UACM

Universidad Autónoma de la Ciudad de México

Nada humano me es ajeno

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

"Diseño e implementación de un sistema para la teleoperación de un robot móvil"

TRABAJO RECEPCIONAL

PARA OBTENER EL TÍTULO DE LICENCIADO EN

INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

PRESENTA:

Victor Manuel Rodríguez Martínez

Directora del trabajo recepcional

M. en I. Diana Aurora Cruz Hernández

México, D.F. noviembre, 2015.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Agradecimientos

A mi padre Victor Manuel Rodríguez, a pesar de nuestra distancia física, siento qué está conmigo y sé que este momento es tan especial para tí como lo es para mí.

A mi madre Rosa María Martínez, por su apoyo, comprensión y ayuda en los momentos difíciles.

A mis hermanos Cynthia y Alejandro, por su apoyo incondicional y haber hecho de este camino lo más fácil cuanto pudieron.

A mis amigos Roberto y Mayra por estar conmigo siempre y por el apoyo incondicional durante la carrera.

Al profesor Juan Carlos Aguilar por el apoyo en este proyecto y su paciencia.

A mí directora Diana Aurora Cruz, por haber creído en mí y haberme dado la oportunidad para realizar este proyecto.

A los profesores y sinodales, que me brindaron su apoyo y consejo a lo largo de mi carrera.

Agradezco las facilidades que tuve en el laboratorio B-114 linidet plantel San Lorenzo Tezonco para el desarrollo de este proyecto.

A la UACM por darme el apoyo económico para impresión y empastado de mi trabajo recepcional.

A la SECITI, por el apoyo para la realización de este proyecto, como parte del proyecto "Robot móvil de servicio para la vigilancia y prevención del delito (PI2011-IR).

¡Gracias!

Resumen

El presente documento describe la implementación de un sistema de teleoperación eficiente que permita el control de un robot móvil desde cualquier parte del mundo, es por ello que surge la necesidad de utilizar mejoras tecnológicas de software y hardware.

Para la elaboración del sistema y el cumplimiento de los objetivos planteados se usaron como guía dos arquitecturas relacionadas al control y comunicación de robots teleoperados que son la arquitectura AURA y la arquitectura cliente-servidor. Adicionalmente, para el desarrollo de la aplicación se utilizaron diversas tecnologías de software y hardware, como el lenguaje de programación HTML 5, CSS 3, el uso de Socket.io, un servidor web Node.js, así como el uso de la tarjeta de desarrollo Intel Galileo, el microcontrolador Mbed y el driver de potencia Pololu jrk.

Se pudo concluir que con el uso de las mejoras tecnológicas, el sistema tiene un buen desempeño y mejora los tiempos de respuesta en la manipulación del robot.

Palabras clave: Internet, Socket, Node.js, Servidor Web, Teleoperación, Robot móvil.

Índice

Capítulo 1. Introducción

Las nuevas tendencias en robótica han ayudado a desarrollar muchos sistemas que pretenden ampliar el uso de los robots teleoperados en la vida diaria. La mayoría de estas tendencias se han hecho posible gracias a la evolución de las tecnologías telemáticas y al diseño de arquitecturas para la automatización de éstos. Además se cuenta con una infraestructura global de comunicación que habilita la fácil implementación de sistemas teleoperados y de comunicación de baja latencia para estos sistemas. Aunque el internet representa un medio de comunicación barato y disponible, existen todavía muchos problemas por resolver antes de desarrollar aplicaciones verdaderamente fiables. Estos problemas incluyen el ancho de banda limitado, la pérdida de información y el retraso de la transmisión que varía arbitrariamente e influye en el rendimiento de los sistemas basados en la transmisión de paquetes.

También esfuerzos considerables de investigación en el campo de robótica móvil teleoperada están dirigiéndose hacia el uso de internet como medio de comunicación para facilitar la interacción remota.

La interacción remota es un tipo especial de la interacción hombre-robot, donde el hombre y el robot están separados por barreras físicas pero se comunican vía las tecnologías telemáticas. Se puede usar este tipo de interacción en muchas aplicaciones útiles como experimentación remota, teleoperación, telepercepción, teleprogramación, entre otras.

El objetivo de este capítulo es presentar el trabajo realizado en este proyecto y describir la estructura del presente documento.

1.1 Planteamiento del problema

La razón principal de éste proyecto es el poder conocer y aplicar nuevas mejoras tecnológicas para el control de un robot vía remota, para realizar actividades de vigilancia en un lugar específico y que puede ser manipulado desde cualquier terminal con acceso a internet de manera eficiente y rápida.

Este tipo de robots son típicamente controlados con tecnologías de hardware y software que tienen muchos defectos y limitaciones, como por ejemplo el procesamiento de información y precisión debido a la velocidad de los procesadores, el uso de bibliotecas con problemas de compatibilidad de hardware, además que no tienen integrado tecnologías de telecomunicaciones y de procesadores más potentes.

Con las nuevas mejoras tecnológicas, tanto de hardware como de software, las cuales resuelven los problemas de los sistemas anteriores, hoy estos sistemas han sido complementados con sistemas de comunicación basados en bluetooth, Wi-Fi y con micro procesadores más potentes.

Está situación lleva a usar dichas tecnológicas que permitan alcanzar mejores rendimientos y resultados, tanto en la transmisión de comandos como de video en tiempo real. Al mismo tiempo resulta importante explorar diversas arquitecturas tanto de software como de hardware, con el fin llegar a una implementación eficiente y robusta.

1.2 Justificación

Elegimos realizar este proyecto debido a que urge la necesidad de tener un sistema mínimo de teleoperación, que permita tener todos los elementos integrados en el robot móvil y usar software libre, a diferencia de otros que dependen de hardware externo y usan software que requiere licencia.

Actualmente los robots son una gran herramienta para sistemas de seguridad en la industria con sistemas inteligentes, monitoreados vía remota desde cualquier parte del mundo con ayuda de internet.

Estamos en una era en la que internet tiene el potencial de mejorar las cosas, de cómo relacionar o comunicar procesos o datos; convirtiendo la información en acciones que crean nuevas capacidades, experiencias más ricas y oportunidades más grandes, sin precedentes para las empresas, los individuos y los países. A esto se le denomina internet de las cosas (Internet of Things o IoT) (Adrian McEwen, 2014).

La tendencia de tener un sistema conectado a la red hoy en día facilita y acorta la comunicación sin tener que estar físicamente en un lugar o país para realizar ciertas operaciones, lo cual permite monitoreo y control desde cualquier lugar, llámese casa, oficina. Esto es importante porque así se simplifica la labor de vigilancia sin temor a que personas externas cuiden nuestros intereses, ya que confiar en ellas puede causar cierta incertidumbre que se puede evitar con un sistema de vigilancia teleoperado.

Otra de las aplicaciones directas para sistemas como el que se plantea, es facilitar muchos trabajos de alto riesgo como se ve en la figura 1 en donde una persona arriesga la vida al tratar de desactivar una bomba, en el caso particular de la vigilancia es de gran ayuda ya que muchas veces las personas dedicadas a esto, exponen hasta su integridad física en cumplimiento de su trabajo y la idea básicamente es tratar de exponer en lo menor de lo posible la salud personal y emocional.

Figura 1. Desactivación de una bomba.

1.3 Objetivos

El presente proyecto está centrado en los sistemas de interacción remota con robots móviles basada en internet. Los objetivos que se pretenden conseguir son:

1.3.1 Objetivo principal

Implementación de un sistema eficiente de teleoperación de un robot móvil utilizando tecnologías de software y hardware para la transmisión de comandos y videos en tiempo real a través de una interfaz web.

1.3.2 Objetivo particulares

• Evaluar las ventajas y los inconvenientes de utilizar internet como medio de comunicación en los sistemas de interacción remota con robots móviles.

• Evaluar las estrategias de control que se pueden aplicar para desarrollar sistemas de interacción remota basados en internet.

• Utilizar software libre para desarrollar una interfaz hombre-robot, extensible y reutilizable.

• Desarrollar una arquitectura de software utilizando el modelo cliente-servidor y arquitecturas de robots autónomos.

1.3 Alcance

En la creación de este sistema se utilizaran algoritmos de control y programación web para realizar operaciones teleguiadas en tiempo real para mejorar los tiempos de respuesta en la capa de transporte del modelo de referencia OSI (en inglés, open system interconnection).

Se implementará una interfaz gráfica en lenguaje HTML5, para el control del robot, utilizando una tarjeta de desarrollo GALILEO de Intel, como servidor embebido, para la recepción de comandos y control, de datos e información del usuario.

Se implementará un servidor de aplicaciones, en este caso usará Node.js, que es un tipo de lenguaje de programación orientado a eventos, que además está adaptado para trabajar de manera más eficaz en la comunicación con el servidor web. Para ello el servidor tiene instalados los módulos necesarios, están configurados de manera correcta todos los puertos y parámetros del sistema y

los MIME (extensiones multipropósito de correo de internet), como Java, flash, JavaScript y HTML5.

El sistema identifica varios tipos de usuarios, mediante un nombre de usuario y una contraseña, los cuales tienen diferentes privilegios sobre el sistema.

Por medio de la interfaz diseñada es posible el control de la navegación del robot mediante el envío de comandos específicos, al tiempo que se visualiza el video en tiempo real, captado por la cámara del robot.

Para fines de control y mantenimiento del robot guardará un histórico con la información de los sensores la cual podrá ser consultada desde la interfaz, sólo por los usuarios autorizados.

1.4 Limitaciones

A pesar de haber explorado las mejores alternativas, las limitaciones en el ancho de banda y alcance de cobertura de red LAN (local area network) y limitando el trabajo solo a la capa de transporte. En un futuro se espera hacer pruebas en una red WIMAX (interoperabilidad mundial para acceso por microondas) que es de mucho mayor alcance y su tasa de transferencia es mayor que la de una red WLAN (wireless local area network) y además es compatibles con LTE, que sería útil para la manipulación vía móviles (smartphone, tabletas y cualquier dispositivo portable).

Por otra parte en cuanto a la seguridad del sistema, está limitado ya que el lenguaje de programación que estamos usando aún está en desarrollo por lo que la seguridad no es fiable ya que los ciberataques prevalecen como el problema común dentro del internet, pues un mundo conectado a internet, abre huecos susceptibles a ataques cibernéticos y al robo de información, en cualquier sistema, aún si se tiene el mejor sistema de seguridad (Adrian McEwen, 2014).

Capítulo 2. Marco teórico

A lo largo de este capítulo se describirá el sustento teórico para el desarrollo de este proyecto, basado en los resultados de la unión de la electrónica y la informática para el control y monitoreo de robots móviles.

2.1 Robots autónomos y teleoperados

El término robot aparece por primera vez en 1921, en la obra teatral R.U.R (Rossum´s Universal Robots), donde la palabra robot significa fuerza de trabajo o servidumbre. Por aquellos años la producción en serie se había introducido en numerosas fábricas automotrices, textiles entre otras. Ya para ese entonces se discute el poder de las máquinas y la dominación de los hombres por las máquinas.

Debido a la necesidad de controlar procesos sin la intervención de agentes externos, especialmente del hombre, surge el control automático de procesos. La automatización industrial con sistemas de control automático comienza también en el siglo XIX pero no es hasta el siglo XX y muy especialmente, después de la segunda Guerra Mundial, cuando empieza a extenderse de forma importante en todos los sectores industriales (Baturone, 2005).

Para 1972 la aparición de los microprocesadores, suministra el impulso decisivo al control por computadora, haciendo rentables numerosas aplicaciones entre las cuales se encuentra el control de robots.

Los sistemas teleoperados se desarrollan en los años cuarenta para manejar materiales radioactivos. Consistían en un par de pinzas: "maestra" y "esclava" acopladas por mecanismos que permitían que las pinzas "esclava", que estaba en contacto con el material radiactivo, reprodujera las acciones de la pinza "maestra" accionada por un operador detrás de un muro protector con ventanas apropiadas para observar la operación.

El primer sistema teleoperado llamado *manipulador programable* fue accionado por servomecanismos eléctricos, se presenta en 1948. Después, en 1974 se introdujeron servosistemas con realimentación de fuerza hacia la pinza "maestra" para permitir que el operador percibiera el esfuerzo desarrollado.

A finales de los sesenta y principio de los setenta, la tecnología de la teleoperación alcanzó su mayor auge con la aplicación especializada. Apareciendo nuevos retos y problemas siendo de especial relevancia la existencia de retrasos temporales en la comunicación entre la zona local y la zona remota (Baturone, 2005).

Por otra parte la evolución de los sistemas como las telecomunicaciones, los sistemas mecánicos, eléctricos, fibra óptica, radio e internet, dio como resultado una mayor distancia de comunicación, entre robot y usuario.

También la incorporación de la tecnología multimedia ha permitido incrementar las capacidades del sistema remoto, especialmente en la interfaz gráfica y la transmisión de video.

Por otro lado hoy en día existen diferentes paradigmas y arquitecturas para la interacción con los robots. Un robot interactuando con el medio que lo rodea, se enfrenta con el problema de decidir qué acción tomar.

En las últimas dos décadas, en la inteligencia artificial se ha establecido que se pueda considerar inteligente a un sistema con las siguientes tres características (Murphy, 2000):

- Percepción: capacidad del robot para disponer de la información de los sensores
- Razonamiento o planificación: capacita al robot para producir las tareas a ejecutar, con base en la información de la entrada y al entorno que lo rodea.
- Ejecución o acción: capacita al robot para la ejecución de tareas mediante la interacción con la información.

2.2 Arquitectura de control para robots móviles

En esta parte abordaremos las dos arquitecturas hibridas más representativas en los robots autónomos (Arkin, 1998).

Existen cuatro criterios para poder comparar las diferentes arquitecturas y son los siguientes:

- Modularidad: se refiere a la independencia de los diferentes componentes.
- Adaptabilidad: se refiere a cómo se adapta la arquitectura a la aplicación que se diseñó.
- Portabilidad: Es la interoperabilidad con otros entornos u aplicaciones.
- Robustez: son los alcances y limitaciones de cada arquitectura.

2.1.1 Arquitectura aura

La arquitectura Aura (Autonomous Robot Architecture) es una de las más antiguas, usada para el control de un robot. Está basada en la teoría de esquemas, se subdivide en dos niveles: nivel de control deliberativo y nivel de control reactivo.

El nivel deliberativo está compuesto a su vez por dos subniveles, el cartográfico y el de planificación.

El nivel de control reactivo está basado en un sistema de esquemas caracterizado según Arkin por los siguientes puntos (Arkin, 1998). Los esquemas forman una red dinámica de procesos y cada esquema es un agente computacional independiente.

La selección de los patrones de comportamiento se realiza en el nivel deliberativo de planificación, usando información sobre la misión, el entorno y el estado interno del robot. Con base en la lista de patrones de comportamientos proporcionada (figura 2) por el nivel deliberativo, en el nivel reactivo selecciona los esquemas de percepción y esquemas motores que formarán los comportamientos.

El comportamiento exhibido por el robot se obtiene de la composición de los componentes básicos (Yagüe, 2003).

Su diseño está motivado por el estudio de sistemas biológicos.

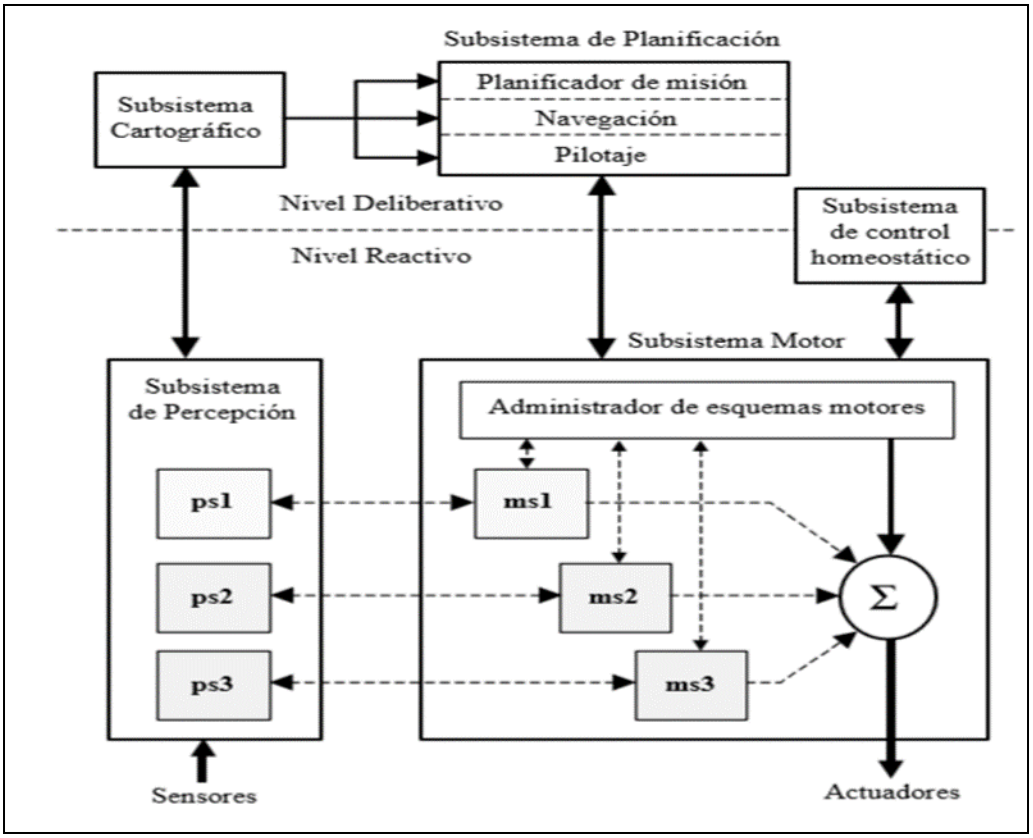

Figura 2. Diagrama a Bloques de la Arquitectura AURA (Yagüe, 2003).

2.1.2 Arquitectura SFX

La arquitectura SFX (Sensor Fusion Effects) fue creada por Robin R.Murphy, esta arquitectura está formada por una unión sensorial y la recuperación de información de errores de sensores, es una mejora de la arquitectura AURA donde el nivel reactivo como el deliberativo, comparten información al mismo nivel, para poder resolver más rápidamente cualquier falla.

Tanto el nivel reactivo como el deliberativo se manejan en un sistema de pizarras, en el cual los dos subsistemas comparten la información sensorial. La siguiente figura 3, muestra el diagrama a bloques de la arquitectura SFX.

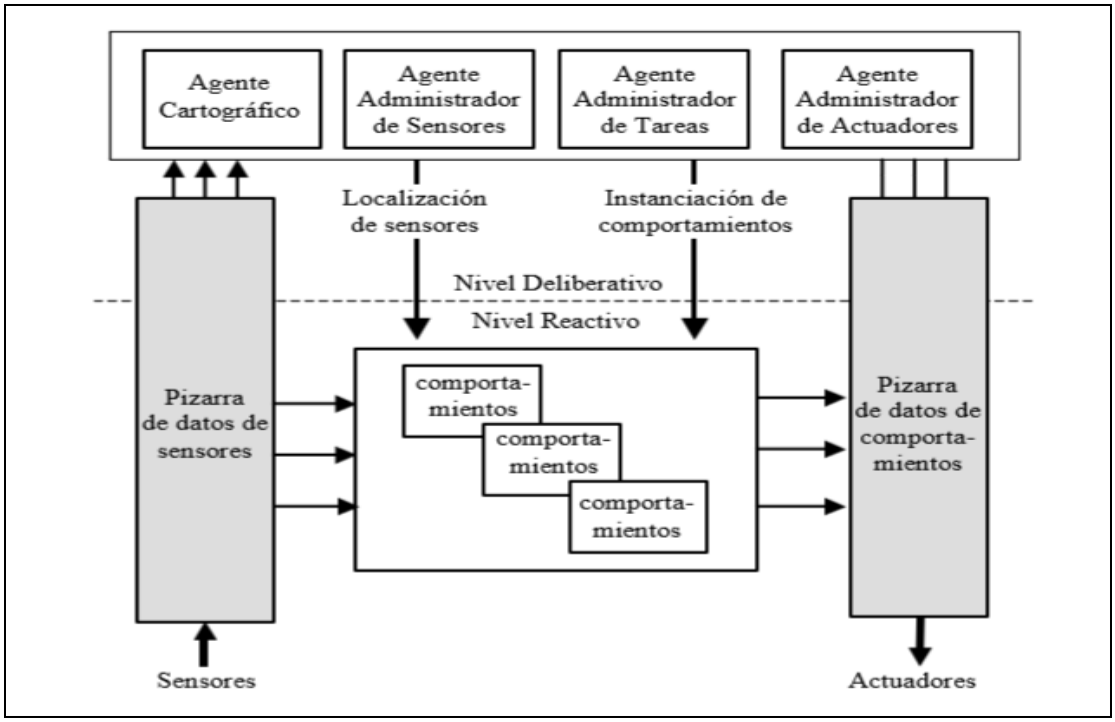

Figura 3. Diagrama a Bloques de los componentes de la Arquitectura SFX (Yagüe, 2003).

El nivel deliberativo es dividido en módulos o clases de objetos como se muestra en la figura 4, también contiene un software independiente especializado denominado agente, este software es programado específicamente para realizar funciones e interactuar con otros agentes. Su tarea principal es planificar tareas. Estos agentes interactúan con el usuario y sirven para marcar las restricciones de las tareas con otros agentes en el nivel deliberativo (Murphy, 2000).

La labor de este software agente también es analizar si hay una falla en algún sensor o en la realización de una tarea y además podrían averiguar por qué falló el sensor y solucionar el problema de manera temporal sustituyéndolo por un sensor alternativo.

Por otra parte el nivel reactivo está dividido en dos sub capas, identificadas por los comportamientos estratégicos y por los comportamientos tácticos. A diferencia de la arquitectura AURA, la arquitectura SFX mezcla comportamientos fundamentales, de tal forma que es capaz de inhibir otros comportamientos secundarios. Por ejemplo en la arquitectura AURA la velocidad de reacción es la suma de todos los componentes y la en la arquitectura SFX la velocidad de

reacción siempre es menor, debido a que el comportamiento de esta arquitectura está basada en movimientos tácticos e inhibidores de acciones, que permiten filtrar la información de tal manera que el sistema conozca la mejor solución con un comportamiento estratégico.

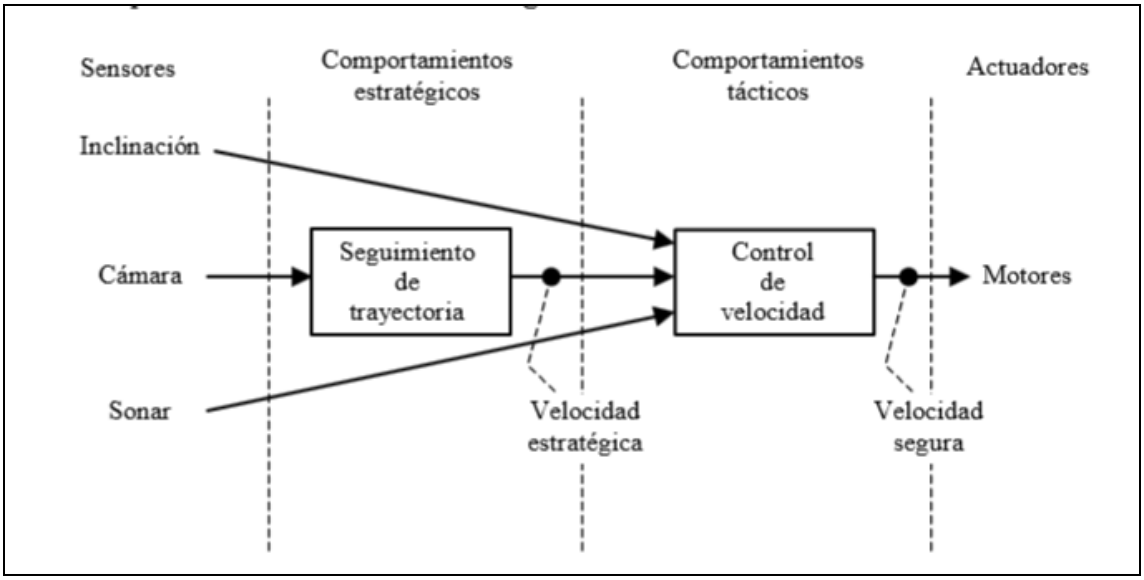

Figura 4. Diagrama de inhibición de comportamientos de la Arquitectura SFX (Yagüe, 2003).

En la figura 4 se muestra un diagrama a bloques de cómo es que está organizado el sistema de inhibición de comportamientos, que permite a la arquitectura SFX, tener una mejor velocidad de respuesta.

2.1.3 Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo que se emplea en sistema de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para el intercambio de información de servicios o recursos.

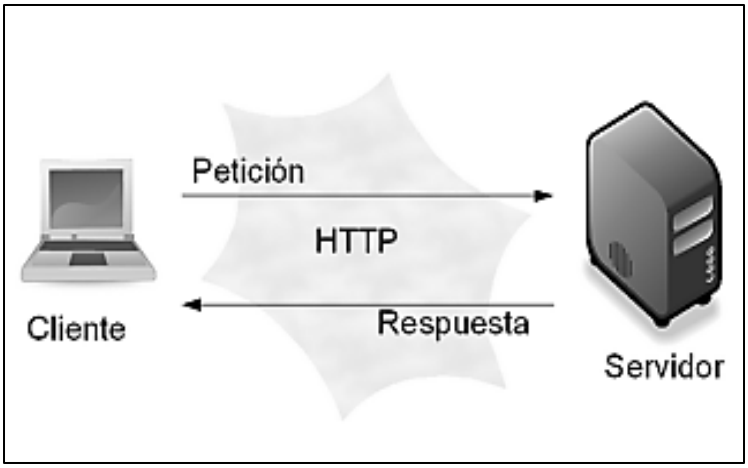

Figura 5. Arquitectura Cliente-Servidor. (web, 2012)

En la figura 5, se muestra la arquitectura cliente-servidor, dónde el servidor es el provedor de servicios y los clientes son los solicitantes de los servicios.

- Cliente: Estará creado principalmente por lenguaje HTML, el cual forma la página web y además contiene código(script´s) adicional para realizar peticiones al servidor, los script´s están desarrollados con programación orientada a eventos. Otras tecnologías usadas ademas de HTML para el desarrollo del programa del cliente son:

  I. CSS 3
  II. JavaScript
  III. Socket.io
  IV. JQuery

- Servidor: Es el programa que atendera las peticiones del cliente, este espera permanentemente solicitudes de conexión mediante el protocolo HTTP, el cual será el encargado de contestar y actuar a las peticiones del cliente.

La idea de hablar de arquitecturas de robots autónomos y la arquitectura cliente-servidor, es para tener un panorama más amplio para el diseño de nuestro

sistema de teleoperación. Se eligió usar la arquitectura AURA para proveer un esquema correcto de cómo integrar todos los elementos al sistema, ya que esta arquitectura es muy básica a diferencia de la arquitectura SFX que es más compleja.

Para la transmisión de información a través de internet se usó la arquitectura *cliente-servidor* que más adelante se abordará con más detalle cómo se usaron.

2.3 Interacción remota

La interacción remota es cuando el hombre-robot está separado por medios físicos o por diferentes escenarios o lugares. La interacción remota se divide en varias etapas y procesos que permiten llevar a cabo la comunicación, con lo cual se deriva la teleoperación (Arkin, 1998).

Este tipo de interacción consta de los componentes básicos que se muestran en la figura 6 para la realización de este proceso.

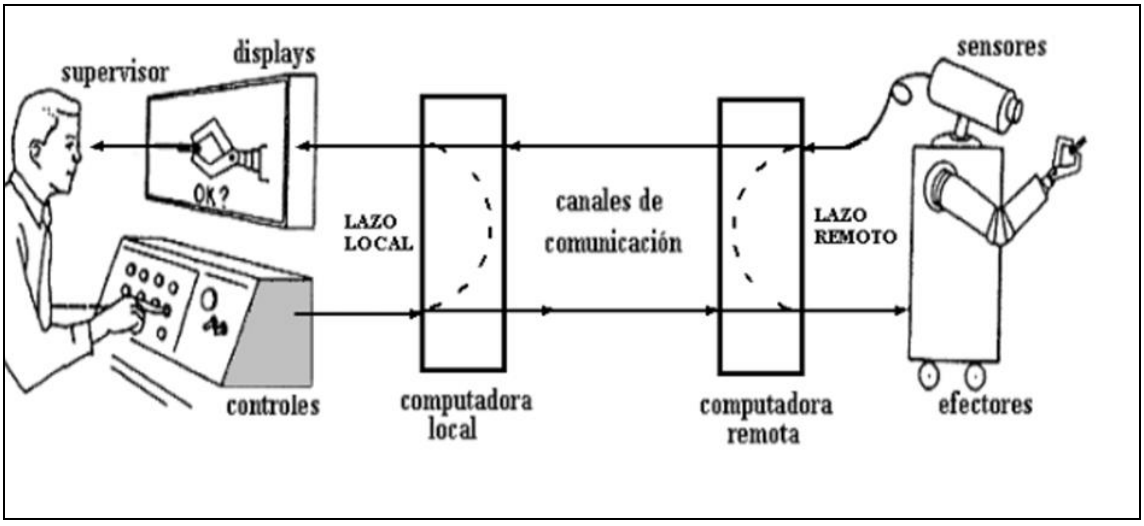

Figura 6. Elementos básicos de un sistema de interacción remota.

Elementos básicos que conforman la interacción remota (Baturone, 2005):

- Interfaz de control: es usada para mandar y recibir información al robot, y existen diferentes dispositivos de control con los que el usuario puede

comunicarse, como por ejemplo: computadoras personales, laptops, Smartphone y tablets.

- Robot móvil: dispositivo que se encarga de ejecutar las órdenes enviadas por el usuario y a su vez de enviar información para su uso o almacenamiento, el cual está conformado por sensores, cámaras, motores y microprocesadores.
- Interfaz de realimentación: La realimentación es básicamente regresar la información adquirida por el sistema sensorial del entorno y sirve para corregir errores de su trayectoria y mantener actualizado el sistema para evitar una posible falla.

2.3.1 Teleoperación

Los robots teleoperados son aquellos controlados por un usuario a distancia desde una estación remota. Dada su gran utilidad, se han empleado en diversos campos, este tipo de manejo supone una ventaja desde el punto de vista de la protección y seguridad del usuario, ya que en caso de realizar trabajos en ambientes inseguros o inestables o con químicos o explosivos, no se arriesga la integridad física de las personas.

En el desarrollo de robots teleoperados se involucran la electrónica, las comunicaciones, el control, la inteligencia artificial y la visión artificial, entre muchas otras disciplinas de la ingeniera.

Los sistemas teleoperados son clasificados en 3 categorías (Rashwan, 2003):

- Teleoperación de rango corto: En estos sistemas, la distancia entre el operador y el sitio remoto está restringida por la necesidad del operador de ver el entorno remoto directamente. En este caso, no hay restricción en el flujo de información entre los dos sitios y existe un retraso mínimo de la comunicación.

- Teleoperación de rango medio: En los sistemas de teleoperación de rango medio, el sistema eléctrico de teleoperación de rango corto se combina con medios para permitir ver el sitio remoto a distancia. La adición de cámaras y monitores significa que la separación entre el operador y el entorno remoto podría aumentarse considerablemente. En tales sistemas, la conexión entre los sitios es completamente eléctrico. El operador puede ver lo que está pasando vía una cámara y un monitor de televisión, puede escuchar lo que pasa vía un micrófono y altavoces, y puede sentir lo que está pasando vía una interfaz.

- Teleoperación de rango largo: Con el aumento de la distancia entre dos sitios, el retraso de comunicaciones aumenta hasta el punto donde los sistemas de teleoperación de rango corto y medio fallan. En el sistema de teleoperación largo, la realimentación juega un papel importante que es mantener una comunicación persistente en caso de que haya una desconexión y el sistema no pierda información y se mantenga el funcionamiento del sistema.

En la actualidad estos sistemas pueden mostrar un gran atractivo para mucha gente debido al uso de la web, que hace que estos sistemas puedan ser utilizados por dos razones importantes. En primer lugar, que pueden ser usados desde cualquier navegador, con una interfaz gráfica amigable al usuario y de fácil uso y la tendencia de poder acceder a estos robots desde cualquier parte del mundo. Además de esto se incluyen varios elementos de información para el usuario como imágenes, películas, sonidos, y estadísticas entre otras aplicaciones como la interacción misma con estos sistemas.

En segundo lugar el protocolo para la transferencia de hipertexto (HTTP), que es un protocolo de comunicación estándar. Este protocolo es el más usado para la conexión a internet y la compatibilidad con cualquier sistema final, ya que todos los sistemas usan el mismo protocolo para la conexión a la red sin tantas restricciones lo que facilita la compatibilidad del robot para comunicarse con la computadora (Rashwan, 2003).

Hoy en día las nuevas tendencias y avances tecnológicos dan lugar al uso de robots teleoperados en muchas áreas, aplicaciones y tareas específicas, como la desactivación de bombas y manejo de sustancias peligrosas.

2.4 Aplicaciones web en tiempo real

Una aplicación web en tiempo real, es aquella que responde de inmediato a las interacciones de los usuarios y que otros usuarios ven reflejados en un valor promedio de milisegundos, que a la vista del usuario parece que el cambio es instantáneo.

La web en tiempo real es algo que millones de personas ven todos los días en sus aplicaciones favoritas, pero la mayoría no se da cuenta de dicha funcionalidad. Empresas como Facebook y Twitter usan los datos en tiempo real para actualizar los flujos de actividad, lo que resulta en una experiencia menos estática y más como la mensajería instantánea (Teixeira, 2012).

El transporte de los datos en tiempo real se usa en aplicaciones web como chats, juegos multijugador, paneles de control, y las experiencias de segunda pantalla. Dos principales beneficios de la web en tiempo real son que se incrementa la participación de los usuarios y se reduce la carga del servidor. Es decir que a diferencia de las tecnologías anteriores que usan una comunicación unidireccional, alta latencia y peticiones independientes, que hacen el proceso de la comunicación entre el servidor y el cliente muy ineficientes, la nueva forma de transmitir información bidireccional y de baja latencia hace posible las aplicaciones en tiempo real.

Las aplicaciones de hoy día usan este tipo de tecnologías para transmitir información a gran velocidad. Puesto que las exigencias del usuario cada día son más grandes, para ello hay varias tecnologías que se están desarrollando basadas en sockets de transmisión, API´s (application programming interface) y diferentes mecanismos de transporte, para poder crear este tipo de aplicaciones.

2.5 Socket

Un socket es la parte intermedia (middleware) en la comunicación cliente-servidor, que es un mecanismo de transporte (la manera en que se realiza la transmisión de datos) de dicha comunicación, lo cual permite que un proceso emita o reciba información con otro proceso estando en diferentes conexiones y esto hace que los programas puedan transmitir datos, para poder ejecutar diferentes acciones o procesos.

El socket queda definido por una dirección IP, un protocolo de conexión como un puerto origen y un puerto destino. El origen y el destino vienen especificados por la IP y el puerto en el cual el socket va a escuchar; en cuanto al protocolo de comunicación, maneja TCP/IP y UDP entre Cliente-Servidor.

El proceso mediante el cual se lleva a cabo la comunicación vía sockets, se genera mediante los siguientes pasos:

1. El proceso en el servidor es crear un socket con nombre y esperar la conexión.
2. El proceso cliente crea un socket sin nombre.
3. El proceso cliente realiza una petición de conexión al socket servidor.
4. El cliente realiza la conexión a través de su socket mientras el proceso servidor mantiene el socket servidor original con nombre.

Los datos enviados en la transmisión por el socket tienen las siguientes características:

- Fiabilidad de transmisión.
- Mantenimiento del orden de los datos.
- No duplicación de los datos.
- El "modo conectado" en la comunicación.
- Envío de mensajes urgentes.

Los tipos de socket depende del tipo de comunicación que se desea usar para el envió de la información como por ejemplo:

SOCK_DGRAM: Este socket es para comunicaciones en modo no conectado, con envío de datagramas de tamaño limitado, en donde el protocolo del nivel de transporte se basa es el UDP (user datagram protocol) (Raus, 2000).

SOCK_STREAM: Socket para comunicaciones fiables en modo conectado, de dos vías y con tamaño variable de los mensajes de datos. Por debajo, en dominios Internet, subyace el protocolo TCP.

SOCK_RAW: Socket que permite el acceso a protocolos de más bajo nivel como el IP (nivel de red)

SOCK_SEQPACKET: Este socket tiene las características del SOCK_STREAM pero además el tamaño de los mensajes es fijo.

Para este proyecto se usará socket.io que es un socket basado en SOCK_STREAM, éste tipo de socket usa el protocolo TCP (transmission control protocol) que garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También es compatible con varios mecanismos de transporte.

2.6 Socket.IO

La intención de esta librería es ofrecer un soporte universal de comunicación en tiempo real, sea cual sea el mecanismo de transporte que el usuario utilice. Los diferentes mecanismos de transporte, que utiliza la biblioteca Socket.IO se mencionan en la sección 2.6.1.

La manera de utilizar socket.io en la parte de servidor es asociándolo a un servidor HTTP, con una función llamada *listen (),* que realiza la conexión entren el servidor y el cliente, además es útil sobre todo si se trabaja con otras librerías para desarrollo web.

2.6.1 Mecanismos de transporte

Cuando se solicita la conexión, el tipo de navegador decide qué mecanismo de transporte es el que va a utilizar de todos los disponibles o de aquellos que el usuario elija. Se entiende por mecanismos de transporte a la tecnología empleada para realizar la conexión de un socket en la arquitectura cliente-servidor, según esté soportada por el agente de usuario. Hay una lista bastante amplia que cubre los distintos tipos de agente de usuario. Los diferentes mecanismos de transporte son:

- Websocket
- Adobe Flash Socket
- AJAX Long Polling

2.6.1.1 Websocket

Websocket engloba tanto el protocolo websocket, definido en el documento RFC6455, como el API que permite a las páginas web el uso del mismo (Melnikov, 2011).

Un websocket es una tecnología que ofrece una comunicación bidireccional y full dúplex cliente-servidor. La comunicación es sobre un único socket TCP, que no envía cabeceras HTTP, lo cual reduce él envió de paquetes. Además la comunicación es persistente, es decir, que si la conexión se pierde el socket se reconecta al servidor en intervalos de tiempo determinados, permitiendo al cliente estar conectado todo el tiempo con la mínima perdida de información.

El WebSocket puede ser considerado por algunos como una mejora de Ajax, pero es en realidad una alternativa totalmente diferente de comunicación que permite

la construcción de aplicaciones en tiempo real en una plataforma escalable por ejemplo video juegos para múltiples jugadores. (Diego, 2012).

El API de websocket es un conjunto de métodos, procesos y eventos, que permiten manejar un websocket, además y más importante es el uso de un constructor, que es el que entablará la comunicación.

2.6.1.2 Adobe Flash Socket

Adobe Flash implementa sockets usando como protocolo de control y transporte (TCP). Las aplicaciones Flash se pueden conectar a otro proceso que actúe como servidor de socket, pero no puede aceptar solicitudes de conexión entrantes de otros procesos. Flash sólo puede servir como socket y no como servidor.

La API de Flash también contiene una clase XMLSocket. La clase XMLSocket utiliza un protocolo específico de Flash Player que permite intercambiar mensajes XML con un servidor que comprende dicho protocolo. La clase XMLSocket se introdujo el lenguaje de programación ActionScript 1 perteneciente a Adobe y sigue admitiéndose para tener compatibilidad con versiones anteriores. En general, la clase socket debe utilizarse para aplicaciones que no se estén conectado a un servidor creado específicamente para comunicarse con XMLSockets de Flash.

El Socket TCP proporciona un modo de intercambiar mensajes a través de una conexión de red. El protocolo TCP garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. Las conexiones TCP, requieren de la arquitectura "cliente-servidor".

2.6.1.3 AJAX

Las aplicaciones construidas con AJAX (JavaScript asíncrono y XML) eliminan el recargar constantemente las páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX

mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se envía en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado, en el que normalmente se efectúan las funciones de llamada de AJAX mientras que el acceso a los datos se realiza mediante XMLHttpRequest que es un objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté en formato XML (ajax, 2008).

En la figura 7, se muestra una comparativa del modelo tradicional de una aplicación web propuesto por AJAX, del lado izquierdo observamos el modelo tradicional, en la parte derecha se muestra el nuevo modelo con la nueva tecnología.

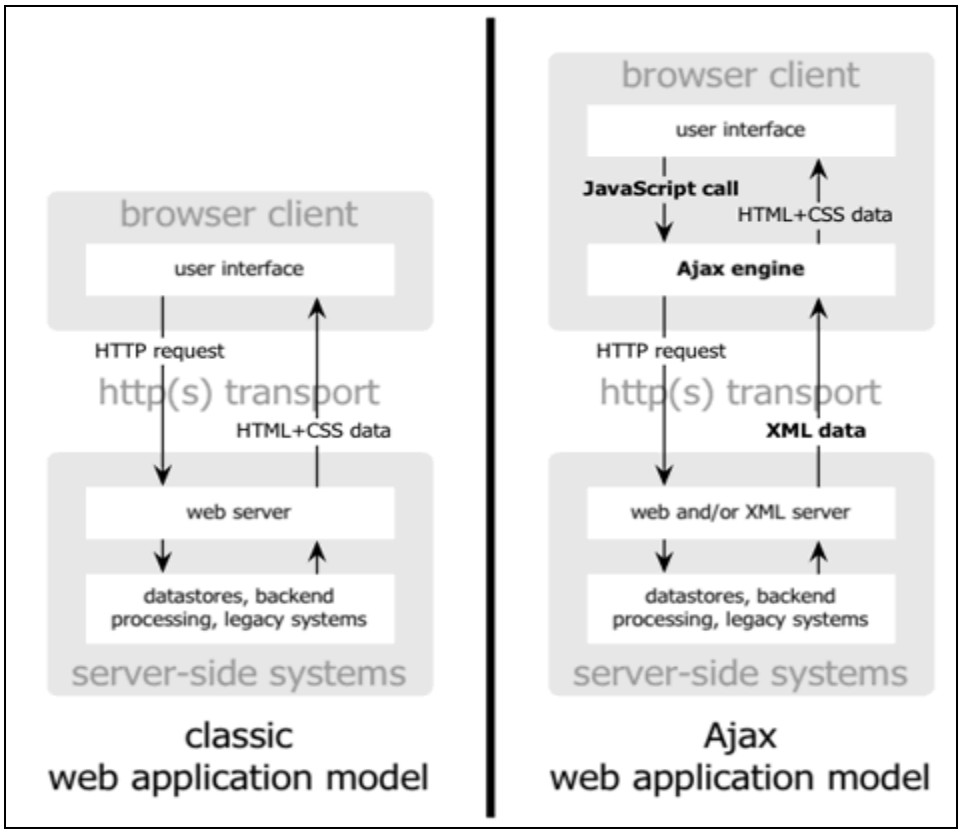

Figura 7. Comparación gráfica del modelo tradicional de aplicación web y del nuevo modelo propuesto por AJAX (AJAX, 2000).

En la técnica tradicional al realizar peticiones continuas al servidor, el usuario debe esperar a que se actualicé la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto y poco eficiente ya que se pierde mucho tiempo y se vuelve algo incómodo para el usuario.

AJAX en cambio permite mejorar completamente la interacción del usuario con la aplicación, evitando las actualizaciones constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

2.7 Transmisión de video vía internet

La transmisión de video a través de internet, involucra el uso de la tecnología streaming. Esta es una tecnología que permite la distribución de archivos multimedia principalmente audio y video a través de la red en tiempo real.

Los tipos de reproductores de video streaming y la selección del reproductor que se puede emplear y cuál resulta mejor y es más compatible con la mayoría de los navegadores actuales, sin la necesidad de la instalación de algún complemento adicional para su uso correcto.

Para transmitir video de manera fluida a través de la red, se necesita por lo menos tener un ancho de banda mínimo igual a la tasa de transferencia, de lo contrario habrá retraso de la información. Esta tecnología almacena el video en un buffer de datos donde se reproduce al mismo tiempo que se descarga y luego se descarta, sin quedar almacenado en el disco duro del cliente como se ilustrar en la figura 8.

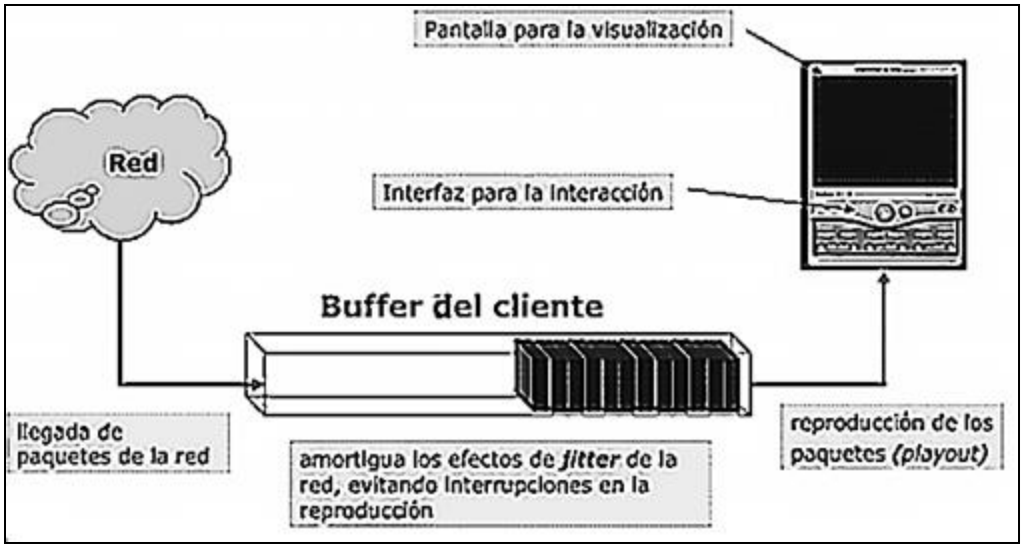

Figura 8.Transmisión de video (González, 2013).

Los componentes básicos de un sistema streaming son (González, 2013):

• Codecs: son archivos residentes en el host cliente que interpretan el contenido multimedia y hacen posible su reproducción.

• Protocolos: RTP, RTCP, TCP, UDP. Siendo los protocolos UDP, RTP (Real Time Transffer Protocol) los más importantes, ya que hacen que la entrega de paquetes de datos desde el servidor al cliente sea más fluida que con los protocolos HTTP y TCP (aplicaciones tolerante a perdidas).

• Precarga: el cliente precarga el archivo multimedia en un buffer de datos antes de reproducirlo, con el fin de evitar interrupciones y mala calidad de reproducción debido al tráfico en la red.

• Red de datos: si un determinado contenido comienza a atraer una cantidad de usuarios mayor que su capacidad de ancho de banda estos usuarios sufrirán interrupciones (también conocido como Lag). Finalmente, se llega a un punto en que la calidad del streaming es pésima.

• Segmentación: la información multimedia es segmentada, para luego ser enviada como paquetes a la red los cuales el cliente recibe y reproduce de forma fluida.

2.7.1 Compresión de video

En la transición de datos de video multimedia mediante la red es necesario hacer un procesamiento digital, para la conversión A/D (analógico-digital) y el tratamiento de la imagen. La digitalización de los datos se debe a que hoy en día se transmite en bits. La compresión de los datos es muy importante porque los datos multimedia (video) contienen mucha información y consumen muchos recursos y ancho de banda, para lo cual es necesaria la compresión para reducir los datos para transmitir la información.

Por ejemplo para transmitir una imagen de 1024 píxeles, en la que cada pixel es representado por 24 bits, de los cuales el color rojo, azul, verde, usan 8 bits por color, implica 3MB para su representación sin compresión. Si se aplica una tasa de compresión de 1:10 la imagen solo necesitará 300 KB para ser representada.

La finalidad de la compresión es reducir la información irrelevante en la señal de video, esto significa que el sistema codifica caracteres de mayor importancia perceptiva y elimina la información irrelevante ahorrando bits de información innecesarios.

Para la compresión de video, tomando en cuenta que el video es una sucesión de imagines que se transmite con una velocidad constante por ejemplo de 20 a 30 imágenes por segundo y cada imagen se representa como una sucesión de pixeles los cuales se representan con un numero de bits los cuales indican el color y luminosidad de cada imagen.

Para la compresión de video hay muchos estándares, el MPEG es uno de los más usados el cual incluye varias versiones, como el MPEG 1 que se usa la compresión de CD de video, otro es el MPEG 2 para DVD y por ultimo está el MPEG 4 que para compresión orientada a objetos. Otros estándares no tan populares pero también usados son el H.261 y QuickTime.

2.7.2 Formatos de video

Existen muchos formatos diferentes de video, a continuación se ve en la tabla la comparativa de los formatos más usados de video y de sus características

generales, los formatos más usados para transmitir video en internet son Windows Media, Flash Video, QuickTime Movie, MPEG y Audio/Video Interleaved (Roca, 2001).

Formato de video	Desarrollador	Streaming	Algoritmo de compresión
MPEG	Moving picture expert group	si	MPEG-1, MPEG-2, MPEG-4
Windows Media	Microsoft	si	WMV, ASF
QuickTime Movie	Apple	si	MOV
Flash Video	Adobe	si	FLV, SWF
AVI	Microsoft	si	AVI
MJPEG		si	JPEG

Tabla 1. Comparación de formatos de video.

2.8 Acceso a video desde la web

Para acceder a archivos de video desde la web, el servidor comparte este tipo de archivos tratándolos como si fuera cualquier objeto, por ejemplo páginas HTML e Imágenes. Cuando se utiliza un navegador para acceder a los archivos de audio o video. Este acceso establecerá una conexión TCP con el servidor y una vez establecida esta conexión, pedirá el recurso solicitado con un mensaje de petición de requerimiento HTTP. El servidor genera un mensaje de respuesta HTTP donde estará encapsulado el archivo solicitado. La recepción del archivo, hará que se abra un programa reproductor y cuando el buffer contenga la suficiente información empezará la reproducción.

La figura 9 muestra el proceso de petición y respuesta de un archivo de audio o video (Ordinas, 2008).

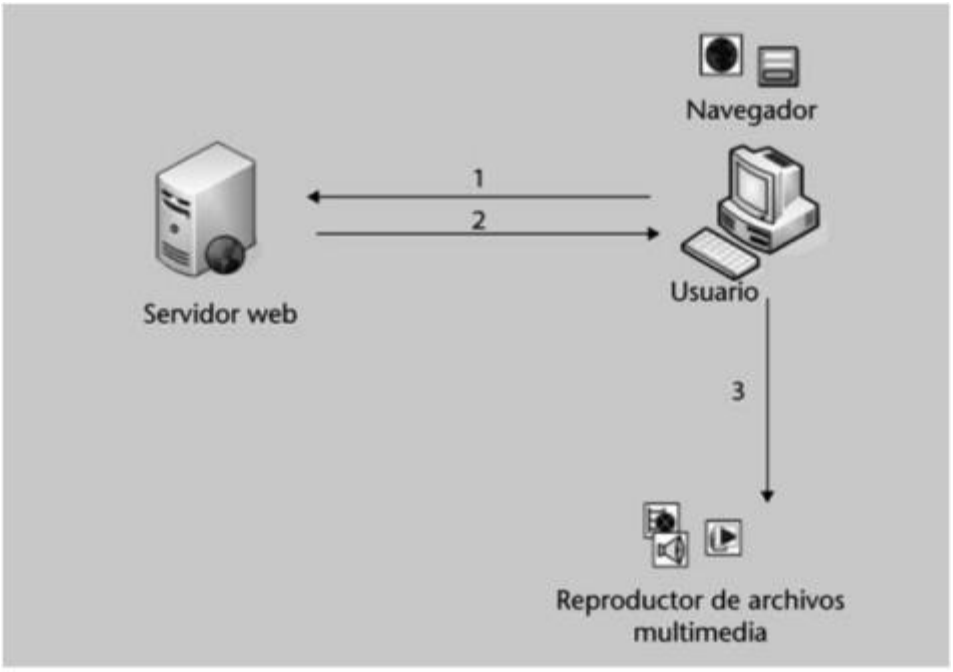

Figura 9. Comunicación cliente-servidor. (Ordinas, 2008)

Por otra parte, en caso de que los archivos no hayan sido recibidos correctamente o la calidad del video sea muy alta para poder reproducirlo, lo que se hace para evitar la pérdida es que el servidor se conecte directamente al reproductor pasándole los parámetros de forma directa, de esta forma se evita que se pierda el archivo y no pueda reproducirse. La manera de enviar el video directamente al reproductor, es cambiando el recurso multimedia por un archivo de metadatos. Un metadato es un campo de texto que va incrustado en un archivo la información de donde se encuentra el recurso multimedia y se puede añadir información adicional como el tamaño, autor, fecha e incluso el tipo de codificación de archivo multimedia. De esta manera el navegador recibe el archivo de metadatos y se lo pasa al reproductor, el reproductor se encarga de conectar de forma directa con el recurso e incluso puede iniciar antes el reproductor sin tener todo el archivo de metadatos completo.

2.9 Protocolos de transporte streaming

Para la transición de streaming también denominado transmisión, lectura en continuo, difusión en flujo, lectura en tránsito, difusión en continuo, descarga

continua, entre otros, se han desarrollado diferentes protocolos para enviar flujos de datos en tiempo real, como RTP, RTCP, RTSP entre otros. A continuación se describe los protocolos usados en el presente trabajo.

2.9.1 RTP

RTP (Real-time Transport Protocol), es un protocolo de transporte diseñado para la transición de recursos en tiempo real (streaming). Algunos de estos recursos son la música, videoconferencia, el video, la telefonía en Internet y más aplicaciones multimedia (Perkins, 2003).

La función del protocolo RTP es multiplexar varios flujos de datos en tiempo real en un solo flujo de paquetes UDP, en el cual envía tanto a un sólo destino (unicast) o múltiples destinos (multicast). Los paquetes son numerados asignando a cada paquete un número mayor que su antecesor. Esto será útil para que la aplicación conozca si existe algún error en el paquete en la transmisión. Si ha fallado, al no tener un control de flujo, de errores, de confirmaciones de recepción ni de solicitud de transmisión, la mejor opción es la interpolación de los datos. También RTP permite controlar la velocidad de transición para que el flujo de datos sea servido con la velocidad correcta con el mínimo de tiempo.

2.9.2 RTCP

El protocolo RTCP es complementario a RTP y le brinda a éste un mecanismo de control. Utiliza el protocolo UDP por el puerto adyacente siguiente al puerto que se utiliza para RTP.

El protocolo RTCP se basa en la periódica transmisión de paquetes de control a todos los participantes en sesión ofreciéndole información sobre la calidad de los datos distribuidos por la fuente. El protocolo subyacente debe proveer de la multiplicación de los datos y de los paquetes de control. Por tanto, la función primordial de RTCP es la de proveer una realimentación de la calidad de servicio (Gil Cabezas, 2008).

2.9.3 RTSP

RTSP es un protocolo que establece y controla tanto uno como varios streams sincronizados de datos multimedia, en lugar de esto el servidor mantiene una sesión asociada a un identificador, en la mayoría de los casos RTSP usa TCP para datos de control del reproductor y UDP para los datos de audio y vídeo aunque también puede usar TCP en caso de que sea necesario. En el transcurso de una sesión RTSP, un cliente puede abrir y cerrar varias conexiones de transporte hacia el servidor.

De forma intencionada, el protocolo es similar en sintaxis y operación a HTTP de forma que los mecanismos de expansión añadidos a HTTP pueden, en muchos casos, añadirse a RTSP. Sin embargo, RTSP difiere de HTTP en un número significativo de aspectos:

- RTSP introduce nuevos métodos y tiene un identificador de protocolo diferente.
- Un servidor RTSP necesita mantener el estado de la conexión al contrario de HTTP
- Tanto el servidor como el cliente pueden lanzar peticiones.

El esquema de funcionamiento de RTSP se observa en la figura 10.

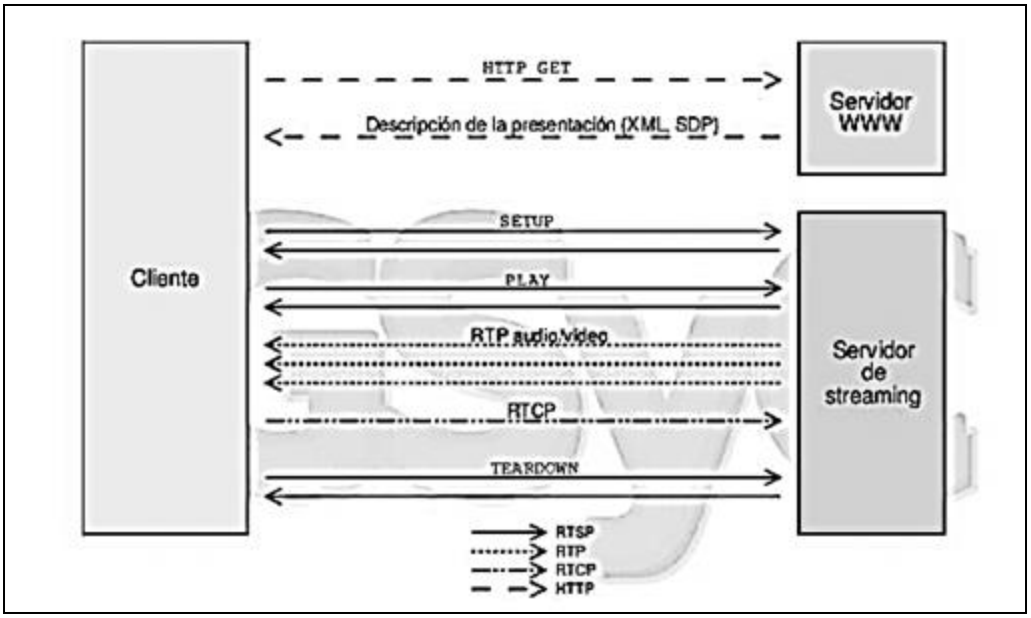

Figura 10. Transmisión RTSP. (Rodríguez, 2007)

2.10 Reproductor de streaming

En esta sección se presenta los reproductores de streaming más usados y sus propiedades más destacadas y algunos reproductores como QuickTime Movie, Windows Media Player y Real Networks. Cada uno de éstos cuenta con una gran variedad de codecs para poder reproducir la gran mayoría de archivos multimedia.

2.10.1 Apple quicktime

QuickTime es un reproductor multimedia gratuito, que es utilizado para visualizar diferentes tipos de archivos de audio, video, e imágenes estáticas, streaming. QuickTime es compatible con los formatos más usados en internet.

QuickTime tiene las siguientes funciones:

- Grabar archivos de internet
- Edición de Audio y Video
- Grabar Audio y Video
- Añadir efectos especiales
- Convertir y guardar video, audio e imagines en más de un formato.
- Compatibilidad con H.264

Además QuickTime incluye una tecnología llamada instantáneo, es una tecnología que reduce el tiempo del buffer cuando se visualiza video en tiempo real, que envía las imagines de manera fluida como si estuvieran ya almacenadas en el disco duro, también se puede modificar el tiempo de espera.

2.10.2 Real networks

Real Networks comenzó en 1995 antes conocido como Progressive Networks, lanzando Real Audio Y Real Video para la transición de contenidos multimedia a través de internet.

Real Networks desarrollo el reproductor real time (antes llamado RealPlayer), el cual ha adoptado el protocolo RTSP, el cual ofrece un estándar para una gran variedad de datos, con el propósito de promover una mayor interoperabilidad entre varias comunicaciones streaming. (Rodríguez, 2007).

Real time tiene las siguientes características:

- Almacenamiento en la nube
- Retransmisión de archivos a todos los dispositivos sin codificación de los archivos.
- Real cloud permite tener una copia de seguridad.
- Descargar contenidos desde la Web

También RealPlayer, cuenta con una versión Premium que permite la retransmisión y grabación en alta definición, reproducción de contenidos en HD, y retransmisiones en TV si se cuenta con aplicaciones como Roque o Chromecast

2.10.3 Windows media player

Windows Media player es un reproductor de Microsoft para la reproducción de todo de archivos multimedia usados en internet. La arquitectura de Windows contiene una extensa biblioteca de codec´s y servicios para la reproducción y distribución de archivos multimedia.

Las características de Windows media player son:

- El narrador es un programa que permite convertir voz en texto
- Uso de redimensionamiento de objetos para facilitar su visualización

- DVS es un programa que ofrece información adicional de las imagines que aparecen en el reproductor o TV.
- Ecualizador grafico de 10 bandas para modificar el audio.

Para este trabajo RealPlayer es la solución más completa, ya que este reproductor, es muy fácil descargarlo y por lo general ya viene instalado de fábrica para PC y tiene una gran variedad de códecs.

2.11 Servidores web

Para la programación del servidor, es necesario explicar qué es un servidor, servidor es una palabra que se usa para darle un nombre a un software que probé una gran cantidad de servicios como:

- Servicio web
- Servicio de correo
- Servicio de base de datos.

En la actualidad existen una gran variedad de servidores que se muestran en la tabla 2.

Servidor	Descripción	Plataforma	Código libre	Complejidad
Apache	Servidor web de propósito general	Windows, Linux, Mac	Si	Multiplataforma Extensible (libre). Modularidad Configuración difícil.

IIS	Servidor web y de alto rendimiento	Windows	No	Extensible (no libre). Modularidad Configuración media.
Node.js	Servidor web de propósito general	Windows, Linux, Mac	Si	Multiplataforma Extensible (libre) Modularidad. Configuración fácil.
NES	Servidor web de alto rendimiento	Unix	No	Multiplataforma Extensible (no libre). Modularidad. Configuración difícil.

Tabla 2. Software de servidores

Se decidió usar Node.js que en comparación con los otros servidores donde cada conexión (petición) genera un nuevo hilo, ocupando memoria RAM del sistema y, finalmente genera un gasto excesivo de salida en la cantidad de RAM disponible. Node.js opera sobre la misma conexión usando el mismo hilo para atender varias peticiones y esto que le permite soportar decenas de miles de conexiones simultáneas, en la figura 11 vemos la comparativa de conexión entre Node.js y los otros servidores (CAPAN, 2015).

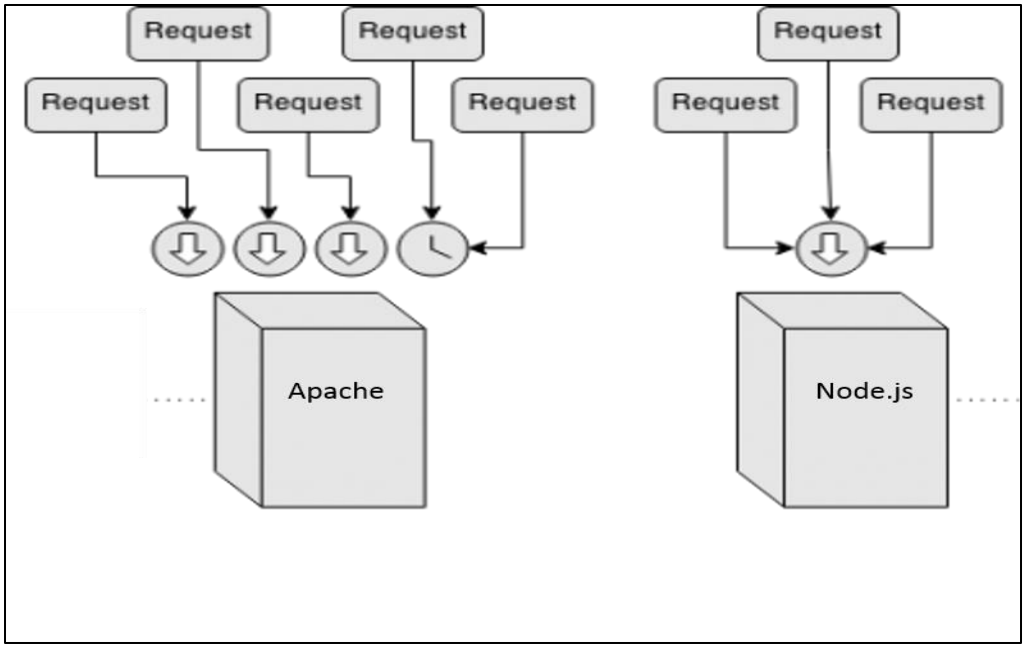

Figura 11. Comparación en el manejo conexiones en Node.js y un servidor Apache (Teixeira, 2012).

Otra ventaja que hayamos en el uso de Node.js, es que se pueden agregar manualmente todas las bibliotecas según las necesidades del servidor a diferencia de Apache que cuando es instalado descarga todas las bibliotecas aunque no las use y esto genera consumo en la memoria del disco del servidor.

2.11.1 Códigos de estado de un servidor

La siguiente es una lista de códigos de respuesta de un servidor y frases estándar asociadas, destinadas a dar una descripción corta del estatus.

100: continuar.

Probé informar al cliente de que el servidor acepta el tipo de petición en base a las cabeceras que le está enviando y puede continuar transmitiendo el cuerpo del mensaje.

2OO: éxito.

La petición fue recibida y procesada satisfactoriamente sin errores.

304: no modificado.

Indica que la petición no ha sido modificada desde que fue requerida por última vez y se entregó correctamente.

500: error interno.

Despliega un error debido a la falla interna del servidor.

2.12 Tarjetas de desarrollo

Para este trabajo tomamos en cuenta una tarjeta de desarrollo con características de una computadora funcional en una sola tarjeta de tamaño reducido, y que tiene todo lo que necesita sobre una placa. Se tomaron en cuenta Galileo y Raspberry Pi que son computadoras individuales de tamaño reducido, lo que significa que puede funcionar con el mínimo de recursos. Por ejemplo, en lugar de utilizar una computadora con todas las funciones de un servidor, se usará una tarjeta de desarrollo que cuenta con las características necesarias de una computadora de mayor tamaño, a continuación mostramos en la tabla 3 la comparativa de estas tarjetas. Hay muchas otras tarjetas de desarrollo disponibles en la actualidad, como el BeagleBoard, Edison, Minnowboard MAX, Wandboard.

Características	**Intel Galileo**	**Raspberry pi**
Dimensiones	123.8 mm x 72.0 mm	85.60mm x 56mm
Procesador	Intel Quark X1000 - solo núcleo	BCM2836 (ARMv7) - Quad Core
Almacenamiento externo	Tarjeta Micro-SD (expandible hasta 32 GB).	Tarjeta Micro-SD (expandible hasta 32 GB).
Velocidad	400MHz	900MHz
E / S analógicas	6 entradas analógicas con una resolución (programable) de 10 ó 12 bits A0 - A5 se puede	26 entradas de propósito general (acceso a I2C, UART y SPI).

	utilizar como E / S digitales utilizando funciones y operando a 3.3V o 5V, 14 E / S digital que se puede utilizar como entrada o salida y operan a 3.3V o 5V	

Tabla 3. Comparación de Intel Galileo y raspberry-pi.

Para este trabajo se elijio la tarjeta Intel Galileo, por que a pesar de no ser mejor en algunos aspectos con respecto a la raspberry- pi, la tarjeta que se observa en la figura 12, la elegi debido a que con tan solo 2Gb de memoria externa puede correr un sistema operativo de linux a diferencia de la raspberry pi que nesecita como mínimo 8Gb de memoriaa y lo que se busca es tener un sistema que ocupe un mínimo e reccursos .

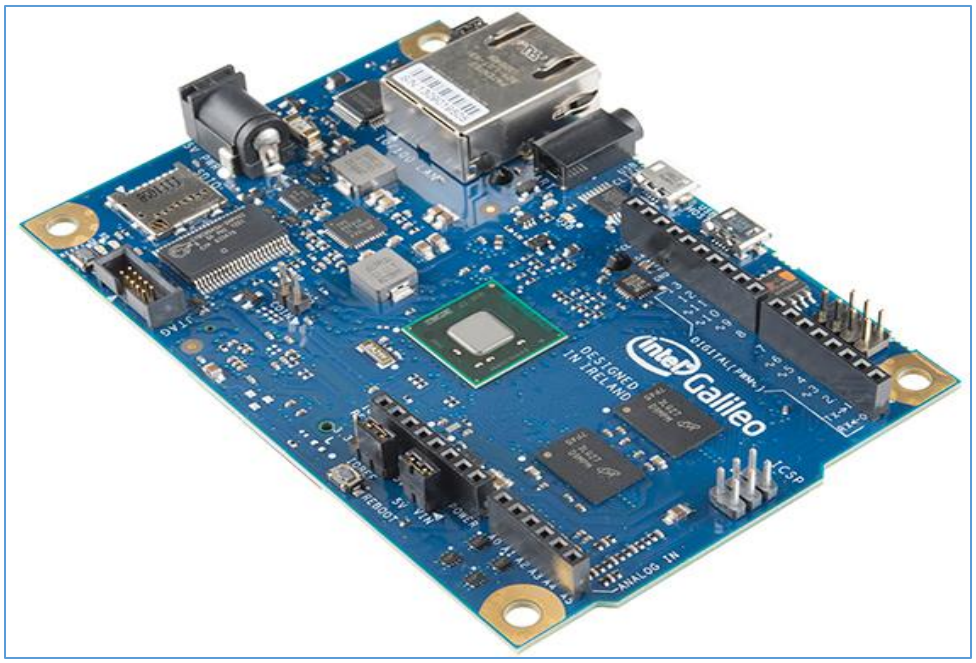

Figura 12. Tarjeta Intel Galileo. (Jimb0, 2014)

Capítulo 3. Implementación y diseño del sistema de teleoperación

Después de realizar la investigación del funcionamiento de cada elemento que integrará el sistema, se procedio a realizar el desarrollo e implementación del mismo. En este capítulo se presentan las características de los elementos usados, el desarrollo del sistema y de cómo se uso la parte física para implementar el sotfware y las razones por las cuáles se usaron.

3.1 Diseño del sistema de teleoperación

Como parte del proyecto se diseñó un sistema para la teleoperación de un robot móvil. El sistema cuenta con un servidor de comandos el cual está embebido en el robot, usando la *tarjeta de desarrollo Intel Galileo Gen 1* que proporcionará una conexión inalámbrica, que permite trabajar en una distribución de Linux embebido (Yocto) y esto hace que trabaje aplicaciones orientadas al Internet de las cosas. Por otra parte el sistema usa una cámara IP con conexión inalámbrica para la transmisión de video, el cual debe mostrarse en una interfaz web en tiempo real. Para los controles de navegación se aprovecha la biblioteca GPIO (General Purpose Input/Output) que permite la comunicación con los puertos de entrada y salida de la tarjeta de desarrollo Intel Galileo; para la comunicación con el microcontrolador MBED lpc 1768 y el microcontrolador se comunicará con los drivers pololu jrv12v28 por el puerto serie para mover al robot.

En la figura 13 se muestra un diagrama a bloques del diseño del sistema.

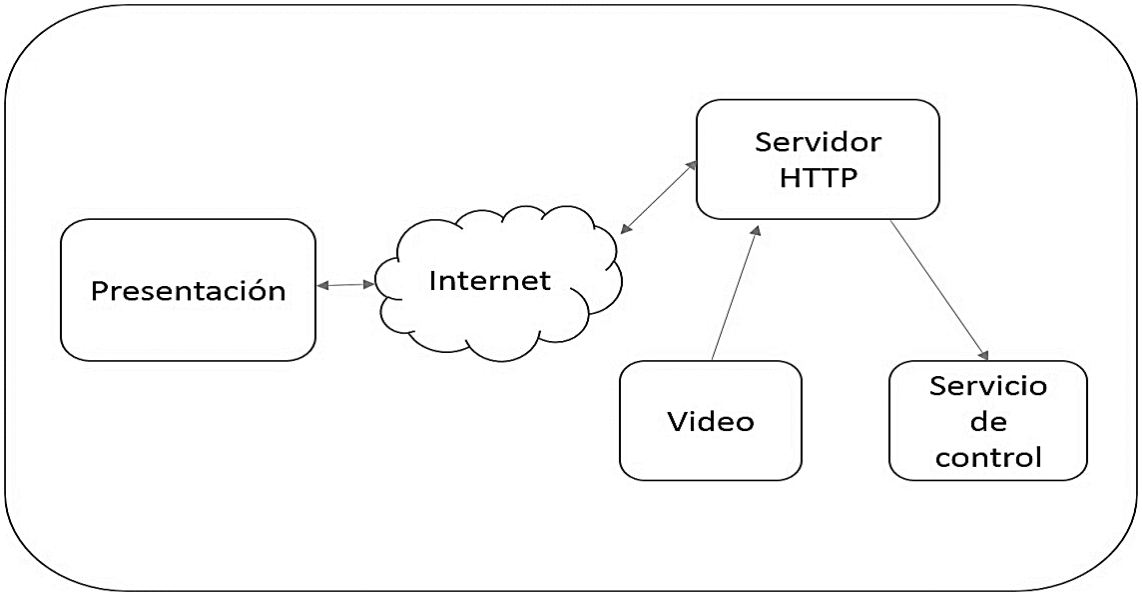

Figura 13. Diagrama a bloques del sistema

Las características de cada bloque de la figura 13, se explican a continuación:

- **Servidor HTTP:** Esta parte del diseño es fundamental, se podría decir que es la columna que sostendrá el funcionamiento correcto del robot. Con base en lenguaje elegido, creamos un servidor web, usando la arquitectura básica de los servidores. EL diseño del servidor se ve en la figura 14.

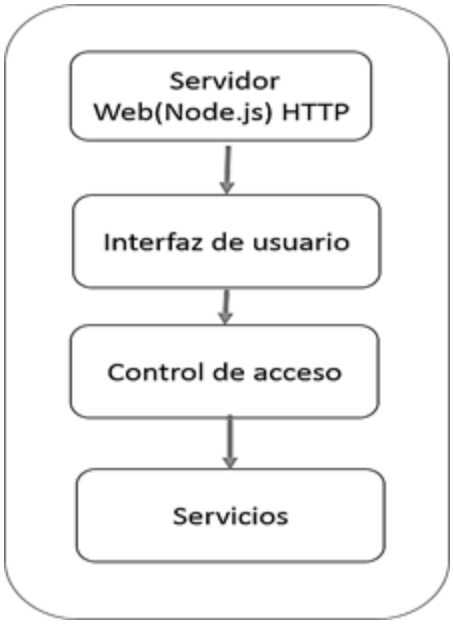

Figura 14. Arquitectura del servidor web.

- **Presentación:** Esta parte consiste en crear 3 páginas web dinámicas tanto para pc como para móviles, estas serán las encargadas de dar servicio al usuario. La primera página contendrá información general del proyecto, la segunda contendrá un sistema de autentificación para poder acceder a la página de control, esta última servirá para la comunicación del usuario con el robot en tiempo real, la interfaz contiene tanto el video de la cámara ip como los controles de navegación del robot, el diseño de las páginas se observan en la figura 15.

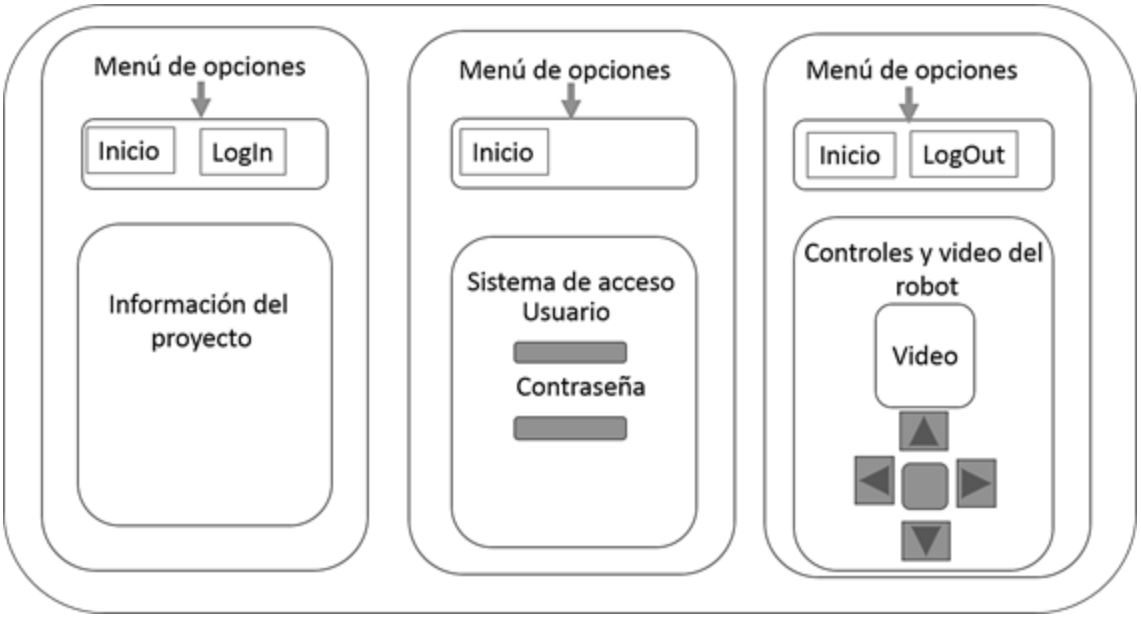

Figura 15. Diseño de las páginas web.

Para la página del sistema de autentificación se uso un algoritmo simple que valide el usuario y la contraseña, el algoritmo de validación se realiza como se observa en el diagrama de flujo de la figura 16, cabe mencionar que este algoritmo solo es temporal y para un solo usuario y en la sección 5.1 menciono los cambios que se porponen hacer a esté sistema.

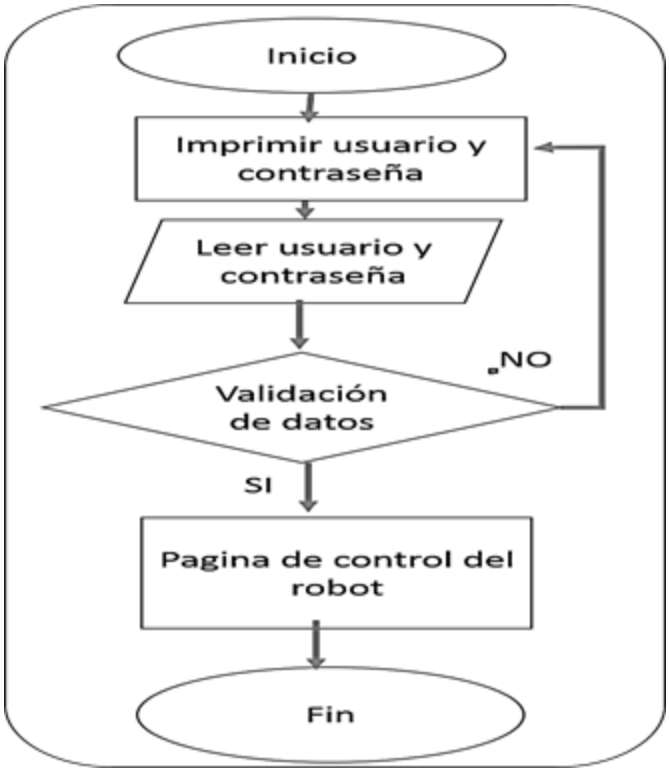

Figura 16. Diagrama de flujo de validación acceso

- **Video:** En esta parte se enfoco a obtener el video de la cámara IP e incrustarlo en la página de control, vease la figura 17. Un requisito indispensable es que el video sea en tiempo real.

Figura 17. Conexión lógica de la cámara IP.

- **Servicio de Control:** En este último bloque los comandos recibidos desde la interfaz gráfica (figura 18) deben ser retransmitidos a los controladores y drivers, los cuales se interconectan mediante transmisión digital y serial.

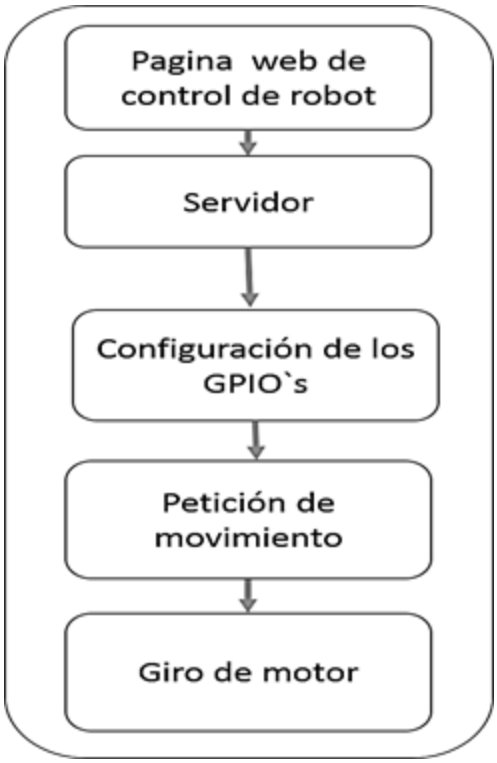

Figura 18. Digrama de conexión de control

3.2 Arquitectura del sistema

El desarrollo del sotfware del robot esta basado en dos arquitecturas cómo se menciono anterior mente, la primera es la arquitectura cliente-servidor y la segunda es la arquitectura AURA.

- Arquitectura cliente-servidor

La primera arquitectura es la cliente-servidor donde en la figura 18 se muestra como se aplico al sistema, la usamos por que es la más popular y usada para el desarrollo de aplicaciones web, ya que facilita la comunicación e intercambio de información a través de la red.

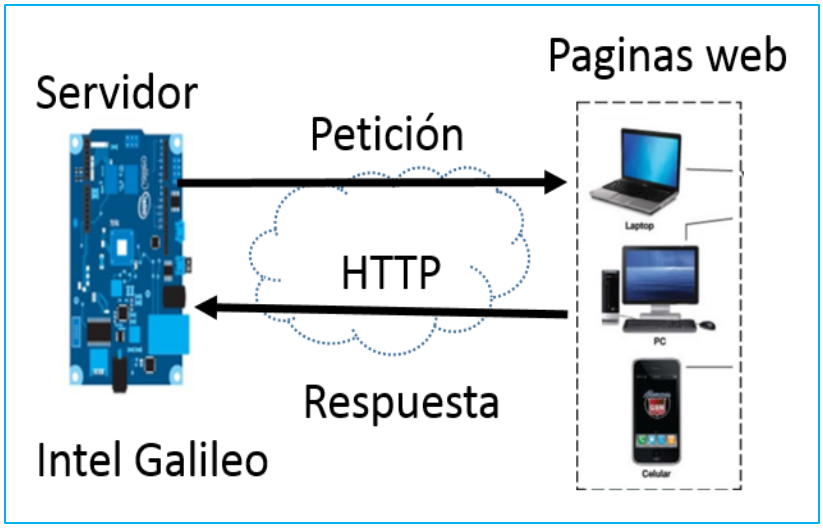

Figura 19. Arquitectura cliente-servidor.

- Arquitectura AURA

La segunda arquitectura que usamos es la AURA (Arquitectura para robots autónomos), el objetivo de usar está arquitectura es tener un sistema de control básico, con las caractrísticas mínimas que requiere el control de un robot autónomo. Como se vio en el capítulo 2, esta arquitectura se basa en 2 niveles básicos las cuales las aplicamos de la manera que se muestra en la figura 20 y con base en las necesidades del diseño ésta fue la arquitectura que más se adecuó al robot movil.

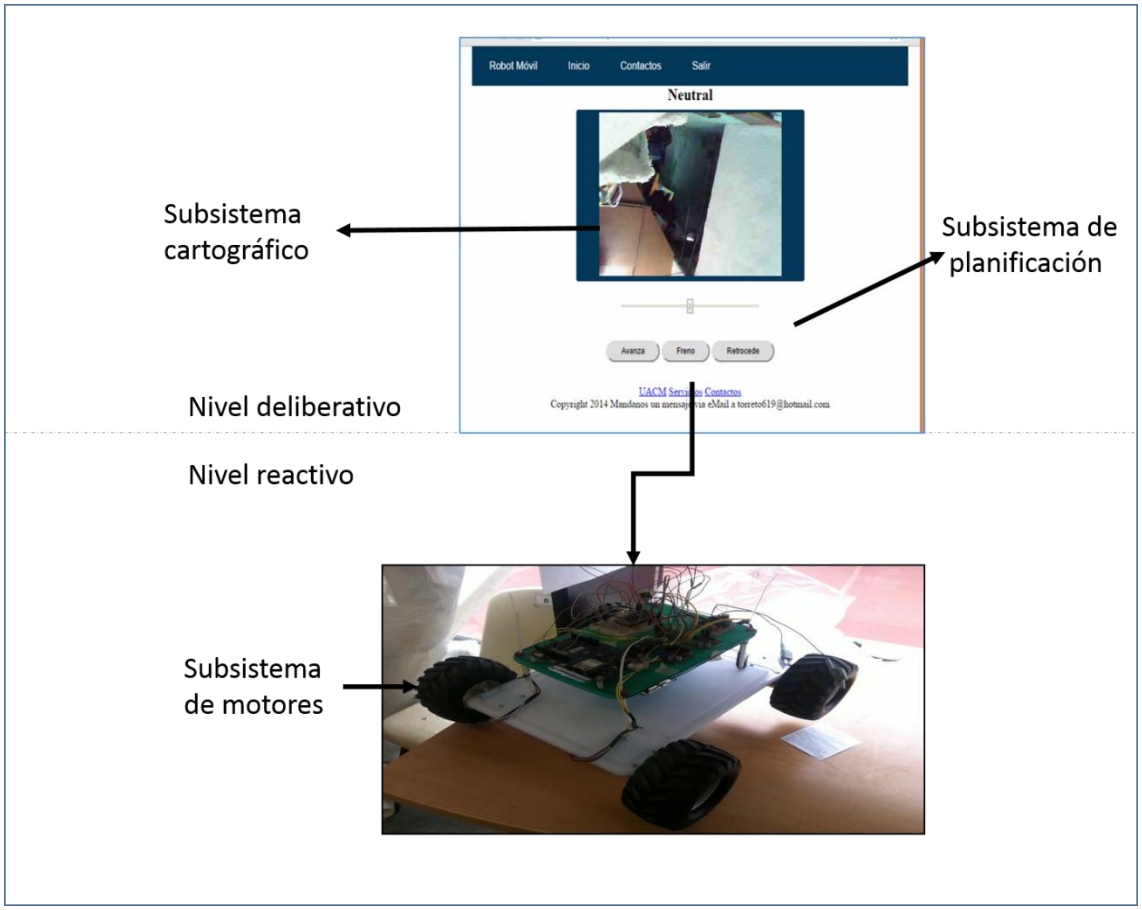

Figura 20. Distrubución de la arquitectura AURA en el sistema de teleoperación

Ahora se explicará como se aplicaron los niveles de dicha arquitectura y por qué.

Para el nivel deliberativo como vimos en capítulo 2 se subdivide en dos subsistemas. El primero es el subsistema es el cartográfico que como plantea esta arquitectura tiene que ver con la información que percibe visualmente y para cumplir este punto integramos una cámara IP para poder ver lo que pasa en el entorno del robot. El segundo subsistema es el de planificación, es la parte donde se llevó acabó las decisiones que va a enviar al nivel reactivo con base a la información del subsistema cartográfico, para esto hicimos un algoritmo que es capaz de indicar la dirección y sentido del robot.

Por último para el nivel reactivo lo representamos como lo especifica la arquitectura con motores que lo que hacen es ejecutar las acciones tomadas en el subsistema de planificación.

3.3 Implementación del sistema

En esta sección se van a mostrar los detalles de la implementación del diseño en base a los requerimientos del mismo.

3.3.1 Hardware

El hardware utilizado para la realización de este trabajo se muestra en la figura 21, a continuación se describen sus características y cómo son empleados dentro del sistema.

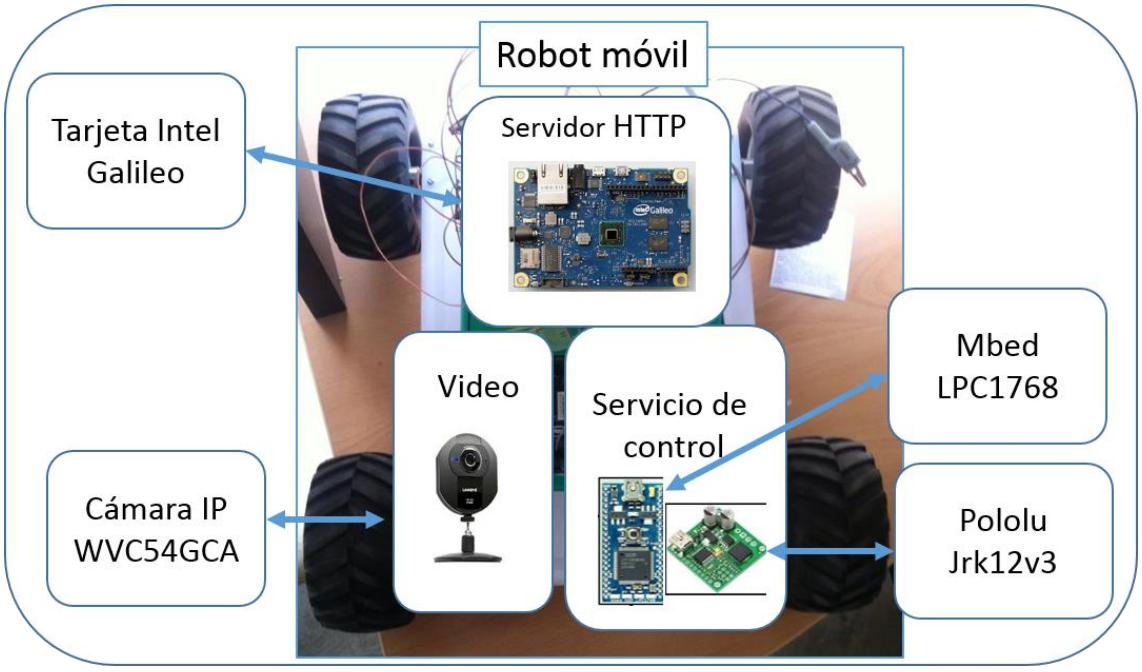

Figura 21. Hardware del robot móvil

3.3.1.1 Integración de la tarjeta Intel Galileo al robot

La tarjeta Galileo se montó en el robot y está aloja al servidor web, en el cual se integran las páginas e imágenes, las cuál se mostrarán al usuario a través de cualquier navegador de Internet, la cual permitirá una conexión inalámbrica. El

diagrama de bloques de la conexión de la tarjeta con los otros elementos físicos se muestra en la figura 22.

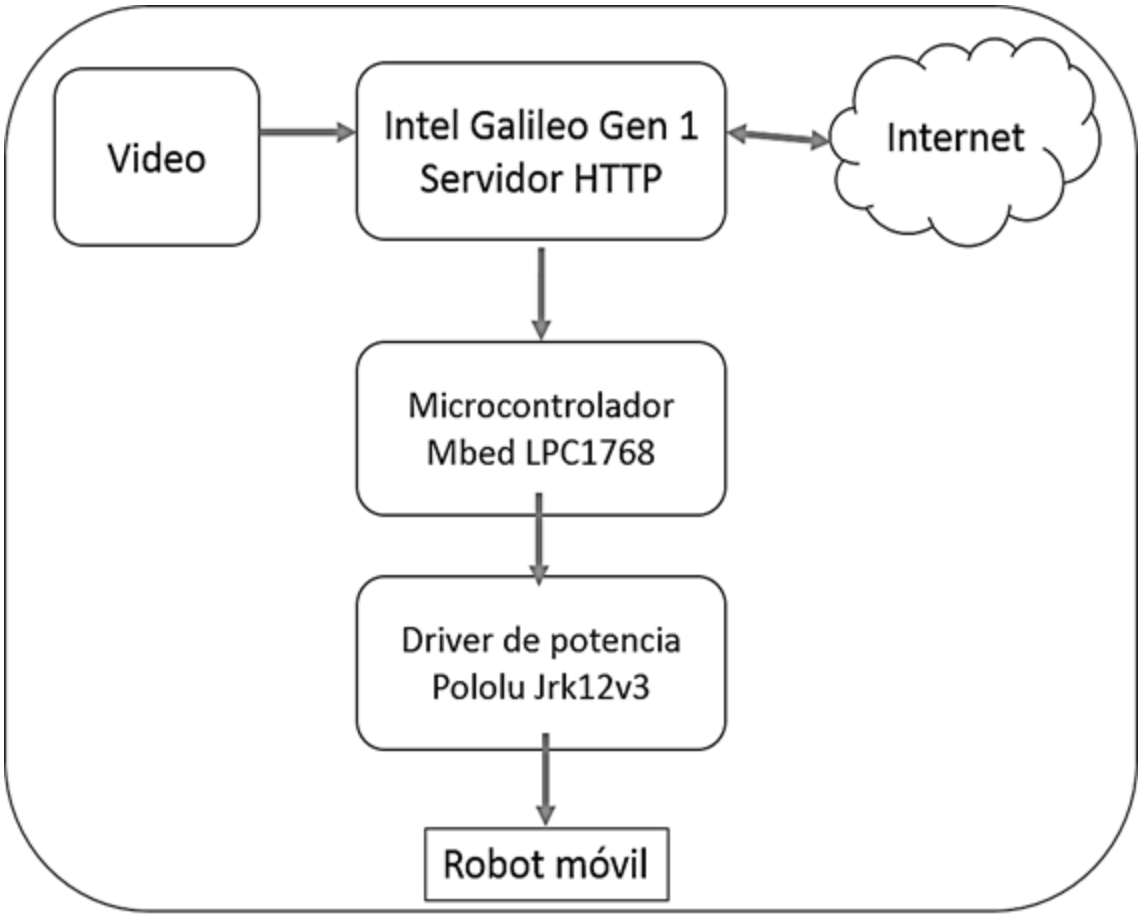

Figura 22. Diagrama de conexión de Galileo.

3.3.1.2 Cámara IP.

Se usa una cámara IP WVC54GCA, la cámara se conecta a la red LAN de forma inalámbrica con una dirección IP dinámica, la conexión de cómo se usa la cámara dentro del sistema se muestra en el diagrama de la figura 23, las especificaciones de la cámara se encuentran en el apendice B.I.

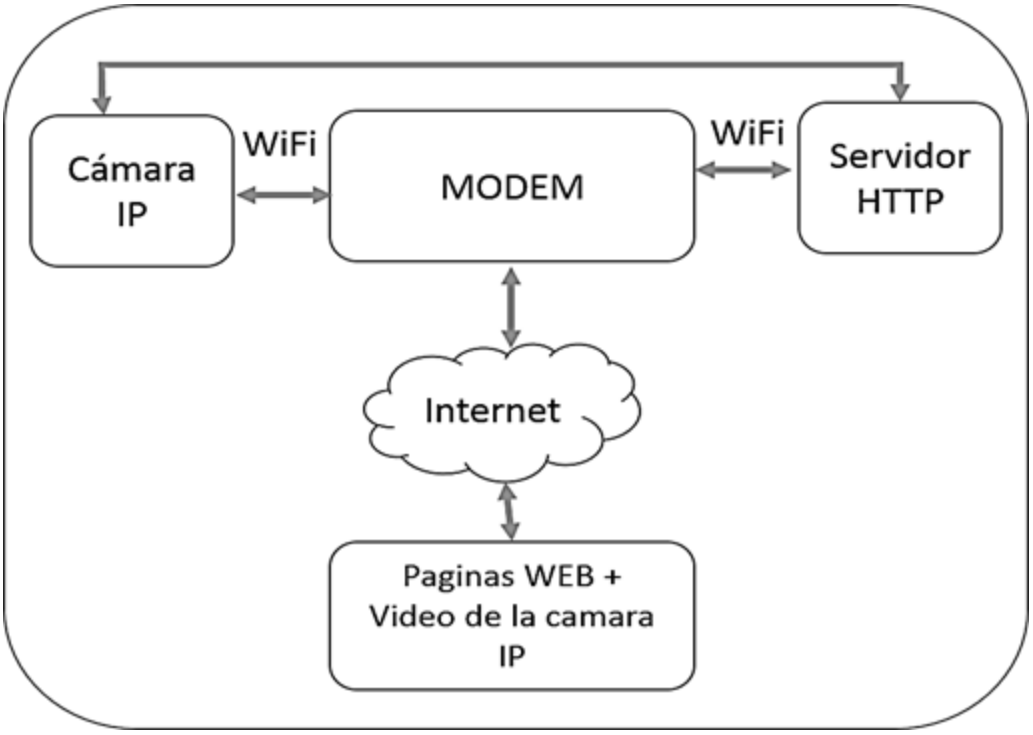

Figura 23 Diagrama de conexión de la cámara IP

3.2.1.3 Microcontrolador Mbed LPC1768

Ahora se muestra el diagrama de conexión del microcontrolador Mbed LPC1768 en la figura 24, el cuál contiene el código de control para el robot, las especificaciones del microcontrolador se encuentran en el apendice B.II.

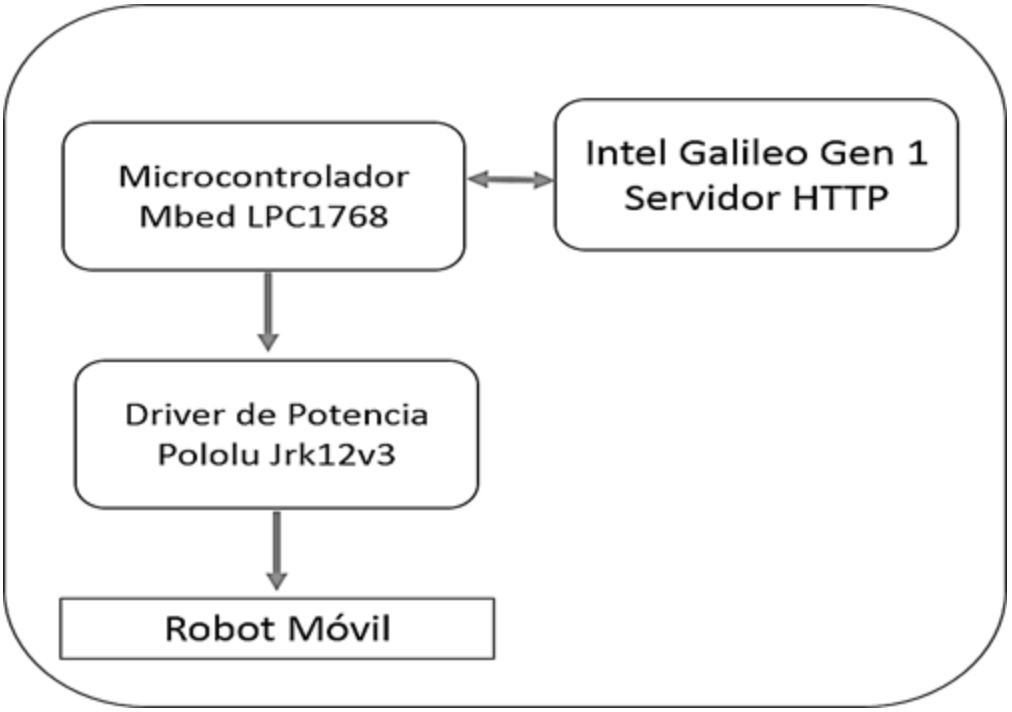

Figura 24. Diagrama de conexión Mbed LPC1768

3.2.1.3 Conexión del driver Pololu jrk12v3

Se ocupó para controlar los motores, que permite que la corriente fluya en sentido bidireccional, dando la posibilidad de cambiar el sentido de giro de los motores, mediante una conexión serial que permite la recepción y envío de información del sentido del giro y magnitud de velocidad de los motores de una manera muy simple, observece la figura 25, para ver el programa completo de conexión del microcontrolador se encuentran en el apendice B.III.

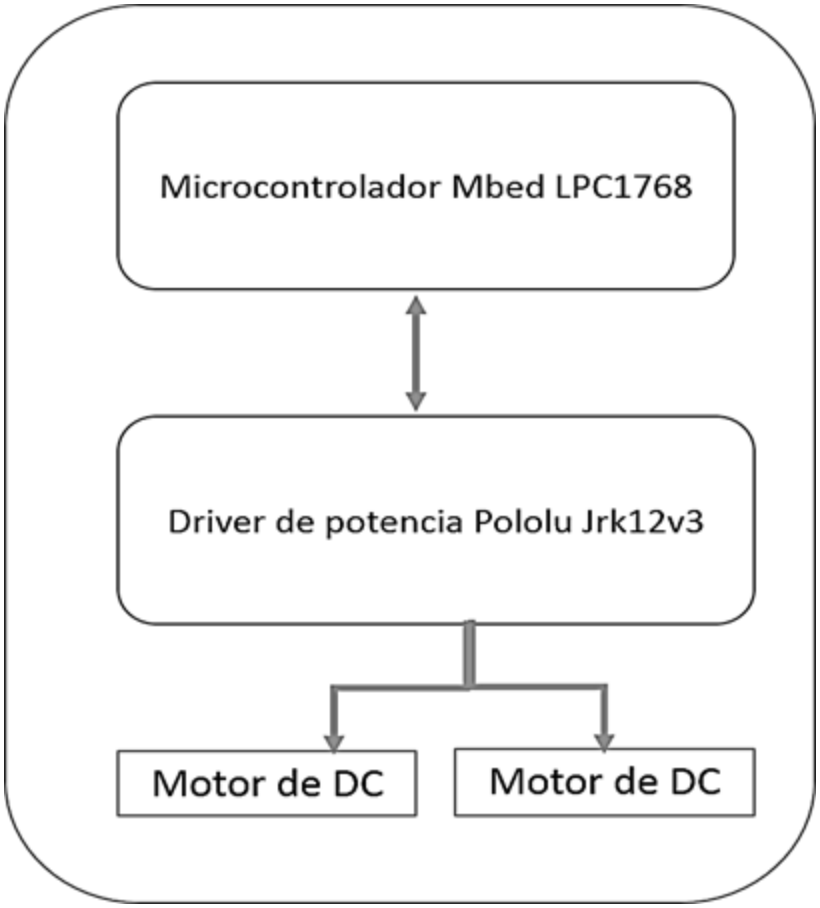

Figura 25. Conexión Pololu Jrk12v3

3.3.2 Robot móvil

El diseño del robot móvil en general y haciendo mención que solo se programaron y conectaron las tarjetas al robot. Se utilizó un diseño con cuatro ruedas tipo coche que proporciona una buena estabilidad, el diseño consta de cuatro motores DC de 12 volts, con cuatro drivers de potencia independientes para cada rueda y cada motor, es decir que podemos controlar cada motor de forma independiente, las velocidades de los motores varían y el robot girará incluso aún cuando se le haya ajustado inicialmente para que vaya recto. Esto quiere decir que la velocidad de cada motor debe ser controlada de modo tal que el robot se mueva en las direcciones o sentido deseado. La alimentación del robot usa una fuente regulada de 5 volts para la alimentación de las tarjetas y una bateria de 12 volts para la alimentación de los motores.

En la siguiente figura 26, se muestra la configuración del robot movil.

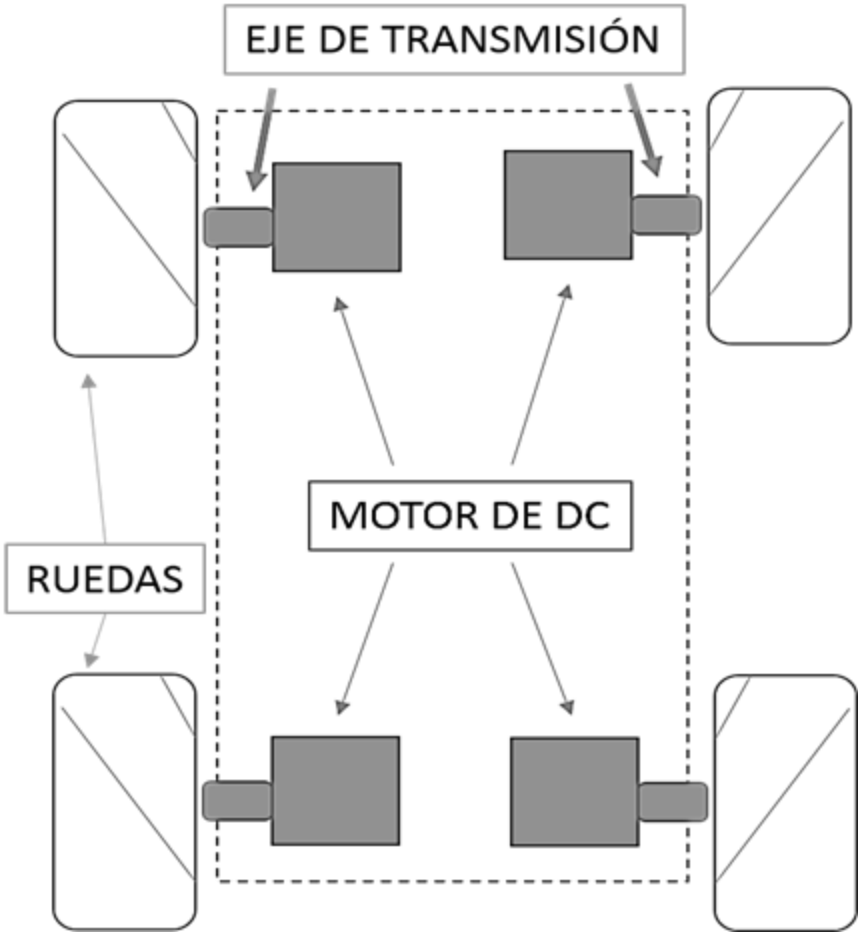

Figura 26. Diseño del robot móvil.

3.4 Programación del sistema

Para la programacion, se empleó un IDE (Entorno de Desarrollo Integrado) llamado Sublime text version 3.0, que es un editor de texto y editor de código, que maneja lenguajes como C++, Python, HTML 5, entre muchos otros, sin embargo no es software libre, puede obtenerse una licencia para su uso limitado y se huso con el fin de facilitar la programación ya que el editor de la tarjeta Galileo es muy limitado para editar el código. El entorno del software se muestra en la figura 27.

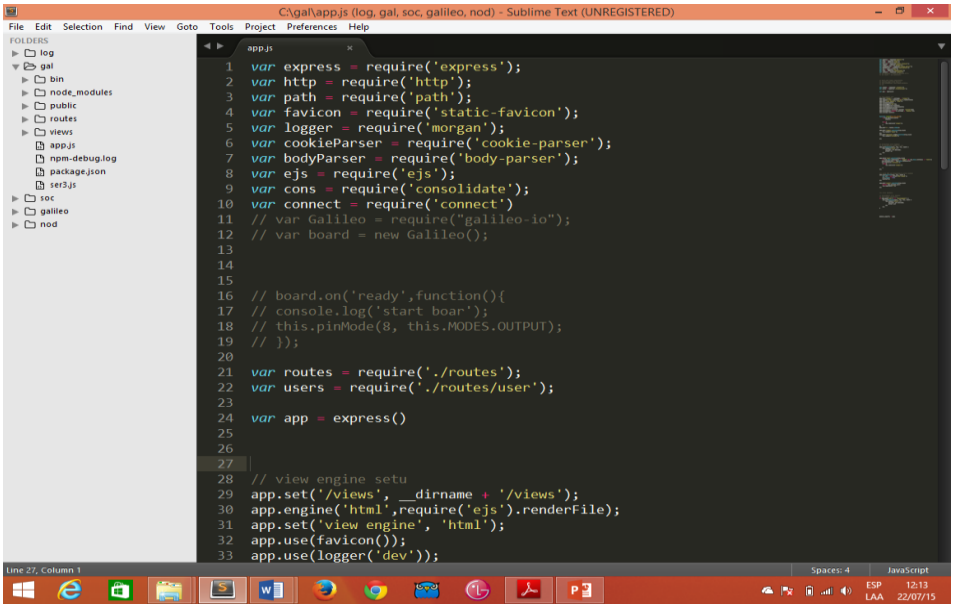

Figura 27. Ambiente de desarrollo sublime text 3.0

Se usaron los siguientes, lenguajes y tecnologías de programación: C/C++, HTML 5, CSS 3, Jquerry, Javascript, Node.js.

3.4.1 Programación del servidor HTTP

Para la implementción del servidor HTTP se uso un framework ya mencionado que está construida sobre el entorno de ejecución javascript de Chrome para construir fácilmente rápidas y escalables aplicaciones de red. Se uso Node.js porque, encaja perfecto en el sistema debido a lo ya comentado anteriormente y cumple con los requisitos del sistema para crear una aplicación(app) en tiempo real, flexible y mínima. Un sistema de tiempo real, es un sistema informático que interactúa repetidamente con su entorno físico y que responde a los estímulos que recibe del mismo dentro de un plazo de tiempo determinado (Monzon, 2013).

Node.js usa un modelo de E/S no bloqueante dirigido por eventos que lo hace ligero y eficiente para aplicaciones tiempo real. Tambien puede manejar decenas de conexiones simultáneas.

El servidor HTTP estará alojado en la tarjeta de desarrollo Intel Galileo Gen 1, la cual tiene una distribución de Linux embebida (Yocto). Para la estructura del

servidor se creó una carpeta principal llamada *proyecto* la cual contendrá todas las subcarpetas las cuales contendrán a su vez los códigos y algoritmos según la aplicación de la carpeta, la figura 28 muestra la estructura del servidor node.js.

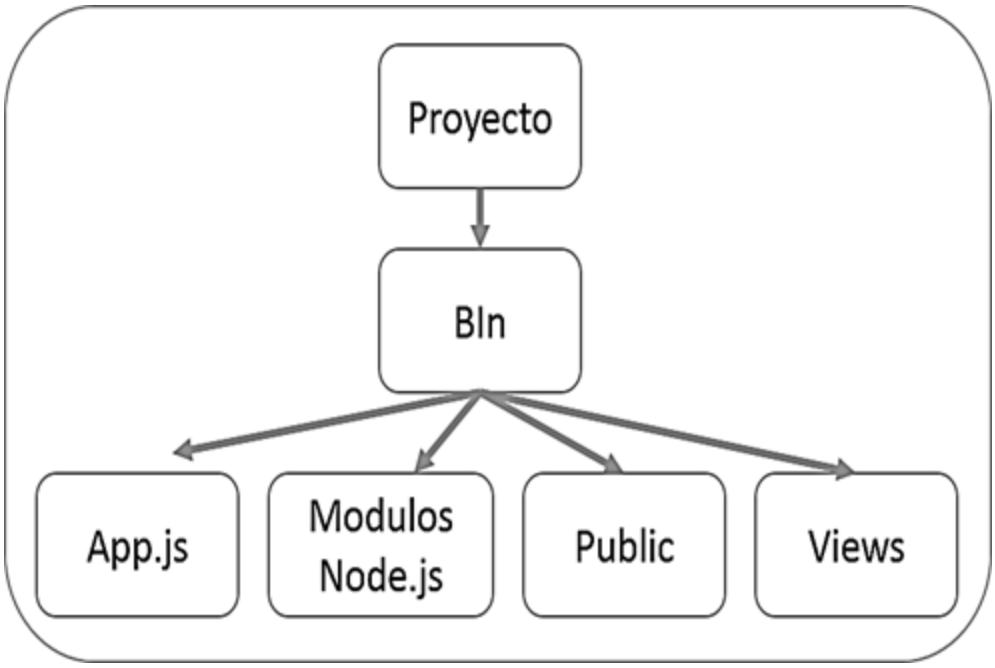

Figura 28. Estructura del servidor node.js

La estructura base del servidor se divide en dos partes para que en caso de que alguien irrumpiera en el sistema no le sea tan fácil modificar el contenido o acceder a los recursos del sistema. La primera parte y más importante es el código que se encuentra en la carpeta "*Bin",* el código completo se encuentra en el apendice A.I.

El siguiente listado explica para que sirve cada línea de código usado para la configuración del servidor.

1. Se exporta la configuración general del sistema.
2. Declarar el protocolo de comunicación.
3. Indicar el puerto por donde saldrá nuestra app.
4. Indicar los parámetros que va usar el servidor antes de iniciar.

```
1. var app = require ('../app');
2. var http = require('http');
3. app.set('port', process.env.PORT || 3000);
4. var server = http.listen(app.get('port'), function()
```

El código anterior muestra los parámetros necesarios para ejecutar el servidor, ahora detallaremos los parámetros básicos de la segunda parte del servidor que es el script *App.js*.

```
1. var app = express();
2. app.set('/views', __dirname + '/views');
3. app.engine('html',require('ejs').renderFile);
4. app.set('view engine', 'html');
5. app.use(express.static(__dirname + '/public'));
6. app.get('/', routes.index);
7.module.exports = app;
```

Es aquí en donde se configura todo lo necesario para atender las peticiones para el acceso a los recursos, a continuación se explican las líneas de código.

1. Se importa la biblioteca express.
2. Configuración de acceso a los recursos de la carpeta *views.*
3. Indicar el tipo de lenguaje de las páginas web.
4. Se incluye la biblioteca HTML al sistema.
5. Configuración de acceso a los recursos de la carpeta *public.*
6. Se crea la página raíz la cual se va a mostrar primero y por último la línea 7 indica que todos los parámetros del programa van hacer usados por la aplicación raíz.

Con la implementación y configuración del servidor, ya se tiene el servidor listo para trabajar y para acceder se ingresa la siguente URL (localizador de recursos uniforme) *http://IP_del servidor:puerto/.*

3.4.2 Páginas Web.

Una vez implementado el servidor, se crearon las tres páginas web en base al diseño hecho en el apartado 3.1, las páginas se guardaron en la carpeta *Public*. La primera página (figura 29) se llama index.html con el contenido que se planteó en el diseño. Todas las páginas tienen una barra de menu Inicio, Contactos, Entrar a exepción de la página privada.html que contiene una opcion extra que es salir que sirve para eliminar la sesión del usuario.

Usando CSS 3 y Jquery se hicieron páginas web adaptables, para cuando las páginas sean vistas desde dispositivos móviles con resolución de menos o igual de *800 pixeles* tenga una forma más estética y distribuir mejor los elementos de la página en una pantalla pequeña a diferencia de cuando se observa de una pantalla mayor a 800 pixeles donde se tiene más espacio para visualizar contenido, los códigos para adaptar las páginas a estos dispositivos se encuentra en el apendice A.III.

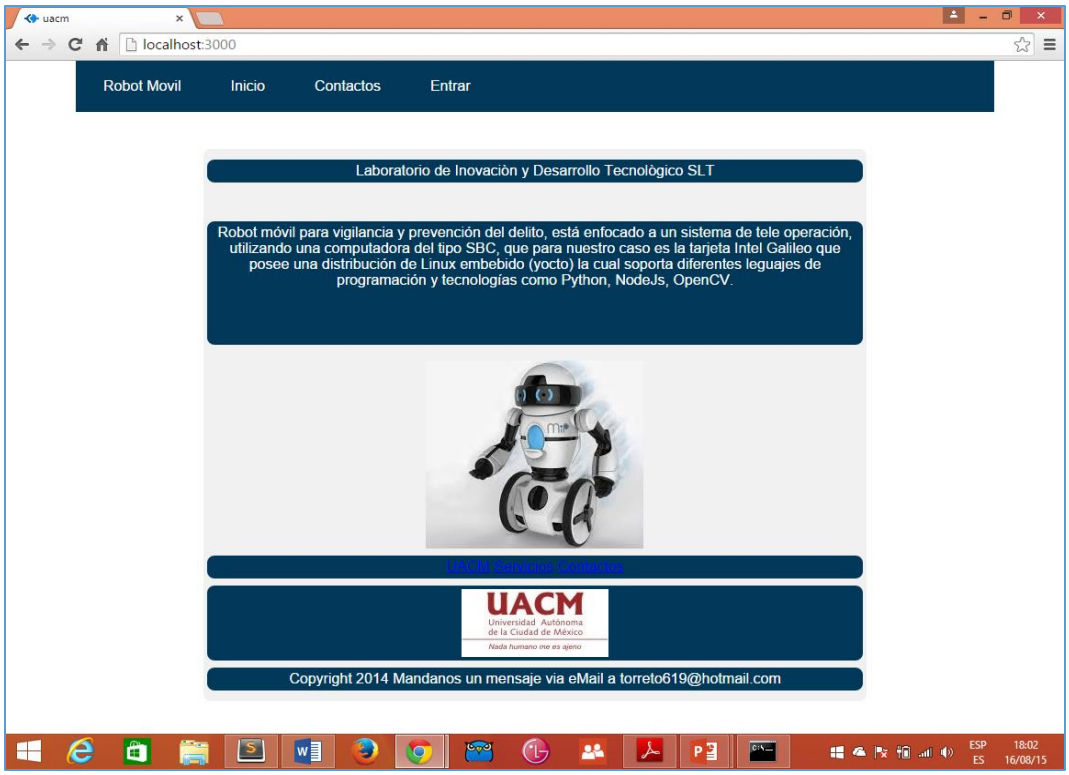

Figura 29. Página principal.

La segunda página la se llamo login.html (figura 30), aquí se implementó el sistema de autentificación.

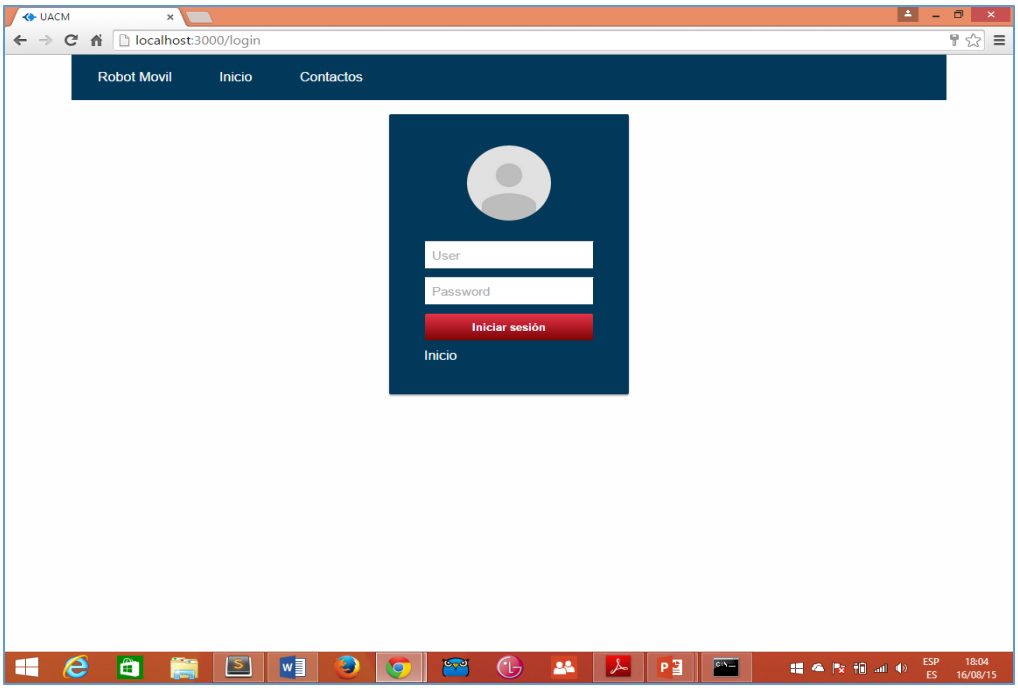

Figura 30. Página de acceso al sistema

Para el sistema de autentificación, se implementó el algoritmo del diagrama de flujo de la figura 31 que se encuentra en el apartado 3.1 de teleoperación, primero del lado del cliente (en la página login.html) se solicita el usario y contraseña después se guardó y se envío la información del formulario usando un método post e indicando a la función que va específicamente para su validación, el código es el siguiente:

1 . <form novalidate action='/aut' method='post'>

<label class="hidden-label" for="nombre">User</label>

2. <input id="txtUsuario" name="txtUsuario" type="text" placeholder="User">

<label class="hidden-label" for="Passwd">Contraseña</label>

3. <input id="txtClave" name="txtClave" type="password" placeholder="Password"class="">

4. <input id="signIn" name="signIn" class="rc-button rc-button-submit" type="submit" value="Iniciar sesi&#243n">

Para el código explicare las líneas más importantes.

1. Se declara la forma en que se envia la infromación y a que función va dirigida la información.
2. Se pone uná cuadro de texto para ingresar el usuario y se le asigna un id.
3. Se pone uná cuadro de texto para ingresar el contraseñas y se le asigna un id.
4. Se agrega la instrucción para qué cuando se presione el bóton enviar se mande la información contenida en los cuadros de texto usuario y contraseña.

Despúes de la recuperación de infromación se hizo un algoritmo de validación que se ve en la figura 31 y se implementó con el código siguiente:

```
app.post('/aut',function(req,res){
if(req.body.txtUsuario == 'Admin' && req.body.txtClave == '1234'){
req.session.user = req.body.txtUsuario;
res.redirect('/privada');
}else{
res.redirect('/login'); }
```

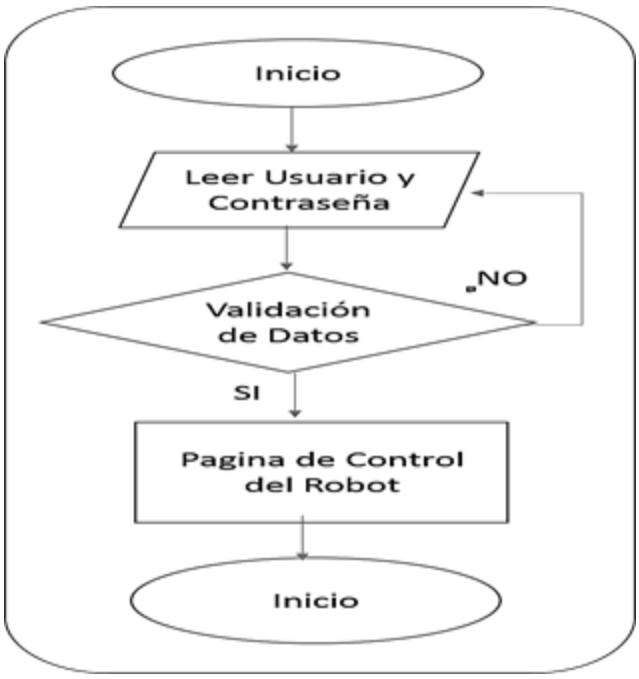

Figura 31 .Algoritmo de validación de usuario y contraseña

Una vez terminada la implementacion del código de la página *login.html,* se creó la última página *privada.html* (figura 32).

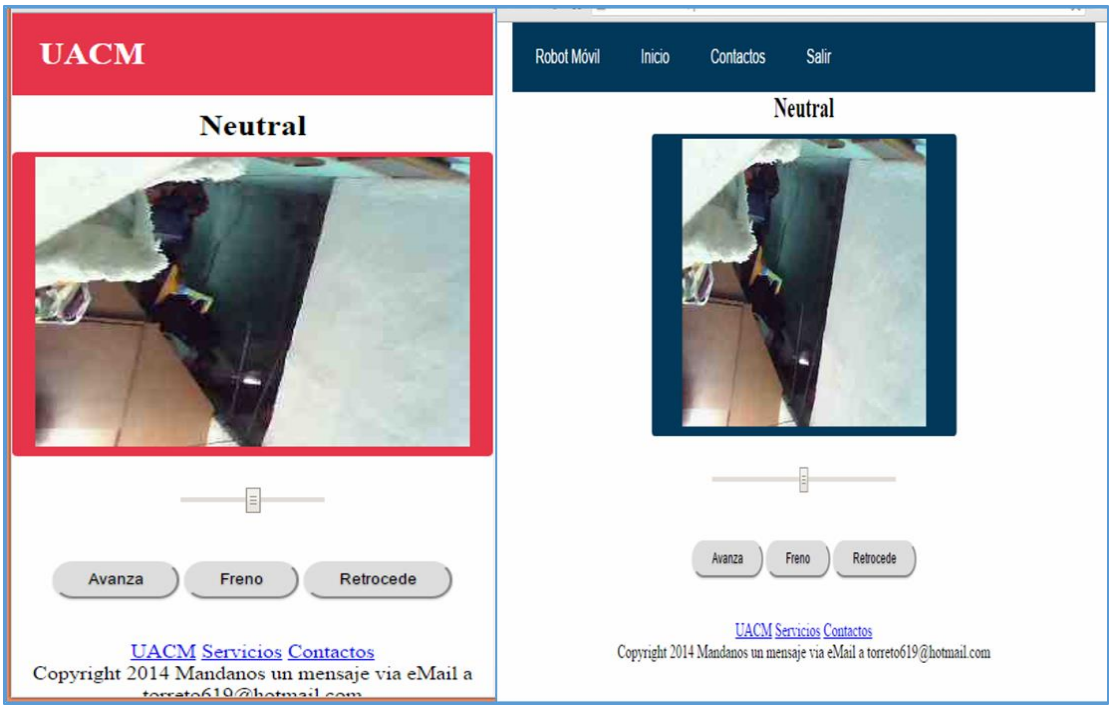

Figura 32. Página de control

A continuación se explicará la programación necesaria para que los botones de control del robot responda en tiempo real, para lo cual se uso un socket en el cliente y en el servidor, ocupando *Socket.io*.

Este socket maneja la conexión cliente-servidor y ofrece varios métodos, pero sólo se usaran dos en específicos para trabajar, estos son el método *on* (escucha eventos) y *emit* (emite eventos), primero se hizo la conexión del socket, para ello se creó un mensaje que indica si la conexión fue exitosa o no. El código del socket del lado del cliente (pagina privada.html):

```
<script src="/socket.io/socket.io.js"></script>
<script>
1.  var socket = io();
</script>
```

En la línea 1, lo único que se hace es indicar la inicialización del socket.

Para hacer la conexión del socket del lado del servidor, el código es el siguiente:

```
1. var http = require('http').Server(app);
2. var io = require('socket.io')(http);
3.  io.on('connection', function(socket){
```

Ahora en el servidor se indica los párametros que haran la conexión con el socket, en las siguientes líneas se explica la configuración del socket del lado del servidor.

1. Se carga la configuración que tiene el script app al servidor.
2. Al socket se le indica que debe escuchar el servidor mediante el protocolo HTTP.
3. Se inicia la conexión del socket.

Una vez hecha la conexión lógica cliente-servidor ahora se indica cómo programar un sólo botón de control, está lógica de conexión es la misma para los demás botones. Se utilizó JQuery para capturar el evento click del botón, asignando un valor al identificador al botón, lo cuál permitira tener un orden de cada botón, una vez hecho esto al presionar el botón se emitirá un evento el cual enviará la infromación al servidor y ésta es recibida en el servidor y una vez recibida se ejecutará una instrucción dependiendo el valor del identificador, el código del botón *Avanza* del lado del cliente es el siguiente, el código completo se encuentra en el apendice A.IV.

```
<button type="button" id="turn-left"> Avanza</button>
$('#turn-left').click(function() {
socket.emit('robot command', { command: 'turn-left' });
$('h2').replaceWith('<h2>Avanza hacia Adelante!');  });
```

Ahora se muestra el código del evento con el cual el servidor recibe las peticiones del cliente mediante el método (on).

```
socket.on('robot command', function (data) {
console.log(data);
var command = data.command;
if (command == 'turn-left') {
turnOffLed();
console.log('led');
```

Usando el código anterior se ejecutan funciones que permiten usar los puertos (GPIOs) de la tarjeta Galileo, las funciones para controlar los GPIOs se explicarán en la parte de control.

3.4.3 Video en tiempo real

Para la transmisión de video en tiempo real se uso un script que permite acceder a los recursos de la cámara IP de manera que la transmisión de video sea en tiempo real, o dicho de otra forma que las actualizaciones de las tramas de imágenes se realicen en milisegundos, de tal forma que el usuario no perciba la transición y note una imagen fluida. Se observa la parte importante del código que permite la extración del video de la cámara IP.

```
function doPlay(){
var ipadd = "http://Ip de la camara";
var dw_jpeg =   '<object classid="'+viewer_ocx_classid+'"
CODEBASE="'+viewer_ocx_codebase+'" id="'+viewer_ocx_id+'" width="640"
height="524">'+'<param name="Text" value="http://'+ipadd'/img/mjpeg.cgi'
'">' +
'<param name="BackColor" value="12632256">'+
'<param name="_Version" value="65536">'+
'<param name="_ExtentX" value="11774">'+
'<param name="_ExtentY" value="6562">'+
```

'</object>'; }

Y con la siguiente etiqueta se incrusta el script que permite ver el video en la página:

<script language="javascript" type="text/javascript">doPlay();</script>

Se visualiza el video en tiempo real, en la figura 33, se observa la imagen del video incrustado en la página web. El código completo de la cámara se encuentra en el apéndice A.II.

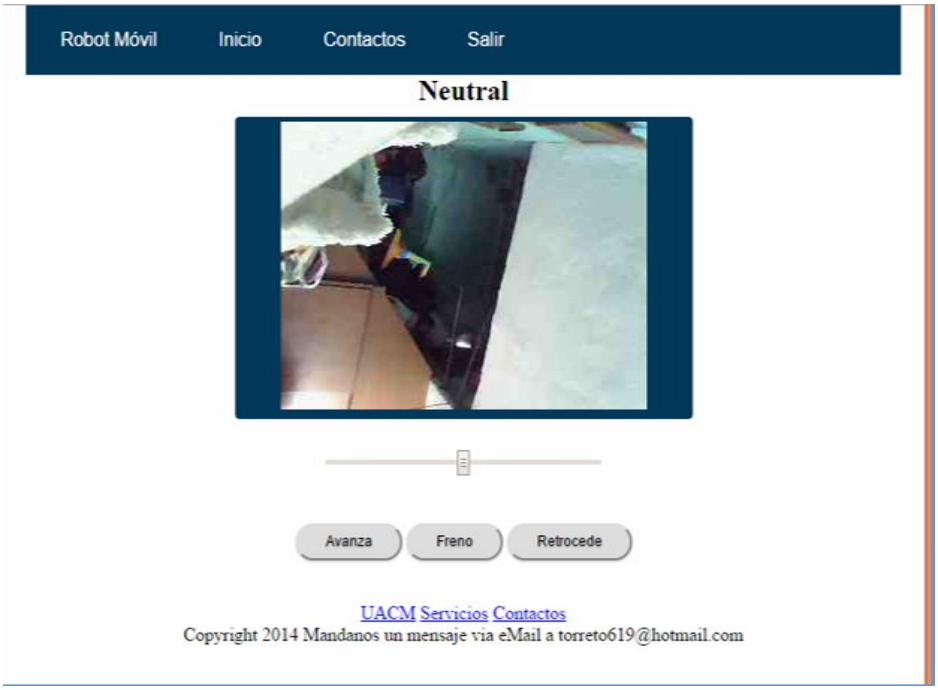

Figura 33.Transmisión de video

3.4.4 Sistema de control.

Para la implementación del control de robot se programaron las tarjetas Intel Galileo, Mbed LPC1768 y Pololu Jrk 12v3.

Primero para programar los puertos de salida de la tarjeta Intel Galileo, se usa una biblioteca de Node.js llamada *galileo-io*, que permite usar los periféricos de entrada y salida de una forma muy simple.

Para que los comandos desde la interfaz sean retransmitidos se usó el socket para relacionarlos con la función que asignan el pin de salida de la tarjeta, que hace que se manipula las salidas para que cambien de estado a alto o bajo según el requerimiento. La siguiente tabla muestra la configuración de las GPIOs en función de los controles del robot, (Tabla 1).

Robot Móvil	Pin 8	Pin 9	Pin 10	Pin 11
Avanza	1	0	0	0
Retrocede	0	1	0	0
Derecha	0	0	1	0
Izquierda	0	0	0	1
Detener	0	0	0	0

Tabla 4. Tabla lógica para el control de los pines

La comunicación entre Galileo y la Mbed Lpc1768 se realizó mediante los pines digitales que poseen ambas tarjetas, la relación de los pines para la comunicación de las tarjetas se muestra en la Tabla 5.

Pines de salida Intel Galileo Gen 1	Pines de entrada Mbed Lpc1768
Pin 8	**Pin 5**
Pin 9	**Pin 6**
Pin 10	**Pin 7**
Pin 11	**Pin 8**

Tabla 5. Conexión de los pines de Galileo y Mbed Lpc1768

Para la programación de la tarjeta Mbed Lpc1768 se utilizó C/C++, la tarjeta se programó de tal manera que se comunica con la tarjeta Galileo, que como se vio en la Tabla 1, usando cambios de estado (alto 5 volts, bajo 0 volts) y para la comunicación con el driver de potencia Pololu Jrk12v3 se usó un protocolo llamado *POLOLU* que utiliza una programación serial.

El código siguiente muestra los parámetros de configuración de Mbed Lpc1768, que se usaron tanto para la comunicación con la tarjeta Intel Galileo como para la Pololu Jrk1768.

```
Serial device (p28, p27); //tx, Rx
Digital IN myled(p6);
DigitalIn myled2(p5);
DigitalIn myled3(p7);
DigitalIn myled4(p8);
device.baud(9600);
```

Una vez realizada la configuración de los pines de entrada como los pines de transmisión serial (tx,rx) y la velocidad de transmisión a una velocidad de 9600 baudios, realizamos la programación de las acciones que debe realizar dependiendo del estado de los pines, el diagrama de flujo del algoritmo se observa en la figura 34 y el código se encuentra en el apéndice A.V.

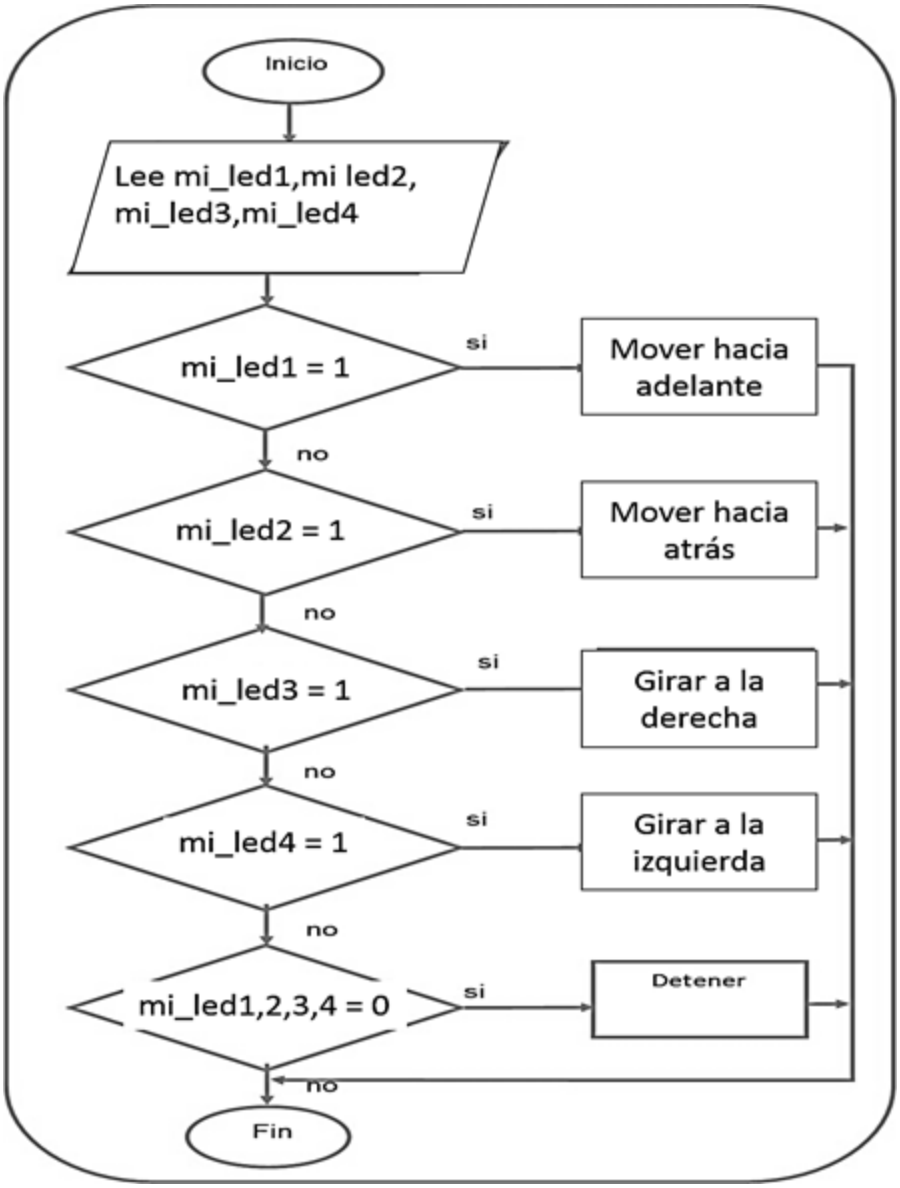

Figura 34. Diagrama de flujo de acciones de la Mbed

La comunicación entre la tarjeta Mbed lpc 1768 y jrk21v3, se realizó mediante el pin9 (rx) y el pin10 (tx) de la tarjeta Mbed 1768 y se conectaron a los pines rx y tx de la Pololu jrk21v3.

Pero antes de conectar los 4 drivers Pololu se tienen que configurar en su interfaz nativa. Los parámetros a configurar son: la velocidad del puerto serial, el tiempo de salida y el número de dispositivo (ID) como se muestra en la figura 35.

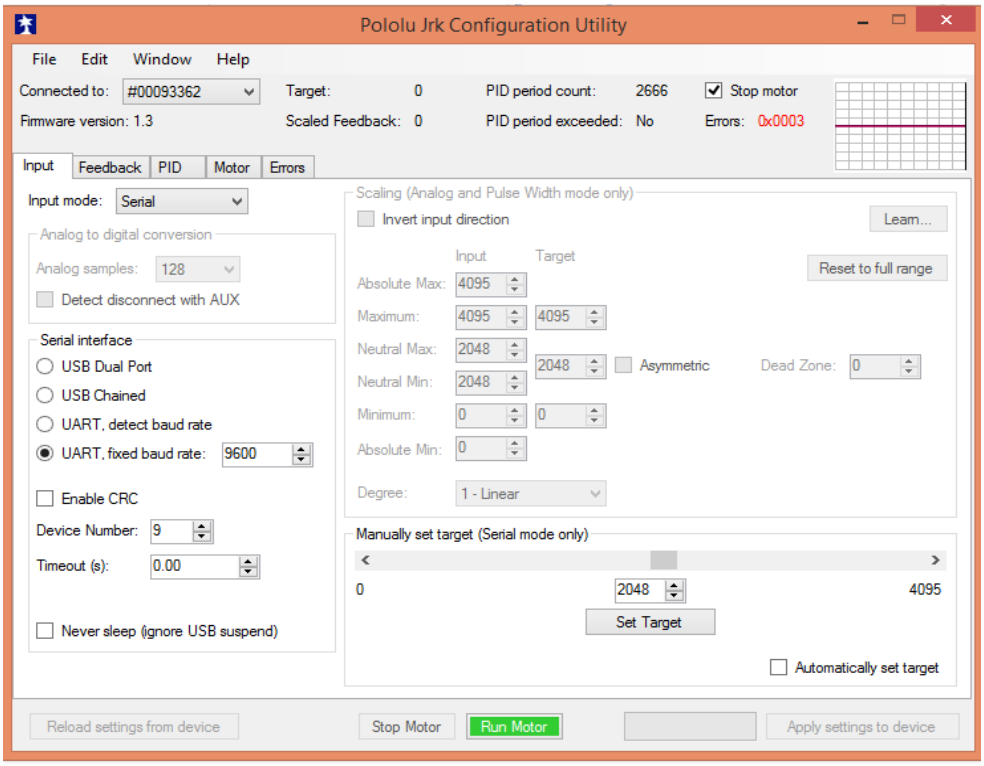

Figura 35.Software de configuración Pololu jrk

Para la comunicación de la Mbed con la Pololu jrk se usó un protocolo propietario de Pololu que es un protocolo compatible con el protocolo serie utilizado por los controladores. Se conectan las 4 pololu jrk en serie, usando este protocolo, enviamos comandos específicamente al driver jrk deseado sin confundir los drivers, ya que como mencionamos en el párrafo anterior cada driver tendrá un número de dispositivo distinto.

Para empezar la comunicación la Mbed transmite el comando 0xAA (170 en decimal), seguido del comando con el número de dispositivo, después el comando que indique el sentido del giro y por último la velocidad deseada (0-127 en decimal).

El siguiente fragmento de código ejemplifica la conexión entre estos dos dispositivos.

device.putc(0xAA); // Byte de arranque

device.putc(0x0B); // Número de Dispositivo

device.putc(0x61); // Sentido del Giro

device.putc(0x7F);//Magnitud de velocidad

Sobre cómo manipular mejor la pololu jrk recomiendo la lectura de su guía de manejo (POLOLU, s.f.), que contiene información a detalle de otras características de este driver.

También para mayor información de la biblioteca *galileo-io* revisar el repositorio de esta API (rwaldron/galileo-io, s.f.), y para la tarjeta Mbed (ARMMBED, s.f.).

Y por último el código completo de todo el sistema se encuentra en el apéndice A.

Capítulo 4. Resultados

En este capítulo se presentan los resultados de las pruebas hechas al sistema. Como resultado final se tiene un robot móvil controlado desde internet usando comunicación inalámbrica, tres páginas web dinámicas y un sistema de control para la navegación del robot.

4.1 Resultados

Se realizaron varias pruebas con el Robot Móvil (figura 36 y 37), con base a cada punto planteado en el diseño del sistema, para comprobar el funcionamiento del sistema.

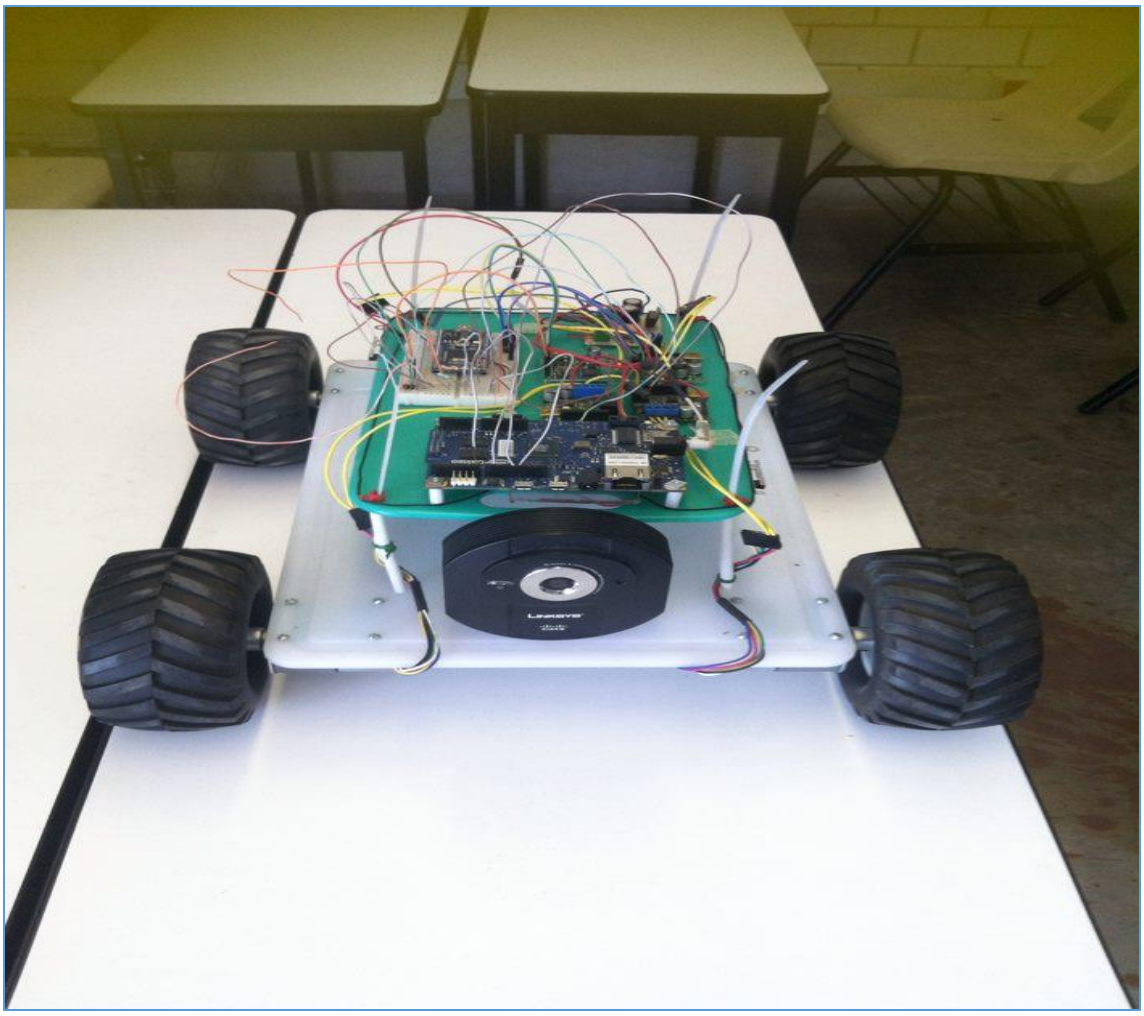

Figura 36. Robot móvil.

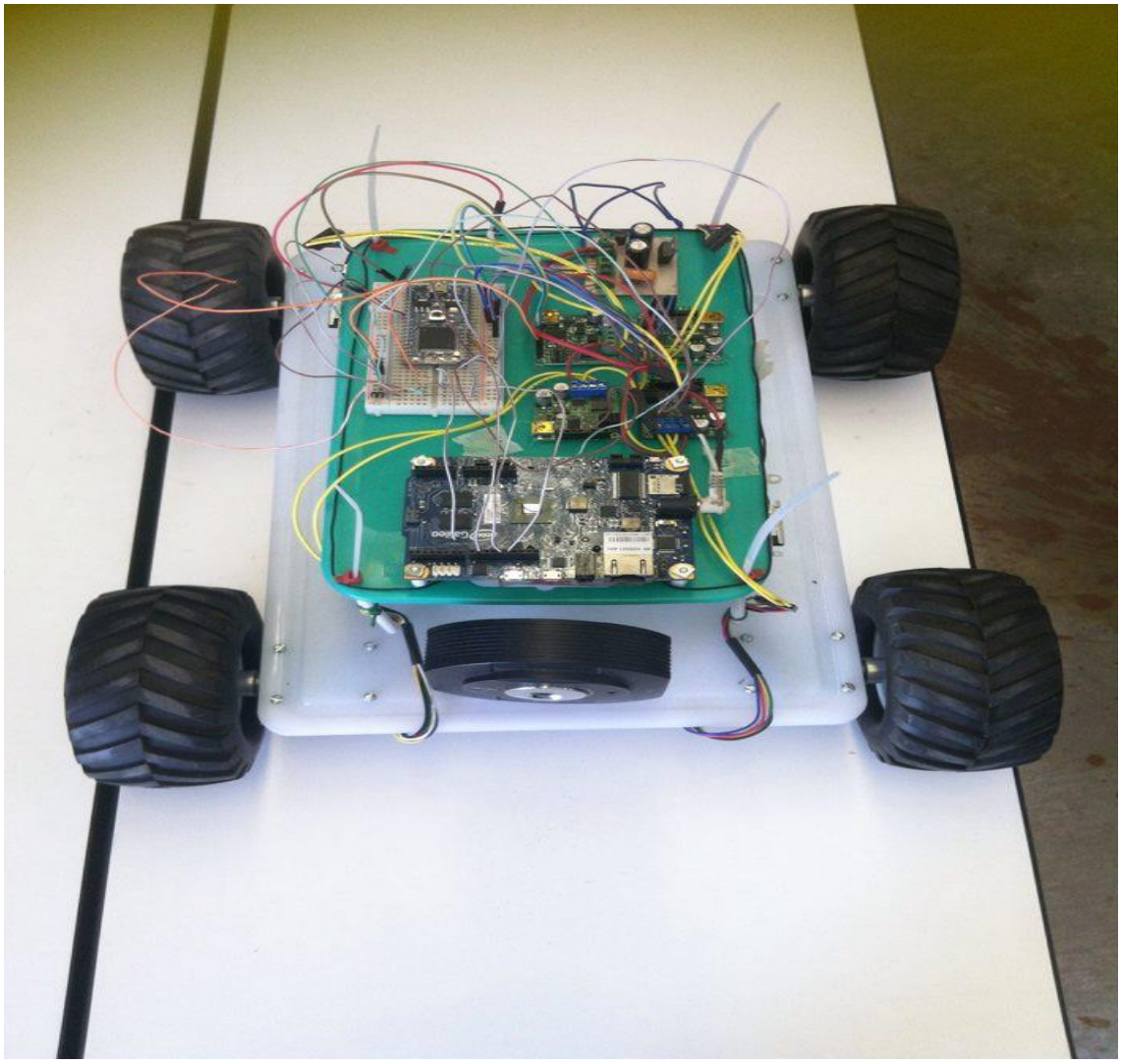

Figura 37. Robot móvil

Las primeras pruebas se realizaron sobre el servidor embebido implementado en la Galileo. Para ello se inició el servidor con el comando ***npm start***, después se conectó al servidor desde una PC para hacer peticiones al servidor. En la figura 38 se muestran los códigos de estado del servidor que indican el resultado de las peticiones, por ejemplo tenemos un código "200" que indica la respuesta a la petición del recurso ha sido correcta, en algunos casos el tamaño del archivo en kilobytes y el tiempo que tarda en cargar el archivo, también tenemos otro código "304" Indica que la petición a la URL no ha sido modificada desde que fue requerida por última vez y su funcionamiento es correcto.

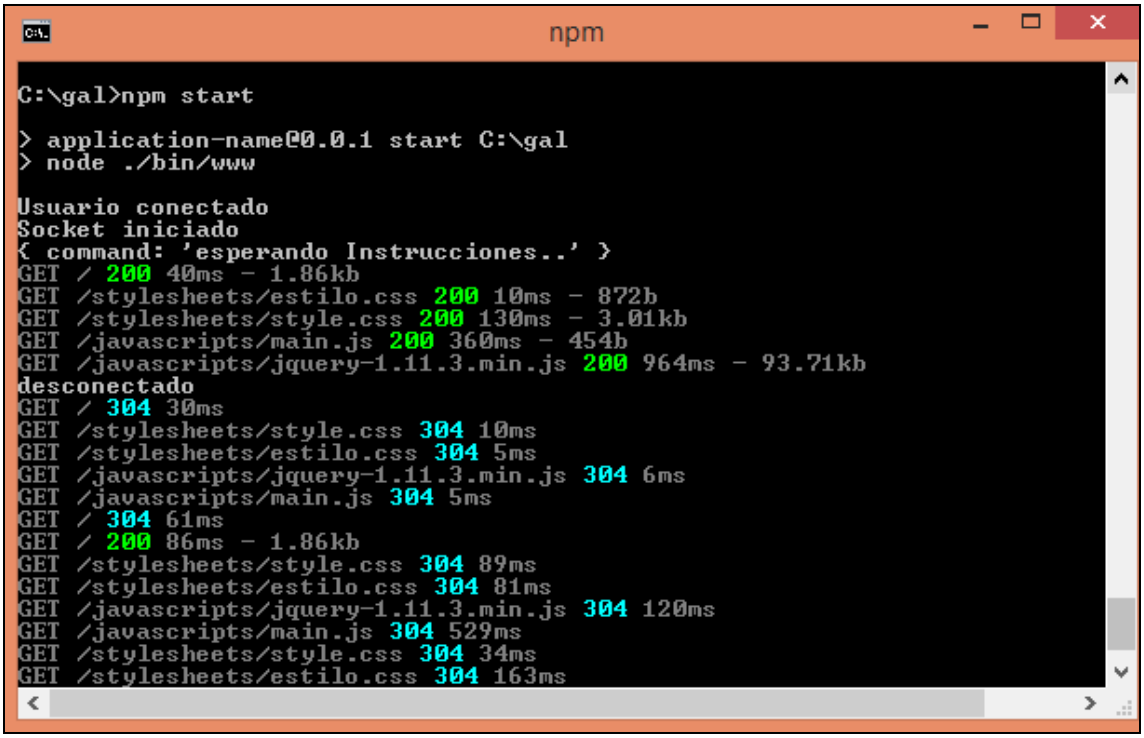

Figura 38. Resultado de los códigos del estado del servidor.

Se realizaron conexiones desde una mini laptop Acer de 10 pulgadas y un iPhone 4 de 3 pulgadas, en la tabla 6 puede observarse el tipo de petición que se ejecuta, el tiempo que tarda el servidor en responder los recursos solicitados, el tiempo estimado para una mala conexión es de 2.5 segundos que depende del contenido de la página y el tipo de recursos requeridos. La tabla dice que el servidor está contestando las peticiones en un rango de tiempo aceptable es decir en milisegundos y esta prueba se hizo en un horario donde las conexiones a la WLAN son en un rango medio.

Tipo de petición	**Recurso Solicitado**	**Tiempo de respuesta Laptop**	**Tiempo de respuesta iPhone**
GET	Página (index.html)	61ms	89ms
GET	CSS	24ms	81ms
GET	Scripts	46ms	529ms
GET	Página (login.html)	42ms	40ms
GET	CSS	22ms	71ms
POST	Autentificación	20ms	45

GET	Página (privada.html)	76ms	82ms
GET	CSS	22ms	56ms
GET	Scripts	204ms	42ms

Tabla 6. Resultado de las peticiones hechas al servidor

Ahora se muestran los resultados finales de cómo quedó la implementación de las interfaces gráficas, en la primera página (index.html) se muestran los resultados con respecto a las pruebas hechas a la compatibilidad con pc, teléfonos celulares, tabletas y otros dispositivos móviles, ya que gran porcentaje del tráfico en la red se hace desde dispositivos móviles. En la figura 39, mostramos el resultado de la página principal para dispositivos mayores a 800 pixeles de resolución.

Figura 39. Resultado final de la interfaz gráfica (index.html) para dispositivos con resolución mayor a 800px.

En la figura 40, vemos el resultado obtenido de la versión para dispositivos menores a 800px de resolución.

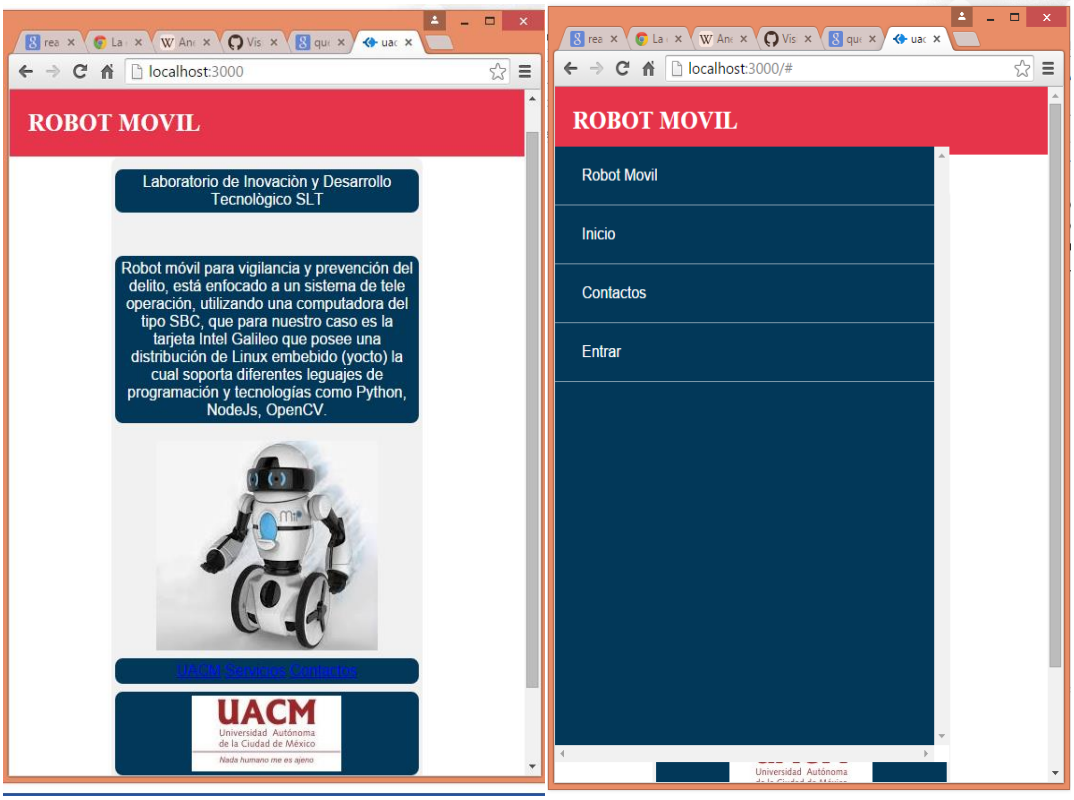

Figura 40. Resultado final de la interfaz gráfica (index.html) para dispositivos menores a 800px

La segunda página (login.html) se muestra en la figura 41, las pruebas realizadas fueron con respecto a la autentificación de usuario para acceder a la página de control, para ello se realizaron dos pruebas.

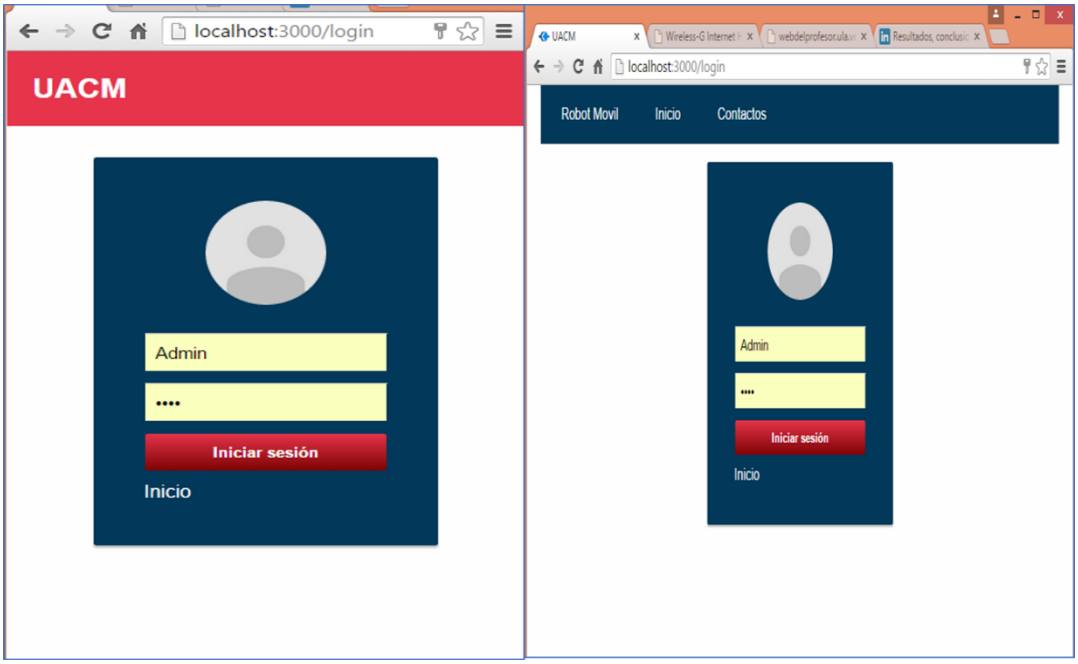

Figura 41. Página de autentificación

1. La primera fue ingresar *usuarios* y *contraseñas* diferentes, para probar que él sistema de autentificación funcionara correctamente a la hora de enviar y comparar el usuario y contraseña.

2. Le segunda prueba fue tratar de acceder directamente a la página escribiendo la URL de la página.
   http://IP:3000/privada.html

   El resultado fue el redireccionamiento hacia la página Login.html y denegarnos la entrada hasta ingresar el *usuario* y *contraseña* correctos, lo cual muestra que la única manera de acceso a la página de control es mediante la autentificación del usuario correcto.

Por último se muestran los resultados de la página de control (figura 42), que es la que permite ver el *video*, la conexión del *socket* con el servidor y del servidor a los GPIO´s de la tarjeta Intel Galileo.

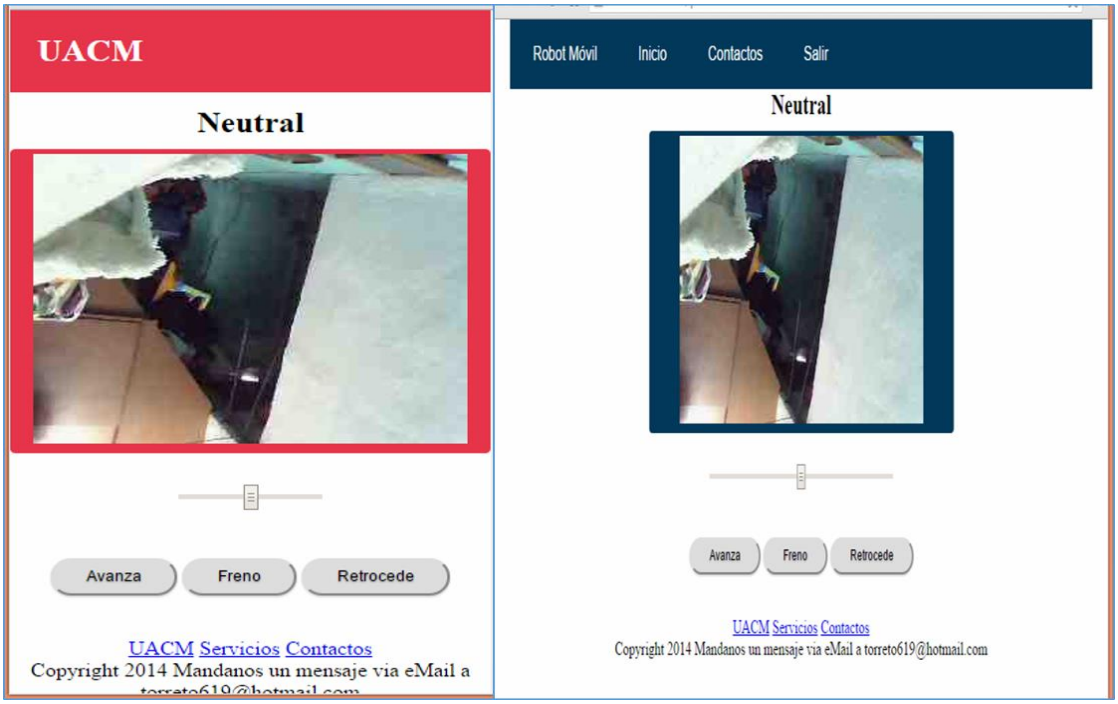

Figura 42. Resultado final de la página de control (privada.html), en la versión para móvil y la versión para Pc.

Los resultados de la visualización del video incrustado en la página web se muestran en la tabla 5. La transmisión video contenida en la página utiliza los parámetros y reproductores mencionados en el capítulo 2. Las pruebas para la transmisión de video se hicieron con los siguientes parámetros:

- Velocidad de transmisión: 256 Kbps
- Resolución: 320x240
- Número de tramas enviadas: 15fps
- Formato de codificación: MJPEG

Reproductor de Video	**Tiempo de Retardo de Transmisión**
Windows Media	10 seg
Real player	5 seg
Activex OCX	5 ms

Tabla 7. Resultados de la transmisión de video de la cámara IP

Los mejores resultados para la transición de video se obtuvieron usando JavaScript mediante el uso de la biblioteca OCX (OLE Control Extensión) que permitió la transmisión con el menor tiempo de pérdida y los parámetros de configuraciones son los ideales para mantener una buena imagen.

Las últimas pruebas realizadas fueron sobre la navegación del robot mediante la interfaz gráfica, dentro de una red LAN con un alcance aproximado de 20 metros fuera de la línea de vista del modem, para lo cual se procedió a dar un recorrido a lo largo del laboratorio y en el pasillo de la universidad y en la figura 43, se muestra un diagrama del recorrido hecho por el robot de manera satisfactoria.

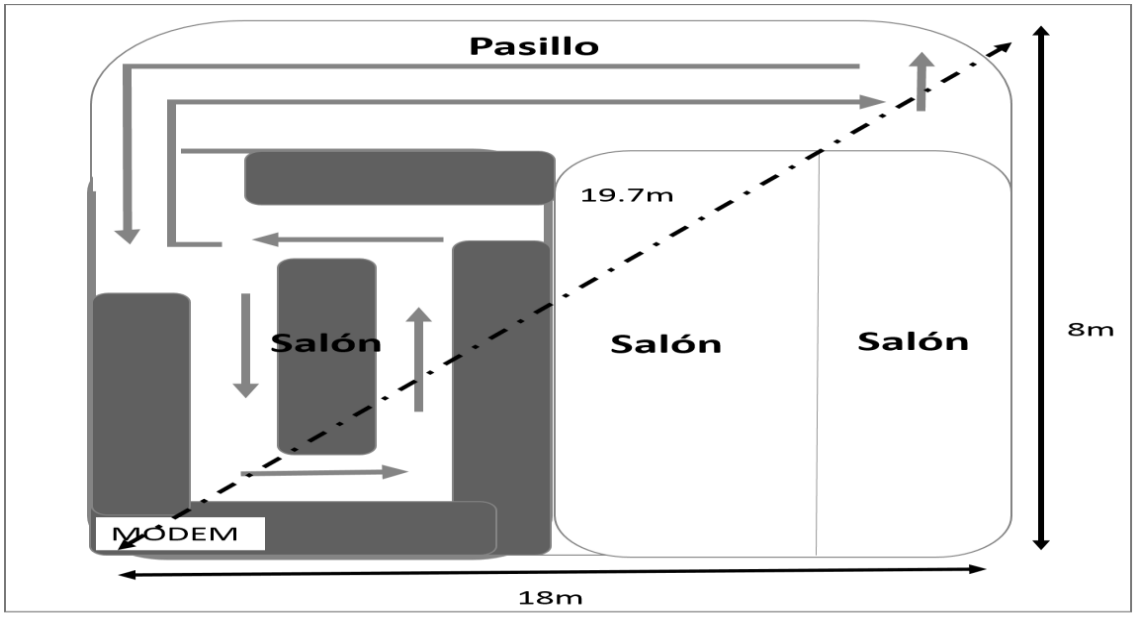

Figura 43. Trayectoria recorrida por el robot móvil.

Otro resultado importante de la página de control es que al usar protocolos de comunicación en tiempo real para el envío de los comandos de control del robot, el envío de comandos se ejecuta en segundo plano de una manera tan rápida que el usuario percibe que el cambio es de inmediato y la actualización de los comandos solo es en la sección que contiene los controles, lo que hace que no afecte la transición de video y que toda la interfaz fluya de manera que la pérdida de información sea la mínima, ya que aproximadamente a los 22 metros de distancia del modem se pierde la comunicación.

Capítulo 5. Conclusiones y trabajo a futuro

La realización de este proyecto cumplió con el objetivo planteado al inicio. En general se resolvió el problema de integrar el sistema en una microcomputadora Intel Galileo para usarla como servidor, usando nuevas tecnologías de software como Node.js, Jquery, HTML5, Css3 y hardware para el monitoreo y control del robot móvil en tiempo real y además la orientación del sistema hacia el internet de las cosas.

Cabe mencionar que para la implementación del servidor web se ocupo una alternativa diferente a servidores convencionales como Apache, IIS entre otros más, la cual ofrece las mismas opciones que un servidor como Apache, para alojar páginas HTML, scripts, y otros recursos. La alternativa utilizada es Node.js que es una API que tiene cinco años de haberse desarrollado, además de tener el servidor en el robot móvil y no depender de un servidor externo que guarde toda la programación. Se utilizó con la intención de hacer un trabajo diferente a trabajos similares a este, por otro lado el aporte de la investigación es como comunicar hardware y software y por se hace un aporte importante para quienes utilicen estos mismos elementos.

También el sistema puede implementarse en otras tarjetas con las características de una microcomputadora, lo único que hay que hacer es adaptar las GPIO´s, ya que en las microcomputadoras sólo varían las direcciones para acceder a los puertos y en cuanto al sistema operativo puede implementarse en Windows, Mac y cualquier distribución de Linux.

Se logró una sinergia entre todos los elementos que permitió la transmisión de datos de la interfaz gráfica al servidor y de ahí a las GPIO´s de forma fluida para después comunicarse con el microcontrolador MbedLPC1768 y de ahí a los driver de potencia de manera serial.

Además en este proyecto se usaron las arquitecturas cliente-servidor y la AURA, para tener un panorama más completo y poder integrar los elementos conforme a las arquitecturas, desde el punto de vista de la robótica como de las comunicaciones, así fue más fácil ubicar y acomodar los elementos de la forma correcta y que todo el sistema cumpla con las características necesarias de un sistema de teleoperación.

En cuanto a la comunicación de los comandos de control del robot se usaron protocolos de comunicación en tiempo real ya que las nuevas tendencias de comunicación para aplicaciones web usan cada vez más estos protocolos para hacer más rápida la interacción entre el cliente y el servidor.

Acerca del robot móvil que se usó para este proyecto puede variar, el robot puede tener forma de triciclo, diferencial, araña, etc., ya que todo el sistema de teleoperación se puede adaptar e implantar en otro robot con características similares con mínimas modificaciones para que funcione de forma adecuada.

5.1 Trabajo a futuro.

Como trabajo a futuro se puede agregar una forma de navegación manual y una forma de navegación automática del robot.

Para lograr una navegación automática se puede agregar más sensores que permitan tener una visión más amplia del panorama de navegación del robot, crear algoritmos de razonamiento lógico dependiendo de la información obtenida, para tomar decisiones con base en la información y asignar tareas específicas, además ampliar la información a desplegar en las páginas web, como el estado

de la carga de la batería o la falla de algún sensor y agregar más usuarios con diferentes privilegios y cambiar de un servidor HTTP a un servidor HTTPS.

Apéndice A.

En este apartado se muestra todo el código del sistema por secciones según las partes del sistema.

A.I. Código del servidor HTTP

```
var express = require('express');
var http = require('http');
var path = require('path');
var favicon = require('static-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var ejs = require('ejs');
var cons = require('consolidate');
var connect = require('connect')
var routes = require('./routes');
var users = require('./routes/user');
var app = express();
app.set('/views', __dirname + '/views');
app.engine('html',require('ejs').renderFile);
app.set('view engine', 'html');
app.use(favicon());
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded());
app.use(cookieParser());
app.use(express.session({ secret: 'secret'}));
app.use(express.static(__dirname + '/public'));
app.use(app.router);
function login(req,res,next){
    if(req.session.user){
        next();
    }else{
        res.redirect('/login');
    }
```

```
};
app.get('/', routes.index);

app.get('/login',function(req,res){
    res.render('login');
});
app.get('/privada',login,function(req,res){
    res.render('privada');

});

app.post('/aut',function(req,res){
    if(req.body.txtUsuario  == 'Admin' && req.body.txtClave  == '1234'){
   req.session.user = req.body.txtUsuario;
   res.redirect('/privada');
    }else{
        res.redirect('/login'); }
});
app.get('/salir',function(req,res){
    delete req.session.user;
    res.redirect('/');
});
app.set('port', process.env.PORT || 3000);
var server = app.listen(app.get('port'), function() {
  debug('Express server listening on port ' + server.address().port);
});
```

A.II. Código para obtención del video

```
<script>
function init()
{ set_Video(); }
function set_Video()
{
    var cf = document.forms[0];
    var vs = document.getElementById("v_play");
```

```
    if (vmode != "")
    {
        vs.style.display = "block";
    }
    else
    {
        vs.style.display = "none";
    }
}
var vmode = "mpeg"; // "";    // mpeg, jpeg,
var ipadd = "http://IP DE CAMARA";     <!-- '172.31.2.125'   /   self.location.host -->
var brow;
if(isIE()) { var brow = "IE"; }
if(isNS()) { var brow = "MZ"; }
if (self.location.protocol != "http:")
{
    if(ipadd.indexOf(":")>0)
    {
        ipadd = ipadd.substring(0, ipadd.indexOf(":"));
    }
}
var dw_jpeg = '<object classid="'+viewer_ocx_classid+'"
CODEBASE="'+viewer_ocx_codebase+'" id="'+viewer_ocx_id+'" width="640" height="524">'+
        '<param name="Text" value="http://'+
        ipadd+
        '/img/mjpeg.cgi'+
        ' '+
        lpp+
        ' '+
        brow+
        ' '+
        lang_str+
        '">' +
        '<param name="BackColor" value="12632256">'+
        '<param name="_Version" value="65536">'+
```

```
                '<param name="_ExtentX"  value="11774">'+
                '<param name="_ExtentY"  value="6562">'+
              '</object>';
var dw_mpeg = '<object classid="'+viewer_ocx_classid+'" CODEBASE="'+viewer_ocx_codebase+'" id="'+viewer_ocx_id+'" width="640" height="524">'+
                '<param name="Text" value="http://'+
                ipadd+
                '/img/video.asf'+
                ' '+
                lpp+
                ' '+
                brow+
                ' '+
                lang_str+
                '">' +
                '<param name="BackColor" value="12632256">'+
                '<param name="_Version" value="65536">'+
                '<param name="_ExtentX" value="11774">'+
                '<param name="_ExtentY" value="6562">'+
              '</object>';
var dw_push = '<img border="0" alt="Video for other Platform/OS" src="http://192.168.2.113/img/video.mjpeg">';
function dw(message)
{ document.write(message); }
function doPlay()
{
   if(vmode != "")
   {
      if(isWin())
      {
         if(isIE())
         {
            if( vmode == "jpeg" )
            { dw(dw_jpeg); }
            else
            { dw(dw_mpeg); }
```

```
            else
            {
                dw(dw_push);
            }
        }
        else
        {
            dw(dw_push); }  }
    else return false;}
var is_auto_reload = true;
function reloadOnErr(obj) {
    if(!is_auto_reload)
        return;
    setTimeout("eval('obj.src = obj.src;'", 500);}
</script>
```

A.III. Código de las páginas web

a) Página principal (index.html)

```
<html>
<head>Pagina Principal</head>
<body>
<imgsrc="http://2.bp.blogspot.com/-
OsTIynMVwv0/TZ_x6urOSiI/AAAAAAAAALU/eco_JhuaaiM/s210/uacm%2Blogo%2B4.png" />
Laboratorio de Inovaciòn y Desarrollo Tecnològico SLT
<hr>
<ul id="nav">
      <li><a href="/login">Login</a></li>
      <li><a href="#">Contact</a></li>
      <li><a href="/">Inicio</a></li>
   </ul>
   </table>
   <br>
   <br>
```

Robot móvil para vigilancia y prevención del delito, está enfocado a un sistema de tele operación, utilizando una computadora del tipo SBC, que para nuestro caso es la tarjeta Intel Galileo que posee una distribución de Linux embebido (yocto) la cual soporta diferentes leguajes de programación y tecnologías como Python, NodeJs, OpenCV.

<img src="http://t2.gstatic.com/images?q=tbn:ANd9GcTO-1ee1dfe-o75GCbrgHa_nY34Ab-ufCror--g9L6mcASjn5a09A">

UACM

Servicios

Contactos

Copyright 2014 Mandanos un mensaje via eMail a torreto619@hotmail.com

</body>

</html>

b) Página de autentificación (login.html)

</head>

<body>

<img src="http://2.bp.blogspot.com/-OsTIynMVwv0/TZ_x6urOSiI/AAAAAAAAALU/eco_JhuaaiM/s210/uacm%2Blogo%2B4.png">

<h2>

Robot m&#243vil de servicio para vigilancia y prevenci&#243n del delito.

</h2><hr>

<img class="profile-img" src="http://ssl.gstatic.com/accounts/ui/avatar_2x.png" alt="">

<form novalidate action='/aut' method='post'>

<label class="hidden-label" for="nombre">User</label>

<input id="txtUsuario" name="txtUsuario" type="text" placeholder="User">

<label class="hidden-label" for="Passwd">Contraseña</label>

<input id="txtClave" name="txtClave" type="password" placeholder="Password" class="">

<input id="signIn" name="signIn" class="rc-button rc-button-submit" type="submit" value="Iniciar sesi&#243n">

<label class="remember">

Home </form> </body> </html>

c) Página principal (privada.html)

```
<html>
<title>ROBOT UACM</title>
<HREF="http://192.168.2.113/std.css"></head>
<body bgcolor="#e6e5e1"><center>
<h2>ROBOT DE VIGILANCIA </h2><hr ><nav><ul>
   <li><a title="Opcion 1" href="/">Inicio</a></li>
   <li><a title="Opcion 2" href="/privada">Principal</a></li>
   <li><a title="Opcion 2" href="#">Base de Datos</a></li>
   <li><a title="Opcion 3" href="/salir">LogOut</a></></ul></nav><tr>
 <td bgcolor="#545454" align="center" valign="middle">
   <scriptlanguage="javascript"
type="text/javascript">doPlay();</script></TD></TR><br><center>
<button type="button" id="turn-left"> Left</button>
<button type="button" id="turn-right" > Right</button>
</center><br></table>
<center><a href="http://www.uacm.edu.mx/uacm/">UACM</a>
<a href="Nosotros.html">Servicios</a>
<a href="contactos.html">Contactos</a><br>
Copyright 2014 Mandanos un mensaje via eMail a torreto619@hotmail.com
</center>
</body></html>
```

A.IV. Conexión de sockets

a) Socket del cliente

```
socket.on('robot connected', function (data) {
     console.log(data);
     socket.emit('robot command', { command: 'nothing' }; })
   $('#turn-left').click(function() {
     socket.emit('robot command', { command: 'turn-left' }); });
   $('#turn-right').click(function() {
     socket.emit('robot command', { command: 'turn-right' }) });
```

b) Socket del servidor

```
var http = require('http').Server(app);
var io = require('socket.io')(http);
var Galileo = require("galileo-io");
```

```
var board = new Galileo();
function turnOnLed(){
board.digitalWrite(8, 1);}
function turnOffLed(){
board.digitalWrite(8, 0);}
io.on('connection', function(socket){
  console.log('conectado');
  socket.emit('robot connected', { data: 'Connected' });
  socket.on('robot command', function (data) {
    console.log(data);
    var command = data.command;
if (command == 'turn-left') {
  turnOffLed();
  console.log('led');}
if (command == 'turn-right') {
 turnOnLed();
 console.log('led2')
```

A.V. Conexión serial de la tarjeta Mbed con la tarjeta Pololu

```
#include "mbed.h"
Serial device(p9, p10); //sda, sc1
int main() {
device.baud(9600);
 wait_ms(20);
wait_ms(20);
device.putc(0xAA); // Byte de arranque
wait_ms(20);
device.putc(0x0C); // Dispositivo ID = 12
wait_ms(20);
device.putc(0x060); // Dirección del motor
wait_ms(20);
device.putc(0x7F); // VELOCIDAD CONTROL
wait_ms(20);
```

Apéndice B

B.I. Cámara IP WVC54GCA

La cámara WVC54GCA es una unidad compacta e independiente que permite controlar desde cualquier sitio. A diferencia de las cámaras Web tradicionales, que necesitan estar conectadas a un PC, la cámara se puede controlar por Interne, sea puede conectar por Wireless-G (802.11g) o mediante un cable Ethernet 10/100. Maneja una compresión de vídeo MPEG-4 y Motion JPEG para crear una secuencia de audio/vídeo de gran calidad, alta velocidad de imágenes y una resolución máxima de 640 x 480.

B.II. Microcontrolador Mbed LPC1768

El mbed LPC1768 es un microcontrolador diseñado para la creación de prototipos de todo tipo de dispositivos, sobre todo las que incluyen Ethernet, USB y la posibilidad de manejar interfaces de periféricos y memoria FLASH. Se basa en el NXP LPC1768, con un núcleo ARM Cortex-M3 de 32 bits funcionando a 96MHz. Incluye 512KB FLASH, 32 KB de RAM y tiene interfaces, como Ethernet, USB Host, CAN, SPI, I2C, ADC, DAC, PWM y otras interfaces de E/S. Los pines numerados de P5 hasta P30 también se pueden utilizar como E/S digitales.

Las características del micro son:

- Procesador de alto rendimiento ARM ® Cortex ™ -M3 Core
- Memoria FLASH de 512KB, 32 KB de RAM.
- Puertos Ethernet, USB Host / Dispositivo, 2xSPI, 2xI2C, 3xUART, CAN, 6xPWM, 6xADC, GPIO.
- 40 pines de dimensiones 54x26mm
- Aliemtacion de 5V
- Programación en Linea
- Lenguae de programación C / C ++
- Bibliotecas y proyectos de ejemplo en la plataforma Mbed.com.

B.III. Driver de potencia Pololu jrk 12v3

El 21v3 jrk es un controlador de motor versátil, de uso general que es compatible con una variedad de interfaces, incluyendo USB. El amplio rango de operación de 5 V a 28 V y corriente de salida continua de 3 A (5 A pico) permite la configuración (para Windows) de una forma fácil a través del puerto USB. (POLOLU, s.f.)

Las mejoras de este controlador sobre los anteriores controladores de motor Pololu con retroalimentación incluyen:

- Conectividad USB que permite un control directo del controlador desde una PC.

- Alta resolución interna (12 bits) para la calibración suave y flexible para varios dispositivos de entrada y de retroalimentación.
- Detección y limitación de corriente.
- Protección de la energía invertida.
- Control bidireccional simple.
- 5 V a 28 V de rango de suministro de operación.
- Salida de corriente continua máxima (5 A pico).
- Configuración sencilla y calibración a través de USB con el software de los controladores compatibles con Windows 8, Windows 7, Vista y XP.
- Parámetros configurables:
    - Período PID y constantes PID (parámetros de ajuste de retroalimentación).
    - Corriente Máxima.
    - Ciclo máximo.
    - La aceleración máxima.
    - Respuesta de errores.
- Detección de errores CRC, elimina los errores de comunicación provocados por el ruido o los fallos de software.
- Protección de la energía invertida.

Bibliografía

Adrian McEwen, H. C. (2014). Internet de las cosas, la tecnología revolucionaria que todo lo conecta. Anaya Multimedia-Anaya Interactiva.

Pérez, J. E. (2010). LibrosWeb. LibrosWeb.[En línea]: http://www. librosweb. es/ajax,

Arkin, R. (1998). Comportamiento basado en robots. Cambridge, MIT Press.

ARMMBED. (s.f.). LPC1768 mbed. Obtenido de https://developer.mbed.org/platforms/mbed-LPC1768/

Baturone, A. O. (2005). Robótica: manipuladores y robots móviles. Marcombo.

CAPAN, T. (2015). Sobre Node.js. http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

Guillermo Rauch. Socket.IO client. https://github.com/LearnBoost/socket.io-client.

I Fette and A Melnikov. Rfc 6455: The websocket protocol. Technical report,

Jimb0. (2014). Galileo Getting Started Guide. Obtenido de https://learn.sparkfun.com/tutorials/galileo-getting-started-guide

Kovanen, T. (2014). sokect.io. Obtenido de http://socket.io/

Linksys. (2007). guía de usuario. Obtenido de http://pdfretriever.com/ref_sien.php?ID=2441281&lg=S&h=d41d8cd98f00b204e9800998ecf8427e&src=15

Monzon, A. M. (2013). Introduccion a Node.js. Espsaña.

Murphy, R. (2000). Introduction to AI Robotics. MIT Press.

Ollero, A. (2001). Robótica y Manipuladores y Robots Móviles. MACOMBO.

Perkins, C. (2003). RTP: Audio and Video for the Internet. Addison-Wesley Professional.

POLOLU. (s.f.). Pololu Jrk 21v3 USB Motor Controller with Feedback. Obtenido de https://www.pololu.com/product/1392.

Raswan, A. M. K., Urbano, F. J. R., & Sánchez-Caballero, M. Á. S. (2003). Interacción remota con robots móviles basada en internet. Universidad Carlos III de Madrid, Departamento de Ingeniería de Sistemas y Automática.

Raus, R. (2000). Computer Networks Super Review. Research & Education Assoc.

Roca, O. F. (2001). Telemedicina. Panama: Ed. Médica Panamericana.

Teixeira, P. (2012). Professional Node.js: Building Javascript Based Scalable Software. John Wiley & Sons.

Thewebsocketprotocol-17. Access Ed 8-February-2012, 2011.

web, C. d. (12 de agosto de 2012). ARQUITECTURA CLIENTE-SERVIDOR. Obtenido de https://cursodecreacionweb.wordpress.com/2012/08/19/arquitectura-cliente-servidor/

Yagüe. (2003). Arquitectura para el control de robots móviles mediante delegación de códigos y agentes. Tesis de Maestría.