UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Megaproyecto RAEC Robot Cartero

Trabajo de graduación presentado por los siguiente estudiantes Alejandro Asensio Lerma, Francisco Javier Mesalles Salcedo para optar al grado académico de Licenciado en Ingeniería Electrónica y Juan Carlos Celada Solares para optar al grado académico de Licenciado en Ingeniería Industrial

Guatemala

2011

Megaproyecto RAEC

Robot Cartero

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Megaproyecto RAEC Robot Cartero

Trabajo de graduación presentado por los siguiente estudiantes Alejandro Asensio Lerma, Francisco Javier Mesalles Salcedo para optar al grado académico de Licenciado en Ingeniería Electrónica y Juan Carlos Celada Solares para optar al grado académico de Licenciado en Ingeniería Industrial

Guatemala

2011

Vo.Bo.:

(f)_____

MAEB. Otto Girón

Tribunal:

(f)_____

MSc. Carlos Esquit

(f)_____

MAEB. Roberto Delgado

(f)_____

MAEB. Otto Girón

Fecha de Aprobación: 29-Nov-2011

PREFACIO

RAEC (Robot Autónomo de Entrega de Correspondencia) surgió de la inquietud de varios compañeros de hacer un proyecto de trabajo de graduación que llamara la atención. Algo que fuera un poco distinto a lo que se ha realizado en los últimos años. Antes de decidirnos a implementar este proyecto se barajearon varias ideas. No nos poníamos de acuerdo sobre qué proyecto implementar. Después de varios días de discusión y analizar un sin fin de ideas, todos coincidimos en implementar RAEC.

Se desea agradecer a Ing. Otto Girón e Ing. Roberto Delgado por todo el tiempo y ayuda dedicada, así como también al departamento de Ingeniería Mecatrónica por el financiamiento de este proyecto.

ÍNDICE

Página

LISTA DE CUADROS

LISTA DE GRÁFICOS

RESUMEN

RAEC (Robot Autónomo para Entrega de Correspondencia) es un robot autónomo que nace con el objetivo de entregar correspondencia dentro del campus de la Universidad del Valle de Guatemala. Este servicio se pretende brindar a todas las oficinas y salones del tercer nivel de los edificios G, H, I y J de la Universidad del Valle de Guatemala. El envío de paquetes o correspondencia con el robot se manejará por medio de una página web.

En esta página web las personas pueden solicitar los servicios del robot. Los usuarios deben especificar entre qué oficinas desea enviar la correspondencia. De una forma autónoma el robot llegará a la oficina de la cual se quiere enviar el paquete para posteriormente llevarlo a la oficina de destino. El robot contará con una interfaz humana, que consta de una pantalla táctil y alarma sonora, que permitirá a los usuarios introducir la correspondencia en el robot y saber cuando el robot deje la correspondencia.

I. INTRODUCCIÓN

En la historia reciente se han creado robots para sustituir a los humanos en diversas tareas que estos realizan. Los robots permiten realizar tareas repetitivas con igual o mayor precisión que los humanos y sin sufrir las consecuencias derivadas del cansancio y del tedio.

Por estas características surge la idea de crear un robot capaz de movilizarse por sí solo entre las oficinas del campus central de la Universidad del Valle para entregar la correspondencia y la papelería que actualmente deben entregar los empleados. Debido a la gran cantidad de correspondencia que se maneja dentro de la Universidad existe una demanda real para este servicio, además en su construcción se ponen en práctica los conocimientos que han adquirido a lo largo de su carrera los estudiantes de Ingeniería Mecatrónica, Ingeniería Electrónica e Ingeniería Industrial.

Se logró construir un robot capaz de viajar entre oficinas orientado por los sensores que posee, además es capaz de evitar obstáculos debido al algoritmo de navegación implementado. El robot tiene un tiempo de autonomía de dos horas con cuarenta y cinco minutos y tiene capacidad de transportar suficientes paquetes dentro de sus compartimentos.

Se determinó que el proyecto genera una Tasa Interna de Retorno de 29% con uno horizonte de cinco años, si logra sustituir veinte viajes diarios para entrega de correspondencia en el campus central de la Universidad del Valle de Guatemala. La rentabilidad del proyecto aumenta a medida que aumenta el número de viajes realizados.

II. OBJETIVOS

A. Objetivo general del megaproyecto

- Construir un robot autónomo capaz de recibir y entregar la correspondencia física del campus central de la Universidad del Valle de Guatemala.

B. Objetivos específicos

1. Módulo administrativo

- Gestionar el capital, los suministros y el tiempo para coordinar la construcción de un robot autónomo para entrega de correspondencia en el tiempo establecido, y realizar el análisis de costo y beneficio que obtendría la Universidad del Valle de Guatemala al contar con el robot en operación dentro del campus central.

2. Módulo de navegación e implementación del algoritmo de navegación

- Diseñar e implementar un circuito electrónico en el cual se interconecten todos los dispositivos necesarios para recolectar datos de distancia, orientación y ubicación del robot RAEC, y poner estos datos a disposición del algoritmo de navegación.

- Realizar un programa en lenguaje ensamblador que por medio de un microcontrolador implemente el algoritmo de navegación.

3. Módulo de diseño y simulación del algoritmo de navegación, interfaz por computadora y comunicación inalámbrica

- Implementar un sistema que permita establecer una comunicación inalámbrica bidireccional entre el servidor y el robot, con la finalidad de controlar y monitorear el estado de este último.

- Desarrollar una página web que permita a los usuarios utilizar los servicios de envío de correspondencia del robot, así como también permita ver el estado de los envíos.

- Desarrollar una aplicación de computadora que permita simular el algoritmo de navegación a ser utilizado por el robot con la finalidad de garantizar que el algoritmo utilizado permitirá al robot llegar a su destino.

III. MARCO TEÓRICO

A. Definición de robot

Etimológicamente la palabra robot es un anglicismo de la palabra "robot" que a su vez se deriva de la palabra checa "robota" que significa trabajo o labor. Un robot puede definirse como un agente virtual o electromecánico que parece tener un propósito propio y muestra algunos indicios inteligencia propia. Generalmente la palabra robot es más utilizada para referirse a agentes de tipo electromecánicos mientras que los agentes virtuales son llamados "bots". Los robots históricamente han sido creados para imitar tareas que realizan los seres humanos. En la actualidad existe una amplia variedad de robots comerciales e industriales que realizan tareas de forma más precisa que los seres humanos [17].

La ventaja que ofrecen los robots es que pueden llevar a cabo tareas de forma repetitiva sin fatigarse, lo que permite realizar trabajos con menor margen de errores, lo que a su vez representa una reducción de costos en la producción.

1. Tipos de robots. Existe una gran cantidad de clasificaciones para los robots, las cuales los agrupan según distintas características. Entre las más importantes están:

Clasificación de robots por grado de autonomía

- Robots autónomos: No requieren de ningún tipo de control humano para operar. Tienen precepciones sobre el entrono que los rodea y pueden tomar decisiones con base a estas percepciones.

- Robots teledirigidos: Son controlados a distancia por medio de ondas electromagnéticas.
- Robots de teledirección asistida: Son un híbrido entre los dos anteriores [18].

Por su arquitectura los robots pueden ser clasificados en cuatro grupos:

- Androides: Tienen forma humana.
- Móviles: Se desplazan sobre ruedas.
- Zoomórficos: Imitan la forma de moverse de los animales.
- Poli articulados: Tienen un rango de movimiento limitado, generalmente usados para tareas específicas en la industria [18].

2. Robot autónomo para entrega de correspondencia. Es un robot que transporta un paquete de un lugar a otro, sin ningún tipo de control humano aparte de las instrucciones de dónde recibir y entregar el paquete. Es un mecanismo que puede recorrer por sí mismo la distancia de un punto a otro, y por la información que recibe del entorno tiene la capacidad de evitar obstáculos y decidir la ruta que debe seguir.

B. Administración de proyectos

Un proyecto es una serie de actividades relacionadas que tiene como fin lograr uno o varios objetivos específicos. Por lo que se define la gestión o administración de proyectos es una serie de procesos que incluyen la planificación, la obtención y la utilización de los recursos con el fin de realizar un proyecto que cumpla los objetivos propuestos, dentro de los límites de tiempo y costos definidos.

1. Etapas de un proyecto. En cualquier proyecto se pueden identificar las siguientes cuatro etapas:

- La idea del proyecto: Consiste en identificar una oportunidad por la cual se puede realizar el proyecto.
- El diseño: En esta etapa se establecen las estrategias a seguir para lograr los objetivos del proyecto.
- Ejecución: Realización del proyecto con base en cómo fue diseñado.
- Evaluación: Revisión final del proyecto para establecer si se lograron los objetivos planteados.

C. Análisis de costo/beneficio

Este análisis consiste en contabilizar todos los costos y beneficios esperados de un proyecto o de cualquier actividad. Al realizarlo se puede estimar el impacto financiero de llevar a cabo el proyecto. Es utilizado como una herramienta para la toma de decisiones para realizar un nuevo proyecto [8].

El análisis de costo/beneficio se divide en seis pasos que son:

- Determinar todos los factores relativos al proyecto
- Contabilizar todos los costos asociados a cada uno de los factores
- Obtener el costo total del proyecto
- Contabilizar todos los beneficios asociados a cada uno de los factores
- Obtener el beneficio total del proyecto
- Obtener la relación entre el beneficio y el costo del proyecto

1. Clasificación de costos y beneficios. Para realizar el análisis costo beneficio primero se deben cuantificar los beneficios y contra-beneficios para el usuario del producto. También se deben considerar los efectos secundarios [8].

Posteriormente se deben clasificar los gastos y ahorros que obtendrá el patrocinador del proyecto. Deben incluirse los costos de realización del proyecto y los costos operativos. Los ingreses que se espera recibir reducen los costos del patrocinador.

2. Razón costo/beneficio. Luego que se tiene la suma total de costos y beneficios del proyecto se calcula la relación costo/beneficio. Si esta relación es mayor a uno se acepta el proyecto porque significa que por cada unidad monetaria invertida se obtiene más de una unidad monetaria de retorno. Si esto no se cumple el proyecto no debe ser aceptado[8].

El análisis costo/beneficio se basa en la obtención de los mayores beneficios con los menores costos por lo que se puede considerar análogo a la medida de eficiencia. Se supone que si los beneficios son mayores a los costos el proyecto es considerado exitoso.

D. Sensores usados en el proyecto

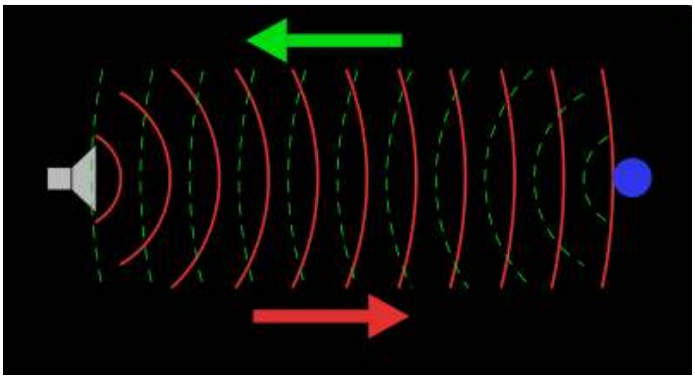
Para que el robot se desplace desde su origen hacia su destino de forma exitosa es necesario saber la distancia a la que se encuentran los objetos que están en el entorno que se está moviendo el robot, también es necesario tener conocimiento de la orientación del robot para que el algoritmo de navegación pueda hacer los ajustes necesarios en su dirección para llegar a su destino.

Para medir las distancias con los objetos se utilizaron sensores ultrasónicos, para medir la orientación del robot se usó un sensor de campo magnético o brújula.

1. Sensores ultrasónicos. Los sensores ultrasónicos fueron usados en el proyecto como sensores de distancia. Estos sensores son necesarios para que el robot reconozca la distancia a la que se encuentra con los objetos a su alrededor en un radio específico. Una de las principales aplicaciones de los sensores ultrasónicos se da en la industria. Estos son usados como sensores de nivel. A través de ellos se puede medir la cantidad de materia prima que hay en un tanque. Sin embargo también son usados en otras aplicaciones. En este caso el sensor ultrasónico es usado como un sensor de distancia.

El sensor ultrasónico funciona de la siguiente manera. Éste genera y envía una onda mecánica (ultrasónica) al espacio libre. La onda sale del sensor y es enviada al aire libre. Al estar en el aire libre la onda viaja hasta encontrarse con algún objeto. Cuando ésta se encuentra con un objeto, éste choca con él, la colisión entre la onda mecánica y el objeto hace que refleje parte de la onda hacia su lugar de origen. Los sensores ultrasónicos funcionan midiendo el tiempo que le toma a la onda desde que sale del sensor hasta que choca con algún objeto y regresa a él nuevamente. Con este tiempo se puede calcular la distancia que viajo la onda, he ahí su aplicación como sensor de distancia o sensor de nivel [7].

Figura 1. Comportamiento de una onda mecánica generada por un sensor ultrasónico. [7]



2. Sensor de campo magnético. El sensor de campo magnético juega un papel fundamental en el proyecto. Por medio de este sensor es que se puede medir el ángulo de orientación al cual se encuentra el robot. Actualmente existen varios tipos de sensores de campo magnético. Muchos de ellos están orientados a detectar o no la presencia de objetos metálicos, otros están orientados a detectar la intensidad del campo magnético y otros están diseñados para detectar la orientación del campo magnético.

Entre los elementos transductores que se encuentran en este tipo de sensores son magnetoresistores. Los magnetoresistores están formados por tiras. Estas tiras están hechas con materiales de características magnéticas por medio de fotolitografía. La resistencia de las tiras cambia en presencia de un campo magnético. La resistencia depende del ángulo de magnetización [9].

E. Protocolos RS-232 e I2C

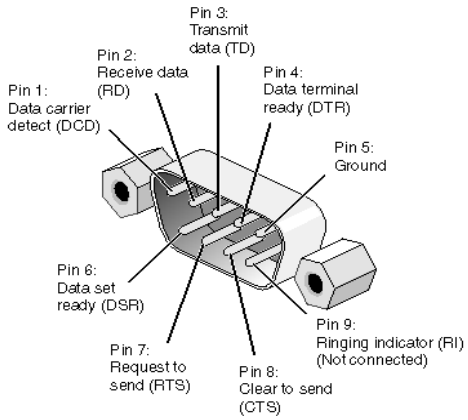
Para el desarrollo del proyecto fue necesario utilizar varios protocolos de comunicación para intercambiar información con los distintos sensores, con otros microcontroladores, con los Xbee o con el dispositivo de GPS.

1. Protocolo RS-232. El protocolo RS-232 es una forma de comunicación serial y fue el más utilizado en el proyecto. Este fue usado para comunicarse con el dispositivo GPS, con los Xbee y entre microcontroladores.

Características [20]

- 9 pines de señal.
- Conector de DTE debe ser macho y el conector de DCE hembra.
- Los voltajes para un nivel lógico alto están entre -3V y -15V, y un nivel bajo +3V y +15V.
- Los voltajes más usados son +12V/-12V, +9V/–9V
- Dependiendo de la velocidad de transmisión empleada, es posible tener cables de hasta 15 metros.
- Velocidad: 300, 600, 1200, 2400, 4800 y 9600 bps

Figura 2. Diagrama de pines de un conector DB9 macho. [2]



Las señales TXD, DTR y RTS son de salida, mientras que RXD, DSR, CTS y DCD son de entrada. La masa de referencia para todas las señales es SG (Tierra de Señal).

2. Protocolo I^2C. En este trabajo el protocolo I^2C fue usado para comunicarse con el sensor de campo magnético o brújula. I^2C es un protocolo de comunicación que comúnmente se utiliza para comunicarse con memorias, microcontroladores y otros dispositivos con cierto nivel de "inteligencia". Fue diseñado por Philips y permite el intercambio de información entre muchos dispositivos a una velocidad de 100 Kbits por segundo.

El protocolo I^2C es un tipo comunicación serial, síncrona y del tipo maestro – esclavo. I^2C utiliza únicamente utiliza 2 pines. Uno se utiliza para transferencia de datos y el otro para reloj.

El bus I2C, un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", sólo requiere de dos líneas de señal y un común. Fue diseñado a este efecto por Philips y permite el intercambio de información entre muchos dispositivos a una velocidad aceptable, de 100 Kbits por segundo, aunque hay casos especiales en los que el reloj llega hasta los 3,4 MHz. La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos [15].

Descripción de las señales

- SCL (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.

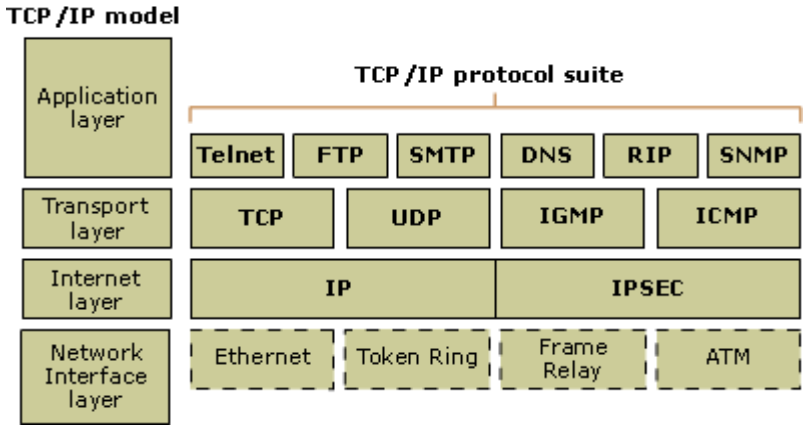
- SDA (System Data) es la línea por la que se envían los datos entre los dispositivos.
- GND común de la interconexión entre todos los dispositivos.

Las líneas SDA y SCL son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (o FET). Se deben polarizar en estado alto (conectando a la alimentación por medio de resistores "pull-up") lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas [1].

F. Arquitectura del protocolo TCP/IP

Esta arquitectura es el resultado de la investigación y el desarrollo que fueron realizados en la red de conmutación de paquetes *ARPANET*. Es conocida globalmente como familia de protocolos TCP/IP y fue financiada por la agencia estadounidense *DARPA* (Agencia de Proyectos de Investigación Avanzada para la Defensa). La familia está formada por una amplia colección de protocolos como se muestra en la Figura 3, los cuales se han especificado como estándares de internet por parte del *IAB* (Junta de Arquitectura de Internet). [12]

Figura 3. Arquitectura TCP/IP [12]



1. Protocolo Ethernet. El término Ethernet se refiere a una familia de productos LAN (Red de Área Local) cubiertos por el estándar IEEE 802.3, el cual define lo que es comúnmente conocido como el protocolo CSMA/CD. [3]

Un sistema Ethernet consiste de tres elementos:

- Un medio físico para transportar señales entre los elementos. [16]
- Un conjunto de normas de acceso al medio. [16]
- Una trama de bits utilizada para transportar datos a través del sistema. [16]

G. Comunicaciones de radiofrecuencia

Las comunicaciones de radiofrecuencia (RF) se basan en las leyes de la física que describen el comportamiento de las ondas de energía electromagnética. Su funcionamiento está basado en una fuente que puede crear una onda electromagnética y en un receptor que puede recoger dicha onda y decodificarla para de esta manera obtener el mensaje transmitido. [5]

Como se muestra en la Tabla 1, mayores frecuencias generan longitudes de onda de menor tamaño y, en términos generales, las señales con longitudes de onda más grande recorren mayores distancias, penetran y rodean objetos con mayor facilidad que aquellas señales con longitudes de onda pequeñas. [5]

Tabla 1. Bandas de radiofrecuencia [13]

F	λ	Banda	Descripción
30-300 Hz	10^4-10^3 km	ELF	Frecuencias extremadamente bajas
300-3000 Hz	10^3-10^2 km	VF	Frecuencias de voz

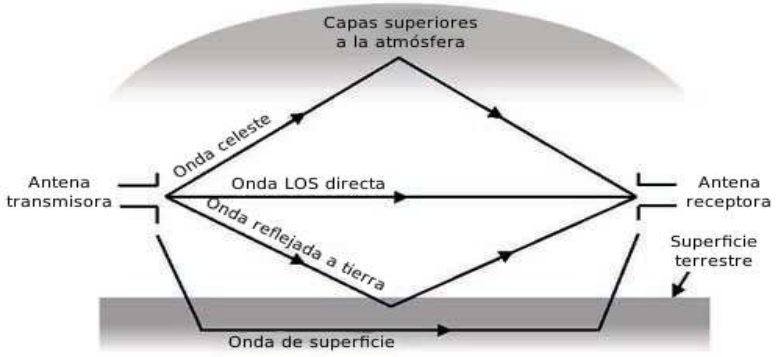
Continuación Tabla 1

F	λ	Banda	Descripción
3-30 kHz	100-10 km	VLF	Frecuencias muy bajas
30-300 kHz	10-1 km	LF	Frecuencias bajas
0.3-3 MHz	1-0.1 km	MF	Frecuencias medias
3-30 MHz	100-10 m	HF	Frecuencias altas
30-300 MHz	10-1 m	VHF	Frecuencias muy altas
300-3000 MHz	100-10 cm	UHF	Frecuencias utra altas
3-30 GHz	10-1 cm	SHF	Frecuencias súper altas
30-300 GHz	10-1 mm	EHF	Frecuencias extremadamente altas

H. Propagación de ondas

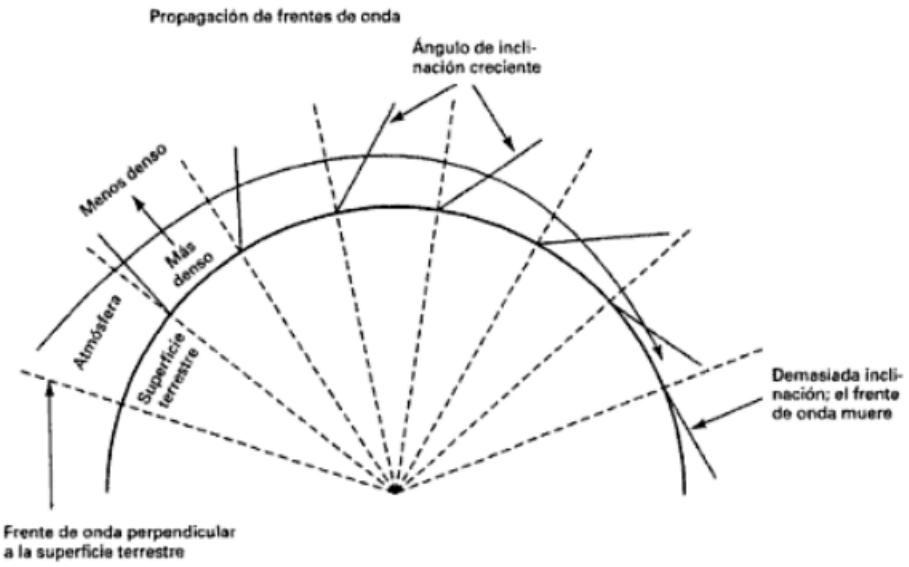
Las ondas de radiofrecuencia se pueden propagar de distintas formas como se muestra en la Figura 4, dependiendo del tipo de sistema y ambiente. Existen tres formas en las que se pueden propagar las ondas electromagnéticas en un sistema de comunicación de radiofrecuencia: ondas de tierra, ondas espaciales y ondas de cielo. [19]

Figura 4. Propagación de ondas [19]



1. Ondas de Tierra. Este tipo de ondas, también llamadas ondas superficiales, viajan por la superficie de la Tierra como se muestra en la Figura 5. Su funcionamiento se basa en un campo eléctrico variante que induce voltajes en la superficie de la tierra, dichos voltajes generan un flujo de corriente muy similar al de una línea de transmisión. Las ondas de Tierra se atenúan conforme se propagan y las pérdidas incrementan rápidamente conforme incrementa la frecuencia, por esa razón la propagación de ondas de tierra está limitada a frecuencias menores a 2MHz. [19]

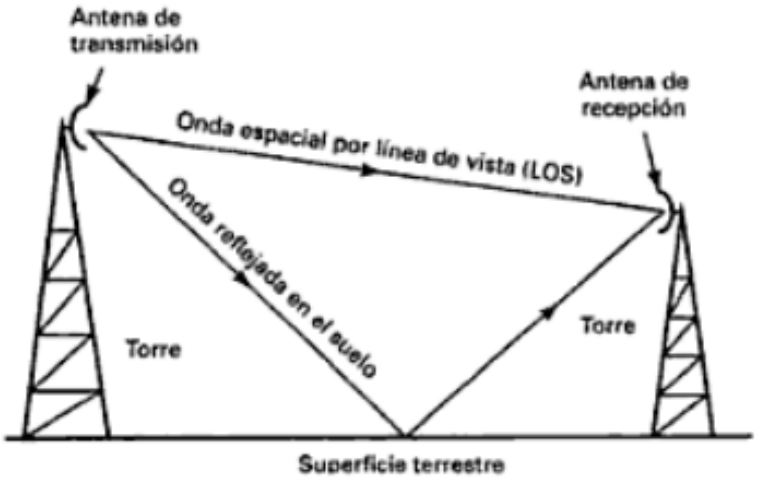
Figura 5. Propagación de ondas de Tierra [19]



2. Ondas espaciales. Este tipo de propagación incluye energía radiada que viaja unas cuantas millas en la parte inferior de la atmósfera de la Tierra, ya sea de forma directa o reflejada en la tierra como se muestra en la Figura 6. Las ondas directas viajan en línea recta entre las antenas transmisora y receptora, este tipo de propagación se conoce como transmisión de línea de vista (LOS), por lo que su propagación está limitada por la curvatura de la tierra. Las ondas

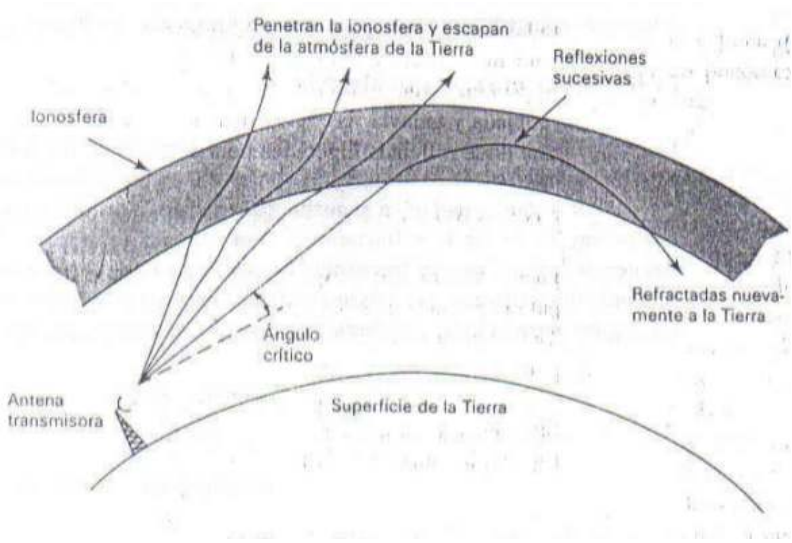
reflejadas rebotan contra la superficie de la Tierra a lo largo de la trayectoria entre la antena transmisora y la receptora. [19]

Figura 6. Propagación de ondas espaciales [19]



3. Ondas de cielo. Este tipo de ondas se envían hacia el cielo, en donde son reflejadas o refractadas a la Tierra por la ionosfera como se muestra en la Figura 7. Estas ondas se irradian en una dirección que produce un ángulo relativamente grande respecto a la superficie de la Tierra, lo cual produce que la onda choque contra la ionosfera, la cual es la porción más alta de la atmósfera y absorbe grandes cantidades de energía radiante del sol, ionizando así las moléculas de aire y creando electrones libres. De esta manera, cuando una onda electromagnética choca contra la ionósfera se va doblando para alejarse de las zonas de mayor densidad de electrones logrando así que la onda refractada regrese a la Tierra. [19]

Figura 7. Propagación de ondas de cielo [19]



I. GPS

GPS significa Global Positioning System. Es un sistema de orientación y navegación que utiliza un conjunto de 24 satélites que orbitan alrededor de la Tierra a una distancia de 20,000 km. Este conjunto de satélites es conocido como Navstar. Debido a la distancia a la que se encuentran de la Tierra y a la velocidad que se mueven, cada satélite da una vuelta al planeta cada doce horas. Sus trayectorias han sido calculadas para que en todo momento haya por lo menos 5 satélites en cualquier lugar del planeta.

Los datos de los GPS son transmitidos por radio frecuencia y emiten continuamente dos códigos de datos distintos. Uno de los datos está reservado exclusivamente para uso militar el otro para uso civil. En los códigos de datos para civiles se transmiten dos tipos de datos que son conocidos como almanaque y efemérides. Estos datos informan sobre el estado operativo de funcionamiento del satélite, su órbita, la fecha y la hora. Cuando un dispositivo receptor GPS capta señal de por lo menos tres satélites de GPS, por triangulación el receptor es capaz de determinar la posición en la que se encuentra sobre la superficie de

la tierra. Al obtener señal de cuatro o más satélites el receptor GPS puede determinar además de la posición la altura a la que se encuentra. [4]

Los satélites de GPS utilizan dos frecuencias distintas para comunicarse con los dispositivos receptores en la tierra. Una frecuencia es 1,227.6 MHz y se utiliza para códigos militares. Para códigos civiles utiliza la frecuencia de 1,575.42 MHz.

El GPS es uno de los más sencillos y precisos sistemas de navegación con el que se dispone en la actualidad. Las señales de GPS se atenúan fácilmente cuando traspasan objetos. Los únicos objetos que traspasan con facilidad son el cristal y el plástico. Cualquier otro tipo de material atenua la señal del GPS a modo que los datos ya no se puedan procesar en el receptor provocando que el aparato no funcione. Es un requisito que la antena con la que se desea captar la señal de los satélites tenga línea de vista hacia el cielo. [4]

IV. ANTECEDENTES

En agosto del 2004 el Doctor Jinsuck Kim presentó su trabajo de graduación titulado *A FRAMEWORK FOR ROADMAP-BASED NAVIGATION AND SECTOR-BASED LOCALIZATION OF MOBILE ROBOTS* para optar al título de *DOCTOR OF PHILOSOPHY* en la Universidad de Texas A&M. [10]

En dicho trabajo describe un método de navegación basado en rutas dentro de un mapa, el cual es adecuado para aplicarse en ambientes de interior parcialmente conocidos, utilizando únicamente sensores de distancia económicos. El navegador selecciona rutas dentro de un mapa y selecciona puntos de localización en las rutas elegidas. El navegador selecciona rutas que sean factibles a las necesidades de la aplicación, como curvas con esquinas no puntiagudas, y a las necesidades del algoritmo de localización. [10]

El doctor Kim presentó un localizador basado en sectores el cual es robusto teniendo en cuenta las limitantes de los sensores y la presencia de objetos desconocidos, manteniendo la eficiencia computacional del navegador. [10]

Por otro lado, el Doctor Kristopher Kriechbaum presentó en 2006 su trabajo de graduación titulado *TOOL AND ALGORITHMS FOR MOBILE ROBOT NAVIGATION WITH UNCERTAIN LOCALIZATION* para optar al título de *DOCTOR OF PHILOSOPHY* en el California Institute of Technology. [11] Presentó nuevas herramientas y algoritmos utilizados en la localización y navegación de un robot móvil, introduciendo un nuevo método de escaneo de distancia el cual incorpora modelos realistas de ruido en sensores. [11]

Teniendo en cuenta los trabajos mencionados se decidió realizar la simulación del algoritmo de navegación tomando en cuenta el error no sólo en los sensores de distancia, sino que en la brújula y en el giro de los motores de nuestro robot.

V. DESARROLLO DEL MÓDULO ADMINISTRATIVO

A. PLANIFICACIÓN DEL DESARROLLO DEL PROYECTO

La planificación de cualquier proyecto es muy importante para poder tener una mejor idea de las tareas que debe realizar cada integrante del equipo y el tiempo en el que las debe realizar. Para la construcción del Robot Autónomo para Entrega de Correspondencia se dividió el trabajo en dos partes, la primera es la fase de investigación de los componentes que iba a utilizar cada módulo y la segunda fase es la construcción del robot.

Para la planificación se utilizó un diagrama de Gantt interactivo en una página web para que los miembros del proyecto pudieran ver a cualquier hora las tareas que debían realizar y el tiempo que tenían disponible. De esta forma se pudo centralizar el control de los avances del proyecto en una plataforma que todos los miembros proyecto pudieran visualizar para dar seguimiento a sus tareas y a sus avances. La plataforma que se utilizó fue el sitio web Viewpath® el cual permite abrir una cuenta gratuita con la cual se puede crear la programación de un proyecto y asignar recursos a cada una de las tareas. En este caso los recursos eran las personas encargadas de cada módulo y el programa permitía ingresar el correo electrónico de cada persona para notificarle las tareas que tenía asignadas y cuando las debía realizar. También permitía enviar un recordatorio cuando las tareas ya estaban retrasadas. Esto permitió dar un seguimiento más preciso al proceso de construcción el robot.

La programación realizada en Viewpath® permite calcular el porcentaje completado del proyecto con base en las tareas completadas de cada módulo, de esta forma se visualiza mejor que tan retrasado va el proyecto respecto a la

fecha esperada de finalización. Se puede ver también el porcentaje que han completado cada uno de los módulos por separado (ver anexo # 1 programación de la fase de construcción).

En el diagrama de Gantt de la fase de construcción se observa que la ruta critica pasa por el módulo de potencia y luego por el ensamble final de todos los módulos. Esto indica que el módulo de potencia no se puede retrasar más de la fecha esperada porque si no se retrasa todo el proyecto. El resto de módulos tienen cierta holgura que les permite retrasarse un poco sin afectar el tiempo total. La estructura del robot es el módulo con menor holgura siendo ésta de una semana. Los módulos de comunicación y de navegación dependen de la realización anterior de la interfaz por computadora y modulo de sensores respectivamente. Por lo que un retraso en los módulos anteriores significa un retraso en los módulos posteriores.

1. Comparación entre programación y tiempo real del proyecto.

La programación del proyecto fue acertada en las fases iniciales de la construcción separada de cada uno de los módulos, se cumplieron los tiempos establecidos para los módulos de estructura interna, interfaz con la computadora y de motores. Existió un retraso en los módulos de sensores y de posicionamiento GPS porque ambos módulos fueron realizados por una sola persona y la carga de trabajo fue mayor. Sin embargo estos módulos estaban dentro de la ruta crítica del proyecto lo que retrasó la integración de todos los módulos y las pruebas finales del robot. Por estos cambios la integración de todos los módulos se retrasó dos semanas más de lo previsto.

B. COSTO DEL PROYECTO

Se contabilizó el costo real del proyecto de acuerdo a los materiales que compró cada uno de los miembros del equipo para la realización de los módulos. Los materiales adquiridos se dividen en dos grupos: los materiales importados y los materiales comprados localmente. Actualmente en Guatemala no existen distribuidores que cuenten con los componentes electrónicos necesarios para el proyecto por lo que estos se compraron por internet a distribuidores en Estados Unidos.

Cuando se importan componentes electrónicos a Guatemala se debe pagar un arancel del 15% y el impuesto del valor agregado de 12% para que puedan ingresar al país, sin embargo debido a que fueron comprados por medio de la Universidad del Valle se tiene la excepción del pago de estos impuestos debido a que no es una entidad con fines de lucro y los componentes son para fines didácticos (según artículo quinto del acuerdo 107-2002, padrón de importadores de la Superintendencia de Administración Tributaria). Por lo tanto el único gasto adicional de importar los componentes son los gastos de envío, que se dividen en gastos de envío dentro de los Estados Unidos y los gastos de la empresa que los trae a Guatemala. Todas las compras de materiales fueron realizadas por el departamento de compras de la Universidad del Valle de Guatemala. La tasa de cambio utilizada fue de Q. 7.83 por cada US. $ 1.00.

Los únicos materiales adquiridos en Guatemala fueron los de la estructura del robot y las partes utilizadas para ensamblar el robot, como son los tornillos, tuercas y componentes de soldadura. Dentro de los costos no se toma en cuenta la mano de obra porque el robot fue construido por estudiantes como trabajo de graduación, pero si el proyecto se llevara a cabo para fines de lucro la mano de obra sería sin duda un importante costo para el robot.

Se observa que los costos reales del proyecto son bastante similares al presupuesto realizado antes de la realización del proyecto, porque las piezas compradas son las mismas que se contemplaron en el presupuesto. Además algunas piezas adicionales que se compraron se compensan con el porcentaje dedicado a imprevistos en el presupuesto del proyecto. Los gastos de importación estuvieron correctos y fueron bastante cercanos a lo que se tenía en el presupuesto.

Tabla 2. Costo del proyecto

	Cantidad	Costo ($.)	Total ($.)	Costo (Q.)	Total (Q.)
Materiales importados					
Transceivers	20	$ 0.65	$ 13.00	Q 5.09	Q 101.79
Sensores de distancia ultrasonicos	8	$ 29.95	$ 239.60	Q 234.51	Q 1,876.07
Sensor de campo magnético	1	$ 33.59	$ 33.59	Q 263.01	Q 263.01
Chip GPS Fastrax UC322	1	$ 50.00	$ 50.00	Q 391.50	Q 391.50
DS PIC	3	$ 5.70	$ 17.10	Q 44.63	Q 133.89
Puente H HB-25	2	$ 50.00	$ 100.00	Q 391.50	Q 783.00
Kit motores Parallax	1	$ 280.00	$ 280.00	Q 2,192.40	Q 2,192.40
Batería Tysonic de 12V y 25 Ah	1	$ 90.00	$ 90.00	Q 704.70	Q 704.70
Convertidor protocolo Ethernet	1	$ 79.95	$ 79.95	Q 626.01	Q 626.01
Transceiver inalámbrico	1	$ 149.00	$ 149.00	Q 1,166.67	Q 1,166.67
Adaptador de voltaje	2	$ 14.99	$ 29.98	Q 117.37	Q 234.74
Pantalla Touch	1	$ 23.00	$ 23.00	Q 180.09	Q 180.09
Total materiales importados		$ -	$ 1,105.22	Q -	Q 8,653.87
Materiales locales					
Metros aluminio 1x1	12	$ 1.92	$ 22.99	Q 15.00	Q 180.00
placa aluminio 4x8 1/16"	1	$ 114.94	$ 114.94	Q 900.00	Q 900.00
Touch panel control proto	1	$ 24.27	$ 24.27	Q 190.00	Q 190.00
GLCD adapter board	1	$ 17.24	$ 17.24	Q 135.00	Q 135.00
Tornillos	200	$ 0.04	$ 7.66	Q 0.30	Q 60.00
Roldanas	200	$ 0.01	$ 2.55	Q 0.10	Q 20.00
Tuercas	200	$ 0.01	$ 2.55	Q 0.10	Q 20.00
Escuadras de 1.5"	70	$ 0.38	$ 26.82	Q 3.00	Q 210.00
Ruedas	2	$ 2.17	$ 4.34	Q 17.00	Q 34.00
Total materiales locales			$ 223.37	Q -	Q 1,749.00
Gastos de importación					
Transporte			$ 110.52		Q 865.39
Costo total del proyecto			$ 1,439.11		Q 11,268.26

C. ANÁLISIS DE COSTO BENEFICIO

Se realizó un análisis del beneficio que significa para la Universidad del Valle contar con el Robot en funcionamiento. El beneficio para la Universidad es evitarse el costo generado por la pérdida de tiempo cuando los empleados entregan la correspondencia entre oficinas. Los costos del robot son la inversión inicial realizada para la construcción, el gasto por consumo de energía eléctrica que tiene para recargar su batería y el mantenimiento anual del robot.

1. Determinación del costo de entregar correspondencia entre las oficinas. Se realizó una encuesta a los empleados de la Universidad del Valle de Guatemala para estimar la cantidad de viajes que se realizan diariamente y la distancia promedio de los mismos. De acuerdo a las respuestas entre las oficinas de origen y las oficinas destino se calculó la distancia entre ellas por medio de un mapa a escala del campus central de la universidad. En total se tuvieron noventa y cinco resultados de viajes distintos para entrega de correspondencia. El promedio entre todos los viajes fue de ciento sesenta y cuatro metros.

Tabla 3. Estadística descriptiva de las distancias

Estadística descriptiva	
Media	164.40
Error estándar	8.59
Mediana	169.20
Moda	188.31
Desviación estándar	83.68
Varianza de la muestra	7,002.50
Kurtosis	0.01
Oblicuidad	0.07
Rango	356.23
Minimo	8.86
Máximo	365.09
Suma	15,618.36
Muestra	95.00

Para calcular el tiempo del viaje se utilizó la velocidad promedio a la que camina una persona que es de cinco kilómetros por hora. Se calculó que una persona tarda dos minutos para entregar la correspondencia en el lugar de destino. Con estos valores se calcula el tiempo total del viaje.

El tiempo que se toman los trabajadores para entregar la correspondencia tiene un costo para la universidad porque es tiempo perdido. Para el costo del tiempo se utilizó el escenario más pesimista con el salario mínimo para un trabajador no agrícola que es de Q. 63.70 el día de trabajo según Acuerdo Gubernativo 388-2010 del Ministerio de Trabajo y Previsión Social. Este valor se dividió dentro de ocho horas que se laboran diariamente en promedio lo que da como resultado un costo por hora de Q. 7.96.

Por último se estimó el número de viajes que reemplazaría el robot al día y con esto se obtiene el costo de oportunidad diario para la universidad de contar con un robot que realice la entrega de correspondencia. El tiempo de operación es la suma de todos los viajes y representa el tiempo que el robot estaría funcionando, sin embargo se sabe que el funcionamiento del robot será más lento por lo que este tiempo será mayor.

Tabla 4. Cálculo del costo de entregar correspondencia entre oficinas

Distancia promedio (metros)		328.00
Velocidad (m/s)		1.39
Tiempo entrega (s)		120
Tiempo total (h)		0.10
Salario	Q	7.96
Costo promedio de viaje	Q	0.79
Viajes diarios		20
Costo diario	Q	15.75
Costo mensual	Q	393.75
Tiempo operación		1.98

2. Determinación del costo de energía eléctrica y mantenimiento. El consumo de energía eléctrica depende de las veces que se va a recargar la batería diariamente y esto depende del tiempo de operación del robot. Con la batería completa el robot puede operar 2.45 horas, entonces el tiempo de operación esperado se divide por este factor para saber cuántas cargas diarias necesitará el robot. Para recargar la batería completamente se necesitan 0.3 kWh, si se carga diariamente una vez la batería se consumen 7.5 kWh al mes lo que tiene un costo mensual de Q. 27.33 (ver anexo # 3 detalle de consumo eléctrico), esto se multiplica por el factor encontrado con base en el tiempo de operación del robot para saber el costo de la energía eléctrica. Con veinte viajes diarios se cargará el 81% de la batería diariamente lo que equivale a un costo mensual de Q. 22.07. Se consideró también un costo de Q. 500.00 por mantenimiento del robot para sustituir partes que se puedan dañar por el uso.

3. Flujo de efectivo. Con los valores calculados se realizó un flujo de efectivo mensual para determinar el valor presente neto del proyecto. Se utilizó la tasa de interés líder actual del banco de Guatemala. Para el costo de energía eléctrica se utilizó un incremento anual del 3% y para el salario mínimo un incremento del 9% de acuerdo a los incrementos históricos de los últimos años (ver anexo 5 y 6). Las reparaciones se incrementan en 10% cada año por el deterioro de un mayor número de partes por mayor tiempo de uso. La Tasa Mínima Atractiva de Retorno elegida para el proyecto es del 15% la cual se basa en la tasa que pagan los bancos en una cuenta de ahorro a plazo fija actualmente en Guatemala y un porcentaje adicional por el trabajo involucrado en realizar un proyecto de este tipo.

Tabla 5. Cálculos realizados para el flujo de efectivo proyectado

Calculo de los valores del flujo de efectivo	
Inversión Inicial	Igual al costo del proyecto
Energía eléctrica	Costo por carga * (tiempo de operación / 2.45) * 12 meses
Mantenimiento	Estimado por el costo de repuestos
Beneficio	Viajes diarios * costo promedio de viajes * 25 días * 12 meses

Tabla 6. Flujo de efectivo proyectado

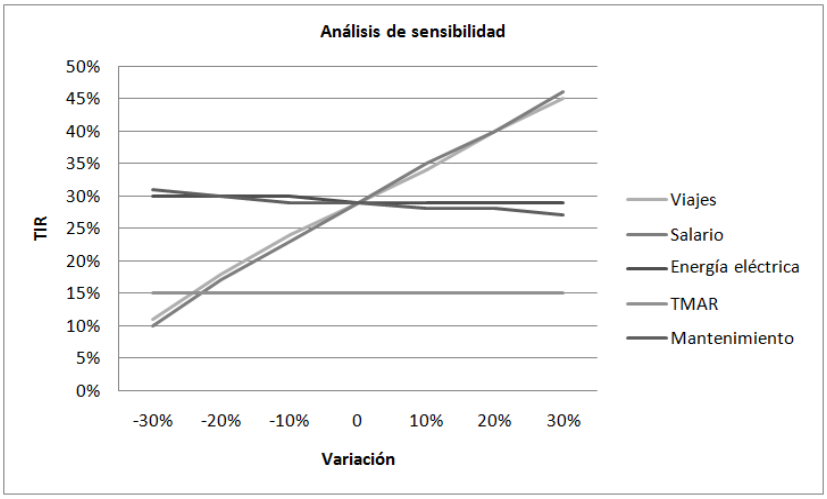
Flujo de efectivo	Año 0	Año 1	Año 2	Año 3	Año 4	Año 5
Inversion Inicial	Q (11,268.26)	Q -	Q -	Q -	Q -	Q -
Energia eléctrica	Q -	Q (264.85)	Q (278.09)	Q (291.99)	Q (306.59)	Q (321.92)
Mantenimiento	Q -	Q (500.00)	Q (550.00)	Q (605.00)	Q (665.50)	Q (732.05)
Beneficio	Q -	Q 4,275.06	Q 5,150.31	Q 5,613.84	Q 6,119.08	Q 6,669.80
Flujo Neto	Q (11,268.26)	Q 3,960.21	Q 4,322.22	Q 4,716.84	Q 5,146.99	Q 5,615.83
VPN	(Q 11,268.26)	(Q 7496.63)	(Q 3,576.25)	Q 498.34	Q 4,732.78	Q 9,132.93
TIR			-18%	7%	21%	29%
VPN Costos	Q (11,268.26)	Q(11,996.69)	Q (12,747.79)	Q (13,522.65)	Q (14,322.39)	Q (15,148.21)
VPN Beneficios	Q -	Q 4,500.05	Q 9,171.54	Q 14,020.98	Q 19,055.17	Q 24,281.13
Razon B/C	-	0.38	0.72	1.04	1.33	1.6

En la Tabla 6 se observa que la rentabilidad del proyecto depende del tiempo que esté en operación el robot. El Valor Presente Neto es positivo cuando el proyecto tiene un horizonte de por lo menos tres años, antes de esto el proyecto tiene más perdidas que beneficios. La Tasa Interna de Retorno en el tercer año es de siete por ciento lo cual es menor a la Tasa Mínima Atractiva de Retorno elegida. En los años siguientes de operación del robot la Tasa Interna de Retorno es mayor y el proyecto se hace más atractivo. Para el quinto año de operación del robot la tasa interna de retorno es de veintinueve por ciento. El gasto más importante de este proyecto es la inversión inicial, luego los demás gastos no son tan importantes comparados con el beneficio que representa, por esta característica es que la rentabilidad del proyecto depende del tiempo en que esté en funcionamiento.

En el análisis de la razón de costos y beneficios en valor presente se observa que en los primeros dos años de operación los costos son mayores a los beneficios por lo que se tienen pérdidas. En el tercer año de operación los beneficios superan a los costos, por cada quetzal invertido se obtienen sólo seis centavos como beneficio. Esta razón mejora en los siguientes años y en el quinto año los beneficios obtenidos son sesenta por ciento mayores a los costos.

4. Análisis de sensibilidad. Se realizó un análisis de sensibilidad para determinar cuáles son las variables que más influyen para la rentabilidad del proyecto. Se escogieron los viajes diarios realizados, el salario mínimo de los trabajadores y el costo de la energía eléctrica. Luego se hicieron cambios porcentuales de igual magnitud en las tres variables por separado y se determinó una nueva tasa interna de retorno con horizonte a cinco años en el flujo de efectivo para cada uno de los casos. Posteriormente se graficaron los resultados.

Gráfica 1. Análisis de sensibilidad



En la gráfica del análisis de sensibilidad se observa que la relación entre los viajes realizados es directamente proporcional a la Tasa Interna de Retorno. Es decir que cuando aumenta el número de viajes diarios realizados por el robot

también aumenta la Tasa Interna de Retorno. Esta misma relación se da con el salario mínimo, si aumenta el salario mínimo el proyecto se hace más rentable porque el beneficio que representa para la Universidad es mayor. Estas dos variables afectan al proyecto en la misma proporción.

Para que el proyecto sobrepase la tasa mínima de retorno con un horizonte de cinco años el robot debe realizar por lo menos quince viajes diariamente. Si realiza en promedio más viajes que estos el proyecto será atractivo. Con el mismo horizonte, el salario mínimo no debe disminuir más de 25% para que el proyecto sea rentable, pero esto es muy poco posible que suceda porque el salario mínimo ha aumentado todos los años.

Se observa que la variación del costo de energía eléctrica es inversa a la Tasa Interna de Retorno, es decir que si aumenta el costo de la energía eléctrica el retorno del proyecto disminuye. Esto se da porque el beneficio neto del proyecto disminuye, sin embargo el costo de la energía eléctrica debe aumentar demasiado para cambiar un poco la Tasa Interna de Retorno del proyecto. Se observa que la variación del costo de energía eléctrica es inversa a la Tasa Interna de Retorno, es decir que si aumenta el costo de la energía eléctrica el retorno del proyecto disminuye. Esto se da porque el beneficio neto del proyecto disminuye, sin embargo el costo de la energía eléctrica debe aumentar demasiado para cambiar un poco la Tasa Interna de Retorno del proyecto. El mantenimiento del robot también disminuye la Tasa Interna de Retorno, pero este gasto debe aumentar a más de mil quinientos quetzales mensuales para que el proyecto deje de ser atractivo en el horizonte de cinco años.

De este análisis se deduce que las variables más importantes para la rentabilidad del proyecto son el número de viajes que el robot realiza diariamente y el salario mínimo considerado. Estas dos variables afectan el

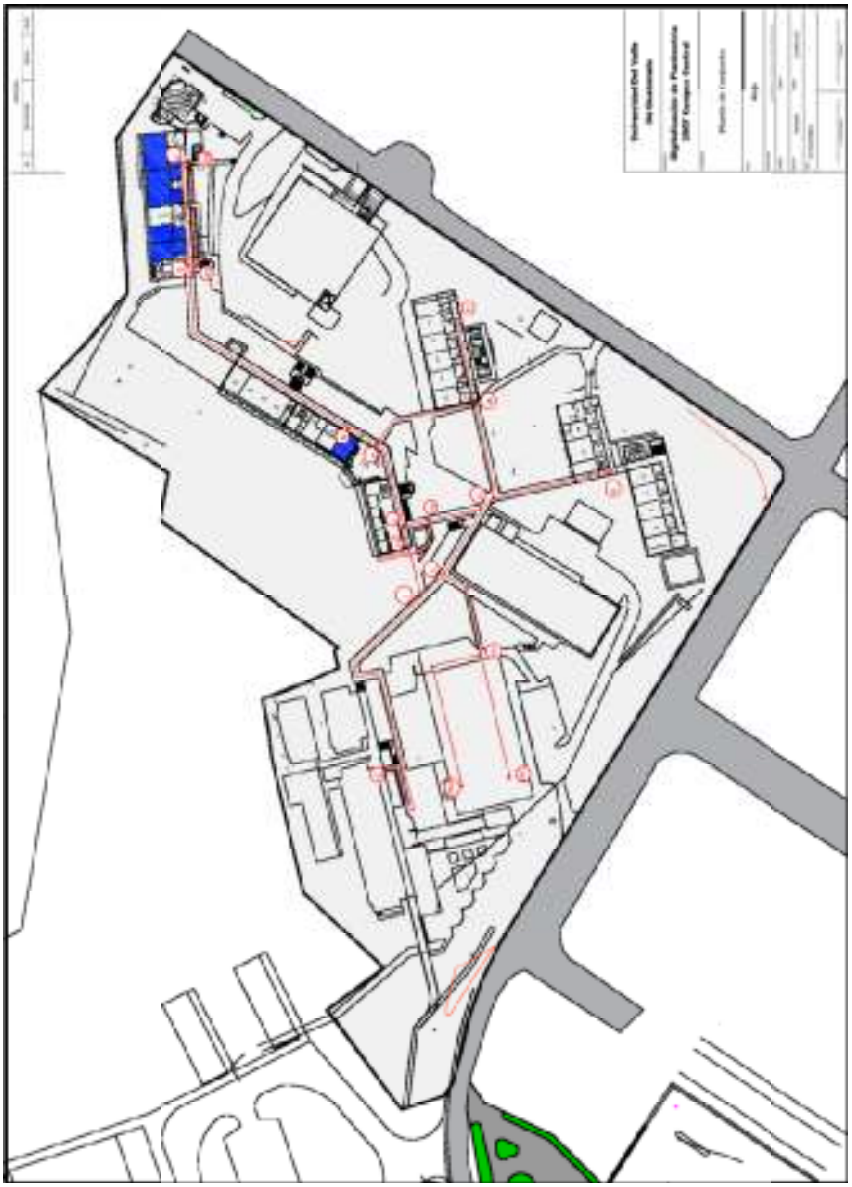
beneficio y no los costos del robot, esto porque el costo mayor es la inversión inicial que ya se realizó y que no puede variar, entonces la rentabilidad del proyecto depende del beneficio que se obtenga de él.

D. ANÁLISIS DE LA RUTA MÁS CORTA

Se realizó un análisis de las rutas más cortas que debe recorrer el robot para optimizar el consumo de energía y el tiempo utilizado para cada uno de los viajes. Esto es importante para poder indicarle al robot la ruta que debe seguir para llegar a su destino de la forma más rápida posible. Si el robot puede hacer los viajes más rápido entonces puede llevar a cabo un mayor número de viajes al día y aumenta la rentabilidad del proyecto.

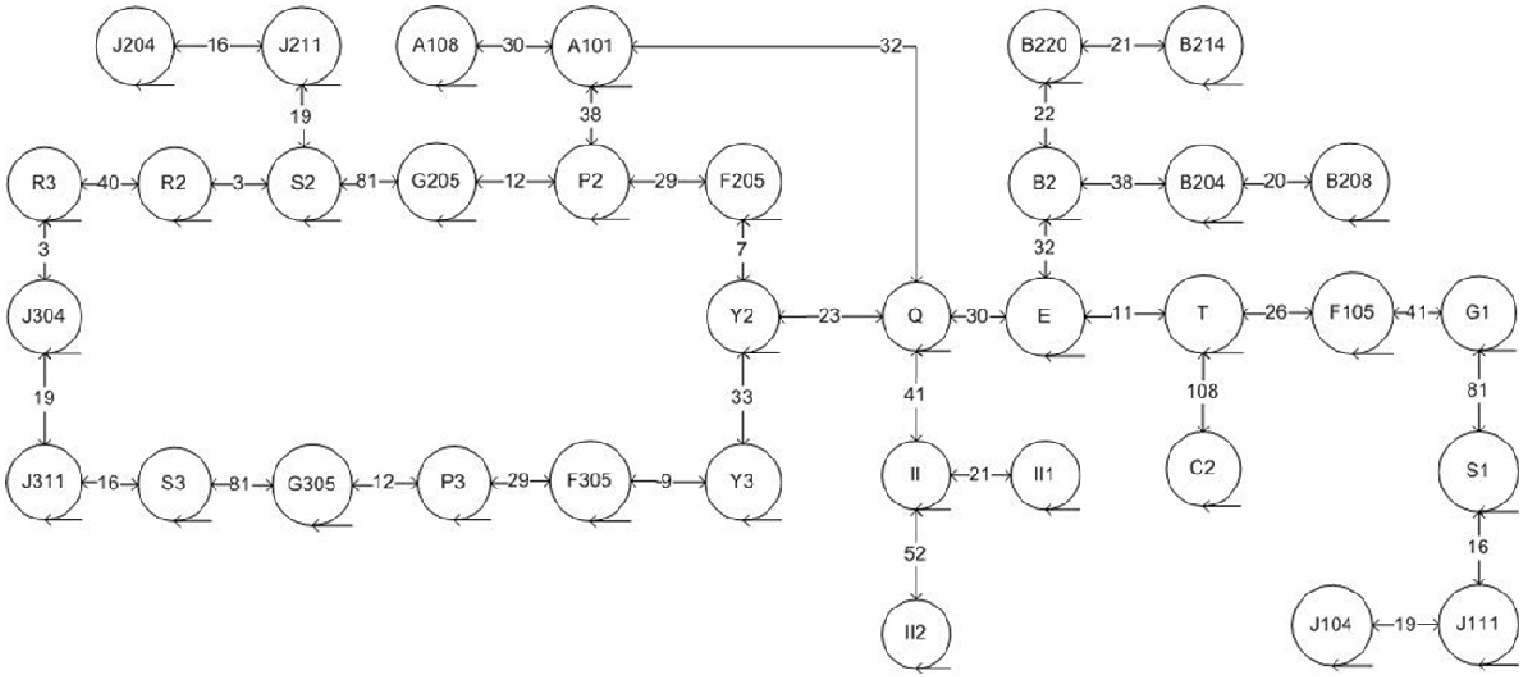
1. Trazo del modelo de red. Para el análisis primero se plantearon los nodos y los arcos de una red que representan los lugares y caminos que el robot debe seguir dentro de la universidad. Se planteó un nodo por cada oficina y algunos otros nodos de referencia. La notación de los nodos corresponde al número de oficinas y los nodos de referencia tienen otra notación para no confundirlas. Se consideraron solamente las oficinas que reciben mayor número de correspondencia y no se tomaron en cuenta los salones de clases. Tampoco se consideró un nodo por los niveles de los edificios que tienen solamente una rampa de acceso porque esta es la única ruta posible que el robot puede seguir. No se consideró ninguna oficina del edificio "E" porque éste no cuenta con rampas de acceso y el robot no puede ingresar.

Figura 8. Trazo del recorrido del robot[1]



[1] Mapa realizado para el proyecto de digitalización del campus central de la Universidad del Valle. Autor desconocido.

Figura 9. Diagrama de red del recorrido del robot



Nomenclatura:

- La primera letra representa el edificio; el primer número representa el nivel y los siguientes son correlativos.
- Los nodos están identificados de la misma forma que la oficina que representan con excepción de los nodos de referencia.
- Las distancias entre nodos están medidas en metros.

2. Solución de la ruta más corta. Al tener planteada la red del recorrido los datos fueron ingresados al programa WinQSB para el análisis de las rutas más cortas. En el programa se ingresan de forma matricial las distancias entre cada uno de los nodos de la red. Para los nodos que no están conectados se ingresa una distancia mucho mayor a las distancias reales para que el programa no considere esta ruta.

El programa permite encontrar la distancia más corta entre cualquier par de puntos de la red ingresada. Sólo se indica el origen y el destino y el programa calcula automáticamente los nodos por los que debe pasar el robot y la distancia total del viaje.

Como ejemplo de los resultados del análisis realizado en WinQSB de la ruta más corta según la red planteada se muestra el recorrido desde la oficina J-304 que es donde está ubicado el departamento de ingeniería mecatrónica y que será donde estará basado el robot hasta la oficina F-205 (secretaría) que es la oficina que recibe la mayor cantidad de correspondencia.

Tabla 7. Resultado de ruta más corta entre oficinas J-304 y F-205

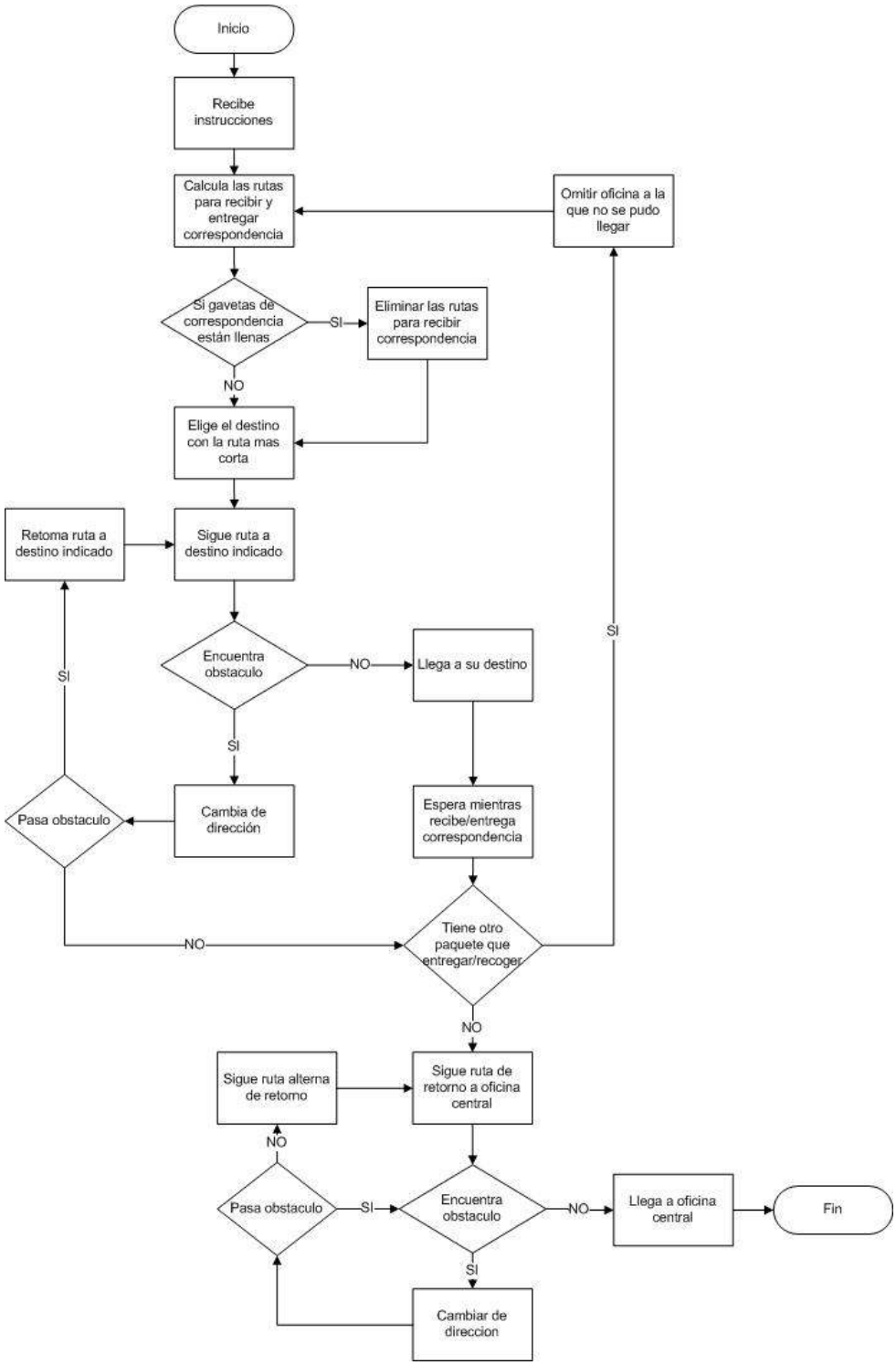
From	To	Distance/Cost	Cumulative Distance/Cost
J304	R3	3	3
R3	R2	40	43
R2	S2	3	46
S2	G205	81	127
G205	P2	12	139
P2	F205	29	168

Lo que se observa en la tabla es que la ruta más corta que el robot puede seguir según la red planteada parte de la oficina de origen y baja por la rampa ubicada en el exterior del edificio J. Luego pasa por el segundo nivel de los edificios I, H, G y F hasta llegar la oficina F-205. La columna de distancia

muestra la distancia entre el nodo de partida y el de destino y en la siguiente columna se muestra la distancia acumulada. La distancia total es la correspondiente a la distancia acumulada hasta el último nodo. De igual forma se puede encontrar la ruta más corta entre dos nodos cualesquiera de la red que representa las oficinas de la universidad. Esto permite reducir la distancia de los recorridos del robot y consumir menos energía de la batería.

3. Diagrama de flujo de operación del robot. Luego de realizar el análisis de la ruta más corta se observó que la optimización de la operación del robot no requiere solamente de calcular la ruta más corta entre dos nodos porque el robot se puede encontrar en una situación en la que debe elegir el orden en que debe entregar los paquetes a las oficinas que tiene asignadas. Si el robot entrega la correspondencia simplemente por el orden en que le fueron asignadas las tareas puede recorrer mucha distancia innecesaria sólo para regresar a un punto cercano para entregar otro paquete. Para optimizar esta operación se desarrolló un diagrama de flujo que resume las operaciones y decisiones que debe tomar el robot para ser más eficiente.

Figura 10. Diagrama de flujo de operación del robot



E. PLAN DE MERCADEO DEL PRODUCTO

1. Definición del mercado, el cliente y el consumidor.

a. El mercado. El mercado objetivo del producto son todos los empleados de la Universidad del Valle a quienes se les intenta vender la idea de utilizar el robot para entregar su correspondencia. El número total de empleados del campus central de la Universidad del Valle es de novecientos setenta, incluyendo empleados por planilla y por honorarios. De este universo de personas se calculó por medio de la encuesta que el 65% realiza viajes para entregar correspondencia, por lo que se calcula que el tamaño del mercado es de seiscientas treinta personas. Todos los empleados de la universidad cuentan con acceso a internet para ingresar a la página web del robot para solicitar los servicios. Para el primer año del proyecto se espera cubrir un 20% del mercado.

b. El cliente. Es la Universidad del Valle de Guatemala a quien se le ofrece el servicio del robot para entrega de correspondencia.

c. El consumidor. Son empleados administrativos, catedráticos, secretarias y cualquier otro empleado de la Universidad del Valle de Guatemala que utiliza una computadora con acceso a internet y por la naturaleza de su trabajo debe movilizarse entre distintas oficinas del campus central para entregar papelería, correspondencia o paquetes pequeños. Los empleados usualmente tienen una apretada agenda de trabajo diaria y están dispuestos a utilizar métodos que les permitan ahorrar tiempo en algunas tareas.

2. Misión, visión y propuesta de valor.

a. Visión. Ser el método más utilizado por los empleados de la Universidad del Valle de Guatemala para la entrega de correspondencia en el campus central.

b. Misión. Ser una herramienta moderna y eficaz para los empleados de la universidad del valle de Guatemala que les permita entregar fácilmente su correspondencia para ahorrarles tiempo en sus actividades diarias.

c. Propuesta de valor. El robot RAEC es un instrumento para los empleados de la Universidad del Valle de Guatemala que les permitirá entregar su correspondencia entre oficinas sin tener que salir de la oficina donde están trabajando, esto le permite a la Universidad del Valle disminuir el tiempo ocioso que es el tiempo de duración del trayecto de los empleados para entregar correspondencia.

El robot es también una muestra de la capacidad y de las habilidades aprendidas por los estudiantes de ingeniería puesta en práctica en un proyecto funcional. Proyecto de este tipo significan un aumento del prestigio del centro de estudios porque pone en práctica lo que sus estudiantes aprendieron a lo largo de la carrera.

3. Análisis FODA

Tabla 8. Análisis FODA del proyecto

Fortalezas	**Debilidades**
- Es un método innovador para entregar la correspondencia y los paquetes pequeños dentro del campus central de la Universidad del Valle de Guatemala. - El proyecto pone en práctica los conocimientos adquiridos por un grupo de estudiantes de ingeniería.	- El robot no puede subir ni bajar gradas. - Para utilizarlo los empleados deben tener acceso a internet. - El robot no puede operar cuando está lloviendo. - El robot no puede transitar por lugares donde transitan automóviles por el riesgo a ser descompuesto. - El tiempo de entrega de correspondencia es mayor a si se realiza por la persona.
Oportunidades	**Amenazas**
- Existe un gran número de viajes que se realizan diariamente dentro de la universidad para entregar correspondencia entre oficinas. - El tiempo liberado a los empleados puede ser aprovechado para realizar otras actividades.	- La disminución del uso de correspondencia física por el uso del correo electrónico. - Fallas no consideradas del robot terminado. - Si aumenta mucho la demanda de utilización del robot existirá un tiempo en que no se podrá utilizar en lo que se carga la batería.

4. Marketing mix

a. Producto. El producto se refiere al bien físico o al servicio que se le entrega al cliente. En este caso es una combinación de ambos porque el robot es un producto que se entrega al cliente, pero también se acompaña del

servicio de la instalación del software para la operación del robot y del análisis de la red de operación del robot dentro del campus. El servicio incluye también asesoría para iniciar las operaciones del robot.

b. Precio. El precio es el valor en moneda que se le da al producto o servicio. En este caso el precio es igual al costo del proyecto que fue de Q. 11,268.26 que fue pagado por el cliente. Los consumidores no pagarán nada por utilizar el servicio porque este ya fue pagado por la universidad.

c. Plaza. La plaza del producto es el campus central de la Universidad del Valle de Guatemala.

d. Promoción. La promoción comprende todas las herramientas de comunicación impresa o digital que motiva al consumidor para que compre o utilice el producto o servicio. Para que los empleados utilicen al robot se les enviará a todos por correo electrónico la página web del robot para que puedan empezarlo a utilizar. Se colocará también un link en la página web de la Universidad del Valle de Guatemala dirigido a la página web del robot.

La promoción del producto se hará por medio de relaciones públicas que son la mejor forma de lanzar un nuevo producto o servicio. Estas relaciones se describen por medio del a herramienta "PENCIL".

- Publicaciones: Se hará un reportaje acerca del proyecto en el boletín mensual de la universidad para dar a conocer el robot dentro de la comunidad estudiantil y docente.
- Eventos: Se realizará una presentación pública del proyecto terminado en el auditorio de la universidad por parte de los integrantes del equipo que realizó el robot.

- Noticias: Se buscará dar a conocer el proyecto en los periódicos de Guatemala como lo han hecho otros proyectos anteriores de la universidad.
- Compromiso con la comunidad: Este proyecto está enfocado a entregar la correspondencia de las oficinas, pero un robot de este tipo puede tener aplicaciones para auxiliar a personas con discapacidades.
- Identificación: Para crear una identidad al proyecto se creó un nombre propio para el robot y un logo para colocar en la página web y en todos los medios donde se promocione al robot. El nombre que se creó es RAEC que es el acrónimo de Robot Autónomo para Entrega de Correspondencia.

Figura 11. Logo del proyecto

- Lobby: Se tendrán reuniones con las autoridades de la universidad para presentar el proyecto y sus beneficios. Con esto se buscará influir en ellos para que motiven a los empleados a utilizar este servicio.

VI. DESARROLLO DEL MÓDULO DE NAVEGACIÓN E IMPLEMENTACIÓN DEL ALGORITMO DE NAVEGACIÓN

A. Medición de distancia con sensor ultrasónico PING

1. Diseño. Con los sensores ultrasónicos se hicieron varias pruebas, entre todas las pruebas se pretendía conocer ciertos aspectos de los sensores, entre ellos la calidad de su medición, interferencia con otros sensores ultrasónicos y el comportamiento de las mediciones con los sensores ya colocados a escala. Para todas las pruebas con los sensores se utilizó un microcontrolador y se desplegaron los datos en la hyperterminal de una computadora.

Para todas las pruebas se utilizó un microcontrolador PIC16F887. Este microcontrolador se programó en lenguaje ensamblador pues de esta manera se tenía un mejor control de los tiempos de ejecución del programa. Para desplegar los datos obtenidos de los sensores se implementó comunicación con una computadora. Esta comunicación se hizo a través de RS-232. De las mediciones arrojadas por los sensores de distancia el microcontrolador convertía los datos a unidades, decenas y centenas. Después estos datos los enviaba a la computadora, así se podía observar las mediciones de una forma más sencilla.

Para todas las pruebas realizadas con los sensores ultrasónicos se utilizó el mismo programa del microcontrolador. A continuación se muestran dos diagramas de flujo del programa, uno representa la función principal del programa y el otro representa el servicio de interrupción del puerto B.

Figura 12. Diagrama de flujo del programa de control de los sensores ultrasónicos.

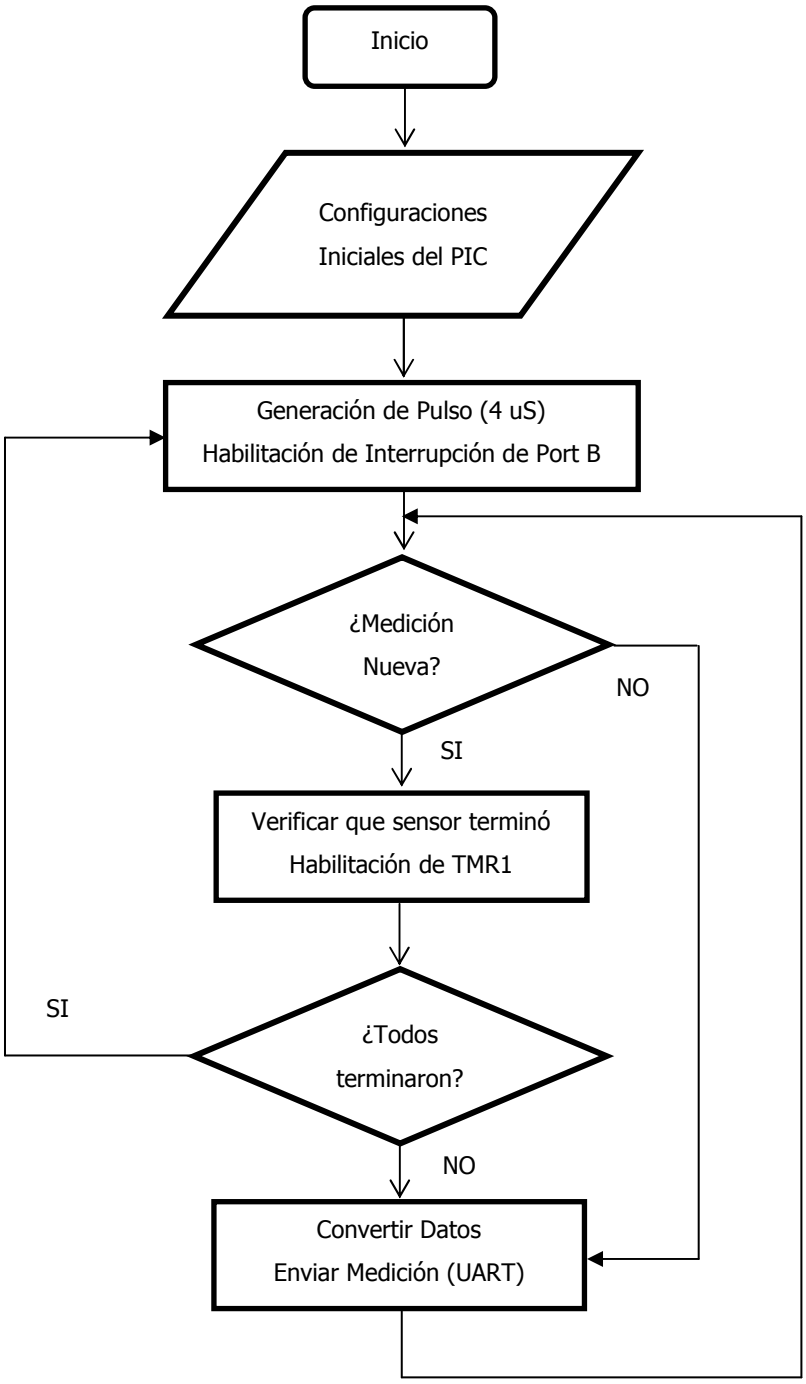
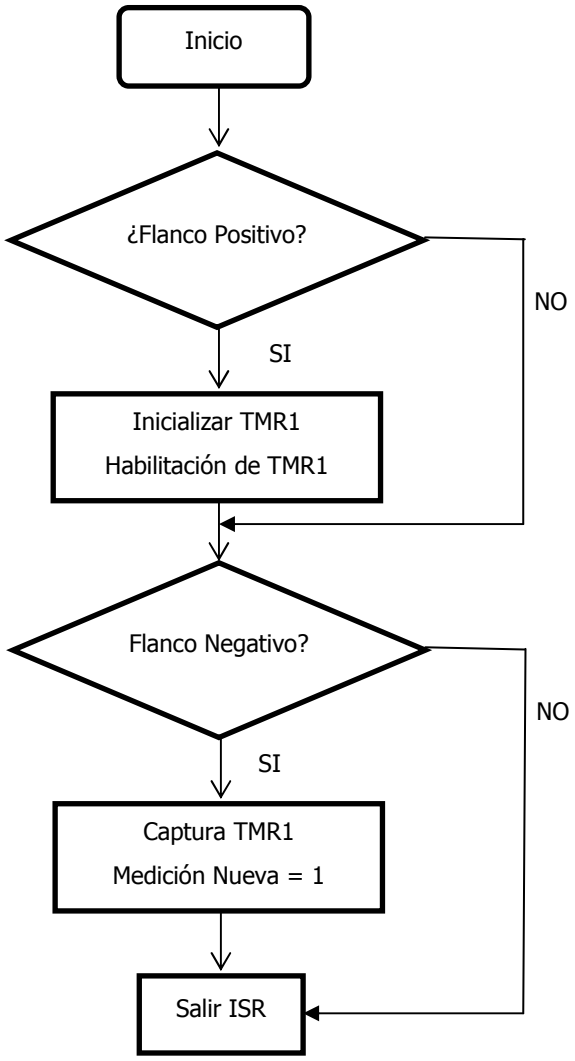


Figura 13. Diagrama de flujo de la interrupción PORTB del programa de control de un sensor ultrasónico.



En la Figura 12 se puede apreciar el diagrama de flujo que se llevó a cabo para la medición de la distancia a partir del microcontrolador y de los sensores ultrasónicos. En la Figura 13 se muestra el diagrama de flujo que se implementó para capturar los cambios en puerto B.

2. Resultados. Con los sensores de distancia se hicieron una serie de pruebas la primera consistió en ver la exactitud de los datos arrojados por un sensor.

Tabla 9. Datos obtenidos por el sensor ultrasónico para una distancia de hasta 50 cm con incrementos de 5 cm.

NO.	DISTANCIA (cm)	MEDICIÓN (cm)
1	2	0
2	5	5
3	10	10
4	15	15
7	20	19
6	25	24
7	30	29
8	35	37
9	40	39
10	45	44
11	50	51

Figura 14. Gráfica de los datos de la Tabla 9.

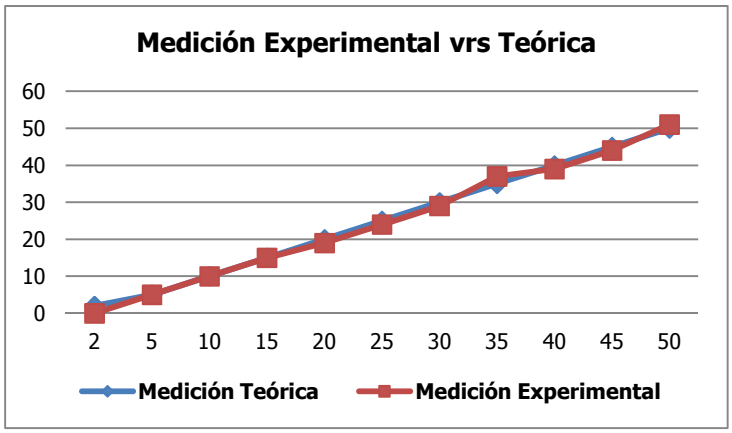
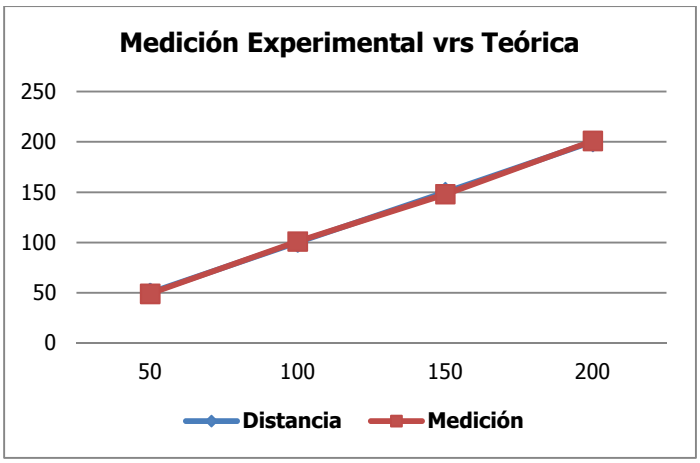


Tabla 10. Datos obtenidos por el sensor ultrasónico para una distancia de hasta 200 cm con incrementos de 50 cm.

NO.	DISTANCIA (cm)	MEDICIÓN (cm)
1	50	49
2	100	101
3	150	148
4	200	201

Figura 15. Gráfica de los datos obtenidos en la Tabla 10.



Después se realizaron pruebas con un sensor para verificar hasta qué punto el ángulo al que está el objeto a detectar afecta significativamente la medición.

Tabla 11. Mediciones realizadas con un sensor ultrasónico variando el ángulo del objeto detectado respecto al sensor.

NO.	ÁNGULO	DISTANCIA (cm)	MEDICIÓN (cm)
1	0	50	50
2	15	50	50
3	30	50	50
4	45	50	45
5	60	50	60

Figura 16. Gráfica de los datos de la Tabla 11.

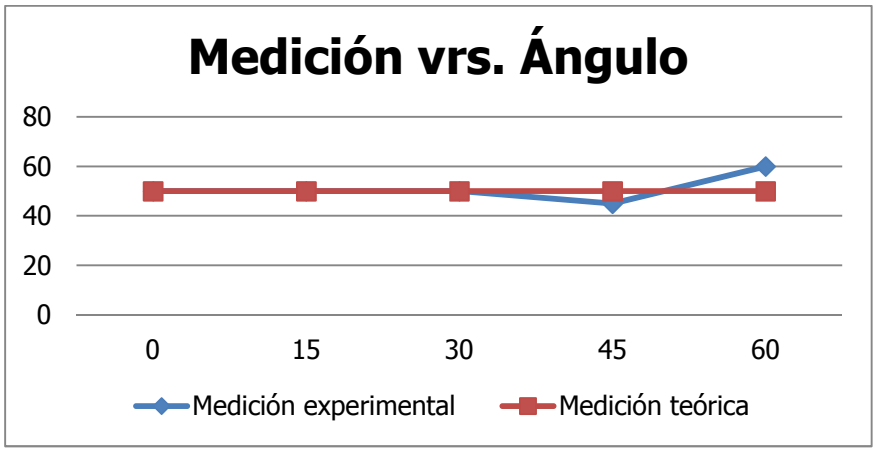
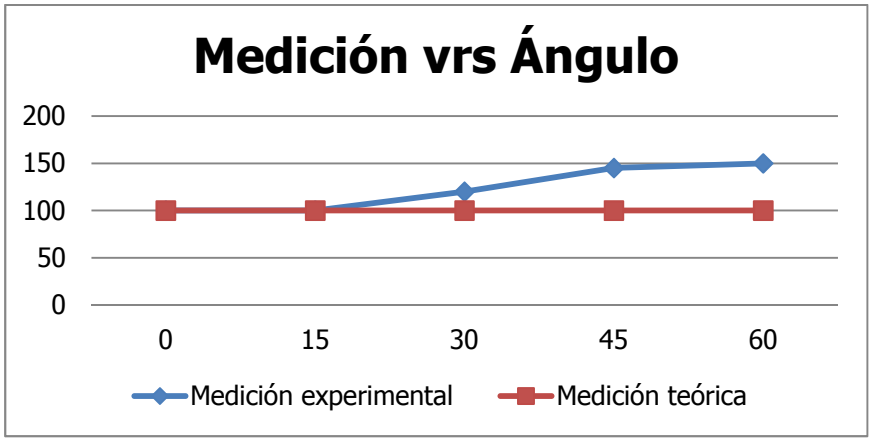


Tabla 12. Mediciones realizadas con un sensor ultrasónico variando el ángulo del objeto detectado respecto al sensor.

NO.	ÁNGULO	DISTANCIA (cm)	MEDICIÓN (cm)
1	0	100	100
2	15	100	100
3	30	100	120
4	45	100	145
5	60	100	150

Figura 17. Gráfica de los datos de la Tabla 12



Después de realizar estos experimentos de forma individual, se procedió a controlar a 5 sensores al mismo tiempo con el fin de poner estos a escala y verificar si con los sensores ya colocados de la misma forma en que estarían colocados en el robot el ángulo de los objetos afectaría de forma significativa las mediciones.

3. Discusión. En la Tabla 10 se muestran los datos obtenidos del sensor ultrasónico. Se hicieron 10 mediciones con incrementos de 5 cm entre cada medición. Se empezó a una distancia de 0 cm y se terminó a 50 cm. En la Tabla 12 se muestran los datos obtenidos nuevamente por el sensor ultrasónico. En este caso se hicieron cuatro mediciones con incrementos de 50 cm. Se empezó a 50 cm y se terminó a 2 m.

Las Figuras 14 y 15 muestran una gráfica de los datos de las Tablas 9 y 10 respectivamente. Al observar las gráficas se puede apreciar que los sensores son bastante exactos. A pesar de que en algunas mediciones se tiene una diferencia de hasta 3 cm con respecto a la distancia real, esto no afecta a los fines del proyecto, pues los sensores se usarán para evitar chocar con objetos. El algoritmo de navegación no permitirá que el robot se acerque a más de 10 cm de cualquier objeto por lo que una diferencia de 3 cm a la distancia real no afectará.

En la Tabla 11 se muestran los datos obtenidos al poner un cuaderno a una distancia de 50 cm del sensor. En esta prueba lo que se hizo fue variar el ángulo del cuaderno con respecto al sensor para ver si esto afectaba la medición de alguna forma. Se hicieron cinco mediciones se comenzó a cero grados. Cero grados se definió como el cuaderno en paralelo al sensor. Luego se fue variando el ángulo del cuaderno de modo que las ondas ultrasónicas generadas por el sensor ya no incidieran de forma ortogonal sobre la superficie del cuaderno. Se

hicieron variaciones de 15° entre cada medición. Luego se repitió el experimento pero con el cuaderno a una distancia de 1 m y se mostraron los datos en la tabla 12.

En las Figuras 16 y 17 se muestran las gráficas de los datos de las Tablas 11 y 12 respectivamente. Al observar estas figuras (16 y 17) se puede apreciar que a partir de 45° mientras mayor es ángulo más error hay en la medición. Para saber si esto era un problema para el proyecto, lo que se hizo fue colocar 5 sensores a escala con respecto de cómo deberían de ir en la estructura del robot.

Figura 18. Imagen de los sensores ultrasónicos colocados a escala.

En la Figura 18 se muestra una imagen de cómo se miraban los sensores colocados a escala. Para esta prueba se pretendían encontrar los espacios ciegos que había entre los sensores y ver si el ángulo de incidencia afectaba a dos sensores simultáneamente.

Con esta prueba de colocar los sensores a escala real, se llegó a la conclusión que en efecto hay un punto ciego entre cada sensor. Estos puntos ciegos son significativos para distancias mayores a 30 cm. A menores distancias se determinó que únicamente objetos muy pequeños no serían detectados. Esto significa que para el proyecto este posicionamiento de los sensores será útil pues si un objeto está en el punto ciego de los sensores y el objeto está a más de 30 cm, cuando el robot se empiece a acercar al objeto éste será capaz de detectarlo.

B. Medición de orientación utilizando brújula HMC

1. Diseño. Para el sensor de campo magnético o brújula se hizo de primero un programa en MikroC de modo de controlar el sensor con el microcontrolador PIC16F887. El programa no se hizo en lenguaje ensamblador inicialmente pues de primero se quería conocer el funcionamiento del sensor y MikroC brinda una librería de I^2C fácil de usar. Este primer programa lo que hacía es que le mandaba un comando a la brújula para que hiciera una medición y después el resultado lo mostraba en la LCD del PIC.

Después se hizo un programa para el PIC 16F887 en MikroC que calibrara la brújula. Esto fue necesario ya que la brújula es susceptible al ambiente en el que se encuentre entonces para obtener más confiables y mejores mediciones.

Por último al programa en lenguaje ensamblador que quedó como resultado de las pruebas de los sensores ultrasónicos se le agregó una rutina que se comunicara con la brújula, obtuviera su medición y el resultado lo guardara en dos variables (esto porque el resultado es dos bytes).

A continuación se muestran los diagramas de flujo de los programas.

Figura 19. Diagrama de flujo del programa en MikroC que controla la brújula.

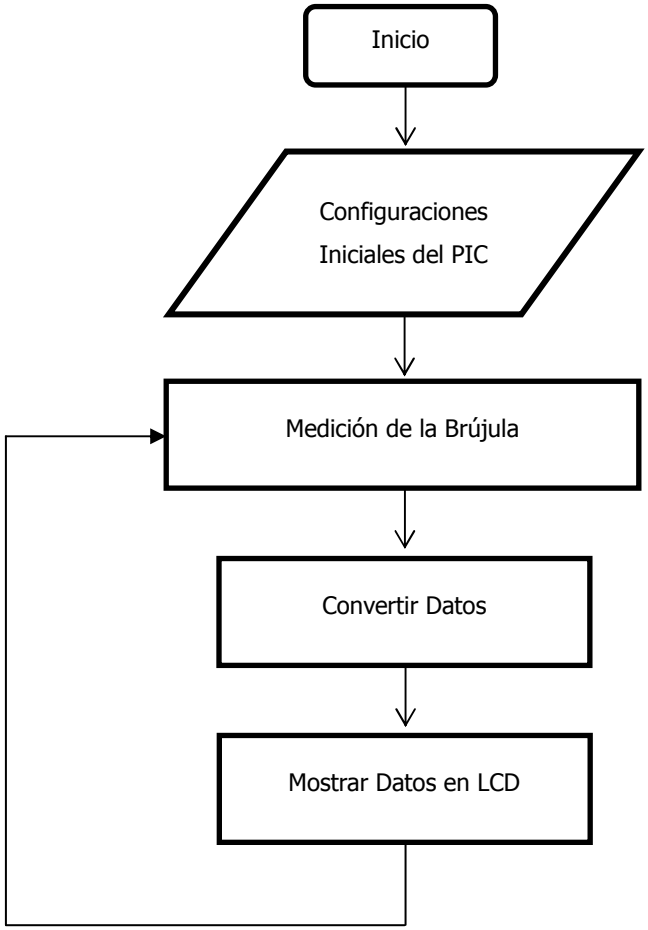


Figura 20. Diagrama de flujo del método llamado Medición de la Brújula.

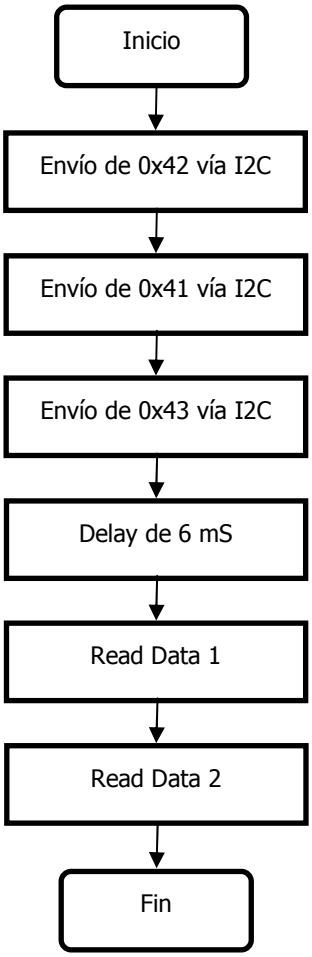
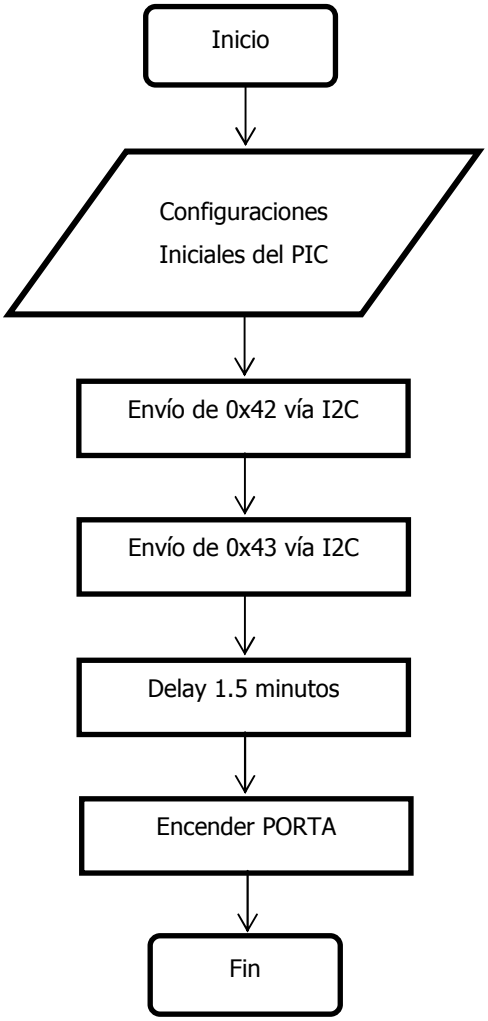


Figura 21. Diagrama de flujo del programa en MikroC que calibra la brújula.



2. Resultados y discusión. Para el primer programa se mostraron los datos en la LCD del EasyPIC. Al principio los datos que arrojaba la brújula eran incorrectos, esto se debía a que ésta no viene calibrada de la fábrica. Calibrar la brújula es sencillo, hay que mandarle el comando 0x43 después de haberla direccionado con un 0x42. Después de haberle mandado el comando hay que girar la brújula 360 ° grados, a una velocidad de más o menos 1 vuelta por cada 10 segundos, esto hay que hacer por entre 20 segundos y 3 minutos. Luego hay

que guardar la configuración en EEPROM para que así la siguiente vez que se encienda la brújula cargue los datos de la calibración.

La brújula es susceptible a campos magnéticos generados por computadoras o celulares, pero cabe mencionar que para que estos campos magnéticos afecten las mediciones de la brújula el objeto que genera el campo magnético tiene que estar por lo menos a una distancia menor a 20 cm de la brújula. Cómo la brújula se ubicará hasta arriba de la estructura en el centro, el campo magnético de las computadoras no le afectará. En esta posición el campo magnético de las computadoras no afectará ya que el robot tiene un radio aproximado de 0.5 m, con este radio se puede asegurar de que si el robot pasa muy cerca de una persona que tenga una computadora portátil ésta estará alejada de al menos 0.5 m de la brújula.

Por último para esta parte se realizó un método en lenguaje ensamblador del PIC 16F887 para que controlara la brújula y guardara los datos de la medición en dos registros o variables. La razón de hacer un método fue para que poco a poco se fuera unificando el programa en lenguaje ensamblador del módulo de navegación. Al programa en lenguaje ensamblador que resultó del manejo de todos los sensores ultrasónicos de distancia se le agregó este método creado, con el fin de tener un programa que por medio de un microcontrolador controlara los sensores de distancia y la brújula de forma simultánea.

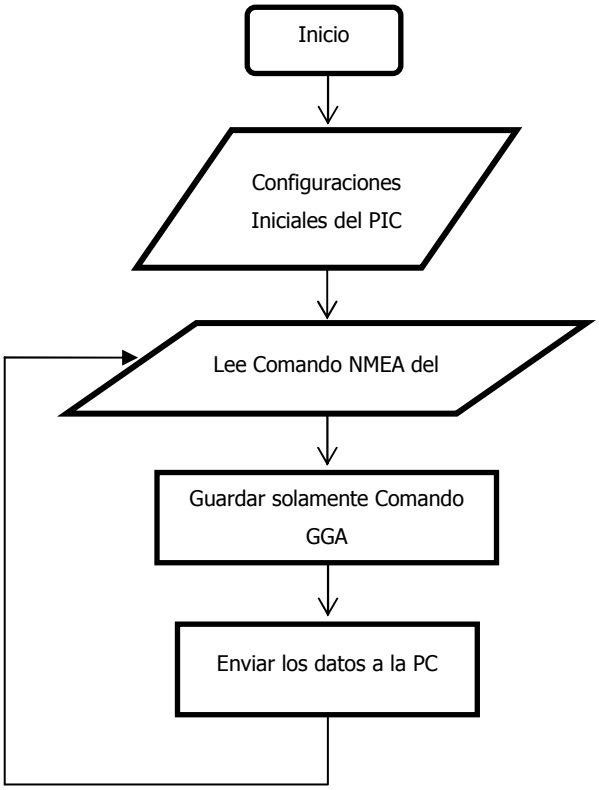
C. Medición de coordenadas de ubicación por medio del receptor GPS UC322

1. Diseño. El receptor GPS seleccionado es uno de los que cuenta con las mejores características en cuanto a precisión y exactitud en el mercado, claro está entre los dispositivos que se encontraban dentro del presupuesto. Este

receptor también tenía una característica particular. Tiene un empaquetado de superficie por lo que para probarlo fue necesario hacer una placa pequeña en donde se soldó el dispositivo. Esta placa se diseñó de modo que tuviera una serie de pines que se pudiera conectar al protoboard.

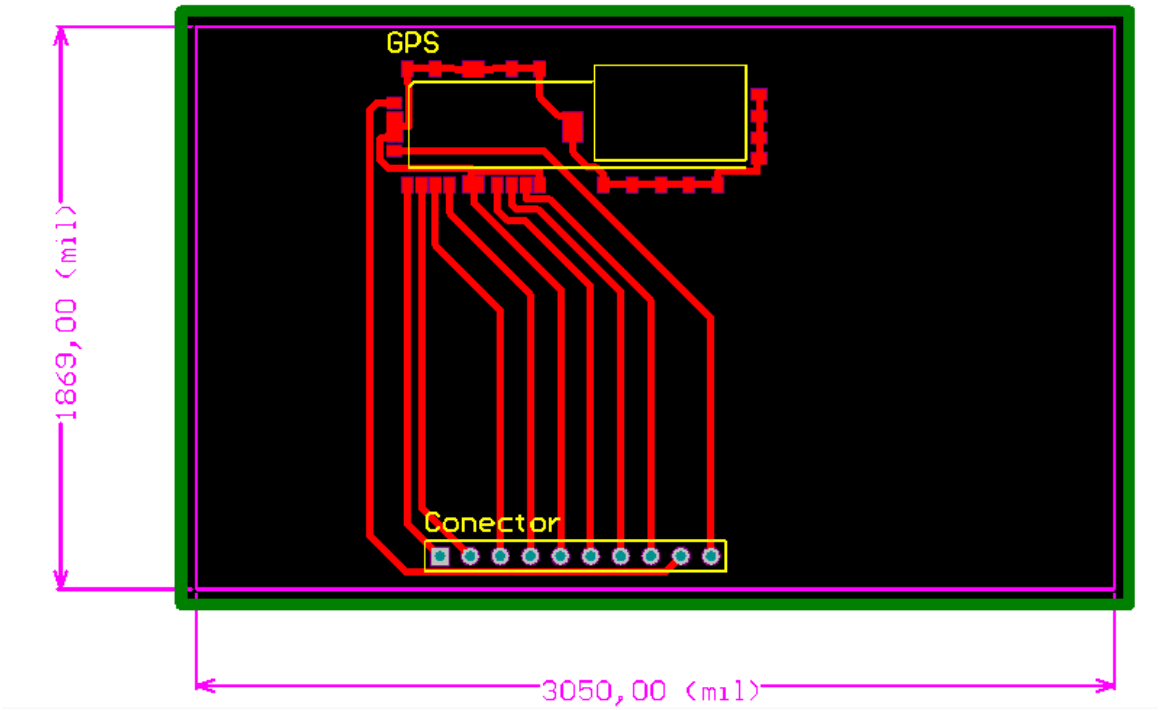
Ya con el dispositivo soldado en la placa que se puede conectar en el protoboard se realizó un programa en MikroC que se comunicaba con el dispositivo GPS y enviaba a la computadora por el puerto serial los datos recopilados. Para esta tarea se decidió utilizar un DSPIC 30F411 ya que este microcontrolador tiene dos interfaces UART. El microcontrolador se decidió programarlo en MikroC pues en este PIC sólo se estarían manejando caracteres ASCII, además se configuró que en el reloj utilizara PLL para que su frecuencia fuera de 80 MHz. A continuación se muestra un diagrama de flujo del programa implementado.

Figura 22. Diagrama de flujo del microcontrolador que controla el receptor GPS.



2. Resultados. El circuito impreso diseñado para utilizar el dispositivo receptor GPS se muestra en la siguiente imagen.

Figura 23. Imagen del PCB para el GPS diseñado en Altium Designer.

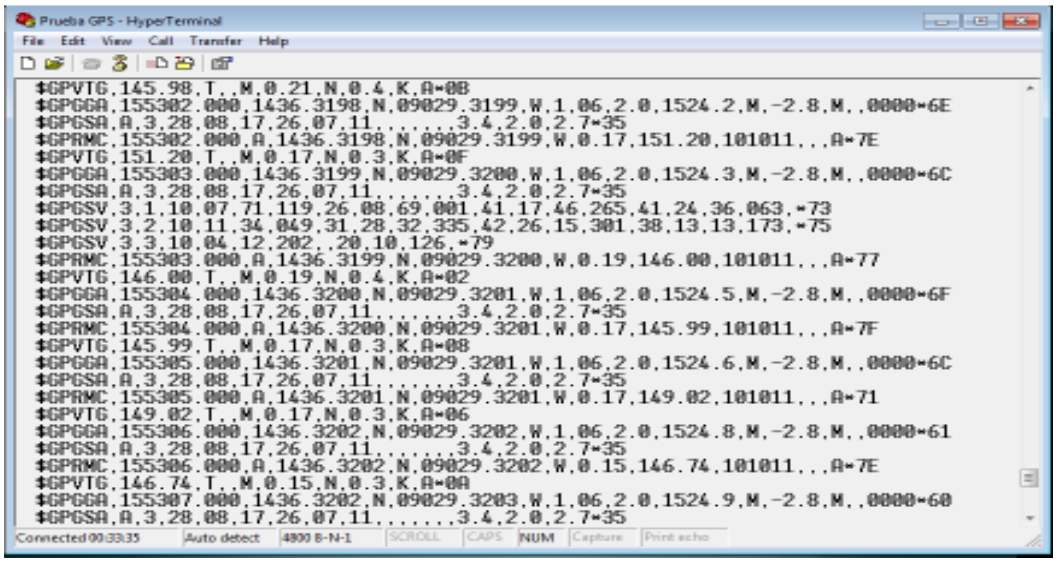


Con el diseño del circuito impreso terminado se procedió a hacer la placa y a soldar el dispositivo receptor GPS y el conector para protoboard. Como resultado se obtuvo la siguiente placa.

Figura 24. Imagen de la placa terminada para el dispositivo receptor GPS.

Por último se probaron todas las soldaduras y caminos de la placa utilizando un multímetro. Posteriormente se procedió a conectar el GPS al microcontrolador para probarlo.

Figura 25. Datos obtenidos mediante el dispositivo receptor GPS.



3. Discusión. Los datos obtenidos del GPS son bastante confiables cuando se está conectado a más de 5 satélites. Si se está conectado a menos de 5 satélites la posición tiene un error muy grande. El error es tan grande que al tomar una medición del GPS en el edificio J con sólo 3 o 4 satélites el punto que devuelve el dispositivo GPS, es un punto que no se encuentra dentro del campus de la UVG. Con la señal de 8 satélites se obtiene un error en la medición de 2.5 metros. En el caso de las mediciones tomadas con señal de 8 satélites los errores son pequeños, pues al corroborar las coordenadas con Google Earth se observa que prácticamente no hay diferencia entre uno y otro. Eso sí hay que recordar que Google Earth no es exacto y también tiene cierto error.

Para no utilizar datos del GPS con un error muy grande únicamente se almacenaban en el PIC los datos del GPS cuando éste estuviera conectado a 5 o

más satélites. Hacer esto fue sencillo porque se estaba utilizando un comando NMEA de tipo GGA para comunicarse con el receptor GPS. En este comando es posible obtener de la información de latitud y longitud, y también obtener información general acerca de la conexión con los satélites. Entre la información general acerca de los satélites que ofrece el comando NMEA GGA se tiene la cantidad de satélites a los que está conectado el receptor GPS.

D. Implementación del algoritmo de navegación

1. Diseño. El robot necesita de un algoritmo de navegación para lograr que con el uso de los ocho sensores ultrasónicos, una brújula, un receptor GPS y el conteo de pasos dados por los motores se logre movilizar al mismo desde un punto a otro en el campus de la UVG. También es por el algoritmo de navegación que el robot va esquivando obstáculos que encuentre en su camino. Este algoritmo fue diseñado, desarrollado y probado por Javier Mesalles integrante del grupo de trabajo del megaproyecto RAEC.

El algoritmo fue diseñado, probado y simulado en un lenguaje de alto nivel (Java), para poder usarlo en el robot fue necesario implementarlo en un microcontrolador.

2. Resultados y discusión. El algoritmo fue implementado en un DSPIC 30F4011, específicamente en el DSPIC RF. Se utilizó el DSPIC RF y no el PIC de Sensores pues muchos de los recurso de este PIC (PIC de Sensores), son usados para comunicarse con los 8 sensores de distancia, la brújula y el dispositivo receptor GPS, además el DSPIC es un PIC orientado hacia el procesamiento de señales digitales y datos, lo que permite tener un mejor rendimiento del algoritmo de navegación.

También otros factores que afectaron para que se decidiera implementar el algoritmo de navegación en el DSPIC RF son que en este PIC se puede tener dos comunicaciones seriales paralelas ya que cuenta con dos módulos UART. Un UART fue destinado para la comunicación inalámbrica y el otro para la comunicación con el PIC de sensores. Debido a que en el DSPICRF se implementó el algoritmo de navegación y se controlan todos los movimientos del robot, este PIC podría ser denominado el PIC de Control del robot.

VII. DESARROLLO DEL MÓDULO DE COMUNICACIÓN INALÁMBRICA

El proceso para que los usuarios del robot puedan enviar paquetes consiste en ingresar a la página web del proyecto e indicar la oficina origen, la oficina destino y presionar el botón de enviar. Sin embargo, esa información estará almacenada en un servidor web el cual querrá transmitir instrucciones al robot para garantizar la entrega del paquete, es por ello que se debe buscar un medio para comunicar dicha información.

El servidor estará colocado en una sala de servidores donde difícilmente se tendrá acceso para conectar o desconectar algún dispositivo, sin embargo, un medio de conexión es la red interna de la universidad a la cual el servidor está conectado, es por ello que se decidió utilizar un dispositivo conversor de protocolo Ethernet a protocolo RS-232. La ventaja de utilizar dicho dispositivo es la relativa portabilidad que podría implicar el poderse conectar en cualquier punto de la red y obtener información del servidor, con lo que el problema de conexión queda superado de esta manera.

El siguiente paso consiste en especificar una forma para transmitir las señales RS-232 de manera inalámbrica hacia el robot. La solución encontrada fue utilizar un dispositivo de transmisión RF para mandar dichas señales.

A. Dispositivo SitePlayer Telnet

El SitePlayer Telnet es un dispositivo conversor de protocolo RS-232 a protocolo Ethernet. Sus parámetros pueden ser configurados mediante el puerto serial o mediante una conexión Ethernet. [14]

Este dispositivo, el cual se puede observar en la Figura 26, funciona como un puente ya que manda por el puerto serial todos los mensajes UDP que recibe y viceversa. La gran ventaja de utilizar este dispositivo es que ya viene encapsulado y solo posee tres puntos de conexión, los cuales son: alimentación, conector RJ45 (protocolo Ethernet) y conector DB9 (protocolo serial RS-232). [14]

Figura 26. Dispositivo SitePlayer Telnet [14]

B. Dispositivo XBee DigiMesh 900 Mesh RF

Este dispositivo se muestra en la Figura 27, utiliza el protocolo DigiMesh p2p en una frecuencia de 900 MHz para aplicaciones de largo alcance. Este protocolo ofrece estabilidad en la red al brindar un auto descubrimiento de los dispositivos y una operación de red densa. [6]

Figura 27. Dispositivo XBee DigiMesh 900 [6]

El dispositivo presenta un alcance de 140 metros en un ambiente de interiores como se muestra en la Tabla 13, lo cual garantiza que se puede obtener un buen área de cobertura de señal del robot dentro de las instalaciones de la universidad.

Tabla 13. Desempeño XBee DigiMesh 900 [6]

Tasa de datos RF	156 Kbps
Alcance en interiores	140 m
Alcance línea de vista	3 km
Alcance línea de vista con antena de alta ganancia	10 km
Potencia de transmisión	50 mW (+17 dBm)
Sensibilidad del receptor	-100 dBm

El dispositivo trabaja con voltajes de 3.3 V como se muestra en la Tabla 14, por lo cual es necesario agregarle un adaptador de voltaje el cual cambie los niveles de voltaje de 3.3 V a 5 V para acoplarlo con el resto del circuito del robot.

Tabla 14. Características XBee DigiMesh 900 [6]

Interfaz de datos serial	3.3V CMOS Serial UART
Banda de frecuencias	900 MHz ISM
Inmunidad a interferencia	FHSS
Tasa de datos serial	Hasta 230 Kbps

El dispositivo implementa acuse de recibos como se muestra en la Tabla 15, lo cual garantiza que la transferencia de datos ha sido exitosa, permitiendo así reenvíos de información cuando la transferencia no se realizó con éxito.

Tabla 15. Seguridad y redes de XBee DigiMesh 900 [6]

Encriptación	128-bit AES
Entrega de paquetes	Reenvíos / Acuse de recibo
Opciones de direccionamiento	PAN ID, canal, direccionamiento de 64 bits
Canales	8 patrones de salto en 12 canales

El consumo energético es un factor importante para determinar las características de la fuente de alimentación a utilizar. Como se puede observar en la Tabla 16, la corriente durante la transmisión de información es de 210 mA, mientras que la corriente durante la recepción es de 80 mA, lo cual nos indica que la batería utilizada debe brindar los niveles de corriente mencionados.

Tabla 16. Consumo de energía XBee DigiMesh 900 [6]

Voltaje de alimentación	3.0 – 3.6 VDC
Corriente de transmisión	210 mA
Corriente de recepción	80 mA

C. Resultados y discusión

Se realizó un conjunto de pruebas corroborando el correcto funcionamiento del sistema de comunicación inalámbrica, para ello se enviaron mensajes UDP al dispositivo SitePlayer. Este dispositivo envió por su puerto serial los mensajes recibidos, a dicho puerto se conectó el módulo XBee utilizando un circuito adaptador a RS-232 (incluido con el kit XBee adquirido).

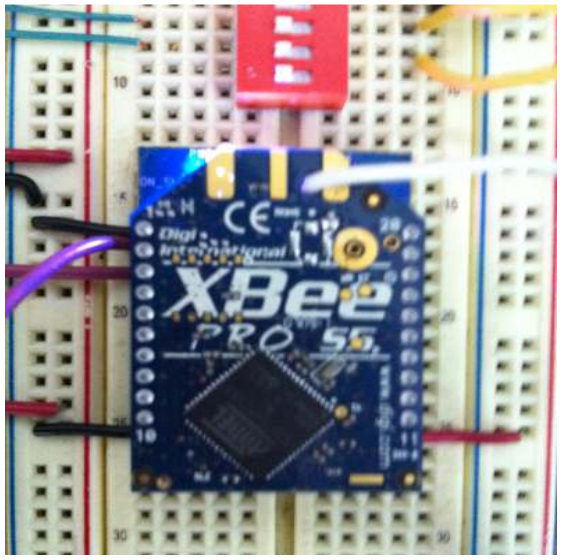
Desde la página web del administrador del robot (ver capítulo VII) se monitorio el estado de los sensores, la orientación de la brújula, la posición del robot y la posición medida por el dispositivo GPS. También se envió comandos de movimientos hacia adelante, atrás, izquierda, derecha y parar. Durante dichas pruebas no se obtuvo ningún inconveniente salvo el hecho de que el dispositivo XBee móvil debe estar a una distancia mínima del dispositivo XBee estacionario, la cual es de aproximadamente 8 m utilizando la antena de alta ganancia.

La Figura 28 muestra del lado izquierdo el módulo XBee montado en el circuito adaptador a RS-232, el cable blanco es el que comunica a este módulo con el dispositivo SitePlayer (lado derecho de la ilustración), este dispositivo está conectado a la red de electrónica de la universidad mediante el cable azul que se observa en la imagen. La ilustración muestra lo que conforma el circuito transmisor/receptor estacionario.

Figura 28. Conexión XBee-SitePlayer

Por otro lado, tenemos el circuito receptor/transmisor móvil mostrado en la Figura 29, el cual es el instalado en el robot y permite a los microcontroladores intercambiar información con el servidor.

Figura 29. Módulo XBee móvil



VIII. DESARROLLO INTERFAZ POR COMPUTADORA

Este capítulo explica el procedimiento utilizado para la creación del sitio web del proyecto, así como también la base de datos creada para el control de usuarios, manejo de envío de paquetes, posición de oficinas y control de estado del robot.

Este capítulo explica también los criterios utilizados para determinar la próxima entrega que debe realizar el robot. De igual manera, se explica el conjunto de instrucciones utilizado para el envío de comandos hacia el robot, y el intercambio de datos con el mismo.

A. Base de datos

Con la finalidad de llevar un control de toda la información referente al sitio web se creó una base de datos llamada RAEC, la cual posee cinco tablas para el control de los distintos aspectos del sitio como se muestra en la Tabla 17.

Tabla 17. Función de tablas con base de datos RAEC

Tabla	Función
Usuarios	Control de cuenta de usuarios registrados
Oficinas	Control del nombre y dirección de oficinas
Nodos	Control del nombre y dirección de nodos, registro de nodos y oficinas adyacentes
Envíos	Registro de paquetes enviados, así como de el estado de cada envío
EstadoRobot	Control de estados del robot (posición, sensores, brújula, GPS, batería, etc.)

1. Tabla Usuarios. Contiene la información de los usuarios registrados. Es utilizada para permitir el acceso a cuentas, recuperación de contraseñas y asignación de privilegios a los usuarios.

2. Tabla Oficinas. Contiene el nombre de las oficinas en las que puede entregar y recoger paquetes el robot, así como también la posición de cada oficina medida en longitud y latitud. Es utilizada para mostrar a los usuarios las oficinas a las cuales el robot brinda el servicio de mensajería, así como para indicar al robot el lugar al que debe dirigirse para determinada oficina.

3. Tabla Nodos. Contiene el nombre de los nodos que se encuentran interconectando dos o más oficinas o nodos, la posición de cada nodo medido en longitud y latitud, y el nombre de los nodos u oficinas con los que tiene conexión. Es utilizada para permitir al robot llegar de un punto a otro recorriendo rutas de menor distancia.

4. Tabla Envíos. Contiene un registro de todos los envíos de paquetes no procesados, los que están en proceso y los que ya han sido entregados. Es utilizada para mostrar al usuario el estado de sus envíos, así como para que el servidor procese los paquetes que aun no han sido enviados y termine la entrega de los paquetes que ya están en proceso.

5. Tabla EstadoRobot. Contiene una lista de variables referentes al estado del robot, así como el valor de cada variable. Es utilizada para el monitoreo de dichas variables y comprobación del correcto funcionamiento del robot.

B. Control de usuarios

Debido a que el sitio web será de acceso para todo el público, es necesario restringir la disponibilidad de los servicios solamente a las personas que le den un uso correcto y responsable al robot. Es por ello que en la base de datos mencionada anteriormente existe una tabla exclusivamente para la información de los usuarios registrados.

1. Registro de usuarios. Todas las personas que deseen utilizar los servicios del robot deberán ingresar al sitio web del proyecto. En la parte derecha de la pantalla se encuentra un área dedicada exclusivamente a los usuarios. En la parte inferior izquierda del área de usuarios se encuentra la palabra "Suscríbase", al hacer click en dicha palabra aparecerá un formulario que debe ser llenado para registrar un nuevo usuario. El registro del nuevo usuario será exitoso únicamente si todos los campos son llenados y tanto el nombre de usuario y la dirección de correo electrónico elegidos aún no existen en la base de datos, en caso de cumplir con los requisitos indicados, se enviará un correo al nuevo usuario indicando su nombre de usuario y su contraseña. Los nuevos usuarios registrados no podrán utilizar los servicios del robot inmediatamente, sino hasta que el administrador de la página le de los privilegios necesarios.

2. Recuperación de contraseña. Los usuarios ya registrados que hayan olvidado su nombre de usuario o su contraseña podrán recuperarlos utilizando la opción "Recuperar Password" que se encuentra en la esquina inferior derecha del área de usuarios del sitio. Para efectuar la recuperación deben escribir la dirección de correo registrada y hacer click en el botón "Recuperar". Una vez realizado lo anterior, si la dirección de correo se encuentra en la base de datos, se enviará un mensaje a dicha dirección indicando el nombre de usuario y una

nueva contraseña generada aleatoriamente por el servidor. La contraseña podrá ser cambiada posteriormente por el usuario.

3. Inicio de sesión. Para iniciar sesión se deben llenar los campos "Usuario" y "Password" que se encuentran en el área de usuarios. Si el nombre de éste existe en la base de datos y la contraseña es correcta, se desplegará la cuenta del usuario. La cuenta de usuario está dividida en tres áreas: "Nombre Usuario", "Estado de Envíos" y "Enviar un Paquete".

a. Nombre de Usuario. Esta área despliega el nombre del usuario y tiene un botón para cerrar sesión, también presenta la opción de cambio de contraseña. Si el usuario desea cambiar de contraseña debe llenar los campos "Actual" y "Nueva" y hacer click en el botón "Cambiar", si la contraseña actual fue escrita correctamente el cambio habrá sido exitoso.

b. Estado de Envíos. Esta área despliega un listado de los códigos de los envíos que se han pedido, así como el estado actual de cada envío. Si el usuario desea ver aspectos como el origen y el destino de cada envío, así como también la fecha y hora de los envíos, puede hacer click en el botón "Ver detalle", desplegando así una pantalla emergente con la información deseada.

c. Enviar un Paquete. Esta área despliega el listado de oficinas en las que tiene cobertura el robot, para hacer un envío basta con elegir la oficina de origen y de destino y hacer click en el botón "Enviar".

C. Control del robot

El robot puede operar en dos modalidades: en control manual y control automático. Cada una de ellas orientadas a satisfacer distintas necesidades del

proyecto, ya que la modalidad de control manual está orientada al administrador, para que pueda realizar pruebas con el robot por si fuera necesario realizar algún ajuste; mientras que la modalidad de control automático está orientada al usuario final, con el propósito de realizar las acciones necesarias que permitan que el robot entregue el paquete de un lugar a otro.

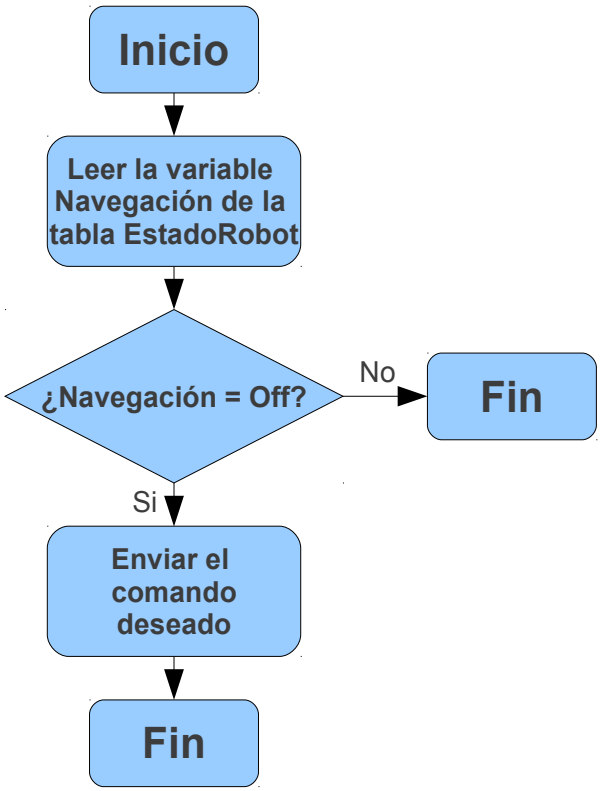
1. Modalidad de control manual. En esta modalidad el robot admite comandos directamente desde la interfaz brindada para uso exclusivo del administrador. Dicha interfaz permite controlar los movimientos del robot (adelante, atrás, izquierda, derecha), monitorear sus variables de estado (sensores, brújula, posición, GPS, batería), enviar al robot a una dirección en específico (longitud, latitud) e indicarle que mensaje desplegar en la pantalla LCD.

2. Modalidad de control automático. Los comandos que se envían al robot no dependen del administrador, sino de una rutina que se ejecuta en el servidor. Dicha rutina toma información de la tabla de envíos de la base de datos, el estado de los envíos es procesado para determinar el siguiente nodo u oficina al que debe dirigirse el robot.

3. Envío de comandos al robot. La forma en la que se envían comandos al robot desde el servidor es por medio del protocolo de datagramas de usuario (UDP) como los descritos en el capítulo V. Básicamente existen dos rutinas para enviar información, una para la modalidad de control manual y otra para la de control automático, la diferencia está en que la primera sólo es llamada cuando el administrador decide enviar un comando, mientras que la segunda se llama una vez y se mantiene ejecutando hasta que se termine la conexión.

a. Rutina de control manual. Esta rutina se ejecuta cada vez que el administrador decide enviar un comando directamente al robot. La rutina lee información de la tabla EstadoRobot de la base de datos y determina si se puede enviar el comando deseado dependiendo si el robot se encuentra en el modo de control manual. La Figura 30 muestra el algoritmo utilizado para esta rutina.

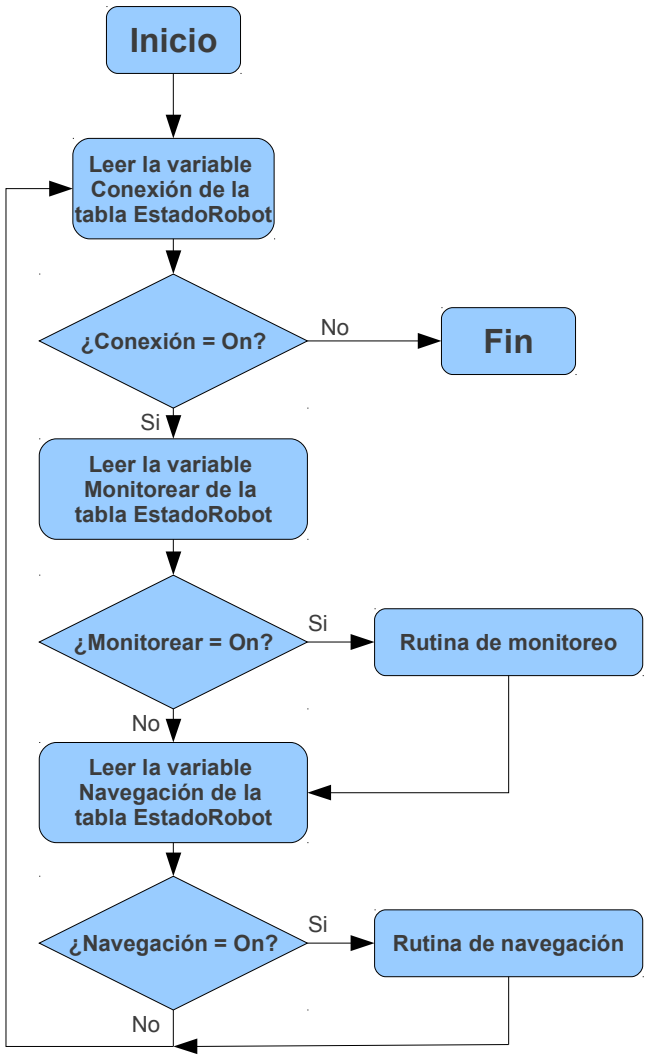
Figura 30. Rutina de control manual



b. Rutina de control automático. Cuando se enciende la comunicación con el robot, se ejecuta una rutina la cual se mantiene en ejecución hasta que se pierda la conexión o hasta que el administrador decida terminarla. La rutina tiene dos funciones, la primera es la de monitorear los sensores, brújula, posición, GPS y batería del robot, y la segunda consiste en

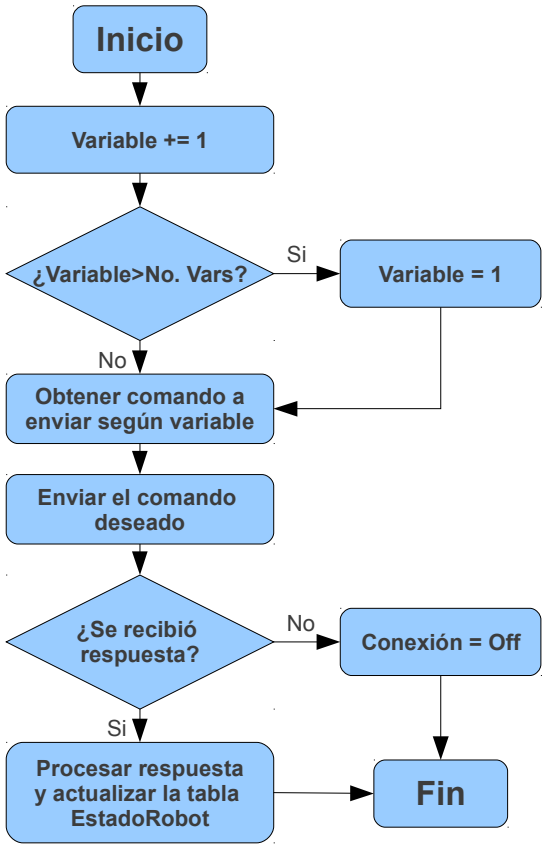
indicarle al robot la dirección (en longitud y latitud) a la cual debe dirigirse. La Figura 31 muestra el algoritmo utilizado para esta rutina.

Figura 31. Rutina de control automático



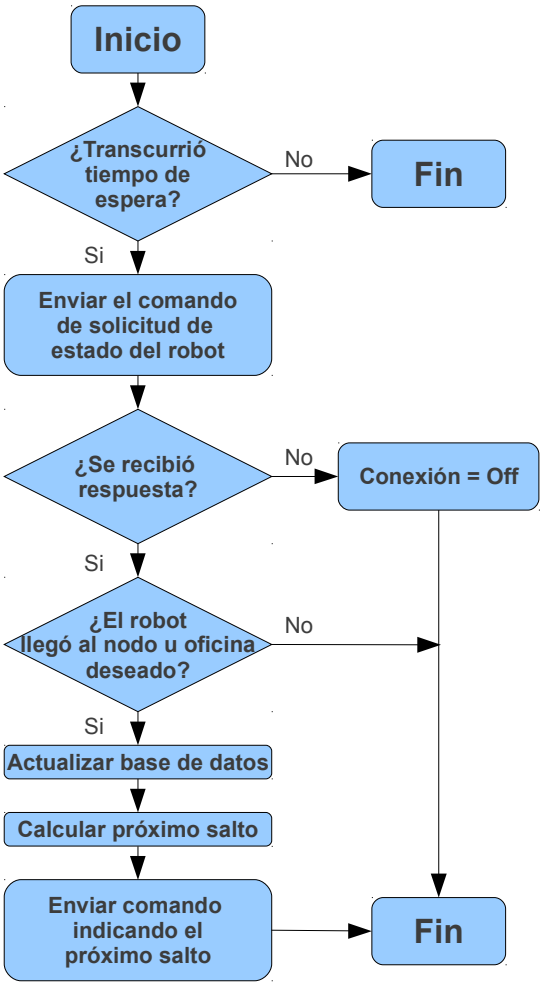
La rutina de monitoreo permite enviar mensajes al robot solicitando el estado de los sensores, brújula, posición, GPS y batería. La Figura 32 muestra el algoritmo utilizado por dicha rutina.

Figura 32. Rutina de monitoreo



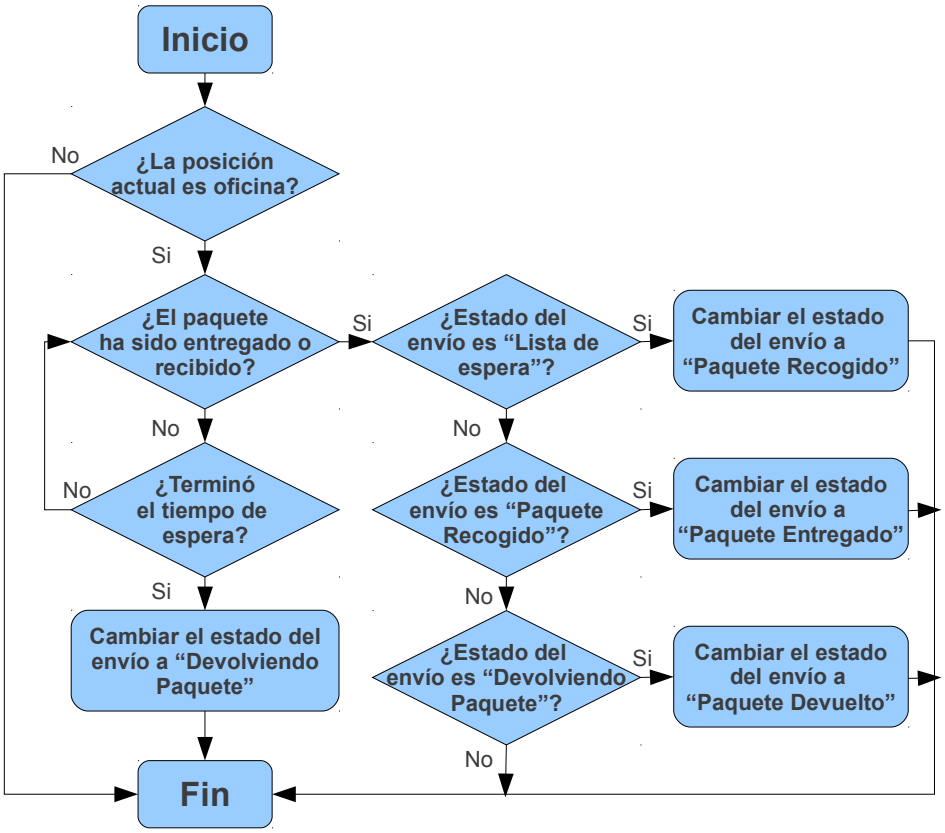
La rutina de navegación permite al servidor verificar la posición del robot y a partir de ello determinar si éste ha llegado a la oficina o nodo que se le indicó, cuando ocurre lo anterior, el servidor procesa la información almacenada en la tabla de envíos de la base de datos y determina la próxima oficina a la cual es necesario enviar al robot. La Figura 33 muestra el algoritmo utilizado para dicha rutina.

Figura 33. Rutina de navegación



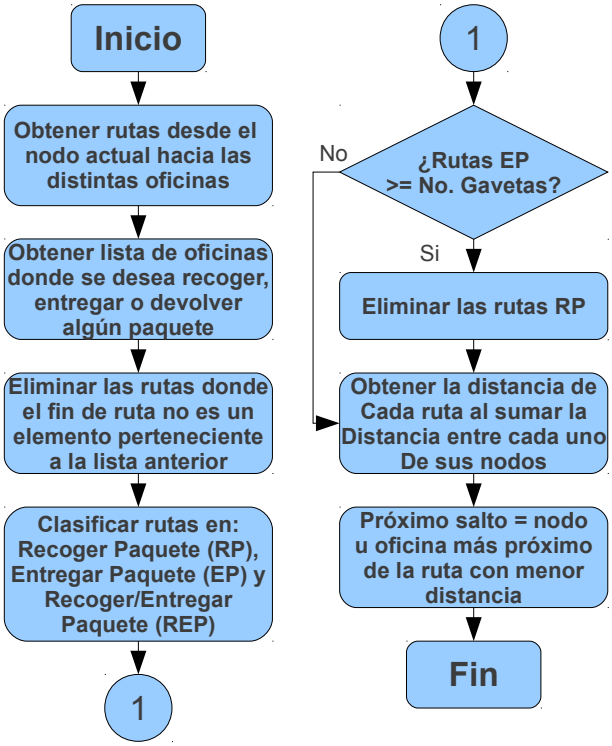
La rutina para actualizar la base de datos se muestra en la Figura 34, su función es la de hacer los cambios necesarios en la base de datos para tener actualizado el estado de los envíos y así poderle dar seguimiento hasta que el paquete sea entregado, y en el caso de que en la oficina de entrega no haya nadie para recibirlo, que el paquete sea devuelto a la oficina de envío.

Figura 34. Actualización de base de datos



La rutina para calcular el próximo salto se muestra en la Figura 35, su función es optimizar los envíos determinando la ruta más corta, para ello obtiene las rutas existentes hacia las oficinas que tienen algún envío relacionado, ya sea para recoger o para entregar un paquete, con dichas rutas se obtiene la distancia total recorrida por cada ruta sumando las distancias entre los nodos u oficinas vecinos de cada ruta y se envía al robot al nodo u oficina siguiente a la posición actual del robot utilizando la ruta con menor distancia.

Figura 35. Cálculo del próximo salto



D. Conjunto de instrucciones robot/servidor

Para intercambiar información entre el robot y el servidor se tiene un conjunto de instrucciones el cual permite identificar el tipo de instrucción, la instrucción y los parámetros deseados para cada una. Se manejan tres tipos de instrucciones: instrucción de movimiento, de solicitud y de transferencia, las cuales se muestran en la Tabla 18.

Tabla 18. Conjunto de instrucciones robot/servidor

Tipo de Instrucción	Instrucción	Comando (hex)
Movimiento	Adelante	FF 01
Movimiento	Atrás	FF 02
Movimiento	Izquierda	FF 03
Movimiento	Derecha	FF 04
Movimiento	Parar	FF 05
Solicitud	Sensores	FE 06
Solicitud	Brújula	FE 07
Solicitud	GPS	FE 08
Solicitud	Posición	FE 09
Solicitud	Batería	FE 10
Solicitud	Estado	FE 11
Transferencia	Oficina origen	FD 12
Transferencia	Oficina destino	FD 13
Transferencia	Numero gaveta	FD 14
Transferencia	Coordenadas destino	FD 15

1. Instrucciones de movimiento. Son utilizadas en la modalidad de control manual y sirven para indicarle al robot si avanzar, retroceder, girar hacia la derecha, hacia la izquierda o detenerse.

2. Instrucciones de solicitud. Sirven para que el servidor obtenga información de las variables de estado del robot, tales como los sensores, la brújula, el GPS, la posición, la batería y el estado del robot.

La instrucción de sensores es utilizada para pedirle al robot la distancia registrada por cada sensor. La de brújula pide al robot el ángulo medido por la brújula. La instrucción de GPS pide la longitud y latitud registradas por el dispositivo receptor GPS. La instrucción de posición es utilizada para pedirle al robot su posición en longitud y latitud estimada por el conteo del giro de las llantas del mismo. La instrucción de batería pide el porcentaje de carga de la batería. La instrucción de estado pide al robot un byte de estado el cual contiene información como el tipo de movimiento que está realizando, si está en proceso de un salto de un punto a otro, si está mostrando información en la pantalla LCD y si el paquete ha sido entregado o recibido por el usuario.

3. Instrucciones de transferencia. Sirven para pasarle parámetros al robot, como el nombre de la oficina de donde se envió el paquete, el nombre de la oficina a donde va dirigido el paquete, el nombre de la gaveta en la que se encuentra el paquete y las coordenadas a donde se desea que vaya el robot.

La instrucción de oficina origen permite al servidor enviarle al robot el nombre de la oficina de donde se ha realizado el envío. La instrucción de oficina destino permite al servidor enviarle al robot el nombre de la oficina hacia dónde va dirigido el envío. La instrucción de número gaveta permite al servidor enviarle al robot el número de la gaveta en la que se encuentra el paquete enviado. La instrucción de coordenadas destino permite al servidor enviarle al robot la longitud y latitud a las cuales debe dirigirse para llegar al nodo u oficina deseados.

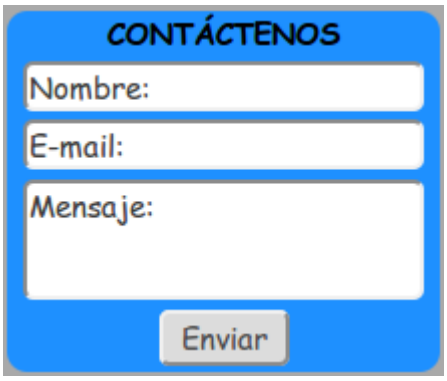
E. Resultados y discusión

La Figura 36 muestra la página principal del sitio, en la parte superior se encuentra el logotipo del proyecto, justo abajo se observa un menú con algunos aspectos de la página como los la galería de fotos, la simulación del algoritmo de navegación y el mapa del campus. En el centro de la página se observa el contenido principal, del lado derecho se encuentra un área para enviar mensajes al administrador y del lado derecho está el área de usuarios.

Figura 36. Ventana principal del sitio web

El área de contacto mostrada en la Figura 37, está hecha para que cualquier visitante pueda contactar al administrador del sitio para hacer algún comentario o para resolver cualquier duda relacionada con el proyecto; también se creó esta área para que si algún usuario tiene algún inconveniente con su cuenta o desee cerrarla, lo haga a través del administrador del sitio.

Figura 37. Área de contacto del sitio web



El área de usuarios permite a los usuarios registrados acceder a su cuenta ingresando su nombre de usuario (sobrenombre) y contraseña como se muestra en la Figura 38.

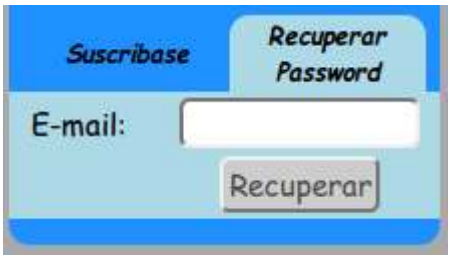
Figura 38. Área de usuario del sitio web

El área de suscripción permite suscribir nuevos usuarios al ingresar su nombre, apellido, E-mail, usuario (sobrenombre) y password (contraseña) como se muestra en la Figura 39. Para que la suscripción sea exitosa se debe elegir un sobrenombre inexistente en la base de datos.

Figura 39. Área de suscripción del sitio web

El área de recuperación de contraseña se muestra en la Figura 40, permite a los usuarios registrados recuperar su nombre de usuario y contraseña si así lo desearan con tan sólo ingresar el correo electrónico con el que fueron suscritos.

Figura 40. Área de recuperación de contraseña del sitio web



El área de cuenta de usuarios se observa en la Figura 41, muestra el nombre del usuario y le permite cerrar sesión si así lo desea, también permite que el usuario cambie su contraseña si así lo desea.

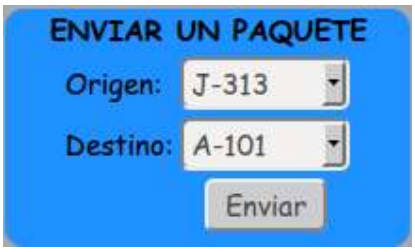
Figura 41. Área de cuenta de usuarios del sitio web

El área de estado de envíos se observa en la Figura 42, muestra el estado de los envíos que ha pedido el usuario.

Figura 42. Área de estado de envíos del sitio web

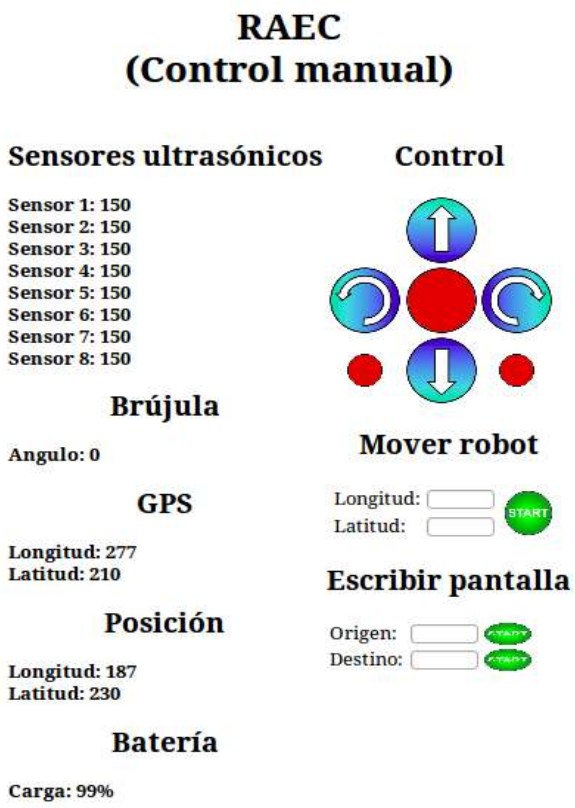
El área de envío de paquetes se muestra en la Figura 43, permite al usuario seleccionar la oficina de donde saldrá el paquete y la oficina a la que va dirigido, luego al hacer click en enviar se registra su envío en la base de datos para ser procesado por la rutina de control automático.

Figura 43. Área de envío de paquetes del sitio web



La ventana de control manual se muestra en la Figura 44, permite al administrador iniciar la conexión con el robot, así como ponerlo en control automático o control manual y activar o desactivar la actualización de las variables de estado del robot. También le permite mover y girar el robot, mandarle los nombres de las oficinas de origen y destino, y enviar al robot a una coordenada en específico.

Figura 44. Ventana de control manual del sitio web



IX. DESARROLLO DEL DISEÑO DE ALGORITMO DE NAVEGACIÓN

En este capítulo se explica el algoritmo diseñado para lograr que con el uso de ocho sensores ultrasónicos, una brújula, un dispositivo GPS y el conteo de pasos dados por el robot se logre movilizar al mismo desde un punto a otro en un mapa.

A. Obtención de posición a partir de los pasos

Para lograr que el robot logre movilizarse de un punto a otro es necesario que sepa el lugar al que se dirige y el lugar en el que se encuentra actualmente. Para obtener su posición actual, el robot carga una posición de origen cada vez que es encendido, sin embargo, cuando empieza a moverse su posición va variando.

Se utilizó un sistema de posicionamiento en coordenadas cartesianas, por lo tanto, cada vez que se mueve el robot es necesario calcular su desplazamiento tanto en la coordenada x como en la coordenada y.

Para mover el robot se utilizó un tipo de motor el cual permite girar una revolución cuando se dan n pasos, dichos motores permiten llevar un control de los pasos que se han dado, por lo tanto es posible calcular el arco de una llanta de radio r cuando el motor da k pasos según lo mostrado en el formulario 1.

Formulario 1. Cálculo de posición utilizando pasos del motor

$$\Delta x = (k / n) * 2\pi r * \cos(\theta)$$
$$\Delta y = (k / n) * 2\pi r * sen(\theta)$$
$$PosiciónX = PosiciónX + \Delta x$$
$$PosiciónY = PosiciónY + \Delta y$$

k : pasos dados por el motor
n : pasos por revolución
r : radio de las llantas
θ : ángulo del robot

B. Calibración de posición por GPS

El GPS viene de las siglas en inglés *Global Positioning System* (Sistema de Posicionamiento Global), es un sistema satelital que permite determinar la posición de un objeto receptor GPS en todo el mundo. Su precisión puede variar desde unos cuantos metros hasta algunos cuantos centímetros, y depende principalmente del número de satélites que el dispositivo receptor tiene enlazados. De un dispositivo GPS se puede obtener información como longitud, latitud, hora y altura.

Ahora bien, se implementó un dispositivo receptor GPS en el robot debido a que a pesar de que se puede obtener la posición de este utilizando los pasos dados por el motor, es cierto que la posición calculada de esta manera irá acumulando un error de posición eventualmente conforme el robot se va moviendo. Dichos errores pueden atribuirse a algún deslizamiento de las llantas mientras el robot avanza o a una colocación no simétrica de las mismas en el ensamblado de la estructura.

El receptor GPS necesita línea de vista con por lo menos tres satélites para dar una aproximación de la posición del mismo, sin embargo habrá ocasiones en las que no se tenga conexión con ningún satélite, o los satélites conectados sean muy pocos que la posición recibida de una mala aproximación, en esos casos las mediciones serán descartadas. Cuando se detecta una medición precisa se calibra la posición del robot como se muestra en el formulario 2.

Formulario 2. Calibración de la posición del robot utilizando GPS

$$PosiciónX = k_x * Longitud$$
$$PosiciónY = k_y * Latitud$$

k_x : metros por grado de longitud*

k_y : metros por grado de latitud*

* . para el área de la ciudad de Guatemala

C. Algoritmo de navegación

El algoritmo de navegación del robot está diseñado para funcionar en tres modos de navegación: directa, correctiva y evasiva. La navegación directa pretende llevar al robot a su destino en línea recta, asumiendo que no hay obstáculos en medio; la navegación evasiva entra en funcionamiento cuando el robot entra en cercanía con un objeto, su función es evadir obstáculos para prevenir colisiones; la navegación correctiva entra en funcionamiento cuando la cercanía entre un obstáculo y el robot es crítica a un grado tal que si no se toman acciones inmediatas para corregir el rumbo del robot habrá una colisión.

1. Navegación directa. La finalidad de este modo de navegación es llevar al robot a su destino trazando una línea recta entre ambos, por lo que es necesario calcular el ángulo que le permita llegar al destino solo moviéndose

hacia adelante. Una vez calculado el ángulo, el robot avanzará hasta estar a menos de 30 cm del destino o hasta encontrarse con algún obstáculo. Las formulas utilizadas para calcular el ángulo de navegación directa se muestran en el formulario 3.

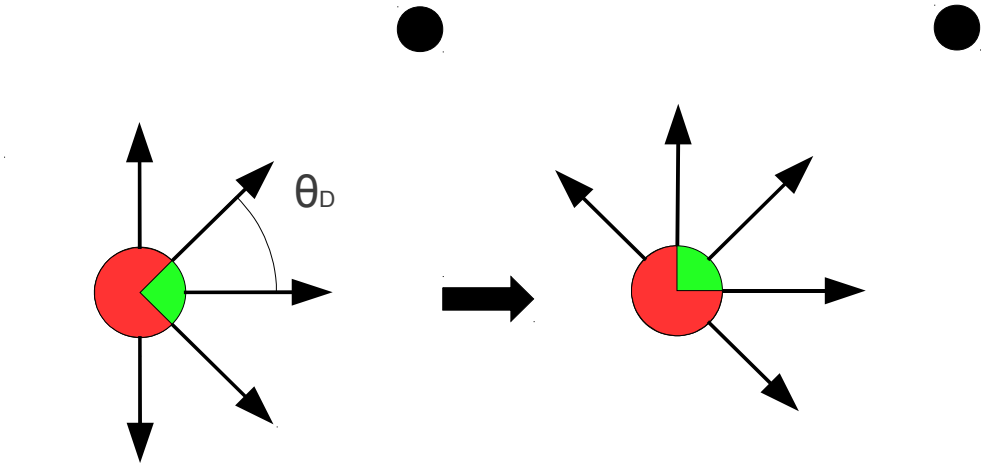
Formulario 3. Cálculos para obtener el ángulo directo

$$\Delta x = x_D - x_R$$
$$\Delta y = y_D - y_R$$
$$\theta_D = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$$

x_D: Posición x del destino
y_D: Posición y del destino
x_R: Posición x del robot
y_R: Posición y del robot
θ_D: Ángulo Directo

La Figura 45 muestra el comportamiento del robot en navegación directa, ya que originalmente el robot lleva una dirección, y luego de calcular el ángulo directo cambia de dirección para posicionarse orientado hacia su destino.

Figura 45. Ejemplo de navegación directa

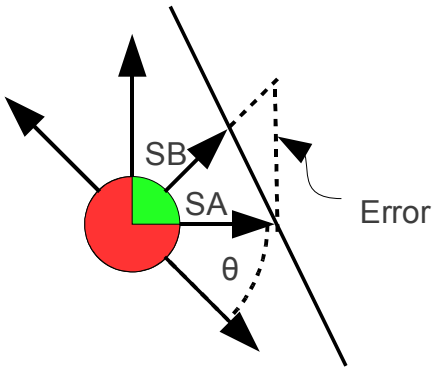


2. Navegación evasiva. Cuando el robot se encuentra en el modo de navegación directa, existe la posibilidad de que se encuentre con algún obstáculo que le impida seguir avanzando, en caso de que ocurra el evento mencionado, será necesario modificar el rumbo de navegación del robot para que de esta manera esquive dicho obstáculo.

Para que el robot se ponga en el modo de navegación evasiva será necesario que por lo menos uno de sus sensores registre una distancia menor a la llamada "DisEvasiva", la cual representa la separación mínima que debe tener el robot de los obstáculos para no tomar acciones evasivas.

El modo de navegación evasiva está basado en posicionar al robot paralelo a la superficie del obstáculo detectado y avanzar hasta esquivarlo. Para ello utiliza la medición de dos sensores, uno de ellos es el que activó el modo de navegación evasiva (lo llamaremos Sensor A) y el otro es uno de sus sensores adyacentes (lo llamaremos Sensor B) como se muestra en la Figura 46.

Figura 46. Ejemplo de activación de navegación evasiva



Para colocar el robot paralelo a la superficie es necesario calcular un ángulo de error, cuyo valor representa el ángulo que debe girar el robot para posicionar el Sensor A perpendicular a la superficie del obstáculo.

Utilizando la ley de cosenos podemos obtener el ángulo que existe entre el sensor A y la superficie del objeto (ángulo "b"), una vez calculado podemos encontrar el ángulo de error el cual es 90 – b. Los cálculos utilizados se muestran en el formulario 4.

Formulario 4. Cálculo del error en navegación evasiva

A = distancia sensor A + radio robot
B = distancia sensor B + radio robot
c = angulo sensor A - angulo sensor B

$$C = \sqrt{A^2 + B^2 - 2AB\cos(c)}$$

$$b = \cos^{-1}\left(\frac{A^2 + C^2 - B^2}{2AC}\right)$$

Si c > 0: $Error = 90 - b$
Si c < 0: $Error = b - 90$

Una vez corregido el ángulo de error, el sensor A quedara perpendicular a la superficie del obstáculo, sin embargo se desea que el robot quede paralelo a la superficie, lo que nos da como resultado un ángulo evasivo (llamaremos así al ángulo final al que debe estar el robot para quedar paralelo a la superficie), el cual se calcula como se muestra en el formulario 5. Cuando el robot haya quedado paralelo a la superficie avanzara hasta esquivar el obstáculo, una vez esquivado, el robot seguirá caminando una distancia de seguridad la cual disminuye la posibilidad de encontrarse de nuevo con el mismo obstáculo.

Formulario 5. Fórmulas para calcular el ángulo evasivo

$$\theta_E = \theta_R + \theta_A - Error$$
$$Cond_A = \theta_A > 90\ y\ \theta_A < 270$$
$$Cond_B = \theta_B > 90\ y\ \theta_B < 270$$

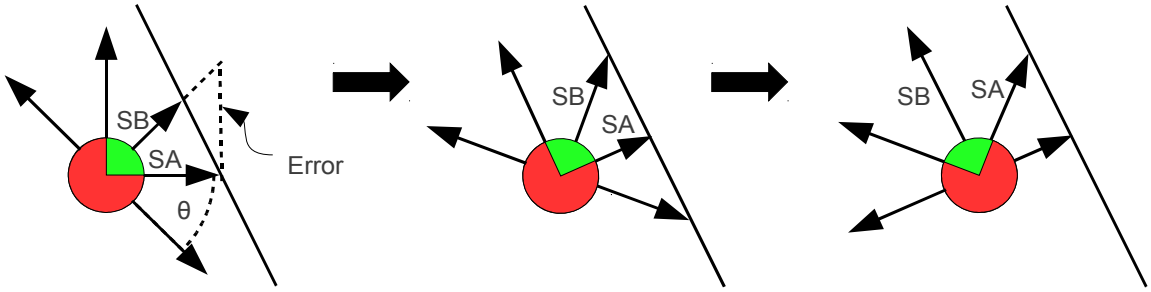
Si: $Cond_A$ ó $Cond_B$

$$\theta_E = \theta_R + \theta_A - (Error + 180)$$

θ_E : Ángulo Evasivo
θ_R : Ángulo del rumbo original del robot
θ_A : Ángulo del sensor A (respecto al robot)
θ_B : Ángulo del sensor B (respecto al robot)
* Todas las operaciones deben ser en módulo 360

La Figura 47 muestra el comportamiento del robot en navegación evasiva, se observa que el robot detecta la presencia de un obstáculo en dos de sus sensores, luego de calcular el ángulo evasivo se posiciona paralelo a la superficie del obstáculo y avanza hasta haberlo esquivado.

Figura 47. Evasión de obstáculo en navegación evasiva



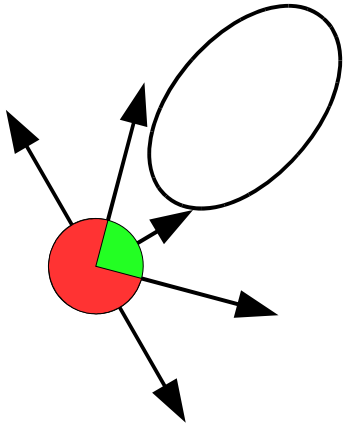
3. Navegación correctiva. Cuando el robot se encuentra en navegación directa puede encontrarse con algún obstáculo lo suficientemente angosto como para no ser detectado por dos sensores consecutivos, no entrando así en el modo de navegación evasiva, lo cual implicaría que el robot seguirá avanzando

hasta chocar contra el obstáculo, por esa razón es necesario utilizar un método que corrija el ángulo del robot para de esta forma evitar una colisión.

Para que el robot se ponga en el modo de navegación correctiva será necesario que por lo menos uno de sus sensores registre una distancia menor a la llamada "DisCorrectiva", la cual representa la separación crítica mínima que debe tener el robot de los obstáculos para no tomar acciones correctivas.

El modo de navegación correctiva está basado en posicionar al robot paralelo a la superficie del obstáculo detectado y avanzar hasta esquivarlo. Para ello utiliza la medición del sensor que ha registrado una distancia menor a "DisCorrectiva" como se muestra en la Figura 48.

Figura 48. Ejemplo de activación de navegación correctiva



Una vez activado el modo de navegación correctiva, el robot escanea la superficie del obstáculo encontrado, primero gira hacia la izquierda y va registrando las distancias medidas por el sensor, cuando la distancia empieza a aumentar el robot empieza a girar en dirección contraria hasta que vuelve a encontrar un incremento en la distancia censada (ver Figura 50). Una vez

realizado lo anterior, el robot se posiciona en el ángulo en el cual detectó la menor distancia del sensor. Si hubiera detectado más de una distancia mínima en diferentes ángulos, se posiciona en el ángulo promedio. Al realizar este posicionamiento se consigue colocar al sensor frontal perpendicular a la superficie del obstáculo.

El siguiente paso consiste en hacer al robot retroceder para guardar una distancia de seguridad del obstáculo, la cual llamaremos "Separación Correctiva". Una vez separado el robot, debe calcular el nuevo ángulo de su orientación como se muestra en el formulario 6, y avanzar hasta esquivarlo, una vez esquivado, el robot seguirá caminando una distancia de seguridad la cual disminuye la posibilidad de encontrarse de nuevo con el mismo obstáculo como se muestra en la Figura 50.

Formulario 6. Fórmulas para calcular el ángulo correctivo

$$\theta_1 = \theta_{RA} + \theta_S$$
$$\theta_2 = \theta_{RA} + \theta_S + 180$$
$$\Delta\theta_{11} = \theta_1 - \theta_R$$
$$\Delta\theta_{21} = \theta_2 - \theta_R$$
$$\Delta\theta_{12} = 360 - \theta_{11}$$
$$\Delta\theta_{22} = 360 - \theta_{21}$$

Si: $\Delta\theta_{12} > \Delta\theta_{11}$ entonces: $\Delta\theta_{12} = \Delta\theta_{11}$
Si: $\Delta\theta_{22} > \Delta\theta_{21}$ entonces: $\Delta\theta_{22} = \Delta\theta_{21}$
Si: $\Delta\theta_{12} \geq \Delta\theta_{22}$ entonces: $\Delta\theta_C = \Delta\theta_1$
Si: $\Delta\theta_{12} < \Delta\theta_{22}$ entonces: $\Delta\theta_C = \Delta\theta_2$

θ_C : Ángulo Correctivo
θ_R : Ángulo del rumbo original del robot
θ_{RA} : Ángulo del rumbo actual del robot
θ_S : Ángulo del sensor (respecto al robot)
* Todas las operaciones deben ser en módulo 360

Figura 49. Escaneo de obstáculo en navegación correctiva

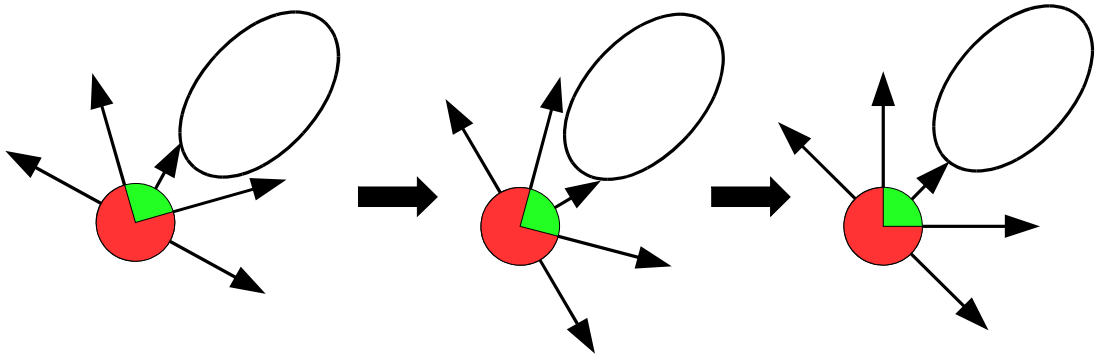
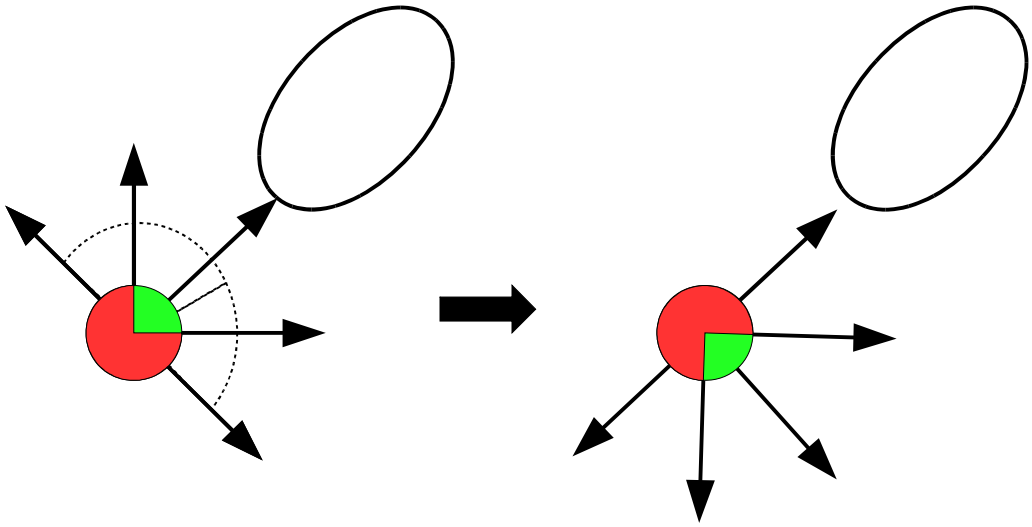


Figura 50. Separación y cambio de rumbo en navegación correctiva



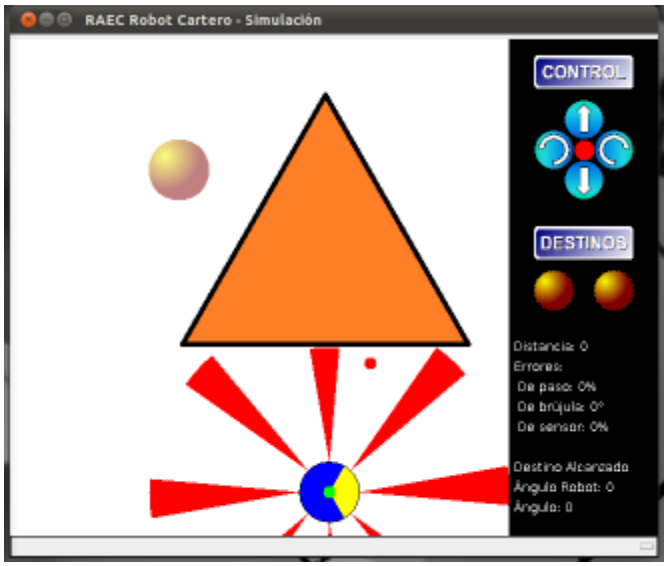
D. Simulación

Se realizó un programa en java con el propósito de probar el funcionamiento del algoritmo de navegación, para ello se construyó un entorno gráfico el cual permite controlar al robot simulado de forma manual o de forma automática. De igual manera se tomó en cuenta el efecto del ruido en la simulación para determinar qué valores de ruido son aceptables para permitir

que el robot llegue a su destino. Se agregó la opción de conectarse con el puerto serial para que de esta forma se puedan recibir comandos provenientes del circuito inalámbrico enviados por el servidor y corroborar así el funcionamiento completo.

1. Entorno gráfico. Este aspecto de la simulación se muestra en la Figura 51, está dividida principalmente en dos áreas: el área de observación; y el área de monitoreo y control.

Figura 51. Entorno gráfico de simulación



El área de observación permite ver en tiempo real la forma en la que se desenvuelve el robot a lo largo de la simulación, en esta área se muestran el mapa, el robot, los sensores, la posición calculada por el giro del motor y la posición obtenida por el GPS.

El área de monitoreo y control permite mover de forma manual al robot mediante los botones de adelante, atrás, izquierda y derecha; se tiene un control que permite agregar y borrar destinos del mapa y enviar al robot al destino

seleccionado; también posee un círculo rojo o verde el cual indica si existe conexión con el puerto serial o no. Otra característica de esta área es que puede mostrar la distancia que ha recorrido el robot a lo largo de la simulación, los porcentajes de error utilizados y el modo de simulación en el que se encuentra.

2. Consideración de errores. La simulación posee la característica de agregar error a la medición de los sensores, al ángulo medido por la brújula y al avance de los motores.

a. Error de paso. Error debido al deslizamiento de las llantas del motor, es el porcentaje de distancia que se desliza la llanta respecto al recorrido total.

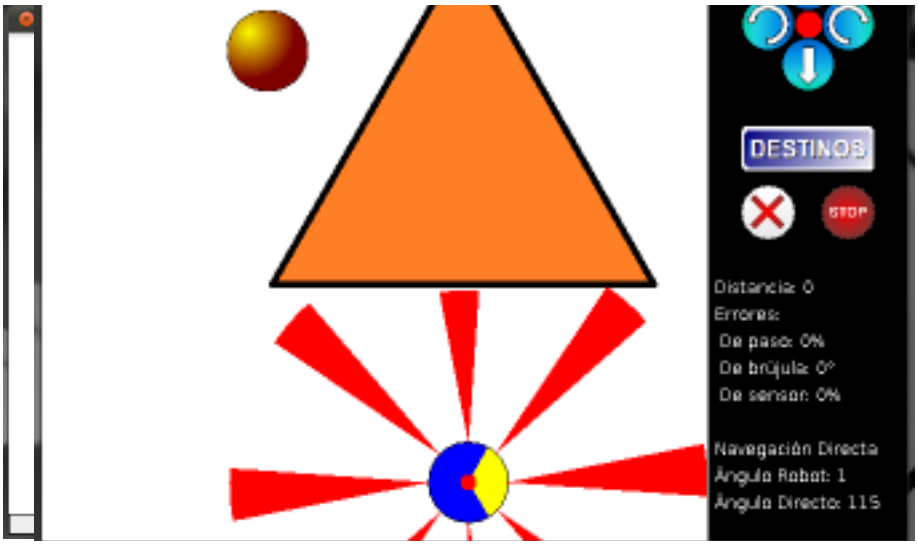
b. Error de brújula. Número de grados que la brújula mide erróneamente respecto al ángulo real.

c. Error de sensores. Porcentaje respecto a la medición máxima de los sensores que puede sumarse o restarse respecto a la distancia real.

E. Resultados y discusión

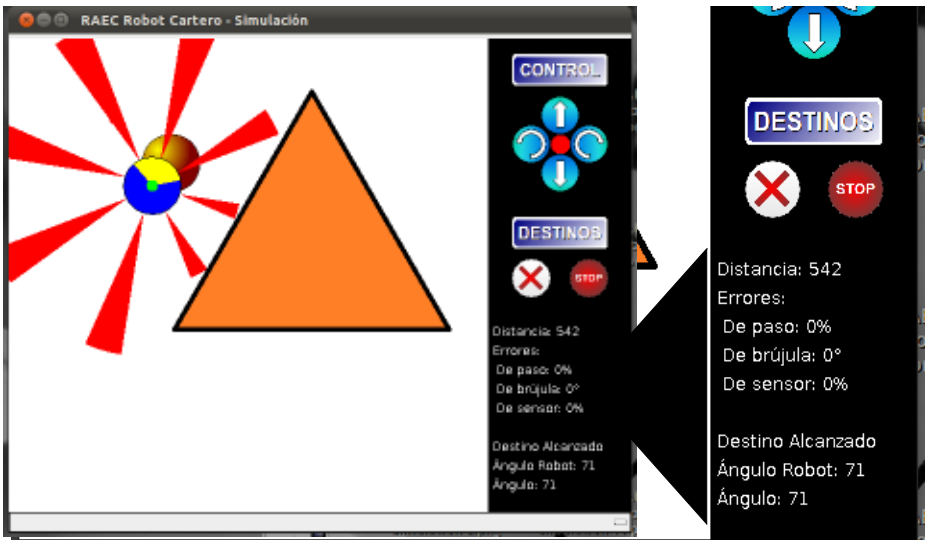
Se realizó una simulación con cero error y una distancia de seguridad de 10 cm, llevando de un punto a otro al robot, la simulación implementó el algoritmo de navegación descrito anteriormente.

Figura 52. Resultado simulación (inicio trayectoria)



La Figura 52 muestra la posición inicial del robot y el objetivo al que se desea llegar, se puede observar claramente la presencia de un obstáculo entre el robot y su destino. Finalmente, después de implementar los tres modos de navegación propuestos, el robot logra llegar al objetivo indicado como se muestra en la Figura 53.

Figura 53. Resultado simulación (fin trayectoria)



Como se pudo observar, gracias al algoritmo de navegación se pudo llevar al robot de un punto A hacia un punto B, con una distancia recorrida de 5.42 metros, utilizando cero error y una distancia de seguridad de 10 cm.

Para verificar el efecto del error en desempeño del robot se realizó una serie de simulaciones para determinar el impacto de los errores de paso (el error debido a deslizamiento de las llantas), el error de la brújula y el error de los sensores en la distancia recorrida desde el punto A al punto B.

El trayecto simulado es el mismo que el descrito anteriormente, solo que ahora incluyendo el error y utilizando una distancia de seguridad de 30 cm.

Tabla 19. Efectos del error en la distancia recorrida

Error			Distancia recorrida en cm					
Paso	Brújula	Sensor	Muestra 1	Muestra 2	Muestra 3	Muestra 4	Muestra 5	Promedio
0%	0°	0%	660	660	660	660	660	660
0%	0°	5%	724	528	506	520	514	558.4
0%	0°	10%	486	504	570	506	526	518.4
0%	5°	0%	534	496	502	686	576	558.8
0%	10°	0%	578	494	502	476	678	545.6
5%	0°	0%	702	706	716	618	770	702.4
10%	0°	0%	634	686	1168	3104	626	1243.6
5%	5°	5%	812	550	648	652	868	706
10%	10°	10%	1226	518	802	1418	1182	1029.2

Como se puede observar en la Tabla 19, la distancia recorrida con condiciones ideales (cero error) es de 660, sin embargo al introducir únicamente error de sensores de 5 y 10%, o solamente error de brújula de 5 y 10°, el promedio del recorrido no varía significativamente.

El error que afectó significativamente la distancia recorrida fue el error de paso, dado que este es un error acumulativo el cual va afectando la posición propia registrada por el robot.

Se puede observar que el promedio de muestras tomadas con 5 (por ciento o grados) en todos los tipos de error no difieren mucho de aquellas tomadas únicamente con 5% de error de paso, por lo cual se puede decir que el error de paso es el error que afecta más seriamente al algoritmo de navegación, y empeora aún mas conforme el robot recorre mayor distancia dado que se va acumulando el error en la posición registrada, por esa razón es necesaria la re calibración por medio del dispositivo receptor GPS.

X. INTEGRACIÓN DE MÓDULOS

A. Integración módulo de comunicación inalámbrica y módulo de interfaz por computadora

EL módulo de comunicación inalámbrica está formado por la interconexión de dos dispositivos, el primero es un dispositivo llamado SitePlayer el cual es un conversor de protocolo Ethernet a protocolo serial RS232. El segundo es el dispositivo XBee DigiMesh, el cual es un transmisor receptor inalámbrico utilizado para enviar la información hacia el robot por medio de la interfaz aire.

Una de las funciones del módulo de interfaz por computadora es manipular la información referente a los envíos almacenada en la base de datos del servidor, con esta información se determina las instrucciones que se deben enviar al robot para indicarle a que oficina debe acudir.

Los datos enviados por el servidor viajan a través de la red de datos interna de la universidad por medio de mensajes UDP los cuales van dirigidos específicamente al dispositivo SitePlayer, dicho dispositivo está configurado para convertir la información contenida en los mensajes UDP recibidos por el puerto 10001 a información serial del protocolo RS232 y viceversa.

Ya que la información se encuentra en el protocolo RS232 es necesario transmitirla por la interfaz aire, para ello es utilizado un dispositivo XBee DigiMesh el cual convierte las señales eléctricas a ondas electromagnéticas las cuales viajan a través del espacio libre para ser detectadas e interpretadas por el dispositivo XBee DigiMesh instalado en el robot.

B. Integración módulo de comunicación inalámbrica y módulo de navegación

Como se mencionó anteriormente, el robot posee un dispositivo XBee DigiMesh el cual recibe las señales electromagnéticas del aire y las convierte a señales eléctricas en protocolo serial con una velocidad de transferencia de 9600 baudios, dichas señales son recibidas por uno de los puertos UART que posee el microcontrolador DSPIC30F4011 utilizado para manejar la comunicación con el módulo inalámbrico.

C. Integración módulo de interfaz por computadora y módulo de navegación

Para que la información que se desea intercambiar pueda ser entendida tanto por el robot como por el servidor fue necesario elaborar una lista de instrucciones y comandos que permite el completo entendimiento de cada mensaje que se transmite.

Se definió un conjunto de tres tipos de instrucciones con las cuales se intercambia información entre el robot y el servidor, dichos tipos son: instrucciones de movimiento, instrucciones de solicitud e instrucciones de transferencia.

Desde la página del administrador se envió cada una de las instrucciones definidas hacia el robot y se desplegó la información recibida en pantalla en el caso de las instrucciones de solicitud. Las pruebas de movimiento y de transferencia lograron que el robot realizara los movimientos deseados con lo que se dio por completado la integración entre el módulo de interfaz por computadora y el módulo de navegación del robot.

XI. CONCLUSIONES Y RECOMENDACIONES

1. La red de sensores ultrasónicos de distancia permite la detección con niveles aceptables de error de objetos u obstáculos que se encuentren a una distancia de 1.5 m del robot.

2. Para que la brújula devuelva mediciones correctas es necesario calibrarla antes de utilizar el robot.

3. La medición de la brújula no se ve afectada por campos magnéticos provenientes de celulares y computadoras que se encuentren ubicados a una distancia mayor de 20 cm de la brújula.

4. El receptor GPS necesita conectarse por lo menos a 5 satélites para que se obtengan datos de posición con un error menor a 1.8 m.

5. El costo de construir el robot fue de Q. 11,268.26.

6. Con un horizonte de cinco años y con el salario mínimo actual en Guatemala el proyecto tiene una Tasa Interna de Retorno de 31% si el robot logra realizar veinte viajes diarios dentro del campus central de la Universidad del Valle de Guatemala.

7. Los beneficios financieros obtenidos al tener el robot en operación son directamente proporcionales al número de viajes diarios que realiza el robot. Si el robot sustituye un mayor número de viajes para correspondencia entonces el beneficio que obtiene la Universidad es mayor.

8. El aumento del salario de los trabajadores de la Universidad aumenta el beneficio obtenido por utilizar el robot porque el costo de los viajes que sustituye es mayor.

9. La optimización de los recorridos que realiza el robot para entregar correspondencia permite que los viajes tengan menor duración y esto permite que el robot realice un mayor número de viajes diariamente.

10. Se debe promover la utilización del robot por los empleados de la universidad para que el proyecto sea más rentable.

11. Utilizar en la menor medida posible el lenguaje de programación MicroC para microcontroladores PIC ya que a pesar de tener incluido varias herramientas o librerías, a la hora de compilar el programa, cada instrucción de MicroC no necesariamente es mapeada a una instrucción de lenguaje ensamblado. Esto puede afectar el rendimiento del programa del PIC.

12. Cuando se diseñe e implemente un protocolo de comunicación entre PICs considerar la pérdida de información en el canal.

13. Antes de realizar alguna compra de un equipo electrónico se recomienda buscar información adicional sobre ejemplos, foros o notas de aplicación, ya que la información que se presenta en la hoja de datos puede ser insuficiente.

14. Se recomienda hacer un estudio de mercado externo a la Universidad del Valle de Guatemala para analizar la factibilidad de establecer una empresa que fabrique y venda el robot.

15. Colocando el módulo inalámbrico estacionario en el laboratorio de ingeniería electrónica se puede obtener comunicación inalámbrica hasta el área de secretaría de la Universidad.

16. Se recomienda colocar el módulo inalámbrico estacionario en secretaría o en alguna oficina disponible del edificio A para aumentar el área de cobertura de la señal inalámbrica.

17. Utilizando una página web se permite a los usuarios registrados en la base de datos enviar paquetes entre las oficinas disponibles en la página.

18. Se recomienda agregar a la página web la opción de visualizar la posición del robot dentro del campus utilizando un mapa de la Universidad.

19. Un algoritmo de optimización de entregas permite que el robot recoja y entregue los paquetes más cercanos a su posición, disminuyendo así la distancia total recorrida.

20. Se recomienda mejorar el algoritmo de optimización de entregas dándole peso a los envíos según su prioridad, de esta manera se garantiza la entrega rápida de paquetes urgentes.

21. Un programa de simulación permite observar el comportamiento del robot dentro de un mapa, logrando así detectar errores, analizar y mejorar el algoritmo de navegación diseñado.

22. Se recomienda modificar la simulación para que entregue reportes de las rutas tomadas y los movimientos hechos por el robot, de esta forma se puede incrementar el análisis para optimizar el algoritmo de navegación.

23. El algoritmo con los modos de navegación directa, evasiva y correctiva permite al robot llegar de un punto a otro esquivando los obstáculos encontrados.

24. Se recomienda implementar una rutina que calcule la distancia que debe recorrer el robot después de haber esquivado un obstáculo con la finalidad de evitar una nueva obstrucción debida al mismo obstáculo.

25. El error que más afecta la eficiencia del algoritmo de navegación diseñado es el error provocado por deslizamiento en las llantas del robot debido a que genera un error acumulativo en la posición registrada por el mismo.

26. Se recomienda buscar métodos de recalibración que brinden mayor precisión que un receptor GPS.

27. Se recomienda que en futuras mejoras al proyecto se implemente una herramienta gráfica que permita a los visitantes de la página visualizar la posición en la que el robot se encuentra dentro del campus de la Universidad.

XII. BIBLIOGRAFÍA

[1]. Carletti J. (2009, May. 18) *Comunicación – Bus I2C, Descripción y Funcionamiento*. [En línea]. http://robots-argentina.com.ar/Comunicacion_busI2C.htm

[2]. Carmona A. (2011) Lap Race. [En línea]. http://lapracer.hostei.com/index.php?sitio=ayu

[3]. Cisco Systems, Inc. (2011). *Ethernet Technologies*. [En línea]. http://docwiki .cisco.com/wiki/Ethernet_Technologies

[4]. Club de la Mar. (2001). El Sistema GPS. [En línea]. http://www.clubdelamar.org/sistemagps.htm

[5]. Digi International Inc. (2011). *RF Basics*. [En línea]. http://www.digi.com/technology/rf-articles/rf-basics

[6]. Digi International Inc. (2011). *XBee DigiMesh 900 RF Modules*. [En línea]. http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-digimesh-900#specs

[7]. Esquit, C. *Curso de Instrumentación Electrónica*. Guatemala: Universidad del Valle de Guatemala, 2010.

[8]. Evaluación de costo beneficio. [En línea]. http://www.mitecnologico.com/Main/EvaluacionDelCostoBeneficio

[9]. Hübschmann S., Schneider (1996, Abr. 1) *Magnetoresistive Sensors, Principles of Operation and Applications.* [En línea]. www.diodes.com/_files/products_appnote_pdfs/zetex/an20.pdf

[10]. Kim J. *A Framework for Roadmap-based Navigation and Sector-based Localization of Mobile Robots*. Estados Unidos, Texas A&M University, 2004.

[11]. Kriechbaum, Kristopher. 2006. *Tools and algorithms for mobile robot navigation with uncertain localization*. California Institute of Technology. 134 págs.

[12]. Modelo OSI y TCP/IP. [En línea]. http://exa.unne.edu.ar/depar/areas/informatica/teleproc/Comunicaciones/Presentaciones_Proyector/ModeloOSIyTCPIP.pdf

[13]. National Instruments. (2011). *Introduction to RF & Wireless Communications Systems*. [En línea]. http://zone.ni.com/devzone/cda/tut/p/id/3541

[14]. Netmedia Inc. (2008). *SitePlayer Telnet – POE System – RoHS*. [En línea]. http://netmedia.com/siteplayer/telnet/sptpoe_u.html

[15]. NXP Semiconductors. (2000, Ene. 1). THE I2C-BUS SPECIFICATION VERSION 2.1 JANUARY 2000. [En línea]. www.nxp.com/documents/other/39340011.pdf

[16]. Redes. [En línea]. http://www.frm.utn.edu.ar/comunicaciones/redes.html

[17]. Robot. [En línea]. http://es.wikipedia.org/wiki/Robot

[18]. Tipos de robots. [En línea]. http://wiki.webdearde.com/index.php/TIPOS_DE_ROBOTS

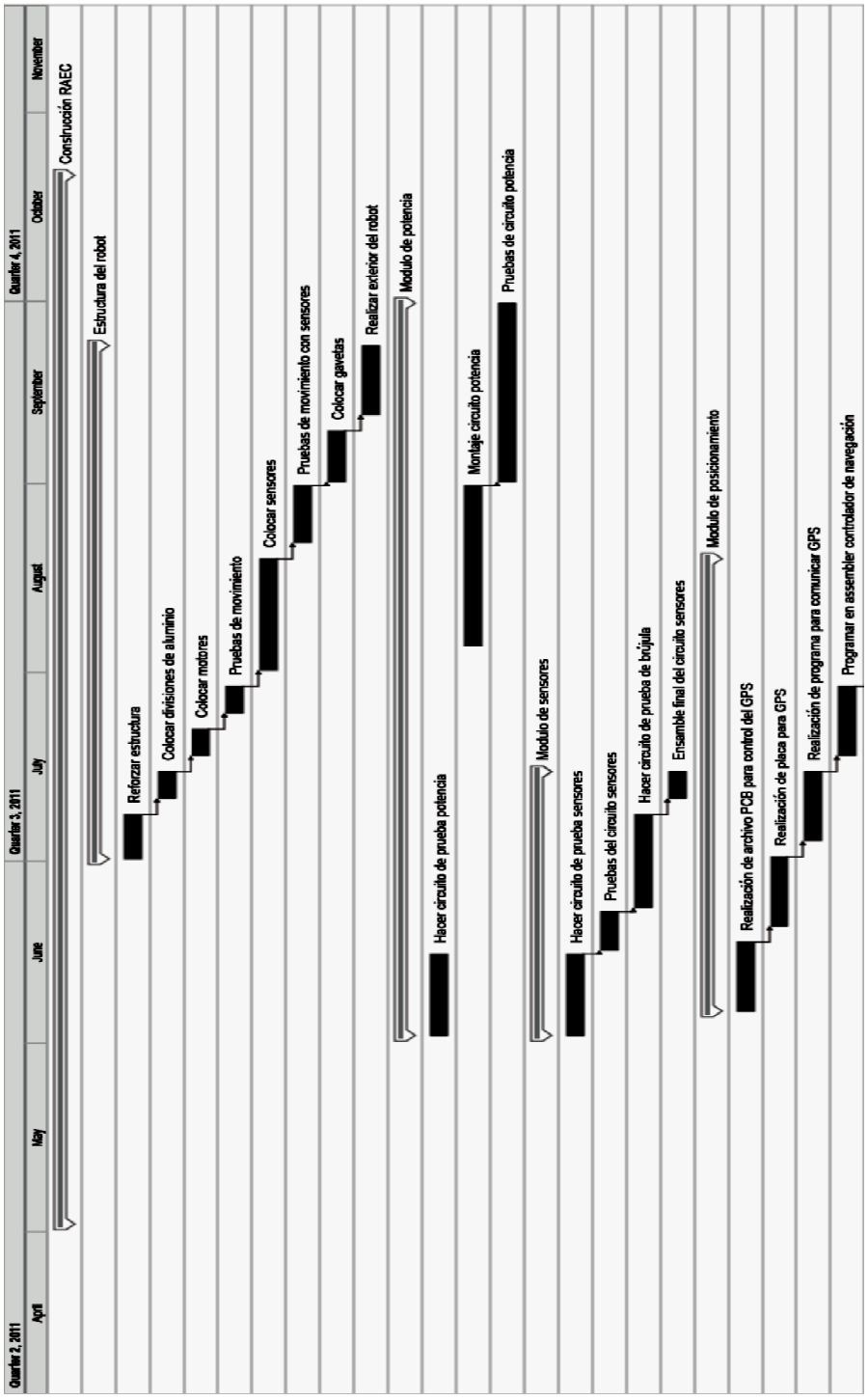
[19]. Tomasí, Wayne. 2003. *Sistemas de Comunicaciones Electrónicas*. 4ª ed. México, Pearson Educación. 935 págs.

[20]. Universitat Politècnica de Catalunya, Departamento de Arquitectura de Computadores. RS-232 y RS-485. [En línea]. http://studies.ac.upc.edu/EPSC/SED/Apuntes/RS-232%20Y%20RS-485.ppt

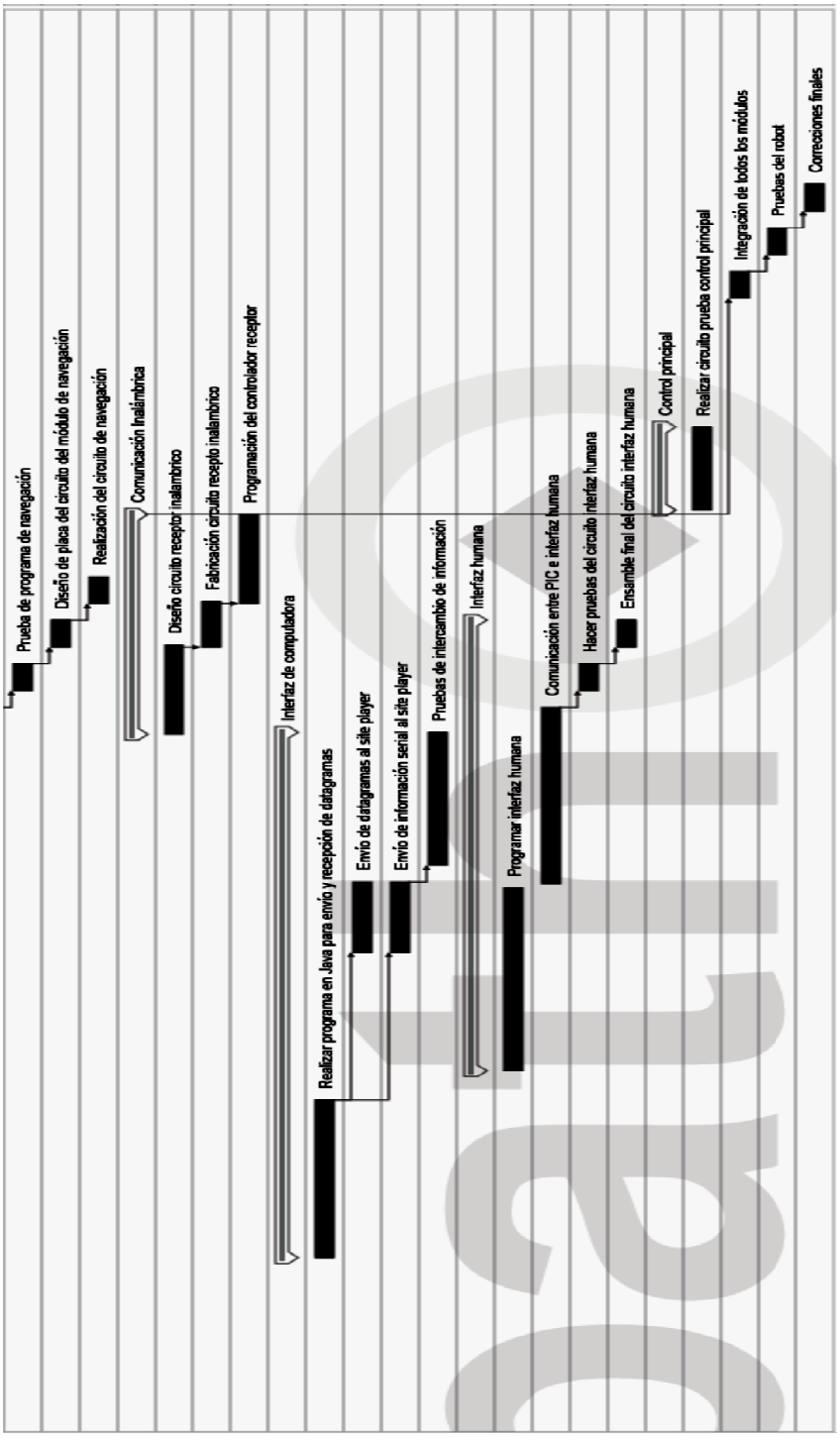
XIII. ANEXOS

A. Anexo 1. Programación de la fase de construcción

WBS	Task	Assignments	Complete	Duration	Start	Finish	Predecessors
1	Construcción RAEC		100%	125 days	2011-05-02 08:00	2011-10-21 16:00	
1.1	Estructura del robot	Sergio Cabrera	100%	61 days	2011-07-01 08:00	2011-09-23 16:00	
1.1.1	Reforzar estructura	Sergio Cabrera	100%	8 days	2011-07-01 08:00	2011-07-08 16:00	
1.1.2	Colocar divisiones de aluminio	Sergio Cabrera	100%	5 days	2011-07-11 08:00	2011-07-15 16:00	Reforzar estructura
1.1.3	Colocar motores	Sergio Cabrera	100%	5 days	2011-07-18 08:00	2011-07-22 16:00	Colocar divisiones de
1.1.4	Pruebas de movimiento	Sergio Cabrera	100%	5 days	2011-07-25 08:00	2011-07-29 16:00	Colocar motores
1.1.5	Colocar sensores	Sergio Cabrera	100%	15 days	2011-08-01 08:00	2011-08-19 16:00	Pruebas de movimiento
1.1.6	Pruebas de movimiento con sensores	Sergio Cabrera	100%	8 days	2011-08-22 08:00	2011-08-31 16:00	Colocar sensores
1.1.7	Colocar gavetas	Sergio Cabrera	100%	7 days	2011-09-01 08:00	2011-09-09 16:00	Pruebas de movimiento
1.1.8	Realizar exterior del robot	Sergio Cabrera	100%	10 days	2011-08-12 08:00	2011-08-23 16:00	Colocar gavetas
1.2	Modulo de potencia	Estuardo Sandoval	100%	87 days	2011-08-02 08:00	2011-09-30 16:00	
1.2.1	Hacer circuito de prueba potencia	Estuardo Sandoval	100%	10 days	2011-08-02 08:00	2011-08-15 16:00	
1.2.2	Montaje circuito potencia	Estuardo Sandoval	100%	19 days	2011-08-05 08:00	2011-08-31 16:00	
1.2.3	Pruebas de circuito potencia	Estuardo Sandoval	100%	22 days	2011-09-01 08:00	2011-09-30 16:00	Montaje circuito
1.3	Modulo de sensores	Alejandro Asensio	100%	32 days	2011-08-02 08:00	2011-07-15 16:00	
1.3.1	Hacer circuito de prueba sensores	Alejandro Asensio	100%	10 days	2011-08-02 08:00	2011-08-15 16:00	
1.3.2	Pruebas del cirouito sensores	Alejandro Asensio	100%	5 days	2011-08-16 08:00	2011-08-22 16:00	Hacer circuito de prueba
1.3.3	Hacer circuito de prueba de brújula	Alejandro Asensio	100%	12 days	2011-08-23 08:00	2011-07-08 16:00	Pruebas del circuito
1.3.4	Ensamble final del circuito sensores	Alejandro Asensio	100%	5 days	2011-07-11 08:00	2011-07-15 16:00	Hacer circuito de prueba
1.4	Modulo de posicionamiento	Alejandro Asensio	100%	56 days	2011-08-06 08:00	2011-08-19 16:00	
1.4.1	Realización de archivo PCB para control del GPS	Alejandro Asensio	100%	10 days	2011-08-06 08:00	2011-08-17 16:00	
1.4.2	Realización de placa para GPS	Alejandro Asensio	100%	10 days	2011-08-20 08:00	2011-07-01 16:00	Realización de archivo
1.4.3	Realización de programa para comunicar GPS	Alejandro Asensio	100%	10 days	2011-07-04 08:00	2011-07-15 16:00	Realización de placa
1.4.4	Programar en assembler controlador de navegación	Alejandro Asensio	100%	10 days	2011-07-18 08:00	2011-07-29 16:00	Realización de


Quarter 2, 2011
April
May
June
Quarter 3, 2011
July
August
September
Quarter 4, 2011
October
November
Construcción RAEC
Estructura del robot
Reforzar estructura
Colocar divisiones de aluminio
Colocar motores
Pruebas de movimiento
Colocar sensores
Pruebas de movimiento con sensores
Colocar gavetas
Realizar exterior del robot
Modulo de potencia
Hacer circuito de prueba potencia
Montaje circuito potencia
Pruebas de circuito potencia
Modulo de sensores
Hacer circuito de prueba sensores
Pruebas del circuito sensores
Hacer circuito de prueba de brújula
Ensamble final del circuito sensores
Modulo de posicionamiento
Realizacion de archivo PCB para control del GPS
Realizacion de placa para GPS
Realizacion de programa para comunicar GPS
Programar en assembler controlador de navegación

25	1.4.5	Prueba de programa de navegación	Alejandro Asensio	100%	5 days	2011-08-01 08:00	2011-08-05 16:00	Programar en
26	1.4.6	Diseño de placa del circuito del módulo de navegación	Alejandro Asensio	100%	5 days	2011-08-08 08:00	2011-08-12 16:00	Prueba de programa de
27	1.4.7	Realización del circuito de navegación	Alejandro Asensio	100%	5 days	2011-08-15 08:00	2011-08-19 16:00	Diseño de placa del
28	1.5	Comunicación Inalámbrica	Javier Mesalles	100%	26 days	2011-07-25 08:00	2011-08-29 16:00	
29	1.5.1	Diseño circuito receptor inalambrico	Javier Mesalles	100%	11 days	2011-07-25 08:00	2011-08-08 16:00	
30	1.5.2	Fabricación circuito recepto inalambrico	Javier Mesalles	100%	6 days	2011-08-08 08:00	2011-08-15 16:00	Diseño circuito receptor
31	1.5.3	Programación del controlador receptor	Javier Mesalles	100%	11 days	2011-08-15 08:00	2011-08-29 16:00	Fabricación circuito
32	1.6	Interfaz de computadora	Javier Mesalles	100%	61 days	2011-05-02 08:00	2011-07-25 16:00	
33	1.6.1	Realizar programa en Java para envío y recepción de datagramas	Javier Mesalles	100%	20 days	2011-05-02 08:00	2011-05-27 16:00	
34	1.6.2	Envío de datagramas al site player	Javier Mesalles	100%	10 days	2011-06-20 08:00	2011-07-01 16:00	Realizar programa en
35	1.6.3	Envío de información serial al site player	Javier Mesalles	100%	10 days	2011-06-20 08:00	2011-07-01 16:00	Realizar programa en
36	1.6.4	Pruebas de intercambio de información	Javier Mesalles	100%	16 days	2011-07-04 08:00	2011-07-25 16:00	Envío de información
37	1.7	Interfaz humana	Sergio Cabrera	100%	53 days	2011-06-01 08:00	2011-08-12 16:00	
38	1.7.1	Programar interfaz humana	Sergio Cabrera	100%	22 days	2011-06-01 08:00	2011-06-30 16:00	
39	1.7.2	Comunicación entre PIC e interfaz humana	Sergio Cabrera	100%	21 days	2011-07-01 08:00	2011-07-29 16:00	
40	1.7.3	Hacer pruebas del circuito interfaz humana	Sergio Cabrera	100%	5 days	2011-08-01 08:00	2011-08-05 16:00	Comunicación entre PIC
41	1.7.4	Ensamble final del circuito interfaz humana	Sergio Cabrera	100%	5 days	2011-08-08 08:00	2011-08-12 16:00	Hacer pruebas del
42	1.8	Control principal	Alejandro Asensio,	100%	10 days	2011-08-30 08:00	2011-09-12 16:00	
43	1.8.1	Realizar circuito prueba control principal	Alejandro Asensio,	100%	10 days	2011-08-30 08:00	2011-09-12 16:00	
44	1.9	Integración de todos los módulos	Alejandro Asensio,	100%	5 days	2011-10-03 08:00	2011-10-07 16:00	Comunicación
45	1.10	Pruebas del robot	Alejandro Asensio,	100%	5 days	2011-10-10 08:00	2011-10-14 16:00	Integración de todos los
46	1.11	Correcciones finales	Alejandro Asensio,	100%	5 days	2011-10-17 08:00	2011-10-21 16:00	Pruebas del robot


Prueba de programa de navegación
Diseño de placa del circuito del módulo de navegación
Realización del circuito de navegación
Comunicación Inalámbrica
Diseño circuito receptor inalambrico
Fabricación circuito recepto inalambrico
Programación del controlador receptor
Interfaz de computadora
Realizar programa en Java para envio y recepción de datagramas
Envio de datagramas al site player
Envio de información serial al site player
Pruebas de intercambio de información
Interfaz humana
Programar interfaz humana
Comunicación entre PIC e interfaz humana
Hacer pruebas del circuito interfaz humana
Ensamble final del circuito interfaz humana
Control principal
Realizar circuito prueba control principal
Integración de todos los módulos
Pruebas del robot
Correcciones finales

B. Anexo 2. Modelo de encuesta realizada

Megaproyecto de Robot Autónomo para Entrega de Correspondencia

Salir de esta encuesta

Por favor conteste las siguientes preguntas que servirán para un estudio del proyecto de graduación de un grupo de estudiantes de la Universidad del Valle. Todas sus respuestas serán anónimas y confidenciales. Gracias por su colaboración.

*** 1. Por favor indique el edificio y número de oficina en la que labora (por ejemplo A-101):**

*** 2. Seleccione el puesto que más se asemeja al que usted realiza:**

- Secretario/a
- Catedrático/a
- Director/a de departamento
- Personal administrativo
- Otro

*** 3. ¿Debe usted entregar correspondencia, papelería o paquetes pequeños en otra oficina dentro del campus central de la universidad?**

- Sí
- No

*** 4. Ingrese las oficinas a las que debe llevar correspondencia con mayor frecuencia (por ejemplo J-204):**

*** 5. Indique el número aproximado de viajes que hace al día a las oficinas mencionadas anteriormente:**

- Cero
- 1 a 2
- 2 a 4
- 4 a 6
- 6 a 8
- 8 o más

Listo

C. Anexo 3. Distancias entre oficinas

Numero	Origen	Destino	Distancia	Numero	Origen	Destino	Distancia
1	G-305	A-101	62.32	51	B-200 A	B-200	8.86
2	J-304	B-202	283.07	52	B-215	B-220	17.72
3	J-204	F-305	143.67	53	C-108	B-200	188.31
4	B-210	F-205	215.64	54	F-105	B-200	169.2
5	J-312	F-205	166.79	55	G-105	F-305	26.73
6	F-305	B-200	306.49	56	J-106	F-305	137.29
7	J-307	J-303	17.28	57	F-105	II1-107	161.21
8	C-107	F-205	198.52	58	B-210	C-111	249.99
9	F-105	G-305	53.12	59	J-304	F-305	137.29
10	C-222	F-105	153.65	60	J-204	F-107	176.77
11	J-300	J-304	11.52	61	J-312	B-200	337
12	II1-311	II2-304	103.55	62	F-305	F-106	266.96
13	C-214	J-309	351.81	63	C-222	F-305	188.31
14	F-314	J-210	166.79	64	J-300	F-301	137.29
15	F-205	A-101	105.16	65	II1-311	B-204	192.89
16	B-223	A-105	129.82	66	C-214	C-112	143.36
17	F-113	B-219	240.08	67	C-207 A	F-305	188.31
18	F-114	F-105	17.72	68	F-105	B-219	169.2
19	C-207 A	B-200	204.06	69	B-215	J-206	306.49
20	G-201	B-200	192.83	70	F-105	A-109	130.69
21	A-308	F-205	101.19	71	G-105	B-213	240.08
22	F-120	F-305	129.88	72	J-106	F-205	171.84
23	F-105	F-305	187.81	73	F-105	B-222	169.2
24	B-207	B-209	8.86	74	C-222	A-101	171.77
25	B-200 A	F-205	116.29	75	C-222	A-108	194.53
26	J-109	J-204	86.59	76	C-222	J-208	325.6
27	B-215	B-218	8.86	77	C-222	B-216	188.31
28	C-108	B-210	204.42	78	II1-311	F-105	140.77
29	F-105	F-205	123.42	79	II1-311	G-105	219.04
30	G-105	F-205	70.87	80	II1-311	F-205	139.9
31	J-106	F-105	166.79	81	II1-311	B-217	192.89
32	B-218	B-200	15.71	82	II1-311	F-305	173.38
33	F-105	C-222	158.82	83	II1-311	B-213	192.89
34	B-210	B-214	15.71	84	C-207 A	F-105	188.31
35	G-205	A-101	109.57	85	C-207 A	J-106	325.6

Continuación Anexo 3

Numero	Origen	Destino	Distancia	Numero	Origen	Destino	Distancia
36	G-305	F-105	100.37	86	C-207 A	J-304	365.09
37	J-304	F-105	201.34	87	C-207 A	J-204	302.48
38	J-204	F-205	166.79	88	C-222	II2-115	249.99
39	B-210	A-101	152.66	89	C-222	B-218	187.47
40	J-312	F-107	198.75	90	C-222	B-213	213.67
41	F-305	F-205	66.41	91	G-105	B-210	194.6
42	C-222	F-205	211.75	92	G-105	B-219	213.67
43	J-300	J-204	86.59	93	G-105	B-218	213.67
44	II1-311	II2-101	103.55	94	J-106	B-210	323.44
45	C-214	A-101	197.98	95	F-105	F-119	26.73
46	F-113	F-205	123.42				
47	F-114	B-200	169.2				
48	C-207 A	F-205	188.31				
49	A-308	B-200	175.42				
50	C-222	B-210	188.31				

D. Anexo 4. Detalle del costo de energía eléctrica

Detalle de cargos	Precios	Consumo (kwh)	Importe Q.
Cargo fijo por usuario	8.89		8.89
Costo de energía kwh	1.93	7	13.51
Ajuste solidario INDE	N.A.	0	0
Total Consumo kwh		7	0
Total Cargo			22.4
Total IVA			2.688
Tasa Municipal			2.24
Total a Pagar			27.328

E. Anexo 5. Incremento anual del salario mínimo en Guatemala en los últimos once años

Año	Salario	Incremento
2000	Q 23.85	
2001	Q 27.67	16%
2002	Q 30.00	8%
2003	Q 34.20	14%
2004	Q 39.67	16%
2005	Q 39.67	0%
2006	Q 43.64	10%
2007	Q 45.82	5%
2008	Q 48.50	6%
2009	Q 52.00	7%
2010	Q 56.00	8%
2011	Q 63.70	14%
Incremento promedio		9%

F. Anexo 6. Incremento anual del costo de la energía eléctrica en Guatemala en los últimos siete años

Fecha	Tarifa	Incremento
May-09	Q 1.23	
Ago-09	Q 1.37	11.38%
Nov-09	Q 1.46	6.57%
Feb-10	Q 1.76	20.55%
May-10	Q 1.94	10.23%
Ago-10	Q 1.74	-10.31%
Nov-10	Q 1.59	-8.62%
Feb-11	Q 1.57	-1.26%
May-11	Q 1.72	9.55%
Ago-11	Q 1.92	11.63%
Incremento Promedio		5.5%

G. Código fuente de microcontroladores

1. Código fuente del PIC Sensores

```
;*******************************************
; Universidad del Valle de Guatemala
; Mega Proyecto RAEC Robot Cartero
; Alejandro Asensio Lerma
; carné: 07441
;*******************************************
                        ; ------------------
                        ;| PIC de Sensores.|
                        ; ------------------
   list          p=16f887     ; list directive to define processor
   #include      <p16f887.inc>    ; processor specific variable definitions
          errorlevel 0
; '__CONFIG' directive is used to embed configuration data within .asm file.
   __CONFIG    _CONFIG1, _LVP_OFF & _FCMEN_ON & _IESO_OFF & _BOR_OFF & _CPD_OFF & _CP_OFF & _MCLRE_ON & _PWRTE_ON & _WDT_OFF & _INTRC_OSC_NOCLKOUT
   __CONFIG    _CONFIG2, _WRT_OFF & _BOR21V
GENERAL UDATA    0x20
          ; VARIABLES MAESTRO
          ESTADO              RES        1
          CONT_GPS            RES        1
          DIRECCION_GPS RES          1
          REC_SER             RES        1
          BANDERA             RES        1
          ACCION              RES        1
          AUX                 RES        1
          AUX_2               RES        1
CONT_AUX          RES        1
; VARIABLES BRUJULA
CONTADOR_DELAY RES        1
ENVIO_I2C         RES        1
DATO_BRUJ1  RES           1 ;0x2B
DATO_BRUJ2        RES        1 ;0x2C
DATO_AUX          RES        1
CON_BRUJ          RES        1
CONT_TMR0_BRUJ RES        1
; VARIABLES SENSORES
REG_SEN           RES        1
REG_SEN2          RES        1
AUX_TMR1L         RES        1
AUX_TMR1H         RES        1
S1                RES        1 ;0x34
S2                RES        1 ;0x35
S3                RES        1 ;0x36
S4                RES        1 ;0x37
S5                RES        1 ;0x38
S6                RES        1 ;0x39
S7                RES        1 ;0x3A
S8                RES        1 ;0x3B
; ENVIO SERIAL
ENV_SER           RES        1
; CONVERSION DE DATOS SENSORES
CEN               RES        1
DE_UN             RES        1
UN                RES        1
DEC_D1            RES        1
DEC_D2            RES        1
DEC_D3            RES        1
DEC_D4            RES        1
MEDICION_H        RES        1
MEDICION_L        RES        1
AUX0              RES        1 ;0x46
AUX1              RES        1 ;0x47
AUX2      RES               1 ;0x48
AUX3      RES        1 ;0x49
AUX4      RES        1 ;0x4A
LAT1      RES        1 ;0x4B
LAT2      RES        1 ;0x4C
LON1      RES        1 ;0x4D
```

```
        LON2        RES     1 ;0x4E
        INST_MOTORES    RES     1 ;0x4F
        BATERIA         RES     1 ;0x50
        ESTADO_PIC      RES     1 ;0x51
        CONTEO_PASOS1   RES     1 ;0x52
        CONTEO_PASOS2   RES     1 ;0x53
        AUX_RF          RES     1
        DIR_DAT_GPS     RES     1
        DIR_DAT_GPS_AUX RES     1
        CON_RF          RES     1
        DIRECCION_RF    RES     1
        DIR_DAT_RF_OR   RES     1
        DIR_DAT_RF_DE   RES     1
        DIR_DAT_RF_GA   RES             1
        CONT_AUX_RF             RES     1
        INSTRUCCION_RF          RES     1
        DIR_DAT_GUARDAR_RF      RES      1
        CONT_DAT_GUARDAR_RF     RES     1
        AUX_PORTB               RES     1
        CONT_TMR0_RF            RES     1
        CON_SEN                 RES     1
        PAG_AUX1                RES     1
        PAG_AUX2                RES     1
        W_TEMP      RES     1
        STATUS_TEMP RES     1
        PCLATH_TEMP RES     1
DATA_RF UDATA   0xA0
        ORIGEN1 RES             1 ;0xA0
        ORIGEN2 RES             1 ;0xA1
        ORIGEN3 RES             1 ;0xA2
        ORIGEN4 RES             1 ;0xA3
        ORIGEN5 RES             1 ;0xA4
        ORIGEN6 RES             1 ;0xA5
        ORIGEN7 RES             1 ;0xA6
        ORIGEN8 RES             1 ;0xA7
        ORIGEN9 RES             1 ;0xA8
        ORIGEN10        RES     1 ;0xA9
        DESTINO1 RES    1 ;0xAA
        DESTINO2        RES     1 ;0xAB
        DESTINO3        RES     1 ;0xAC
        DESTINO4        RES     1 ;0xAD
        DESTINO5        RES     1 ;0xAE
        DESTINO6        RES     1 ;0xAF
        DESTINO7        RES     1 ;0xB0
        DESTINO8        RES     1 ;0xB1
        DESTINO9        RES     1 ;0xB2
        DESTINO10       RES     1 ;0xB3
        GAVETA1         RES     1 ;0xB4
        GAVETA2         RES     1 ;0xB5
        GAVETA3         RES     1 ;0xB6
        GAVETA4         RES     1 ;0xB7
        GAVETA5         RES     1 ;0xB8
        GAVETA6         RES     1 ;0xB9
        GAVETA7         RES     1 ;0xBA
        GAVETA8         RES     1 ;0xBB
        GAVETA9         RES     1 ;0xBC
        GAVETA10        RES     1 ;0xBD
INICIO CODE 0x0000 ; processor reset vector
   GOTO    MAIN            ; go to beginning of program
; SERVICIO DE INTERRUPCION.
INTERRUPCIONES CODE     0x0004 ; interrupt vector
location
        ; SE GUARDA EL AMBIENTE DEL PROGRAMA
   MOVWF   W_TEMP
   MOVF    STATUS,W
   MOVWF   STATUS_TEMP
   MOVWF   PCLATH_TEMP
        BANKSEL PORTB
        MOVF    PORTB, W
        BANKSEL AUX_PORTB
        MOVWF   AUX_PORTB
        MOVF    AUX_PORTB
        BANKSEL PAG_AUX1
        CLRF    PAG_AUX1
        BTFSC   PCLATH_TEMP, 3
        BSF     PAG_AUX1, 0
        BTFSC   PCLATH_TEMP, 4
        BSF     PAG_AUX1, 1
```

```
        PAGESEL 0
        CLRF    PAG_AUX2
        BANKSEL PAG_AUX1
        MOVF    PAG_AUX1, W
        BTFSS   STATUS, Z
        GOTO    SIG_SET_PAG_1
        BSF     PAG_AUX2, 0
SIG_SET_PAG_1
        MOVLW   D'1' ; ADELANTE
        SUBWF   PAG_AUX1, W
        BTFSS   STATUS, Z
        GOTO    SIG_SET_PAG_2
        BSF             PAG_AUX2, 1
SIG_SET_PAG_2
        MOVLW   D'2' ; ADELANTE
        SUBWF   PAG_AUX1, W  ; REVISA SI LA PAG
ES 1.
        BTFSS   STATUS, Z
        GOTO    SIG_SET_PAG_3
        BSF     PAG_AUX2, 2
SIG_SET_PAG_3
        MOVLW   D'3' ; ADELANTE
        SUBWF   PAG_AUX1, W     BTFSS
        STATUS, Z
        GOTO    SIG_SET_PAG_4
        BSF     PAG_AUX2, 3
SIG_SET_PAG_4
        BANKSEL INTCON
        BTFSC   INTCON, 0
        GOTO    INT_SENSORES
        BANKSEL PIR1
        BTFSC   PIR1, SSPIF
        GOTO    INT_I2C
        BANKSEL PIR1
        BTFSC   PIR1, RCIF
        GOTO    RECIBIR_DATO_SERIAL
        BANKSEL INTCON
        BTFSC   INTCON, 2
        GOTO    INT_TMR0
        GOTO    SALIR_ISR
; INTERRUPCION DE RECEPCION DE DATO
SERIALRECIBIR_DATO_SERIAL
        BANKSEL PIR1
        BCF             PIR1, RCIF    ; APAGA
BANDERA DE USART
        BANKSEL RCREG
        MOVF    RCREG, W
        BANKSEL REC_SER
        MOVWF   REC_SER     ; GUARDA EL DATO
RECIBIDO
;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3 RF, 4
MOTORES
        BTFSC   ACCION, 0
        GOTO    GPS
        BTFSC   ACCION, 4
        GOTO    MOTORES
        BTFSC   ACCION, 5
        GOTO    LCD
        BTFSC   ACCION, 3
        GOTO    COM_RF
        GOTO    SALIR_ISR
GPS
        BANKSEL DIR_DAT_GPS_AUX
        MOVF    DIR_DAT_GPS_AUX, W
        MOVWF   FSR
        MOVF    CONT_GPS, W
        ADDWF   FSR,F
        BANKSEL REC_SER
        MOVF    REC_SER, W
        BANKSEL INDF
        MOVWF   INDF   ; PASA EL DATO RECIBIDO
A LA VARIABLE QUE CORRESPONDE
        BANKSEL CONT_GPS
        MOVF    CONT_GPS, W
        BTFSS   STATUS, Z
        GOTO    NO_PRIM_DATO_GPS
        MOVLW   D'240'
        BANKSEL REC_SER
```

```
        SUBWF   REC_SER,W       ;   VERIFICA   SI
RECIBIO LA DIRECCION ADECUADA.
        BTFSC   STATUS, Z       ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NO_PRIM_DATO_GPS
        CLRF    CONT_GPS
        GOTO    SALIR_ISR
NO_PRIM_DATO_GPS
        ; COMPARACION DE LA DIRECCION
        INCF    CONT_GPS
        MOVLW   D'4'
        SUBWF   CONT_GPS,W  ; VERIFICA SI YA SE
DIRECCIONARON TODAS LAS VARIABLES
        BTFSS   STATUS, Z       ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NO_LLEGO_4
        ; LLEGO A 4
        CLRF    CONT_GPS        ;   REINICIA   EL
CONTADOR
        BSF     BANDERA,  1  ;  INDICA  QUE  YA
RECIBIO 4 CARACTERES
        BCF     ESTADO, 0 ; YA TERMINO CON EL
GPS
        ;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3
RF, 4 MOTORES
        CLRF    ACCION   ; APAGA EL GPS
        BSF     ACCION, 3 ; TOCA RF
; APAGA TRANSIEVER
        BANKSEL PORTA
        MOVLW   B'01010101'
        MOVWF   PORTA
NO_LLEGO_4
        GOTO    SALIR_ISR
MOTORES
GOTO    SALIR_ISR
LCD
GOTO    SALIR_ISR
COM_RF
        BANKSEL CON_RF
        BTFSC   CON_RF, 3
        GOTO    ESPERA_DATOS_RF
        BTFSC   CON_RF, 2
        GOTO    ESPERA_AKN_INSTRUCCION_RF
        MOVLW   D'224'  ; 0xE0
        BANKSEL REC_SER
        SUBWF   REC_SER,W
        BTFSC   STATUS, Z
        GOTO    DIRECCIONADO_RF
        CLRF    CON_RF
        GOTO    SALIR_ISR
DIRECCIONADO_RF
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        BANKSEL CONT_AUX_RF
        CLRF    CONT_AUX_RF
        BSF     CON_RF, 1
        GOTO    SALIR_ISR
ESPERA_AKN_INSTRUCCION_RF
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF ; TIME OUT
        BANKSEL REC_SER
        SUBWF   REC_SER, W
        BANKSEL INSTRUCCION_RF
        MOVF    INSTRUCCION_RF, W
        BANKSEL REC_SER
        SUBWF   REC_SER, W
        BTFSC   STATUS, Z
        GOTO    AKN_INST_CORRECTO
        CLRF    CONT_TMR0_RF
        CLRF    ACCION
        BSF     ACCION, 4 ; HACER MOTORES
        BCF     ESTADO,   0   ;YA   SE   PUEDE
INTERACTUAR CON OTRO MÓDULO.
        MOVLW   D'5'
        MOVWF   INST_MOTORES
        CLRF    CON_RF
        BANKSEL INTCON
        BCF     INTCON, T0IE
```

```
        CLRF    DIR_DAT_GUARDAR_RF
        CLRF    CONT_DAT_GUARDAR_RF
        GOTO    SALIR_ISR
AKN_INST_CORRECTO
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        BSF             CON_RF, 3 ; AKN DE
INSTRUCCION CORRECTO
        CLRF    CONT_AUX_RF
        CLRF    AUX_RF
        GOTO    SALIR_ISR
ESPERA_DATOS_RF
        BANKSEL DIR_DAT_GUARDAR_RF
        MOVF    DIR_DAT_GUARDAR_RF,W
        MOVWF   FSR
        MOVF    CONT_AUX_RF, W
        ADDWF   FSR,F
        BANKSEL REC_SER
        MOVF    REC_SER, W
        MOVWF   INDF
        BANKSEL CONT_AUX_RF
        INCF    CONT_AUX_RF, F
        MOVF    CONT_DAT_GUARDAR_RF, W
        SUBWF   CONT_AUX_RF,W
        BTFSS   STATUS, Z
        GOTO    NO_LLEGO_AL_VALOR
        BTFSS   CON_RF, 6
        GOTO    FALTAN_INSTRUCCIONES_RF1
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        BANKSEL INSTRUCCION_RF
        CLRF    INSTRUCCION_RF
        ;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3
RF, 4 MOTORES
        CLRF    ACCION
        BSF     ACCION, 4 ; HACER MOTORES
        ;BSF    ACCION, 3 ; HACER RF
        BCF     ESTADO, 0 ; YA SE PUEDE
INTERACTUAR CON OTRO MODULO
        CLRF    CON_RF
        BANKSEL INTCON    ; INTERRUPCION DE
TMR0
        BCF     INTCON, T0IE ; BIT 5C
        BANKSEL PORTA
        MOVLW   B'01010101'
        MOVWF   PORTA
        GOTO    SALIR_ISR
FALTAN_INSTRUCCIONES_RF1
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        MOVLW   B'00000011' ; LISTO PARA ENVIAR
LA SIGUIENTE INSTRUCCION
        MOVWF   CON_RF
        GOTO    SALIR_ISR
NO_LLEGO_AL_VALOR
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        GOTO    SALIR_ISR
GOTO    SALIR_ISR
INT_TMR0
        BANKSEL INTCON
        ; APAGA INT DE TMR0
        BCF     INTCON, T0IF ; BIT 2
        BANKSEL TMR0
        CLRF    TMR0
;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3 RF, 4
MOTORES
        BTFSC   ACCION, 1
        GOTO    TIME_OUT_SENS
        BTFSC   ACCION, 2
        GOTO    TIME_OUT_BRUJ
        BTFSC   ACCION, 3
        GOTO    TIME_OUT_RF
        GOTO    SALIR_ISR
TIME_OUT_RF
        BANKSEL CONT_TMR0_RF
        INCF    CONT_TMR0_RF, F
        MOVLW   D'46'  ; 1 SEG.
```

```
        SUBWF   CONT_TMR0_RF,W  ; VERIFICA SI
YA SE DIERON TODOS LOS TIME OUTS DE RF
        BTFSS   STATUS, Z       ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    SALIR_ISR
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        CLRF    ACCION
        BSF     ACCION, 4 ; HACER MOTORES
        ;BSF    ACCION, 3 ; HACER RF
        BCF     ESTADO, 0 ; YA SE PUEDE
INTERACTUAR CON OTRO MÓDULO.
        MOVLW   D'5'
        MOVWF   INST_MOTORES
        CLRF    CON_RF
        BANKSEL INTCON
        BCF     INTCON, T0IE
        BANKSEL PORTA
        MOVLW   B'01010101'
        MOVWF   PORTA
        GOTO    SALIR_ISR
TIME_OUT_BRUJ
        INCF    CONT_TMR0_BRUJ, F
        BTFSS   CONT_TMR0_BRUJ, 1
        GOTO    SALIR_ISR
        BANKSEL CONT_TMR0_BRUJ
        CLRF    CONT_TMR0_BRUJ
        BSF     CON_BRUJ, 6
        BANKSEL INTCON
        BCF     INTCON, T0IE

        GOTO    SALIR_ISR
TIME_OUT_SENS
        BANKSEL REG_SEN
        CLRF    REG_SEN
        CLRF    BANDERA
        BANKSEL INTCON
        BCF     INTCON, T0IE
        BCF     INTCON, RBIE
        BCF     INTCON, T0IE
        CLRF    ACCION  BSF
        ACCION, 1; TOCA SENSORES
        ;BSF    ACCION, 2
        BCF     ESTADO, 0 ; YA SE PUEDE
INTERACTUAR CON OTRO MÓDULO.
GOTO    SALIR_ISR
INT_I2C
        BANKSEL PIR1
        BCF     PIR1, 3
        BSF     CON_BRUJ, 1
        GOTO    SALIR_ISR
INT_SENSORES
        BANKSEL PORTB
        MOVF    PORTB, W
        BANKSEL INTCON
        BCF     INTCON, 0
        BANKSEL PORTB
        BTFSC   PORTB, 0
        BSF     REG_SEN2, 0
        BTFSC   AUX_PORTB, 1
        BSF     REG_SEN2, 1
        BTFSC   AUX_PORTB, 2
        BSF     REG_SEN2, 2
        BTFSC   AUX_PORTB, 3
        BSF     REG_SEN2, 3
        BTFSC   AUX_PORTB, 4
        BSF     REG_SEN2, 4
        BTFSC   AUX_PORTB, 5
        BSF     REG_SEN2, 5
        BTFSC   PORTB, 6
        BSF     REG_SEN2, 6
        BTFSC   AUX_PORTB, 7
        BSF     REG_SEN2, 7
        BANKSEL CON_SEN
        MOVF    CON_SEN, F
        BTFSS   STATUS, Z
        GOTO    TMR1_INICIADO
        BANKSEL TMR1H
```

```
        CLRF    TMR1H
        CLRF    TMR1L
        BANKSEL T1CON
        MOVLW   B'10000001'  ; PREESCALER DE 1
        MOVWF   T1CON
        BANKSEL CON_SEN
        BSF     CON_SEN, 2
        CLRF    REG_SEN
        GOTO    SALIR_ISR
TMR1_INICIADO
        BANKSEL TMR1H
        MOVF    TMR1H, W
        MOVWF   AUX_TMR1H
        MOVF    TMR1L, W
        MOVWF   AUX_TMR1L
        RLF     AUX_TMR1H, F
        BCF     AUX_TMR1H, 0
        BTFSC   AUX_TMR1L, 7
        BSF     AUX_TMR1H, 0
VERIF_S1
        BANKSEL REG_SEN2
        BTFSS   REG_SEN2, 0
        GOTO    VERIF_S2
        BANKSEL PORTB
        BTFSC   PORTB, 0
        GOTO    VERIF_S2
        BANKSEL REG_SEN
        BTFSC   REG_SEN, 0
        GOTO    VERIF_S2
        BANKSEL AUX_TMR1H
        MOVF    AUX_TMR1H, W
        BANKSEL S1
        MOVWF   S1
        BANKSEL REG_SEN
        BSF             REG_SEN, 0
        BANKSEL REG_SEN2
        BCF     REG_SEN2, 0
VERIF_S2
        BANKSEL REG_SEN2
        BTFSS   REG_SEN2, 1
        GOTO    VERIF_S3
        BANKSEL PORTB
        BTFSC   PORTB, 1
        GOTO    VERIF_S3
        BANKSEL REG_SEN
        BTFSC   REG_SEN, 1
        GOTO    VERIF_S2
        BANKSEL AUX_TMR1H
        MOVF    AUX_TMR1H, W
        BANKSEL S2
        MOVWF   S2
        BANKSEL REG_SEN
        BSF     REG_SEN, 1
        BANKSEL REG_SEN2
        BCF     REG_SEN2, 1
VERIF_S3
        BANKSEL REG_SEN2
        BTFSS   REG_SEN2, 2
        GOTO    VERIF_S4
        BANKSEL PORTB
        BTFSC   PORTB, 2
        GOTO    VERIF_S4
        BANKSEL REG_SEN
        BTFSC   REG_SEN, 2
        GOTO    VERIF_S4
        BANKSEL AUX_TMR1H
        MOVF    AUX_TMR1H, W
        BANKSEL S3
        MOVWF   S3
        BANKSEL REG_SEN
        BSF     REG_SEN, 2
        BANKSEL REG_SEN2
        BCF     REG_SEN2, 2
VERIF_S4
        BANKSEL REG_SEN2
        BTFSS   REG_SEN2, 3
        GOTO    VERIF_S5
        BANKSEL PORTB
```

```
        BTFSC   PORTB, 3
        GOTO    VERIF_S5
        BANKSEL REG_SEN
        BTFSC   REG_SEN, 3
        GOTO    VERIF_S5
        BANKSEL AUX_TMR1H
        MOVF    AUX_TMR1H, W
        BANKSEL S4
        MOVWF   S4
        BANKSEL REG_SEN
        BSF     REG_SEN, 3
        BANKSEL REG_SEN2
        BCF     REG_SEN2, 3
VERIF_S5
        BANKSEL REG_SEN2
        BTFSS   REG_SEN2, 4
        GOTO    VERIF_S6
        BANKSEL PORTB
        BTFSC   PORTB, 4
        GOTO    VERIF_S6
        BANKSEL REG_SEN
        BTFSC   REG_SEN, 4
        GOTO    VERIF_S6
        BANKSEL AUX_TMR1H
        MOVF    AUX_TMR1H, W
        BANKSEL S5
        MOVWF   S5
        BANKSEL REG_SEN
        BSF     REG_SEN, 4
        BANKSEL REG_SEN2
        BCF     REG_SEN2, 4
VERIF_S6
        BANKSEL REG_SEN2
        BTFSS   REG_SEN2, 5
        GOTO    VERIF_S7
        BANKSEL PORTB
        BTFSC   PORTB, 5
        GOTO    VERIF_S7
        BANKSEL REG_SEN
        BTFSC   REG_SEN, 5
        GOTO    VERIF_S7
        BANKSEL AUX_TMR1H
        MOVF    AUX_TMR1H, W
        BANKSEL S6
        MOVWF   S6
        BANKSEL REG_SEN
        BSF     REG_SEN, 5
        BANKSEL REG_SEN2
        BCF     REG_SEN2, 5
VERIF_S7
        BANKSEL REG_SEN2
        BTFSS   REG_SEN2, 6
        GOTO    VERIF_S8
        BANKSEL PORTB
        BTFSC   PORTB, 6
        GOTO    VERIF_S8
        BANKSEL REG_SEN
        BTFSC   REG_SEN, 6
        GOTO    VERIF_S8
        BANKSEL AUX_TMR1H
        MOVF    AUX_TMR1H, W
        BANKSEL S7
        MOVWF   S7
        BANKSEL REG_SEN
        BSF     REG_SEN, 6
        BANKSEL REG_SEN2
        BCF     REG_SEN2, 6
VERIF_S8
SAL_VERIF_S
        ; VERIFICA SI YA TODOS BAJARON
        MOVLW   B'01111111'
        ;MOVLW  B'01000001'
        BANKSEL PORTB
        ANDWF   PORTB, W
        BTFSS   STATUS, Z
        GOTO    SALIR_ISR
        BANKSEL REG_SEN
        CLRF    REG_SEN
```

```
        CLRF    REG_SEN2
        CLRF    CON_SEN
        BANKSEL INTCON
        BCF     INTCON, T0IE    ; DESHABILITA INT DE AUX_PORTB 5
        BCF     INTCON, RBIE    ; DESHABILITA INT DE AUX_PORTB 3
        ;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3 RF, 4 MOTORES
        BANKSEL ACCION
        CLRF    ACCION  ; YA TERMINO DE MEDIR SENSORES
        BSF     ACCION, 1       ; TOCA MEDIR SENSORES
        ;BSF    ACCION, 2       ; TOCA MEDIR BRUJULA
        BANKSEL CON_BRUJ
        CLRF    CON_BRUJ
        BANKSEL ESTADO
        BCF     ESTADO, 0 ; YA SE PUEDE INTERACTUAR CON OTRO MÓDULO.
        GOTO    SALIR_ISR
SALIR_ISR
        ; SE RECUPERA EL AMBIENTE DEL PROGRAMA
   MOVF    PCLATH_TEMP,W   ; retrieve copy of PCLATH register
   MOVWF   PCLATH          ; restore pre-isr PCLATH register contents
   MOVF    STATUS_TEMP,W   ; retrieve copy of STATUS register
   MOVWF   STATUS          ; restore pre-isr STATUS register contents
   SWAPF   W_TEMP,F
   SWAPF   W_TEMP,W        ; restore pre-isr W register contents
        BTFSC   PAG_AUX2, 0
        PAGESEL 0
        BTFSC   PAG_AUX2, 1
        PAGESEL 1
        BTFSC   PAG_AUX2, 2
        PAGESEL 2
        BTFSC   PAG_AUX2, 3
        PAGESEL 3
   RETFIE          ; return from interrupt
;*****************************************************
;*****************************************************
MAIN
        ;**************************
        ; INIALIZACION DE VARIABLES
        ;**************************
        BANKSEL OSCCON
        ; SE SELECCIONA EL OSCILADOR INTERNO DE 4MHZ
        BSF     OSCCON, IRCF2
        BSF     OSCCON, IRCF1
        BCF     OSCCON, IRCF0
        ; HABILITA INTERRUPCION GLOBAL Y PERIFERICOS
        BSF     INTCON, 7
        BSF     INTCON, 6
        ; *********************************
        ;   CONFIGURACION DE PUERTOS
        ; *********************************
        BANKSEL TRISA
        CLRF    TRISA
        CLRF    TRISB
        CLRF    TRISC
        CLRF    TRISD
        MOVLW   B'11111000'
        MOVWF   TRISE
        BANKSEL PORTA
        CLRF    PORTA
        CLRF    PORTB
```

```
        CLRF    PORTC
        CLRF    PORTD
        BSF     PORTC, 5
        ;-------------------
        ; APAGA TRANSIEVER
        ;-------------------
        BANKSEL PORTA
        MOVLW   B'01010101'
        MOVWF   PORTA
        BANKSEL ANSEL
        CLRF    ANSEL
        BANKSEL ANSELH
        CLRF    ANSELH
        BANKSEL PIR1
        BCF     PIR1,  3
        CALL    INICIALIZAR_VARIABLES
        CALL    DELAY_50MS
        CALL    DELAY_50MS
        BANKSEL PORTC
        BCF     PORTC, 5
        BANKSEL TRISC ; DEBUG
        CLRF    TRISC; DEBUG
        BANKSEL PORTC; DEBUG
        CLRF    PORTC
;*****************************************
;          CICLO PRINCIPAL
;*****************************************
CICLO_PRINCIPAL
; SENSORES ---> BRUJULA ---> GPS ---> RF --->
MOTORES ---> SENSORES
        BANKSEL ESTADO
        BTFSC   ESTADO, 0
        GOTO    OTRAS_ACCIONES
        BANKSEL ACCION
        BTFSS   ACCION, 0
        GOTO    SIG_ACCION_1
        PAGESEL RQST_DATOS_GPS
        CALL    RQST_DATOS_GPS
        PAGESEL 0
SIG_ACCION_1
        BANKSEL ACCION
        BTFSS   ACCION, 1
        GOTO    SIG_ACCION_2
        PAGESEL LEER_SENSORES
        CALL    LEER_SENSORES
        PAGESEL 0
SIG_ACCION_2
        BANKSEL ACCION
        BTFSS   ACCION, 2
        GOTO    SIG_ACCION_3
        PAGESEL TOMAR_MEDICION_BRUJULA
        CALL    TOMAR_MEDICION_BRUJULA
        PAGESEL 0
SIG_ACCION_3
        BANKSEL ACCION
        BTFSS   ACCION, 3
        GOTO    SIG_ACCION_4
        PAGESEL DIRECCIONAR_RF
        CALL    DIRECCIONAR_RF
        PAGESEL 0
SIG_ACCION_4
        BANKSEL ACCION
        BTFSS   ACCION, 4
        GOTO    SIG_ACCION_5
        PAGESEL ENVIAR_INST_MOTORES
        CALL    ENVIAR_INST_MOTORES
        PAGESEL 0
SIG_ACCION_5
OTRAS_ACCIONES
        BANKSEL BANDERA
        BTFSS   BANDERA, 1
        GOTO    SIG_ACCION_6
        PAGESEL PASAR_DATOS_GPS_VALIDOS
        CALL    PASAR_DATOS_GPS_VALIDOS
        PAGESEL 0
SIG_ACCION_6
        BANKSEL CON_SEN
        BTFSS   CON_SEN, 3
```

```
        GOTO    SIG_ACCION_7
        PAGESEL OBTENER_DATO_SENSORES
        CALL    OBTENER_DATO_SENSORES
        PAGESEL 0
SIG_ACCION_7
        BANKSEL BANDERA
        BTFSS   BANDERA, 4
        GOTO    SIG_ACCION_8
        PAGESEL REINICIAR_COMUNICACION_GPS
        CALL    REINICIAR_COMUNICACION_GPS
        PAGESEL 0
SIG_ACCION_8
        BANKSEL CON_RF
        BTFSS   CON_RF, 1
        GOTO    SIG_ACCION_9
        PAGESEL ESTABLECER_COMUNICACION_RF
        CALL    ESTABLECER_COMUNICACION_RF
        PAGESEL 0
SIG_ACCION_9
        CALL    DELAY_50MS
        BANKSEL PORTE
        INCF    PORTE
        GOTO    CICLO_PRINCIPAL
; PAGINA 0
        #include    <Inicializar_Var.asm>
        #include    <Brujula.asm>
; PAGINA 1
PAGINA_1 CODE 0x0800
        #include    <GPS.asm>
        #include    <Motores.asm>
; PAGINA 2
PAGINA_2 CODE 0x1000
        #include    <RF_PIC.asm>
        #include    <Sensores.asm>
; PAGINA 3
   END                  ; directive 'end of program'
<Iniciarlizar_Var.asm>
;*****************************************
;       INICIALIZA VARIABLES DEL PROGRAMA
;*****************************************
INICIALIZAR_VARIABLES
        ; VARIABLES MAESTRO
        BANKSEL ESTADO
        CLRF    ESTADO
        CLRF    CONT_GPS
        CLRF    REC_SER
        CLRF    BANDERA
        CLRF    ACCION
        ;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3
RF, 4 MOTORES
        BSF     ACCION, 2; COMENZAR CON
BRUJULA
        CLRF    AUX
        CLRF    AUX2
        CLRF    CONT_AUX
        MOVLW   0XF0
        CLRF    DIRECCION_GPS
        MOVWF   DIRECCION_GPS
        CLRF    CON_RF
        MOVLW   0xE0
        MOVWF   DIRECCION_RF
        ; VARIABLES BRUJULA
        CLRF    CONTADOR_DELAY
        CLRF    ENVIO_I2C
        CLRF    DATO_BRUJ1
        CLRF    DATO_BRUJ2
        MOVLW   0x44
        MOVWF   DATO_BRUJ1
        MOVLW   0x18
        MOVWF   DATO_BRUJ2
        CLRF    DATO_AUX
        CLRF    CON_BRUJ
        CLRF    CONT_TMR0_BRUJ
        CLRF    CONT_TMR0_RF
        ; VARIABLES SENSORES
        CLRF    REG_SEN
        CLRF    REG_SEN2
        CLRF    AUX_TMR1L
```

```
CLRF    AUX_TMR1H
CLRF    S1
CLRF    S2
CLRF    S3
CLRF    S4
CLRF    S5
CLRF    S6
CLRF    S7
CLRF    S8
; ENVIAR SERIAL
CLRF    ENV_SER
; ENVIAR SENSORES SERIAL
CLRF    CEN
CLRF    DE_UN
CLRF    UN
CLRF    DEC_D1
CLRF    DEC_D2
CLRF    DEC_D3
CLRF    DEC_D4
CLRF    MEDICION_H
CLRF    MEDICION_L
CLRF    AUX0
CLRF    AUX1
CLRF    AUX2
CLRF    AUX3
CLRF    AUX4
CLRF    LAT1
CLRF    LAT2
CLRF    LON1
CLRF    LON2
MOVLW   D'1'
MOVWF   AUX1
MOVLW   D'2'
MOVWF   AUX2
MOVLW   D'3'
MOVWF   AUX3
MOVLW   D'4'
MOVWF   AUX4
MOVLW   0x4B
MOVWF   DIR_DAT_GPS
MOVLW   0xAA
MOVWF   DIR_DAT_RF_OR
MOVLW   0xB4
MOVWF   DIR_DAT_RF_DE
MOVLW   0xBE
MOVWF   DIR_DAT_RF_GA
CLRF    CONT_AUX_RF
CLRF    INSTRUCCION_RF
CLRF    DIR_DAT_GUARDAR_RF
CLRF    CONT_DAT_GUARDAR_RF
CLRF    INST_MOTORES
CLRF    BATERIA
CLRF    ESTADO_PIC
MOVLW   B'11110001'
MOVWF   ESTADO_PIC
CLRF    CONTEO_PASOS1
CLRF    CONTEO_PASOS2
CLRF    AUX_RF
CLRF    AUX_PORTB
CLRF    CON_SEN
CLRF    PAG_AUX1
CLRF    PAG_AUX2
CLRF    REG_SEN2
BANKSEL ORIGEN1
CLRF    ORIGEN1
CLRF    ORIGEN2
CLRF    ORIGEN3
CLRF    ORIGEN4
CLRF    ORIGEN5
CLRF    ORIGEN6
CLRF    ORIGEN7
CLRF    ORIGEN8
CLRF    ORIGEN9
CLRF    ORIGEN10
CLRF    DESTINO1
CLRF    DESTINO2
CLRF    DESTINO3
CLRF    DESTINO4
```

```
        CLRF    DESTINO5
        CLRF    DESTINO6
        CLRF    DESTINO7
        CLRF    DESTINO8
        CLRF    DESTINO9
        CLRF    DESTINO10
        CLRF    GAVETA1
        CLRF    GAVETA2
        CLRF    GAVETA3
        CLRF    GAVETA4
        CLRF    GAVETA5
        CLRF    GAVETA6
        CLRF    GAVETA7
        CLRF    GAVETA8
        CLRF    GAVETA9
        CLRF    GAVETA10
        BANKSEL CONT_AUX_RF
        RETURN
;*****************************************
;       CONFIGURACION DEL PUERTO SERIAL
;*****************************************
CONFIG_SERIAL_9600
        BANKSEL TRISC
        BCF     TRISC, 6
        BSF     TRISC, 7
        BANKSEL TXREG
        CLRF    TXREG
        BANKSEL TXSTA
        BCF     TXSTA,SYNC      ; BIT SYNC
        BSF     TXSTA,BRGH      ; BIT BRGH
        BANKSEL RCSTA
        BSF     RCSTA,SPEN      ; BIT SPEN
        BANKSEL BAUDCTL
        BCF     BAUDCTL,BRG16   ; BIT BRG16
        BANKSEL SPBRGH
        CLRF    SPBRGH
        MOVLW   D'25'
        MOVWF   SPBRG
        BANKSEL TXSTA
        BSF     TXSTA,TXEN      ; BIT TXEN
        BANKSEL INTCON
        BSF     INTCON, PEIE ; HABILITA
INTERRUPCION DEL PERIFERICO DEL SERIAL
        BANKSEL PIE1
        BCF     PIE1,TXIE    ;    DESHABILITA
INTERRUPCION DE TRANSMISION
        BSF     PIE1,RCIE    ;    HABILITA
INTERRUPCION DE RECEPCION
        BANKSEL RCSTA
        BSF     RCSTA, CREN     ; HABILITA LA
RECEPCION
        BANKSEL TXREG
        CLRF    TXREG
        CALL    DELAY_5MS
        RETURN
;*****************************************
;       ENVIA UN PAQUETE POR EL PUERTO SERIAL
;*****************************************
ENV_CAR_SERIAL
        BANKSEL ENV_SER
        MOVF    ENV_SER,W
        BANKSEL TXREG
        MOVWF   TXREG
        BANKSEL PIR1
        BTFSS   PIR1,TXIF
        GOTO    $-1
        RETURN
;*****************************************
; GENERAL: DELAY DE 5 mS
;*****************************************
DELAY_5MS
        CALL DELAY_1MS
        CALL DELAY_1MS
        CALL DELAY_1MS
        CALL DELAY_1MS
        CALL DELAY_1MS
        RETURN
;*****************************************
```

```
; GENERAL: DELAY DE 1 mS
;*****************************************
DELAY_1MS
        CALL DELAY_05MS
        CALL DELAY_05MS
        RETURN
;*****************************************
; GENERAL: DELAY DE 0.5 mS
;*****************************************
DELAY_05MS
        CLRF    CONTADOR_DELAY
DELAY_D
        INCF    CONTADOR_DELAY
        MOVLW   D'82'
        SUBWF   CONTADOR_DELAY,W
        BTFSS   STATUS, 2
        GOTO    DELAY_D
        CLRF    CONTADOR_DELAY
        NOP
        NOP
        RETURN
```

<Brujula.asm>

```
;*****************************************
;       BRUJULA: TOMA UNA MEDICION DE LA BRUJULA
;*****************************************
TOMAR_MEDICION_BRUJULA
        BANKSEL ESTADO
        BSF     ESTADO, 0 ; INTERACTUANDO CON BRUJULA.
        BANKSEL CON_BRUJ
        CLRF    CON_BRUJ
        BANKSEL DATO_BRUJ1
        CLRF    DATO_BRUJ1
        CLRF    DATO_BRUJ2
        ; CONFIGURACION DEL TMR0
        MOVLW   B'11010111'
        BANKSEL OPTION_REG
        MOVWF   OPTION_REG
        BANKSEL TMR0
        CLRF    TMR0
        BANKSEL INTCON    ;  INTERRUPCION  DE TMR0
        BSF     INTCON, GIE
        BSF     INTCON, T0IE ; BIT 5
        BANKSEL CONT_TMR0_BRUJ
        CLRF    CONT_TMR0_BRUJ
        CALL    INICIALIZAR_I2C
        CALL    SEC_START  ; SECUENCIA START
        MOVLW   0x42            ;DIRECCION  DEL DISPOSITIVOS
        BANKSEL ENVIO_I2C
        MOVWF   ENVIO_I2C
        CALL    WR_I2C
        MOVLW   0x41  ; HACER MEDICION
        BANKSEL ENVIO_I2C
        MOVWF   ENVIO_I2C
        CALL    WR_I2C
        CALL    SEC_STOP  ; SECUENCIA STOP
        CALL    SEC_START  ; SECUENCIA START
        MOVLW   0x43  ; LEER LOS DATOS MEDIDOS
        BANKSEL ENVIO_I2C
        MOVWF   ENVIO_I2C
        CALL    WR_I2C
        ; DELAY DE 6000 uS O 6 mS
        CALL DELAY_5MS
        CALL DELAY_1MS
        CALL    RD_I2C_1
        CALL    RD_I2C_2
        CALL    SEC_STOP ; SECUENCIA STOP
        CALL    APAGAR_I2C
        BANKSEL CON_BRUJ
        CLRF    CON_BRUJ
        BANKSEL INTCON    ;  INTERRUPCION  DE TMR0
        BCF     INTCON, T0IE ; BIT 5C
        BANKSEL ESTADO
```

```
        CLRF    BANDERA
        BCF     ESTADO, 0 ; YA TERMINO CON LA BRUJULA
        ;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3 RF, 4 MOTORES
        CLRF    ACCION  ; APAGA LA BRUJULA
        BSF     ACCION, 3 ; TOCA RF
        RETURN
;*****************************************
;       BRUJULA: INICIALIZA EL I2C A 100KHz
;*****************************************
INICIALIZAR_I2C
        BANKSEL PIE1
        BSF     PIE1,   SSPIE   ;   HABILITA INTERRUPCION DE MSSP
        BANKSEL SSPADD      ;   SELECCIONA EL BAUDRATE DE 100 KHz
        MOVLW   D'10'
        MOVWF   SSPADD
        BANKSEL SSPMSK
        MOVWF   SSPMSK
        BANKSEL TRISC   ; PONE LOS PINES DEL MSSP COMO OPEN SOURCE
        BSF     TRISC, 3
        BSF     TRISC, 4
        BANKSEL SSPSTAT
        BSF     SSPSTAT,    SMP ;   SLEW    RATE    CONTROL DISABLED FOR STANDARD SPEED MODE (100KHz)
        BANKSEL SSPCON
        BSF     SSPCON, SSPEN ; HABILITA LOS PINES SDA Y SCL
        BSF     SSPCON, 4 ; HABILITA EL CLK
        BSF     SSPCON,  3  ;   SELECCIONA  EL  MASTER MODE DEL MSSP EN I2C
        BCF     SSPCON, 2
        BCF     SSPCON, 1
        BCF     SSPCON, 0
        RETURN
;*****************************************
;       BRUJULA: APAGA EL MSSP CON I2C
;*****************************************
APAGAR_I2C
        BANKSEL SSPADD
        CLRF    SSPADD
        BANKSEL SSPMSK
        CLRF    SSPMSK
        BANKSEL TRISC   ; PONE LOS PINES DEL MSSP COMO SALIDAS
        BCF     TRISC, 3
        BCF     TRISC, 4
        BANKSEL SSPSTAT
        CLRF    SSPSTAT
        BANKSEL SSPCON
        CLRF    SSPCON
        BANKSEL SSPCON2
        CLRF    SSPCON2
        RETURN
;*****************************************
;       BRUJULA: SECUENCIA START
;*****************************************
SEC_START
        BANKSEL PIR1
        BCF     PIR1, 3
        BANKSEL SSPCON2
        BSF     SSPCON2,    SEN     ;   MANDA CONDICION DE START 1
        BANKSEL CON_BRUJ
        BTFSC   CON_BRUJ, 6 ; HAY TIME OUT?
        GOTO    SALIR_START_I2C
        BANKSEL CON_BRUJ
        BTFSS   CON_BRUJ, 1
        GOTO    $-5
SALIR_START_I2C
        BANKSEL CON_BRUJ
        BCF     CON_BRUJ,   1       ;   APAGA   LA  INTERRUPCION
        RETURN
```

```
;*******************************************
;       BRUJULA: SECUENCIA STOP
;*******************************************
SEC_STOP
        BANKSEL PIR1
        BCF     PIR1,   3       ;   APAGA   LA
INTERRUPCION
        BANKSEL SSPCON2
        BSF   SSPCON2, 2
        BANKSEL CON_BRUJ
        BTFSC   CON_BRUJ, 6 ; HAY TIME OUT?
        GOTO    SALIR_STOP_I2C
        BANKSEL CON_BRUJ
        BTFSS   CON_BRUJ, 1
        GOTO    $-5
SALIR_STOP_I2C
        BANKSEL CON_BRUJ
        BCF     CON_BRUJ,   1       ;   APAGA   LA
INTERRUPCION
        RETURN
;*******************************************
;       BRUJULA: ENVIA EN EL MSSP EN I2C EL
CONTENIDO DE ENVIO_I2C
;*******************************************
WR_I2C
        BANKSEL CON_BRUJ
        BCF     CON_BRUJ,   1       ;   APAGA   LA
INTERRUPCION
        BANKSEL ENVIO_I2C
        MOVF    ENVIO_I2C, W
        BANKSEL SSPBUF
        MOVWF   SSPBUF      ;   LOAD    AL  DATO    A
MANDAR
        BANKSEL CON_BRUJ
        BTFSC   CON_BRUJ, 6 ; HAY TIME OUT?
        GOTO    SALIR_WRITE_I2C
        BANKSEL SSPSTAT
        BTFSC   SSPSTAT, 2   ; VERIFICA QUE SE
TERMINE DE TRANSMITIR
        GOTO    $-5
        BANKSEL CON_BRUJ
        BTFSC   CON_BRUJ, 6 ; HAY TIME OUT?
        GOTO    SALIR_WRITE_I2C
        BANKSEL CON_BRUJ
        BTFSS   CON_BRUJ, 1     ; VERIFICA QUE
YA HAYA SIDO LA INTERRUPCION
        GOTO    $-5
SALIR_WRITE_I2C
        BANKSEL CON_BRUJ
        BCF     CON_BRUJ,   1       ;   APAGA   LA
INTERRUPCION
        RETURN
;*******************************************
;       BRUJULA: LEE EN EL MSSP EN I2C EL
CONTENIDO DE ENVIO_I2C
;*******************************************
RD_I2C_1
        BANKSEL CON_BRUJ
        BCF     CON_BRUJ,   1       ;   APAGA   LA
INTERRUPCION
        BANKSEL SSPCON2
        BSF   SSPCON2, 3  ; HABILITA RECEPCION
DEL MSSP
        BANKSEL CON_BRUJ
        BTFSC   CON_BRUJ, 6 ; HAY TIME OUT?
        GOTO    SALIR_RD1_I2C
        BANKSEL CON_BRUJ
        BTFSS   CON_BRUJ, 1     ; VERIFICA QUE
YA HAYA SIDO LA INTERRUPCION
        GOTO    $-5
        BANKSEL CON_BRUJ
        BCF     CON_BRUJ,   1       ;   APAGA   LA
INTERRUPCION
        BANKSEL SSPBUF
        MOVF    SSPBUF, 0
        BANKSEL DATO_BRUJ1
        MOVWF  DATO_BRUJ1  ; PASA LO RECIBIDO
A
```

```
        BANKSEL SSPCON2     ; SI MANDAR BIT DE
ACKN
        BCF     SSPCON2, 5
        BANKSEL  SSPCON2    ; MANDA BIT DE ACKN
   BSF     SSPCON2, 4
        BANKSEL CON_BRUJ
        BTFSC     CON_BRUJ, 6 ; HAY TIME OUT?
        GOTO      SALIR_RD1_I2C
        BANKSEL CON_BRUJ
        BTFSS     CON_BRUJ, 1      ; VERIFICA QUE
YA HAYA SIDO LA INTERRUPCION
        GOTO      $-5
SALIR_RD1_I2C
        BANKSEL CON_BRUJ
        BCF       CON_BRUJ, 1
        RETURN
;*****************************************
;       BRUJULA: LEE EN EL MSSP EN I2C EL
CONTENIDO DE ENVIO_I2C
;*****************************************
RD_I2C_2
        BANKSEL CON_BRUJ
        BCF       CON_BRUJ, 1
        BANKSEL SSPCON2
        BSF             SSPCON2, 3      ; HABILITA
RECEPCION DEL MSSP
        BANKSEL CON_BRUJ
        BTFSC     CON_BRUJ, 6 ; HAY TIME OUT?
        GOTO      SALIR_RD2_I2C
        BANKSEL CON_BRUJ
        BTFSS     CON_BRUJ, 1      ; VERIFICA QUE
YA HAYA SIDO LA INTERRUPCION
        GOTO      $-5
        BANKSEL CON_BRUJ
        BCF       CON_BRUJ, 1
        BANKSEL SSPBUF
        MOVF    SSPBUF, 0
        BANKSEL DATO_BRUJ2
        MOVWF   DATO_BRUJ2   ; PASA LO RECIBIDO
        BANKSEL SSPCON2     ; SI MANDAR BIT DE
ACKN
        BSF     SSPCON2, 5
        BANKSEL SSPCON2    ; MANDA BIT DE ACKN
   BSF     SSPCON2, 4
        BANKSEL CON_BRUJ
        BTFSC     CON_BRUJ, 6 ; HAY TIME OUT?
        GOTO      SALIR_RD2_I2C
        BANKSEL CON_BRUJ
        BTFSS     CON_BRUJ, 1      ; VERIFICA QUE
YA HAYA SIDO LA INTERRUPCION
        GOTO      $-5
SALIR_RD2_I2C
        BANKSEL CON_BRUJ
        BCF       CON_BRUJ, 1
        RETURN
<GPS.asm>
;*****************************************
;       ENVIA LOS DATOS AL PIC DE GPS
;*****************************************
RQST_DATOS_GPS
        BANKSEL BANDERA
        BCF       BANDERA, 4
        BCF       BANDERA, 1
        BCF       BANDERA, 0
        BANKSEL PORTA
        MOVLW   B'01010101'     ;     TODOS     LOS
TRANSCIEVERS EN TRI ESTADO
        MOVWF   PORTA
        BANKSEL ESTADO
        BSF       ESTADO, 0 ; INTERACTUANDO CON
GPS.
        CLRF      CONT_GPS
        CALL      CONFIG_SERIAL_9600_GPS
;    INT.H (DE)      ;
IN.H (RE) GPS(DE) GPS(RE) MOT(DE) MOT(RE) RF(DE)
RF(RE)
; DE <---- 1 ESCRIBRE
```

```
; RE <---- 0 LEE
        BANKSEL PORTA
        MOVLW   B'01100101'         ;       SOLO TRANSCIEVERS DE GPS ACTIVADOS, LOS DEMAS EN TRI ESTADO
        MOVWF   PORTA
        ;PIC_MASTER_TRANSCEIVERS_ENCENDER
        BANKSEL PORTC
        BCF     PORTC, 0 ; DE
        BSF     PORTC, 1 ; RE
        BANKSEL DIRECCION_GPS
        MOVF    DIRECCION_GPS,W
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_GPS    ; ENVIA DIRECCION DE GPS
        ; CONFIGURACION DEL TMR0
;       BANKSEL OPTION_REG
;       MOVLW   B'11010111'
;       MOVWF   OPTION_REG
;       BANKSEL TMR0
;       CLRF    TMR0
;       BANKSEL INTCON    ; INTERRUPCION DE TMR0
;       BSF     INTCON, T0IE ; BIT 5
        PAGESEL 0
        RETURN
;*****************************************
;       PASA LOS DATOS DEL GPS DE LAS VAR AUX
;       A LAS QUE CORRESPONDEN
;*****************************************
PASAR_DATOS_GPS_VALIDOS
        BANKSEL CONT_AUX
        CLRF    CONT_AUX
AGAIN1
        BANKSEL DIR_DAT_GPS_AUX
        MOVF    DIR_DAT_GPS_AUX, W
        MOVWF   FSR
        INCF    FSR
        BANKSEL CONT_AUX
        MOVF    CONT_AUX, W
        ADDWF   FSR,F
        BANKSEL INDF
        MOVF    INDF, W     ; PASA EL DATO RECIBIDO A LA VARIABLE QUE CORRESPONDE
        BANKSEL AUX
        MOVWF   AUX
        BANKSEL DIR_DAT_GPS
        MOVF    DIR_DAT_GPS, W
        MOVWF   FSR
        BANKSEL CONT_AUX
        MOVF    CONT_AUX, W
        ADDWF   FSR,F
        MOVF    AUX, W
        BANKSEL INDF
        MOVWF   INDF   ; PASA EL DATO RECIBIDO A LA VARIABLE QUE CORRESPONDE
        BANKSEL CONT_AUX
        INCF    CONT_AUX
        MOVLW   D'4'   ; VERIFICA SI YA TERMINO
        SUBWF   CONT_AUX,W
        BTFSS   STATUS, Z         ; SI Z ESTA EN UNO EL RESULTADO FUE CERO
        GOTO    AGAIN1
        BANKSEL BANDERA
        BCF     BANDERA, 1 ; YA PASO LOS DATOS
        BANKSEL CONT_AUX
        CLRF    CONT_AUX
        BANKSEL ESTADO
;       BCF     ESTADO, 0 ; YA TERMINO CON EL GPS
        PAGESEL 0
        RETURN
;*****************************************
;       VERIFICA SI COMUNICACION CON GPS FALLO
;       SI SI, REINICIA TODO LO DEL GPS
;*****************************************
```

```
REINICIAR_COMUNICACION_GPS
        ; REINICIAR GPS
        BANKSEL ESTADO
        BCF     ESTADO,  0      ;  YA  PUEDE
INTERACTUAR CON OTRO DISPOSITIVO
        CLRF    CONT_GPS      ;  REINICIA  EL
CONTADOR
;       CLRF    ACCION  ; APAGA EL GPS
;       BSF     ACCION, 1 ; TOCA SENSORES
        PAGESEL 0
        RETURN
;*****************************************
;       CONFIGURACION DEL PUERTO SERIAL PARA
EL GPS
;*****************************************
CONFIG_SERIAL_9600_GPS
        BANKSEL TRISC
        BCF     TRISC, 6
        BSF     TRISC, 7
        BANKSEL TXREG
        CLRF    TXREG
        BANKSEL TXSTA
        BCF     TXSTA,SYNC      ; BIT SYNC
        BSF     TXSTA,BRGH      ; BIT BRGH
        BANKSEL RCSTA
        BSF     RCSTA,SPEN      ; BIT SPEN
        BANKSEL BAUDCTL
        BCF     BAUDCTL,BRG16   ; BIT BRG16
        BANKSEL SPBRGH
        CLRF    SPBRGH
        MOVLW   D'25'
        MOVWF   SPBRG
        BANKSEL TXSTA
        BSF     TXSTA,TXEN      ; BIT TXEN
        BANKSEL INTCON
        BSF     INTCON,  PEIE      ;  HABILITA
INTERRUPCION DEL PERIFERICO DEL SERIAL
        BANKSEL PIE1
        BCF     PIE1,TXIE      ;      DESHABILITA
INTERRUPCION DE TRANSMISION
        BSF     PIE1,RCIE      ;      HABILITA
INTERRUPCION DE RECEPCION
        BANKSEL RCSTA
        BSF     RCSTA, CREN      ; HABILITA LA
RECEPCION
        BANKSEL TXREG
        CLRF    TXREG
        CALL    DELAY_5MS_GPS
        RETURN
;*****************************************
;       ENVIA UN PAQUETE POR EL PUERTO SERIAL
;*****************************************
ENV_CAR_SERIAL_GPS
        BANKSEL ENV_SER
        MOVF    ENV_SER,W
        BANKSEL TXREG
        MOVWF   TXREG
        BANKSEL PIR1
        BTFSS   PIR1,TXIF
        GOTO    $-1
        RETURN
;*****************************************
; GENERAL: DELAY DE 5 mS
;*****************************************
DELAY_5MS_GPS
        CALL DELAY_1MS_GPS
        CALL DELAY_1MS_GPS
        CALL DELAY_1MS_GPS
        CALL DELAY_1MS_GPS
        CALL DELAY_1MS_GPS
        RETURN
;*****************************************
; GENERAL: DELAY DE 1 mS
;*****************************************
DELAY_1MS_GPS
        CALL DELAY_05MS_GPS
        CALL DELAY_05MS_GPS
```

```
        RETURN

;*****************************************
; GENERAL: DELAY DE 0.5 mS
;*****************************************
DELAY_05MS_GPS
        CLRF    CONTADOR_DELAY
DELAY_D_GPS
        INCF    CONTADOR_DELAY
        MOVLW   D'82'
        SUBWF   CONTADOR_DELAY,W
        BTFSS   STATUS, 2
        GOTO    DELAY_D_GPS
        CLRF    CONTADOR_DELAY
        NOP
        NOP
        RETURN
<Motores.asm>
;*****************************************
; ENVIA UNA INSTRUCCION A LOS MOTORES
;*****************************************
ENVIAR_INST_MOTORES
        BANKSEL ESTADO
        BSF     ESTADO, 0 ; NO SE PUEDE
INTERACTUAR CON OTRO MODULO
        BANKSEL PORTA
        MOVLW   B'01010101'   ;   TODOS   LOS
TRANSCIEVERS EN TRI ESTADO
        MOVWF   PORTA
        CALL    CONFIG_SERIAL_19200
;   INT.H (DE)      IN.H (RE)
; GPS(DE) GPS(RE) MOT(DE) MOT(RE) RF(DE) RF(RE)
; DE <---- 1 ESCRIBRE
; RE <---- 0 LEE
        BANKSEL PORTA
        MOVLW   B'01011001'         ;         SOLO
TRANSCIEVERS DE MOTORES ACTIVADOS, LOS DEMAS
EN TRI ESTADO
        MOVWF   PORTA
        ; PIC_MASTER_TRANSCEIVERS_ENCENDER
        BANKSEL PORTC
        BCF     PORTC, 0 ; DE
        BSF     PORTC, 1 ; RE
        CALL    DELAY_1MS_MOT
;INST_MOTORES
        MOVLW   D'1' ; ADELANTE
        SUBWF   INST_MOTORES, W  ; REVISA SI LA
DIRECCION ES 1.
        BTFSS   STATUS, Z        ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    SIG1
        CALL    MOVER_ADELANTE
        GOTO    SALIR_ENV_INST_MOTORES
SIG1
        MOVLW   D'2' ; ATRAS
        SUBWF   INST_MOTORES, W  ; REVISA SI LA
DIRECCION ES 1.
        BTFSS   STATUS, Z        ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    SIG2
        CALL    MOVER_ATRAS
        GOTO    SALIR_ENV_INST_MOTORES
SIG2
        MOVLW   D'3' ; IZQUIERDA
        SUBWF   INST_MOTORES, W  ; REVISA SI LA
DIRECCION ES 1.
        BTFSS   STATUS, Z        ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    SIG3
        CALL    MOVER_IZQUIERDA
        GOTO    SALIR_ENV_INST_MOTORES
SIG3
        MOVLW   D'4' ; DERECHA
        SUBWF   INST_MOTORES, W  ; REVISA SI LA
DIRECCION ES 1.
        BTFSS   STATUS, Z        ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    SIG4
```

```
        CALL    MOVER_DERECHA
        GOTO    SALIR_ENV_INST_MOTORES
SIG4
        MOVLW   D'5' ; PARAR
        SUBWF   INST_MOTORES, W  ; REVISA SI LA
DIRECCION ES 1.
        BTFSS   STATUS, Z        ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    SALIR_ENV_INST_MOTORES
        CALL    CLEAR_POSICION
SALIR_ENV_INST_MOTORES
        CLRF    ACCION
;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3 RF, 4
MOTORES
        BSF     ACCION, 2 ; HACER BRUJULA
        ;BSF    ACCION, 1 ; HACER SENSORES
        BCF     ESTADO,  0  ;  YA  SE  PUEDE
INTERACTUAR CON OTRO MODULO
        CALL    DELAY_20MS_MOT
        CALL    DELAY_10MS_MOT
        CALL    DELAY_5MS_MOT
        BANKSEL PORTA
        MOVLW   B'01010101'
        MOVWF   PORTA
        ;PIC_MASTER_TRANSCEIVERS_APAGAR
        BANKSEL PORTC
        BSF     PORTC, 0 ; DE
        BCF     PORTC, 1 ; RE
        PAGESEL 0
        RETURN
;*****************************************
; HACE UN CLEAR DE LAS POSICIONES DEL MOTOR
;*****************************************
CLEAR_POSICION
        MOVLW   H'28'    ;   CLEAR,   INICIALIZAR
POSICION
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        RETURN
;*****************************************
; MUEVE LOS MOTORES HACIA ADELANTE
;*****************************************
MOVER_ADELANTE
        CALL    CLEAR_POSICION
; SON 16 BITS PARA MOVER LOS MOTORES, LAS
POSICIONES SON EN COMPLEMENTO A DOS.
        MOVLW   H'20'   ; INSTRUCCION MOVER
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT

; INS MOVER, (PARTE ALTA DE CANT DE POSICION,
PARTE BAJA DE CANT DE POSICIONES)
; UNA VUELTA TIENE 37 POSICIONES
; UNA VUELTA SON 48.64848 CM
        MOVLW   H'0'    ; PARTE ALTA
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        MOVLW   D'255' ; PARTE BAJA
        ;MOVLW  D'37'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        RETURN
;*****************************************
; MUEVE LOS MOTORES HACIA ATRAS
;*****************************************
MOVER_ATRAS
        CALL    CLEAR_POSICION
        MOVLW   H'20' ; INSTRUCCION MOVER
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        MOVLW   H'FF'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
```

```
        CALL    ENV_CAR_SERIAL_MOT
        ;MOVLW  H'01'
        MOVLW   H'E0'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        RETURN
;*****************************************
; MUEVE LOS MOTORES HACIA IZQUIERDA
;*****************************************
MOVER_IZQUIERDA
        CALL    CLEAR_POSICION
        MOVLW   H'22'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        MOVLW   H'0'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
;       MOVLW   D'255'
        MOVLW   D'37'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        RETURN
;*****************************************
; MUEVE LOS MOTORES HACIA DERECHA
;*****************************************
MOVER_DERECHA
        CALL    CLEAR_POSICION
        MOVLW   H'21'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        MOVLW   H'0'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        MOVLW   D'255'
        MOVLW   D'37'
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_MOT
        RETURN
;*****************************************
;       CONFIGURACION DEL PUERTO SERIAL A
19200
;*****************************************
CONFIG_SERIAL_19200
        BANKSEL TRISC
        BCF     TRISC, 6
        BSF     TRISC, 7
        BANKSEL TXREG
        CLRF    TXREG
        BANKSEL TXSTA
        BCF     TXSTA,SYNC      ; BIT SYNC
        BSF     TXSTA,BRGH      ; BIT BRGH
        BANKSEL RCSTA
        BSF     RCSTA,SPEN      ; BIT SPEN
        BANKSEL BAUDCTL
        BCF     BAUDCTL,BRG16   ; BIT BRG16
        BANKSEL SPBRGH
        CLRF    SPBRGH
        MOVLW   D'12'
        MOVWF   SPBRG
        BANKSEL TXSTA
        BSF     TXSTA,TXEN
        CALL    DELAY_5MS_MOT
        RETURN
;*****************************************
;       ENVIA UN PAQUETE POR EL PUERTO SERIAL
;*****************************************
ENV_CAR_SERIAL_MOT
        BANKSEL ENV_SER
        MOVF    ENV_SER,W
        BANKSEL TXREG
        MOVWF   TXREG
```

```
        BANKSEL PIR1
        BTFSS   PIR1,TXIF
        GOTO    $-1
        RETURN
;*****************************************
; GENERAL: DELAY DE 50 mS
;*****************************************
DELAY_50MS_MOT
        CALL DELAY_20MS_MOT
        CALL DELAY_20MS_MOT
        CALL DELAY_10MS_MOT
        RETURN
 ;*****************************************
; GENERAL: DELAY DE 20 mS
;*****************************************
DELAY_20MS_MOT
        CALL DELAY_10MS_MOT
        CALL DELAY_10MS_MOT
        RETURN
;*****************************************
; GENERAL: DELAY DE 10 mS
;*****************************************
DELAY_10MS_MOT
        CALL DELAY_5MS_MOT
        CALL DELAY_5MS_MOT
        RETURN
;*****************************************
; GENERAL: DELAY DE 5 mS
;*****************************************
DELAY_5MS_MOT
        CALL DELAY_1MS_MOT
        CALL DELAY_1MS_MOT
        CALL DELAY_1MS_MOT
        CALL DELAY_1MS_MOT
        CALL DELAY_1MS_MOT
        RETURN
;*****************************************
; GENERAL: DELAY DE 1 mS
;*****************************************
DELAY_1MS_MOT
        CALL DELAY_05MS_MOT
        CALL DELAY_05MS_MOT
        RETURN
;*****************************************
; GENERAL: DELAY DE 0.5 mS
;*****************************************
DELAY_05MS_MOT
        CLRF    CONTADOR_DELAY
DELAY_D_MOT
        INCF    CONTADOR_DELAY
        MOVLW   D'82'
        SUBWF   CONTADOR_DELAY,W
        BTFSS   STATUS, 2
        GOTO    DELAY_D_MOT
        CLRF    CONTADOR_DELAY
        NOP
        NOP
        RETURN
<RF.asm>
;*****************************************
;       DIRECCIONA EL PIC DE RF
;*****************************************
DIRECCIONAR_RF
        BANKSEL CON_RF
        MOVF    CON_RF, F
        BTFSC   STATUS,  Z  ;  SE  PUEDE
DIRECCIONAR RF
        GOTO    NO_SET_RF1
        PAGESEL 0
        RETURN
NO_SET_RF1
        BANKSEL PORTA
        MOVLW   B'01010101'  ;  TODOS  LOS
TRANSCIEVERS EN TRI ESTADO
        MOVWF   PORTA
        CALL    CONFIG_SERIAL_9600_RF
        BANKSEL ESTADO
```

```
        BSF     ESTADO, 0 ; NO SE PUEDE INTERACTUAR CON OTRO MODULO
        ;----------------------------
        ; HABILITAR PIN DE TRANSCIEVER.
        ;----------------------------
;  INT.H (DE)   IN.H (RE)  GPS(DE)  GPS(RE)  MOT(DE) MOT(RE) RF(DE) RF(RE)
; DE <---- 1 ESCRIBRE
; RE <---- 0 LEE
        BANKSEL PORTA
        MOVLW   B'01010110'     ; SOLO TRANSCIEVERS DE RF ACTIVADOS, LOS DEMAS EN TRI ESTADO
        MOVWF   PORTA
;PIC_MASTER_TRANSCEIVERS_ENCENDER
        BANKSEL PORTC
        BCF     PORTC, 0 ; DE
        BSF     PORTC, 1 ; RE
        BANKSEL DIRECCION_RF
        MOVF    DIRECCION_RF, W
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_RF   ; ENVIA DIRECCION DE RF
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF ; TIME OUT
        BANKSEL CON_RF
        CLRF    CON_RF
        BSF     CON_RF, 0
        CLRF    INSTRUCCION_RF
        ; CONFIGURACION DEL TMR0
        BANKSEL OPTION_REG
        MOVLW   B'11010111'
        MOVWF   OPTION_REG
        BANKSEL TMR0
        CLRF    TMR0
        BANKSEL INTCON  ; INTERRUPCION DE TMR0
        BSF     INTCON, GIE ; BIT 7
        BSF     INTCON, T0IE ; BIT 5
        PAGESEL 0
        RETURN
;*****************************************
;       SE COMUNICA CON EL PIC RF
;*****************************************
ESTABLECER_COMUNICACION_RF
        BANKSEL CON_RF
        BTFSC   CON_RF, 2
        GOTO    YA_ENVIO_INSTRUCCION_RF
        BANKSEL INSTRUCCION_RF
        INCF    INSTRUCCION_RF, F
        CALL    ENVIAR_INSTRUCCION_RF ; ENVIA LA INSTRUCCION QUE ESTA EN LA VAR INSTRUCCION_RF
        BANKSEL INSTRUCCION_RF
        MOVLW   D'10';
        SUBWF   INSTRUCCION_RF, W   ; VERIFICA SI YA ENVIO TODAS LAS INSTRUCCIONES
        BTFSC   STATUS, Z       ; SI Z ESTA EN UNO EL RESULTADO FUE CERO
        BSF     CON_RF, 6 ; YA MANDO TODAS LAS INSTRUCCIONES
        PAGESEL 0
        RETURN
YA_ENVIO_INSTRUCCION_RF
        BANKSEL CON_RF
        BTFSC   CON_RF, 3 ; YA HUBO AKN DE INSTRUCCION
        GOTO    YA_AKN_INSTRUCCION
        PAGESEL 0
        RETURN ; NO HAY AKN INSTRUCCION
YA_AKN_INSTRUCCION
        BANKSEL CON_RF
        BTFSS   CON_RF, 4
        GOTO    NO_SET_RF2
        PAGESEL 0
        RETURN
NO_SET_RF2
```

```
        BANKSEL CON_RF
        BTFSC   CON_RF, 5
        GOTO    NO_SET_RF3
        PAGESEL 0
        RETURN
NO_SET_RF3
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        MOVF    DIR_DAT_GUARDAR_RF,W
        MOVWF   FSR
        MOVF    AUX_RF, W
        ADDWF   FSR, F
        MOVF    INDF, W
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        PAGESEL ENV_CAR_SERIAL_RF
        CALL    ENV_CAR_SERIAL_RF    ; ENVIA
DATOS AL PIC RF
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        BANKSEL AUX_RF
        INCF    AUX_RF, F
        MOVF    CONT_DAT_GUARDAR_RF, W
        SUBWF   AUX_RF, W  ; VERIFICA SI YA SE
DIRECCIONARON TODAS LAS VARIABLES
        BTFSS   STATUS, Z          ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    FALTA_ENVIAR_DATOS_RF
        BTFSS   CON_RF, 6
        GOTO    FALTAN_INSTRUCCIONES_RF2
        BANKSEL INSTRUCCION_RF
        CLRF    INSTRUCCION_RF
        ;ACCION 0 GPS, 1 SENSORES, 2 BRUJULA, 3
RF, 4 MOTORES
        CLRF    ACCION
        BSF     ACCION, 4 ; HACER MOTORES
        ;BSF    ACCION, 3 ; HACER RF
        BCF     ESTADO,  0  ;  YA  SE  PUEDE
INTERACTUAR CON OTRO MODULO
        CLRF    CON_RF
        BANKSEL INTCON    ;  INTERRUPCION  DE
TMR0
        BCF     INTCON, T0IE ; BIT 5C
        CALL    DELAY_1MS_RF
        ;PIC_MASTER_TRANSCEIVERS_APAGAR
        BANKSEL PORTC
        BSF     PORTC, 0 ; DE
        BCF     PORTC, 1 ; RE
        BANKSEL PORTA
        MOVLW   B'01010101'
        MOVWF   PORTA
        PAGESEL 0
        RETURN
        FALTAN_INSTRUCCIONES_RF2
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        MOVLW   B'00000011' ; LISTO PARA ENVIAR
LA SIGUIENTE INSTRUCCION
        MOVWF   CON_RF
        CLRF    AUX_RF
FALTA_ENVIAR_DATOS_RF
        CALL    DELAY_5MS_RF
        PAGESEL 0
        RETURN
;*****************************************
;       DIRECCIONA EL PIC DE RF
;*****************************************
ENVIAR_INSTRUCCION_RF
        MOVF    INSTRUCCION_RF, W
        BANKSEL ENV_SER
        MOVWF   ENV_SER
        CALL    ENV_CAR_SERIAL_RF    ; ENVIA
INSTRUCCION AL RF
        BANKSEL CONT_TMR0_RF
        CLRF    CONT_TMR0_RF
        BANKSEL CON_RF
        BSF     CON_RF, 2
        CLRF    DIR_DAT_GUARDAR_RF
```

```
        CLRF    CONT_DAT_GUARDAR_RF
        CLRF    CONT_AUX_RF
        MOVLW   D'1' ; MOTORES
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z          ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT1
        BSF     CON_RF, 4
        MOVLW   D'1'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0x4F
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT1
        MOVLW   D'2' ; ORIGEN
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z          ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT2
        BSF     CON_RF, 4
        MOVLW   D'10'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0xA0
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT2
        MOVLW   D'3' ; DESTINO
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z          ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT3
        BSF             CON_RF, 4
        MOVLW   D'10'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0xAA
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT3
        MOVLW   D'4' ;GAVETA
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z          ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT4
        BSF     CON_RF, 4
        MOVLW   D'10'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0xB4
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT4
        MOVLW   D'5' ; SENSORES
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z          ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT5
        BSF     CON_RF, 5
        MOVLW   D'8'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0x34
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT5
        MOVLW   D'6' ; BRUJULA
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z          ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT6
        BSF     CON_RF, 5
        MOVLW   D'2'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0x2B
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT6
        MOVLW   D'7' ; GPS
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
```

```
        BTFSS   STATUS, Z       ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT7
        BSF     CON_RF, 5
        MOVLW   D'4'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0x4B
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT7
        MOVLW   D'8' ; CONTEO DE PASOS, POR EL
MOMENTO VOY A PONER LOS DATOS DE BATERIA
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z       ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT8
        BSF     CON_RF, 5
        MOVLW   D'2'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0x52
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT8
        MOVLW   D'9' ; BATERIA
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z       ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT9
        BSF     CON_RF, 5
        MOVLW   D'1'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0x50
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT9
        MOVLW   D'10' ; ESTADO
        SUBWF   INSTRUCCION_RF, W  ; REVISA SI
LA DIRECCION ES 1.
        BTFSS   STATUS, Z       ; SI Z ESTA EN
UNO EL RESULTADO FUE CERO
        GOTO    NEXT10
        BSF             CON_RF, 5
        MOVLW   D'1'
        MOVWF   CONT_DAT_GUARDAR_RF
        MOVLW   0x51
        MOVWF   DIR_DAT_GUARDAR_RF
NEXT10
        RETURN
;*****************************************
;       CONFIGURACION DEL PUERTO SERIAL
;*****************************************
CONFIG_SERIAL_9600_RF
        BANKSEL TRISC
        BCF     TRISC, 6
        BSF     TRISC, 7
        BANKSEL TXREG
        CLRF    TXREG
        BANKSEL TXSTA
        BCF     TXSTA,SYNC      ; BIT SYNC
        BSF     TXSTA,BRGH      ; BIT BRGH
        BANKSEL RCSTA
        BSF     RCSTA,SPEN      ; BIT SPEN
        BANKSEL BAUDCTL
        BCF     BAUDCTL,BRG16   ; BIT BRG16
        BANKSEL SPBRGH
        CLRF    SPBRGH
        MOVLW   D'25'
        MOVWF   SPBRG
        BANKSEL TXSTA
        BSF     TXSTA,TXEN      ; BIT TXEN
        BANKSEL INTCON
        BSF     INTCON, PEIE    ; HABILITA
INTERRUPCION DEL PERIFERICO DEL SERIAL
        BANKSEL PIE1
        BCF     PIE1,TXIE       ;       DESHABILITA
INTERRUPCION DE TRANSMISION
        BSF     PIE1,RCIE       ;       HABILITA
INTERRUPCION DE RECEPCION
        BANKSEL RCSTA
```

```
        BSF     RCSTA, CREN     ; HABILITA LA
RECEPCION
        BANKSEL TXREG
        CLRF    TXREG
        CALL    DELAY_5MS_RF
        RETURN
;*****************************************
;       ENVIA UN PAQUETE POR EL PUERTO SERIAL
;*****************************************
ENV_CAR_SERIAL_RF
        BANKSEL ENV_SER
        MOVF    ENV_SER,W
        BANKSEL TXREG
        MOVWF   TXREG
        BANKSEL PIR1
        BTFSS   PIR1,TXIF
        GOTO    $-1
        CALL    DELAY_1MS_RF
        CALL    DELAY_05MS_RF
        PAGESEL ESTABLECER_COMUNICACION_RF
        RETURN
;*****************************************
; GENERAL: DELAY DE 10 mS
;*****************************************
DELAY_10MS_RF
        CALL DELAY_5MS_RF
        CALL DELAY_5MS_RF
        RETURN
;*****************************************
; GENERAL: DELAY DE 5 mS
;*****************************************
DELAY_5MS_RF
        CALL DELAY_1MS_RF
        CALL DELAY_1MS_RF
        CALL DELAY_1MS_RF
        CALL DELAY_1MS_RF
        CALL DELAY_1MS_RF
        RETURN
;*****************************************
; GENERAL: DELAY DE 1 mS
;*****************************************
DELAY_1MS_RF
        CALL DELAY_05MS_RF
        CALL DELAY_05MS_RF
        RETURN
;*****************************************
; GENERAL: DELAY DE 0.5 mS
;*****************************************
DELAY_05MS_RF
        CLRF    CONTADOR_DELAY
DELAY_D_RF
        INCF    CONTADOR_DELAY
        MOVLW   D'82'
        SUBWF   CONTADOR_DELAY,W
        BTFSS   STATUS, 2
        GOTO    DELAY_D_RF
        CLRF    CONTADOR_DELAY
        NOP
        NOP
        RETURN
<Sensores.asm>
;*****************************************
;       SENSORES: GENERA PULSO PARA PEDIR
DATOS A LOS SENSORES
;*****************************************
LEER_SENSORES
        BANKSEL ESTADO
        BSF     ESTADO, 0 ; INTERACTUANDO CON
SENSORES.
        CLRF    CON_SEN
        BANKSEL REG_SEN2
        CLRF    REG_SEN2
        ; DESHABILITA INTERRUPCION DE PORTB
        BANKSEL INTCON
        CLRF    INTCON
        ;BSF    INTCON, 7
        BCF     INTCON, 3
        BANKSEL IOCB
```

```
        CLRF    IOCB
        ; PINES DEL PORTB COMO SALIDA
        BANKSEL TRISB
        CLRF    TRISB
        ; GENERA PULSO PARA PEDIR DATOS AL S1,
S2, S3,S4,S5
        BANKSEL PORTB
        MOVLW   B'01111111'
        ;MOVLW  B'01000001'
        ;MOVLW  B'01111111'
        MOVWF   PORTB
        NOP
        NOP
        NOP
        ; APAGA EL PULSO
        CLRF    PORTB
        NOP
        ; PORTB COMO ENTRADA
        BANKSEL TRISB
        ;MOVLW  B'11111111'
        ;MOVLW  B'00000001'
        MOVLW   B'01111111'
        ;MOVLW  B'01000001'
        MOVWF   TRISB
        ; HABILITA INTERRUPCION DE PORTB
        BANKSEL INTCON
        BSF             INTCON, GIE
        BSF             INTCON, 3
        ; HABILITA INTERRUPCION DE PINES DE
PORTB
        BANKSEL IOCB
        ;MOVLW  B'11111111'
        ;MOVLW  B'01000001'
        ;MOVLW  B'00000001'
        MOVLW   B'01111111'
        MOVWF   IOCB
        PAGESEL 0
        RETURN
;*****************************************
; GENERAL: DELAY DE 10 mS
;*****************************************
DELAY_10MS_SEN
        CALL DELAY_5MS_SEN
        CALL DELAY_5MS_SEN
        RETURN
;*****************************************
; GENERAL: DELAY DE 5 mS
;*****************************************
DELAY_5MS_SEN
        CALL DELAY_1MS_SEN
```

2. Código fuente del DSPIC GPS

```
// Programa que recibe hasta que encuentra una @ y
despliega los primeros 5
char  recibido[78]=" ";
char  dat_valido[24]= " ";
char  lat_lon_reducido[8] = " ";
char bandera,dato_rec,dato_rec2;
char bandera2;
int cont,cont_pasar,cont_env;
int i,j,k;

void inicio()
{
    // CONFIGURACION DE PUERTOS
```

```
    TRISE = 0;//PUERTOS DE SALIDA DE BANCO DE P|WM
    LATE = 0;  // SE LIMPIA A UN ESTADO INICIAL
    TRISF = 0;  // PUERTOS PARA RELOJ
    LATF = 0;               //     SE LIMPIA EL ESTADO INICIAL
    LATB = 0;          //     SE LIMPIA ESTADO INICIAL
    ADPCFG = 0xFFFF;   // configurar IO como digitales
    // Inicializacion de Variables
    UART2_init(4800);  //    SE INICIALIZA EL MODULO
UART1_init(9600);    //     SE INICIALIZA EL MODULO CON
    for (i=0;i<24+1;i++){ // son 23 por la ,
        dat_valido[i] = 55;
    }
    for (i=0;i<8+1;i++){ // son 23 por la ,
        lat_lon_reducido[i] = 0;
    }
    cont = -1;
    j = 0;
    k = 0;
    bandera=0;
    bandera2=0;
    dato_rec=0;
    dato_rec2=0;
    cont_pasar = 0;
    cont_env = 0;
}

void main()
{
      inicio();
      while(1)
      {
            if (UART2_Data_Ready()) {          // if data is received
                  dato_rec = UART2_Read();
                  cont = cont+1;
                  if(dato_rec!=64){
                        recibido[cont] = dato_rec;
                  }
                  if(dato_rec==64){
                        if(cont == 77){
                              bandera.F0=1;
                        }
                        cont=-1;
                  }


            }
            // aca va uart1

            if (UART1_Data_Ready()) {
                  dato_rec2 = 0;
                  UART1_Write(dato_rec2);
                  if (dato_rec2==240){

             dato_rec2 = UART1_Read();

             bandera2.F0 = 1;
                     k = 0;
                  }
            }

            if (bandera.F0){

if((recibido[4]==71)&&(recibido[43]==49)){
                        dat_valido[j] = recibido[j+18];
                        j = j+1;
                        if(j==24){
                            bandera.F0 = 0;
                            bandera.F1 = 1;
                            j = 0;
                            cont_pasar = 0;
                        }
                     }
            }
```

```
            if (bandera.F1){
lat_lon_reducido[0] = (char)(dat_valido[0]-48)*10 +
dat_valido[1]-48;
lat_lon_reducido[1] = (char)(dat_valido[2]-48)*10 +
dat_valido[3]-48;
lat_lon_reducido[2] = (char)(dat_valido[5]-48)*10 +
dat_valido[6]-48;
lat_lon_reducido[3] = (char)(dat_valido[7]-48)*10 +
dat_valido[8]-48;
lat_lon_reducido[4] = (char)(dat_valido[13]-48)*10 +
(dat_valido[14]-48);
lat_lon_reducido[5] = (char)(dat_valido[15]-48)*10 +
dat_valido[16]-48;
lat_lon_reducido[6] = (char)(dat_valido[18]-48)*10 +
dat_valido[19]-48;
lat_lon_reducido[7] = (char)(dat_valido[20]-48)*10 +
dat_valido[21]-48;
                bandera.F1 = 0 ;
            }

            if (bandera2.F0){
                UART1_Write(lat_lon_reducido[k]);
                k = k+1;
                if(k==8){
                    bandera2.F0 = 0;
                    k = 0;
                }

            }

        }
    }
```

3. Código fuente del DSPIC RF

```
// Probar simular el Xbee, mostrar en UART1 lo que
recibio o envio.

// Instrucciones
char inst_motores, solicitud, transferencia;

// Datos Instrucciones de Solicitud RF
char sensores[8]= " ";
char brujula[2]= " ";
char gps[4] = " ";
char posicion_pic[2] = " ";
char bateria;
char estado;
char posicion_robot_lat_lon[4] = " ";
char RFmover;
char RFsolicitud;
char RFtransferencia;
char RFinstruccionTipo;
char RFinstruccion;
char RFparametro;
char RFuartIn;
char RFuartOut;

char RFTXarray[15] = " ";
char RFTXarrayLength;
char RFTXarrayCont;
char RFTXtransmitting;
```

```
// Datos Instrucciones de Transferencias RF
char origen[10] = " ";
char destino[10] = " ";
char gaveta[10] = " ";
char posicion[4] = " ";

// Variables
char instruccion1[2] = " ";
char instruccion2[2] = " ";
char auxiliar1[8] = " ";
char auxiliar2[10] = " ";
char rec1[2] = " ";
char rec2[2] = " ";
char rec_transf;
char cont1, cont2, cont_aux, cont_transf,cont_rec_datos;
char cont_aux2,cont_aux3, cont_aux4;
int maximo;
int max_aux2,max_aux1,max1;
char bandera1,bandera2,bandera_aux;
int i,j,k;

char debug,cosa;

void inicio(){
// CONFIGURACION DE PUERTOS
TRISE = 0;        //    PUERTOS DE SALIDA DE BANCO DE P|WM
LATE = 0;             //      SE LIMPIA A UN ESTADO INICIAL
TRISF = 0;        //    PUERTOS PARA RELOJ Y RESET DE CONTADOR OCTAL
LATF = 0;         //    SE LIMPIA EL ESTADO INICIAL
TRISB = 0;          //      PRIMEROS TRES PINES PARA ENTRADAS DE SENSORES HALL
LATB = 0;         //    SE LIMPIA ESTADO INICIAL
ADPCFG = 0xFFFF;  // configurar IO como digitales

// Inicializacion de Variables
UART2_init(9600);  //    SE INICIALIZA EL MODULO CON UN BAUDRATE DE 9600
UART1_init(9600);  //    SE INICIALIZA EL MODULO CON UN BAUDRATE DE 9600

for (i=0;i<8;i++){
  sensores[i] = 48;
  auxiliar1[i] = 0;
}
for (i=0;i<4;i++){
  gps[i] = 50;
  posicion[i] = 54;
  posicion_robot_lat_lon[i] = 54;
}
for (i=0;i<10;i++){
  origen[i] = 51;
  destino[i] = 52;
  gaveta[i] = 53;
  auxiliar2[i] = 99;
}
for (i=0;i<2;i++){
  brujula[i] = 49;
  instruccion1[i] = 0;
  instruccion2[i] = 0;
  rec1[i] = 0;
  rec2[i] = 0;
  posicion_pic[i] = 54;
}

bateria = 55;
estado = 56;

cont1 = 0;
cont2 = 0;
cont_aux = 0;
cont_aux2 = 0;
cont_aux3 = 0;
cont_aux4 = 0;
cont_transf = 0;
```

```
cont_rec_datos = 0;

max_aux1 = 0;
max_aux2 = 0;

bandera1 = 0;
bandera2 = 0;
bandera_aux = 0;
maximo = 0;
max1 = 0;

rec_transf = 0;

i = 0;
j = 0;
k = 0;

debug = 0;

inst_motores = 0;

RFmover = 0;
RFsolicitud = 0;
RFtransferencia = 0;
RFinstruccionTipo = 0;
RFinstruccion = 0;
RFparametro = 0;
RFuartIn = 0;
RFuartOut = 0;
}

void Leer_Xbee (){
if (UART2_Data_Ready()) {          // if data is received
          RFuartIn = UART2_Read();
          if (RFinstruccionTipo==0){

          if(RFuartIn>=253&&RFuartIn<=255){
                    RFinstruccionTipo = RFuartIn;
                    }
                    else{
          if (RFinstruccionTipo == 255){
                    if          (RFuartIn>=1          &&
RFuartIn<=5){
                              RFmover = RFuartIn;
                              inst_motores = RFuartIn;
                    }
                    RFinstruccionTipo = 0;
                    RFinstruccion = 0;
          }
          if (RFinstruccionTipo == 254){
                    if (RFuartIn==6){
                              RFsolicitud.F0 = 1;
                    }
                    if (RFuartIn==7){
                              RFsolicitud.F1 = 1;
                    }
                    if (RFuartIn==8){
                              RFsolicitud.F2 = 1;
                    }
                    if (RFuartIn==9){
                              RFsolicitud.F3 = 1;
                    }
                    if (RFuartIn==10){
                              RFsolicitud.F4 = 1;
                    }
                    if (RFuartIn==11){
                              RFsolicitud.F5 = 1;
                    }
                    RFinstruccionTipo = 0;
                    RFinstruccion = 0;
          }
          if (RFinstruccionTipo == 253){
                    if (RFinstruccion == 0){
                    if(RFuartIn>=12&&
RFuartIn<=15){
                              RFinstruccion = RFuartIn;
                    }
```

```
                else{
                        RFinstruccionTipo = 0;
                        RFinstruccion = 0;
                }
                }
                else{
                if (RFinstruccion == 12){
                origen[RFparametro] = RFuartIn;
                }
                if (RFinstruccion == 13){
                destino[RFparametro] = RFuartIn;
                }
                if (RFinstruccion == 14){
                gaveta[RFparametro] = RFuartIn;
                }
                if (RFinstruccion == 15){
                posicion[RFparametro] = RFuartIn;
                }
                RFparametro += 1;

        if((RFinstruccion==15&&RFparametro>=4) ||
(RFparametro>=10)){
                        RFinstruccionTipo = 0;
                        RFinstruccion = 0;
                        RFparametro = 0;
                }
                }
        }
        }
}
}
void Acciones_Mover_RF(){
if(RFmover>=1 && RFmover<=5){
if (RFTXtransmitting==0){
        RFTXarray[0] = 255;
        RFTXarray[1] = RFmover;
        RFTXarrayLength = 2;
        RFTXtransmitting = 1;
        RFmover = 0;
}
}
}
void Acciones_Solicitud_RF(){
// Acciones a ejecutar Instruccion de Solicitud
if(RFsolicitud.F0==1){
if (RFTXtransmitting==0){
        RFTXarray[0] = 254;
        RFTXarray[1] = 6;
        RFTXarray[2] = sensores[0];
        RFTXarray[3] = sensores[1];
        RFTXarray[4] = sensores[2];
        RFTXarray[5] = sensores[3];
        RFTXarray[6] = sensores[4];
        RFTXarray[7] = sensores[5];

        RFTXarray[8] = sensores[6];
        RFTXarray[9] = sensores[7];
        RFTXarrayLength = 10;
        RFTXtransmitting = 1;
        RFsolicitud.F0 = 0;
}
}
if(RFsolicitud.F1==1){
if (RFTXtransmitting==0){
        RFTXarray[0] = 254;
        RFTXarray[1] = 7;
        RFTXarray[2] = brujula[0];
        RFTXarray[3] = brujula[1];
        RFTXarrayLength = 4;
        RFTXtransmitting = 1;
        RFsolicitud.F1 = 0;
}
}
if(RFsolicitud.F2==1){
if (RFTXtransmitting==0){
        RFTXarray[0] = 254;
        RFTXarray[1] = 8;
        RFTXarray[2] = gps[0];
```

```
            RFTXarray[3] = gps[1];
            RFTXarray[4] = gps[2];
            RFTXarray[5] = gps[3];
            RFTXarrayLength = 6;
            RFTXtransmitting = 1;
            RFsolicitud.F2 = 0;
}
}
if(RFsolicitud.F3==1){
if (RFTXtransmitting==0){
            RFTXarray[0] = 254;
            RFTXarray[1] = 9;
            RFTXarray[2] = posicion_robot_lat_lon[0];
            RFTXarray[3] = posicion_robot_lat_lon[1];
            RFTXarray[4] = posicion_robot_lat_lon[2];
            RFTXarray[5] = posicion_robot_lat_lon[3];
            RFTXarrayLength = 6;
            RFTXtransmitting = 1;
            RFsolicitud.F3 = 0;
}
}
if(RFsolicitud.F4==1){
if (RFTXtransmitting==0){
            RFTXarray[0] = 254;
            RFTXarray[1] = 10;
            RFTXarray[2] = bateria;
            RFTXarrayLength = 3;
            RFTXtransmitting = 1;
            RFsolicitud.F4 = 0;
}
}
if(RFsolicitud.F5==1){
if (RFTXtransmitting==0){
            RFTXarray[0] = 254;
            RFTXarray[1] = 11;
            RFTXarray[2] = estado;
            RFTXarrayLength = 3;
            RFTXtransmitting = 1;
            RFsolicitud.F5 = 0;
}
}
}
void Comunicacion_RF(){
Leer_Xbee (); // Recibe datos del Xbee
Acciones_Mover_RF();
Acciones_Solicitud_RF();
}

void Leer_PIC_Master (){
if (UART1_Data_Ready()) {          // if data is received
                        rec1[0] = UART1_Read();

            if    ((rec1[0]>=1)&&(rec1[0]<=11)){        //
Instruccion ?

            instruccion1[0] = rec1[0];

            UART1_Write(rec1[0]);   // AKN

            bandera1.F0 = 1; // Instruccion Nueva Valida
de PIC Master

            bandera1.F1 = 1;
            }


            if(rec1[0]==224){   // Direccionando?

            UART1_Write(rec1[0]);   // AKN

            bandera1.F5 = 1;
            }

}
}
void Comprobacion_Instruccion_PM(){
bandera1.F1 = 0;
if ((instruccion1[0]>=1)&&(instruccion1[0]<=4)){
```

```
bandera1.F2 = 1; // Instruccion de Solicitud PM
                    cont_aux3 = 0;
}
if ((instruccion1[0]>=5)&&(instruccion1[0]<=10)){
bandera1.F3 = 1; // Instruccion de Recibir Datos PM
                    cont_rec_datos = 0;
}
if (instruccion1[0]==5){
                    max1 = 8;
}
if (instruccion1[0]==6){
                    max1 = 2;
}
if (instruccion1[0]==7){
                    max1 = 4;
}
if (instruccion1[0]==8){
                    max1 = 2;
}
if (instruccion1[0]==9){
                    max1 = 1;
}
if (instruccion1[0]==10){
                    max1 = 1;
}
}
void Acciones_Solicitud_PM(){
max_aux1 = 0;
if (instruccion1[0]==1){
                    UART1_Write(inst_motores);
                    max1 = 1;
}
if (instruccion1[0]==2){
                    UART1_Write(origen[cont_aux3]);
                    max1 = 10;
}
if (instruccion1[0]==3){
                    UART1_Write(destino[cont_aux3]);
                    max1 = 10;
}
if (instruccion1[0]==4){
                    UART1_Write(gaveta[cont_aux3]);
                    max1 = 10;
}
cont_aux3 = cont_aux3 + 1;
if (cont_aux3 == max1){
          bandera1 = 0;  // Se reinician banderas
          cont_aux3 = 0;
}
}
void Leer_Recibir_Datos_PM(){
if (UART1_Data_Ready()) {          // if data is received

          auxiliar1[cont_rec_datos]=UART1_Read();
          cont_rec_datos = cont_rec_datos + 1;
                    if    (cont_rec_datos    ==    max1){
                              cont_rec_datos = 0;
                              bandera1.F3 = 0;
                              bandera1.F4 = 1;
                              cont_aux4 = 0;
                    }
}
}
void Acciones_Recibir_Datos_PM(){
if (instruccion1[0]==5){
          sensores[cont_aux4] = auxiliar1[cont_aux4];
          max1 = 8;
}
if (instruccion1[0]==6){
          brujula[cont_aux4] = auxiliar1[cont_aux4];
          max1 = 2;
}
if (instruccion1[0]==7){
          gps[cont_aux4] = auxiliar1[cont_aux4];
          max1 = 4;
}
if (instruccion1[0]==8){
```

```
            posicion_pic[cont_aux4]=auxiliar1[cont_aux4]
;
            max1 = 2;
}
if (instruccion1[0]==9){
            bateria = auxiliar1[cont_aux4];
            max1 = 1;
}
if (instruccion1[0]==10){
            estado = auxiliar1[cont_aux4];
            max1 = 1;
}
cont_aux4 = cont_aux4 + 1;
if (cont_aux4 == max1){
            cont_aux4 = 0;
            bandera1 = 0;
}
}
void Comunicacion_PIC_Master(){
if (!bandera1.F0){ // Leer PIC Master ?
            Leer_PIC_Master();
}
if (bandera1.F1){ // Instruccion Nueva de PIC Master ?
            bandera1.F1 = 0;
            Comprobacion_Instruccion_PM();
}
if (bandera1.F2){ // Instruccion de Solicitud PM
            Acciones_Solicitud_PM();
}
if (bandera1.F3){ // Instruccion de Recibir Datos PM
            Leer_Recibir_Datos_PM();
}
if (bandera1.F4){ // Instruccion de Recibir Datos PM
            Acciones_Recibir_Datos_PM();
}

}

void RFTXmanager(){
if (RFTXtransmitting==1){
if (UARTenable){ // IFS1, U2TXIF, BIT 9 EN 1
TRANSMITIO.
            U2TXREG = RFTXarray[RFTXarrayCont];
            RFTXarrayCont += 1;
            if (RFTXarrayCont >= RFTXarrayLength){
                        RFTXarrayCont = 0;
                        RFTXarrayLength = 0;
                        RFTXtransmitting = 0;
            }
}
}
}

void main(){
inicio();
while(1){
RFTXmanager();
Comunicacion_RF();
Comunicacion_PIC_Master();}}
```

H. Código de simulación de algoritmo de navegación

1. Código de navegación del robot

```
import java.util.Random;
import javax.swing.Timer;
import java.awt.event.*;
import java.util.Random;

public class MobileRobot{
	// Map Variables
	public int[][] Map;
	// MobileRobot Variables
	public int MobileRobotRadius = 30;
	public double StepError = 0;
	public double[] MobileRobotPosition =
{450,230};
	public double[] MobileRobotRealPosition =
{450,230};
	public double[] MobileRobotTempPosition =
{450,230};
	public double[] MobileRobotGPS = {450,230};
	public double MobileRobotBattery = 100;
	public int refreshRateGPS = 500;
	public int refreshCountGPS = 0;
	// SonarSensor Variables
	public int SensorError = 0;
	public int SensorMaxDistance = 150;
	public int SensorBeamAngle = 15;
	public int[] SensorAngle =
{0,45,90,135,180,225,270,315};
	public int[] SensorDistance = new
int[SensorAngle.length];
	// Compass Variables
	public double CompassAngle = 0;
	public double CompassRealAngle = 0;
	public int CompassError = 0;
	public int refreshRateCompassError = 100;
	public int refreshCountCompassError = 0;
	// Clock Simulation
	public ActionListener clkSimulation = new
ActionListener() {public void
actionPerformed(ActionEvent
actionEvent){mainLoop();}};
	public Timer clk = new Timer (10,
clkSimulation);
	// Sensor Read Simulation
	public class readPorts extends Thread {public
void run(){getSensorDistance();}}
	public readPorts ioInterface = new
readPorts();
	// State Variables
	public boolean
StepForward=false,StepBackward=false,TurnLeft=false,
TurnRight=false;
	// Navigation Variables
	public int[] Goal = {0,0};
	public int GoalRatio = 30;
	public int MinAngleSensor = 0;
	public int EvasiveNavigationStep = 0;
	public int EvasiveNavigationSubStep = 0;
	public int CorrectiveNavigationStep = 0;
	public int CorrectiveNavigationSubStep = 0;
	public int ExtraStep = 0;
	public int PasosExtra = 30;
	public int distanciaRecorrida = 0;
	public double NavigationAngle = 0;
	public double[][] SensorHistory = new
double[2][3];
	// General purpose flags
	public boolean Reverse = false;
	public boolean Navigation = false;
	public boolean DestinationReached = false;
	public boolean PackageReceived = false;
	public boolean DirectNavigation = false;
	public boolean EvasiveNavigation = false;
	public boolean CorrectiveNavigation = false;
	public Random rand = new Random();
	public double radioAux=0,anguloAux=0;
	public String Origen="", Destino="";

	/* Inicializacion del programa */
	public MobileRobot(int[][] localMap){
		Map = localMap;
		clk.start();
	}
	/* Ciclo principal del robot */
	public void mainLoop(){
		refreshCountGPS += 1;
		if (refreshCountGPS >=
refreshRateGPS){
			refreshCountGPS = 0;
			radioAux =
150*rand.nextInt(100)/100;
			anguloAux =
2*Math.PI*rand.nextInt(100)/100;
		}
		refreshCountCompassError += 1;
		if (refreshCountCompassError >=
refreshRateCompassError){

	refreshCountCompassError = 0;
			CompassAngle =
CompassRealAngle+(2*(Math.random()-
0.5)*CompassError);
			CompassAngle =
(CompassAngle+360)%360;
		}
		MobileRobotGPS[0] =
MobileRobotRealPosition[0]+(int)(radioAux*Math.cos(a
nguloAux));
		MobileRobotGPS[1] =
MobileRobotRealPosition[1]+(int)(radioAux*Math.sin(an
guloAux));
		getSensorDistance();
		if (!Navigation){
			if (StepForward)
```

```
stepMobileRobot(2);
                        if (StepBackward)

stepMobileRobot(-2);
                        if (TurnLeft)

turnMobileRobot(1);
                        if (TurnRight)

turnMobileRobot(-1);
                }
                if (Navigation){
                        navigationAlgorithm();
                }
        }
        /* Metodo encargado de hacer
avanzar/retroceder al robot */
        public void stepMobileRobot(int step){
                int margin = 40;
                boolean condition1 =
(int)(MobileRobotRealPosition[0]+step*Math.cos(Math.P
I*(CompassRealAngle)/180))>= margin;
                boolean condition2 =
(int)(MobileRobotRealPosition[1]+step*Math.sin(Math.P
I*(CompassRealAngle)/180))>= margin;
                boolean condition3 =
(int)(MobileRobotRealPosition[0]+step*Math.cos(Math.P
I*(CompassRealAngle)/180))<= Map.length-margin;
                boolean condition4 =
(int)(MobileRobotRealPosition[1]+step*Math.sin(Math.P
I*(CompassRealAngle)/180))<= Map[0].length-margin;
                double dx,dy;
                if (condition1 && condition2 &&
condition3 && condition4){
                        dx =
step*Math.cos(Math.PI*(CompassRealAngle)/180);
                        dy =
step*Math.sin(Math.PI*(CompassRealAngle)/180);
                        if ((Math.random()*100)
> StepError){

        MobileRobotRealPosition[0] += dx;

        MobileRobotRealPosition[1] += dy;
                        }
                        MobileRobotPosition[0]
+= dx;
                        MobileRobotPosition[1]
+= dy;
                }
                consumeEnergy();
                distanciaRecorrida +=
Math.abs(step);
        }
        /* Metodo encargado de hacer girar el robot
*/
        public void turnMobileRobot(double angle){
                CompassRealAngle += angle;
                CompassAngle += angle;
                boolean cond1,cond2;
                do{
                        cond1 =
CompassAngle<0;
                        cond2 =
CompassAngle>=360;
                        if (cond1)CompassAngle
+= 360;
                        if (cond2)CompassAngle -
= 360;
                }while(cond1 || cond2);
                do{
                        cond1 =
CompassRealAngle<0;
                        cond2 =
CompassRealAngle>=360;
                        if (cond1)
        CompassRealAngle += 360;
                        if (cond2)
        CompassRealAngle -= 360;
                }while(cond1 || cond2);
                consumeEnergy();
        }
        /* Metodo para obtener la distancia
registrada por los sensores */
        public void getSensorDistance(){
                int[] SensorDistance2 = new
int[SensorAngle.length];
                for (int i=0; i<SensorAngle.length;
i++)
                        SensorDistance2[i] =
SensorMaxDistance;
                for (int i=SensorMaxDistance; i>0;
i--){
                        for (int ii=0;
ii<SensorAngle.length; ii++){
                                for (int iii=-
(int)(SensorBeamAngle/2);
iii<=(int)(SensorBeamAngle/2); iii++){
                                        int px
=
(int)((MobileRobotRadius+i)*Math.cos(Math.PI*(Sensor
Angle[ii]+iii+CompassRealAngle-
90)/180)+MobileRobotRealPosition[0]);
                                        int py
=
(int)((MobileRobotRadius+i)*Math.sin(Math.PI*(Sensor
Angle[ii]+iii+CompassRealAngle-
90)/180)+MobileRobotRealPosition[1]);
                                        if
((px>=0 && py>=0) && (px<Map.length &&
py<Map[0].length))

        if (0==(0x00ffffff & Map[px][py]))

                SensorDistance2[ii] = i;
                                }
                        }
                }
                for (int i=0; i<SensorAngle.length;
i++){
                        SensorDistance2[i] +=
(int)(SensorError*2*(Math.random()-
0.5)*SensorMaxDistance/100);
                        if (SensorDistance2[i]<0)

        SensorDistance2[i] = 0;
                }
                for (int i=0; i<SensorAngle.length;
i++)
                        SensorDistance[i] =
SensorDistance2[i];
```

```
		}
		/* Metodo que simula el consumo de energia
*/
		public void consumeEnergy(){
			MobileRobotBattery -= 0.001;
			if (MobileRobotBattery <= 0)
				MobileRobotBattery = 0;
		}
		/* Metodo para iniciar la navegacion */
		public void startNavigation(int px, int py){
			Goal[0] = px;
			Goal[1] = py;
			Navigation = true;
			DirectNavigation = true;
			EvasiveNavigation = false;
			CorrectiveNavigation = false;
			PackageReceived = false;
			DestinationReached = false;
		}
		/* Metodo para detener la navegacion */
		public void stopNavigation(){
			CorrectiveNavigationStep = 0;
			CorrectiveNavigationSubStep = 0;
			Navigation = false;
		}
		/* Algoritmo de navegacion automatica */
		public void navigationAlgorithm(){
			Navigation =
!goalReachedVerification();
			if (Navigation){

		processDirectNavigation();

		processEvasiveNavigation();

		processCorrectiveNavigation();
			}
			else{
				PackageReceived = true;
				DestinationReached =
true;
			}
		}
		/* Metodo que verifica si el destino se ha
alcanzado */
		public boolean goalReachedVerification(){
			double dx = Goal[0]-
MobileRobotPosition[0];
			double dy = Goal[1]-
MobileRobotPosition[1];
			int ratio =
(int)Math.sqrt((dx*dx)+(dy*dy));
			return ratio<GoalRatio;
		}
		/* Metodo encargado de hacer mover al robot
*/
		public boolean navigate(){
			double StepAngle = 0;
			double DifAngle = NavigationAngle-
CompassAngle;
			if (Math.abs(DifAngle)>0.2){
				if (DifAngle<-180)
		DifAngle += 360;
				if (DifAngle>180)
		DifAngle -= 360;
				if (DifAngle>0)
		StepAngle = 1;
				if (DifAngle<0)
		StepAngle = -1;
				DifAngle =
Math.abs(DifAngle);
				if (DifAngle<1)
		StepAngle = DifAngle*StepAngle;

		turnMobileRobot(StepAngle);
			}
			else{
				if (Reverse)

		stepMobileRobot(-2);
				else

		stepMobileRobot(2);
			}
			consumeEnergy();
			DifAngle = NavigationAngle-
CompassAngle;
			return Math.abs(DifAngle)<=0.2;
		}
		/* Metodo encargado de calcular el angulo
evasivo */
		public void processDirectNavigation(){
			if (!EvasiveNavigation &&
!CorrectiveNavigation){
				if (DirectNavigation){
					NavigationAngle
= (360+(int)(180*Math.atan2((Goal[1]-
MobileRobotPosition[1]),(Goal[0]-
MobileRobotPosition[0]))/Math.PI))%360;

		//NavigationAngle = 0;
					Reverse =
false;
					navigate();
				}
			}
		}
		/* Metodo encargado de determinar la
necesidad navegación evasiva */
		public void processEvasiveNavigation(){
			if (!CorrectiveNavigation){
				Reverse = false;
				// ---------- Calculate
Sensor with Minimun Distance ----------
				int index = 0;
				for(int i=1;
i<SensorDistance.length; i++)

		if(SensorDistance[i]<SensorDistance[index])

		index=i;
				// ---------- Determines
the need of the Evasive Navigation ----------
				if (!EvasiveNavigation &&
SensorDistance[index]<50){
					// ----------
Calculate Neighbor Sensor with Minimun Distance -------
---
					int
indexL=index-1, indexR=index+1;
```

```
                                        if (indexL<0)
            indexL = SensorDistance.length-1;
                                        if
(indexR>=SensorDistance.length)         indexR = 0;
                                        int
index2=indexL;
                                        if
(SensorDistance[indexR]<SensorDistance[indexL])
            index2 = indexR;
                                        // ----------
Calculate Error Angle ----------
                                        int A =
SensorDistance[index]+MobileRobotRadius, B =
SensorDistance[index2]+MobileRobotRadius;
                                        double c =
SensorAngle[index]-SensorAngle[index2];
                                        double C =
Math.sqrt((A*A)+(B*B)-
2*A*B*Math.cos(Math.PI*c/180));
                                        double b =
Math.acos(((A*A)+(C*C)-
(B*B))/(2*A*C))*180/Math.PI;
                                        double Error =
90-b;
                                        double
EvasiveAngle = 0;
                                        if (c < 0) Error
= -Error;
                                        // ----------
Calculate new Angle ----------
                                        if
((SensorAngle[index]>90&&SensorAngle[index]<270)||
(SensorAngle[index2]>90&&SensorAngle[index2]<270)
){

            EvasiveAngle =
CompassAngle+SensorAngle[index]-Error-180;

            MinAngleSensor = 4;
                                        }
                                        else{


            EvasiveAngle =
CompassAngle+SensorAngle[index]-Error;

            MinAngleSensor = 0;
                                        }
                                        if ((Error>=-80
&& Error<=80)){//||(Error<-1 && Error>=-10)){

            double DirectAngle =
(360+(int)(180*Math.atan2((Goal[1]-
MobileRobotPosition[1]),(Goal[0]-
MobileRobotPosition[0]))/Math.PI))%360;

            double DifAngle = Math.abs(DirectAngle-
CompassAngle);
                                                if
(!EvasiveNavigation && DifAngle<0.2 &&
SensorDistance[index2]<=(int)(0.9*SensorMaxDistance
)){

            NavigationAngle = (360+EvasiveAngle)%360;

            EvasiveNavigation = true;

            EvasiveNavigationStep = 0;

            MobileRobotTempPosition[0] =
MobileRobotPosition[0];

            MobileRobotTempPosition[1] =
MobileRobotPosition[1];
                                                }else
                                                if
(EvasiveNavigation)

            NavigationAngle = (360+EvasiveAngle)%360;
                                                }
                                    }
                                    else
                                    if (EvasiveNavigation){
                                            switch
(EvasiveNavigationStep){
                                            case 0:
                                                    if
(navigate())

            if (SensorDistance[MinAngleSensor]>100){

                    EvasiveNavigationStep = 1;

                    ExtraStep = 0;

            }
                                            break;
                                            case 1:

            navigate();

            ExtraStep += 1;
                                                    if
(ExtraStep>=PasosExtra)

            EvasiveNavigation = false;
                                            break;
                                            }
                                            double dx =
MobileRobotTempPosition[0] - MobileRobotPosition[0];
                                            double dy =
MobileRobotTempPosition[1] - MobileRobotPosition[1];
                                            double radius =
Math.sqrt(dx*dx+dy*dy);
                                            if (radius >
500) EvasiveNavigation = false;
                                    }
                            }
            }
            /* Metodo encargado de determinar la
necesidad navegación correctiva */
            public void processCorrectiveNavigation(){
                    // ---------- Calculate Sensor with
Minimun Distance ----------
                    int index = 0;
                    for(int i=1;
i<SensorDistance.length; i++)

            if(SensorDistance[i]<SensorDistance[index])
                                            index=i;
                    // ---------- Determines the need of
the Corrective Navigation ----------
```

```
                        if (SensorDistance[index]<20){
                                // ---------- If no
Corrective Navigation Already ----------
                                if (!CorrectiveNavigation
|| (CorrectiveNavigationStep==2)){

        SensorHistory[0][0] = index;

        SensorHistory[0][1] = CompassAngle;

        SensorHistory[0][2] = SensorDistance[index];

        EvasiveNavigation = false;

        CorrectiveNavigation = true;

        CorrectiveNavigationStep = 0;

        CorrectiveNavigationSubStep = 0;

        MobileRobotTempPosition[0] =
MobileRobotPosition[0];

        MobileRobotTempPosition[1] =
MobileRobotPosition[1];
                                }

                        }
                        // ---------- If Corrective Navigation
Already ----------
                        if (CorrectiveNavigation){
                                if
(CorrectiveNavigationStep == 0)

        correctiveNavigationPerpendicularDetection();
                                if
(CorrectiveNavigationStep == 1)

        correctiveNavigationPerpendicularSeparation()
;
                                if
(CorrectiveNavigationStep == 2)

        correctiveParallelNavigation();
                                double dx =
MobileRobotTempPosition[0] - MobileRobotPosition[0];
                                double dy =
MobileRobotTempPosition[1] - MobileRobotPosition[1];
                                double radius =
Math.sqrt(dx*dx+dy*dy);
                                if (radius > 500)
CorrectiveNavigation = false;
                        }
        }
        /* Metodo encargado de escanear la
superfície del obstáculo */
        public void
correctiveNavigationPerpendicularDetection(){
                        switch
(CorrectiveNavigationSubStep){
                        case 0:
                                turnMobileRobot(-0.3);
                                Navigation = true;
                                if
(SensorDistance[(int)(SensorHistory[0][0])]>SensorHist
ory[0][2]){

        CorrectiveNavigationSubStep = 1;
                                }
                                SensorHistory[0][1] =
CompassAngle;
                                SensorHistory[0][2] =
SensorDistance[(int)(SensorHistory[0][0])];
                                SensorHistory[1][1] =
CompassAngle;
                                SensorHistory[1][2] =
SensorDistance[(int)(SensorHistory[0][0])];
                        break;
                        case 1:
                                turnMobileRobot(0.3);
                                if
(SensorDistance[(int)(SensorHistory[0][0])]>SensorHist
ory[0][2]){

        CorrectiveNavigationStep = 1;

        CorrectiveNavigationSubStep = 0;
                                }
                                if
(SensorDistance[(int)(SensorHistory[0][0])]==SensorHi
story[0][2]){

        SensorHistory[1][1] = CompassAngle;

        SensorHistory[1][2] =
SensorDistance[(int)(SensorHistory[0][0])];
                                }
                                if
(SensorDistance[(int)(SensorHistory[0][0])]<SensorHist
ory[0][2]){

        SensorHistory[0][1] = CompassAngle;

        SensorHistory[0][2] =
SensorDistance[(int)(SensorHistory[0][0])];

        SensorHistory[1][1] = CompassAngle;

        SensorHistory[1][2] =
SensorDistance[(int)(SensorHistory[0][0])];
                                }
                        break;
                        };
        }
        /* Metodo encargado de separar al robot de
la superficie */
        public void
correctiveNavigationPerpendicularSeparation(){
                        switch
(CorrectiveNavigationSubStep){
                        case 0:

        CorrectiveNavigationSubStep = 1;
                                int Correction = 0;
                                double dAngle =
Math.abs(SensorHistory[0][1]-SensorHistory[1][1]);
                                if (dAngle>180)
        Correction = 180;
                                NavigationAngle =
Correction+(SensorHistory[0][1]+SensorHistory[1][1])/
2 - (90-SensorAngle[(int)(SensorHistory[0][0])]);
```

```
                        NavigationAngle =
(360+NavigationAngle)%360;
                        Reverse = true;
                break;
                case 1:
                        if (navigate()){
                                if
(SensorDistance[2]>40){
        CorrectiveNavigationStep = 2;

        CorrectiveNavigationSubStep = 0;
                                }
                        }
                break;
                };
        }
        /* Metodo encargado avance paralelo a la
superficie del obstaculo */
        public void correctiveParallelNavigation(){
                switch
(CorrectiveNavigationSubStep){
                case 0:

        CorrectiveNavigationSubStep = 1;
                        Reverse = false;
                        if
(SensorAngle[(int)(SensorHistory[0][0])] < 90){
                                NavigationAngle
= CompassAngle + 90;
                                MinAngleSensor
= 0;
                        }
                        if
(SensorAngle[(int)(SensorHistory[0][0])] > 90){
                                NavigationAngle
= CompassAngle - 90;
                                MinAngleSensor
= 4;
                        }
                        if
(SensorAngle[(int)(SensorHistory[0][0])] == 90){
                                NavigationAngle
= CompassAngle - 90;
                                MinAngleSensor
= 4;
                        }
                        NavigationAngle =
(360+NavigationAngle)%360;
                break;
                case 1:
                        if (navigate())
                        if
(SensorDistance[MinAngleSensor]>100){

        CorrectiveNavigationSubStep=2;
                                ExtraStep = 0;
                        }
                break;
                case 2:
                        navigate();
                        ExtraStep += 1;
                        if
(ExtraStep>=PasosExtra){

        CorrectiveNavigation = false;
        CorrectiveNavigationStep=0;

        CorrectiveNavigationSubStep=0;
                                }
                break;
                };
        }
}
```

2. Código de interfaz gráfica de la simulación

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.io.*;
import java.awt.image.PixelGrabber;
import java.nio.ByteBuffer;
import javax.swing.JFrame;
import java.awt.image.BufferedImage;
import javax.swing.Timer;
import java.util.BitSet;

public class RAEC implements MouseListener, MouseMotionListener, KeyListener {

	private MobileRobot R2D2;
	// ---------- Serial Port Variables ----------
	private TwoWaySerialComm SerialManager = new TwoWaySerialComm();
	private LeerPuerto leerPuerto = new LeerPuerto();
	private String[][] L_puertos;
	private Choice C_puertos;
	private boolean SerialConexion = false;
	// ---------- Applet Variables ----------
	private int AppletWidth = 650;
	private int AppletHeight = 500;
	// ---------- Image Variables ----------
	private Image Map;
	private Image ControlHeaderImage;
	private Image[] ControlForwardImage = new Image[3];
	private Image[] ControlBackwardImage = new Image[3];
	private Image[] ControlLeftImage = new Image[3];
	private Image[] ControlRightImage = new Image[3];
	private Image[] GoalImage = new Image[4];
	private Image GoalHeaderImage,GoalNoneImage,GoalInsertImage,GoalDeleteImage;
	private Image GoalSelectedImage,GoalStarImage,GoalStartImage,GoalStopImage,GoalTransparentImage;
	// ---------- Control Variables ----------
	private int[] ControlTitleSettings = {100,35};
	private int[] ControlForwardSettings = new int[4];
	private int[] ControlBackwardSettings = new int[4];
	private int[] ControlLeftSettings = new int[4];
	private int[] ControlRightSettings = new int[4];
	// ---------- Goal Control Variables ----------
	private int[] GoalHeaderSettings = {100,35};
	private int[] Goal1Settings = new int[4];
	private int[] Goal2Settings = new int[4];
	private boolean GoalControl1=false,GoalControl2=false;
	private boolean GoalSearched=false,GoalSelected=false;
	private boolean GoalInsert1=false,GoalInsert2=false;
	// ---------- Goal Variables ----------
	private int[][] Goal = null;
	private int GoalSelectedNo = 0;
	// ---------- Zoom Variables ----------
	private int Zoom=100, ZoomTemp;
	private int[] ClickedPosition = new int[2];
	private int[] ClickedPositionMap = new int[2];
	// ---------- Movement Variables ----------
	private int[] MovedPosition = new int[2];
	private int[] MovedPositionMap = new int[2];
	// ---------- Map Variables ----------
	private int[] MapOriginPosition = {-130,-180};
	private int MapWidth,MapHeight;
	private boolean Moving=false, Zooming=false;
	// ---------- Graphics Variables ----------
	private JFrame frame = new JFrame("Title");
	private BufferedImage Imagen = new BufferedImage(AppletWidth,AppletHeight,BufferedImage.TYPE_INT_ARGB);
	private Graphics2D g2d = Imagen.createGraphics();
	private BufferedImage dbImage = new BufferedImage(AppletWidth,AppletHeight,BufferedImage.TYPE_INT_ARGB);
	private Graphics2D dbg = dbImage.createGraphics();
	private JLabel l = new JLabel();
	// ---------- Timer Variables ----------
	public ActionListener clkSimulation = new ActionListener() {public void actionPerformed(ActionEvent actionEvent){repaint();}};
	public Timer clk = new Timer (20, clkSimulation);
	public boolean updating = false;

	public static void main (String args[])throws IOException{
		System.out.println("Hola");
		new RAEC();
	}

	//
************************************
	//	**********	INITIAL SETTINGS **********
	//
************************************

	public RAEC(){
		// Define Size

	frame.setSize(AppletWidth,AppletHeight+10);
		frame.setTitle("RAEC Robot Cartero - Simulación");
		// Add Panel
		JPanel p1 = new JPanel();

	frame.setSize(AppletWidth,AppletHeight);
		// Add Listeners
		p1.addMouseListener(this);
		p1.addMouseMotionListener(this);
```

```
                p1.addKeyListener(this);
                // Load Images
                loadImages();
                // Load MapArray and initialize MobileRobot
                int[][] MapArray = loadMapArray(Map);
                R2D2 = new MobileRobot(MapArray);
                // ConfigureSettings
                setControlSettings();
                setGoalSettings();
                // Add graphics Elements
                p1.add(l);
                frame.setLayout( new BorderLayout() );
                l.setIcon(new ImageIcon(Imagen));
                frame.add("Center",p1);
                // Add serial port choice
                C_puertos = new Choice();
                ListCommPorts Puertos = new ListCommPorts();
                L_puertos = Puertos.listPorts();
                if (L_puertos!=null){
                        for (int i=0; i<L_puertos.length; i++)

        C_puertos.addItem(L_puertos[i][0] + " - " + L_puertos[i][1]);
                }
                frame.add("South",C_puertos);
                // ----------

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
                clk.start();
                this.repaint();
        }
        /* Carga las imagenes utilizadas */
        public void loadImages(){
                Toolkit tool = Toolkit.getDefaultToolkit();
                // ---------- Mapa Image ----------
                Map = tool.getImage("Imagenes/Mapa.png");
                // ---------- Control Images ----------
                ControlHeaderImage = tool.getImage("Imagenes/Control/ControlHeader.png");
                for  (int i=0;i<3;i++){
                        ControlForwardImage[i]
        =
tool.getImage("Imagenes/Control/Adelante"+(i+1)+".png");
                        ControlBackwardImage[i]
        =
tool.getImage("Imagenes/Control/Atras"+(i+1)+".png");
                        ControlLeftImage[i]
        =
tool.getImage("Imagenes/Control/Izquierda"+(i+1)+".png");
                        ControlRightImage[i]
                =
tool.getImage("Imagenes/Control/Derecha"+(i+1)+".png");
                }
                // ---------- Goal Images ----------
                GoalHeaderImage = tool.getImage("Imagenes/Goals/GoalHeader.png");
                GoalNoneImage = tool.getImage("Imagenes/Goals/GoalNone.png");
                GoalTransparentImage = tool.getImage("Imagenes/Goals/GoalTransparent.png");
                GoalInsertImage = tool.getImage("Imagenes/Goals/GoalInsert.png");
                GoalDeleteImage = tool.getImage("Imagenes/Goals/GoalDelete.png");
                GoalSelectedImage = tool.getImage("Imagenes/Goals/GoalSelected.png");
                GoalStarImage = tool.getImage("Imagenes/Goals/GoalStar.png");
                GoalStartImage = tool.getImage("Imagenes/Goals/GoalStart.png");
                GoalStopImage = tool.getImage("Imagenes/Goals/GoalStop.png");
        }
        /* Devuelve una matriz de enteros obtenida del mapa */
        public int[][] loadMapArray(Image Map){
                do{
                        MapWidth = Map.getWidth(null);
                        MapHeight = Map.getHeight(null);
                }while(MapWidth<=0);
                int[][] MapArray = new int[MapWidth][MapHeight];
                int[] Pixels = new int[MapWidth*MapHeight];
                PixelGrabber pg = new PixelGrabber(this.Map,0,0,MapWidth,MapHeight,Pixels,0,MapWidth);
                try{
                        pg.grabPixels();
                } catch(InterruptedException e) {}
                for (int i=0; i<MapWidth; i++)
                        for (int ii=0; ii<MapHeight; ii++)

        MapArray[i][MapHeight-(ii+1)] = Pixels[ii*MapWidth+i];
                return MapArray;
        }
        /* Define las caracteristicas del mando de control */
        public void setControlSettings(){
                int[] ControlCenterPosition = {(int)(AppletHeight+AppletWidth)/2,(int)(0.78*AppletHeight)};
                int Ratio = 20;
                int[] CFS = {ControlCenterPosition[0],ControlCenterPosition[1]+(int)(1.5*Ratio),Ratio,0};
                int[] CBS = {ControlCenterPosition[0],ControlCenterPosition[1]-(int)(1.5*Ratio),Ratio,0};
                int[] CLS = {ControlCenterPosition[0]-(int)(1.5*Ratio),ControlCenterPosition[1],Ratio,0};
```

```
                int[] CRS = {ControlCenterPosition[0]+(int)(1.5*Ratio),ControlCenterPosition[1],Ratio,0};
                ControlForwardSettings = CFS;
                ControlBackwardSettings = CBS;
                ControlLeftSettings = CLS;
                ControlRightSettings = CRS;
        }
        /* Define las caracteristicas del mando de destinos */
        public void setGoalSettings(){
                int[] GoalCenterPosition = {(int)(AppletHeight+AppletWidth)/2,(int)(0.50*AppletHeight)};
                int Ratio = 20;
                int[] G1S = {GoalCenterPosition[0]-(int)(1.5*Ratio),GoalCenterPosition[1],Ratio,0};
                int[] G2S = {GoalCenterPosition[0]+(int)(1.5*Ratio),GoalCenterPosition[1],Ratio,0};
                Goal1Settings = G1S;
                Goal2Settings = G2S;
        }

        //************************************
        // ********** DRAWING METHODS **********
        //************************************

        public void paint(Graphics2D g){

        frame.resize(AppletWidth,AppletHeight+50);
                frame.setBackground(Color.black);
                drawMap(g);
                drawGoals(g);
                drawDraggedGoalControlMap(g);
                drawMobileRobot(g);
                drawMovementControl(g);
                drawGoalControl(g);
                drawDraggedGoalControl(g);
                g.setColor(Color.white);
                String NavigationType = "";
                String NavigationType2 = "";
                if (R2D2.CorrectiveNavigation){
                        NavigationType = "Navegación Correctiva";
                        NavigationType2 = " Correctivo";
                }else
                if (R2D2.EvasiveNavigation){
                        NavigationType = "Navegación Evasiva";
                        NavigationType2 = " Evasivo";
                }else
                if (R2D2.DirectNavigation){
                        NavigationType = "Navegación Directa";
                        NavigationType2 = " Directo";
                }
                if (!R2D2.Navigation){
                        NavigationType = "Destino Alcanzado";
                        NavigationType2 = "";
                }

                g.drawString("Distancia: "+R2D2.distanciaRecorrida,AppletHeight+5,AppletHeight-190);

        g.drawString("Errores:",AppletHeight+5,AppletHeight-170);
                g.drawString(" De paso: "+(int)(R2D2.StepError)+"%",AppletHeight+5,AppletHeight-150);
                g.drawString(" De brújula: "+R2D2.CompassError+"°",AppletHeight+5,AppletHeight-130);
                g.drawString(" De sensor: "+R2D2.SensorError+"%",AppletHeight+5,AppletHeight-110);


        g.drawString(NavigationType,AppletHeight+5,AppletHeight-70);
                g.drawString("Ángulo Robot: "+(int)R2D2.CompassAngle,AppletHeight+5,AppletHeight-50);

        g.drawString("Ángulo"+NavigationType2+": "+(int)R2D2.NavigationAngle,AppletHeight+5,AppletHeight-30);

                R2D2.StepForward = ControlForwardSettings[3]==2;
                R2D2.StepBackward = ControlBackwardSettings[3]==2;
                R2D2.TurnLeft = ControlLeftSettings[3]==2;
                R2D2.TurnRight = ControlRightSettings[3]==2;
        }

        /* Metodos de redibujado */
        public void repaint(){
                if (!updating)
                        update();
        }
        public void refresh(){
                if (GoalSearched)
                        GoalSearched = R2D2.Navigation;
                if (R2D2.StepForward || R2D2.StepBackward || R2D2.TurnLeft || R2D2.TurnRight || GoalSearched)
                        if (!updating)
                                update();
        }
        /* Dibuja el mapa */
        public void drawMap(Graphics2D h){
                int pX = MapOriginPosition[0];
                int pY = -(int)(MapHeight*Zoom/100)+AppletHeight-MapOriginPosition[1];
                int sizeX = (int)(MapWidth*Zoom/100);
                int sizeY = (int)(MapHeight*Zoom/100);
```

```
		h.drawImage(Map,pX,pY,sizeX,sizeY,null);
		}
		/* Dibuja los destinos */
		public void drawGoals(Graphics2D h){
			h.setColor(Color.green);
			if (Goal != null)
			for (int i=0; i<Goal.length; i++){
				int[] Obj = scaleMapObjects(Goal[i][0]-Goal[i][2],Goal[i][1]+Goal[i][2],2*Goal[i][2],2*Goal[i][2]);
				if (GoalSelected && GoalSelectedNo==i)

		h.drawImage(GoalInsertImage,Obj[0],AppletHeight-Obj[1],Obj[2],Obj[3],null);
				else

		h.drawImage(GoalTransparentImage,Obj[0],AppletHeight-Obj[1],Obj[2],Obj[3],null);
			}
		}
		/* Dibuja el robot */
		public void drawMobileRobot(Graphics2D h){
			int[] Obj;
			// ---------- Load Mobile Robot Position, Radius and Angle ----------
			int[] RobotPosition = {(int)(R2D2.MobileRobotRealPosition[0]),(int)(R2D2.MobileRobotRealPosition[1])};
			int RobotRadius = (int)(R2D2.MobileRobotRadius);
			int RobotAngle = (int)(R2D2.CompassRealAngle);
			// ---------- Draw Mobile Robot Body ----------
			h.setColor(Color.blue);
			Obj = scaleMapObjects(RobotPosition[0]-RobotRadius,RobotPosition[1]+RobotRadius,2*RobotRadius,2*RobotRadius);
			h.fillOval(Obj[0],AppletHeight-Obj[1],Obj[2],Obj[3]);
			h.setColor(Color.yellow);
			Obj = scaleMapObjects((int)(RobotPosition[0]-1*RobotRadius),(int)(RobotPosition[1]+1*RobotRadius),(int)(2*1*RobotRadius),(int)(2*1*RobotRadius));
			h.fillArc(Obj[0],AppletHeight-Obj[1],Obj[2],Obj[3],-(int)(120/2)+RobotAngle,120);
			h.setColor(Color.black);
			Obj = scaleMapObjects(RobotPosition[0]-RobotRadius,RobotPosition[1]+RobotRadius,2*RobotRadius,2*RobotRadius);
			h.drawOval(Obj[0],AppletHeight-Obj[1],Obj[2],Obj[3]);
			// ---------- Load Mobile Robot Sensors ----------
			int[] Obj1,Obj2;
			int[] SensorAngle = R2D2.SensorAngle;
			int[] SensorDistance = R2D2.SensorDistance;
			// ---------- Draw Mobile Robot Sensors ----------
			h.setColor(Color.red);
			for (int i=0; i<SensorAngle.length; i++){
				int dx2 = (int)((RobotRadius+SensorDistance[i])*Math.cos(Math.PI*(SensorAngle[i]+RobotAngle-90)/180))+RobotPosition[0];
				int dy2 = (int)((RobotRadius+SensorDistance[i])*Math.sin(Math.PI*(SensorAngle[i]+RobotAngle-90)/180))+RobotPosition[1];

				int dx1 = (int)(RobotRadius*Math.cos(Math.PI*(SensorAngle[i]+RobotAngle-90)/180))+RobotPosition[0]-((int)(SensorDistance[i]));
				int dy1 = (int)(RobotRadius*Math.sin(Math.PI*(SensorAngle[i]+RobotAngle-90)/180))+RobotPosition[1]+((int)(SensorDistance[i]));
				Obj1 = scaleMapObjects(dx1,dy1,2*SensorDistance[i],2*SensorDistance[i]);
				Obj2 = scaleMapObjects(dx2,dy2,0,0);
				int ang1 = (SensorAngle[i]+RobotAngle-90+2-(int)(R2D2.SensorBeamAngle/2))%360;

			h.fillArc(Obj1[0],AppletHeight-Obj1[1],Obj1[2],Obj1[3],ang1,R2D2.SensorBeamAngle);
			}
			// ---------- Draw Mobile Robot Known Position ----------
			h.setColor(Color.green);
			RobotPosition[0] = (int)(R2D2.MobileRobotPosition[0]);
			RobotPosition[1] = (int)(R2D2.MobileRobotPosition[1]);
			RobotRadius = (int)(R2D2.MobileRobotRadius/5);
			Obj = scaleMapObjects(RobotPosition[0]-RobotRadius,RobotPosition[1]+RobotRadius,2*RobotRadius,2*RobotRadius);
			h.fillOval(Obj[0],AppletHeight-Obj[1],Obj[2],Obj[3]);
			// ---------- Draw Mobile Robot GPS ----------
			h.setColor(Color.red);
			RobotPosition[0] = (int)(R2D2.MobileRobotGPS[0]);
			RobotPosition[1] = (int)(R2D2.MobileRobotGPS[1]);
			RobotRadius = (int)(R2D2.MobileRobotRadius/5);
			Obj = scaleMapObjects(RobotPosition[0]-RobotRadius,RobotPosition[1]+RobotRadius,2*RobotRadius,2*RobotRadius);
			h.fillOval(Obj[0],AppletHeight-Obj[1],Obj[2],Obj[3]);
		}
```

```
	/* Dibuja el mando de control */
	public void drawMovementControl(Graphics2D
h){
		h.setColor(Color.black);

	h.fillRect(AppletHeight,0,AppletWidth-
AppletHeight,AppletHeight);
		int px,py,ImageType;

		px = ControlForwardSettings[0]-
(int)(ControlTitleSettings[0]/2);
		py = AppletHeight-
(ControlForwardSettings[1]+ControlForwardSettings[2]
+ControlTitleSettings[1]+10);

	h.drawImage(ControlHeaderImage,px,py,Cont
rolTitleSettings[0],ControlTitleSettings[1],null);

		px = ControlForwardSettings[0]-
ControlForwardSettings[2];
		py = AppletHeight-
(ControlForwardSettings[1]+ControlForwardSettings[2])
;
		ImageType =
ControlForwardSettings[3];

	h.drawImage(ControlForwardImage[ImageTy
pe],px,py,2*ControlForwardSettings[2],2*ControlForwar
dSettings[2],null);

		px = ControlBackwardSettings[0]-
ControlBackwardSettings[2];
		py = AppletHeight-
(ControlBackwardSettings[1]+ControlBackwardSettings[
2]);
		ImageType =
ControlBackwardSettings[3];

	h.drawImage(ControlBackwardImage[ImageT
ype],px,py,2*ControlBackwardSettings[2],2*ControlBac
kwardSettings[2],null);

		px = ControlLeftSettings[0]-
ControlLeftSettings[2];
		py = AppletHeight-
(ControlLeftSettings[1]+ControlLeftSettings[2]);
		ImageType =
ControlLeftSettings[3];

	h.drawImage(ControlLeftImage[ImageType],
px,py,2*ControlLeftSettings[2],2*ControlLeftSettings[2]
,null);

		px = ControlRightSettings[0]-
ControlRightSettings[2];
		py = AppletHeight-
(ControlRightSettings[1]+ControlRightSettings[2]);
		ImageType =
ControlRightSettings[3];

	h.drawImage(ControlRightImage[ImageType]
,px,py,2*ControlRightSettings[2],2*ControlRightSettings
[2],null);

		h.setColor(Color.red);
		if (SerialConexion)
			h.setColor(Color.green);

		int R = 10;
		px = ControlForwardSettings[0]-R;
		py = AppletHeight-
(int)((ControlForwardSettings[1]+ControlBackwardSetti
ngs[1])/2)-R;
		ImageType =
ControlForwardSettings[3];
		h.fillOval(px,py,2*R,2*R);
	}
	/* Dibuja el mando de destinos */
	public void drawGoalControl(Graphics2D h){
		int px,py,ImageType;
		Image
Goal1Image=GoalNoneImage,Goal2Image=GoalNoneI
mage;

		if (!GoalSelected && !GoalInsert1)
			Goal1Image =
GoalInsertImage;
		if (!GoalSelected && !GoalInsert2
&& !GoalSearched)
			Goal2Image =
GoalInsertImage;
		if (GoalSearched)
			Goal2Image =
GoalStopImage;
		if (!GoalSearched && GoalSelected)
			Goal2Image =
GoalStartImage;

		px =
(int)((Goal1Settings[0]+Goal2Settings[0])/2)-
(int)(GoalHeaderSettings[0]/2);
		py = AppletHeight-
(Goal1Settings[1]+Goal1Settings[2]+GoalHeaderSetting
s[1]+10);

	h.drawImage(GoalHeaderImage,px,py,GoalHe
aderSettings[0],GoalHeaderSettings[1],null);

		px = Goal1Settings[0]-
Goal1Settings[2];
		py = AppletHeight-
(Goal1Settings[1]+Goal1Settings[2]);

	h.drawImage(Goal1Image,px,py,2*Goal1Setti
ngs[2],2*Goal1Settings[2],null);
		if (GoalSelected)

	h.drawImage(GoalDeleteImage,px,py,2*Goal
1Settings[2],2*Goal1Settings[2],null);

		px = Goal2Settings[0]-
Goal2Settings[2];
		py = AppletHeight-
(Goal2Settings[1]+Goal2Settings[2]);

	h.drawImage(Goal2Image,px,py,2*Goal2Setti
ngs[2],2*Goal2Settings[2],null);
	}
	/* Dibuja los destinos arrastrados */
	public void
drawDraggedGoalControlMap(Graphics2D h){
		if (!GoalInsert1 && !GoalInsert2)
```

```
                return;
            if (MovedPosition[0]>=AppletHeight)
                return;
            int px = MovedPositionMap[0]-30;
            int py = MovedPositionMap[1]+30;
            int[] Obj = scaleMapObjects(px,py,2*30,2*30);

        h.drawImage(GoalInsertImage,Obj[0],AppletHeight-Obj[1],Obj[2],Obj[3],null);
        }
        public void drawDraggedGoalControl(Graphics2D h){
            if (!GoalInsert1 && !GoalInsert2)
                return;
            if (MovedPosition[0]<AppletHeight)
                return;
            int px = MovedPosition[0]-Goal1Settings[2];
            int py = AppletHeight-(MovedPosition[1]+Goal1Settings[2]);

        h.drawImage(GoalInsertImage,px,py,2*Goal1Settings[2],2*Goal1Settings[2],null);
        }
        /* Cambia la escala de los objetos para ajustarlos al zoom y posicion */
        public int[] scaleMapObjects(int ObjX, int ObjY, int ObjWidth, int ObjHeight){
            int ObjXrelative = (int)(ObjX*Zoom/100)+MapOriginPosition[0];
            int ObjYrelative = (int)(ObjY*Zoom/100)+MapOriginPosition[1];
            int ObjWidthRelative = (int)(ObjWidth*Zoom/100);
            int ObjHeightRelative = (int)(ObjHeight*Zoom/100);
            int[] ObjectRelative = {ObjXrelative,ObjYrelative,ObjWidthRelative,ObjHeightRelative};
            return ObjectRelative;
        }

        // **********************************
        // ********** GOAL METHODS **********
        // **********************************

        /* Selecciona un destino en el mapa */
        public void mouseClickedGoal(){
            if (ClickedPosition[0]>AppletHeight)
                return;
            GoalSelected = false;
            if (Goal == null)
                return;
            int dx,dy,Ratio;
            for (int i=0; i<Goal.length; i++){
                dx = Goal[i][0]-ClickedPositionMap[0]; dy = Goal[i][1]-ClickedPositionMap[1];
                Ratio = (int)Math.sqrt((dx*dx)+(dy*dy));
                if (Ratio<Goal[i][2]){
                    GoalSelected = true;
                    GoalSelectedNo = i;
                }
            }

        }

        //
****************************************
        // ********** GOAL CONTROL METHODS **********
        //
****************************************

        /* Agrega un destino */
        public void insertGoal(int GoalX, int GoalY){
            int[][] GoalAux = new int[1][3];
            int GoalRatio = 30;
            if (Goal != null){
                GoalAux = new int[Goal.length+1][3];
                for (int i=0; i<Goal.length; i++){
                    GoalAux[i][0] = Goal[i][0];
                    GoalAux[i][1] = Goal[i][1];
                    GoalAux[i][2] = Goal[i][2];
                }
            }
            GoalAux[GoalAux.length-1][0] = GoalX;
            GoalAux[GoalAux.length-1][1] = GoalY;
            GoalAux[GoalAux.length-1][2] = GoalRatio;

            Goal = new int[GoalAux.length][3];
            for (int i=0; i<GoalAux.length; i++){
                Goal[i][0] = GoalAux[i][0];
                Goal[i][1] = GoalAux[i][1];
                Goal[i][2] = GoalAux[i][2];
            }
        }
        /* Quita un destino */
        public void deleteGoal(int GoalNumber){
            if (Goal.length > 1){
                int[][] GoalAux = new int[Goal.length-1][3];
                for (int i=0; i<Goal.length; i++){
                    int ii = 0;
                    if (i > GoalNumber)
                        ii = 1;
                    if (i != GoalNumber){

        GoalAux[i-ii][0] = Goal[i][0];

        GoalAux[i-ii][1] = Goal[i][1];
```

```
GoalAux[i-ii][2] = Goal[i][2];
                                }
                        }

                        Goal = new
int[GoalAux.length][3];
                        for (int i=0;
i<GoalAux.length; i++){
                                Goal[i][0] =
GoalAux[i][0];
                                Goal[i][1] =
GoalAux[i][1];
                                Goal[i][2] =
GoalAux[i][2];
                        }
                        if
(GoalNumber<GoalSelectedNo)
                                GoalSelectedNo
-= 1;
                }
                else{
                        Goal = null;
                        GoalSelectedNo = 0;
                }
        }
        public void mouseClickedGoalControl(){
                int dx,dy,Ratio;
                dx = Goal1Settings[0]-
ClickedPosition[0]; dy = Goal1Settings[1]-
ClickedPosition[1];
                Ratio =
(int)Math.sqrt((dx*dx)+(dy*dy));
                GoalControl1 = Ratio <=
Goal1Settings[2];

                dx = Goal2Settings[0]-
ClickedPosition[0]; dy = Goal2Settings[1]-
ClickedPosition[1];
                Ratio =
(int)Math.sqrt((dx*dx)+(dy*dy));
                GoalControl2 = Ratio <=
Goal2Settings[2];

                if (!GoalSelected)
                        GoalInsert1 =
GoalControl1;
                if (!GoalSelected && !GoalSearched)
                        GoalInsert2 =
GoalControl2;
                if (GoalSelected && GoalControl1){

        deleteGoal(GoalSelectedNo);
                        GoalSelected = false;
                }
                if (GoalSelected && GoalControl2
&& !GoalSearched){
                        GoalSearched = true;

        R2D2.startNavigation(Goal[GoalSelectedNo][0
],Goal[GoalSelectedNo][1]);
                        return;
                }
                if (GoalSelected && GoalControl2
&& GoalSearched){
                        GoalSearched = false;
                                R2D2.stopNavigation();
                        }
                }
        public void mouseReleasedGoalControl(){
                if (MovedPosition[0]<AppletHeight)
                        if
(MovedPositionMap[0]<MapWidth &&
MovedPositionMap[1]<MapHeight)
                                if
(MovedPositionMap[0]>0 && MovedPositionMap[1]>0)
                                        if
(GoalInsert1||GoalInsert2)

        insertGoal(MovedPositionMap[0],MovedPositio
nMap[1]);
                GoalInsert1 = false;
                GoalInsert2 = false;
        }

        //
******************************************
        // ********** CONTROL EVENT METHODS
**********
        //
******************************************

        public void mouseOverControl(int X, int Y,
boolean Reset){
                if (Reset ||
ControlForwardSettings[3]!=2)
        ControlForwardSettings[3] = 0;
                if (Reset ||
ControlBackwardSettings[3]!=2)
        ControlBackwardSettings[3] = 0;
                if (Reset ||
ControlLeftSettings[3]!=2)
        ControlLeftSettings[3] = 0;
                if (Reset ||
ControlRightSettings[3]!=2)
        ControlRightSettings[3] = 0;

                int dx,dy,Ratio;
                dx = ControlForwardSettings[0]-X;
        dy = ControlForwardSettings[1]-Y;
                Ratio =
(int)Math.sqrt((dx*dx)+(dy*dy));
                if (Ratio <=
ControlForwardSettings[2])
        ControlForwardSettings[3] = 1;

                dx = ControlBackwardSettings[0]-X;
        dy = ControlBackwardSettings[1]-Y;
                Ratio =
(int)Math.sqrt((dx*dx)+(dy*dy));
                if (Ratio <=
ControlBackwardSettings[2])
        ControlBackwardSettings[3] = 1;

                dx = ControlLeftSettings[0]-X;
        dy = ControlLeftSettings[1]-Y;
                Ratio =
(int)Math.sqrt((dx*dx)+(dy*dy));
                if (Ratio <= ControlLeftSettings[2])
                ControlLeftSettings[3] = 1;
```

```
                dx = ControlRightSettings[0]-X;
        dy = ControlRightSettings[1]-Y;
                Ratio =
(int)Math.sqrt((dx*dx)+(dy*dy));
                if (Ratio <=
ControlRightSettings[2])
        ControlRightSettings[3] = 1;
        }

        //
****************************************
        // ********** APPLET EVENT METHODS
**********
        //
****************************************

        public void start(){}
        public void run(){}
        public void stop(){}
        public void destroy(){}

        //
*************************************
        // ********** KEY EVENT METHODS
**********
        //
*************************************

        public void keyTyped(KeyEvent e){}
        public void keyReleased(KeyEvent e){}
        public void keyPressed(KeyEvent e){}

        //
****************************************
        // ********** MOUSE EVENT METHODS
**********
        //
****************************************

        public void mouseEntered(MouseEvent e){}
        public void mouseExited(MouseEvent e){}
        public void mouseDragged(MouseEvent e){
                int X = e.getX();
                int Y = AppletHeight-e.getY();
                MovedPosition[0] = X;
                MovedPosition[1] = Y;
                MovedPositionMap[0] = (int)(X-
MapOriginPosition[0])*100/Zoom;
                MovedPositionMap[1] = (int)(Y-
MapOriginPosition[1])*100/Zoom;

                if (Moving){
                        MapOriginPosition[0] -=
ClickedPosition[0]-X;
                        MapOriginPosition[1] -=
ClickedPosition[1]-Y;
                        ClickedPosition[0] = X;
                        ClickedPosition[1] = Y;
                }
                if (Zooming){
                        Zoom = ZoomTemp-
(int)(ClickedPosition[1]-Y)/2;
                        if (Zoom<30)    Zoom
= 30;
                        if (Zoom>500)   Zoom
= 500;
                        MapOriginPosition[0] = -
(int)(ClickedPositionMap[0]*Zoom/100)+ClickedPosition
[0];
                        MapOriginPosition[1] = -
(int)(ClickedPositionMap[1]*Zoom/100)+ClickedPosition
[1];
                }
                this.repaint();
        }
        public void mouseClicked(MouseEvent e){
                serialPortConnect();
                int X = e.getX();
                int Y = AppletHeight-e.getY();
                ClickedPosition[0] = X;
                ClickedPosition[1] = Y;
                ClickedPositionMap[0] = (int)(X-
MapOriginPosition[0])*100/Zoom;
                ClickedPositionMap[1] = (int)(Y-
MapOriginPosition[1])*100/Zoom;
                if (e.getButton()==1){
                        mouseClickedGoal();
                }
                this.repaint();
        }
        public void mouseMoved(MouseEvent e){
                int X = e.getX();
                int Y = AppletHeight-e.getY();
                mouseOverControl(X,Y,false);
                this.repaint();
        }
        public void mousePressed(MouseEvent e){
                int X = e.getX();
                int Y = AppletHeight-e.getY();
                ClickedPosition[0] = X;
                ClickedPosition[1] = Y;
                ClickedPositionMap[0] = (int)(X-
MapOriginPosition[0])*100/Zoom;
                ClickedPositionMap[1] = (int)(Y-
MapOriginPosition[1])*100/Zoom;
                MovedPosition[0] = X;
                MovedPosition[1] = Y;
                MovedPositionMap[0] = (int)(X-
MapOriginPosition[0])*100/Zoom;
                MovedPositionMap[1] = (int)(Y-
MapOriginPosition[1])*100/Zoom;
                ZoomTemp = Zoom;
                if (e.getButton()==1){
                        if
(ControlForwardSettings[3]==1)
        ControlForwardSettings[3] = 2;
                        if
(ControlBackwardSettings[3]==1)
        ControlBackwardSettings[3] = 2;
                        if
(ControlLeftSettings[3]==1)
        ControlLeftSettings[3] = 2;
                        if
(ControlRightSettings[3]==1)
        ControlRightSettings[3] = 2;

        mouseClickedGoalControl();
                }
                if (e.getButton()==3)
                        Moving = true;
                if (e.getButton()==2)
                        Zooming = true;
```

```
                this.repaint();
    }
        public void mouseReleased(MouseEvent e){
                int X = e.getX();
                int Y = AppletHeight-e.getY();
                Moving = false;
                Zooming = false;
                mouseOverControl(X,Y,true);
                if (e.getButton()==1){

        mouseReleasedGoalControl();
                }
                this.repaint();
        }

        //
****************************************
        //  **********  SERIAL  PORT  METHODS
**********
        //
****************************************

        public int TipoIns = 0;
        public int Instruccion = 0;
        public int nBit = 0;
        public int[] bits = new int[0];
        public byte[] bitsByte = new byte[0];
        /* Conecta con el puerto serial */
        public void serialPortConnect(){
                if (!SerialConexion){
                        int          op          =
C_puertos.getSelectedIndex();

        if(SerialManager.in==null){
                                try{

        SerialConexion = true;
                                        if
(L_puertos!=null)

        SerialManager.connect(L_puertos[op][0]);
                                        else

        SerialConexion = false;
                                }
                                catch(Exception
e){

        SerialConexion = false;
                                }
                                if
(SerialConexion)

        leerPuerto.start();
                        }
                        else
                                SerialConexion
= true;
                }
        }
        /* Lee el puerto serial */
        public void readPort(){
                byte[] buffer = new byte[1];
                byte[] buffer2 = new byte[1];
                int len = -1;
                try{
                        while          (
(SerialConexion)&&(len=SerialManager.in.read(buffer))
>-1 ){
                                ByteBuffer  buf
= ByteBuffer.wrap(buffer);
                                buffer2 = new
byte[buf.remaining()];
                                buf.get(buffer2,
0, len);

        processBufferIn(buffer2);
                                }
                        }
                        catch ( IOException e ){}
        }
        /* Procesa los datos recibidos por el puerto
serial */
        public void processBufferIn(byte[] buffer){
                int[]      bufferAux      =      new
int[buffer.length];
                for (int i=0; i<buffer.length; i++){
                        bufferAux[i]          =
fromByteToInt(buffer[i]);
                        if (TipoIns == 0){
                                if
(bufferAux[i]>=253 && bufferAux[i]<=255)

        TipoIns = bufferAux[i];
                        }
                        else{
                                if  (TipoIns  ==
255){

        ControlForwardSettings[3]=0;

        ControlBackwardSettings[3]=0;

        ControlLeftSettings[3]=0;

        ControlRightSettings[3]=0;
                                        if
(bufferAux[i] == 1)

        ControlForwardSettings[3]=2;
                                        if
(bufferAux[i] == 2)

        ControlBackwardSettings[3]=2;
                                        if
(bufferAux[i] == 3)

        ControlLeftSettings[3]=2;
                                        if
(bufferAux[i] == 4)

        ControlRightSettings[3]=2;

        TipoIns = 0;

        this.repaint();
                                }
                                if  (TipoIns  ==
254){
                                        if
(bufferAux[i] == 6){
```

```
            Instruccion = 6;

            byte[] msj = new byte[10];

            msj[0] = get8Bit(254);

            msj[1] = get8Bit(6);

            for(int ii=0; ii<8; ii++)

                        msj[ii+2]                                   =
get8Bit((int)(255*R2D2.SensorDistance[ii]/R2D2.Sensor
MaxDistance));

            try{

                        SerialManager.out.write(msj);

            }catch ( IOException e ){}
                                                                    }
                                                                    if
(bufferAux[i] == 7){

            Instruccion = 7;

            byte[] msj = new byte[4];

            msj[0] = get8Bit(254);

            msj[1] = get8Bit(7);



            int                     Angle                   =
(int)(R2D2.CompassAngle*2047/359);

            int AngleH = (int)(Angle/256);

            int AngleL = Angle-AngleH*256;

            msj[2] = get8Bit(AngleH);

            msj[3] = get8Bit(AngleL);

            try{

                        SerialManager.out.write(msj);

            }catch ( IOException e ){}
                                                                    }
                                                                    if
(bufferAux[i] == 8){

            Instruccion = 8;

            byte[] msj = new byte[6];

            msj[0] = get8Bit(254);

            msj[1] = get8Bit(8);



            int posx = (int)(R2D2.MobileRobotGPS[0]);

            int posxH = (int)(posx/100);

            int posxL = posx-posxH*100;

            int posy = (int)(R2D2.MobileRobotGPS[1]);

            int posyH = (int)(posy/100);

            int posyL = posy-posyH*100;

            msj[2] = get8Bit(posyH);

            msj[3] = get8Bit(posyL);

            msj[4] = get8Bit(posxH);

            msj[5] = get8Bit(posxL);

            try{

                        SerialManager.out.write(msj);

            }catch ( IOException e ){}
                                                                    }
                                                                    if
(bufferAux[i] == 9){

            Instruccion = 9;

            byte[] msj = new byte[6];

            msj[0] = get8Bit(254);

            msj[1] = get8Bit(9);



            int                     posx                    =
(int)(R2D2.MobileRobotPosition[0]);

            int posxH = (int)(posx/256);

            int posxL = posx-posxH*256;

            int                     posy                    =
(int)(R2D2.MobileRobotPosition[1]);

            int posyH = (int)(posy/256);

            int posyL = posy-posyH*256;

            msj[2] = get8Bit(posyH);

            msj[3] = get8Bit(posyL);

            msj[4] = get8Bit(posxH);

            msj[5] = get8Bit(posxL);

            try{

                        SerialManager.out.write(msj);

            }catch ( IOException e ){}
                                                                    }
```

```
                                        if
(bufferAux[i] == 10){
        Instruccion = 10;
        byte[] msj = new byte[3];
        msj[0] = get8Bit(254);
        msj[1] = get8Bit(10);

        int                bat                =
(int)(255*R2D2.MobileRobotBattery/100);
        msj[2] = get8Bit(bat);
        try{
                SerialManager.out.write(msj);
        }catch ( IOException e ){}
                                        }
                                        if
(bufferAux[i] == 11){
        Instruccion = 11;
        byte[] msj = new byte[3];
        msj[0] = get8Bit(254);
        msj[1] = get8Bit(11);

        int estado = 0;
        if (R2D2.StepForward)       estado += 1;
        if (R2D2.StepBackward)      estado += 2;
        if (R2D2.TurnLeft)          estado += 4;
        if (R2D2.TurnRight)         estado += 8;
        if (R2D2.Navigation)
        estado += 16;
        if (R2D2.DestinationReached) estado += 32;
        if (R2D2.PackageReceived)
        estado += 64;
        msj[2] = get8Bit(estado);
        try{
                SerialManager.out.write(msj);
        }catch ( IOException e ){}
                                        }
        TipoIns = 0;
        Instruccion = 0;
        this.repaint();
                                        }
                                        if (TipoIns ==
253){
                                                if
(Instruccion == 0){
        if (bufferAux[i] >= 12 && bufferAux[i] <=
15)
                Instruccion = bufferAux[i];
                                                }
                                                else{
        if (Instruccion == 12){
                if (nBit == 0)
                        bitsByte = new byte[10];
                if (nBit >= 0 && nBit <= 9){
                        bitsByte[nBit] = buffer[i];
                        nBit += 1;
                }
                if (nBit == 10){
                        nBit = 0;
                        TipoIns = 0;
                        Instruccion = 0;
                        String   Oficina   =   new
String(bitsByte);
                        R2D2.Origen = Oficina;
                }
        }
        if (Instruccion == 13){
                if (nBit == 0)
                        bitsByte = new byte[10];
                if (nBit >= 0 && nBit <= 9){
                        bitsByte[nBit] = buffer[i];
                        nBit += 1;
                }
                if (nBit == 10){
                        nBit = 0;
                        TipoIns = 0;
```

```
                        Instruccion = 0;
                        String Oficina = new
String(bitsByte);
                        R2D2.Destino = Oficina;
                }
        }
        if (Instruccion == 15){
                switch (nBit){
                case 0:
                        nBit = 1;
                        bits = new int[4];
                        bits[0] = bufferAux[i];
                break;
                case 1:
                        nBit = 2;
                        bits[1] = bufferAux[i];
                break;
                case 2:
                        nBit = 3;
                        bits[2] = bufferAux[i];
                break;
                case 3:
                        nBit = 0;
                        TipoIns = 0;
                        Instruccion = 0;
                        bits[3] = bufferAux[i];
                        int px =
bits[0]*256+bits[1];
                        int py =
bits[2]*256+bits[3];
                        GoalSearched = true;

        R2D2.startNavigation(px,py);
                break;
                }
        }
        }
                }
                }
        }
        /* Convierte un byte a entero */
        public int fromByteToInt(byte b){
                int num = 0;
                for (int i = 0; i < 8; i++){
                        if ((b & 1) == 1)
                                num =
num+(int)Math.pow(2,i);
                        b >>= 1;
                }
                return num;
        }
        /* Convierte un entero a byta */
        public byte get8Bit(int num){
                if (num<0)              num
= 0;
                if (num>255)    num = 255;
                BitSet Num = new BitSet(8);
                for (int i=7; i>=0; i--){

        Num.set(i,num>=Math.pow(2,i));
                        if (Num.get(i))
                                num     -=
Math.pow(2,i);
                }

                byte NUM = 0;
                for (int i=0; i<8; i++)
                        if (Num.get(i))   NUM
|= 1<<(i%8);
                return NUM;
        }
        /* Thread para leer el puerto serial */
        public class LeerPuerto extends Thread {
                LeerPuerto(){}
                public void run(){
                        readPort();
                }
        }

        // ********************************
        //      **********      UPDATE      METHOD
**********
        // ********************************

        public void update(){
                updating = true;

        dbg.setColor(frame.getBackground());
                dbg.fillRect(0,                 0,
frame.getSize().width, frame.getSize().height);

        dbg.setColor(frame.getForeground());
                paint(dbg);
                g2d.drawImage(dbImage,  0,  0,
null);
                l.setIcon(new ImageIcon(Imagen));
                updating = false;
        }
}
```

I. Código de control implementado en el servidor

```
<?php
$con = null;
$socket = null;
$estado = null;
$PosicionActual = null;
$FailMessageCont = 0;
$Solicitud = 6;
$horaSolicitud = time(date("Y-m-d H:i:s"));

// Conecta con la base de datos RAEC
function databaseConect() {
	global $con;
	$con = mysql_connect('localhost','usuario','password');
	if (!$con){die('Could not connect: ' . mysql_error());}
	mysql_select_db("RAEC", $con);
}

// Establece la conexión con el módulo de comunicación inalámbrica
function setConexion() {
	global $estado;
	getEstado();
	// Establece la conexión si no está establecida
	if ($estado=="Off")$estado = "On";
	else
		$estado = "Off";
	// Actualiza el estado de conexión con el robot
	setEstado($estado);
}

// Obtiene el estado de la conexión inalámbrica
function getEstado() {
	global $estado;
	$table = "EstadoRobot";
	$variable = "RobotOnOff";
	$sql="SELECT Estado FROM ".$table." WHERE Variable = '".$variable."'";
	$result = mysql_query($sql);
	$row = mysql_fetch_array($result);
	$estado = $row[0];
}

// Asigna el estado de la conexión inalámbrica
function setEstado($estado) {
	$table = "EstadoRobot";
	$variable = "RobotOnOff";
	$sql = "UPDATE ".$table." SET Estado = '".$estado."' WHERE Variable = '".$variable."'";
	mysql_query($sql);
}

// Devuelve el nodo u oficina mas cercano
function getPosicionCercana() {
	$sql="SELECT Estado FROM EstadoRobot WHERE Variable = 'LongitudPos'";
	$result = mysql_query($sql);
	$row = mysql_fetch_array($result);
	$robotLon = (int)$row[0];
	$sql="SELECT Estado FROM EstadoRobot WHERE Variable = 'LatitudPos'";
	$result = mysql_query($sql);
	$row = mysql_fetch_array($result);
	$robotLat = (int)$row[0];

	$Oficina = "";
	$disMin = "";

	$sql="SELECT * FROM Oficinas";
	$result = mysql_query($sql);
	while ($row = mysql_fetch_array($result)){
		$lon = (int)$row['Longitud'];
		$lat = (int)$row['Latitud'];
		$dlon = $lon - $robotLon;
		$dlat = $lat - $robotLat;
		$dis = sqrt(($dlon*$dlon)+($dlat*$dlat));
		if (($disMin=="")||($dis<$disMin)){
			$Oficina = $row['Oficina'];
			$disMin = $dis;
		}
	}

	$sql="SELECT * FROM Nodos";
	$result = mysql_query($sql);
	while ($row = mysql_fetch_array($result)){
		$lon = (int)$row['Longitud'];
		$lat = (int)$row['Latitud'];
		$dlon = $lon - $robotLon;
		$dlat = $lat - $robotLat;
		$dis = sqrt(($dlon*$dlon)+($dlat*$dlat));
		if (($disMin=="")||($dis<$disMin)){
			$Oficina = $row['Nodo'];
			$disMin = $dis;
		}
	}
	return $Oficina;
}

// Crea el Socket a ser utilizado para enviar el datagrama
function createUDPSocket() {
	global $socket;
	$socket = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
	socket_bind($socket, '192.168.2.120', 47676);
	$timeout = array('sec'=>1,'usec'=>000000);
	socket_set_option($socket,SOL_SOCKET,SO_RCVTIMEO,$timeout);
}

// Envia un datagrama al Socket especificado
function sendUDPMsg($sock,$msg) {
	$len = strlen($msg);
	socket_sendto($sock, $msg, $len, 0, 'siteplayer-telnet-7a3e81.local', 10001);
}

// Lee un puerto del Socket especificado
function receiveUDPMsg($sock) {
```

```
	$from = '';
	$port = 47676;
	socket_recvfrom($sock, $buf, 512, 0, $from,
$port);
	return $buf;
}

// Procesa e interpreta el mensaje UDP recibido
function processReceivedInfo($buf) {
	$estado = "";
	$variable = "";
	$table = "EstadoRobot";

	// Genera un buffer de enteros a partir del
buffer de Strings
	for ($i=0; $i<strlen($buf); $i++)
		$Buf[$i] = ord($buf[$i]);

	// Procesa datos de sensores
	if($Buf[0]==254 && strlen($buf)>=3 &&
$Buf[1]==6){
		for($i=2; $i<strlen($buf); $i++) {
			$variable = "Sensor".($i-
1);
			$estado =
$Buf[$i]*150/255;
			$sql = "UPDATE ".$table."
SET Estado='".$estado."' WHERE
Variable='".$variable."'";
			mysql_query($sql);
			if($i>=9) $i =
strlen($buf);
		}
	}
	// Procesa datos de brujula
	if($Buf[0]==254 && strlen($buf)==4 &&
$Buf[1]==7){
		$variable = "Brujula";
		$estado =
($Buf[2]*256+$Buf[3])*359/2047;
		$sql = "UPDATE ".$table." SET
Estado='".$estado."' WHERE Variable='".$variable."'";
		mysql_query($sql);
	}
	// Procesa datos de GPS
	if($Buf[0]==254 && strlen($buf)==6 &&
$Buf[1]==8){
		$variable = "LatitudGPS";
		$estado = ($Buf[2]*100+$Buf[3]);
		$sql = "UPDATE ".$table." SET
Estado='".$estado."' WHERE Variable='".$variable."'";
		mysql_query($sql);
		$variable = "LongitudGPS";
		$estado = ($Buf[4]*100+$Buf[5]);
		$sql = "UPDATE ".$table." SET
Estado='".$estado."' WHERE Variable='".$variable."'";
		mysql_query($sql);
	}
	// Procesa datos de posición
	if($Buf[0]==254 && strlen($buf)==6 &&
$Buf[1]==9){
		$variable = "LatitudPos";
		$estado = ($Buf[2]*256+$Buf[3]);
		$sql = "UPDATE ".$table." SET
Estado='".$estado."' WHERE Variable='".$variable."'";
		mysql_query($sql);
		$variable = "LongitudPos";
		$estado = ($Buf[4]*256+$Buf[5]);
		$sql = "UPDATE ".$table." SET
Estado='".$estado."' WHERE Variable='".$variable."'";
		mysql_query($sql);
	}
	// Procesa datos de bateria
	if($Buf[0]==254 && strlen($buf)==3 &&
$Buf[1]==10){
		$variable = "Bateria";
		$estado = ($Buf[2]*100/255);
		$sql = "UPDATE ".$table." SET
Estado='".$estado."' WHERE Variable='".$variable."'";
		mysql_query($sql);
	}
}

// Se encarga de manejar la solicitud de datos al robot
function manageSolicitud() {
	global $socket;
	global $Solicitud;
	global $FailMessageCont;

	// Si no está en modo de monitoreo sale del
método
	$sql = "SELECT Estado FROM EstadoRobot
WHERE Variable = 'RobotMonitor'";
	$result = mysql_query($sql);
	$row = mysql_fetch_array($result);
	if ($row[0]!="On")
		return;

	// Envía UDP solicitando información al robot
	sendUDPMsg($socket,chr(254).chr($Solicitud)
);
	$buf = receiveUDPMsg($socket);

	// Si no se recibió respuesta, verifica si la
	// cantidad de intentos fallidos llegó al
máximo
	// y termina la conexión
	if($buf==null){
		$FailMessageCont =
$FailMessageCont + 1;
		if ($FailMessageCont >= 10)
			setEstado("Off");
		return;
	}

	// Se procesa la respuesta recibida
	$FailMessageCont = 0;
	processReceivedInfo($buf);

	// Cambia la solicitud a pedir
	$Solicitud = $Solicitud+1;
	if($Solicitud>10)	$Solicitud = 6;
}


// Convierte un String a un Arreglo de 1's y 0's
function stringToArray($buf) {
	$Buf = ord($buf);
	$BufArray = array(0,0,0,0,0,0,0,0);

	$BufArray[7] = floor($Buf/128);
	$Buf[2] = $Buf - $BufArray[7]*128;
```

```
		$BufArray[6] = floor($Buf/64);
		$Buf[2] = $Buf - $BufArray[6]*64;
		$BufArray[5] = floor($Buf/32);
		$Buf[2] = $Buf - $BufArray[5]*32;
		$BufArray[4] = floor($Buf/16);
		$Buf[2] = $Buf - $BufArray[4]*16;
		$BufArray[3] = floor($Buf/8);
		$Buf[2] = $Buf - $BufArray[3]*8;
		$BufArray[2] = floor($Buf/4);
		$Buf[2] = $Buf - $BufArray[2]*4;
		$BufArray[1] = floor($Buf/2);
		$Buf[2] = $Buf - $BufArray[1]*2;
		$BufArray[0] = $Buf[2];

		return $BufArray;
}

// Envía al robot la oficina de origen del paquete
function sendOficinaOrigen($Oficina) {
		global $socket;
		for ($i=0; $i<10; $i++){
			if ($i < strlen($Oficina))
		$O[$i] = ord($Oficina[$i]);
			else $O[$i] = 0;
		}
		$mensaje =
array(253,12,$O[0],$O[1],$O[2],$O[3],$O[4],$O[5],$O[
6],$O[7],$O[8],$O[9]);
		$mensajeString = "";
		for ($i=0; $i<count($mensaje); $i++)
			$mensajeString =
$mensajeString.chr($mensaje[$i]);
		sendUDPMsg($socket,$mensajeString);
}

// Envía al robot la oficina destino del paquete
function sendOficinaDestino($Oficina) {
		global $socket;
		for ($i=0; $i<10; $i++){
			if ($i < strlen($Oficina))
		$O[$i] = ord($Oficina[$i]);
			else $O[$i] = 0;
		}
		$mensaje =
array(253,13,$O[0],$O[1],$O[2],$O[3],$O[4],$O[5],$O[
6],$O[7],$O[8],$O[9]);
		$mensajeString = "";
		for ($i=0; $i<count($mensaje); $i++)
			$mensajeString =
$mensajeString.chr($mensaje[$i]);
		sendUDPMsg($socket,$mensajeString);
}

// Envía al robot la gaveta donde está o debe ser
colocado el paquete
function sendGaveta($Gaveta) {
		global $socket;
		for ($i=0; $i<10; $i++){
			if ($i < strlen($Gaveta))
		$O[$i] = ord($Gaveta[$i]);
			else $O[$i] = 0;
		}
		$mensaje =
array(253,14,$O[0],$O[1],$O[2],$O[3],$O[4],$O[5],$O[
6],$O[7],$O[8],$O[9]);
		$mensajeString = "";
		for ($i=0; $i<count($mensaje); $i++)
			$mensajeString =
$mensajeString.chr($mensaje[$i]);
		sendUDPMsg($socket,$mensajeString);
}

// Envía al robot la longitud y latitud a donde debe
dirigirse
function sendDestino($longitud,$latitud) {
		global $socket;
		$lonH = intval($longitud/256);
		$lonL = intval($longitud-($lonH*256));
		$latH = intval($latitud/256);
		$latL = intval($latitud-($latH*256));

		$mensaje =
array(253,15,$lonH,$lonL,$latH,$latL);
		$mensajeString = "";
		for ($i=0; $i<count($mensaje); $i++)
			$mensajeString =
$mensajeString.chr($mensaje[$i]);
		sendUDPMsg($socket,$mensajeString);
}

// Devuelve un arreglo con los nodos de la red así como
sus nodos
// y oficinas vecinas y las distancias entre cada uno de
ellos
function getNodos() {
		$sql = "SELECT * FROM Nodos";
		$result = mysql_query($sql);

		while ($row = mysql_fetch_array($result)) {
			$nodo = $row['Nodo'];
			$nodoLon = (int)($row['Longitud']);
			$nodoLat = (int)($row['Latitud']);

			for ($ii=1; $ii<=4; $ii++){
				$vecino =
$row['Vecino'.$ii];
				if ($vecino != ""){
					$sql = "SELECT
* FROM Nodos WHERE Nodo = '".$vecino."'";
					$result2 =
mysql_query($sql);
					$row2 =
mysql_fetch_array($result2);

					$sql = "SELECT
* FROM Oficinas WHERE Oficina = '".$vecino."'";
					$result3 =
mysql_query($sql);
					$row3 =
mysql_fetch_array($result3);

					if ($row2 !=
false){

		$nodos[$nodo][$vecino]['Longitud']
=(int)$row2['Longitud'];

		$nodos[$nodo][$vecino]['Latitud'] =
(int)$row2['Latitud'];
$dLon
= $nodos[$nodo][$vecino]['Longitud'] - $nodoLon;
```

```
                                        $dLat = $nodos[$nodo][$vecino]['Latitud'] - $nodoLat;
                                        $dis = sqrt(($dLon*$dLon) + ($dLat*$dLat));

                $nodos[$nodo][$vecino]['Distancia'] = (int)$dis;
                                }
                                else if ($row3 != false){

                $nodos[$nodo][$vecino]['Longitud'] = (int)$row3['Longitud'];

                $nodos[$nodo][$vecino]['Latitud'] = (int)$row3['Latitud'];
                                        $dLon = $nodos[$nodo][$vecino]['Longitud'] - $nodoLon;
                                        $dLat = $nodos[$nodo][$vecino]['Latitud'] - $nodoLat;
                                        $dis = sqrt(($dLon*$dLon) + ($dLat*$dLat));

                $nodos[$nodo][$vecino]['Distancia'] = (int)$dis;
                                }
                        }
                }
        }

        $sql = "SELECT * FROM Oficinas";
        $result = mysql_query($sql);
        while ($row = mysql_fetch_array($result)) {
                $oficina = $row['Oficina'];
                $oficinaLon = (int)($row['Longitud']);
                $oficinaLat = (int)($row['Latitud']);

                $sql = "SELECT * FROM Nodos";
                $sql .= " WHERE Vecino1 = '".$oficina."'";
                $sql .= " OR Vecino2 = '".$oficina."'";
                $sql .= " OR Vecino3 = '".$oficina."'";
                $sql .= " OR Vecino4 = '".$oficina."'";
                $result2 = mysql_query($sql);
                while ($row2 = mysql_fetch_array($result2)){
                        $nodo = $row2['Nodo'];

                $nodos[$oficina][$nodo]['Longitud'] = $row2['Longitud'];

                $nodos[$oficina][$nodo]['Latitud'] = $row2['Latitud'];
                        $dLon = $nodos[$oficina][$nodo]['Longitud'] - $oficinaLon;
                        $dLat = $nodos[$oficina][$nodo]['Latitud'] - $oficinaLat;
                        $dis = sqrt(($dLon*$dLon) + ($dLat*$dLat));

                $nodos[$oficina][$nodo]['Distancia'] = (int)$dis;
                        }
                }
                return $nodos;
}

// Devuelve un arreglo con todas las rutas posibles que se generan
// a partir del nodo indicado
function getTree($nodo,$nodos,$rama) {
        $ramas = array();
        if (array_key_exists($nodo, $nodos)){
                $vecinos = array_keys($nodos[$nodo]);
                for ($i=0; $i<count($vecinos); $i++)
                        if (!array_key_exists($vecinos[$i], $rama)){
                                $ramaAux = $rama;

        $ramaAux[$vecinos[$i]] = $nodos[$nodo][$vecinos[$i]];

        array_push($ramas,$ramaAux);
                        }
        }

        $ramas2 = array();
        for ($i=0; $i<count($ramas); $i++){
                $ramaAux = $ramas[$i];
                $ramaLen = count($ramaAux);
                $nodosNombre = array_keys($ramaAux);
                $ramasHijo = getTree($nodosNombre[$ramaLen-1],$nodos,$ramaAux);
                for ($ii=0; $ii<count($ramasHijo); $ii++)

        array_push($ramas2,$ramasHijo[$ii]);
        }

        for ($i=0; $i<count($ramas2); $i++)
                if (count($ramas2[$i]) > 0)

        array_push($ramas,$ramas2[$i]);

        return $ramas;
}

// Devuelve las rutas que terminan en el nodo u oficina indicado
function getRoutesTo($nodosFin, $ramas) {
        $rutas = array();
        for ($i=0; $i<count($ramas); $i++) {
                $rama = $ramas[$i];
                $nodos = array_keys($rama);
                if (in_array($nodos[count($nodos)-1],$nodosFin))

        array_push($rutas,$rama);
        }
        return $rutas;
```

```
}

// Devuelve la ruta con la distancia mínima
function getMinRoute($rutas) {
	if (count($rutas)==0)
		return array();

	$Distancias = array();
	for ($i=0; $i<count($rutas); $i++){
		$ruta = $rutas[$i];
		$nodos = array_keys($ruta);
		$distancia = 0;
		for ($ii=0; $ii<count($ruta); $ii++)
			$distancia = $distancia + $ruta[$nodos[$ii]]['Distancia'];
		array_push($Distancias,$distancia);
	}
	if (count($Distancias) > 0){
		$DisMin = $Distancias[0];
		$iMin = 0;
	}
	for ($i=0; $i<count($Distancias); $i++){
		if ($Distancias[$i]<$DisMin){
			$DisMin = $Distancias[$i];
			$iMin = $i;
		}
	}
	return $rutas[$iMin];
}

// Se encarga de enviar los mensajes al robot que contienen la oficina de origen, destino y gaveta
// según sea el envío que se está atendiendo
function sendMessage($estado,$posicion) {
	if ($estado[5] == 1)
		return false;

	$sql="SELECT * FROM Envios WHERE Estado = 'Paquete Recogido' AND Destino = '".$posicion."' ORDER BY Origen";
	$result = mysql_query($sql);
	if ($row = mysql_fetch_array($result)){
		if($estado[6] == 1){
			$sql = "UPDATE Envios SET Estado = 'Paquete Entregado' WHERE Estado = 'Paquete Recogido' AND Destino = '".$posicion."'";
			$sql .= " ORDER BY Origen";
			mysql_query($sql);
			return false;
		}
		else {
			sendGaveta("Gaveta ".$row['Gaveta']);

	sendOficinaOrigen($row['Origen']);
			return false;
		}
	}

	$gavetas = array(0,0,0,0,0);
	$sql="SELECT * FROM Envios WHERE Estado = 'Paquete Recogido' OR Estado = 'Lista de Espera'";
	$result = mysql_query($sql);
	while($row = mysql_fetch_array($result))
		$gavetas[(int)$row['Gaveta']-1] = 1;
	$gaveta = 0;
	for ($i=4; $i>=0; $i--)
		if ($gavetas[$i]==0)
			$gaveta = $i+1;

	if ($gaveta==0)
		return true;

	$sql="SELECT * FROM Envios WHERE Estado = 'Lista de Espera' AND Origen = '".$posicion."' ORDER BY Destino";
	$result = mysql_query($sql);
	if ($row = mysql_fetch_array($result)){
		if($estado[6] == 1){
			$existe = true;
			$row2 = $row;
			while ($row2['Gaveta']=="" && $existe){
				$existe = false;
				if ($row2 = mysql_fetch_array($result))

	$existe = true;
			}
			if (!$existe) return true;
			$sql = "UPDATE Envios SET Estado = 'Paquete Recogido' WHERE Estado = 'Lista de Espera' AND Origen = '".$posicion."'";
			$sql .= " AND Usuario = '".$row2['Usuario']."' AND Code = '".$row2['Code']."'";
			//mysql_query($sql);
		}
		else {
			if ($row['Gaveta']==""){
				$sql = "UPDATE Envios SET Gaveta = '".$gaveta."' WHERE Estado = 'Lista de Espera' AND Origen = '".$posicion."'";
				$sql .= " AND Usuario = '".$row['Usuario']."' AND Code = '".$row['Code']."'";

	mysql_query($sql);

	sendGaveta("Gaveta ".$gaveta);

	sendOficinaDestino($row['Destino']);
			}
		}
		return false;
	}
	return true;
}

// Se encarga de manejar el control automático de envíos del robot
function manageControlAuto() {
	global $TareaActual;
	global $horaSolicitud;
	global $FailMessageCont;
	global $socket;

	// Verifica si transcurrió tiempo de espera
```

```
	$horaActual = time(date("Y-m-d H:i:s"));
	if (($horaActual-$horaSolicitud) < 3)
		return;
	$horaSolicitud = $horaActual;

	// Procesa si el control automático esta activado
	$sql="SELECT Estado FROM EstadoRobot WHERE Variable = 'ControlAuto'";
	$result = mysql_query($sql);
	$row = mysql_fetch_array($result);
	if ($row[0]!="On")
		return;

	// Envía UDP solicitando estado del robot
	sendUDPMsg($socket,chr(254).chr(11));
	$buf = receiveUDPMsg($socket);

	// Si no se recibió respuesta, verifica si la
	// cantidad de intentos fallidos llegó al máximo
	// y termina la conexión
	if($buf==null){
		$FailMessageCont = $FailMessageCont + 1;
		if ($FailMessageCont >= 10)
			setEstado("Off");
		return;
	}
	$FailMessageCont = 0;

	// Separa el byte de estado en un arreglo de enteros
	if (strlen($buf)==3 && ord($buf[0])==254 && ord($buf[1])==11)
		$Estado = stringToArray($buf[2]);
	if (strlen($buf)!=3 || $Estado[4]==1)
		return;

	// Envía UDP solicitando posicion
	sendUDPMsg($socket,chr(254).chr(9));
	$buf = receiveUDPMsg($socket);
	if ($buf != null)
		processReceivedInfo($buf);
	$PosicionActual = getPosicionCercana();

	if (!sendMessage($Estado,$PosicionActual))
		return;
	$sql = "UPDATE EstadoRobot SET Estado='".$PosicionActual."' WHERE Variable='Datos'";
	mysql_query($sql);
	$sql = "UPDATE Envios SET Estado = 'Paquete Recogido' WHERE Estado='Lista de Espera' AND Origen='".$PosicionActual."'";
	mysql_query($sql);
	$sql = "UPDATE Envios SET Estado = 'Paquete Entregado' WHERE Estado='Paquete Recogido' AND Destino='".$PosicionActual."'";
	mysql_query($sql);

	$objetivos = array();
	$sql="SELECT * FROM Envios WHERE Estado = 'Paquete Recogido'";
	$result = mysql_query($sql);
	while ($row = mysql_fetch_array($result)){
		//setEstado("Off");
		array_push($objetivos,$row['Destino']);
	}
	if (count($objetivos)<5){
		$sql="SELECT * FROM Envios WHERE Estado = 'Lista de Espera'";
		$result = mysql_query($sql);
		while ($row = mysql_fetch_array($result)){
			//setEstado("Off");

		array_push($objetivos,$row['Origen']);
		}
	}
	$inicio[$PosicionActual]['Longitud'] = 0;
	$inicio[$PosicionActual]['Latitud'] = 0;
	$inicio[$PosicionActual]['Distancia'] = 0;
	$nodos = getNodos();
	$arbol = getTree($PosicionActual,$nodos,$inicio);
	$rutasValidas = getRoutesTo($objetivos,$arbol);
	$ruta = getMinRoute($rutasValidas);

	$nodos = array_keys($ruta);
	if (count($nodos)==0) return;
	$nodo = $nodos[count($nodos)-1];
	$objetivo = $nodos[1];
	$Oficina = $objetivo;

	$sql="SELECT * FROM Oficinas WHERE Oficina = '".$Oficina."'";
	$result = mysql_query($sql);
	if ($row = mysql_fetch_array($result)){
		$longitud = $row['Longitud'];
		$latitud = $row['Latitud'];
		sendDestino($longitud,$latitud);
	}

	$sql="SELECT * FROM Nodos WHERE Nodo = '".$Oficina."'";
	$result = mysql_query($sql);
	if ($row = mysql_fetch_array($result)){
		$longitud = $row['Longitud'];
		$latitud = $row['Latitud'];
		sendDestino($longitud,$latitud);
	}
}

// Conecta con la base de datos
databaseConect();
// Conecta con el dispositivo inalámbrico
setConexion();

if($estado=="On"){
	vaciarGavetas();
	createUDPSocket();
	while($estado=="On") {
		manageSolicitud();
		manageControlAuto();
		getEstado();
		usleep(50000);
	}
}
?>
```

XIV. GLOSARIO

Almanaque: Información enviada de forma periódica por los satélites GPS, informando sobre ellos mismos y el resto de satélites.

ARP: Protocolo de resolución de direcciones del inglés Address Resolution Protocol.

ARPANET: Red de la agencia de proyectos de investigación avanzada del inglés Advanced Research Projects Agency Network.

DARPA: Agencia de proyectos de investigación avanzada para la defensa del inglés Defence Advanced Research Projects Agency.

DCE: Data Communication Equipment. Equipo de comunicación de datos.

DIP: Dual In Line Package. Tipo de empaquetado utilizado en circuitos integrados. Utiliza tecnología "threw hole".

DNS: Servidor de nombre de dominio del inglés Domain Name Server.

DTE: Data Terminal Equipment. Equipo terminal de datos.

EasyPIC: Es una herramienta de desarrollo para los microcontroladores PIC.

EEPROM: Electrically Erasable Programmable Read Only Memory. Tipo de memoria que puede ser programada, borrada y reprogramada eléctricamente.

Efemérides: Información enviada por los satélites GPS, dando la posición precisa de los mismos.

FTP: Protocolo de transferencia de archivos del inglés File Transfer Protocol.

HTTP: Protocolo de transferencia de hipertexto del inglés Hypertext Transfer Protocol.

Hz: Dimensional de frecuencia.

IAB: Junta de arquitectura de internet del inglés Internet Architeccture Board.

ICMP: Protocolo de control de mensajes de internet del inglés Internet Control Message Protocol.

IP: Protocolo de internet del inglés Internet Protocol.

LCD: Liquid Crystal Display. Pantalla de cristal líquido.

LOS: Line of Sight. Línea de vista.

Magnetoresistor: Componente eléctrico que cambia su resistencia debido a la presencia de un campo magnético.

Microcontrolador: Circuito integrado que tiene incorporado las tres unidades básicas de una computadora (unidad de procesamiento, memoria y unidades de entrada-salida).

PCB: Printed Circuit Board. Diseño de un circuito impreso en una placa de cobre.

PIC: Familia de microcontroladores con arquitectura Harvard fabricada por Microchip Technology.

Protoboard: Es una placa genérica de pruebas para componentes eléctricos y electrónicos.

RARP: Protocolo de resolución de direcciones invertido del inglés Reverse Address Resolution Protocol.

RS-232. Estándar recomendado 232 del inglés Recommended Standard 232.

SMTP: Protocolo simple de transferencia de correos del inglés Simple Mail Transfer Protocol.

SNMP: Protocolo de manejo de red simple del inglés Simple Network Management Protocol.

SOP: Small Outline Package. Tipo de empaquetado utilizado en circuitos integrados. Conocido también como "de superficie".

TCP: Protocolo de control de transmisión del inglés Transmission Control Protocol.

Telnet: Red de telecomunicación del inglés Telecommunication Network.

TFTP: Protocolo trivial de transferencia de archivos del inglés Trivial File Transfer Protocol.

UDP: Protocolo de datagramas de usuario del inglés User Datagram Protocol.

Xbee: Módulos de comunicación de radio frecuencia.