ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“Aplicación de visión robótica con MATLAB”

INFORME DE MATERIA DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentada por:

Angelo Moisés Abad Eras

Hugo Ernesto Acaro Gallegos

GUAYAQUIL – ECUADOR

Año: 2009

AGRADECIMIENTO

A Dios por permitirnos culminar una etapa muy importante de nuestras vidas. Al Ing. Carlos Valdivieso, Director de la materia de graduación por su colaboración y ayuda en la realización del mismo. A nuestras familias.

DEDICATORIA

Es mi anhelo dedicar y agradecer este trabajo, producto de mucho sacrificio y esfuerzo:

A ***Dios*** por estar presente en cada momento de mi vida, por ayudarme y permitirme terminar este proyecto.

A mi madre ***Mercedes*** por su amor y apoyo incondicional que desde el cielo esta presente en cada momento, a mi padre ***Hugo***, por su amor, comprensión, y apoyo incondicional con el fin de lograr alcanzar esta meta.

A mis hermanos y hermanas: **Karina, Bernardita y Jefferson**, quienes me brindaron todo su apoyo incondicional.

A mi enamorada: ***María Fernanda***, por ser mi fuente de comprensión, paciencia y apoyo durante todo este tiempo.

A mis **compañeros de trabajo**: por todo el apoyo durante todo este tiempo de realización del proyecto.

Y a todas aquellas personas que de una u otra manera me ayudaron para culminar mis estudios superiores.

Hugo Acaro Gallegos

DEDICATORIA

Mi anhelo desde mi niñez, camino muy largo que lo tome con responsabilidad y sabiduría se lo dedico con mucho amor:

A ***Dios*** por darme las fuerzas necesarias e inteligencia para poder alcanzar una de mis metas.

A mi madre **Gladys** por su amor infinito y su apoyo desde toda mi vida que siempre estuvo alado mío dándome fuerzas para finalizar mi carrera y ser una persona de bien, a mi padre **Moisés**, por sus consejos y apoyo con el fin de lograr a alcanzar esta meta.

A mis hermanos, por su ayuda y consejos incondicionales que me han inculcado para la finalización de mi carrera, en especial a mi hermano **Ing. Glen Abad** que siempre ha estado dándome sus consejos y apoyo cuando lo necesitaba.

A mi novia, **Dra. Rita Campos**, por su amor, comprensión y apoyo para culminar este proyecto.

A mis padrinos, **Walter Landeta y Cleotilde Lindao**, por sus consejos y apoyo incondicional me ayudaron con un granito de arena en el transcurso de mi carrera.

A mis amigos de trabajo, que me brindaron su apoyo cuando más lo necesitaba para poder terminar el proyecto de graduación.

Angelo Abad Eras

TRIBUNAL DE GRADUACION

Ing. Carlos Valdivieso A.

DIRECTOR DE TESIS

Ing. Hugo Villavicencio.

DELEGADO DEL DECANO

Ing. Jorge Aragundi.

SUBDECANO DE LA FIEC

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este trabajo final, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”

(Reglamento de Graduación de la ESPOL)

Angelo Moisés Abad Eras

Hugo Ernesto Acaro Gallegos

RESUMEN

El presente trabajo describe el estudio y análisis para la implementación de un Sistema de Monitoreo Visual Remoto utilizando Aplicativos de MATLAB, que visualice información por medio de una cámara IP Inalámbrica de bajo costo la misma que es dirígida por MINDSTORM ROBOT ESPIA que es operado remotamente con un sistema de control diseñado en MATLAB R2009a. Dicho sistema también permite el acceso via VNC para llevar un control sistemático y concurrente del entorno ya sea de manera local o remota mediante la creación de un acceso Web al sistema.

En el capítulo 1 se muestra una perspectiva general del sistema de visualización robótica a diseñar. Se presentan las consideraciones generales que nos permitirán el desarrollo de un diseño óptimo. Además realizamos un breve estudio de la realidad actual en cuanto a los sistemas robóticos y los diferentes sistemas disponibles en el mercado actual para conocimiento de los usuarios.

Aquí se presentan los distintos problemas y posibles soluciones existentes en el mercado que debemos considerar a la hora de seleccionar un buen sistema de seguridad robótico que sea capaz de discernir y actuar ante una posible amenaza al entorno en que operan.

En el capítulo 2 se explica en detalle las diferentes herramientas de administración y desarrollo necesarias para la implementación de este sistema robótico, así como también su posible utilidad en el desarrollo de las industrias, comunicación, y seguridad con una buena herramienta de control como lo es MATLAB.

En el capítulo 3 se explican las características del diseño y programación, al abrir una conexión USB o Bluetooth a un dispositivo de NXT y devuelve un identificador para el uso futuro, se activa el microcontrolador cuando recibe del sistema de transmisión la señal de encendido una vez cuando la programación en MATLAB este completamente. Además se explica en detalle los distintos programados que permiten el correcto funcionamiento de nuestro sistema.

En el capítulo 4 se muestran las pruebas realizadas en el Sistema Robótico permitiendo demostrar que la aplicación puede ser implementada a futuro en ambientes reales como en la industria, mecánica y comunicación que satisfagan las necesidades del hombre.

En general, este proyecto de graduación tiene como objetivo principal cubrir todos los aspectos necesarios para el desarrollo e implementación de sistemas industriales y comunicación, para brindar la seguridad de empresas y domicilios a bajo costo que puedan ser usados en cualquier ambiente con el afán de brindar seguridad y tranquilidad a los usuarios.

ÍNDICE

Pág.

CAPITULO 3

CAPITULO 4

INDICE DE FIGURAS

INTRODUCCION

Este proyecto busca diseñar y construir un móvil con herramienta de LEGO MINDSTORM NXT que sea capaz de moverse y visualizar un área determinada mediante el uso de una cámara integrada al LEGO. Todo esto se desarrolla con ayuda de programación elaborada en MATLAB R2009A donde hemos creado una interfaz gráfica que permite la comunicación entre ambos dispositivos mediante el protocolo BLUETOOTH, la detección de los diferentes obstáculos o objetos que deseemos percibir con este equipo se hacen a través de una cámara inalámbrica instalada encima de su carrocería, dicha cámara envía la información al router inalámbrico conectado al computador y ésta a la vez se comunica por una INTRANET hacia el Centro de Control desde el cual se monitorean todo lo percibido por el ROBOT, mediante el uso de IP Públicas Fijas. La comunicación es constante y se realiza vía inalámbrica de forma bidireccional, es decir la señal viaja de la computadora al móvil y viceversa; una vez realizado esto el operador puede determinar las direcciones posibles que se necesite.

Los límites del Bluetooth es de 10 a 15 metros desde la máquina que envía la señal hasta el móvil, esto limita la cobertura de dominio en el robot pero así mismo se evita errores o perdidas de comunicación. La cámara tiene un alcance de hasta 70 metros que es el límite del router.

El robot debe moverse dentro de los límites permitidos entre el Bluetooth y el móvil, para así no perder comunicación; además para poder transmitir se necesita contar con un Bandwidth (Ancho de Banda) lo suficientemente confiable para no tener robotización al momento de ser visualizado desde el cuarto de control a través de la INTRANET.

Uno de los propósitos de este proyecto es dar apoyo a estudiantes de otras carreras a que aporten ideas y nuevos usos al diseño, por ejemplo ser un detective en casos de incendios, en empresas se lo puede aplicar para monitorear varias localidades dentro de un mismo recinto, brindar mejoras mecánicas significativas y otras que le permitan al robot tener otro tipo de aplicaciones importantes.

CAPITULO 1

DESCRIPCION GENERAL DEL SISTEMA

1.1 ANTECEDENTES

La definición de autómata programable es aquel equipo electrónico, mecánico y computacional que realiza un programa de forma repetitiva pero que puede ser interrumpida para la realización de otras sin dejar de cumplir su programa principal.

Un autómata puede realizar actividades limitadas y repetitivas ya que son sistemas con movimiento limitado y su inteligencia no proviene de ellos mismos, generalmente proviene de otros sistemas externos. Actualmente los autómatas se encuentran en casi todas las industrias por su gran diversidad de aplicaciones, pero nosotros hemos visto el incentivo del medio en las comunicaciones para así darle un valor agregado a este sistema que estamos implementando.

Este proyecto no puede considerarse como un autómata aunque su inteligencia proviene de un sistema remoto y su movilidad es algo limitada, no puede programarse para realizar una actividad repetitiva ya que el objetivo de MINDSTORM ROBOT ESPIA es de recorrer distintos lugares en una misma localidad o donde su operador lo amerite.

Una definición más apropiada para MINDSTORM ROBOT ESPIA diseñado en este proyecto es la de robot móvil, ya sea mediante ruedas o extremidades móviles controladas por el operador que puede pensar, decidir y aprender de su experiencia lo que les permite reconocer y aprender ante las situaciones que se les presenten en su interacción con el mundo que le rodea. Todo el contacto con su ambiente externo y su interacción con el mismo se efectúa a través de la cámara inalámbrica que esta montado en su carrocería. A continuación muestro la figura del robot.

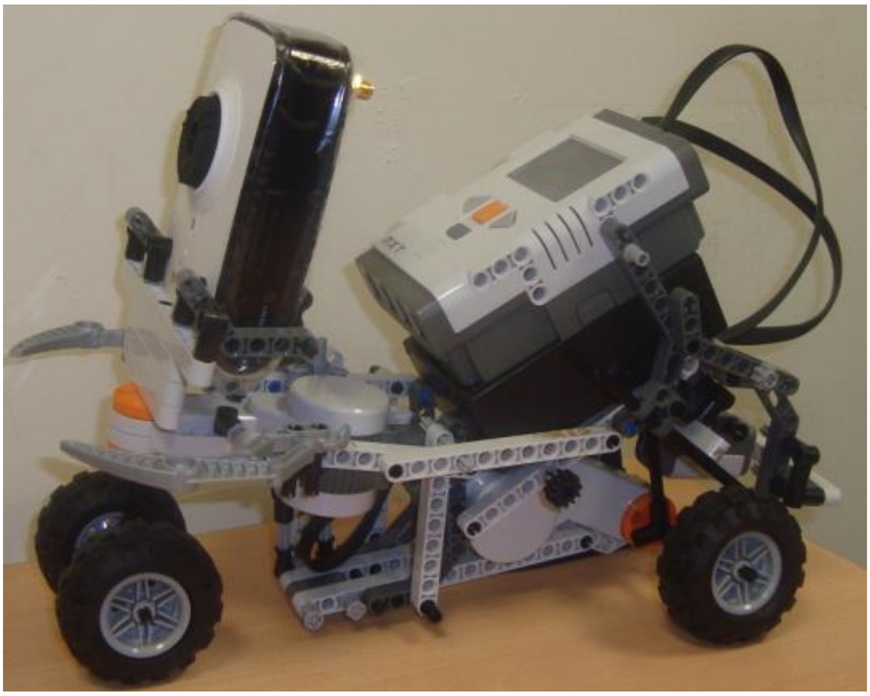


Fig.1.1 Estructura del Robot

MINDSTORM ROBOT ESPIA no es capaz de pensar y decidir por sí mismo ya que es controlado por una terminal remota, puede cumplir con la definición de robot si ha dicha terminal se le pueden implementar capacidades de aprendizaje y de tipo evolutivo ya que es capaz de interactuar con el ambiente que lo rodea el cual es un establecimiento de suma importancia para el usuario.

La inteligencia artificial se desarrollará posteriormente en futuros proyectos y podrá tener las capacidades de aprendizaje que tiene un robot las cuales ya existen y en sistemas computacionales se les conocen como agentes.

Según la definición existen tres elementos importantes en un robot que determinan su funcionalidad:

1. Programabilidad: el poder disponer de todas las capacidades computacionales y de programación en nuestro caso MATLAB.
2. Capacidad Mecánica: su capacidad para realizar acciones en un entorno y no solo un procesador de datos.
3. Flexibilidad: puede operar en un amplio rango de programas y manipular su entorno en distintas formas

Resumiendo nuestra definición, un robot es un circuito digital que ejecuta un programa, el cual puede ser modificado, para el movimiento de un sistema mecánico en base a un medio externo.

Un resumen del proceso que realiza un robot en 5 pasos es el siguiente:

Paso 1: Una alimentación del sistema.

Paso 2: Adquisición de datos del ambiente.

Paso 3: Un procesamiento de los datos adquiridos.

Paso 4: El resultado de dicho procesamiento aplicado en los actuadores o mecanismos de salida.

Paso 5: Variación del sistema en relación al ambiente en base a los resultados.

La aplicación de este tipo de robots que son capaces de reconocer el ambiente a su alrededor y transmitirlo a una terminal remota es bastante común en estos días. Un ejemplo es la exploración espacial, más concretamente la exploración a Marte con sondas espaciales que toman imágenes del planeta y las envían a la tierra para su estudio, estas sondas son móviles controlados desde una terminal remota y que interactúan con su ambiente.

No solo en el espacio, también en la tierra se usan este tipo de robots para la exploración o vigilancia de establecimientos donde un ser humano no puede estar, por ejemplo el fondo del mar, donde las altas presiones e inmensa oscuridad hacen peligrosa la exploración para los seres humanos o en el desiertos donde son usados para el estudio de fenómenos ecológicos en condiciones climáticas riesgosas para el hombre.

Fig. 1.2 Estructura del robot con la cámara incorporada

1.2 OBJETIVOS

Al final del proyecto se espera tener las siguientes características:

1. MINDSTORM ROBOT ESPIA sea capaz de recibir y ejecutar las órdenes de un operador de forma eficiente.
2. Ser capaz de ir a lugares más pequeños dentro del establecimiento para determinar su situación y darla a conocer a su operador para que tome una decisión adecuada.
3. Tener un movimiento continúo y pueda cambiar de dirección sin perder las ordenes de su operador.
4. Detectar límites en una localidad donde se esta realizando el monitoreo.
5. Establecer una comunicación inalámbrica bidireccional con un operador de forma ininterrumpida para mandar y recibir información en cualquier momento.
6. Completo control de su movilidad y suficientemente preciso para no dejar de transmitir en su jornada de trabajo.
7. Se puede alimentar con 6 pilas "AA" para un voltaje de 9 volts.

1.3 DESCRIPCION DEL SISTEMA

Este proyecto nace de las ideas implementadas en nuestro desenvolvimiento profesional de captar ideas innovadoras en las comunicaciones relacionando ramas distintas en un mismo objetivo, de las cámaras estáticas tradicionales que se ven en el mercado, darle una mejor visión en realizarlas con movimiento con ayuda de un robot.

Este proyecto consiste en un móvil que es capaz de moverse por todo un establecimiento y que envíe la información captada a una terminal remota de forma inalámbrica y reciba de esta las instrucciones para moverse a lugares distintos del establecimiento.

La aportación a este proyecto fue exclusivamente la parte de programación entre LEGO MINDSTORM NXT la misma que va hacer ordenada por comandos de programación por MATLAB las cuales incluyen los mecanismos que le dan movimiento al móvil, la parte que controla dichos mecanismos, la parte que transmite y recibe información de las cámara va hacer por un router inalámbrico que lo conectaremos hacia una computadora que tenga salida a la INTERNET por medio de IP Públicas Fijas, por medio de estas podremos ingresar vía escritorio remoto desde cualquier punto del mundo y poder manipular a nuestro MINDSTORM ROBOT ESPIA.

1.4 ANALISIS Y JUSTIFICACIÓN DEL PROYECTO

1.4.1 Análisis

Este proyecto consiste en el análisis y obtención de capturas de videos en tiempo real con ayuda de robot móvil. Este robot presenta una configuración entre el LEGO MINDSTORM NXT y MATLAB y puede ser utilizado para fines de vigilancia, seguimientos de objetos, entre otros. En este móvil se monta una cámara inalámbrica la cual apunta a la posición donde el operador haga la manipulación a MINDSTORM ROBOT ESPIA. Este robot presenta dos servomotores, el primero configurado para que le de la fuerza al móvil y el segundo su direccionamiento.

1.4.2 Justificación

La influencia de la automatización y la robótica en la sociedad actual es cada vez más notable, tanto desde el punto de vista estrictamente social en lo que hace a hábitos e incremento del confort y calidad de vida, como a los aspectos económicos directa e indirectamente relacionados. Como consecuencia, la investigación y desarrollo en este campo es de vital importancia, y marca claramente la diferencia entre países desarrollados y países en desarrollo. En estos últimos se comercializa o en el mejor de los casos se fabrican algunos productos (básicamente debido a un costo de producción más bajo) pero normalmente no se dispone de conocimiento y capacidad para realizar innovación, ni para formar personal para este fin, hecho éste que marca la diferencia. En este marco, el formar profesionales con capacidad de innovación y capacidad de acción en ámbitos de gran especialización en el contexto de las últimas tecnologías, resulta en una contribución directa y evidente a nuestra sociedad.

1.5 ALCANCES Y LIMITACIONES

El proyecto busca implementar un sistema de monitoreo visual remoto a través de un sistema robótico con el fin de emplear equipos de reducido tamaño y fácil movilidad que permitan la visualización y reconocimiento de áreas de difícil acceso dentro de las cuales únicamente podríamos acceder a través de equipos tele operados o en su defecto pre-programados para operar ante la presencia de cualquier tipo de obstáculo.

CAPITULO 2

2. FUNDAMENTACION TEORICA

MatLab es un software matemático muy poderoso, comercialmente disponible, desarrollado y distribuido por Mathworks, Inc. Es utilizado ampliamente en la academia y la industria debido a sus capacidades avanzadas, además posee una serie de herramientas que contienen funciones comúnmente usadas en ingeniería.

Todas estas herramientas facilitan el desarrollo de aplicaciones complejas de una forma más versátil. En este proyecto de graduación se hará una pequeña descripción del manejo, comportamiento del puerto COM del PC y el manejo de interfaces gráficas, mediante el uso de las herramientas proporcionadas por el software MatLab, para desarrollar aplicaciones de robótica móvil con ayuda de Protocolo Bluetooth.

La robótica es una disciplina, que tiene múltiples campos de acción. Inicialmente se diseñaron robots para uso industrial, pero con el paso del tiempo, se han ampliado sus campos de acción. Hoy en día, es común observar aplicaciones de robots en áreas como: la industrial, la investigación, la salud y el entretenimiento, entre muchas otras.

Fig. 2.1 Fotografía de MINDSTORM ROBOT ESPIA

Dentro de la robótica móvil podemos destacar los prototipos desarrollados por la NASA, para sus experimentos en el espacio exterior, los cuales efectúan tareas en entornos hostiles para el ser humano, MINDSTORM ROBOT ESPIA es un ejemplo de este tipo de aplicaciones, el cual hizo parte de la misión de explorar un establecimiento y enviar el video en tiempo real, el cual puede ser observado en la Fig.2.2.

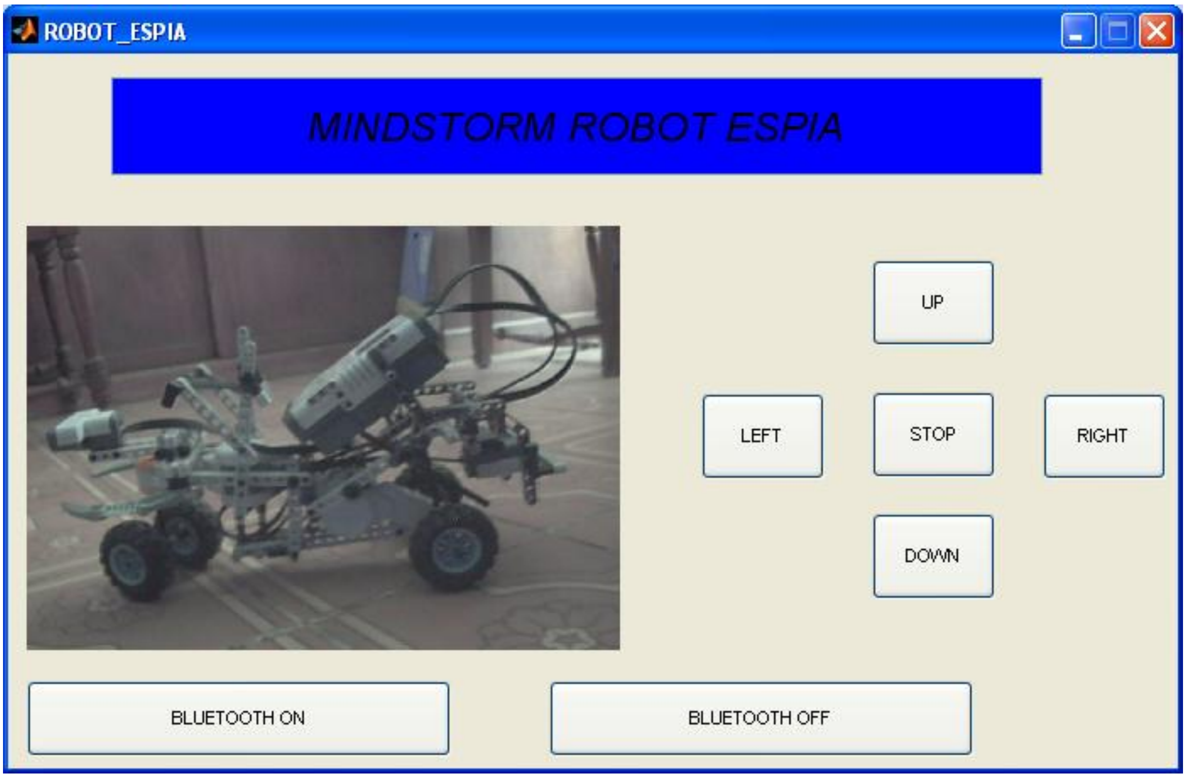


Fig. 2.2. Fotografía de MINDSTORM ROBOT ESPIA con video en línea

2.1 COMPONENTES DEL SISTEMA

2.1.1 Comunicación De Datos:

En los casos que se debe establecer una comunicación de datos por medio de una INTRANET, entre dos o más dispositivos, es necesario implementar un sistema que permita detectar errores en la comunicación. Estos errores son introducidos en la información, debido a perturbaciones en el medio de transmisión.

En aquellas situaciones que involucran comunicaciones entre computadoras, y aun dentro de un sistema de computación, existe la posibilidad de que se reciba información con errores, debido a ruidos en el canal de comunicaciones. En realidad, los símbolos binarios adoptan formas físicas, como tensiones y corrientes eléctricas. La forma física está sujeta al ruido que se introduce desde el ambiente, por ejemplo los fenómenos atmosféricos, rayos gamma y fluctuaciones de la alimentación, para nombrar solo algunas de las posibles causas. El ruido puede ocasionar errores, también conocidos como fallas, en las que un cero se convierte en un uno, o un uno se convierte en un cero.

Es posible hacer que el transmisor envíe bits adicionales de verificación junto con los bits de información. El receptor puede examinar estos bits de verificación y, ante ciertas condiciones, no solo puede detectar errores, sino también corregirlos.

MatLab:

MatLab es el nombre abreviado de “MATrix LABoratory”, es un programa especializado para realizar cálculos numéricos. Una de las capacidades más atractivas es la de poseer un lenguaje de programación propio.

Hoy en día, MatLab es usado en una variedad de áreas de aplicación incluyendo procesamiento de señales e imágenes, diseño de sistemas de control, ingeniería financiera e investigación médica.

En general, matrices y vectores son el corazón de MatLab, todos los datos son almacenados como vectores. Además de esto cuenta con herramientas de desarrollo como la GUI (Graphic User Interface), la cual permite generar entornos gráficos para las diferentes aplicaciones.

2.1.2 Interfaz Gráfica:

Se diseño una interfaz gráfica híbrida, como resultado de las pruebas realizadas al generar interfaces gráficas mediante código y mediante la GUI de MatLab.

Como resultado final se obtuvo una interfaz gráfica capaz de generar y obtener información del puerto serial, con la opción de decodificar la información adquirida, la cual se almaceno en un sistema matricial, para su posterior manipulación y estudio. La imagen de la interfaz gráfica adjunto a continuación Fig.2.3

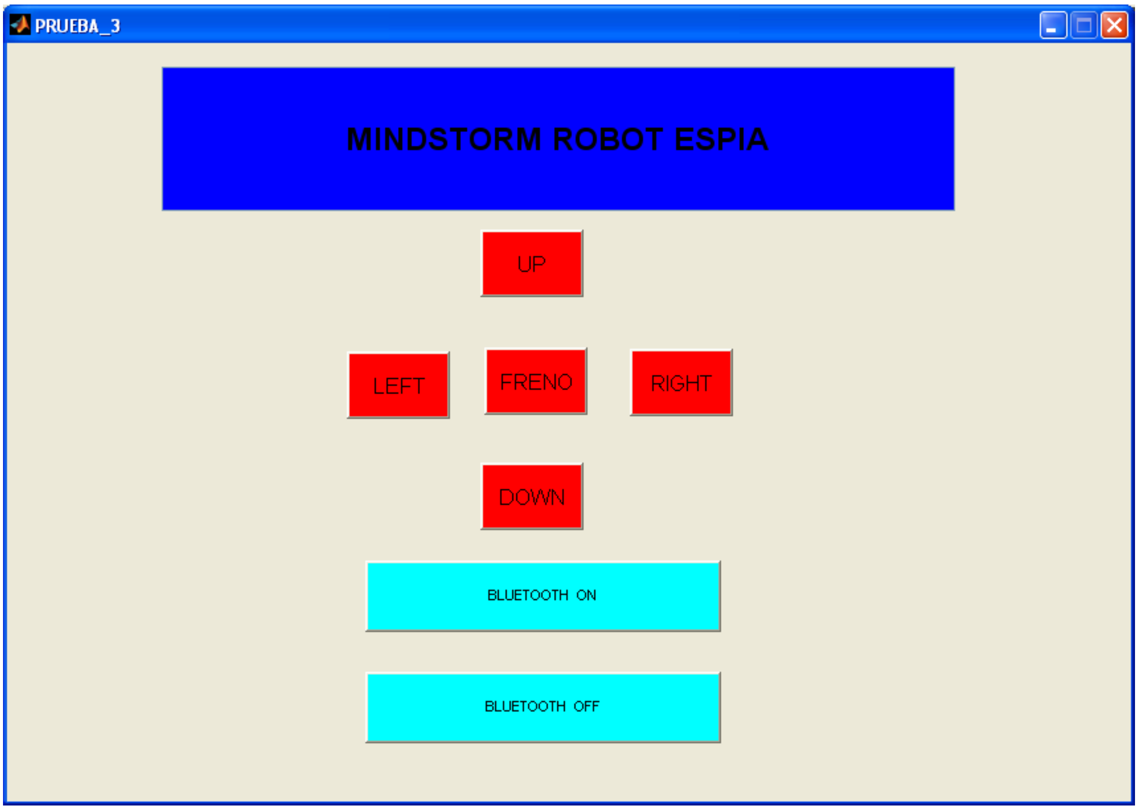


Fig. 2.3 Interfaz gráfica del programa

La interfaz gráfica fue diseñada, con la finalidad de ser manipulada fácilmente por un usuario inexperto en la materia, para lo cual fue dotada con una serie de ayudas que guían paso a paso al usuario a largo de toda la aplicación. En la figura 10, se observa el aspecto de la interfaz desarrollada.

2.1.3 Sistema de Comunicación Inalámbrico:

Se implemento un sistema inalámbrico de comunicación tanto en la unidad móvil como en el periférico externo al PC, con la finalidad de enlazar vía radiofrecuencia (RF), al microbot con el PC. Este tipo de comunicación se implemento con módulos especializados, los cuales están diseñados para codificar datos y transmitirlos ó recibirlos y decodificarlos, según sea el caso. En la figura 2.4, podemos observar una representación del sistema de comunicación inalámbrico utilizado.

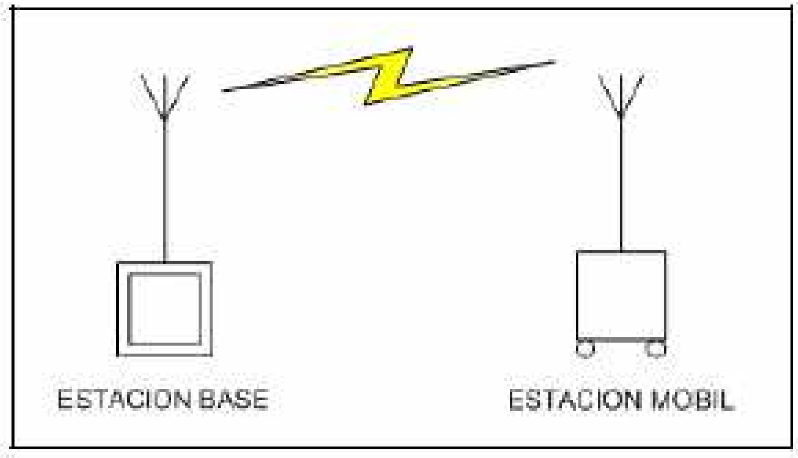


Fig.2.4. Sistema inalámbrico implementado

En las comunicaciones inalámbricas es necesario tener en cuenta que el medio introduce errores debido a perturbaciones en él, por lo cual se hace necesario emplear un sistema detector de errores. En nuestro caso especifico, se implemento el código de redundancia cíclica, para evaluar los datos recibidos que posteriormente se almacenarían en el PC.

Para implementar un sistema de comunicación inalámbrico fiable, se hicieron múltiples pruebas de alcance, transmisión y recepción

2.2 Funcionamiento conjunto de la aplicación:

Para verificar el funcionamiento de las partes anteriores descritas, se procedió a acoplar cuidadosamente todas las secciones desarrolladas de forma individual, para conformar una sola aplicación. Con las pruebas realizadas, se detectaron algunas falencias que fueron enmendadas para optimizar el funcionamiento de la aplicación implementada.

En la figura 2.5, podemos observar una representación de toda la aplicación en conjunto.

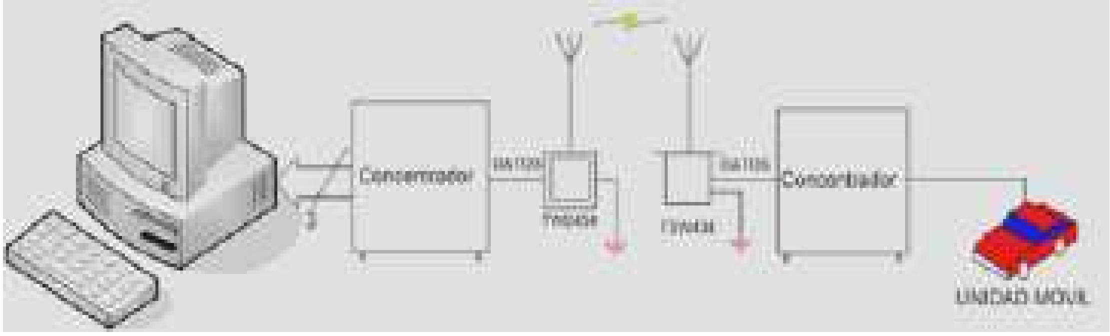


Fig.2.5. Representación en bloques de la aplicación desarrollada

CAPITULO 3

3. DISEÑO E IMPLEMENTACION DEL SISTEMA

El sistema mecánico es el que se encarga de dar movimiento y dirección al móvil, dentro de esta parte se encuentran los servomotores, engranes, mecanismos y partes móviles del sistema, en Robótica todo esto se conoce como actuadores y son los que le permiten a los robots interactuar con su ambiente junto con los sensores.

Fig. 3.1 Visión posterior del sistema mecánico

Esta parte incluye también el armazón que sostiene todos los sistemas del móvil. La figura 3.1, muestra la vista posterior del sistema mecánico sin los sistemas de control y de transmisión.

Primero se diseñó el armazón que sostiene todos los mecanismos y sistemas en base a la forma y con las características que se plantearon desde un principio, sus medidas son de 33 x 14 cm y es de forma rectangular, se hizo de esta forma para tener espacio suficiente para colocar todos los sistemas utilizados y hacer la base para el giro lo suficientemente grande para que el móvil sea capaz de apoyarse en ella. El material que se utilizó para el armazón es elaborado por LEGO MINDSTORMS.

Todos los mecanismos que se ocuparon para dar movimiento al móvil se agruparon en 2 sistemas mecánicos, el primer sistema se encarga de darle movimiento hacia delante o en reserva, el segundo sistema se encarga de hacer un cambio de dirección al móvil de 90^0 en ambos sentidos, el giro se realiza a través de un eje central capaz de soportar el móvil. Cada sistema consiste de un servomotor y señales de control específicas para cada uno.

El acoplamiento de los motores a los mecanismos de movimiento como las ruedas o la base de giro no se hace directamente, sino por medio de programación utilizando software MATLAB.

3.1 SISTEMA DE MOVIMIENTO

Para que el móvil sea capaz de moverse se uso un sistema de rodamiento que consiste básicamente de 4 ruedas en dos ejes independientes, los ejes están separados la parte posterior 14 cm y la parte delantera de 8.5 cm para poderle dar estabilidad al móvil. Se puede visualizar en la Fig. 3.2

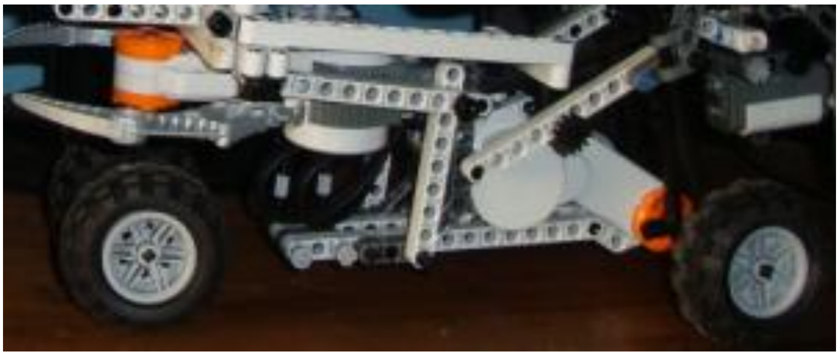


Fig. 3.2 Estructura del sistema de rodamiento

Fig. 3.3 Figura del servomotor

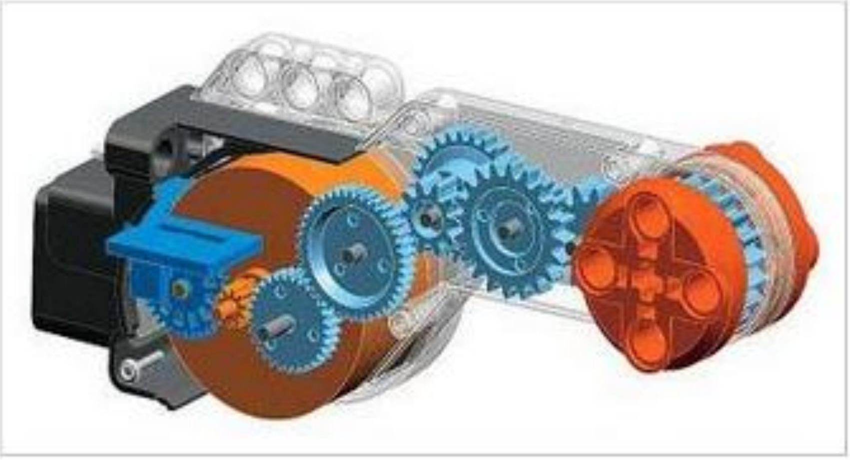


Fig. 3.4 Figura del sistema de engranajes internos del servomotor

Para controlar el funcionamiento y sentido del motor de movimiento se utilizó un servo motor (Fig. 3.3) cuyo interior esta compuesto de un sistema de control llamado puente H (Fig. 3.4), este sistema nos permite controlar el sentido en el que girará el motor por medio de dos señales de control, que pueden ser señales digitales para mantener un solo voltaje o de pulso controlado (PWM), para variar el voltaje a través de los motores y por consiguiente la velocidad del motor pero para este caso si vamos a variar la velocidad para que su movimiento sea cada vez mas rápido cuando lo amerite.

El puente H que se utilizó esta diseñado para que se controle con señales digitales las cuales serán enviadas desde un microcontrolador 32-bit ARM7.

Este sistema se activa cuando el microcontrolador recibe del sistema de transmisión la señal de encendido y no se detendrá hasta que no se lo frenemos.

3.1.1 SISTEMA DE DIRECCION

Para controlar el sentido de giro del motor se utilizó otro puente H igual al utilizado para el motor de movimiento. Como la función de este motor es de posición tampoco requiere una variación de velocidad, simplemente mandamos la señal digital de control correspondiente. En la figura 3.5 Se muestra el sistema de giro móvil que consta de un motor de DC acoplado al eje de giro.

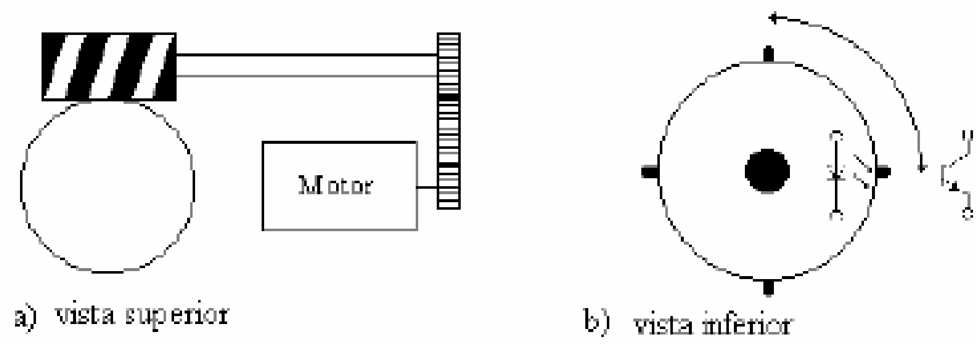


Fig. 3.5 Diagrama del servomotor

3.2 IMPLEMENTACION DEL SISTEMA

El sistema de control del móvil es aquel que hace funcionar nuestro robot el mismo que consta de una estructura física que consta de dos servomotores; el primero que da fuerza a la estructura física y el segundo que brinda la dirección. En esta estructura está ubicada la cámara que visualizará y monitoreará el sector.

Para desarrollar el movimiento empleamos el programa Robótica_matlab que fue desarrollado en GUI que da la dirección hacia adelante, atrás izquierda y derecha, así como abrir la comunicación del protocolo Bluetooth por medio del botón **Bluetooth On**; una vez realizado el enlace bluetooth nosotros tenemos control sobre el robot. Asi mismo al momento de cerrar la conexión bluetooth se debe aplastar el botón de **Bluettoh Off** para detener el programa.

A continuación presentamos los comandos utilizados en el programa con su respectivo detalle del funcionamiento de cada uno de ellos.

COM_OpenNXT

Abre una conexión USB o Bluetooth a un dispositivo de NXT y devuelve un identificador para el uso futuro

Contenidos

- **Sintaxis**
- **Descripción**
- **Limitaciones de COM_CloseNXT**
- **Ejemplo**

Sintaxis

handle = COM_OpenNXT()

handle = COM_OpenNXT(inifilename, 'Check')

Descripción

handle = COM_OpenNXT() intenta abrir una conexión vía USB. El primer dispositivo NXT que se encuentra para ser utilizado. Los controladores de dispositivos (Fantom en Windows, libusb en Linux) tienen que estar ya instalados para que el USB trabaje.

handle = COM_OpenNXT(inifilename, 'Check') buscará el bus USB para dispositivos de NXT, así como la sintaxis sin ningún parámetro. Si esto no funciona por alguna razón (no hay conexión USB a la NXT disponible, no hay controladores de dispositivos instalados, o el dispositivo de NXT está ocupado), la función intentará establecer una conexión vía Bluetooth, utilizando el archivo de configuración de Bluetooth dado (usted también lo puede crear fácilmente con la función COM_MakeBTConfigFile). Los parámetros opcionales "se puede omitir”, pero que no es recomendable (ya que detecta los errores de la conexión Bluetooth fuera de MATLAB de inmediato). Para obtener más información acerca del« chequeo »de la opción, consulte la documentación sobre NXT_CloseNXTEx.

Tenga en cuenta que esta función es la forma más sencilla de obtener el manejo del NXT. Si usted necesita un método de acceso a NXTs múltiples o más opciones, consulte la función avanzada COM_OpenNXTEx. De hecho, COM_OpenNXT es sólo un envoltorio conveniente COM_OpenNXTEx('all', ...).

Este comando es utilizado en la linea que a continuación se indica:

```
handle = COM_OpenNXT('bluetooth.ini','check');%abre comunicacion
BLUETOOTH
COM_SetDefaultNXT(handle);%configura el puerto COM bluetooth
```

Limitaciones de COM_CloseNXT

Si usted llama COM_CloseNXT('all') después de que un comando clear all se ha publicado, la función no será capaz de cerrar todas las USB libres restantes que fueron ocupadas, ya que han sido retirados de la memoria. Este es un problema en sistemas Linux. Usted no será capaz de utilizar el dispositivo NXT sin reiniciar. Solución: o bien usar sólo clear en sus programas, o bien utilizar el COM_CloseNXT ('all') antes de la declaración en clear all. La mejor manera sin embargo, es el seguimiento de sus NXT con cuidado y cerrar manualmente (COM_CloseNXT (handle)) antes de salir siempre que sea posible!

Ejemplo

```
handle = COM_OpenNXT( 'bluetooth.ini', 'check');
```

```
    COM_SetDefaultNXT(handle);
    NXT_PlayTone(440,10);
    COM_CloseNXT(handle);
```

COM_CloseNXT

Se cierra y se elimina una NXT específica manejada, o borra todos los NXT existentes manejados.

Contenidos

- **Sintaxis**
- **Descripción**
- **Limitaciones**
- **Ejemplo**

Sintaxis

COM_CloseNXT(handle)

COM_CloseNXT('all')

COM_CloseNXT('all', inifilename)

Descripción

Después de usar NXT handle, un usuario debe liberar el dispositivo (y la memoria ocupada por el handle), llamando a este método. Después de la limpieza invocada por la presente convocatoria, un ladrillo NXT puede acceder y ser utilizado de nuevo por COM_OpenNXT o COM_OpenNXTEx.

COM_CloseNXT(handle) se cerrará y borrara el dispositivo especificado. handle tiene que ser un identificador válido es una estructura creada por cualquiera de las COM_OpenNXT o COM_OpenNXTEx.

COM_CloseNXT('all') se cierra y borra todos los dispositivos existentes NXT de la memoria (siempre y cuando la caja de herramientas puede hacer un seguimiento de ellos). Todos los USB manejados serán destruidos, todos los puertos abiertos de serie (para las conexiones Bluetooth) serán cerrados. Esto puede ser útil en el inicio de un programa para crear un "nuevo comienzo" y en un conjunto bien definido de partida. Tenga en cuenta que el comando clear all puede mostrar un mensaje de falla (esto se debe a que no todos los dispositivos USB se encuentran cerrados, ya que toda la información sobre ellos se ha liberado de la memoria de MATLAB). Si esto ocurre, un dispositivo de NXT puede parecer que esta ocupado y no puede ser utilizado. Por lo general reiniciar el NXT ayuda, si no se intenta reiniciar MATLAB también. Así que tenga cuidado con el uso de clear all antes | COM_CloseNXT ('all').

COM_CloseNXT('all', inifilename) hará lo mismo que arriba, pero en lugar de cerrar todos los puertos abiertos de serie, sólo cierra el COM-puerto especificado en inifilename serán usados (un archivo válido de configuración de Bluetooth puede ser creado por la función COM_MakeBTConfigFile). Esta sintaxis ayuda a evitar las interferencias con otros puertos en serie que podría ser utilizado por otros programas (MATLAB) al mismo tiempo. Tenga en cuenta que aun estando abiertos todos los dispositivos USB se cerrarán.

Limitaciones

Si usted llama COM_CloseNXT('all') después que un comando clear all se ha publicado, la función no será capaz de cerrar todas las USB libre restantes ocupadas, ya que han sido retirados de la memoria. Este es un problema en sistemas Linux. Usted no será capaz de utilizar el dispositivo NXT sin reiniciar. Solución: o bien usar sólo clear en sus programas, o bien utilizar el COM_CloseNXT('all') antes de la declaración clear all. La mejor manera sin embargo, es el seguimiento de sus NXT con cuidado y cerrar manualmente antes de salir siempre que sea posible.

Ejemplo

```
handle = COM_OpenNXT('bluetooth.ini', 'check');
  COM_SetDefaultNXT(handle);
  NXT_PlayTone(440,10);
  COM_CloseNXT(handle);
```

En nuestro programa el comando interviene en la siguiente función que a continuación indico:

```
function bluetoff_Callback(hObject, eventdata, handles)
% hObject    handle to bluetoff (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
COM_CloseNXT('all');% cierra comunicacion bluetooth
```

COM_SetDefaultNXT

Establece un valor predeterminado global NXT handle que será utilizado por otras funciones si no se especifica.

Contenidos

- **Sintaxis**
- **Descripción**
- **Ejemplo**

Sintaxis

COM_SetDefaultNXT(h)

Descripción

COM_SetDefaultNXT(h) establece el manejo dado a la h NXT mundial handle, que es utilizado por todos los NXT-funciones por defecto si no se especifica ningún otro handle. Para crear y abrir una NXT handle (Bluetooth o USB), las funciones COM_OpenNXT y COM_OpenNXTEx pueden ser utilizadas. Para recuperar el valor predeterminado global manejar COM_GetDefaultNXT usuario.

Ejemplo

```
 MyNXT = COM_OpenNXT ( 'bluetooth.ini', 'check');
    COM_SetDefaultNXT (MyNXT);
```

En nuestro programa el comando interviene en la función blueton que adjunto a continuación:

```
function blueton_Callback(hObject, eventdata, handles)
% hObject    handle to blueton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handle   =   COM_OpenNXT('bluetooth.ini','check');%abre   comunicacion
BLUETOOTH
COM_SetDefaultNXT(handle);%configura el puerto COM Bluetooth
```

StopMotor

Paradas / frenos de motor especificado. (La sincronización se perderá después de esto)

Contenidos

- **Sintaxis**
- **Descripción**
- **Ejemplo**

Sintaxis

StopMotor(port, mode)

StopMotor(port, mode, handle)

Descripción

StopMotor(port, mode) detiene el motor conectado al puerto determinado. El valor port pueden ser abordadas por las constantes simbólicas MOTOR_A, MOTOR_B, MOTOR_C y 'all' (todos los de motor a la vez) analógico al etiquetado sobre el brick NXT. El modo de argumentación puede ser igual a "off" que corta la corriente eléctrica al motor específico, llamado "Coast" de modo. La opción de freno 'brake' activa detener el motor en la posición actual (hasta el siguiente comando).

El argumento opcional último NXT puede ser un identificador válido. Si no se especifica ninguno, el valor predeterminado para manejar serán usados (llamando COM_SetDefaultNXT para establecer uno).

Nota:

El puerto con valor igual a 'all' se utilizará para detener todos los motores, al mismo tiempo con un solo paquete de Bluetooth. Después de un comando StopMotor la snychronization del motor se perderán.

Con el modo igual a "off", el motor deje de girar lentamente, pero usando «brake» se aplica la fuerza real para el motor a permanecer inmóvil en la posición actual, al igual que un freno real.

Ejemplo

```
SetMotor(MOTOR_B);
            SetPower(-55);
```

```
             SetAngleLimit(1200);
   SendMotorSettings();
 StopMotor(MOTOR_B, ’brake’);
```

A continuación adjunto la función en la que interviene el comando StopMotor:

```
function IZQUIERDA_Callback(hObject, eventdata, handles)
% hObject    handle to IZQUIERDA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
StopMotor('all','off');% comando que detiene el motor B
SetMotor(MOTOR_B);% se inicializa motor B
SetPower(-100);%se configura el motor a 100% en forma inversa
SetAngleLimit(15);%configura a 15 grados giro motor izquierda
SetTurnRatio(0);%realiza sincronismo entre los motores
SendMotorSettings();%envia los datos configurados al motor
StopMotor(MOTOR_B,'off');% se detiene el motor B
```

SetMotor

Juegos de la corriente del motor a utilizar para fijar los comandos de motor

Contenidos

- **Sintaxis**
- **Descripción**
- **Ejemplo**

Sintaxis

SetMotor(port)

Descripción

SetMotor(port), establece la corriente del motor en el puerto dado en que el motor de corriente esta activo y que pueden ser controlados por SendMotorSettings. Cambiar la configuración del motor toma sólo un efecto en el motor activo. El puerto de valor puede ser dirigido por las constantes simbólicas MOTOR_A, MOTOR_B y analógicas MOTOR_C al etiquetado sobre el ladrillo NXT.

El ajuste de la corriente del motor puede ser devuelto por la función GetMotor.

Ejemplo

```
SetMotor (MOTOR_C);
            SetPower(76);
            SetAngleLimit(720);
  SendMotorSettings();
```

SetPower

Establece la potencia del motor de corriente activos

Contenidos

- Sintaxis
- Descripción
- Ejemplo

Sintaxis

`SetPower(power)`

Descripción

SetPower(power) establece el valor de potencia del motor de corriente activos que pueden ser controlados por SendMotorSettings. El valor de power tiene que ser un valor entero dentro del intervalo de -100 --- 100. Se especifica el porcentaje de la máxima potencia disponible. Los valores negativos indican una dirección de rotación inversa. El ajuste de la potencia se afecta sólo con el siguiente comando SendMotorSettings.

Ejemplo

```
SetMotor(MOTOR_B);
            SetPower(-55);
            SetAngleLimit(1200);
   SendMotorSettings();
```

SetAngleLimit

Establece el límite de ángulo (en grados) del puerto de corriente del motor

Contenidos

- Sintaxis
- Descripción
- Ejemplo

Sintaxis

SetAngleLimit(limt)

Descripción

SetAngleLimit(limt) establece el límite de ángulo en grados de la corriente del motor, que se fija por la función SetMotor. Este ajuste sólo se afectan con el siguiente comando SendMotorSettings.

Utilice SetAngleLimit (0) para desactivar este límite (luego de ello el motor funciona siempre).

Nota:

Este comando proporciona una forma automática de rotación del motor limitado. El ladrillo NXT tratará de girar sólo el ángulo que se establece. Desafortunadamente, si el poder es demasiado alto, el ángulo de destino será un poco perdido, como el motor tiene todavía suficiente momento angular para mantenerse girando un poco. Vías alrededor pueden frenar el motor antes de que se acerque a su límite de ángulo y luego seguir con mucho cuidado, o de buen grado perder el límite de ángulo y luego revertir de nuevo a donde quieres ir, precisamente.

Ejemplo

```
SetMotor (MOTOR_B);
          Setpower (76);
          SetAngleLimit (720);
  SendMotorSettings ();
```

SendMotorSettings

Envía previamente el dato especificado para la configuración de motor activo.

Contenidos

- Sintaxis
- Descripción
- Ejemplo

Sintaxis

SendMotorSettings()

SendMotorSettings(port, power, angle, speedRegulation, syncedToMotor, turnRatio, rampMode)

Descripción

SendMotorSettings() envía los ajustes previamente determinado de corriente del motor. Los ajustes de motor son establecidos por las funciones SetMotor, fijapoder, SetAngleLimit, SpeedRegulation, SyncToMotor, SetTurnRatio y SetRampMode.

SendMotorSettings(port, power, angle, speedRegulation, syncedToMotor, turnRatio, rampMode) envía la configuración dada, como el puerto de motor (MOTOR_A, MOTOR_B o MOTOR_C), el poder (-100 ... 100, el límite del ángulo, speedRegulation ("on" , 'off'), (syncedToMotor MOTOR_A, MOTOR_B, MOTOR_C), turnRatio (-100 ... 100) y rampMode ('off', 'up', 'abajo').

Nota:

Todos los ajustes como el poder, limitar el ángulo y así sucesivamente, sólo tendrán efecto una vez que se envían al motor que utilizan esta función. Tenga en cuenta que si tiene dos motores sincronizados juntos, esta función internamente envía dos paquetes tanto para los motores. Esto es necesario, pero puede trabajar como si fuera un solo comando.

Ejemplo

```
SetMotor (MOTOR_B);
            SyncToMotor(MOTOR_C);
            Setpower(76);
            SetAngleLimit(4 * 360);
            SetTurnRatio(20);
   SendMotorSettings();
```

CAPITULO 4

4. SIMULACION, IMPLEMENTACION Y PRUEBAS EXPERIMENTALES

4.1 Introducción

El programa fue diseñado en un entorno de desarrollo visual de MATLAB llamado GUI en cual permite la programación en un entorno amigable. Este programa desarrolla 2 archivos, el Robotica_matlab.fig que es el entorno gráfico ejecutable y el Robótica_matlab.m que contiene las líneas de códigos del programa. A continuación se indica el gráfico del archivo Robótica_matlab.fig

Fig. 4.1 Robot móvil en pruebas experimental

4.2 SIMULACION E IMPLEMENTACION

El programa fue diseñado en un entorno de desarrollo visual de MATLAB llamado GUI en cual permite la programación en un entorno amigable. Este programa desarrolla 2 archivos, el Robótica_matlab.fig que es el entorno gráfico ejecutable y el Robotica_matlab.m que contiene las líneas de códigos del programa. A continuación se indica el gráfico del archivo Robótica_matlab.fig en la fig. 4.2

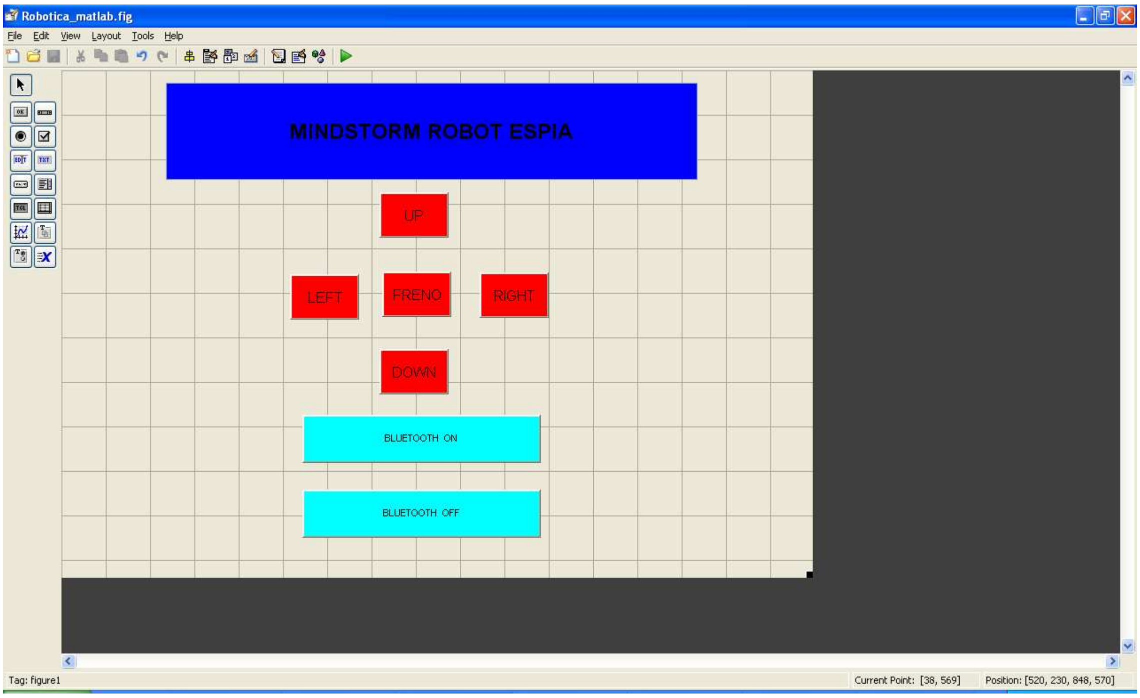


Fig. 4.2 Gráfico del programa en entorno GUI

A continuación se muestra el programa ejecutable el cual maneja al robot con las flechas direccionales que se especifican a continuación:

UP: El robot se desplaza hacia adelante.

DOWN: El robot se desplaza hacia atrás.

LEFT: El robot se desplaza hacia la izquierda.

RIGHT: El robot se desplaza hacia la derecha.

BLUETOOTH ON: Se encarga de iniciar el protocolo bluetooth.

BLUETOOTH OFF: Se encarga de finalizar el protocolo bluetooth.

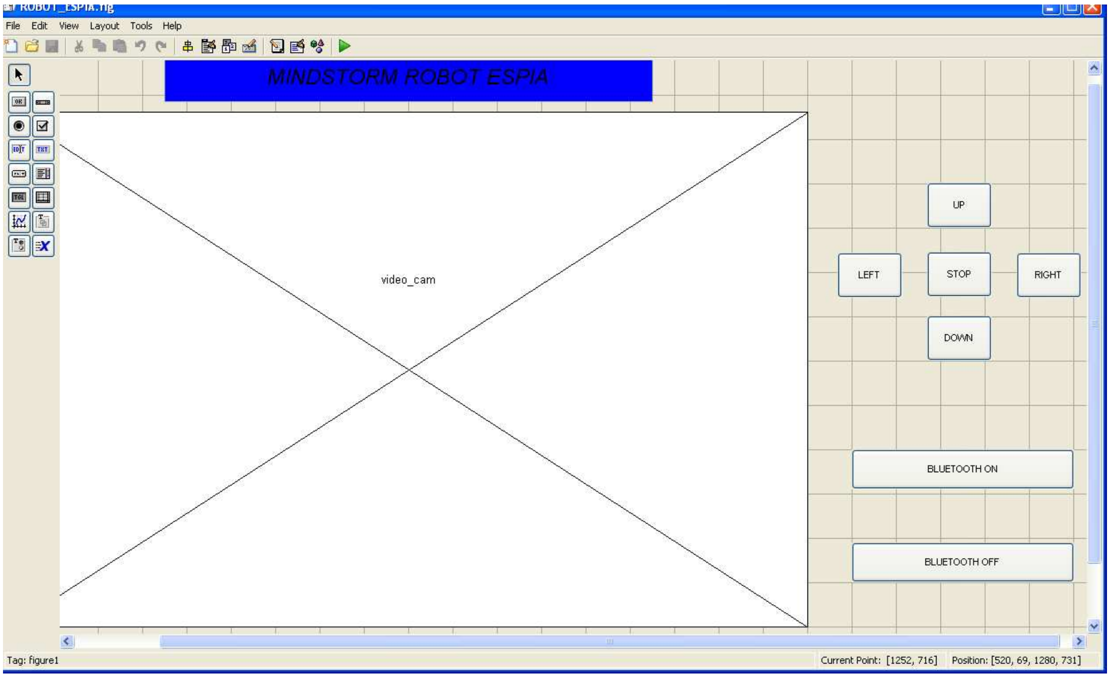
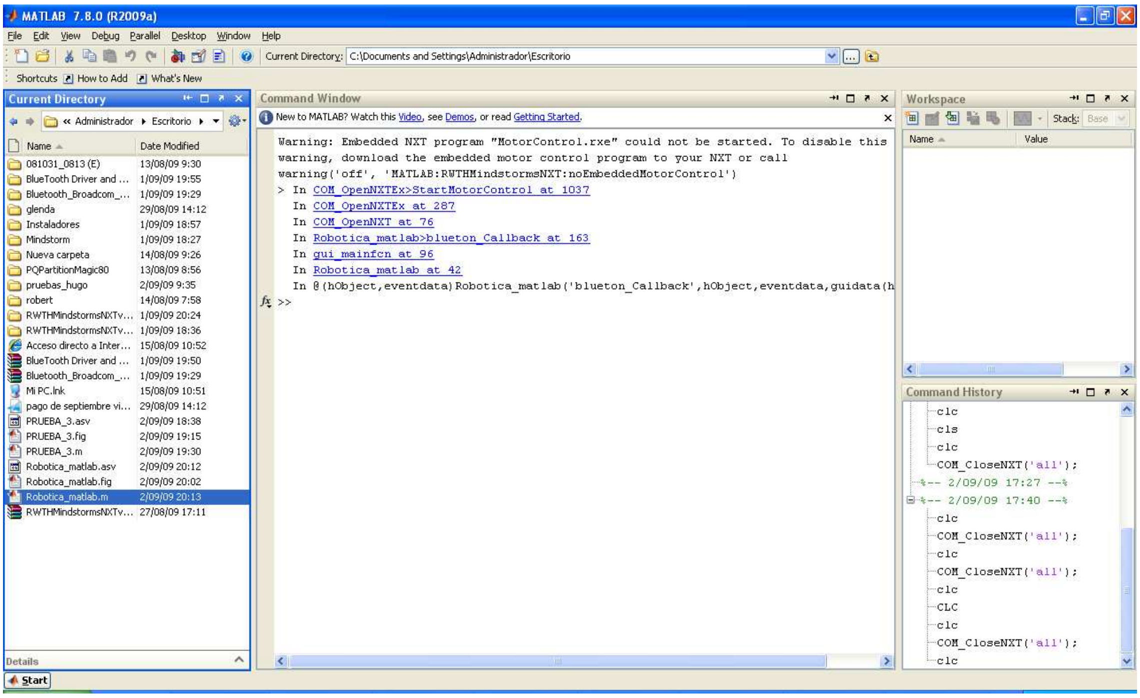


Fig. 4.3 Programa del Midstorm Robot Espia en entorno GUI

Una vez presionado el botón **BLUETOOTH ON** se inicia el protocolo en el cual MATLAB verifica que las funciones estén inicializándose, una vez realizado la comunicación se tiene control al robot. Fig. 4.3



Current Directory: C:\Documents and Settings\Administrador\Escritorio

« Administrador ▸ Escritorio ▸

Name	Date Modified
081031_0813 (E)	13/08/09 9:30
BlueTooth Driver and ...	1/09/09 19:55
Bluetooth_Broadcom_...	1/09/09 19:29
glenda	29/08/09 14:12
Instaladores	1/09/09 18:57
Mindstorm	1/09/09 18:27
Nueva carpeta	14/08/09 9:26
PQPartitionMagic80	13/08/09 8:56
pruebas_hugo	2/09/09 9:35
robert	14/08/09 7:58
RWTHMindstormsNXTv...	1/09/09 20:24
RWTHMindstormsNXTv...	1/09/09 18:36
Acceso directo a Inter...	15/08/09 10:52
BlueTooth Driver and ...	1/09/09 19:50
Bluetooth_Broadcom_...	1/09/09 19:29
Mi PC.lnk	15/08/09 10:51
pago de septiembre vi...	29/08/09 14:12
PRUEBA_3.asv	2/09/09 18:38
PRUEBA_3.fig	2/09/09 19:15
PRUEBA_3.m	2/09/09 19:30
Robotica_matlab.asv	2/09/09 20:12
Robotica_matlab.fig	2/09/09 20:02
Robotica_matlab.m	2/09/09 20:13
RWTHMindstormsNXTv...	27/08/09 17:11

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

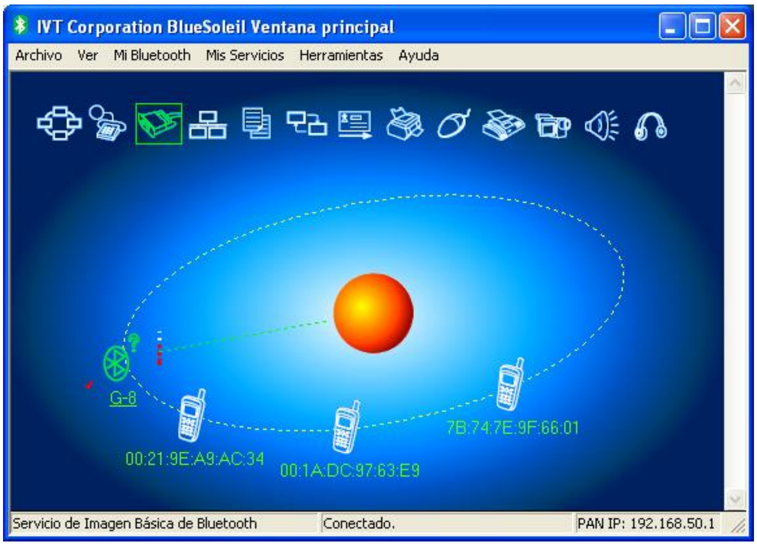
```
Warning: Embedded NXT program "MotorControl.rxe" could not be started. To disable this
warning, download the embedded motor control program to your NXT or call
warning('off', 'MATLAB:RWTHMindstormsNXT:noEmbeddedMotorControl')
> In COM_OpenNXTEx>StartMotorControl at 1037
  In COM_OpenNXTEx at 287
  In COM_OpenNXT at 76
  In Robotica_matlab>blueton_Callback at 163
  In gui_mainfcn at 96
  In Robotica_matlab at 42
  In @(hObject,eventdata)Robotica_matlab('blueton_Callback',hObject,eventdata,guidata(h
>>
```

```
clc
cls
clc
COM_CloseNXT('all');
%-- 2/09/09 17:27 --%
%-- 2/09/09 17:40 --%
clc
COM_CloseNXT('all');
clc
COM_CloseNXT('all');
clc
CLC
clc
COM_CloseNXT('all');
clc
```

Fig. 4.4 Indica el inicio del programa en MATLAB

Una vez inicializado el programa el MATLAB indica las funciones que fueron utilizadas. Fig. 4.4

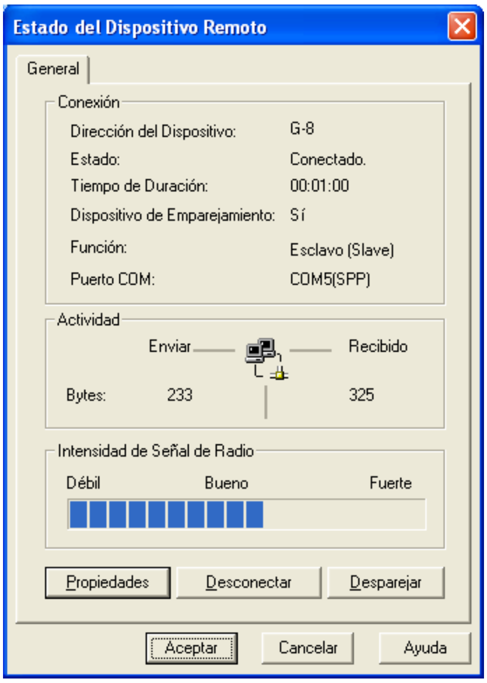
Se puede observar como inicia la conexión entre el protocolo Bluetooth y MatLab, él cual la Laptop le asigna los puertos que tiene disponible, en nuestro caso el puerto disponible es el puerto COM5.



Servicio de Imagen Básica de Bluetooth | Conectado. | PAN IP: 192.168.50.1

Fig..4.5 Indica el estado del la conexión Bluetooth

Una vez realizada la conexión, en el dispositivo USB Bluetooth indica que se ha establecido la conexión. Fig. 4.5



Conexión

Dirección del Dispositivo: G-8
Estado: Conectado.
Tiempo de Duración: 00:01:00
Dispositivo de Emparejamiento: Sí
Función: Esclavo (Slave)
Puerto COM: COM5(SPP)

Actividad

	Enviar	Recibido
Bytes:	233	325

Intensidad de Señal de Radio

Débil Bueno Fuerte

Fig. 4.6 Puerto COM se establece la conexión

El dispositivo indica el puerto COM en la que se establece la conexión y el estado del mismo. Fig. 4.6

4.3 PRUEBAS EXPERIMETALES

Las pruebas fueron realizadas en un nodo de comunicación en el cual se simuló el monitoreo de local desde el laboratorio de la universidad con lo que se pudo observar un nivel de retardo en la imagen entre las cuales se pudo deducir cual es el consumo del ancho de banda para que la imagen pueda visualizarse con mayor fluidez. Fig. 4.7

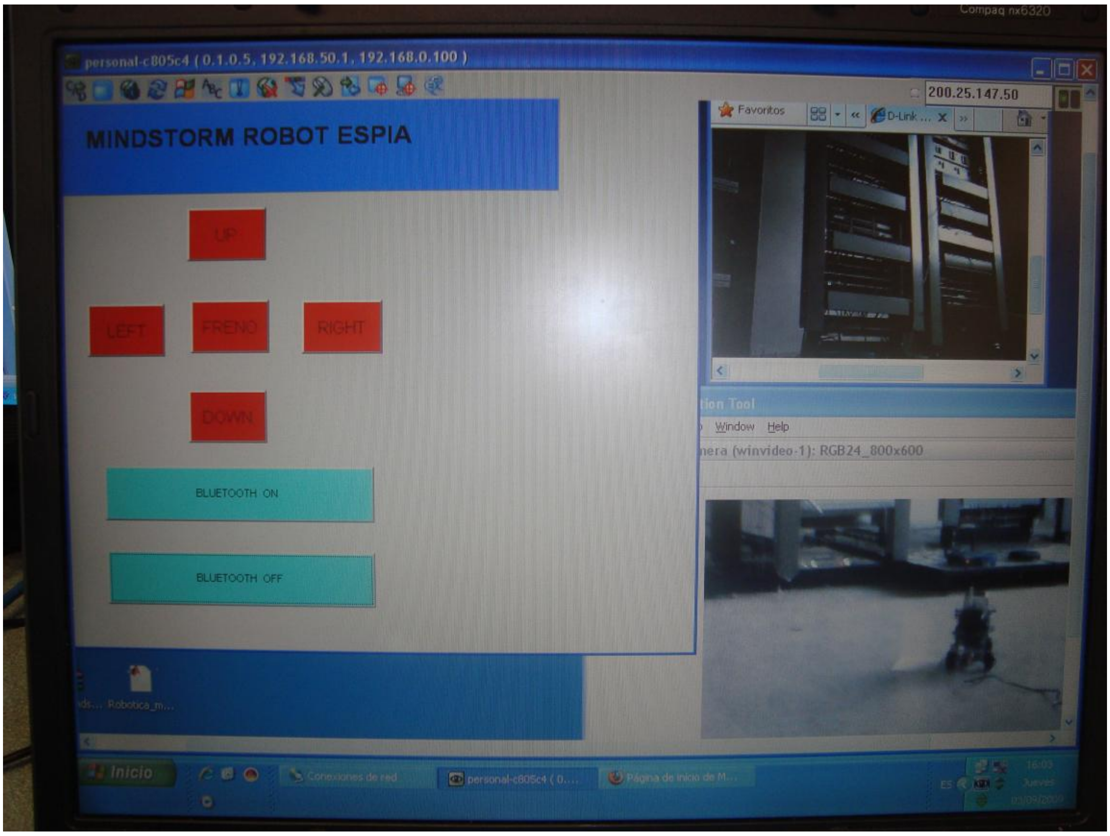


Fig. 4.7 Funcionamiento del MINDSTORM ROBOT ESPIA

Conclusiones

1. La instalación de un sistema Robótico de seguridad como el presentado en este proyecto, constituye una característica interesante aplicable a nivel industrial. El control interno y externo del monitoreo remoto se maximiza con el uso de un móvil que permite tomar capturas requeridas por el operador.

2. MATLAB en el sistema Robótico de seguridad no tan solo ha sido una herramienta de desarrollo, sino que también es servidor del mismo, ya que no solo es la interfaz (Usuario – Sistema Robótico de Seguridad); es decir, es el medio a través del cual podemos manipular el monitoreo de varias opciones donde lo podamos emplear, accediendo remotamente desde cualquier PC con disponibilidad de Internet; a más de eso, es el medio en el cual interactúan todos los dispositivos de hardware y software.

3. Comenzamos con una idea innovadora para dar uso a LEGO MINDSTORM NXT, cuyos objetivos propuestos los hemos alcanzado; es decir, hemos podido desarrollar e implementar un buen sistema de seguridad robótico, pero como en la mayoría de los casos siempre estaremos atados a ciertas limitaciones que fueron mencionadas en capítulos anteriores.

4. Un sistema de seguridad robótico por más sofisticado que sea se encontrará atado al factor humano por su programación; es decir, al sistema de seguridad se lo utilizará como herramienta de prevención y monitoreo más no en un sistema de protección invulnerable.

5. Al concluir el proyecto se observa que cuando ingresamos al área monitoreada por MINDSTORM ROBOT ESPIA debemos tener muy en cuenta las relaciones que hay entre los aspectos: tecnológicos y el hombre, ya que si uno de estos fallan, entonces falla todo el sistema robótico.

Recomendaciones

1. MINDSTORM ROBOT ESPIA debe ubicarse en un lugar libre de humedad y obstáculos que lo interfieran, y no fuera del alcance del dispositivo Bluetooth.

2. Al momento de instalar y configurar la cámara IP inalámbrica obligatoriamente se debe seguir las recomendaciones e instrucciones de los mismos, ya que un caso particular de la cámara ubicada en el móvil podría dar robotización de la imagen por el movimiento acelerado, se requiere ubicarlos a una altura determinada.

3. El Servidor Central del sistema; en este caso, el PC que posee la aplicación en MATLAB debe mantenerse encendido y con conexión on en BLUETOOTH para poder mantener la visualización que será captada MINDSTORM ROBOT ESPIA.

4. Se debe evitar el uso del PC servidor para otras aplicaciones ya que el mismo puede poner lento el sistema; es decir, si se requiere acceder desde un PC remoto debe de tener un Ancho de Banda de 1000Kbps, de lo contrario su acceso externo al monitoreo se hará con cierto retardo

5. Para incrementar la seguridad del enlace remoto vía Internet, se recomienda el uso del protocolo http, el mismo que utiliza certificados de seguridad y encripta los datos intercambiados entre el servidor y la estación remota.

6. El sistema de seguridad diseñado en este proyecto utiliza parámetros que garantizan que las imágenes captadas por la cámara IP son lo suficientemente claros. El aumento en la calidad de imagen del sistema es totalmente independiente de cada usuario; es decir, a conveniencia propia pueden utilizar mejor resolución en caso remoto incrementando su Ancho de Banda y segmentándola para al momento de la transmisión.

7. Las seguridades y protecciones configuradas en el sistema de seguridad presentado en este proyecto, garantizan que sólo personal autorizado tenga acceso a la información del servidor y las imágenes captadas por la cámara. Cualquier modificación en las configuraciones se la puede realizar primeramente apagando la comunicación Bluetooth.

Anexos

```
function varargout = Robotica_matlab(varargin)
% Robotica_matlab M-file for Robotica_matlab.fig
%      Robotica_matlab, by itself, creates a new Robotica_matlab or
raises the existing
%      singleton*.
%
%      H = Robotica_matlab returns the handle to a new
Robotica_matlab or the handle to
%      the existing singleton*.
%
%      Robotica_matlab('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in Robotica_matlab.M with the given
input arguments.
%
%      Robotica_matlab('Property','Value',...) creates a new
Robotica_matlab or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Robotica_matlab_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Robotica_matlab_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Robotica_matlab

% Last Modified by GUIDE v2.5 02-Sep-2009 21:02:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Robotica_matlab_OpeningFcn,
...
                   'gui_OutputFcn',  @Robotica_matlab_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Robotica_matlab is made visible.
function Robotica_matlab_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Robotica_matlab (see
VARARGIN)

% Choose default command line output for Robotica_matlab
handles.output = hObject;



% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Robotica_matlab wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = Robotica_matlab_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in ADELANTE.
function ADELANTE_Callback(hObject, eventdata, handles)
% hObject    handle to ADELANTE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
StopMotor('all','off');% comando que detiene el motor A
SetMotor(MOTOR_A);% se inicializa motor A
SetPower(100);% se configura el motor al 100%
SetAngleLimit(0);%se configura los grados del movimiento
```

```
SetTurnRatio(0);%realiza sincronismo entre los motores
SendMotorSettings();%envia los datos configurados al motor


% --- Executes on button press in ATRAS.
function ATRAS_Callback(hObject, eventdata, handles)
% hObject    handle to ATRAS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
StopMotor('all','off');% comando que detiene el motor A
SetMotor(MOTOR_A);% se inicializa motor A
SetPower(-50);%se configura el motor a 50% en forma inversa
SetAngleLimit(0);%se configura los grados del movimiento
SetTurnRatio(0);%realiza sincronismo entre los motores
SendMotorSettings();%envia los datos configurados al motor



% --- Executes on button press in DERECHA.
function DERECHA_Callback(hObject, eventdata, handles)
% hObject    handle to DERECHA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
StopMotor('all','off');% comando que detiene el motor B
SetMotor(MOTOR_B);% se inicializa motor B
SetPower(100);%se configura el motor a 100%
SetAngleLimit(15);%configura a 15 grados giro motor derecha
SetTurnRatio(0);%realiza sincronismo entre los motores
SendMotorSettings();%envia los datos configurados al motor
StopMotor(MOTOR_B,'off');% se detiene el motor B


% --- Executes on button press in IZQUIERDA.
function IZQUIERDA_Callback(hObject, eventdata, handles)
% hObject    handle to IZQUIERDA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
StopMotor('all','off');% comando que detiene el motor B
SetMotor(MOTOR_B);% se inicializa motor B
SetPower(-100);%se configura el motor a 100% en forma inversa
SetAngleLimit(15);%configura a 15 grados giro motor izquierda
SetTurnRatio(0);%realiza sincronismo entre los motores
SendMotorSettings();%envia los datos configurados al motor
StopMotor(MOTOR_B,'off');% se detiene el motor B


% --- Executes on key press with focus on ADELANTE and none of its
controls.
function ADELANTE_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to ADELANTE (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%   Key: name of the key that was pressed, in lower case
```

```
%   Character: character interpretation of the key(s) that was
pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift)
pressed
% handles    structure with handles and user data (see GUIDATA)


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
ADELANTE.
function ADELANTE_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to ADELANTE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in freno.
function freno_Callback(hObject, eventdata, handles)
% hObject    handle to freno (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
StopMotor(MOTOR_A,'off');%detiene motor A


% --- Executes on button press in blueton.
function blueton_Callback(hObject, eventdata, handles)
% hObject    handle to blueton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handle = COM_OpenNXT('bluetooth.ini','check');%abre comunicacion
BLUETOOTH
COM_SetDefaultNXT(handle);%configura el puerto COM bluetooth


% --- Executes on button press in bluetoff.
function bluetoff_Callback(hObject, eventdata, handles)
% hObject    handle to bluetoff (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
COM_CloseNXT('all');% cierra comunicacion bluetooth



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

BIBLIOGRAFIA

1. BRINCK GUSTAVO, Guía de robótica Lego Mindstorms NXT V2,
2. BARRAGAN, DIEGO. Manual de interfaz gráfica de usuario en Matlab,
3. BARRAGAN, DIEGO. Manual de interfaz gráfica de usuario en Matlab Parte 1,
4. DE GARCIA DE JALON, J. RODRIGUEZ,J Y VIDAL, J, APRENDA MATLAB 7.0 Como si estuviera en primero, Universidad Politécnica de Madrid
5. GARCIA PEREZ, DAVID. Manejo Básico de Imágenes con Matlab, Grupo de Vision Artificial,
6. BARRAGAN, DIEGO. GUIDE DE MATLAB -- USO DE 'HANDLES' EN UNA GUI, Video tutorial de uso de GUI.
7. Matlab & Simulink , MATLAB GUI- Creating A Graphical User Interface with MATLAB, Video tutorial de uso de Matlab & Simulink.
8. YORKE, R. Electric Circuit Theory. Editorial Pergamon Press, 1986.
9. THOMAS, R.E. Circuitos y Señales. Editorial Reverté, 1991.
10. JOHNSON, D.E. Análisis Básico de Circuitos Eléctricos. Editorial Prentice Hall Hispanoamericana, 1996.

11. SANJURJO, R. Y DE MIGUEL, P. Teoría de Circuitos Eléctricos. Editorial McGraw-Hill, 1997.
12. CARLSON, A. Bruce. Teoría de Circuitos. Editorial Thompson, 2002.
13. MATTHIAS SCHOLZ, PAUL. Advanced NXT - The Da Vinci Inventions Book, Apress 2007.
14. BENEDETTELLI, DANIELLE. Creating Cool Mindstorms NXT Robots, Apress 2008.
15. FLOYD KELLY, JAMES. Lego Mindstorms NXT G Programming Guide, Apress 2007.
16. FLOYD KELLY, JAMES. Lego Mindstorms NXT The Mayan Adventure, Apress 2006.
17. GASPERI M, HURBAIN P Y HURBAIN I. Extreme Nxt Extending the lego mindstorms nxt to the next level technology in action, Apress 2007.