los momentos centrales μ_{pq} son definidos como:

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\bar{x})^p(y-\bar{y})^q\, p(x,y)d_x d_y \tag{3.32}$$

donde,

$$\bar{x} = \frac{m_{10}}{m_{00}} \qquad \bar{y} = \frac{m_{01}}{m_{00}} \tag{3.33}$$

Los momentos centrales expresados en términos de los momentos ordinarios de orden 1, 2 y 3 se muestran a continuación:

La ecuación 3.32 representa la masa de la figura u objeto en la imagen.

$$\mu_{00} = m_{00} \tag{3.34}$$

$$\mu_{10} = 0 \tag{3.35}$$

$$\mu_{01} = 0 \tag{3.36}$$

$$\mu_{20} = m_{20} - \bar{x}^2 m_{00} \tag{3.37}$$

$$\mu_{02} = m_{02} - \bar{y}^2 m_{00} \tag{3.38}$$

$$\mu_{11} = m_{11} - \bar{x}\bar{y} m_{00} \tag{3.39}$$

$$\mu_{30} = m_{30} - 3\bar{x} m_{20} + 2\bar{x}^2 m_{10} \tag{3.40}$$

$$\mu_{03} = m_{03} - 3\bar{y} m_{02} + 2\bar{y}^2 m_{01} \tag{3.41}$$

$$\mu_{21} = m_{21} - 2\bar{x} m_{11} - \bar{y} m_{20} + 2\bar{x}^2 m_{01} \tag{3.42}$$

$$\mu_{12} = m_{12} - 2\bar{y} m_{11} - \bar{x} m_{02} + 2\bar{y}^2 m_{10} \tag{3.43}$$

Los momentos centrales son invariantes en translación; para que también sean invariantes en escala, se normalizan

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\left[\frac{p+q}{2}\right]+1}} \tag{3.44}$$

Los momentos de Hu invariantes en rotación se calculan a partir de los momentos normalizados, los más usados son los de primero y segundo orden,

$$\phi_1 = \eta_{20} + \eta_{02} \tag{3.45}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{3.46}$$

Otro método utilizado para parametrizar una forma es el cálculo de la compacidad,

$$C = \frac{\mathsf{L}^2}{A} \tag{3.47}$$

donde L es el perímetro y A es el área de la figura.

Los momentos y el valor del perímetro se calculan a partir de la frontera del objeto, tomando en cuenta que dos puntos de la frontera se pueden conectar topológicamente por segmentos que pasan por el interior del objeto [Gauthier 90].

En una dirección dada (X ó Y), y moviéndose del infinito hacia cero, se define un punto de entrada al objeto y un punto de salida del objeto. Los puntos entrada-salida en dirección de X

se caracterizan por su coordenada *Y* (llamada y_{pos}) y los puntos en dirección *Y* se caracterizan por su coordenada *X* (llamada x_{pos}).

Al punto de salida en la dirección *X* se le llama $x_{pos\min}$ y al punto de entrada se le llama $x_{pos\max}$. En la dirección *Y*, al punto de salida se le llama $y_{pos\min}$ y al punto de entrada $y_{pos\max}$.

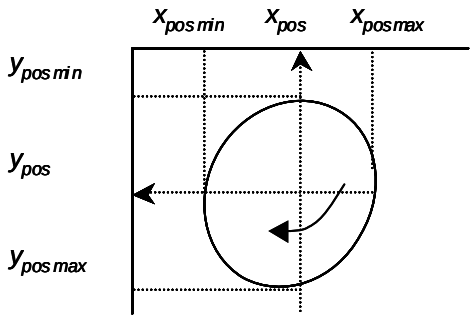


Figura 3.13: Búsqueda del borde y recorrido: los momentos son calculados a partir de la frontera del objeto

Los momentos ordinarios de orden m_{pq} se desarrollan basados en la figura 3.13 donde (x_i, y_j) es una posición en la imagen y $b(x_i, y_j)$ es el valor binario (fondo / objeto):

$$m_{pq} = \sum\sum x_i^p y_j^q b(x_i, y_j) = \sum x_i^p \sum y_j^q b(x_i, y_j) \tag{3.23}$$

para una x_i común (llamada x_{pos}) se tiene la suma:

$$\sum y_j^q b(x_{pos}, y_j) \tag{3.24}$$

la sumatoria se realiza considerando sólo los términos diferentes de cero, en el intervalo de coordenadas *Y* dados entre los puntos $y_{pos\min}$ y $y_{pos\max}$, rango en el cual $b(x_{pos}, y_j)$ pertenece al objeto (con valor de 1).

Por lo tanto:

$$\sum y_j^q b(x_{pos}, y_j) = \sum_{y_{pos\min}}^{y_{pos\max}} y_j^p = fq(y_{pos\max}) - fq(y_{pos\min}) \tag{3.25}$$

donde,

$$fq(y_{pos}) = \sum_{y=1}^{y_{pos}} y_j^q \tag{3.26}$$

Finalmente, para calcular los momentos ordinarios se tiene la siguiente ecuación:

$$m_{pq} = \sum x_i^p \sum y_j^q b(x_i, y_j) = x_{ipos}^p \left[fp(y_{pos\max}) - fp(y_{pos\min}) \right] \tag{3.27}$$

donde $y_{pos\max}$ y $y_{pos\min}$ dependen de x_{ipos}.

Recorriendo la frontera, se pueden encontrar todos los momentos de todos los órdenes m_{pq} encontrando la función f_k apropiada, resolviendo las sumatorias a una forma cerrada.

Para orden 0:

$$f_{0(n)} = \sum_{i=1}^{n} i^0 = n \tag{3.28}$$

Para orden 1:

$$f_{1(n)} = \sum_{i=1}^{n} i^1 = \frac{n(n+1)}{2} \tag{3.29}$$

Para orden 2:

$$f_{2(n)} = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \tag{3.30}$$

Para orden 3:

$$f_{3(n)} = \sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2} \right)^2 \tag{3.31}$$

3.3.4 Construcción de los descriptores

Los valores de $x_{pos\max}$ y $x_{pos\min}$ se definen a partir de la posición (x_i, y_i) actual de la frontera según recorran la curva en un sentido. Estando en una posición (x_i, y_i) y teniendo un código de cadena dado, el valor que se debe asignar a x_{pos} y y_{pos}, se encuentran mediante la tabla 3.1. Los valores resultantes se aplican a las funciones de los momentos.

Por ejemplo si el código de Freeman es 0 o 4, significa que el punto (x_i, y_i) avanza sólo sobre el eje *X*, por lo que $x_{pos} = 0$ y $y_{pos} = y_i$.

Tabla 3.1: Calculo de momentos para vecindad 8

Código	$\mathbf{x_{pos}}$	y_{pos}
0	-	y_i
1	$x_i + 0.5$	$y_i - 0.5$
2	x_i	-
3	$x_i - 0.5$	$y_i - 0.5$
4	-	y_i
5	$x_i - 0.5$	$y_i + 0.5$
6	x_i	-
7	$x_i + 0.5$	$y_i + 0.5$

3.3.5 Conversión de área a distancia

El dispositivo más simple de captura de imágenes es la cámara. El modelo de la cámara tiene una apertura muy pequeña "*pin-hole*" a través de la cual entra la luz a la cámara y forma una imagen en la superficie de la cámara. Geométricamente, la imagen esta formada por rayos de luz rectos que viajan del objeto a través de la apertura al plano de la imagen.

Una transformación de perspectiva (también denominada transformación de imagen) proyecta puntos del espacio tridimensional sobre un plano; estas transformaciones proporcionan una aproximación al modo en que se forma una imagen viendo un mundo tridimensional.

Los valores que se tienen a la salida del reconocedor son la posición y la escala (el área) de la figura dentro de la imagen. Conociendo el área real de la marca y el área calculada por el reconocedor, se calcula la distancia que existe entre el *Vehículo* y la marca; siendo así, la distancia es el parámetro utilizado para controlar al *Vehículo*.

Cada punto de la marca es proyectado a una superficie sobre una línea a través del punto focal. La superficie de proyección es un plano.

Utilizando la igualdad de triángulos se relacionan el área real de la marca (A_R) con el área de la marca en la imagen obtenida por el reconocedor (A_I), en función de la distancia real entre la marca y el punto focal (D) relativo a la distancia que ve la cámara (d), según se muestra en la figura 3.14.

$$\frac{A_I}{d^2} = \frac{A_R}{D^2} \tag{3.48}$$

donde A_I es el área de la imagen que ve la cámara, d es la distancia entre el punto focal y la imagen, A_R es el área real de la marca, D es la distancia real entre la cámara y la marca.

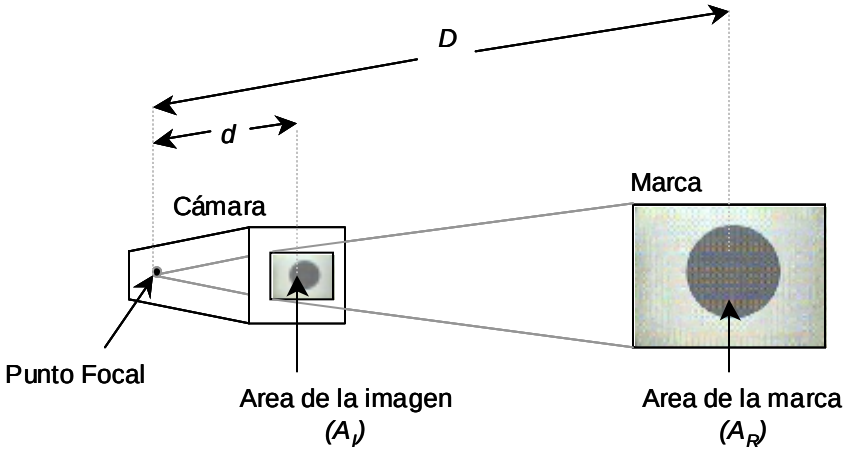


Figura 3.14: Conversión del área de la marca a distancia utilizando el valor real de la imagen y el área obtenida de la imagen recibida

Entonces, para fijar la distancia a la cual el *Vehículo* se mantiene de la marca, a partir del área obtenida por el reconocedor, se despeja:

$$D^2 = \frac{A_R}{A_I d^2} \tag{3.49}$$

Considerando una constante de escala S que depende de la figura a utilizar y es calculada mediante experimentación (colocando la marca a una distancia conocida y realizando medidas hasta obtener un valor estadístico satisfactorio),

$$S = \frac{\sqrt{A_R}}{d} \tag{3.50}$$

Finalmente, la siguiente relación calcula la distancia real entre el *Vehículo* y la marca (D), conociendo solamente el área obtenida por el reconocedor,

$$D = \frac{S}{\sqrt{A_I}} \tag{3.51}$$

La distancia real entre el *Vehículo* y la marca es transmitida a la etapa de control.

3.4 Control y transmisión de instrucciones

Una vez que se conoce la distancia a la que se encuentra el *Vehículo* de la marca, y la distancia que se obtuvo en la imagen anterior, se determinan las acciones que debe ejecutar el

Vehículo para que cumpla su objetivo. El propósito del controlador, es que el *Vehículo* siga a la marca.

En el módulo de transmisión, el *Controlador Anfitrión* es encargado de enviar los valores de velocidad y sentido obtenidos a la salida del modulo de control, con el formato de instrucciones para que el *Vehículo* las ejecute.

De acuerdo al procedimiento descrito en el Apéndice C, para obtener el modelo de control del vehículo se obtuvieron las ecuaciones mostradas a continuación.

$$\begin{cases} v = 104 - 1.4d & 20 \le d \le 50 \\ v = 0 & 50 < d < 70 \\ v = -73 + 1.5d & 70 \le d \le 100 \end{cases} \tag{3.52}$$

donde v es la velocidad enviada al Vehículo y d es la distancia entre el *Vehículo* y la marca.

A la entrada del controlador se tiene la distancia actual entre el *Vehículo* y la marca (d) y la diferencia entre la distancia actual y anterior (Δd) , y a la salida se obtiene la velocidad (v) y sentido.

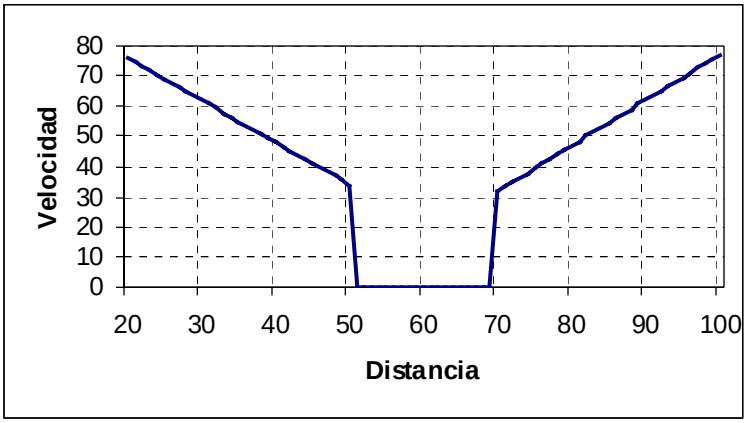


Figura 3.15: Controlador lineal de velocidad obtenido

Como se muestra en la figura 3.15, se estableció un umbral de los 50 a los 70 cm. donde el *Vehículo* permanece detenido. Si la distancia es menor que el umbral, se considera que el *Vehículo* está muy cerca, por lo que debe retroceder; si es mayor, el *Vehículo* esta muy lejos por lo que se debe acercar.

Capítulo 4

Implementación y Experimentación

En este capítulo se describen los componentes utilizados para la implementación del sistema propuesto, mostrados en la figura 4.1; así como la interfaz desarrollada para el control del *Vehículo* en las dos formas de operación: teleoperación y autonomía. Por último, se muestran los experimentos realizados para validar la propuesta de la tesis.

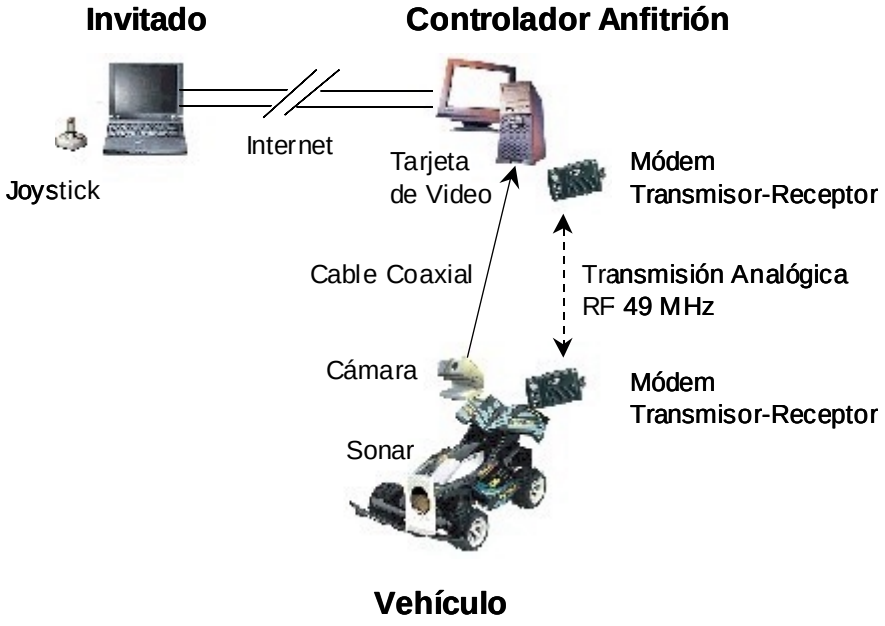


Figura 4.1: Componentes utilizados para implementar el sistema propuesto

En la figura 4.1, se observa que además de la computadora, el *Invitado* cuenta con una palanca de mando. Por su parte, el *Controlador Anfitrión*, cuenta con una tarjeta de captura de video y un módem; mientras que el *Vehículo* cuenta con una cámara, un sonar y un módem. Estos componentes son descritos en la siguiente sección.

4.1 Descripción del equipo utilizado

En esta sección se muestran las características del equipo utilizado, incluyendo el *Controlador Anfitrión*, el *Vehículo* y el *Invitado*.

4.1.1 *Anfitrión*

El *Controlador Anfitrión* reside una Sun Sparc Ultra 10 con procesador UltraSPARC-IIi a 333MHz y 320 MB de RAM, que se encuentra dentro del Laboratorio de Robótica Integrado a la Manufactura (LabRIM), que forma parte de la red del campus Monterrey. La dirección IP del *Controlador Anfitrión* es 131.178.63.16 y su DNS es "*dali.mty.itesm.mx*". El *Controlador Anfitrión* cuenta con una tarjeta de captura de video "*Sun Video*".

El *Vehículo* que se automatizó, se muestra en la figura 4.2; es un carro eléctrico a escala, de control remoto modelo "*Tempest*" de *RadioShack*. Este *Vehículo* cuenta originalmente con un control de velocidad mecánico, control de dirección mediante un servomotor y dispositivo receptor de radio y un emisor para su control remoto. Además, el *Vehículo* cuenta con tres motores: uno para controlar el avance o retroceso en el eje de las llantas traseras, otro para las llantas delanteras, y el último para dar vuelta a la derecha o izquierda los cuales le permiten alcanzar velocidades de hasta 6 metros por segundo.

Figura 4.2: *Vehículo* utilizado para la implementación. Para que el usuario tenga mayor conocimiento del ambiente remoto, se cuenta con una cámara montada sobre el *Vehículo*

Por sus características, se trata de un *Vehículo* no holonómico que se mueve hacia adelante y hacia atrás, además cambia de dirección a la derecha e izquierda, aunque por sus restricciones cinemáticas no se mueve de lado. Para cambiar de dirección sólo se utilizan las dos llantas delanteras, cuyo cambio de dirección tiene un límite mecánico. El *Vehículo* cuenta con un potenciómetro para medir el ángulo de dirección, y se agregó un codificador óptico para medir la velocidad y la distancia recorrida. Se cuenta con una cámara montada sobre el *Vehículo* para que el usuario tenga un mayor conocimiento del ambiente remoto.

Para tener mayor precisión en el control del *Vehículo*, se sustituyó la tarjeta de control general que venía integrada, por una tarjeta para controlar cada dispositivo (un módulo de velocidad, un módulo de dirección y un módulo de potencia para la alimentación de los circuitos y motores). En la figura 4.3 se muestran los componentes del *Anfitrión*, incluyendo al *Vehículo*.

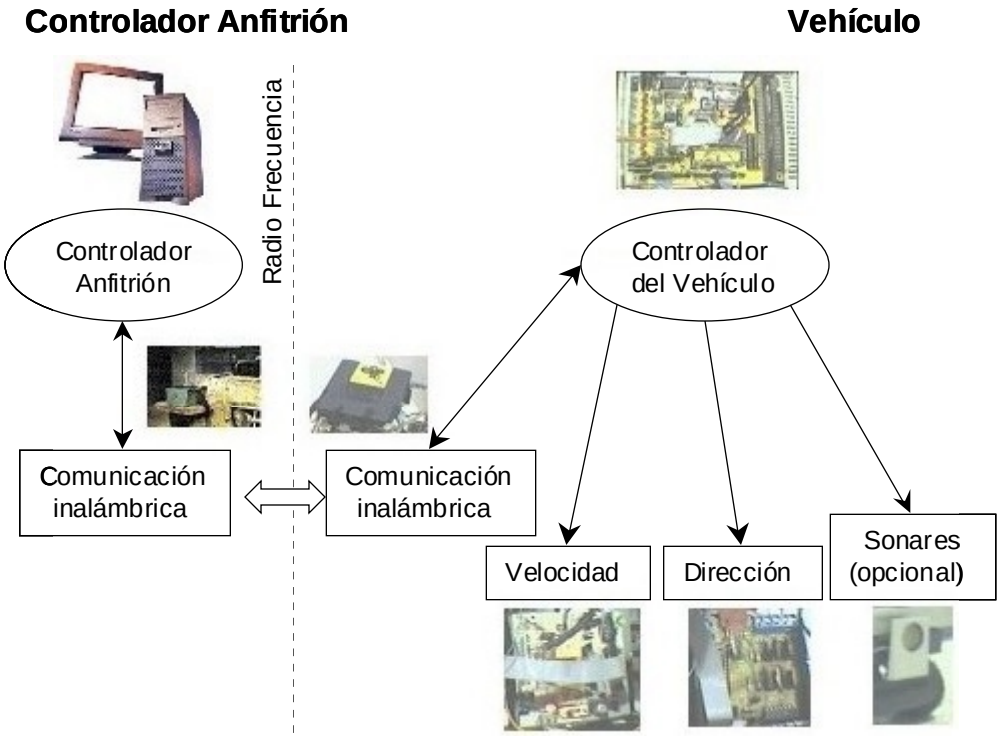


Figura 4.3: Diagrama que muestra los componentes del *Anfitrión* y del *Vehículo*

Se utiliza el microcontrolador "*Atmel 8515"* para el control de velocidad y dirección del *Vehículo*. Para la transmisión de datos, del *Vehículo* al *Controlador Anfitrión*, se construyeron dos módems Transmisor-Receptor que operan a 1200 bits por segundo, los cuales son descritos en el Apéndice A. Se realizaron pruebas para controlar al *Vehículo* en un radio máximo de 15 metros a línea de vista.

La alimentación de los circuitos utiliza un módulo de potencia compuesto de una tarjeta, a la que son conectadas 3 baterías de 7.2V, para alimentar a todas las tarjetas. Los motores se alimentan con 7.2V, el encoder con 5V. Una pila se usa para la tarjeta maestra y el módem, otra para alimentar los motores delanteros, y la última para el motor trasero.

El procesamiento de la imagen utiliza una cámara "*SunCamera II"* de CCD a color, con la tarjeta de compresión "*Sun Video*", la cual cuenta con un canal NTSC de entrada. La transmisión de imágenes, de la cámara que se encuentra sobre el *Vehículo*, a la tarjeta del *Controlador Anfitrión*, es en una sola dirección y se realiza utilizando un cable coaxial. Todo el procesamiento de las imágenes se realiza en el *Controlador Anfitrión.*

Las imágenes analizadas contienen 320x240 *píxeles* de resolución, en niveles de gris sin compresión. Para capturar la imagen se utiliza un método nativo de Java descrito en el Apéndice B, ya que la tarjeta de captura del *Anfitrión* se programa utilizando las librerías XIL, que son un conjunto de funciones propietarias de Sun bajo el sistema operativo Solaris. Estas funciones sirven para programar aplicaciones como compresión, manipulación y descompresión de imágenes o video; las cuales son llamadas como subrutinas de C.

4.1.2 *Invitado*

El sistema se probó utilizando varias computadoras que funcionaron como *Invitado.* Las características de uno de los *Invitados* es una PC Pentium III a 550 MHz, que tiene como Sistema Operativo Windows98 y cuenta con 128 MB en RAM.

Para usar la palanca de mando (*joystick*), Java no cuenta con librerías que manejen ese puerto por lo que se utilizó un método nativo de Java, descrito en el Apéndice B, que permiten ejecutar programas escritos en otros lenguajes. Como las únicas computadora que cuentan con el puerto de juegos son las PC, y por lo general tienen Windows como sistema operativo, el archivo que se necesita crear para esta aplicación es una librería de enlace dinámico (DLL) descrita en el Apéndice B. En la figura 4.4 se muestran los valores que toma la palanca de mando dependiendo en que posición que se encuentre.

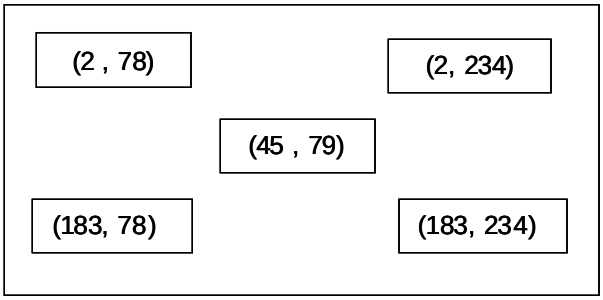


Figura 4.4: Posiciones que puede tomar la palanca de mando utilizada

4.2 Interfaz de operación

En la figura 4.5 se muestra la pantalla con la que cuenta el *Invitado* para teleoperar al *Vehículo*. Esta pantalla consta de dos ventanas: la primer ventana muestra las imágenes en tiempo real, de la cámara que se encuentra sobre el vehículo; la segunda ventana, es la ventana de control. Dicha ventana de control está compuesta de tres partes principales: la barra de opciones, el panel de control y el panel de retroalimentación.

La barra de opciones se encuentra en la parte superior de la ventana de control, y se tienen las opciones de iniciar la transferencia de datos e imágenes, seleccionar el modo de operación o escoger utilizar la palanca de mando. Para iniciar la transferencia de datos se selecciona el botón "*Data*". Si el usuario desea usar la palanca de mando (*joystick*), aparece una ventana adicional donde se muestra la posición y el estado de los botones. En la parte central de la

ventana de control aparece el panel de control con las opciones que tiene el usuario para controlar el *Vehículo*. El panel de control es utilizado para enviar los comandos de velocidad, dirección y sentido al *Vehículo* por medio de botones o deslizadores (*sliders*). En la parte inferior de la ventana de control se recibe la retroalimentación de los datos enviados por el *Vehículo*.

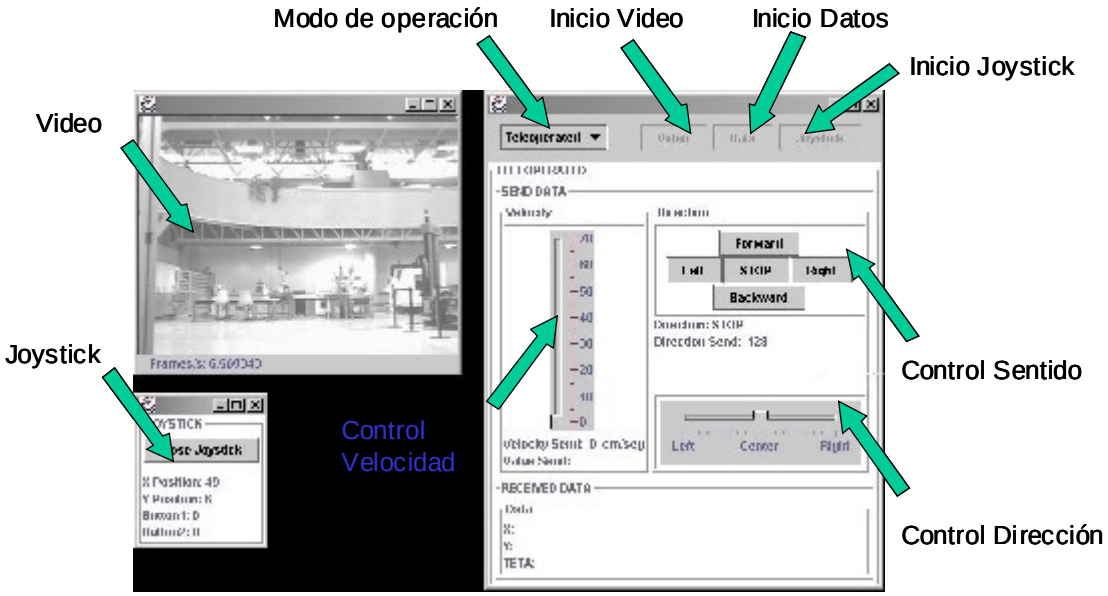


Figura 4.5: Interfaz utilizada por el *Invitado* para la teleoperación

En la figura 4.6 se muestra la interfaz que tiene el *Invitado* en la operación autónoma. La ventana de control está divida en tres partes: la barra de opciones, el panel de imágenes y el panel de retroalimentación.

En la parte superior de la ventana de control, aparece la barra de opciones donde el usuario puede cambiar el modo de operación. En la parte central aparece el panel de imágenes donde se muestra la imagen de la marca; en la imagen de la marca, se traza una cruz para facilitar que el usuario detecte si el *Vehículo* reconoció la marca. En la parte inferior aparece el panel de retroalimentación que muestra los datos enviados al *Vehículo*.

Cabe recordar que en la operación autónoma el usuario solamente supervisa la tarea; por lo cual sólo cuenta con las opciones de iniciar la transmisión de imágenes para ver la marca utilizando el botón "*Start*", iniciar el control autónomo con el botón "*Run*" o detener la operación autónoma por medio de "*Stop*".

En la figura 4.7 se muestra la ventana que aparece en el *Controlador Anfitrión* durante la operación autónoma, la cual está dividida en dos partes: el panel de imágenes y el panel de retroalimentación. En este trabajo se utiliza una marca móvil; por lo tanto, se necesita tener la imagen en el *Controlador Anfitrión* para asegurarse que la cámara ve completa la marca.

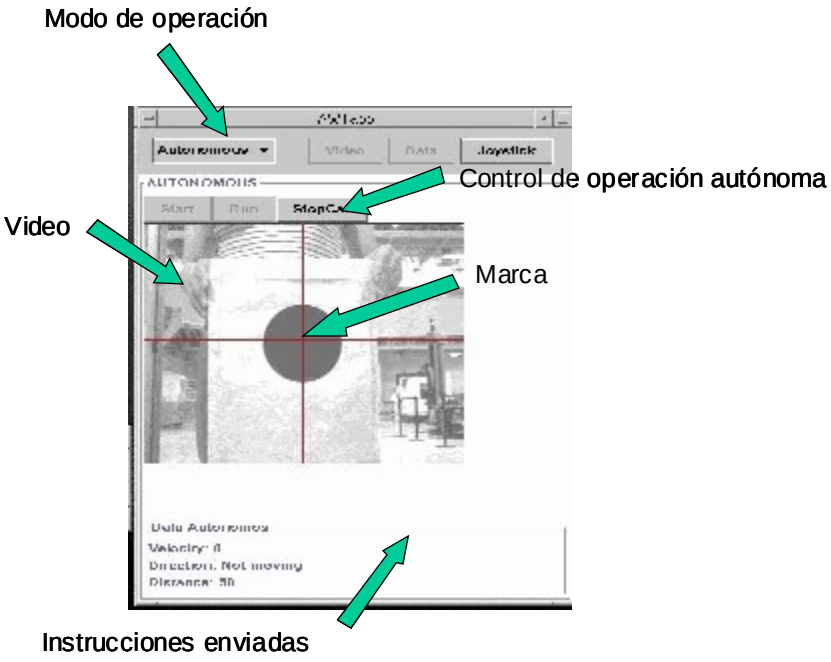


Figura 4.6: Interfaz utilizada por el *Invitado* para la operación autónoma

Figura 4.7: Interfaz utilizada en el *Controlador Anfitrión* durante la operación autónoma para asegurarse que la cámara vea la imagen completa

4.3 Experimentos realizados

En esta sección se mencionan los controladores que se desarrollaron para la operación autónoma y se presentan los resultados obtenidos durante la experimentación. Una forma de probar la eficiencia del sistema desarrollado es midiendo la cantidad de imágenes que recibe el cliente al utilizar el sistema. Otros experimentos realizados, fueron para probar la confiabilidad del controlador y de la transmisión de datos.

4.3.1 Control del *Vehículo* durante la operación autónoma

Se desea controlar a la salida la velocidad y sentido del *Vehículo*. Aunque el *Vehículo* puede alcanzar velocidades de 6 metros por segundo, se decidió que en modo autónomo sólo tome valores en el rango de 0 a 20 centímetros por segundo, debido al tiempo que tarda el procesamiento de la imagen y a que el retraso en Internet es variable.

El controlador tiene como parámetros de entrada la distancia actual entre la cámara y la marca *(d)*, que se define entre los 20 y los 100 centímetros, y la diferencia entre distancia actual y distancia anterior *(Δd*).

Primeramente se implementó un controlador utilizando lógica difusa, descrito en el Apéndice C. Para la base de conocimientos se convirtieron los valores reales obtenidos de la planta (*Vehículo*) a valores difusos con los que se establecieron dos conjuntos difusos de entrada (*d* y *Δd*) y uno de salida (*v*). Se utilizaron dos tablas de reglas obtenidas experimentalmente: una para controlar el avance y otra para el retroceso para diferenciar su comportamiento, pues cuando el *Vehículo* avanza, se acerca a la marca, y cuando retrocede, se aleja de ella. Después del análisis difuso, se aplicó el método de Mamdani o de las alturas para convertir los valores difusos a valores reales [Driankov 96].

Después se utilizó un controlador lineal descrito en el Capítulo 3, donde el *Vehículo* avanza si la distancia es mayor o igual al cierto valor, y retrocede si es menor que cierto valor; así mismo, se fijó un margen donde el *Vehículo* permanece detenido.

Al comparar los controladores, como se muestra en el Apéndice C, se encontró que al utilizar el controlador lineal, la cantidad de cuadros por segundo procesador era ligeramente mayor que utilizando el controlador difuso por lo que se escogió el lineal, ya que uno de los problemas que se tienen con el control, es el retraso en las imágenes.

4.3.2 Resultados experimentales

En todos los experimentos realizados se consideraron imágenes en niveles de gris de tamaño 320x240 píxeles.

En la tabla 4.1, se muestra que, para la captura de video en el *Controlador Anfitrión*, se obtuvieron 27 cuadros por segundo. Para la ejecución autónoma, de forma local, se obtuvieron 6.5 cuadros por segundo.

Tabla 4.1: Valores promedio de imágenes recibidas durante la operación Local

	Frecuencia (Cuadros / Segundo)	Tamaño de la imagen (Píxeles)
Captura de Video	27	320x240
Ejecución autónoma	6.5	320x240

En los experimentos realizados se consideraron tres casos: ejecución local, en la misma red y en otra red. Se considera la ejecución local cuando el *Controlador Anfitrión* y el *Invitado* se encuentran en la misma máquina llamada *Dali*. Si se encuentran dentro de la misma red, el *Invitado* es *Toledo*. Para el caso de los experimentos realizados en otra red, se utilizó la red de Informática donde el *Invitado* es *Acantu*.

En la tabla 4.2 se muestran los valores promedio obtenidos utilizando RMI, para la transmisión de video, sin controlar el *Vehículo*. Si el *Invitado* se encuentra en la misma red que el *Controlador Anfitrión*, se tienen en promedio 10.5 cuadros por segundo; si se encuentra en otra subred, se tienen en promedio 8.5 cuadros por segundo.

Tabla 4.2: Valores promedio de imágenes recibidas durante la transmisión de video usando RMI, antes de iniciar la operación del *Vehículo*

	Frecuencia (Cuadros / Segundo)	Tamaño de la imagen (Píxeles)
Local	22	320x240
Misma Red (LABRIM)	10.5	320x240
Red Informática	8.5	320x240

En la tabla 4.3 se muestran los valores promedio obtenidos utilizando RMI para la teleoperación. Si el *Invitado* se encuentra en la subred de Informática, se reciben en promedio 5.95 cuadros por segundo; si se encuentran en otra subred, se tienen en promedio 4.9 cuadros por segundo.

Tabla 4.3: Valores promedio de imágenes recibidas por el *Invitado* durante la teleoperación del *Vehículo*

	Frecuencia (Cuadros / Segundo)			
	Experimento 1	Experimento 2	Experimento 3	Experimento 4
Hora	8:00 AM	2:00 PM	6:00 PM	10:00 PM
Local	16	16.5	12	12
Misma Red	8.5	9	7	5.6
Informática	4.8	5.8	5	4

En la figura 4.8 se muestran gráficamente los resultados promedio de los valores obtenidos al ejecutar la teleoperación, en varios puntos dentro de la red del campus mostrados en la tabla 4.3.

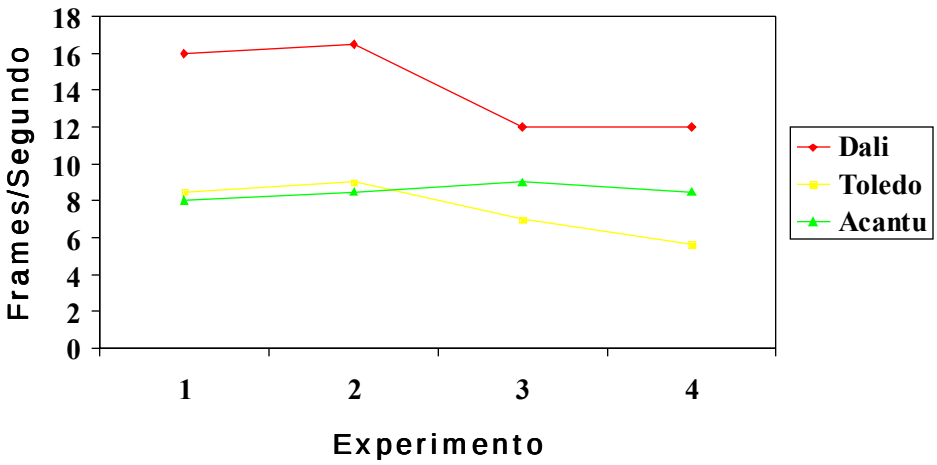


Figura 4.8: Resultados obtenidos en la teleoperación. Dentro de la red de Informática se obtuvieron en promedio 4.9 cuadros / segundo

En la tabla 4.4 se muestran los valores promedio obtenidos utilizando RMI durante la operación autónoma. Si el *Invitado* se encuentra en la misma red, se reciben en promedio 5.125 cuadros por segundo; si se encuentra en la red de Informática, se reciben en promedio 5.05 cuadros por segundo.

Tabla 4.4: Valores promedio de imágenes recibidas por el *Invitado* durante la operación autónoma

	Frecuencia (Cuadros / Segundo)			
	Experimento 1	Experimento 2	Experimento 3	Experimento 4
Hora	8:00 AM	2:00 PM	6:00 PM	10:00 PM
Local	1.5	1	1	1
Misma Red	5	5.3	4.6	5.6
Informática	4.5	5.7	5.2	4.8

En la figura 4.9 se muestran en forma gráfica, la cantidad de imágenes promedio al ejecutar la operación autónoma en varios puntos dentro de la red del campus mostrados en la tabla 4.4.

En la figura 4.10, la gráfica muestra el del tráfico de salida en la red del campus en MB por segundo, durante el día entero, de acuerdo a la hora del día.

Aunque no siempre hay el mismo trafico a la misma hora, la gráfica proporciona una idea general de las horas en las que hay mayor y menor trafico. De acuerdo a los valores obtenidos en la figura 4.10, se encuentra que el valor promedio en un día es 3285.7 Kbits por segundo, como se muestra en la tabla 4.5.

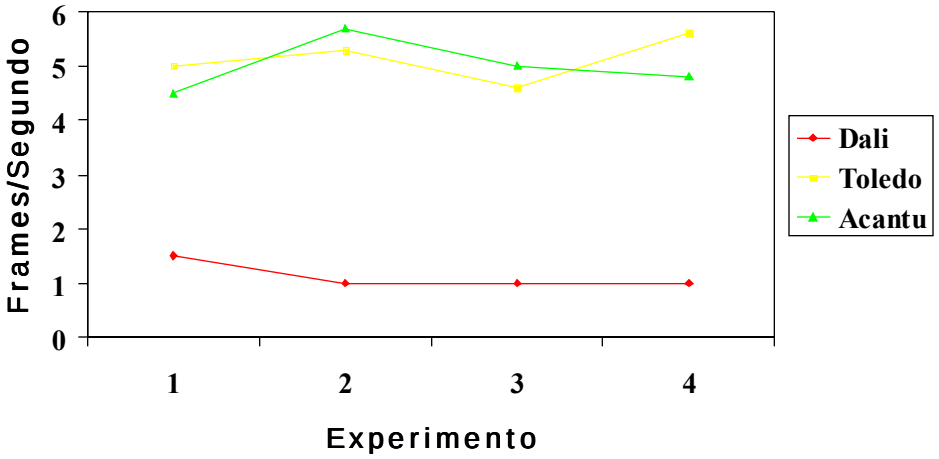


Figura 4.9: Resultados obtenidos en la ejecución autónoma. Dentro de la red de Informática se obtuvieron en promedio 5.05 cuadros / segundo

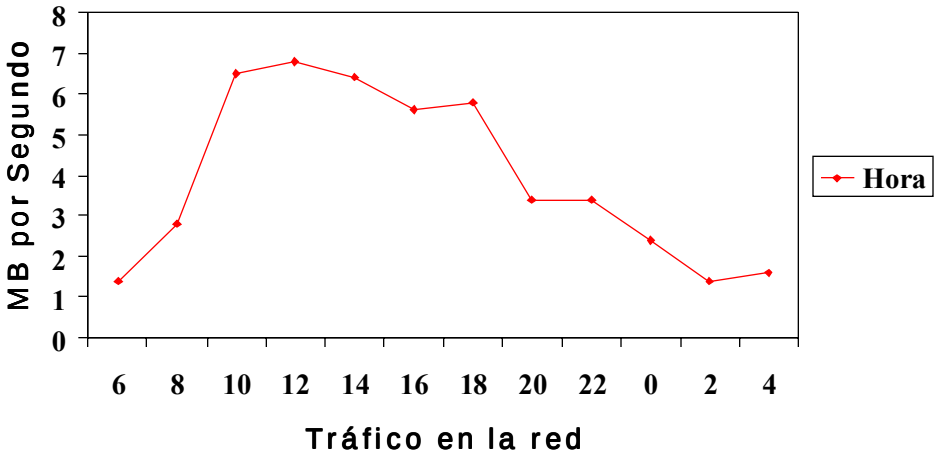


Figura 4.10: Análisis de tráfico para la red de Internet de acuerdo a la hora del día dentro del Campus

Tabla 4.5: Valores del trafico en la red del Campus.

	Kbits por segundo
Valor Máximo	7618.2
Valor Promedio	3285.7

Otras medidas de eficiencia del sistema son el control y la comunicación, sin embargo estos datos sólo se pueden obtener de forma aproximada.

La eficiencia en la comunicación se refiere a la confiabilidad de la transmisión de datos; sin embargo, resulta muy complicado medir el porcentaje de confiabilidad de los módems, puesto que influyen muchos factores, como la interferencia que puede existir en un momento determinado, debido a que se utiliza un canal comercial de FM para la transmisión. Sin embargo, utilizando el programa de pruebas descrito en el Apéndice A, en el 100% de los casos de la transmisión y recepción no se dividieron los paquetes enviados. La distancia máxima a la que se realizaron estos experimentos, fue a 40 metros a línea de vista.

Dentro de la operación del *Vehículo,* los módems tuvieron un desempeño menor; durante la transmisión, se recibieron los datos correctos aproximadamente 80% de las ocasiones; sin embargo, en la recepción sólo un 20% de las veces. Estos datos son aproximados, ya que se obtuvieron de acuerdo a las veces que el *Vehículo* obedeció las instrucciones enviadas durante uno de los experimentos.

Para medir la eficiencia del controlador, se realizó otro experimento, colocando la marca fija, para comprobar que el controlador se estabiliza en un valor determinado y no oscila indefinidamente.

Se considera que el controlador utilizado aproximadamente en el 90% de los casos permite situar al *Vehículo* a la distancia deseada, ya que el problema del retraso de la imagen, hace que el *Vehículo* oscile un poco, debido a que ve una imagen mas lejos de la distancia establecida y avanza. Sin embargo, cuando recibe la siguiente imagen, ya esta más cerca por lo que tiene que retroceder antes de detenerse. En la tabla 4.6 se muestran los resultados obtenidos experimentalmente, considerando que la marca se encuentra fija en el punto 1.

Tabla 4.6: Resultados obtenidos experimentalmente de la oscilación del controlador con la marca fija

Tiempo	Distancia
1	55
2	135
3	105
4	110

El *Vehículo* inicia a 55 cm de la marca y se desea encontrar un punto en donde se estabilice y se detenga. Como se muestra en la figura 4.11, en el *tiempo 1*, el vehículo se encuentra a 55 cm. de la marca; en el *tiempo 2*, el vehículo retrocede y se pasa de la meta deseada, ya que llega a 135 cm de la marca; en el *tiempo 3* el vehículo avanza y se pasa de la meta, ya que llega a 105 cm de la marca; finalmente, en el *tiempo 4*, el vehículo retrocede y se estabiliza a 110 cm de la marca.

De acuerdo a los resultados obtenidos en la figura 4.11, durante la operación autónoma, el vehículo se estabiliza a la distancia establecida, después de una ligera oscilación.

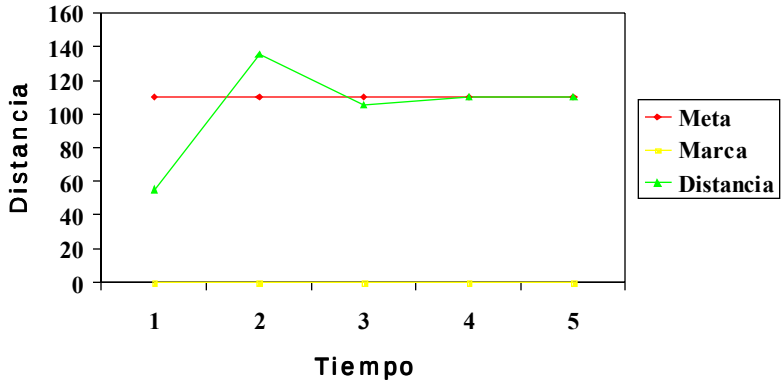


Figura 4.11: Resultados de la oscilación del controlador colocando la marca fija

Por otra parte, si se considera el caso donde el *Vehículo* inicia a 150 cm de la marca y se desea encontrar un punto en donde se estabilice y se detenga. En la tabla 4.7 se muestran los resultados obtenidos experimentalmente, considerando que la marca se encuentra fija en el punto 1.

Tabla 4.7: Resultados obtenidos en el segundo experimento del controlador con la marca fija

Tiempo	Distancia
1	150
2	90
3	130
4	95
5	110

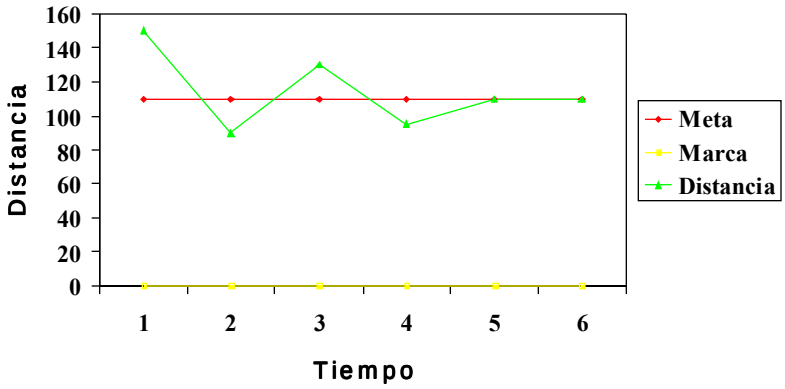


Figura 4.12: Resultados de la oscilación del controlador colocando la marca fija en el segundo experimento realizado

Como se muestra en la figura 4.12, en el *tiempo 1*, el vehículo se encuentra a 150 cm. de la marca; en el *tiempo 2*, el vehículo avanza y se pasa de la meta deseada, ya que llega a 90 cm de la marca; en el *tiempo 3* el vehículo retrocede y se pasa de la meta, ya que llega a 130 cm de la marca; en el *tiempo 4*, el vehículo avanza y se pasa de la meta llegando a 95 cm; finalmente, en el *tiempo5,* el *Vehículo* se estabiliza a 115 cm de la marca.

Al igual que en el caso anterior mostrado en la figura 4.11, durante la operación autónoma, el vehículo se estabiliza a la distancia establecida, después de una ligera oscilación como se muestra en la figura 4.12.

Capítulo 5

Conclusiones

La principal aportación de esta tesis es el desarrollo de un modelo para el control a distancia de un *Vehículo* con dos formas de operación: teleoperado o autónomo, las cuales pueden ser conmutadas durante la ejecución, de acuerdo a la actividad que el *Invitado* pretenda realizar.

El modelo desarrollado cuenta con tres módulos: el *Invitado*, el *Anfitrión* y el *Medio* de comunicación entre ambos; a su vez, el *Anfitrión* está compuesto por el *Controlador Anfitrión*, el *Vehículo* y la comunicación entre el *Vehículo* y el *Controlador Anfitrión*.

El *Vehículo* utilizado para la implementación, es un *Vehículo* a control remoto comercial en el cuál se sustituyeron las tarjetas originales, por tarjetas diseñadas para controlar su velocidad y dirección. Para la transmisión de los datos obtenidos por los sensores del *Vehículo*, se diseñó un módem transmisor "*half-duplex*" a 1200 bps.

El modelo de comunicación entre el *Vehículo* y el *Invitado* se desarrolló en Java utilizando RMI para el manejo de los datos y las imágenes. El uso de RMI para la comunicación entre el *Controlador Anfitrión* y el *Invitado*, permite que el manejo de objetos entre ambos lados de la conexión sea transparente; además, al ser implementado en Java, el sistema es multiplataforma.

Con este trabajo se consiguió tener funcionando un sistema básico, en el que se manejan todos los aspectos necesarios para que cualquier clase de vehículo pueda ser controlado a distancia; por lo tanto se espera que este trabajo sirva de base para futuros proyectos de investigación.

De acuerdo a los experimentos realizados, se demostró que la cantidad de imágenes y datos transmitidos, resultan suficientes para poder controlar el *Vehículo* a distancia, en ambos modos de operación.

Algunas de las limitantes que se encontraron durante la experimentación, son que el *Vehículo* requiere de una velocidad alta para empezar a moverse y que la carga de las baterías

influye ligeramente en la velocidad del *Vehículo*. Aunque los resultados obtenidos son satisfactorios, se propone el cambio del controlador actual por un controlador más eficiente que contemple estas situaciones.

En este trabajo se limita la velocidad máxima que puede alcanzar el *Vehículo*, ya que entre mayor es la velocidad, es más difícil controlarlo y se requiere mayor velocidad en la retroalimentación.

Para aumentar la retroalimentación de datos recibida por el *Invitado* durante la teleoperación, se propone la utilización de transmisión "*full-duplex*", ya que al tener transmisión "*half-duplex*", la prioridad es el envío de instrucciones.

Para la transmisión de imágenes de la cámara que se encuentra sobre el *Vehículo*, se utilizó un cable coaxial, pero se espera que a corto plazo se utilice un transmisor inalámbrico, porque al utilizar un cable, se limita la libertad de movimiento del *Vehículo*.

Apéndice A

Diseño de módems para la transmisión de datos por RF

Las computadoras procesan y almacenan datos en formato digital. La transmisión se realiza en formato digital o analógico. Una señal digital es aquella que transmite bits como pulsos altos y bajos (1 o 0); una señal analógica, transmite datos por medio de un patrón de onda que varía continuamente. Cuando se transmite e con dispositivos digitales, la computadora envía los datos directamente sobre el canal sin que la señal sea procesada.

Para transmitir utilizando medios analógicos, la señal digital es convertida a analógica por un proceso llamado modulación. De forma inversa, cuando es recibida, la señal analógica debe ser regresada a su formato digital por medio de la demodulación. Ambos procesos, la modulación y la demodulación, son ejecutados por el módem. La transmisión analógica as más económica y fácil de obtener que la digital.

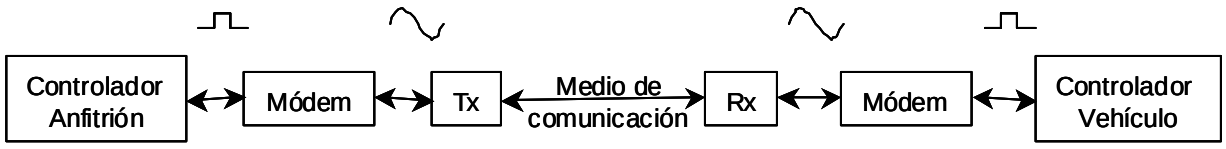


Figura A.1: Esquema básico de comunicación

Como se muestra en la figura A.1, el *Controlador Anfitrión* envía una señal digital al módem para que la convierta en analógica y la transmita a través del medio de comunicación. En este caso, se utiliza Radio Frecuencia (RF). En el lado del *Vehículo* se recibe la señal analógica y al pasar por el módem se convierte a digital para que la utilice el C*ontrolador del Vehículo.*

A.1 Descripción del módem

El módem esta basado en el circuito integrado 73M223 de TDK, el cual provee el filtrado, modulación y demodulación necesarios para implementar un canal de comunicación de datos serial asíncrono usando FSK (*Frequency Shift Keying*). En la transmisión asíncrona, los datos son enviados en paquetes de caracteres; un caracter a la vez. Cada grupo de caracteres incluye un "*bit*" de paridad, rodeado por un "*bit*" de inicio y un "*bit*" de final. El "*bit*" de paridad es utilizado para chequeo de errores.

Debido a que este circuito opera con el estándar TTL/CMOS, se requiere de una interfaz para adaptar las señales al estándar RS-232 del puerto serial de una PC, esta conversión la realiza el circuito integrado MAX232.

La velocidad de transmisión con la que los "*bits*" son enviados por el canal en el 73M223 es de 1200 "*bits*" por segundo. Debido a que tiene tecnología CMOS, el circuito se alimenta con 5V y tiene un bajo consumo de potencia (10mW) [Tdk 98].

El canal de comunicaciones es el camino físico de los datos entre el transmisor y el receptor y en este caso utilizó comunicación inalámbrica en la que el canal de comunicación es el medio ambiente, a éste tipo de transmisión se le llama por radio frecuencia (RF).

Se utilizaron como transmisores RF, dos radios "*walkie-talkie*" RadioShack de un solo canal que operan en la banda FM a 49.830MHz, aunque tienen un rango de hasta 400 metros al aire libre, la mayor distancia a la que se probaron fueron 40 metros dentro del laboratorio. Los radios requieren una alimentación de 9V de C. D. [RadioShack 99] que se toman del regulador de 9V (78L09) de la tarjeta del módem. En la figura A.2 se muestra el módem construido.

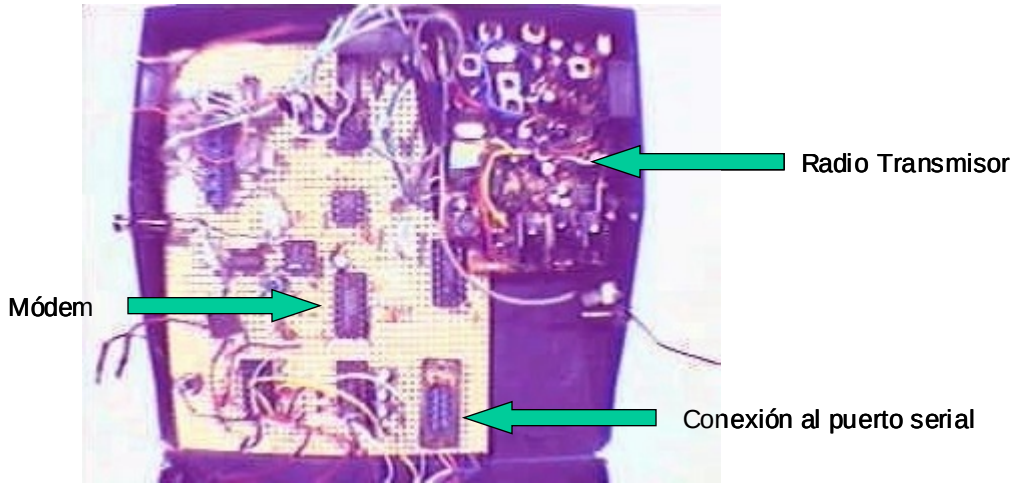


Figura A.2: Imagen del módem construido, el cual está compuesto por el módem y por el radio transmisor

En la tabla 1 se muestra la lista de componentes utilizados.

Tabla A.1: Lista de componentes utilizados

Componente		**Cantidad**	**Descripción**
73M223		1	Módem
MAX232		1	Convertidor de RS-232 a niveles TTL
CD4049		1	Buffer inversor hexadecimal
LMC7660		1	Convertidor de voltaje
74L32		1	Compuerta OR
LM567		1	Decodificador de tonos
LM358		2	Amplificador operacional
Cristal de 3.1872 M Hz.		1	Cristal
Transistor 2N2222		1	Transistor
Diodo 1N4001		1	Diodo
Relevador de 12 V		1	Relevador
Transformador de acoplamiento		2	Transformador
Regulador 78L05		1	Regulador de 12 a 5V
Regulador 78L09		1	Regulador de 12 a 9V
Header de 7 entradas		1	
Conector DB9 macho para circuito impreso		1	
Led		4	
Led bicolor		1	
Potenciómetro			
	10 kΩ	2	
	2 kΩ	1	
Resistencia			
	1 MΩ	3	
	10 kΩ	1	
	2.2 kΩ	1	
	1 kΩ	2	
	540 Ω	2	
	220 Ω	1	
	82 Ω	1	
	15 Ω	1	
Capacitor Electrolítico			
	220 µF 25V	1	
	10.32 µF	1	
	10 µF 35V	1	
	10 µF 16V	5	
	1 µF 50V	1	
Capacitor Cerámico			
	33 µF	2	
	0.1 µF	6	

En la figura A.3 se muestra el diagrama del circuito del módem.

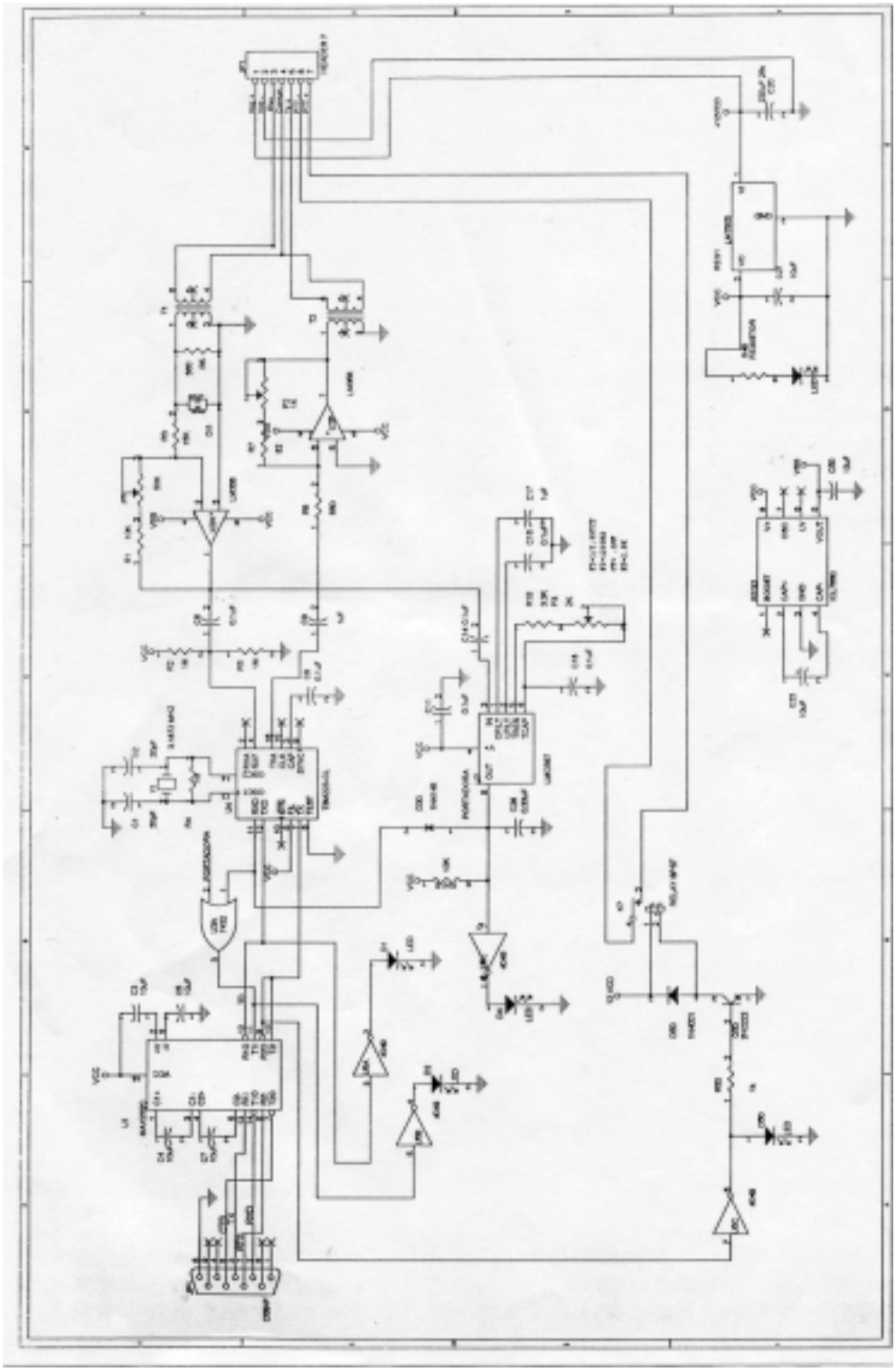


Figura A.3: Diagrama del módem

A.2 Funcionamiento del módem

El módem usa las señales básicas de transmisión: transmisión, recepción, RTS (*Request to send*), CTS (*Clear to send*) y la referencia a tierra. La dirección en la que los datos pueden fluir en un canal puede variar: en la transmisión "*half-duplex*" los datos fluyen en ambas direcciones, pero no al mismo tiempo, mientras que en la transmisión "*full-duplex*" los datos fluyen en ambas direcciones simultáneamente. En el circuito construido la transmisión es "*half-duplex*" debido a que el radio es de un solo canal, aunque es posible utilizar el módem en "*full-duplex*" al usar circuitos separados de transmisión y recepción.

La PC activa la señal de RTS y espera la señal de CTS o un cierto tiempo para empezar a transmitir. La señal de RTS se utiliza para encender el transmisor del radio. Es necesario que exista un retraso entre esta señal y el inicio de la transmisión de los datos, debido a que el transmisor tarda en estabilizar la frecuencia de la señal portadora. Este retraso se puede controlar por "*software*" o usando la señal de CTS. De la misma forma, al fin de la transmisión del paquete de datos debe haber un retraso para desactivar la señal de RTS, sin causar perdida de datos. En este caso los retrasos se controlaron por "*software*".

La señal de RTS es recibida en el MAX232 [TI 99] y pasa a un inversor que convierte esta señal a un uno lógico; esta señal polariza un transistor que activa un relevador, de esta forma, el transmisor del radio se enciende. Los datos son recibidos por MAX232 que convierte los niveles RS-232 a niveles lógicos TTL; de ahí pasan al circuito 73M223 que recibe los datos digitales, los modula y a la salida entrega los datos de forma analógica que son transmitidos por el radio. Las señales son recibidos por el radio receptor que se conecta a otro módem de las mismas características donde la señal análoga es convertida a digital y transmitida hacia la computadora receptora.

A.2.1 Forma de Operación

El módem requiere una alimentación de 12V de C.D. Para el envío y recepción de datos, el módem se conecta al puerto serial de la PC con un cable que tiene la configuración mostrada en la tabla A.2.

Tabla A.2: Pines del puerto serial (Conector DB-9)

No. de Pin en la PC	No. de Pin en el módem	Nombre	Función
3	2	Transmit Data (TD)	Salida de datos serial
2	3	Receive Data (RD)	Entrada de datos serial
7	7	Request To Send (RTS)	Informa al módem que la UART esta lista para intercambiar datos
8	8	Clear To Send (CTS)	Indica que el módem esta listo para intercambiar datos
5	5	Signal Ground (SG)	Tierra

Si se considera un puerto serial de 9 "*pines*", la línea de transmisión es el "*pin*" 2, la línea de recepción es el "*pin*" 3 y la tierra es el "*pin*" 5. Además de lo anterior, y debido al tipo de

transmisión, se requiere conectar el "*pin*" 7 de "*Request to Send*" (RTS) que sirve para notificar al módem, que está listo para intercambiar datos. Se necesita activar el RTS cada vez que se necesite mandar un dato y deshabilitarlo inmediatamente después para que este listo para recibir.

A.2.2 Programa de pruebas para revisar los módems

Para probar el funcionamiento de los módem, se desarrolló un protocolo muy simple donde se envían por RF los caracteres tecleados en una computadora a la otra. Cuando ocurre alguna falla en el control del *Vehículo*, este programa permite revisar que los módems estén funcionando correctamente. Para filtrar el ruido se mandan dos caracteres de inicio "(", el dato que se desea enviar y un caracter de final ")", como se muestra en la figura A.4.

(	(	Dato	)

Figura A.4: Formato del paquete de datos en el programa de pruebas

Tabla A.3: IRQs y direcciones del puerto serial

Nombre	Dirección	IRQ	Dirección de Inicio
COM1	3F8	4	0000:0400
COM2	2F8	3	0000:0402

Tabla A.4: Tabla de registros del puerto serial utilizados

Dirección / Valor	Nombre del registro
+0 / 0x60	Divisor de cierre del byte bajo (DLAB) 1200 bps
+1 / 0x00	Divisor de cierre del byte alto (DLAB) 1200 bps
+2 / 0xC7	Registro de identificación de interrupción (IIR) 1 1 0 0 0 1 1 1 No interrupciones pendientes FIFO habilitado Status de interrupción de línea
+3 / 0x80 +3 / 0x03	Registro de control de línea 1 1 0 0 0 1 1 1 Sin paridad 8 bits de datos Bit de acceso del divisor de cierre Un bit de stop
+4 / 0x02	Registro de control del módem (MCR) 0 0 0 0 0 0 1 0 Forzar el RTS
+5	Registro de estado de línea x x x x x x x x Datos listos

Al inicio del programa se declaran la dirección del puerto que se va a usar dentro de las opciones mostradas en la tabla A.3 y se inicializan los registros del puerto serial. Estos datos se

encuentran en la tabla A.4 [Cpeacock 98]. El DLAB es 1200, el FIFO esta habilitado, se desea utilizar 8 bits de datos, 1 "*bit*" de paro, sin paridad.

A continuación se muestra el código del programa utilizado para revisar el funcionamiento de los módems y se corrija cuando ocurra algún error.

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>

#define PORT1 0x3F8             /*COM1*/

void main(void)
{
   int c, ch, aux1, aux2=0, inicio, final;
   /*------ Inicializar registros -----*/
   outportb(PORT1 + 1, 0);
   outportb(PORT1 + 3, 0x80); /* 8 bits de datos, 1 de paro, sin paridad, DLAB */
   outportb(PORT1 + 0, 0x60); /* 1200 bps */
   outportb(PORT1 + 1, 0x00); /* 1200 bps */
   outportb(PORT1 + 3, 0x03);
   outportb(PORT1 + 2, 0xC7); /* FIFO habilitado, UART 16550, */
   clrscr();
   printf("\n Presiona ESC para salir \n");
   do {
       /*--- Rx ---*/
       c = inportb(PORT1 + 5);         /* checa si un caracter ha sido recibido */
       if (c & 1 ) {
              ch = inportb(PORT1);     /* si es recibido, toma el carácter*/
              aux1=aux2;
              aux2=ch;
              if (ch==41) {
                  printf ("\n%c\n",aux1);     /* imprimo solo el caracter */
              }
       }
       /*--- Tx ---*/
       if (kbhit()) {                  /* si una tecla es presionada */
              ch = getch();            /* toma el caracter  */
              inicio = 40;             /* caracter inicial = "(" */
              final = 41;              /* caracter final = ")" */
              outportb(PORT1 + 4, 0x02);      /* forzar el RTS */
              delay (5000);
              printf("%c", ch);
              outportb(PORT1, inicio);  /* manda caracter inicial al puerto serial
*/
              delay(30);               /* el retardo depende de la computadora */
              outportb(PORT1, inicio);  /* manda caracter inicial al puerto serial
*/
              delay(30);
              outportb(PORT1, ch);     /* manda el caracter al puerto serial */
              delay(30);
              outportb(PORT1, final);   /* manda el caracter final al puerto
serial*/           delay(1000);
              outportb(PORT1 + 4, 0x00);      /* deshabilita el RTS */
       }
   } while (ch !=27);                  /* mientras sea diferente de "Esc" */

   }
```

La parte principal del programa es un ciclo de recepción y transmisión que se interrumpe al presionar la tecla "*Esc*". En la recepción se verifica si un carácter ha sido recibido; si no, se

supone que es ruido y se imprime. En la transmisión se verifica si una tecla es presionada; de ser así, se fuerza el RTS a 1, se pone un retardo y se mandan el paquete de datos con un retardo entre cada uno de los caracteres; por último, se pone un retardo antes de deshabilitar el RTS. Estos retardos dependen de la computadora utilizada.

Apéndice B

Sistema realizado para el control de un vehículo a distancia

En la primera sección se muestran los diagramas del sistema desarrollo para la conexión entre el *Anfitrión* y el *Invitado,* a través de RMI. En las siguientes secciones se describe el funcionamiento de RMI y de los métodos nativos, utilizados para la captura de la imagen y para la palanca de mando.

B.1 Diagramas de las clases del sistema

En los siguientes diagramas se muestran las relaciones entre las clases del sistema desarrollado.

En la figura B.1 se ilustra la clase "RemoteSunVideo" que implementa la interfaz "VideoServer", es el *Anfitrión* que espera la conexión del *Invitado*; esta clase se ejecuta en el *Controlador Anfitrión*.

La clase "Sabueso" que implementa la interfaz "Auto", es la clase principal del comportamiento autónomo.

La clase "SunVideo" que implementa la interfaz "Video", es la clase encargada de la captura de las imágenes recibidas en el *Controlador Anfitrión*.

La clase "SerialData" que implementa la interfaz "Data", es la clase principal, encargada de la transmisión y recepción de datos en el puerto serial.

Por su parte, la clase "StartUp" que implementa la interfaz "VideoClient" y es llamada por la clase "ControlFrame" es el *Invitado* que se conecta al *Anfitrión* y es mostrada en la figura B.2. La clase "PortStatus" maneja el estado del puerto serial mientras que la clase "SabuesoStatus" maneja el estado de la operación autónoma. La clase "Instruction" es la

clase encargada de establecer los modos de operación y maneja el estado de los datos recibidas y de las instrucciones enviadas.

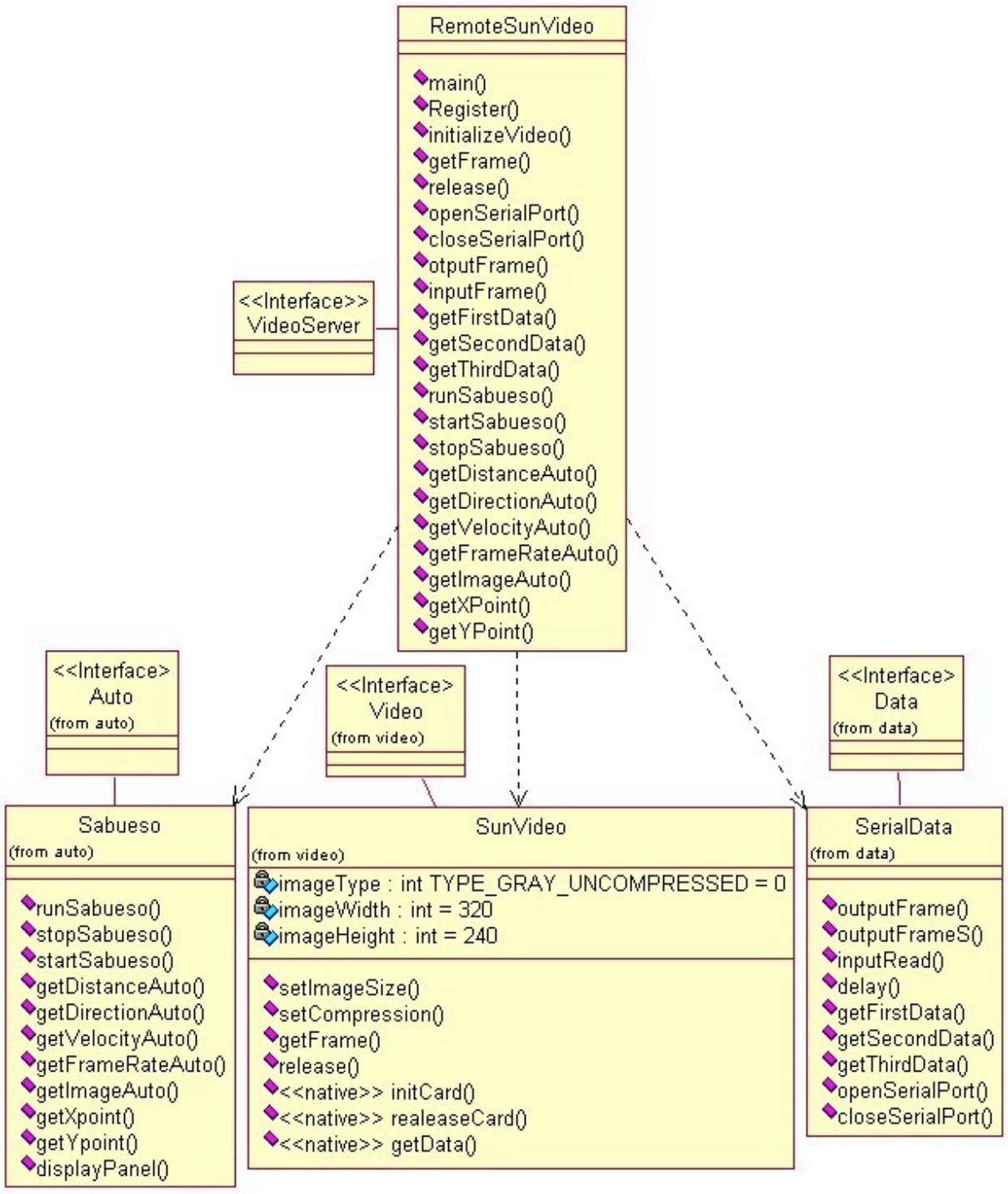


Figura B.1: Diagrama que muestra las clases y los métodos que establecen la conexión entre el *Invitado* y el *Anfitrión*

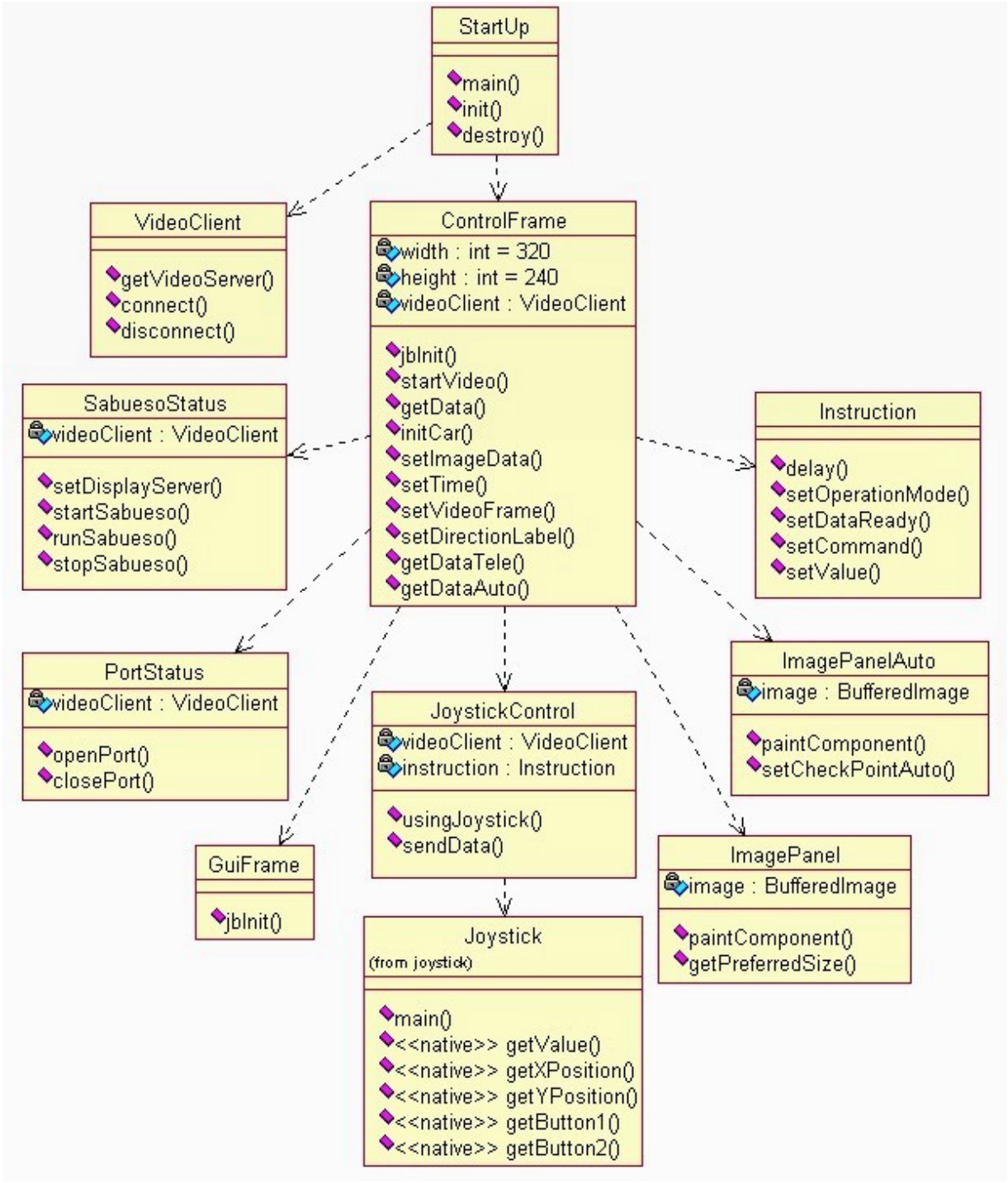


Figura B.2: Diagrama que muestra la clase "StartUp" la cual inicia la conexión entre el *Invitado* y el *Anfitrión*

En la clase "GuiFrame" se maneja la presentación del sistema al usuario. Para mostrar las imágenes al usuario, se utilizan la clase "ImagePanel" que muestra la imagen capturada y la clase "ImagePanelAuto" que muestra la imagen después de haber sido procesada en el modo

autónomo. La clases "Joystick" y "JoystickControl" se utilizan cuando el usuario quiere controlar al vehículo utilizando una palanca de control.

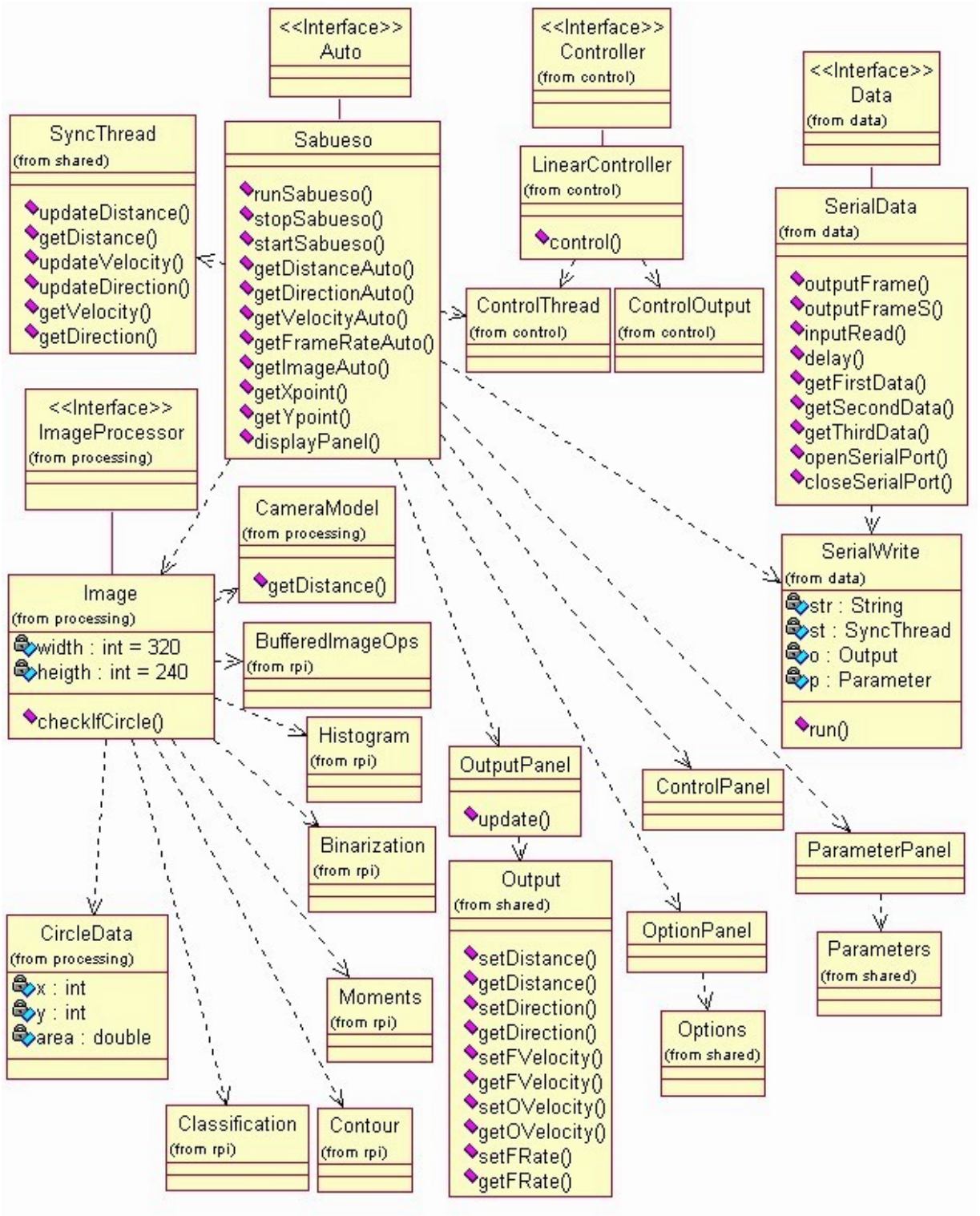


Figura B.3: Diagrama que muestra las clases utilizadas en el comportamiento autónomo

En la figura B.3 se muestra el diagrama del sistema de comportamiento autónomo. La clase "Sabueso" que implementa la interfaz "Auto" es la clase principal.

Para el control del *Vehículo* se utilizan las clases dentro del paquete "control". La clase donde se encuentra el controlador utilizado es la clase "LinearController" que implementa la interfaz "Controller". Para el manejo de los datos se tiene al paquete "data"cuya clase principal "SerialData" implementa la interfaz "Data".

Para el procesamiento de las imágenes se tiene la clase "Image" que implementa la interfaz "ImageProcessor". Las clases encargadas de hacer el reconocimiento de la imagen se encuentran en el paquete "rpi".

B.2 RMI

Para utilizar RMI primero se define una interfaz que crea una clase, a la que se pueda tener acceso remotamente. Los parámetros y valores regresados pueden ser de cualquier tipo. Los flujos de objetos manejan la transferencia de datos automáticamente. La clase debe implementar la interfaz. Es generado una plantilla (*stub*) y un esqueleto (*skeleton*) usando el compilador *rmic* que es un método RMI. La plantilla es una clase que automáticamente traslada las llamadas y los parámetros a métodos remotos en la red de comunicación. El esqueleto es la clase que reside en la misma máquina virtual como el objeto remoto el cual acepta estas conexiones y las traslada a las llamadas al método actual en el objeto. En la figura B.4 se muestra el diagrama de la arquitectura RMI.

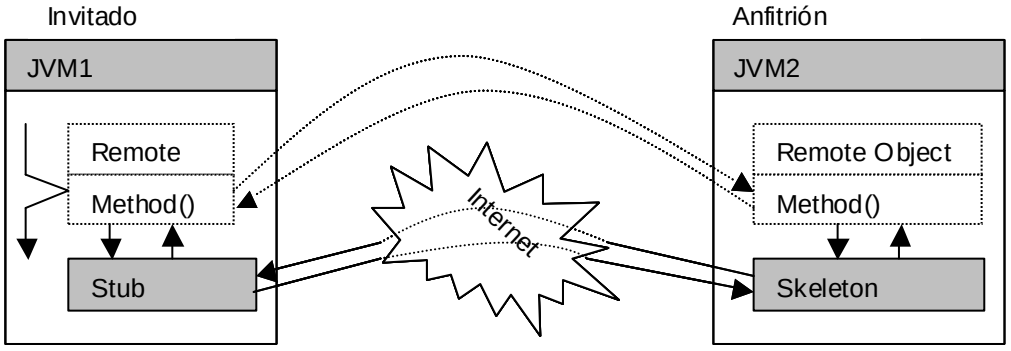


Figura B.4: Arquitectura RMI

El paso final consiste en que el objeto remoto debe ser registrado con un nombre de servicio que permita a los clientes direccionarse al objeto por el nombre de forma similar a los puertos de TCP. El objeto remoto está corriendo en un puerto remoto particular. Primero el cliente busca al objeto con el nombre del servicio. La solicitud regresa una referencia al objeto en cuestión que automáticamente informa al objeto que tiene un cliente remoto como se muestra en la figura B.5 [Virtej 98].

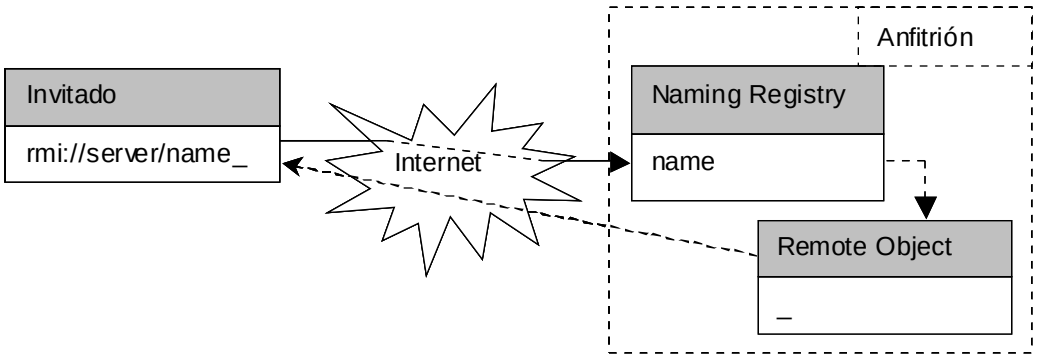


Figura B.5: Registro del *Anfitrión*

Para arrancar el *Controlador Anfitrión*, se crea una instancia de la clase que podrá ser utilizada remotamente y se registra en el puerto 1099. Una vez registrado, el servidor corre continuamente esperando la conexión de un *Invitado*.

B.3 Métodos nativos

Java provee la habilidad de acceso a código ejecutable escrito en otro lenguaje de programación como C, directamente a través de los métodos nativos. La Interface Java Nativa (JNI) permite al código Java que corre dentro de la Maquina Virtual de Java (VM) operar con aplicaciones y librerías escritas en otros lenguajes.

Para escribir métodos nativos se siguen varios procesos: primero se escribe el código Java; se crea una clase de Java que declara el método nativo y que incluye un método "*main*" que llama al método nativo y se compila normalmente utilizando "*javac*". Después se crea el archivo de cabecera .h para el método nativo usando "*javah*" con la interface nativa "*-jni*". Una vez creado este archivo se escribe la implementación del método nativo en el lenguaje de programación que se desea utilizar y se crear una librería compartida en Solaris (.so) o en Windows (dll), compilando el archivo de cabecera y los archivos de implementación. Por último, se corre el programa Java normalmente usando el interprete "java".

Las únicas computadora que cuentan con el puerto de juegos son las PC, y por lo general tienen Windows como sistema operativo; por lo cual el archivo que se necesita crear para utilizar la palanca de mando es una librería de enlace dinámico (DLL). Una librería de enlace dinámico (*Dynamic Link Library*) es una librería de funciones que puede ser llamada por una aplicación en el momento de su ejecución. Cuando el programa necesita una función que no es un archivo ejecutable, Windows carga la conexión a la libraría dinámica (el DLL), haciendo todas sus funciones disponibles para la aplicación. Un DLL de Windows requiere solo una función llamada *LibMain*; si no la tiene, no va a funcionar, ya que llama esta función para inicializarlo. También se necesita un archivo especial DEF para el DLL, que usualmente es generado por la mayoría de los compiladores como Visual C++.

Apéndice C

Controladores

En este apéndice se describe el procedimiento utilizado para encontrar el controlador del *Vehículo* en la forma autónoma. En la primer sección se describe el procedimiento para encontrar la función del controlador lineal. En la segunda sección se describe el experimento que se realizó utilizando lógica difusa. Finalmente se establece la comparación entre ambos controladores.

C.1 Controlador Lineal

Para obtener una ecuación para el controlador lineal que simule el comportamiento deseado primeramente se hizo una simulación utilizado el método de Montecarlo que utiliza números aleatorios donde se consideraron que los valores deseados de velocidad estaban en el rango de 30 a 80 y la distancia de 20 a 50 para la reversa, de 50 a 70 detenido y de 70 a 100 para el avance.

Como se desean obtener 10 muestras, los primeros dos dígitos de los números aleatorios obtenidos en la tabla de números aleatorios [Miller 84] fueron: 83, 09, 66, 88, 83, 67, 47, 84, 36, 67. Si se considera que se desea tener un rango de velocidad de 30 a 80, se dividen en 5 grupo, asignando las velocidades de acuerdo al dígito de inicio de los números aleatorios como se muestra en la tabla C.1.

Tabla C.1: Asignación de los rangos de velocidad a los números aleatorios

1er Digito	Velocidad
1, 2	70-79
3, 4	60-69
5, 6	50-59
7, 8	40-49
9, 0	30-39

Para obtener las distancias asociadas con las velocidades, se hizo lo mismo, sólo que en este caso se tomaron los dos primeros dígitos como se muestre en la tabla C.2.

Tabla C.2: Asignación de los rangos de distancia a los números aleatorios

1er Digito	2do Digito	Rango Reversa	Distancia Reversa	Rango Avance	Distancia Avance
1, 2	1	20-26	20.6	94-100	100
	2		21.2		99.4
	3		21.8		98.8
	4		22.4		98.2
	5		23		97.6
	6		23.6		97
	7		24.2		96.4
	8		24.8		95.8
	9		25.4		95.2
	0		26		94.6
3, 4	1	26-32	26.6	88-94	94
	2		27.2		93.4
	3		27.8		92.8
	4		28.4		92.2
	5		29		91.6
	6		29.6		91
	7		30.2		90.4
	8		30.8		89.8
	9		31.4		89.2
	0		32		88.6
5, 6	1	32-38	32.6	82-88	88
	2		33.2		87.4
	3		33.8		86.8
	4		34.4		86.2
	5		35		85.6
	6		35.6		85
	7		36.2		84.4
	8		36.8		83.8
	9		37.4		83.2
	0		38		82.6
7, 8	1	38-44	38.6	76-82	82
	2		39.2		81.4
	3		39.8		80.8
	4		40.4		80.2
	5		41		79.6
	6		41.6		79
	7		42.2		78.4

	8		42.8		77.8
	9		43.4		77.2
	0		44		76.6
9, 0	1	44-50	44.6	70-76	76
	2		45.2		75.4
	3		45.8		74.8
	4		46.4		74.2
	5		47		73.6
	6		47.6		73
	7		48.2		72.4
	8		48.8		71.8
	9		49.4		71.2
	0		50		70.6

Una vez que se obtuvieron estos valores, se calculan las velocidades y las distancias como se muestra en la tabla C.3.

Tabla C.3: Obtención de la velocidad y distancia en la simulación

Aleatorio	Distancia	Velocidad
83	39.8	43
9	49.4	39
66	35.6	56
88	42.8	48
83	39.8	43
67	36.2	57
47	30.2	64
84	40.4	44
36	29.6	63
67	36.2	57

Ajustando los valores obtenidos en la simulación a una recta por el método de mínimos cuadrados se obtiene la ecuación de la forma mostrada en la formula C.1 para el control del avance y otra para la reversa.

$$v = a + bd \tag{C.1}$$

donde v es la velocidad, d es la distancia y a y b son constantes.

Para la reversa se obtuvo que a = 104.6 y b = -1.4. y para el avance a = -73.8 y b = 1.5. Por lo tanto,

$$\left.\begin{array}{ll} v = 104 - 1.4d & 20 \le d \le 50 \\ v = 0 & 50 < d < 70 \\ v = -73 + 1.5d & 70 \le d \le 100 \end{array}\right\} \tag{C.2}$$

El controlador obtenido se encuentra en la figura C.1. Se puede observar que el rango de valores de velocidad es entre 30 y 80. La distancia esta comprendida entre 20 y 100. Sin embargo, de los 50 a los 70 cm se considera que el *Vehículo* está detenido ya que las velocidades que se obtienen de la ecuación utilizada no son suficientes para mover al *Vehículo*.

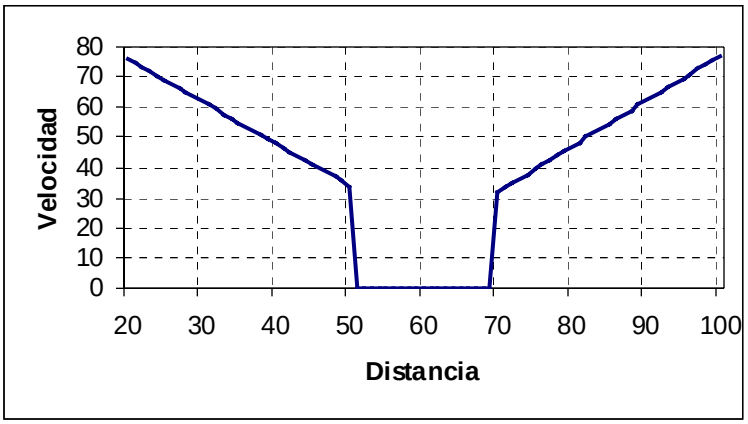


Figura C.1: Controlador lineal de velocidad obtenido en la simulación de acuerdo a los rangos deseados de velocidad y distancia.

Los resultados obtenidos mediante el procedimiento anterior, son los que aparecen en el Capítulo 3 y son usados durante la experimentación.

C.2 Controlador Difuso

Un controlador difuso debe tener los siguientes componentes:

1. Fuzzificación: convierte los valores nítidos obtenidos de la planta (*Vehículo*) a valores difusos.

2. Base de conocimiento y método de inferencia: donde se encuentran los conjuntos difusos y las reglas y se realiza el análisis difuso.

3. Defuzzificación: convierte la salida difusa a valores reales de la planta.

Para la implementar el controlador difuso la distancia se definió entre 0 y 200 centímetros. Se dividió de acuerdo a los rangos en tres conjuntos difusos: pequeño (S), mediano (M) y grande (L).

Δ_d es la diferencia entre la distancia actual y la distancia anterior. Se definió entre 100 y -100 centímetros y se dividió en cinco conjuntos difusos: negativo grande (NB), negativo (N), cero (Z), positivo (P) y positivo grande (PB). La velocidad se definió en el rango de 0 a 255 y tiene los siguientes conjuntos difusos: detenido (D), muy lento (VS), lento (S), mediano (M) y

rápido (F). Los conjuntos difusos correspondientes a la distancia, a la delta de distancia (Δ_d) y a la velocidad se presentan en la figura C.2.

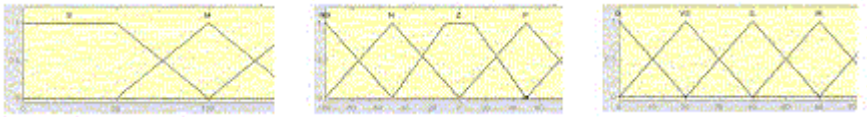


Figura C.2: Conjuntos difusos de distancia, Δ_d y velocidad

Se decidió utilizar dos tablas de reglas, una para controlar el avance y otra para retroceso debido a que el *Vehículo* se comporta de manera diferente en cada situación. Cuando el *Vehículo* va hacia adelante se acerca a la marca, pues trata de seguirse moviendo mientras la distancia sea grande y el Δ_d negativo hasta llegar a una distancia corta y un Δ_d cero. Cuando va hacia atrás se aleja de la marca, pues trata de seguirse moviendo mientras la distancia es corta y el Δ_d positivo, hasta llegar a una distancia corta un Δ_d cero. Las tablas utilizadas se muestran a continuación, donde las columnas están determinadas por los conjuntos difusos de la distancia y los renglones por los conjuntos difusos de la Δ_d.

Tabla C.4: Tabla de avance

Δ\ d	**S**	**M**	**L**
NB	D	S	M
N	D	VS	S
Z	D	S	M
P	VS	S	M
PB	S	M	F

Los parámetros que se tienen a la entrada del controlador difuso son distancia y la diferencia entre la distancia actual y la distancia anterior, y se desea controlar a la salida la velocidad.

Tabla C.5: Tabla de retroceso

Δ\ d	**S**	**M**	**L**
NB	S	D	D
N	VS	D	D
Z	D	D	D
P	D	D	D
PB	D	D	D

La manera de representar estas reglas fue definiendo un arreglo tridimensional, en donde el primer índice determina la tabla (0 ó 1 para avance ó retroceso respectivamente), el segundo determina el renglón que se activa en la tabla (0, 1, 2, 3, 4, por cada conjunto difuso de Δ_d), y el tercero la columna (0, 1, 2). Dentro del arreglo se guarda el conjunto difuso correspondiente para la velocidad cuando se cumplen las condiciones renglón-columna para cada tabla, representados como 0, 1, 2, 3, 4 (D, VS, S, M, F, respectivamente).

El método de defuzzificación que se utilizó, para obtener los valores nítidos de velocidad a partir de los valores obtenidos mediante el análisis difuso, fue el método de las alturas debido a que requiere poco tiempo computacional [Driankov 96], el cual se vuelve un factor crítico cuando tratamos imágenes y respuestas en tiempo real. Este método utiliza los valores en donde se presentan los picos de los conjuntos difusos de salida. En nuestro caso, los valores de la velocidad en donde se encuentran los picos de los conjuntos difusos son: 0 para D, 20 para VS, 40 para S, 60 para M y 80 para F. Con estos valores se utilizó la siguiente fórmula para calcular la altura,

$$H = \sum_{K} \frac{C^{(k)} f_k}{f_k} \qquad \text{(C.3)}$$

donde $c^{(k)}$ representa los valores donde se encuentran los picos para cada conjunto difuso y f_k representa los pesos para cada uno de dichos conjuntos para la variable de salida, a los cuales también se les llama "alturas", de ahí el nombre de éste método.

La obtención de estas alturas se hace mediante la aplicación del método sugerido por Mamdani, el cual consiste en tomar la mínima de las dos alturas correspondientes a renglón-columna para cada una de las reglas disparadas. Por ejemplo, si tenemos una combinación renglón-columna que signifique: *Si distancia = S y* Δ_d *= NS entonces muévete VS*, en donde las alturas para distancia y para Δ_d fueron calculadas en 0.3 y 0.6 respectivamente, entonces se toma la mínima de éstas dos, 0.3, para asignarla como altura para el conjunto difuso de salida.

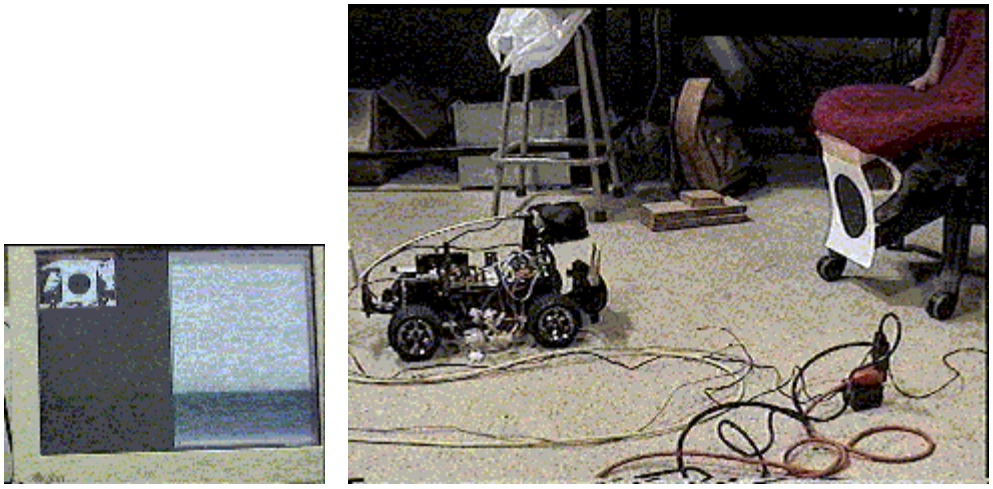


Figura C.2: Experimento realizado donde el *Vehículo* sigue a la marca utilizando un controlador con lógica difusa

En la figura C.2 se muestra el experimento realizado utilizando el controlador con lógica difusa. En la primer figura se puede observar la pantalla de la computadora donde se realizó el experimento; a la izquierda se observa la imagen obtenida por la cámara después de ser binarizada y a la derecha se ve una secuencia de datos desplegados para ir observando el comportamiento del programa en general. En la otra figura se muestra la marca utilizada y la distancia a la que se mantiene el *Vehículo*. Se realizaron varias pruebas con el objetivo de

obtener cual era el tamaño mas adecuado de la marca y a que distancia debería permanecer el carro de la marca. Se decidió que el tamaño mas adecuado para la marca era un círculo de 11.5 cm (4½ pulgadas) de diámetro y que el *Vehículo* debería permanecer a 50 cm de la marca.

C.3 Comparación entre el controlador lineal y el controlador difuso

La comparación del controlador lineal y el controlador difuso se realizó de acuerdo a la cantidad de cuadros por segundo obtenidos por el *Controlador Anfitrión* localmente, sin ser enviados al *Invitado.* Los resultados se muestran en la tabla C.6.

Tabla C.6: Valores promedio de cuadros por segundo obtenidos por el *Controlador Anfitrión* en los dos controladores utilizados.

	Cuadros / Segundo
Controlador Lineal	6.5
Controlador Difuso	5.5

De acuerdo a los resultados mostrados en la tabla C.6, el controlador lineal tiene mayor cantidad de cuadros por segundo; al no haber mucha diferencia entre el funcionamiento de ambos controladores, y al ser la implementación del controlador lineal más eficiente y flexible, se escogió el controlador lineal para ser usado por el *Vehículo* en el comportamiento autónomo.

Bibliografía

[Cimar 98] Crane C. "Autonomous Vehicle Development". The Center for Intelligent Machines and Robotics (CIMAR), Department of Mechanical Engineering, University of Florida. URL. **http://cimar.me.ufl.edu/~carl/af/af.html** Noviembre 1997.

[Cruz 01] Cruz A. "Implementación de una arquitectura de control jerárquica, genérica y modular para un vehículo autónomo". Tesis en progreso, 2001.

[Driankov 96] Dimiter Driankov, Hans Hellendoorn, Michael Reinfrank. "An introduction to fuzzy control". Springer-Verlag , 1996.

[Fernández 97] Fernández J. "JunioR" Departamento de Ingeniería de sistemas y Automática, Universidad de Málaga. URL. **http://www.isa.uma.es/personal/jafma/junior.htm** Noviembre 1997.

[Gauthier 90] Gauthier A., Duque J.C., Camargo J.A.. "Algunas realizaciones en el campo de la visión artificial". IV Congreso Latinoamericano de Control Automático. Puebla, México, 1990.

[González 96] González R., Woods Richard."Tratamiento digital de imágenes".1996.

[Gordillo 98] Gordillo J.L., López I., Olvera M., Cantú A., Berzunza J.L., Mora P.: "Reporte del estudio de factibilidad de un vehículo controlado a distancia que actualice la topografía de la mina". Centro de Inteligencia Artificial del ITESM. URL. **http://renoir.mty.itesm.mx/~gordillo/VA/** Noviembre 1998.

[Jokelainen 98] Jokelainen J. "Telerobotics using Internet". Information and Computer Systems in Automation, Helsinki University of Technology, URL **http://www.hut.fi/Yksikot/Auttieto/rap1/pre7/Jokelain.html** Julio 1998.

[Miller 84] Miller I. "Probabilidad y estadística para ingenieros". Página 377. Editorial Printice Hall, 1984.

[Otsu 79] N. Otsu. "A threshold selection method from gray-level histogram". *IEEE Trans. Syst., Man Cybernet.* 9(1):62-66, 1979.

[Palacios 00] Palacios G. “Control de dirección de un vehículo autónomo”. Mayo 2000.

[Sayers 98] Sayers C. “Remote control robotics”. New York. Editorial Springer, 1998.

[Tdk 98] TDK Semiconductor, “Data Sheet “, URL. **http://www.tsc.tdk.com/73m223_product_page.html** Abril 1998.

[Virtej 98] Virtej I. “Process monitoring and control-distributed technologies in automation”. Information and Computer Systems in Automation, Helsinki University of Technology, URL. **http://www.hut.fi/Yksikot/Auttieto/rap2/virtej/virtej.htm** Julio 1999.