TECNOLÓGICO
DE MONTERREY.

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Ciudad de México

Departamento de Ingeniería Eléctronica y Mecánica

División de Ingeniería y Arquitectura

Proyectos de Ingeniería

"Diseño e implementación de prototipo para un carrito de supermercado con capacidad de seguimiento"

Asesores

Dra. Katya Romo Medrano
Dr. Víctor Manuel de la Cueva

Autores

Arturo Javier Solares Peña [redacted]
Víctor Sánchez Soto [redacted]
Enrique Garavito López [redacted]
José Alberto Rincón Espinosa [redacted]
IEC

Diciembre 2008

Contenido

Capítulo 1
Introducción

1.1 Objetivo

Programación de un robot móvil, para su aplicación como carrito de supermercado con capacidad de seguimiento a un usuario determinado, utilizando como plataforma base el robot Pioneer 2.

El carrito contará con una aplicación inalámbrica para la localización de productos dentro del supermercado.

Dentro de los objetivos específicos se tienen:

- Desarrollar un sistema de visión por computadora que permita la identificación de colores.
- Realizar la programación para lograr que el robot Pioneer 2 logre una navegación autónoma.
- Integrar el sistema de visión al robot Pioneer 2 para lograr un seguimiento por color.

1.2 Justificación

En México viven aproximadamente 8 millones de personas de la tercera edad o con alguna discapacidad como se ve en las Fig. 1.1 y 1.2, lo que les dificulta realizar actividades de la vida diaria. Un ejemplo de estas actividades es el simple hecho de realizar compras en un supermercado.

Los supermercados, hoy en día, no cuentan con la infraestructura necesaria para que personas con capacidades diferentes puedan realizar sus compras sin ayuda de alguien más. Es cierto que muchas tiendas cuentan con sillas de ruedas eléctricas en las que las personas pueden desplazarse, sin

embargo, su número es limitado y no todas las personas pueden usarlas, además de que no son del todo prácticas. En la actualidad se cuenta con sistemas de compras por teléfono o por Internet que dan una solución a este problema, no obstante, no brindan la oportunidad de interactuar con el medio de un centro comercial, permitiéndole realizar actividades como cualquier otra persona.

Además de los discapacitados, existen otros grupos de personas que enfrentan complicaciones al realizar sus compras, como lo son personas de la tercera edad, mujeres embarazadas, etc. Para ellos, estas actividades aparentemente sencillas pueden tener complicaciones especialmente por el peso de los productos. Los supermercados no han puesto real atención a este tipo de clientes, por lo que existe un gran nicho de servicios que pueden ser ofrecidos específicamente para ellos.

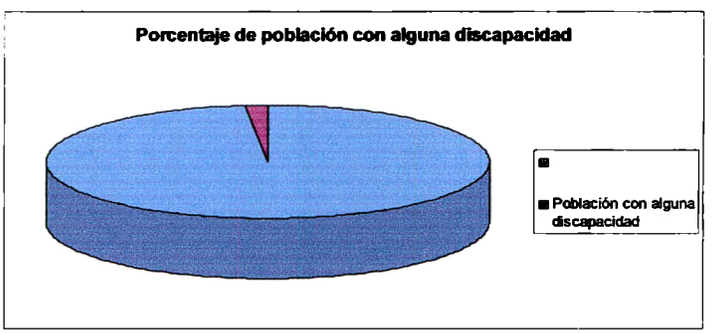


Fig. 1.1. Porcentaje de discapacitados [1]

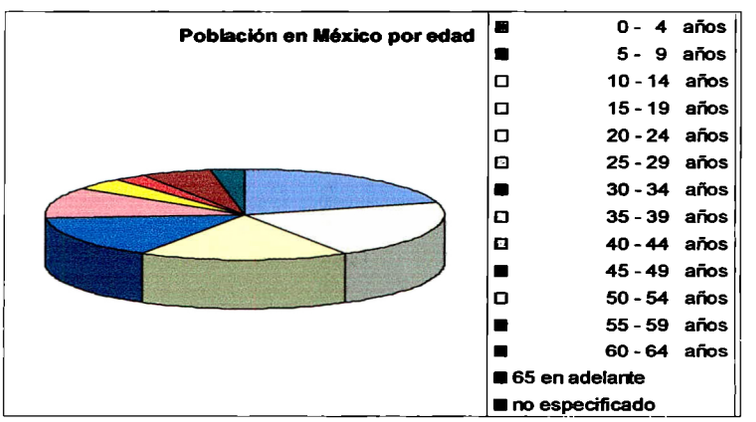


Fig. 1.2. Población en México por edad [1]

Es por esto que se plantea el desarrollo de un prototipo de carrito motorizado montado en el Pioneer 2 para poder ofrecer a estas personas una alternativa que haga más sencillas sus compras en el supermercado. Esto permitirá que puedan llevar a cabo estas actividades de una forma normal y eficiente.

Sin embargo no solo se debe enfocar en este segmento, sino que es el principio en una evolución del concepto de compras en el supermercado, que al día de hoy esta estancado en el simple carrito que lleva existiendo muchos años y que no se ha visto beneficiado por la tecnología.

Lo esperado es llegar al punto donde todas las personas dentro de estos establecimientos puedan usar un carrito de supermercado que cuente con la capacidad de seguirlo mientras realizan sus compras y que se puedan ver beneficiados gracias a su aplicación de localización de productos y enlace de telecomunicaciones. La localización de productos se vuelve muy benéfica desde el punto de vista del usuario al recorre menor distancia para lograrlo.

1.3 Alcances

Este trabajo comprende el diseño, caracterización e implementación de un robot móvil con capacidad de seguimiento para su utilización como carrito de supermercado utilizando como plataforma base el robot Pioneer 2. Para esto se diseñaran algoritmos para distintos tipos de seguimientos, los cuales estarán dirigidos al movimiento y navegación del carro.

Además, se contará con un enlace de telecomunicaciones Wi-Fi y una aplicación para la localización de productos dentro del supermercado.

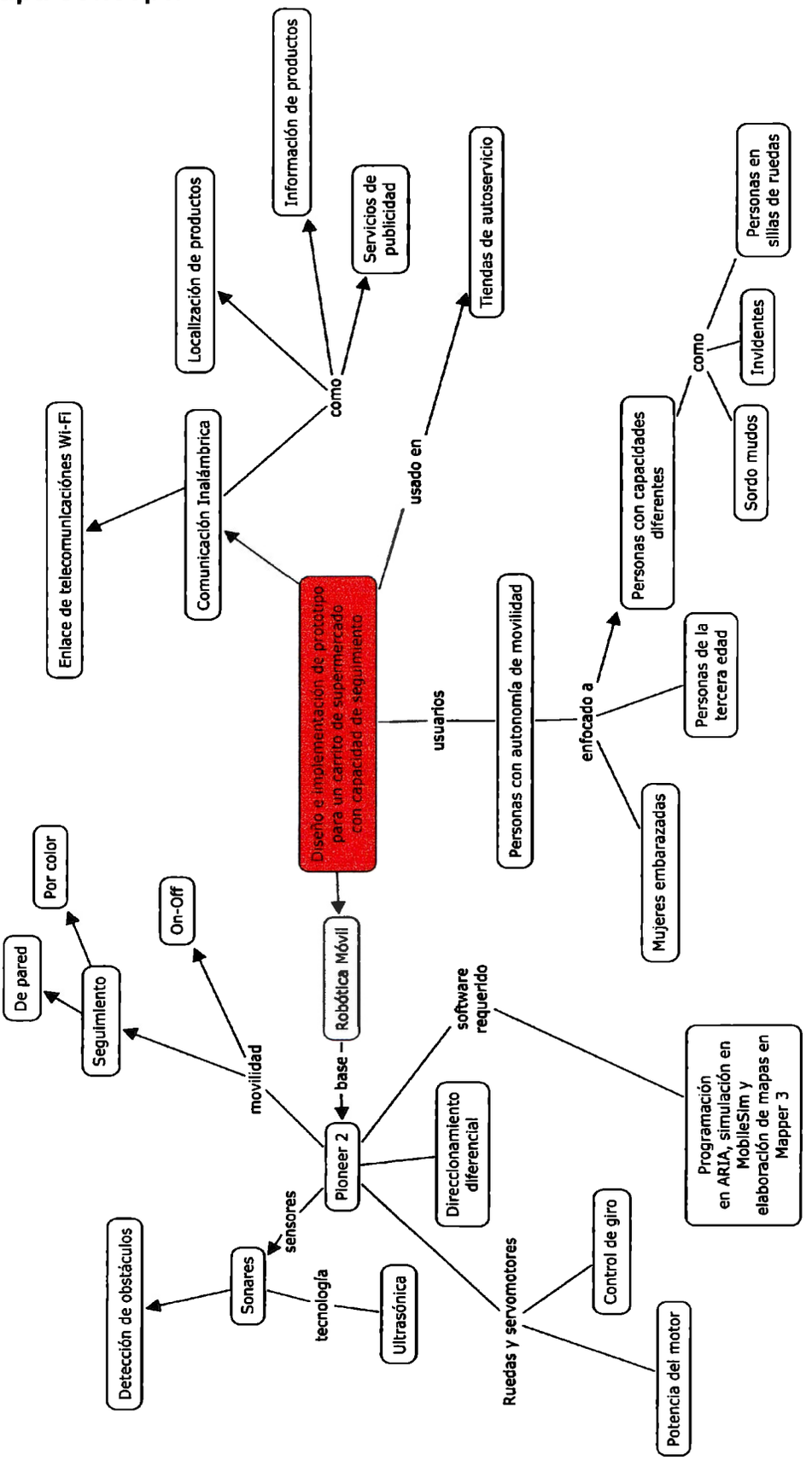
1.4 Limitaciones

El prototipo será diseñado haciendo uso del recurso tecnológico que representa el robot Pioneer 2, el cual será la base del proyecto. Las bases que se establezcan en este desarrollo podrán reutilizarse para llevarse a otro tipo de robots o inclusive el diseño de nuevos robots, sin embargo no es la intención para este proyecto el diseño ni realización de uno propio.

Este dispositivo no pretende de ninguna manera reemplazar la presencia humana en las compras dentro de un supermercado y de ninguna manera representa un medio de transporte para la persona. A pesar de que el carro esta dirigido a personas con capacidades diferentes, únicamente está dirigido a aquellas que puedan manejarse con autonomía como lo son personas en silla de ruedas, invidentes, etc.

Una limitación física que tendrá el prototipo es la de la capacidad de carga, estando sujeta a la carga máxima que puede soportar el Pioneer 2 que es de 23 Kg.

1.5 Mapa conceptual



Capítulo 2
Marco Teórico

2.1 Estado del arte

La robótica es una ciencia que se ha centrado en la construcción y desarrollo de máquinas que imitan el funcionamiento de partes del cuerpo humano. Un robot es el resultado tangible que surge tras la integración de la investigación, el desarrollo e implementación de conocimiento. Es importante mencionar que existen robots de muchos tipos, con tareas variadas tanto de propósitos generales como robots dedicados a la realización de tareas sumamente específicas [3].

Los robots han evolucionado notablemente, desde los primeros construidos hasta los actuales. Estas mejoras se deben a dos factores principalmente, el desarrollo de nuevas tecnologías y a la necesidad de los seres humanos por facilitar actividades. El concepto de robot ha ido evolucionando con el tiempo; en sus principios los robots eran maquinas fabricadas para ayudar en la industria, sin embargo en la actualidad los robots son dispositivos sofisticados que pueden servir inclusive como compañía o juguetes.

Los primeros robots de los que se tienen patentes aparece a mediados de los años cuarenta, estos eran bastante simples dedicados al transporte de maquinaria [3]. Un desarrollo notable se da en la década de los años cincuenta gracias a la aparición de la inteligencia artificial. Las primeras computadoras totalmente eléctricas y las primeras máquinas digitales abrieron un nuevo horizonte de posibilidades para la implementación de robots, ya que son la base de lo que se tiene en la actualidad. Los años cincuentas se ven marcados por la aparición del primer robot programable y la introducción al mercado del primer

robot comercial, hecho que marca el inicio de la industria. Los años sesenta y setenta se ven marcados por la construcción de robots de ayuda en procesos industriales como robots ensambladores y brazos mecánicos, así como por el desarrollo de lenguajes de programación de robots. Para los años ochenta se fabrican robots más sofisticados como el robot de impulsión directa RS-1 de IBM, robot de montaje que utiliza un brazo construido por dispositivos de deslizamiento ortogonales [2]." La década de noventas continúa con los desarrollos anteriores; asimismo comienza el cambio de concepto de robot, ya que la percepción de que el robot es una máquina únicamente para la industria cambia por la de hacer de un robot una maquina para toda situación. El Internet y las tecnologías inalámbricas abren paso a desarrollos aún más complejos y sofisticados que buscan la creación de modelos capaces de expresar emociones como el robot IT o la emulación de mascotas para entretenimiento como el SONY iVO (Fig. 2.1).

Fig. 2.1. SONY iVO

Robots de ambientes bajo techo

Es de suma importancia conocer los robots existentes en el mercado que cuenten con las características y/o objetivos similares a los propuestos en este trabajo. Para ello se hace una recopilación de los principales junto con una pequeña explicación de cada uno de ellos.

El espectro de posibilidades se acota al realizar la investigación únicamente de robots diseñados y desarrollados como guías dentro de ambientes bajo techo.

- Polly: Creado por el MIT, tiene capacidad de navegación por medio de una cámara y de movimiento a velocidades de 1m/s. La cámara le permite reconocer obstáculos [3].
- RHINO: Robot guía de turistas interactivo empleado en un mueso alemán que cuenta con sensores de varios tipos para el reconocimiento del medio como sonares, láser entre otros [4].
- MINERVA: De similares características al anterior, desarrollado por la empresa Thrun, es utilizado en el Museo Nacional de Historia Americana en Washington [2].
- HARINOBU-6: Al igual que los anteriores es un robot guía dentro de una universidad. Se encuentra acoplado en una silla de ruedas motorizada, la cual esta equipada con sistemas de visión, sonares, GPS y GIS portables [7].
- Guido: Es una andadera robótica diseñada especialmente para invidentes o personas con dificultades de movilidad. Cuenta con sonares a bordo que escanean el alrededor en busca de obstáculos. Mantiene comunicación con el usuario mediante advertencias previamente grabadas [5].

Robots enfocados a supermercados

Para el diseño del prototipo propuesto será de suma importancia el tener conocimiento y documentación de los robots existentes en el mercado que están dirigidos al sector de los supermercados. En la actualidad existen algunos robots que cumplen con propósitos similares al propuesto en este proyecto.

- B.O.S.S: Battery Operated Smart Servant por sus siglas en inglés, es un prototipo de carro de supermercado (Fig. 2.2) cuya principal característica su capacidad de seguimiento, el cual se logra gracias a sensores que se adaptan a las necesidades del comprador. El vehículo es capaz de reducir su velocidad si es necesaria la introducción de objetos dentro del carro, además de su capacidad de evasión de obstáculos lo que evita

colisiones con objetos dentro del supermercado. Los sensores incluidos son de color. Estos permiten realizar seguimiento a través del tono de tela. Asimismo cuenta con un sistema de control que le permite acelerar y desacelerar de acuerdo a la velocidad y la distancia del objeto a seguir. [3]

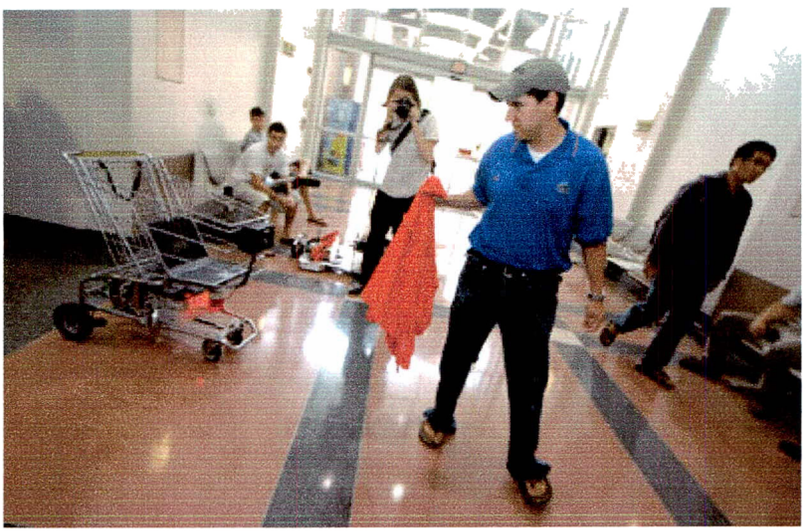


Fig. 2.2. Carro de supermercado B.O.S.S durante la realización de pruebas de seguimiento de color

- Navis Wagon: Es un carro de supermercado equipado con lector de RFID y pantalla de LCD (Fig. 2.3). Estos componentes facilitan las compras dentro del supermercado, ya que se despliegan en pantalla los datos de los objetos escaneados por el lector de RFID y que se encuentran dentro del carrito. Además de eso se despliegan precios, información de productos y promociones en pantalla. [6]

Fig. 2.3. Navis Wagon

- RoboCart: Es un prototipo que tiene como finalidad la ayuda a personas con limitaciones visuales. Se encuentra montado sobre la plataforma del Pioneer 2, pero se le adicionaron componentes como un manubrio, una laptop conectada a un microcontrolador, un láser, un lector de RFID y una antena. El propósito del RFID es el de satisfacer necesidades de navegación dentro del supermercado. Mediante la implementación de "tags" de RFID en el supermercado, el robot es capaz de calcular distancias y tiempos para poder moverse satisfactoriamente dentro del mismo. Este robot no es capaz de identificar un producto en específico, sino posiciona al usuario en las coordenadas donde se encuentra el tipo de producto deseado. El robot cuenta con una serie de algoritmos que hacen que pueda tomar decisiones correctas con base a necesidades específicas, y a partir de esas necesidades ejecutar un plan. El diseño del mismo se puede observar en la Fig. 2.4.

Fig. 2.4. RoboCart durante una sesión de pruebas en un supermercado

- SHOPPER: Este es un robot simulado que utiliza regularidades funcionales como estructurales de supermercados (distribución típica de tiendas) para optimizar y simplificar rutinas de sensado para la realización de compras eficientes [7]. Si bien por el momento es un robot virtual se tiene pensado llevarlo a un prototipo, sin embargo se ha desistido por el momento para evitar problemas mecánicos y prácticos. Es importante que a pesar de que es un ambiente virtual el que se maneja, se le introdujo realismo para que no funcionara como un robot trabajando en un ambiente ideal. Se creo una tienda virtual con base en 75 000 imágenes las cuales son las capaces de identificar las diferentes situaciones. El supermercado virtual cuenta con pasillos y estantes, cuya distribución y almacenaje esta basado en los supermercados reales (acomodo, ubicaciones, etc.). La característica principal del SHOPPER es que es un robot capaz de identificar un producto específico dentro del supermercado. La limitante que tiene para la identificación de productos que es que únicamente es capaz de localizar productos en cajas rectangulares y no en forma de tubos. [8]

Es importante resaltar que ninguno de los robots anteriormente mencionados y que se encuentran en el mercado cuenta con las funciones y características propuestas en este documento, por lo nuestro diseño entrará en el mismo nicho como un prototipo único.

2.2 Marco Teórico

En la actualidad millones de personas simplifican sus tareas por medio de máquinas capaces de realizar trabajos repetitivos o monótonos, en condiciones peligrosas, o que simplemente requerirían de mucho tiempo y esfuerzo para las capacidades humanas.

Los robots son máquinas en las que se integran componentes mecánicos, eléctricos, electrónicos y de comunicaciones, y dotados de un sistema informático para su control en tiempo real, percepción del entorno y programación [9].

Robótica Móvil

En ocasiones la robótica, principalmente en la industria, es relacionada con brazos manipuladores que se encuentran fijos; sin embargo, el mundo de la robótica móvil se ha desarrollado ampliamente por la necesidad de extender las aplicaciones y usos de dichas máquinas a cualquier lugar dentro de un entorno definido por el usuario o el programador.

Las formas más comunes de estos robots son los vehículos autónomos; un robot autónomo es aquel capaz moverse por si mismo de manera inteligente en entornos con situaciones cambiantes y desconocidos sin la necesidad de supervisión o monitoreo por parte de un operador. Dichos movimientos se hacen

a través del uso de diversos dispositivos de entre los que destacan los motores, sensores, cámaras, actuadores, etc.

Los motores son aquellos dispositivos que convierten la energía eléctrica en mecánica por medio de fenómenos electromagnéticos. Son fundamentales en el campo de la robótica móvil ya que son los encargados de la acción del movimiento, sin ellos, el robot sería incapaz de cambiar su posición por sí mismo dentro de un área determinada.

Los robots que funcionan por medio de motores que accionan un sistema de ruedas se pueden clasificar en varios grupos que se muestran a continuación:

- Ackerman: Esta es una forma sencilla para la maniobrabilidad de un robot, en este sistema están basados los automóviles y de forma más rudimentaria los carritos de supermercado, entre otros. Su funcionamiento está dado por el giro de las ruedas delanteras. Como se puede ver en la Fig. 2.5, la rueda interior (la rueda que se encuentra del lado al que se desea girar) tiene que virar un poco más que la rueda exterior, es decir, el ángulo de giro de la rueda interior tiene que ser mayor para evitar que alguna de las llantas se deslice o derrape.

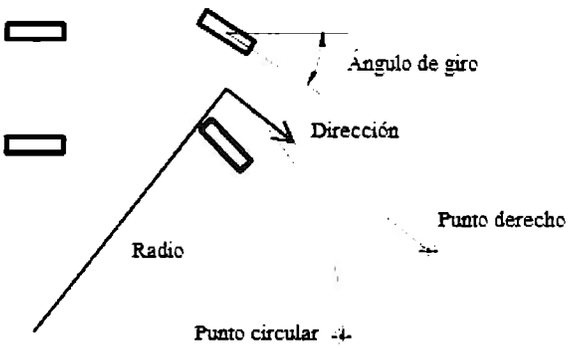


Fig. 2.5 Sistema de maniobrabilidad Ackerman

- Triciclo clásico: este sistema, como su nombre lo indica, está basado en tres ruedas, este sistema se muestra en la Fig. 2.6. La rueda delantera sirve para dar tracción además de ser el que determina la dirección al robot, mientras que las traseras sólo pueden moverse hacia delante y hacia atrás brindándole la tracción al vehículo. Este sistema presenta mayor maniobrabilidad sin embargo también es muy inestable.

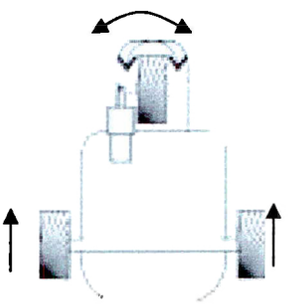


Fig. 2.6 Sistema de maniobrabilidad
de triciclo clásico

- Skid Steer: Este sistema está formado por varias ruedas moviéndose de manera simultánea; el movimiento y la velocidad son el resultado de la combinación de las velocidades de las ruedas de cada lado Fig. 2.7. Este sistema de movilidad es muy popular entre los vehículos utilizados para la construcción como excavadoras, barredoras, etc.

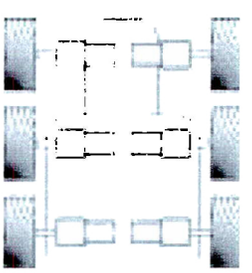


Fig. 2.7. Sistema de maniobrabilidad
Skid Steer

- Pistas de deslizamiento: Este sistema de movilidad consta de una banda que rodea una serie de ruedas (en configuración similar a Skid Steer), formando una oruga parecida a la encargada del movimiento del movimiento en tanques militares (Fig. 2.8). Dada la semejanza del efecto que genera este sistema con uno de llantas grandes, es muy útil para avanzar en terrenos difíciles o irregulares.

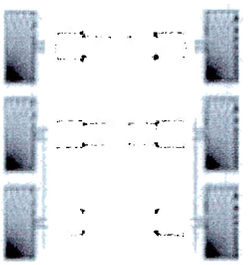


Fig. 2.8. Sistema de maniobrabilidad
por pista de deslizamiento

- Síncronas: Es uno de los más complicados de los que funcionan por medio de ruedas ya que cada rueda debe de actuar de manera simultánea para realizar algún movimiento ya sea de simple tracción o de direccionamiento (Fig. 2.9). El sistema síncrono es muy utilizado para robots que están diseñados para funcionar en áreas de terrenos parejos.

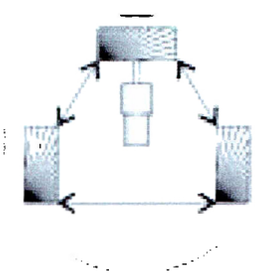


Fig. 2.9. Sistema de maniobrabilidad síncrona

- Direccionamiento diferencial: Esta es una de las maneras más sencillas pero a la vez más recurridas en el diseño de robots móviles. Necesita por lo menos de dos ruedas las cuales se encuentran conectadas a dos motores independientes, esto con el fin de lograr que cada rueda pueda tener una velocidad o dirección diferente lo que le brinda la capacidad de direccionamiento. A pesar de que cada rueda puede tener diferente velocidad el torque aplicado a cada una de ellas es igual. Robots comerciales como el Pioneer 1, Pioneer 2 DX, etc. basan su movimiento de tracción y dirección en un sistema diferencial (Fig. 2.10).

Fig. 2.10. Sistema de direccionamiento diferencial

Este sistema será el encargado de la movilidad de nuestro prototipo de supermercado, ya que es el sistema en el que se basa el robot Pioneer 2.

Existen otros sistemas como el omnidireccional, en el que cada rueda tiene la capacidad de direccionamiento y tracción de manera independiente; por otro lado existen robots que están basados no en ruedas sino en patas (Fig. 2.11) los cuales pueden moverse de manera similar a la de un insecto o en algunos casos hasta bípedos que tratan de imitar el movimiento de las piernas en el ser humano (Fig. 2.12). Algunos robots utilizan turbinas para desplazarse en medios acuáticos o aéreos. De la misma forma se han desarrollado robots capaces de balancearse sobre una esfera para fines de movilidad y direccionamiento.

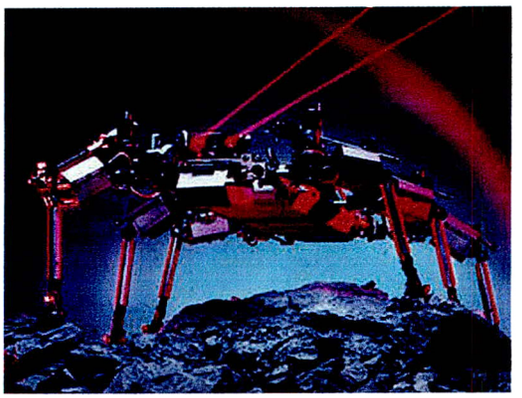


Fig. 2.11. Attila (MIT)

Fig. 2.12. ASIMO (Honda)

Una de las formas más comunes de manejar estos sistemas es por medio de servomecanismos. Los servomecanismos o servos son dispositivos automáticos que usan retroalimentación de error de sensado para el correcto desempeño de un mecanismo.

No todo servomecanismo es un servomotor. Los servos comunes proveen control de posición, normalmente son eléctricos o electrónicos y utilizan un motor eléctrico para crear fuerza mecánica. Operan bajo el principio de retroalimentación negativa, donde la entrada de control es comparada con la posición actual del sistema mecánico. Cualquier diferencia entre los valores actuales y los no deseados, señal de error, es amplificada y usada para llevar al sistema en la dirección necesaria para reducir o eliminar el ruido.

Un servomotor es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación y mantenerse estable en dicha posición.

Sensores

Otro de los aspectos importantes del movimiento en lo que a robótica móvil se refiere es la de los sensores. Los sensores son dispositivos capaces de detectar o medir propiedades físicas en el medio, tales como temperatura, luminiscencia, resistencia al tacto, peso, tamaño, etc.[10]; por medio de los sensores los robots son capaces de obtener información del mundo externo [11]. Los sensores, a partir de la energía del medio donde se mide, dan una señal de salida transductible que es función de la variable medida [12]. Los robots móviles comúnmente tienen sensores de diferentes tipos según sea la aplicación que se les quiera dar.

Existen diversas clasificaciones de para los sensores, se pueden catalogar por el tipo de conversión (ya sea directa: energía→señal; o indirecta energía→modulación→señal), si son absolutos o relativos (absolutos: dan un valor determinado a diferentes cantidades de la energía; relativos: muestran la magnitud del cambio de cierta energía), según el campo de medida (pueden entregar una cantidad relacionada con la energía recibida o pueden ser ON-OFF).

Los sensores táctiles, como su nombre lo indica, detectan cuando se encuentran en interacción física directa con algún otro material, es decir, envían una señal cuando se encuentran en contacto con algo. Por lo general están formados por microswitches que son presionados por el objeto que los toca cerrando o abriendo un circuito de detección.

Por otra parte los sensores ópticos toman gran importancia en lo que a detección y evasión de obstáculos se refiere. Los sensores ópticos están formados comúnmente por fotorresistencias, fotodiodos, y fototransistores.

Los sensores infrarrojos son una de las tecnologías ópticas más populares en la construcción de robots móviles dada su simplicidad. Su funcionamiento se basa en la emisión de una luz infrarroja (longitudes de onda de aproximadamente 880nm) que es reflejada por algún material que se pueda encontrar dentro de un rango de visión. Es posible que se presenten problemas como el de la presencia de señales infrarrojas ajenas como las provenientes de los rayos solares al sensor que puedan afectar en su funcionamiento, no obstante esto se puede solucionar al modular la luz infrarroja con una señal de baja frecuencia como puede ser una de entre 100Hz y 500Hz. La mayor desventaja que se le puede encontrar al infrarrojo es la forma en la que cada color refleja de diferente manera la luz, ya que por ejemplo los objetos obscuros tienden a absorber la luz provocando que las lecturas pueden ser confusas o imprecisas. Para su funcionamiento es necesario un diodo infrarrojo como emisor y un fototransistor como receptor.

Los robots seguidores de línea basan su funcionamiento en los sensores infrarrojos, ya que la línea debe contrastar con el color de la superficie sobre la que debe moverse el robot; dicho contraste hace que el sensor envíe la señal al robot para que ejecute una acción.

Los sonares son sensores muy populares en el ámbito de la robótica. Estos operan por medio de señales acústicas de poca duración pero de alta frecuencia que viajan del emisor hasta encontrar algún tipo de obstáculo, este obstáculo refleja la señal la cual regresa hacia la posición del emisor. Al utilizar sonares es recomendable usar arreglos de varios ya que con esta tecnología es difícil distinguir la posición del objeto en el que rebotó la señal sonora. Una ventaja muy importante de los sonares es que se puede obtener un dato de la distancia a la que se encuentra el obstáculo. Para su funcionamiento es necesario un transductor para la emisión del ultrasonido y otro transductor para la detección del eco. La distancia se determina por el tiempo que tarda la señal acústica en llegar al receptor.

Es importante tomar en cuenta varios factores y parámetros en el uso de sonares:

- Ángulo del cono: Este se refiere a la falta de dato de la posición angular del obstáculo, es decir, el dato de distancia que se obtiene por el sonar es de una distancia radial pero se puede encontrar a cualquier ángulo.
- Zona muerta: En caso de usar un solo transductor para la emisión y la recepción de datos se debe de tener una distancia mínima entre el objeto detectado y el sensor ya que en caso de no existir esta, el sensor no será capaz de detectar dicho obstáculo. Esto se debe a que el transductor debe de tener un tiempo de relajación para poder terminar de transmitir y empezar a recibir.
- Distancia máxima: Es el rango máximo al que se puede encontrar un objeto para poder ser detectado, en el caso de sonares en robots comerciales como el Pioneer 2 la distancia máxima es de 5 metros.

El proceso de detección en el receptor de los sonares se puede hacer por dos métodos:

- Detección de umbral: Este se da por una diferencia de voltaje en el detector que después de ser amplificada, determina si existe presencia de algún objeto al rebasar algún parámetro de tensión preestablecido. Su desventaja es la gran susceptibilidad al ruido.
- Detección de tonos: Esta forma combina el método anterior con la detección de la frecuencia de la señal transmitida. Al ser recibida la señal del "eco" detectada como mayor que el umbral, se coteja con la frecuencia de la señal sónica emitida y si son compatibles entonces se interpreta como un obstáculo a cierta distancia.

El problema que se puede presentar con este tipo de sensores es la posibilidad de recibir señales que han rebotado varias veces en el medio y llegan

de nuevo al sensor pareciendo como una señal proveniente de un nuevo obstáculo. Otro problema se presenta con la existencia de varios transmisores o distintas fuentes de ultrasonido.

Existen otros sensores utilizados en la robótica móvil como los capacitivos e inductivos, que detectan posibles obstáculos o superficies por medio de cambios en la capacitancia o inductancia, según sea el caso; sensores de efecto Hall que relaciona un campo eléctrico al entrar en un campo magnético; además de otros como los sensores de esfuerzo, de visión, etc.

Posición y orientación

Dentro de la robótica resulta imprescindible el ser capaces de representar posiciones y orientaciones en el espacio. En la robótica móvil se consideran inclusive modelos en tres dimensiones. Estos modelos se usan en vehículos autónomos, pero también para la navegación en terrenos no planos de robots con ruedas o patas [9].

La posición de un punto en el espacio se puede representar en un eje coordenado XYZ, pero también se puede hacer mediante coordenadas cilíndricas [13]. En este caso las coordenadas son la distancia ρ, el ángulo θ y la distancia *z* entre el punto y su proyección en el plano X – Y (Fig. 2.13).

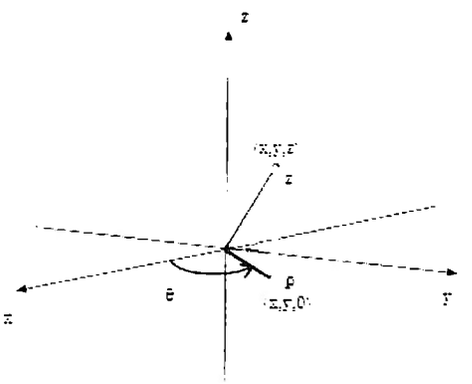


Fig 2.13. Proyección de coordenadas rectangulares

La orientación de un cuerpo se describe mediante un sistema de coordenadas {B}. Si se desea representar la orientación con respecto a un sistema de referencia {A}, solo se debe expresar {B} con respecto a {A} [13].

Principios de navegación

Para cualquier dispositivo móvil, la habilidad de navegar es una de las características más importantes que debe poseer. Ser operacional para evitar cualquier tipo de obstáculos y de este modo evitar colisiones que puedan dañar su funcionamiento [13].

La navegación puede ser definida como la combinación de tres aspectos fundamentales [13]:

- Auto localización
- Planeación de camino
- Creación e interpretación de mapas

Los mapas generados no necesariamente son una copia del mundo real, sino que en realidad son parámetros que se introducen al dispositivo para obtener la respuesta deseada en determinado ambiente.

La localización denota que tan competente puede ser el dispositivo para establecer su posición con respecto a un punto de referencia.

Cuadro de navegación con referencia

El cuadro de navegación es un componente crucial sobre cualquier robot que tenga un sistema de navegación, dado que esto define los tres aspectos fundamentales para la navegación que ya han sido mencionados. Este cuadro de navegación es alineado con respecto a una referencia, sobre el cual se mide el desplazamiento total del dispositivo.

En el más simple de los casos, este cuadro de referencias es un sistema de coordenadas cartesianas [13].

Para procesos internos como el sistema de navegación cartesiano para la localización, planeación de camino y mapeo de territorio, las posiciones son guardadas como coordenadas con respecto a la referencia [11]. Asumiendo que toda la navegación tuvo un punto de partida conocido y que la velocidad y la dirección de desplazamiento es conocida, entonces el sistema integra estos datos para realizar una operación de desplazamiento entre tiempo.

En este sistema el principal problema es que el cuadro no esta completamente definido, ya que existen algunos otros parámetros que afectan en el desplazamiento del dispositivo como lo son: frenado, deslizamiento de las llantas. Estos movimientos son realizados en la vida real, pero dentro del cuadro de navegación estos problemas son imperceptibles.

Navegación basada en señalización

Una forma de solucionar los problemas que existen es detectar mediante sensores y en tiempo real los obstáculos que se encuentren en el mundo real y no predefinirlos como se hacia en el cuadro de navegación. La navegación con respecto a una señal se puede llamar "pilotear" [11].

Señalización perceptual para la navegación

Para obtener un sistema de navegación por señalización se debe cumplir con ciertos requerimientos:

- Visibilidad desde varias posiciones
- Ser reconocible bajo diferentes condiciones de luminosidad y ángulos de vista.

- Tener conocimiento en todo momento del estado estacionario o desplazamiento del dispositivo.

Los primeros dos puntos están a la espera de la representación interna de la señalización con respecto al sistema de navegación. Guardando las percepciones de los sensores del dispositivo sobre un objeto no es lo más eficiente dado que cada vez que sea censado el objeto se hará desde diferente ángulo y punto de vista y la señal de percepción cambiará con respecto a la anterior [12]. El punto uno hace referencia al problema del reconocimiento de la señal, dado que no se asegura que el sistema de navegación pueda reconocer el mismo objeto desde diferentes puntos de vista y desde diferentes posiciones. Sin embargo, dadas las características de los sonares es especial de la reflexión especular, la cual esta determinada según el ángulo de incidencia de la señal, este método de navegación no siempre funciona.

Teleoperación

La teleoperación es otro aspecto muy importante de la robótica móvil, le brinda la posibilidad de manejo del robot de manera remota. Este aspecto resulta muy importante en el trabajo en regiones peligrosas como minas, áreas submarinas, áreas de muy alta temperatura o presión, etc. Por otro lado, la telerrobótica permite mayor autonomía del robot operado. Esta se basa en poner una inteligencia programada en alguna locación aparte al robot, es decir, no es necesario tenerla montada en alguna computadora a bordo del robot.

Un problema que se presenta en la teleoperación y en la telerrobótica es el posible retardo en la comunicación, depende del medio transmisión. Un retardo se toma como significativo si se encuentra en el orden de los 0.4 segundos [13]; en estas situaciones la retroalimentación se dificulta por lo que tiende a ser más difícil el control del robot. Sin embargo en control por medio de

Internet estos retardos son pequeños por lo que nos e consideran significativos y no representan un problema mayor en el control del robot.

Vision por computadora

El término "visión por computadora", puede considerarse como el conjunto de todas aquellas técnicas y modelos que nos permitan el procesamiento, análisis y explicación de cualquier tipo de información espacial obtenida a través de imágenes digitales.

Dado que la información visual es una de las principales fuentes de datos del mundo real, resulta útil que una computadora cuente con el sentido de la vista (a partir de imágenes tomadas con cámaras digitales o analógicas), que junto con otros mecanismos como el aprendizaje hagan de ésta una herramienta capaz de detectar y ubicar objetos en el mundo real.

Algunos conceptos importantes que se deben tomar en cuenta [20]:

- Imagen binaria: Contiene sólo información de brillo, no de color y se conocen como imágenes de un bit por pixel. Así sólo se tiene el valor uno para representar al(los) objeto(s) y un valor cero para el fondo de la imagen. Este tipo de imágenes se usan en aquellas aplicaciones donde la información del contorno es importante, en el caso de que haya objetos con intensidad o brillo contrastantes.
- Imagen en tonos de gris o monocromática: Al igual que las binarias sólo contienen información de brillo, sin embargo el número de bits usados para cada pixel varía dependiendo de la cantidad de tonos de gris que se desea representar.
- Imágenes a color: Se describen por 3 imágenes monocromáticas, cada una sintonizada en una banda (rojo, verde y azul). Este tipo de imágenes se utilizan en aquellas aplicaciones donde la información de color es indispensable.

Sistemas de visión

Los sistemas de visión en robótica se implementan para además de tener la posibilidad de reconocer objetos, también tener una alternativa para control de navegación. Con este tipo de sistemas, el robot puede tener la capacidad evadir obstáculos y para el caso de este proyecto, hace posible la identificación de colores.

Para lograr esa visión se requiere de hardware que permita la adquisición de imágenes, su digitalización y el procesamiento. Una cámara conectada al robot es la solución a este problema, ésta, envía la imagen a un software de computadora que es capaz de procesarla para la identificación del color. Con esta información se puede identificar el lugar donde se encuentra el objetivo, y se realizan cálculos del ángulo existente entre el robot y el objetivo. Junto con la información de los sonares, el software calcula el lugar exacto donde se encuentra el objetivo.

Reconocimiento de color

Para entender como se lleva a cabo la identificación de color, es necesario saber como es que la computadora "ve" la imagen. Dado que las computadoras solo trabajan con números, cuando la computadora procesa una imagen, lo que ve, es una serie de números. Existen diferentes formas de para codificar las imágenes usando diferentes espacios de color.

Espacios de color

Un espacio de color, describe el rango de colores que una cámara puede ver. Es una serie de códigos para cada color. Cada pixel en la imagen tiene un color que se describe en el espacio de color, es por esto el espacio de color, se

puede usar para el etiquetado de pixeles. Existen diferentes espacios, algunos de los más conocidos, son RGB, HSV y LAB.

- RGB

En este espacio, cada color se describe como una combinación de tres principales colores: Rojo, Verde y Azul. Este espacio de color, se visualiza en una matriz en 3 dimensiones, donde los valores para cada color varían de 0 a 1. Cada color se codifica con un valor para cada color, y en una imagen digital, se describen 3 matrices con los valores para cada pixel de cada color [21] como se muetra en la Fig. 2.14.

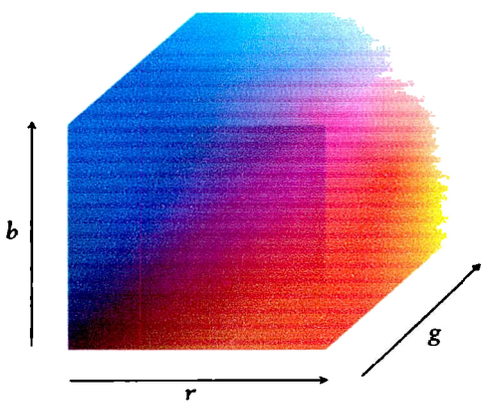


Fig. 2.14 Espacio RGB

- Espacio HSV

El espacio HSV es un espacio cilíndrico (Fig. 2.15a y Fig. 2.15b) que puede ser modelado como un cono. La posición vertical define el brillo (V), la posición angular el tono (H) y la posición radial representa la saturación (S). Para cada valor de V, puede ser dibujado un círculo como una sección transversal del cono. La saturación varía de 0 a 1, y especifica la posición relativa desde el eje vertical a la orilla del círculo. Cero saturación indica que no existe un tono de color, solamente una escala de grises. El tono es lo que

normalmente es considerado el tono de color, y varía de 0 a 1. El brillo también tiene un intervalo de 0 a 1, donde V=0 es un negro [21].

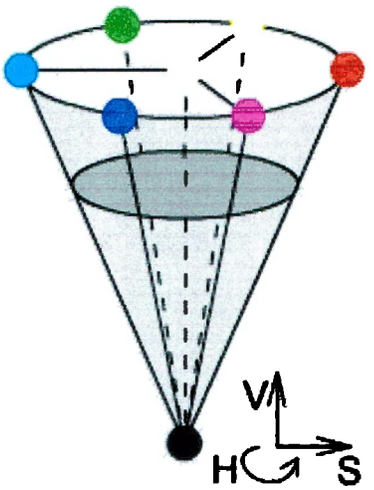


Fig. 2.15a Espacio HSV

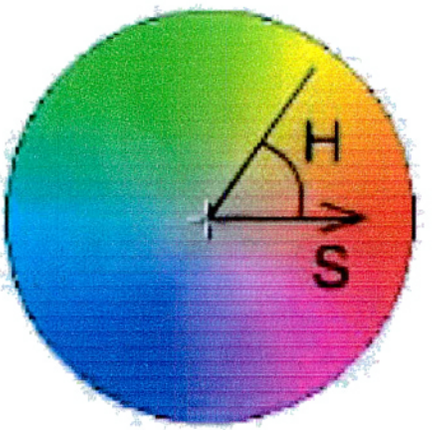


Fig. 2.15b Sección transversal del modelo HSV

- Espacio LAB

El espacio LAB (Fig. 2.16), se define con tres parámetros: L, define la luminosidad, A denota el valor de color rojo/verde y B, el valor de amarillo/azul. Este espacio de color no es tan fácil de visualizar comparándolo con RGB o HSV, pues el valor de L tiene un intervalo de 0 a 100 y para cada valor de L, diferentes imágenes se pueden representar. El valor de A se describe de forma horizontal y B en el eje vertical.

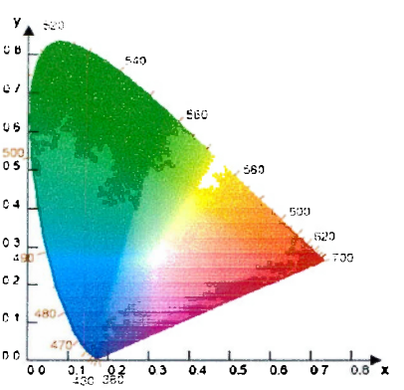


Fig. 2.16 Espacio LAB [22]

El espacio de color escogido para el desarrollo del proyecto, fue el espacio HSV, y eso fue debido a que tiene una ventaja sobre los otros dos espacios y es que un color se puede definir en únicamente dos variables (H y S), el valor de V no es importante para definir un color, y esto se debe a que mientras el valor de V no sea igual a cero, caso en el que se tendría un color negro, cualquier color se puede definir en solo dos dimensiones. De esta manera, es posible definir un color que será el color al que se realizará el seguimiento, es decir el objetivo.

Etiquetado de pixeles

Los colores en el espacio HSV se pueden visualizar dentro de un círculo, y cuando se hace la definición del color a seguir, se define solo una parte de ese círculo, el valor se V no es necesario tomarlo en cuenta, por lo que no es necesario visualizar el espacio HSV como un cono, solo como un círculo como se ve en la Fig. 2.17.

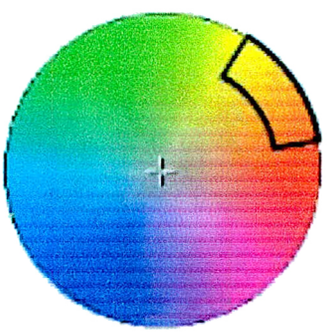


Fig. 2.17 Parte del espacio HSV definido

Con una matriz en tres dimensiones de tamaño 256x256x256, con los rangos de H, S y V definidos es posible crear una imagen binaria donde los pixeles que corresponden al color definido, se definen con un 1, y los pixeles con valores diferentes al valor de color definido, se etiquetan con un 0. El proceso en el que valores de 1 y 0 es dado a los pixeles que corresponden al color que se definió, es conocido como etiquetado de pixeles.

Funciones de *opening* y *closening*

En las imágenes binarias que se tienen como resultado del etiquetado de pixeles, no todos los pixeles correspondientes al color que se definió son marcados con 1, y esto es debido a que las condiciones de luz pueden hacer que un color se vea diferente con diferente iluminación, es por eso que dicha imagen binaria, no siempre muestra una imagen homogenea del objetivo detectado, es decir, presenta pixeles dispersos

Para solucionar estos problemas, se pueden aplicar diferentes operadores morfológicos para: separar zonas, delimitar áreas de determinados tamaños, erosionar o dilatar regiones, etc.

Dos de las operaciones morfológicas más usadas en el análisis de imágenes son la dilatación y erosión. Las operaciones de dilatación y erosión añaden y eliminan píxeles, respectivamente, de una imagen binaria. Para la dilatación, si algún píxel del vecindario de entrada está a "1" la salida se pone a "1" en caso contrario pasa a cero. Para la erosión, si todos los píxeles del vecindario de entrada están a "1" la salida es "1", en caso contrario será "0".

El vecindario se define mediante una matriz de "1s" y "0s" que se denomina elemento estructurante (SE). El píxel central representa el píxel de interés mientras que los elementos de la matriz a "1" definen la vecindad.

La operación de *opening*, consiste en una erosión seguida de una dilatación, se llama así porque lo que hace es separar (abrir) los pixeles de acuerdo a las vecindades.

La operación de *closening*, es una dilatación seguida de una erosión y recibe ese nombre porque lo que hace es cerrar huecos de pixeles, también de acuerdo a las vecindades.

. Cuando se quiere lograr una detección correcta del objetivo y al mismo tiempo tener un seguimiento confiable, se implementan las funciones de *opening* y *closening* para garantizar una imagen con pixeles homogéneos que permitan una detección confiable, y al mismo tiempo, un correcto seguimiento [23].

Comunicacion Inalámbrica

Wi-Fi

La función principal de este tipo de redes es la proporcionar conectividad y acceso a las tradicionales redes cableadas (Ethernet, Token Ring, etc.), como si se tratara de una extensión de éstas últimas, pero con la flexibilidad y movilidad que ofrecen las comunicaciones inalámbricas. El momento decisivo para la consolidación de estos sistemas fue la conclusión del estándar IEEE 802.11 en junio de 1997 [24].

En este estándar se encuentran las especificaciones tanto físicas como a nivel MAC[1] que hay que tener en cuenta a la hora de implementar una red de área local inalámbrica. Otro de los estándares definidos y que trabajan en este mismo sentido es el ETSI HIPERLAN[2] [25].

La norma 802.11 ha sufrido diferentes extensiones para obtener modificaciones y mejoras, es por esto que hoy en día tenemos las siguientes especificaciones:

- 802.11 Especificación para 1-2 Mbps en la banda de los 2.4 GHz, usando

[1] MAC. Control de acceso al medio es el conjunto de mecanismos y protocolos por los que varios "interlocutores" (dispositivos en una red, como ordenadores, teléfonos móviles, etcétera) se ponen de acuerdo para compartir un medio de transmisión común (por lo general, un cable eléctrico u óptico, o en comunicaciones inalámbricas el rango de frecuencias asignado a su sistema).

[2] ETSI HIPERLAN. High Performance Radio LAN version 1 es un estándar del ETSI (European Telecomunications Standards Institute). Es similar a 802.11a (5 GHz) y es diferente de 802.11b/g (2,4 GHz).

salto de frecuencias (FHSS) o secuencia directa (DSSS) [24].

- 802.11b Extensión de 802.11 para proporcionar 11Mbps usando DSSS.
- Wi-Fi (Wireless Fidelity) Promulgado por el WECA para certificar productos.
- 802.11b capaces de dar servicio no importando el fabricante.
- 802.11a Extensión de 802.11 para proporcionar 54Mbps usando OFDM.
- 802.11g Extensión de 802.11 para proporcionar 20-54Mbps usando DSSS.
- OFDM. Es compatible hacia atrás con 802.11b. Tiene mayor alcance y menor consumo de potencia que 802.11^a [24].

Red de Área Local Inalámbrica

Una red de área local inalámbrica puede definirse como a una red de alcance local que tiene como medio de transmisión el aire. Por red de área local entendemos una red que cubre un entorno geográfico limitado, con una velocidad de transferencia de datos relativamente alta (mayor o igual a 1 Mbps tal y como especifica el IEEE), con baja tasa de errores y administrada de forma privada. Utiliza ondas electromagnéticas como medio de transmisión de la información que viaja a través del canal inalámbrico enlazando los diferentes equipos o terminales móviles asociados a la red. Estos enlaces se implementan básicamente a través de tecnologías de microondas e infrarrojos.

En las redes tradicionales cableadas esta información viaja a través de cables coaxiales, pares trenzados o fibra óptica. Una red de área local inalámbrica, también llamada wireless LAN (WLAN), es un sistema flexible de comunicaciones que puede implementarse como una extensión o directamente como una alternativa a una red cableada. Este tipo de redes utiliza tecnología de radiofrecuencia minimizando así la necesidad de conexiones cableadas. Este hecho proporciona al usuario una gran movilidad sin perder conectividad [25] [26].

Configuraciones WLAN

El grado de complejidad de una red de área local inalámbrica, cuyas características se muestran en el anexo I, es variable, dependiendo de las necesidades a cubrir y en función de los requerimientos del sistema que queramos implementar podemos utilizar diversas configuraciones de red.

Peer to peer o redes *ad-hoc*

La configuración más básica es la llamada *de igual a igual* o *ad-hoc* y consiste en una red de dos terminales móviles equipados con la correspondiente tarjeta de red para establecer las comunicaciones inalámbricas. En la Fig. 2.18 se muestra un ejemplo.

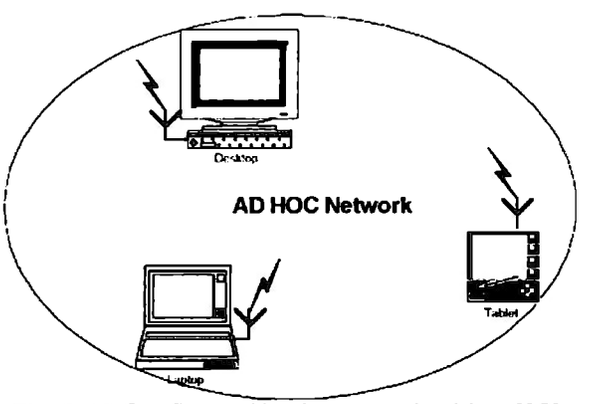


Fig. 2.18 Configuración de una red *ad-hoc* [27].

Para que la comunicación entre estas dos estaciones sea posible hace falta que se vean mutuamente de manera directa, es decir, que cada una de ellas esté en el rango de cobertura radioeléctrica de la otra. Las redes de tipo *ad-hoc* son muy sencillas de implementar y no requieren ningún tipo de gestión administrativa.

B. Modo Infraestructura

Para aumentar el alcance de una red del tipo anterior hace falta la instalación de un *punto de acceso*. Con este nuevo elemento doblamos el alcance de la red inalámbrica (ahora la distancia máxima permitida no es entre estaciones, sino entre cada estación y el punto de acceso). En la Fig. 2.19 se muestra un ejemplo.

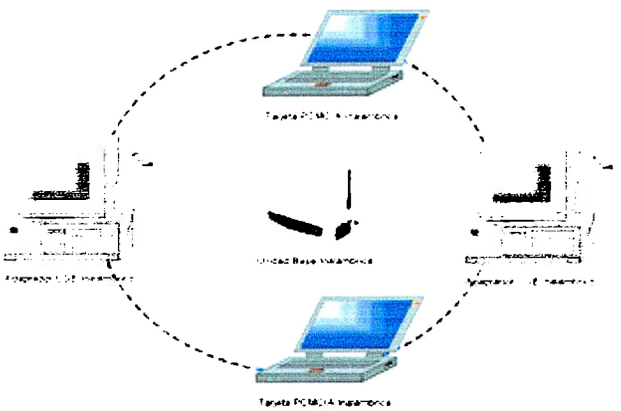


Fig 2.19 Configuración de una red modo infraestructura [28].

Además, los *puntos de acceso* se pueden conectar a otras redes, y en particular a una red fija, con lo cual un usuario puede tener acceso desde su terminal móvil a otros recursos. Para dar cobertura en una zona determinada habrá que instalar varios puntos de acceso de tal manera que podamos cubrir la superficie necesaria con las celdas de cobertura que proporciona cada punto de acceso y ligeramente cerradas para permitir el paso de una celda a otra sin perder la comunicación.

C. Enlace entre varias LAN o WMAN

Otra de las configuraciones de red posibles es la que incluye el uso de antenas direccionales. El objetivo de estas antenas direccionales es el de enlazar redes que se encuentran situadas geográficamente en sitios distintos tal y como se muestra en la Fig. 2.20. Un ejemplo de esta configuración lo tenemos

en el caso en que tengamos una red local en un edificio y la queramos extender a otro edificio.

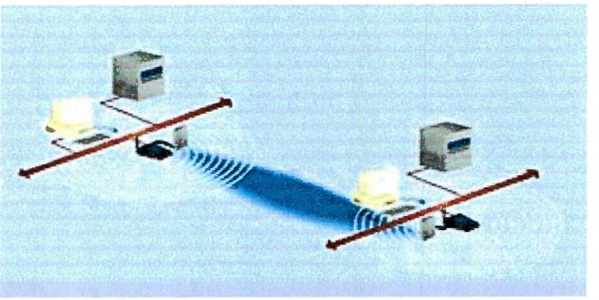


Fig. 2.20 Configuración de red de interconexión LAN o WLAN [27].

Una posible solución a este problema consiste en instalar una antena direccional en cada edificio apuntándose mutuamente. A la vez, cada una de estas antenas está conectada a la red local de su edificio mediante un punto de acceso. De esta manera podemos interconectar las dos redes locales [26], [27], [29].

Nivel Físico Arquitectura y Tecnologías de Modulación

- Arquitectura de capas 802.11

La capa física proporciona una serie de servicios a la capa MAC o capa de acceso al medio. Diferentes tecnologías de capa física se definen para transmitir por el medio inalámbrico. La capa física de servicios consiste en dos protocolos:

Una función de convergencia de capa física, que adapta las capacidades del sistema físico dependiente del medio (PMD). Esta función es implementada por el protocolo PLCP o procedimiento de convergencia de capa física, que define una forma de mapear MPDUs[3] o unidades de datos MAC en un formato

[3] MPDU. Son unidades fragmentadas del MSDU, el cual es una unidad de información recibida de la subcapa que está por encima de la capa MAC (LCC) dentro de la organización del protocolo.

de tramas susceptibles de ser transmitidas o recibidas entre diferentes estaciones a través de la capa PMD.

Un sistema PMD, cuya función define las características del medio sin cables, entre dos o más estaciones, sobre el cual se va a transmitir y recibir. La comunicación entre MACs de diferentes estaciones se realizará a través de la capa física mediante de una serie de puntos de acceso al servicio, donde la capa MAC invocará las funciones básicas para ofrecer el servicio.

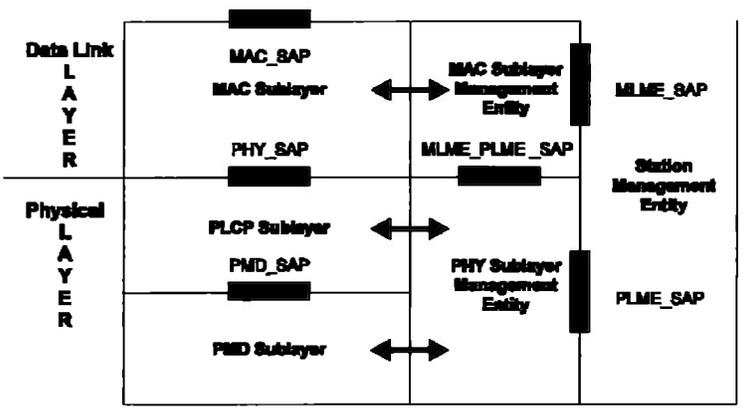


Fig. 2.21. Porción del modelo de referencia ISO/IEC comprendido entre la capa física y MAC [30].

Además de estas capas, podemos distinguir la capa física de gestión. En esta capa podemos distinguir la estructura MIB (Management Information Base) que contienen por definición las variables de gestión, los atributos, las acciones y las notificaciones requeridas para gestionar una estación. Consiste en un conjunto de variables donde podemos especificar el estado y la configuración de las comunicaciones de una estación [27], [29].

Tecnologías utilizadas en las Redes Inalámbricas

Podemos distinguir tres tecnologías, dos de espectro ensanchado y una de infrarrojos.

- Tecnologías de espectro ensanchado

La tecnología de espectro ensanchado consiste en difundir la señal de información a lo largo del ancho de banda disponible, es decir, en vez de concentrar la energía de las señales alrededor de una portadora concreta lo que se hace es repartirla por toda la banda disponible. Este ancho de banda total se comparte con el resto de usuarios que trabajan en la misma banda frecuencial. Existen dos tipos de tecnologías de espectro ensanchado:

- Espectro Ensanchado por Secuencia Directa (DSSS)
- Espectro Ensanchado por Salto en Frecuencia (FHSS)

A.1 Tecnología de espectro ensanchado por secuencia directa (DSSS)

Esta técnica consiste en la generación de un patrón de bits redundante llamado *señal de chip* para cada uno de los bits que componen la señal de información y la posterior modulación de la señal resultante mediante una portadora de RF. En recepción es necesario realizar el proceso inverso para obtener la señal de información original. La secuencia de bits utilizada para modular cada uno de los bits de información es la llamada secuencia de Barker y tiene la siguiente forma:

+1, -1, +1, +1, -1, +1, +1, +1, -1, -1, -1

En la Fig. 2.22 se muestra el aspecto de una señal de dos bits a la cual se le ha aplicado la secuencia de Barker. DSSS tiene definidos dos tipos de modulaciones a aplicar a la señal de información una vez se sobrepone la señal de *chip* tal y como específica el estándar IEEE 802.11:

- La modulación DBPSK, Differential Binary Phase Shift Keying
- La modulación DQPSK, Differential Quadrature Phase Shift Keying

Estas proporcionan velocidades de transferencia de 1 y 2 Mbps respectivamente.

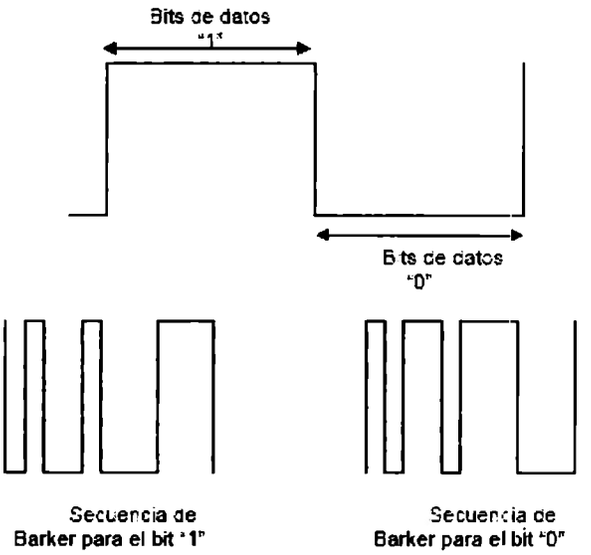


Fig. 2.22 Codificación de información mediante la secuencia Barker [31].

En el caso de Estados Unidos y de Europa la tecnología de espectro ensanchado por secuencia directa, DSSS, opera en el rango que va desde los 2.4 GHz hasta los 2.4835 GHz, lo que nos da un ancho de banda total disponible de 83.5 MHz. Este mismo se divide en un total de 14 canales con un ancho de banda por canal de 5 MHz de los cuales cada país utiliza un subconjunto de los mismos según las normas reguladoras para cada caso particular. En topologías de red que contengan varias celdas, ya sean separadas o adyacentes, los canales pueden operar simultáneamente sin apreciarse interferencias en el sistema si la separación entre las frecuencias centrales es como mínimo de 30 MHz. Esto significa que de los 83.5 MHz de ancho de banda total disponible podemos obtener un total de 3 canales independientes que pueden operar simultáneamente en una determinada zona geográfica sin que aparezcan interferencias en un canal procedentes de los otros dos canales. Esta independencia entre canales nos permite aumentar la capacidad del sistema de forma lineal con el número de puntos de acceso operando en un canal que no se esté utilizando y hasta un máximo de tres canales [26].

- Tecnología de espectro ensanchado por salto en frecuencia (FHSS)

La tecnología de espectro ensanchado por salto en frecuencia consiste en transmitir una parte de la información en una determinada frecuencia durante un intervalo de tiempo llamada *dwell time,* inferior a los 400ms. Pasado este tiempo se cambia la frecuencia de emisión y se sigue transmitiendo a otra frecuencia. Un ejemplo de esto se presenta en la Fig. 2.23, en donde se muestran diferentes tramas de información por medio de la tecnología FHSS, de manera que cada uno de estos se transmite en una frecuencia distinta durante un intervalo muy corto de tiempo.

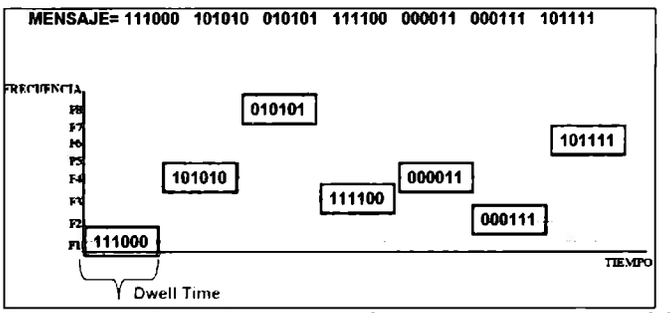


Fig. 2.23 Transmisión de tramas de información mediante FHSS [33].

Cada una de las transmisiones a una frecuencia concreta se realiza utilizando una portadora de banda estrecha que va cambiando (saltando) a lo largo del tiempo. Este procedimiento equivale a realizar una partición de la información en el dominio temporal. El orden en los saltos en frecuencia que el emisor debe realizar viene determinado según una secuencia pseudoaleatoria que se encuentra definida en unas tablas que tanto el emisor como el receptor deben conocer. La ventaja de estos sistemas frente a los sistemas DSSS es que con esta tecnología podemos tener más de un punto de acceso en la misma zona geográfica sin que existan interferencias si se cumple que dos comunicaciones distintas no utilizan la misma frecuencia portadora en un mismo instante de tiempo. Si se mantiene una correcta sincronización de estos saltos entre los dos extremos de la comunicación el efecto global es que aunque vamos cambiando de canal físico con el tiempo se mantiene un único canal

lógico a través del cual se desarrolla la comunicación. Para un usuario externo a la comunicación la recepción de una señal FHSSS equivale a la recepción de ruido impulsivo de corta duración [26]. El estándar IEEE 802.11 describe esta tecnología mediante la modulación en frecuencia FSK, Frequency Shift Keying, y con una velocidad de transferencia de 1Mbps pudiendo elevarse hasta los 2Mbps bajo condiciones de operación óptimas.

Tecnología de infrarrojo

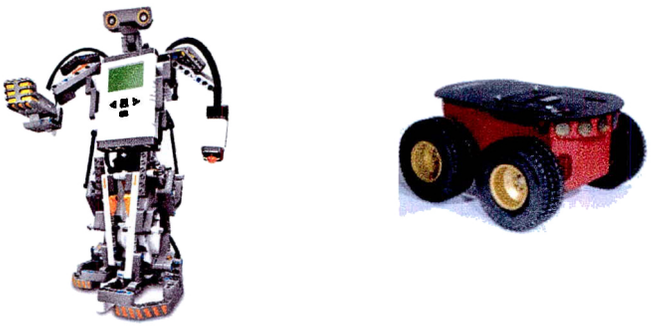
Una tercera tecnología, de momento no demasiado utilizada a nivel comercial para implementar WLANs, es la de infrarrojos. Los sistemas de infrarrojos se sitúan en altas frecuencias, justo por debajo del rango de frecuencias de la luz visible. Las propiedades de los infrarrojos son, por tanto, las mismas que tiene la luz visible. De esta forma los infrarrojos no pueden pasar a través de objetos opacos pero se pueden reflejar en determinadas superficies. Las longitudes de onda de operación se sitúan alrededor de los 850-950 nm, es decir, a unas frecuencias de emisión que se sitúan entre los $3.15X10^{14}$ Hz y los $3.52X10^{14}$ Hz. Los sistemas que funcionan mediante infrarrojos se clasifican según el ángulo de apertura con el que se emite la información en el emisor en:

- Sistemas de corta apertura, de haz dirigido o de visibilidad directa que funcionan de manera similar a los mandos a distancia de los aparatos de televisión. Esto supone que el emisor y el receptor tienen que estar orientados adecuadamente antes de empezar a transmitirse información.
- Sistemas de gran apertura, reflejados o de difusión que radian tal y como lo haría una bombilla, permitiendo el intercambio de información en un rango más amplio. La norma IEEE 802.11 especifica dos modulaciones para esta tecnología: la modulación 16 ppm y la modulación 4 ppm proporcionando unas velocidades de transmisión de 1 y 2 Mbps respectivamente. Esta tecnología se aplica típicamente en entornos de interior para implementar enlaces punto a punto de corto alcance o redes locales en entornos muy localizados como puede ser una aula concreta o un laboratorio [26].

Capítulo 3
Desarrollo del proyecto

3.1 Desarrollo práctico con el robot Pioneer 2

Para lograr los objetivos planteados en este proyecto, se comenzó con la elección de una base que brindara las características necesarias de navegación, interacción con el medio y compatibilidad multiplataforma. Para lo anterior se consideran robots comerciales como LEGO NXT y el Pioneer 2, se eligieron estos dos debido a que son las dos plataformas con las que se cuenta en la universidad (Fig. 3.1). El robot LEGO NXT, tiene muchas características útiles, como es el soporte para conexión USB y *Bluetooth*, sensores de luz, sensores de presión y ultrasónicos, pero que resultan insuficientes para el desarrollo de este proyecto, pues su tamaño no resulta práctico en el proyecto, además, no esta diseñado para soportar peso. El Pioneer 2, cuenta con un sistema más robusto, una capacidad de movimiento mayor, soporta pesos de hasta 23 kg, así como láser y sonares para evasión de obstáculos. Fueron estas características las que inclinaron la decisión por este robot. Además de la capacidad de expansión por medio de puertos y entradas analógicas y digitales.



a) LEGO NXT b)Pioneer 2

Fig. 3.1. Robots Comerciales

Una vez elegida la plataforma y teniendo una visión general del robot se pasó a la fase de la investigación y exploración de la misma, dentro de la cual se plantearon objetivos definidos.

Investigación relativa a requerimientos

Se realizó una investigación y consulta del software y hardware requerido para la operación del Pioneer 2. Resultando en distintos paquetes de software tanto en la parte de programación del robot, como de simulaciones y de creación de entornos gráficos. Los paquetes de programación que se consideraron fueron ARIA (Advanced Robot Interface for Applications), Microsoft Robotics y Player Stage. En el área de simulaciones se consideró la utilización de MobileSim y en el ámbito de creación de entornos gráficos, mapas, en dos dimensiones se encontró el Mapper3.

Las pruebas que se hicieron sobre el Microsoft Robotics no fueron útiles, pues aunque se pudieron simular dentro del programa, se necesita Windows instalado en la computadora del robot, y se tiene instalado Linux RedHat. Player Stage es mucho más sencillo y si es compatible con el robot. Estos dos programas son utilizados para la programación de diversos robots, no solo el Pioneer 2.

En el caso de ARIA, no se trata de un programa, es un conjunto de librerías cuya arquitectura (Fig. 3.2) esta enfocada en crear acciones con el robot que son programadas en lenguajes de alto nivel (Java y C++), por medio de éstas librerías se controlan los movimientos, motores, la conexión al robot o al simulador y los accesorios como el sonar o láser, su mayor ventaja es que están hechas ex profeso para los robots de ActivMedia. Es importante que mencionar que tanto MobileSim como Mapper3 fueron integrados a los paquetes de trabajo, y también están hechos para interactuar con la familia de robots ActivMedia, y con las librerías de ARIA.

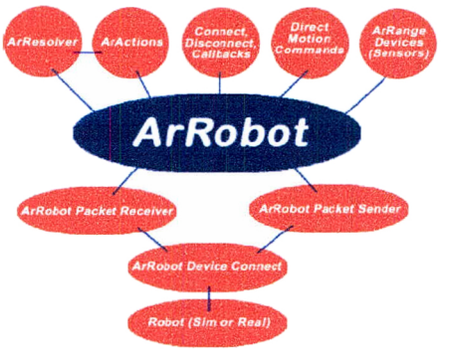


Fig. 3.2. Arquitectura de ARIA [17]

En los requerimientos de hardware se necesita una computadora personal donde se ubica la inteligencia, en nuestro caso usamos *laptops* comerciales con Windows XP como sistema operativo, con procesador Centrino y 512 MB en RAM. Por otro lado se determinó la necesidad de un cable tipo serial-USB para la realizar la conexión entre el robot y la computadora.

Primeros movimientos del robot

El siguiente objetivo fue el movimiento del robot, es decir ser capaces de controlar los movimientos del mismo. Para esto se hizo indispensable tener un conocimiento de sus funciones. Se comenzó el trabajo estudiando y entendiendo los programas de demostración incluidos con ARIA en las carpetas de *examples y advanced.* Estos programas de demostración ofrecen una gama muy variada de acciones que se pueden realizar, ya que permiten movimientos del robot en diversos modos, así como la prueba de los distintos equipos del robot. Se probaron diferentes programas que permiten entre otras cosas el movimiento del robot mediante el teclado de la computadora externa. Las pruebas fueron hechas tanto en el sistema operativo de Windows como Linux Ubuntu, esto debido a que Linux permite una compilación sencilla en comparación con Windows.

Programación básica

Entendiendo los programas para obtener movimientos por parte del robot, se elaboraron programas que permitieran el control del robot. Se inició con un programa muy básico donde el robot fue capaz de recorrer un cuadrado de lados determinados. El algoritmo a seguir es básicamente la programación de distancias definidas, y al término de el recorrido de cada lado una orden de giro de 90°. El programa anterior fue probado en primera instancia en el simulador MobileSim y posteriormente de manera práctica con el robot. MobileSim es una herramienta de mucha utilidad y valor, ya que permite simulaciones de manera rápida y precisa. Además es capaz de representar todos los sensores que se tienen físicamente en el robot (sonares, láser, cámara, etc.), con esto, garantiza que un programa que funciona sobre el simulador, funcionará sobre el robot.

En la Fig. 3.3 se tiene el programa MobileSim, donde el Pioneer 2 se encuentra en un ambiente con paredes. Del robot salen líneas grises y azules que corresponden a los sonares y al láser respectivamente.

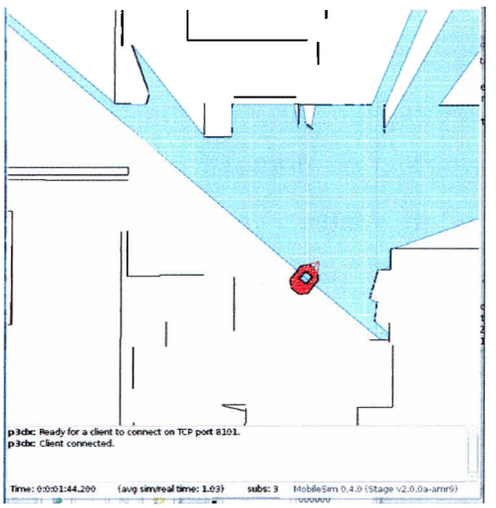


Fig. 3.3. Simulador MobileSim con sonares (líneas grises) y láser (línea azul)

Se utilizó el programa Mapper 3, ya que permite el diseño de mapas, que pueden ser cargados directamente en el simulador MobileSim. También es

posible la creación de paredes, obstáculos, puntos de inicio y puntos destino como se muestra en la Fig. 3.4. Se diseñaron diferentes mapas para probar el robot en simulaciones, uno de ellos, fue el diseño de un mapa parecido a la distribución de un supermercado.

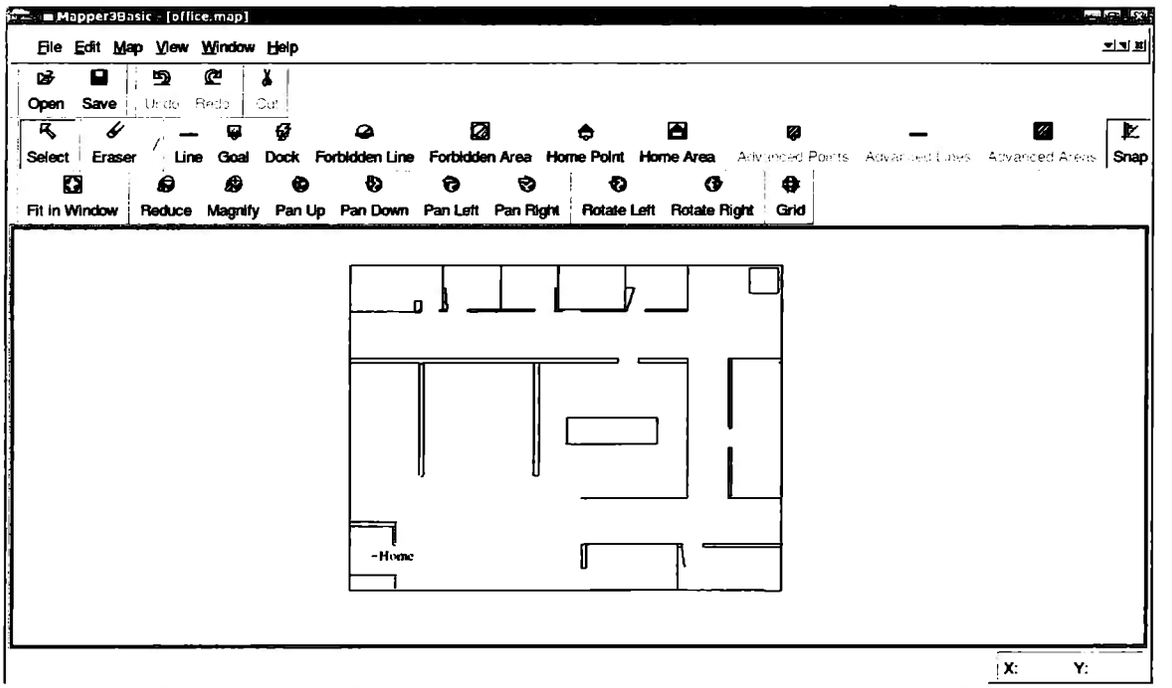


Fig. 3.4. Mapper3 usado para crear mapas para el simulador MobileSim

Obtención de lectura de sonares

La utilización de los sonares es importante para la detección de obstáculos para la navegación. Además se tiene la idea de realizar las primeras pruebas para hacer el seguimiento por este medio. Los sonares son capaces de realizar lecturas que van desde los 10 hasta los 5000mm.

Las primeras pruebas al sonar, se realizaron con los ejemplos que incluye el ARIA, para aprender a registrar las mediciones y tomar las decisiones de cómo se debe esquivar el obstáculo. También para conocer los diversos comandos que existen para las lecturas del sonar, como son medir el arreglo en

el orden que se quiere, leer solo la mitad, tomar el que tiene la lectura más cercana o más lejana, y después con esos datos elegir la acción a seguir.

Una vez realizadas estas actividades, se procede a la integración de los conocimientos adquiridos en la exploración de la plataforma para comenzar a desarrollar la solución propuesta en el proyecto.

Metodología

Para el avance en la navegación del robot móvil se implementó una metodología mediante etapas, siguiendo el esquema de la Fig. 3.5, con el fin de conocer las características de movimiento y programación además del comportamiento del prototipo en ambientes con obstáculos similar al existente en un supermercado.

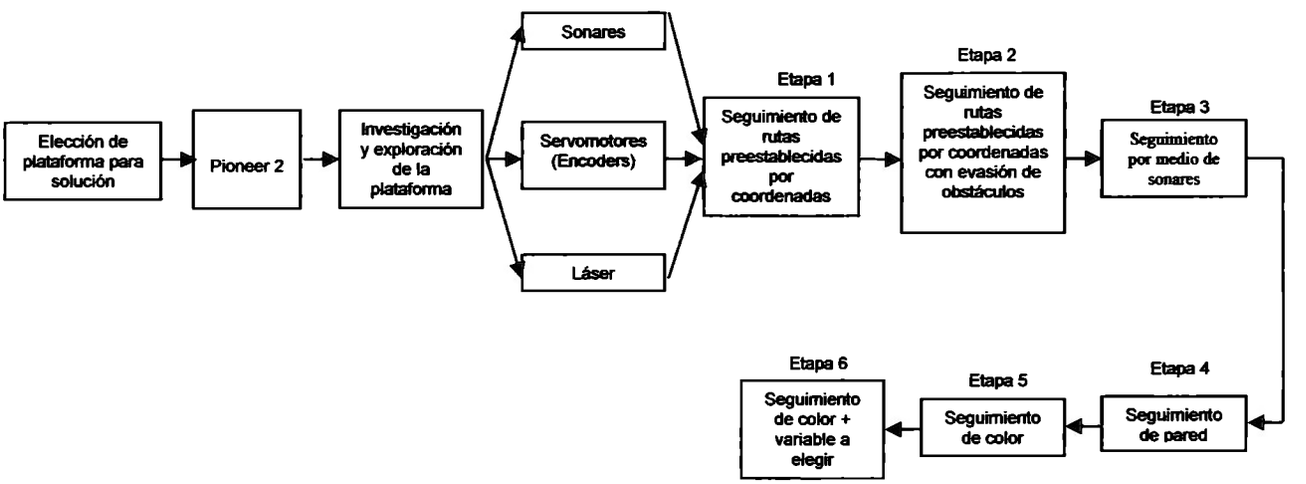


Fig. 3.5 Metodología de etapas a seguir.

Primera etapa: Seguimiento de rutas preestablecidas por medio de coordenadas.

Esta etapa consiste en la programación de una ruta preestablecida en el robot en base a coordenadas para una navegación independiente en un entorno conocido. Este tipo de navegación es independiente del usuario, ya que no requiere ningún tipo de instrucción externa, únicamente con las direcciones ya incluidas en la programación de la ruta. De esta forma, una vez corriendo el programa, el robot hace el recorrido de forma autónoma.

Los requisitos para lograr lo anterior, fueron la habilitación de la conexión serial entre una computadora y el robot para el envío de datos, inicio de programación en lenguaje C++, haciendo uso de las librerias ARIA (Advanced Robot Interface for Applications) propias de los robot de ActivMedia, y que será el lenguaje usado a lo largo del proyecto.

Se requirió además de aprender a controlar los motores y saber variar la velocidad, así como la manipulación de ángulos y velocidad de giro; finalmente, saber las coordenadas necesarias para establecer una ruta a seguir. Es importante mencionar que este tipo de navegación no considera la evasión de obstáculos, se supone un ambiente en el cual no existen obstáculos intermedios en la ruta a seguir.

Extrapolando esto al entorno final de nuestro proyecto, un supermercado, la navegación preestablecida sería capaz de guiar al robot a través de los diferentes pasillos, pero cualquier obstáculo intermedio implicaría una colisión.

El objetivo que persigue esta etapa fue conocer las bases de la navegación del robot, su desempeño, la capacidad de respuesta para movimientos simples y la forma en que realiza los recorridos.

Se realizaron diferentes pruebas respecto a las instrucciones a utilizar dentro de la programación. Esto debido a que si se programa al robot que llegue de un punto a otro únicamente a través de coordenadas como se muestra en el Anexo IIa, el seguimiento de la ruta no es preciso, la vueltas no son exactas. Es por eso que la ruta no se recorre de una forma perfecta, en la Fig. 3.6 se observa con una línea azul la trayectoria recorrida por el robot.

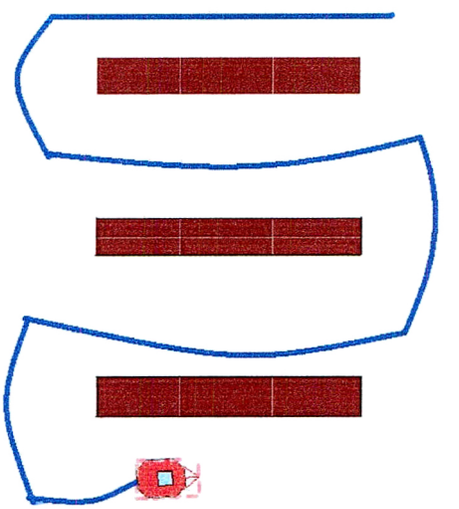


Fig. 3.6 Seguimiento de ruta preestablecida,
no se sigue de forma exacta.

La metodología de solución fue programar las rutas por coordenadas en conjunción con instrucciones de movimiento, así, las vueltas necesarias en la ruta se realizan con una instrucción específica de "girar", y una vez realizada la vuelta, el robot avanza hacia las coordenadas predeterminadas. Gracias a esta modificación la ruta se sigue de manera precisa evitando que al girar las vueltas sean amplias y la trayectoria se siga de forma correcta como se puede observar en la Fig. 3.7. El código correspondiente a esta etapa se encuentran en el anexo II.

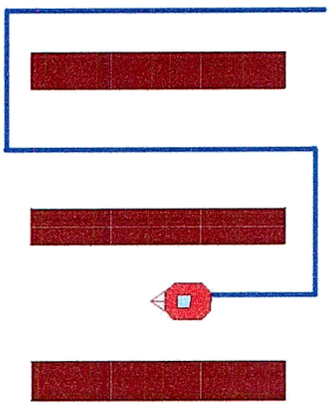


Fig. 3.7. Seguimiento exacto de la ruta preestablecida

Segunda etapa: Seguimiento de rutas preestablecidas por coordenadas con evasión de obstáculos

Para ésta etapa se retoma todo lo aplicado en la etapa anterior, pero con el objetivo de evadir posibles obstáculos. Es necesario contar con el arreglo de sonares funcionando; las lecturas son registradas, y en caso de detectar un obstáculo a una distancia determinada, el robot gira para esquivarlo.

El objetivo principal de esta etapa, es lograr la integración de los sonares para lograr una navegación segura. Una vez que ya se ha logrado el seguimiento de una ruta preestablecida, el programa usado permite detectar objetos y evadir los obstáculos sin perder la ruta que se está siguiendo.

La evasión es similar a un *bug algorithm,* donde el robot solo tiene conocimiento local del entorno y un objetivo global, es por eso que para la evasión del obstáculo debe realizar múltiples movimientos para garantizar que el obstáculo ya no está y lo pudo esquivar para seguir con la ruta establecida. La evasión de obstáculos, se muestra en la Fig. 3.8.

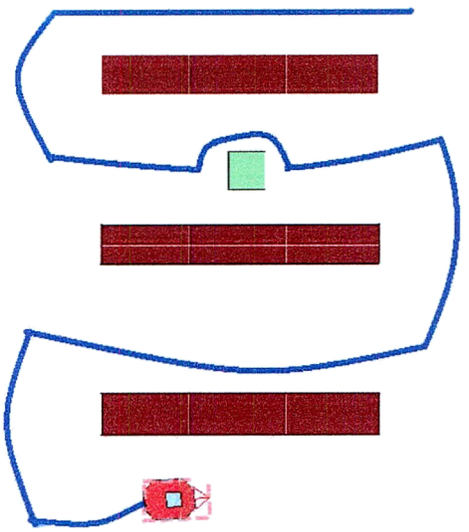


Fig. 3.8 Seguimiento de una ruta preestablecida con evasión de obstáculos.
El seguimiento de la ruta no es preciso.

Es importante mencionar que el algoritmo se enfoca a obstáculos estáticos, aun no se ha implementado para obstáculos móviles, dado que las decisiones a tomar serían diferentes, pues podría estarse moviendo la misma dirección que el obstáculo y no habría evasión; esto se puede observar en el anexo IIIa. Sin embargo, para el caso de objetos estáticos la evasión toma menos variables dado que el obstáculo no se moverá.

Esto es posible, como se mencionó anteriormente, gracias al arreglo de sonares con el que se cuenta, donde la distribución (Fig. 3.9) brinda un monitoreo del entorno donde se está moviendo el robot. La visión de 180°, repartida en 8 sonares, hace posible no solo determinar la existencia de un obstáculo si no, saber si éste se encuentra enfrente, a la izquierda o la derecha, datos que permiten tomar la decisión de hacia donde se debe mover el robot para una evitar la colisión. El código C++ correspondiente se encuentra en el anexo III.

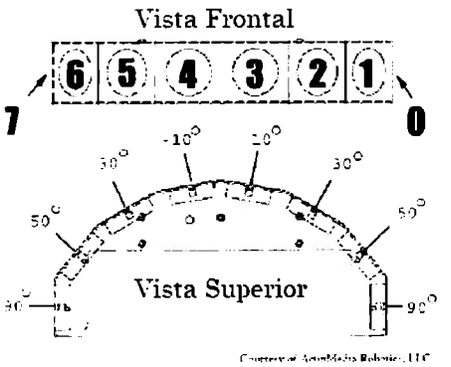


Fig. 3.9. Arreglo y distribución de sonares. [18]

Tercera etapa: Seguimiento por medio de sonares

El objetivo de esta etapa de desarrollo representa el primer acercamiento al seguimiento de un objetivo móvil. Este brindará la posibilidad de conocer el comportamiento del robot bajo estas condiciones en un entorno sin obstáculos y más adelante con obstáculos simples por medio de una señal ultrasónica generada y detectada por lo sonares.

La metodología para lograr el seguimiento por medio de sonares se realizó mediante la determinación de rangos como condicional del avance del robot. Se propusieron tres condiciones, una para movimientos de avance, retroceso y una de espera. De los ocho sonares que conforman el arreglo se utilizan únicamente los dos frontales para lograr el seguimiento. Los movimientos hacia adelante se encuentran regidos por la condición de un rango determinado, en este caso entre 1 y 2 metros de distancia entre el robot y el objeto debe realizar un movimiento hacia adelante a una velocidad previamente programada dada en milímetros sobre segundo. El segundo intervalo establecido esta dado entre una distancia de 0 a 0.5 metros como se puede observar en el anexo IVa. Cuando el objeto es localizado en este intervalo el robot retrocede a una velocidad establecida que es comparativamente menor a la velocidad de avance. El tercer intervalo, entre 0.5 y 1 metro, es un punto donde el robot no realiza ninguna acción, esto significa que se queda estático.

En la Fig. 3.10, se observa una fotografía donde se puede ver el seguimiento por parte del robot.

En esta modalidad no hay distinción entre objetos predeterminados, sino que se realizará el seguimiento de cualquiera de ellos que se encuentre dentro del rango de los sonares.

Los requerimientos para esta etapa son los elementos básicos de navegación mencionados en las etapas anteriores sumando a estos el arreglo de sonares. De igual manera fueron necesarios comandos de declaración y obtención de lecturas independientes de cada uno de los 8 sonares.

Fig. 3.10. Seguimiento de persona por el robot.

Evoluciones y mejoras al algoritmo anterior

Mejoras al algoritmo anterior permitirán al robot realizar seguimiento no únicamente lineal como se describió anteriormente, sino que ahora el robot tendría capacidad de giros o movimientos que pudiera realizar el usuario como se muestra en el anexo Va. La finalidad de este algoritmo es lograr un

acercamiento a una situación más real a la situación anterior. El código correspondiente al seguimiento por sonar, se encuentra en el anexo IV.

El siguiente paso para completar ésta etapa será la realización de un algoritmo que permitirá al robot, además de lo anterior, realizar vueltas de 90° al final de un pasillo sin perder el seguimiento que hacía de la persona.

Cuarta etapa: Seguimiento de pared

El seguimiento de pared es uno de los algoritmos más utilizados en el mundo de la robótica móvil. Este se basa en guiar el movimiento del robot de manera paralela a una pared tomando como referencia la distancia arrojada por un sensor que monitorea la presencia del muro. La planeación de la programación de este algoritmo consiste en permitir una navegación libre en un ambiente desconocido hasta encontrar una pared por medio de los sonares, posicionarse paralelamente a ella usando los sonares laterales (0 y 7), y avanzar manteniendo distancia y velocidad constantes como se muestra en la Fig. 3.11. Es importante mencionar que la presencia de la estructura es una condición suficiente para el avance del robot.

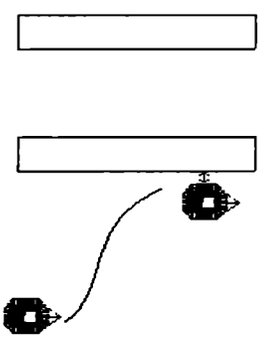


Fig. 3.11 Camino a seguir en algoritmo de seguimiento de pared

El implementar una solución de este tipo, podría ser útil en un ambiente de supermercado ya que el robot podría recorrer los pasillos. Con ayuda de los sonares se condicionan los giros y el avance del robot evitando así colisiones.

Si bien una etapa de estas características es importante en la navegación de robots móviles, se determinó que este algoritmo no era necesario ni vital para el avance del proyecto, principalmente debido a que el movimiento del robot es independiente al movimiento de la persona, acción contraria a lo deseado.

Por otro lado, las características del seguimiento por pared que podrían resultar de mayor utilidad para lograr un seguimiento de color están contenidas en la Etapa 3: Seguimiento por Sonar. Como se mencionó en dicho apartado, el seguimiento por sonar provocaría el frenado del robot en caso de detectar un objeto cualquiera a un rango de distancia predeterminado; esto evitaría cualquier choque de con cualquier objeto en el ambiente.

Quinta etapa: Seguimiento por color

Como su nombre lo dice la quinta etapa tiene como objetivo lograr el seguimiento de un objeto móvil predeterminado por medio de color. Este método tiene ventajas y desventajas, éstas radican principalmente en la facilidad para la obtención de los medios y recursos requeridos; es por ello que se consideró como una muy buena opción de seguimiento. Es importante mencionar que esta etapa se divide en 2 fases, la de adquisición y segmentación de las imágenes y la programación del robot.

Fase I

Para poder partir se tomaron en cuenta consideraciones y requerimientos que eran necesarios tanto de *software* como de *hardware* que permitirían la adquisición de las imágenes y su procesamiento. El *software* que se eligió para el procesamiento de las imágenes es Matlab debido a que es una herramienta muy completa para el procesamiento de señales, sin embargo al tener una gran variedad de posibilidades resulta sobrado para esta aplicación específica, lo que

se traduce en mayor utilización de recursos en la máquina. El *hardware* utilizado es una cámara Web que es la encargada de la adquisición de las imágenes.

Mediante la conjunción de estas 2 herramientas y tras la programación de códigos que permiten el procesamiento de la imagen se logró la localización de objetivos específicos. La metodología antes mencionada se expresa en el siguiente diagrama (Fig. 3.12).

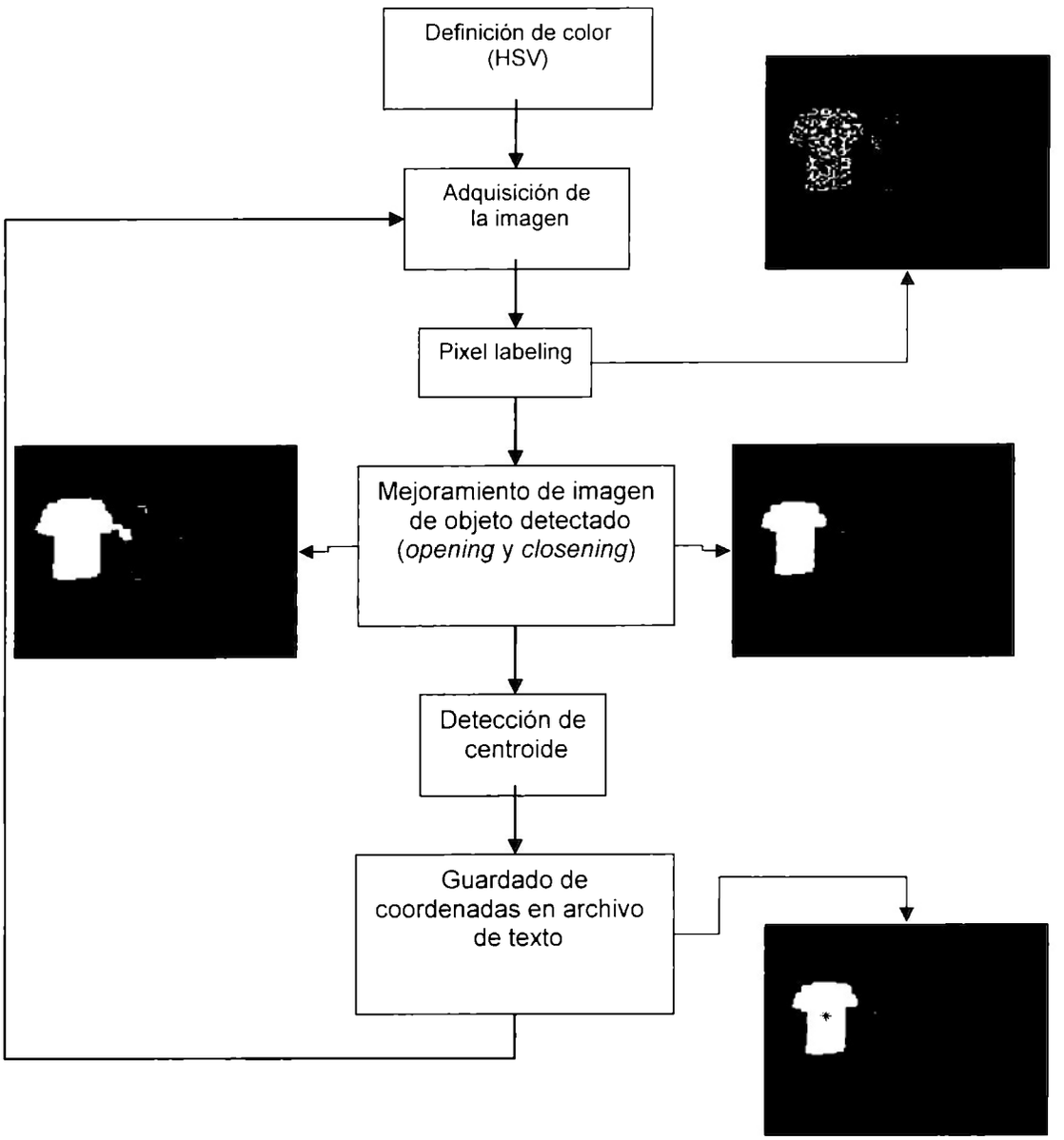


Fig. 3.12 Metodología de procesamiento de la imagen

Para poder llevar a cabo la localización del objetivo que se ha de seguir, se llevó a cabo una segmentación de color, en las imágenes que se obtienen de la cámara Web.

La definición del color a seguir, se hace en el espacio HSV, y para escogerlo, solo se definen los valores de H (tonalidad) y S (saturación). De esta forma, se definió el color a seguir, en esta etapa, solo se escogió un color naranja, así, los intervalos rangos de H y S para este color, fueron 0.03 - 0.13 y 0.8 - 1 respectivamente. El valor de V no define el color, se debe cuidar que su valor sea diferente de cero, para que el color no se considere un negro.

La definición de cada uno de los intervalos para H, S y V se hace de la siguiente forma, indicando los valores de inicio y fin, así como el tamaño de paso en la matriz.

```
h = inih:0.002:termih ;
s = inis:0.002:termis ;
v = repmat([iniv:0.002:termiv],[length(hh),1],1) ;
```

Una vez que el color se ha definido, se procede a obtener una imagen de la cámara Web, lo que se obtiene, es una imagen parecida a la que se presenta a continuación (Fig 3.13.):

Fig. 3.13 Imagen obtenida de la cámara Web

Con la imagen, ya se puede hacer la segmentación de color. Esto es a través de un proceso de etiquetado de píxeles que permite identificar dentro la imagen, los píxeles que corresponden al color que se quiere seguir, devolviendo una imagen binaria con los píxeles identificados como se muestra en la Fig. 3.14. Lo anterior se realizó utilizando el código de la sección

Fig 3.14 Imagen binaria con pixeles en blanco correspondiente al color establecido.

Este proceso de definir la matriz y etiquetar los píxeles, es una adaptación del algoritmo usado para la identificación de color en los robots usados en la RoboCup [34]. El código referente a lo anterior se encuentra en el anexo VII a.

Una vez realizados los dos procesos anteriores, se puede considerar que el objetivo ha sido localizado, sin embargo, se aplican dos métodos para mejorar la imagen y así mismo, la confiabilidad sobre el objeto detectado. Específicamente, se hace una implementación de algoritmos que usan funciones de *opening y closening*.

El *closening* permite "cerrar" todos los huecos dentro de la imagen, para hacerla más homogénea, esto es, que la imagen ya no sea un montón de píxeles separados, sino una imagen más sólida con una detección del objetivo más confiable. Acto seguido, se realiza un *opening*, permite eliminar píxeles dispersos dentro de la imagen, esto garantiza que si se localizaron píxeles con

tonalidades similares pero no corresponden al objetivo original, son eliminados para evitar confusiones al momento de hacer el seguimiento, esto permitirá que la imagen sea mucho más acertada con respecto al objetivo localizado

Ya que se garantiza la localización del objetivo, se obtienen las coordenadas del centroide del objetivo localizado, de esta forma, se sabe de forma específica, donde se localiza el objetivo, con esto, el robot tendrá la información necesaria para tomar decisiones respecto a los movimientos pertinentes para el seguimiento.

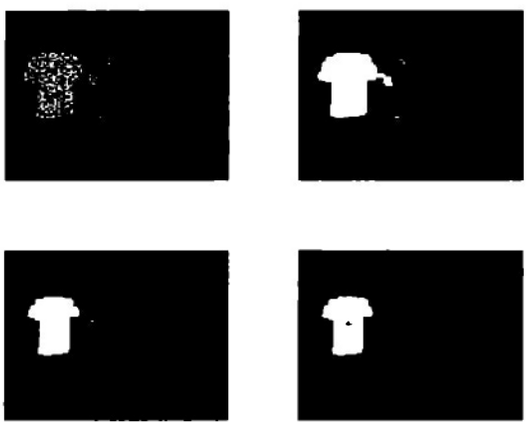


Fig. 3.15 Segmentación de color, *opening*, *closening*
Y localización de centroide

En la imagen (Fig. 3.16) se pueden observar los procesos antes mencionados, y se aprecia que de la segmentación original de la imagen, donde se hace una detección poco confiable, se logra una imagen más homogénea y confiable. El cuadro superior derecho, muestra el proceso de *closening*, y el inferior izquierdo el *opening*. Se observa que a pesar de que se realizó una identificación de píxeles con tonalidades similares, después del *closening* y *opening* ya no se aprecian. Finalmente, se tiene la localización del centroide del objeto detectado que en la imagen se aprecia con un asterisco. El código correspondiente a estas funciones se encuentra en el anexo VIIb.

Cuando se han obtenido las coordenadas del objetivo, se debe informar al robot acerca de esas coordenadas. Para hacerlo, el programa en Matlab genera una serie de archivos *.txt*. El primer archivo es llamado *bandera* que permitirá la sincronización con el robot, para informarle si puede, o no puede hacer la lectura de las coordenadas. El segundo archivo es llamado *coordenada*, que es donde se guarda la coordenada en x del objetivo que se localizó. El código de esta operación, se encuentra en el anexo VIIc. Cuando el robot lee el valor de esa coordenada, podrá tomar la decisión del movimiento que ha de realizar. Este proceso se profundiza en la fase II de esta etapa.

Fase II

Esta fase toma como punto de partida el archivo de texto generado por Matlab con la bandera y coordenada "x" del centroide del objeto. El archivo de texto es leído mediante una rutina encontrada en una referencia bibliográfica [35]. El primer archivo de texto leído por el programa es el de la bandera. En este se encuentra guardado un valor de tipo *float* que puede ser un "1" o un "0" y lo guarda en una variable (bandera). En caso de "0", el programa no continúa hacia la lectura de el archivo de texto que contiene la coordenada "x", sino que se repite dicha instrucción hasta que el estado de la bandera cambie a "1" (es decir, que la medición en Matlab haya terminado). Una vez que la variable (bandera) tenga el valor de "1.00", comienza la lectura del archivo de texto que contiene la coordenada "x". Cuando se ha leído y guardado la coordenada, se activa una segunda bandera (bandex) que indica que el programa ha terminado de hacer las lecturas y ahora se puede proseguir a la interpretación de los datos obtenidos.

Para lograr el movimiento del robot, se toman las siguientes consideraciones:

- La pantalla está dividida en tres secciones (Fig. 3.16) (la central de avance y las laterales de giro).

•La cámara utilizada para la obtención de imágenes tiene una apertura de 46°. (Fig. 3.16)

•La imagen tiene un ancho de 117 pixeles.

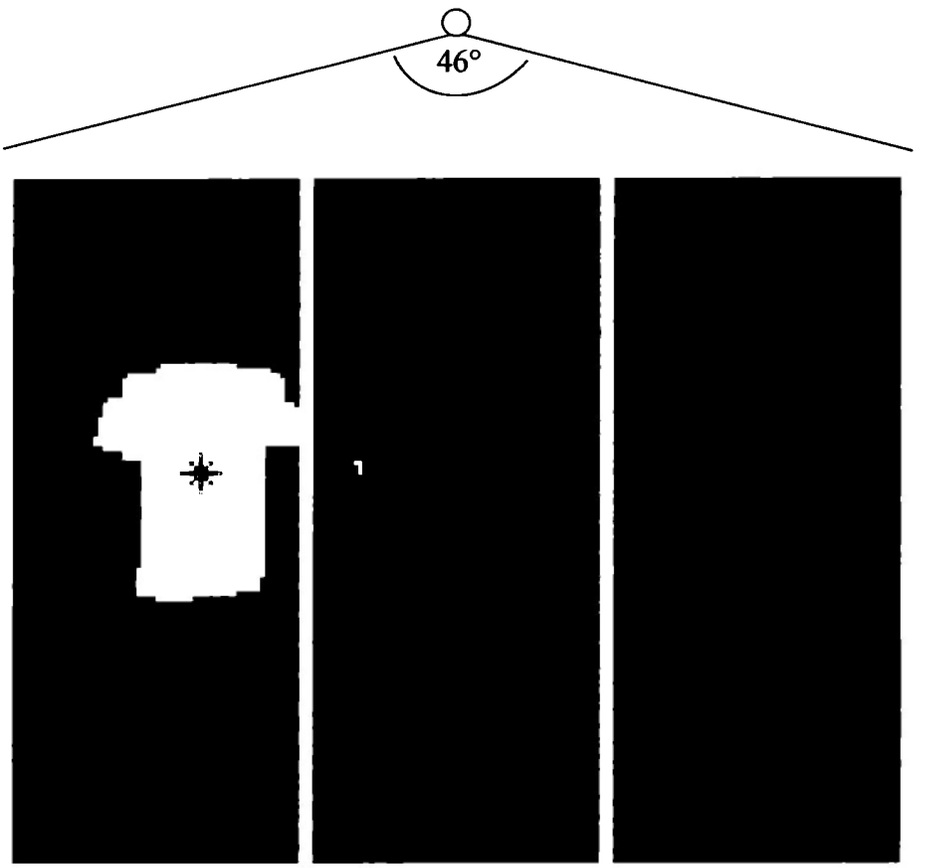


Fig. 3.16 División de la imagen en secciones de avance y giro

La división del conjunto de valores de "x" que se pueden recibir en tres intervalos rangos (que representan su valor en pixeles) condiciona el accionar del robot; dichos rangos son:

- De 0 a 39 (giro a la izquierda)

•De 39 a 78 (avance)

•De 78 a117 (giro a la derecha)

Estas mediciones se hacen si y sólo si la bandera de finalización de lecturas (bandex) está encendida (en "1").

Giro

Al detectar un valor que se encuentre en un rango de giro, el programa realiza el siguiente cálculo para obtener el ángulo de rotación:

1. Se divide 46 (apertura de la cámara en grados) entre el ancho de la imgen (117 pixeles) para obtener el número de pixeles al que equivale un grado.
2. Se resta el valor del punto medio de la imagen ($\frac{117}{2} = 58.5$) al valor obtenido de las lecturas de matlab. Lo anterior se hace con la finalidad de ubicar la distancia entre el centro y la coordenada x del centroide
3. El resultado de la resta anterior se multiplica por el valor de un grado en pixeles y por menos uno para para ajustar el giro del robot como se muestra en la Fig. 3.17 (si el ángulo es positivo el giro es a la izquierda, si es negativo el giro es a la derecha).

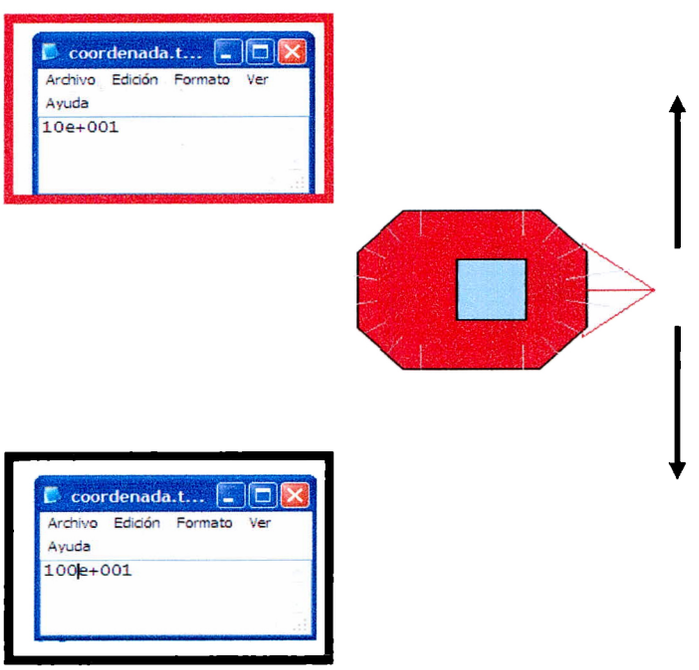


Fig. 3.17 Giro del robot

Una vez que se localizó el rango de la imagen, el robot gira hacia ese rango hasta lograr ubicar el objeto como centro de la imagen. Cómo se muestra en el anexo VIa.

La navegación del robot se encuentra condicionada a la detección de color, si el procesamiento en Matlab no detecta algún color (lectura del archivo de texto arroja un valor de 400), entonces el robot tiene la instrucción de girar hacia la izquierda sobre su eje hasta que la cámara detecte el color (Fig. 3.18). Esto se puede ver en el anexo VIb.

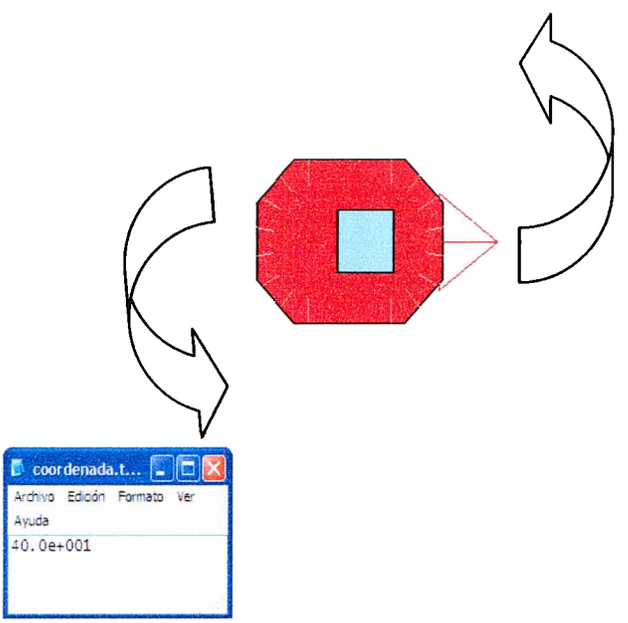


Fig. 3.18 Robot gira sobre su eje en caso de no encontrar objetivo

Cabe resaltar que los giros no están condicionados a las lecturas de los sonares para tener evasión de obstáculos simple.

Avance

Como se mencionó anteriormente, si la programación del robot lee un valor de la coordenada x que se encuentre entre 39 y 78, el robot avanzará hacia el frente a una velocidad constante de 300mm/s. Esta velocidad fue elegida por medio de pruebas tomando en cuenta el movimiento de una persona dentro de un supermercado.

El movimiento hacia el frente del robot esta condicionado por las lecturas de los sonares, en caso de cercanía detectada en los sensores [1, 2, 3, 4, 5, 6] el tiene la instrucción de detenerse. Dicha programación fue pensada para permitir al usuario detenerse a tomar algún producto e introducirlo a la canastilla del carrito. Esto se programó mediante el código del anexo VIc.

La metodología seguida en la programación del robot se muestra en siguiente diagrama (Fig. 3.19) y el código correspondiente, en el anexo VIId.

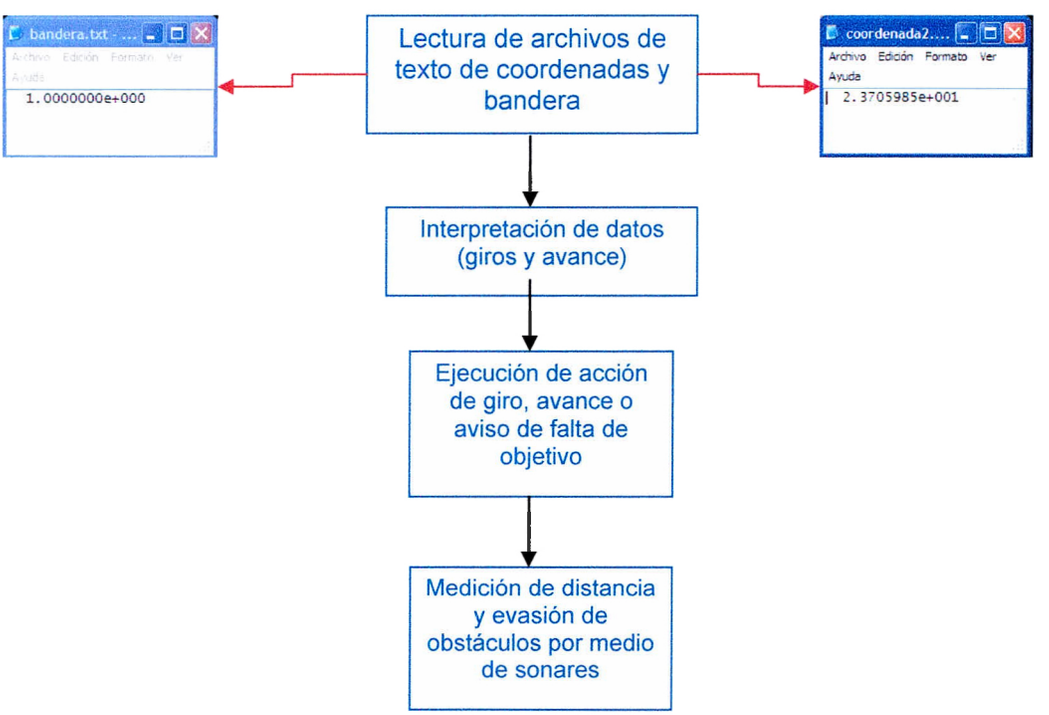


Fig. 3.19 Metodología de programación del robot

Sexta Etapa: Seguimiento de color más una variable a elegir

Al igual que la etapa anterior, esta etapa se encuentra dividida en una fase de segmentación de color y otra de programación del robot.

Fase I

El proceso seguido en esta etapa está basado completamente en el utilizado para la etapa 5. El principal cambio, es que al considerarse dos colores a localizar, se hace la definición de una matriz para cada color, después una segmentación independiente para cada color, lo que permite identificar los píxeles correspondientes a cada color definido. Al tenerse diferentes colores, se definen diferentes rangos para cada uno de los colores. Se realizaron pruebas con 3 diferentes colores y con los pares de combinaciones posibles. Los intervalos de color utilizados se muestran a continuación en Tab. 3.1.

Tab. 3.1 Rangos de Colores

Color	H	S
Naranja	0.03 – 0.13	0.5 – 1
Verde	0.25 – 0.35	0.2 – 0.6
Rosa	0.88 – 0.98	0.3 – 0.8

Los colores escogidos para las pruebas finales fueron la combinación de verde y rosa, se escogieron esos colores debido a fueron las tonalidades que menos se ven afectadas respecto a la luz dentro de las pruebas de iluminación controlada que se tuvieron.

Una vez que se hace la segmentación de cada color, se juntan ambas imagen, y se realiza una evaluación del número de píxeles identificados para cada color, si el número de píxeles identificados para alguno de los colores, es

cero, significará que no se hizo una identificación del objetivo deseado. Sin embargo, si se tienen píxeles de ambos colores, se procederá a la aplicación de *closening* y *opening* para mejorar la imagen. Una vez mejorada la imagen del objeto de dos colores localizado, se obtendría las coordenadas del centroide y su correspondiente escritura en un archivo txt así como la creación del archivo txt *bandera.* Los pasos mencionados se muestran en el siguiente diagrama (Fig. 3.20) y el código correspondiente en el anexo VIIIa.

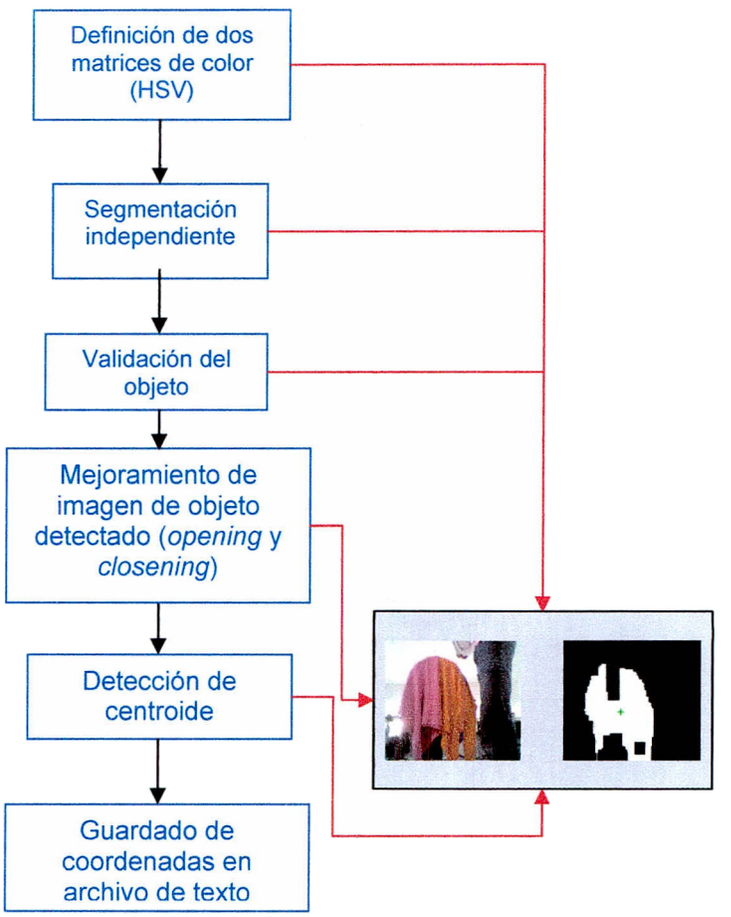


Fig. 3.20 Metodología de procesamiento de imagen para Segmentación de dos colores.

Prueba de patrones de colores

Como parte de las pruebas realizadas para la segmentación de dos colores se consideraron diferentes patrones en la posición de los mismos con el fin de buscar el patrón más adecuado para un mejor seguimiento.

Se consideraron tres patrones, que se muestras a continuación en las figuras (Fig. 3.21, Fig. 3.22 y Fig. 3.23), con su segmentación correspondiente.

1er patrón.

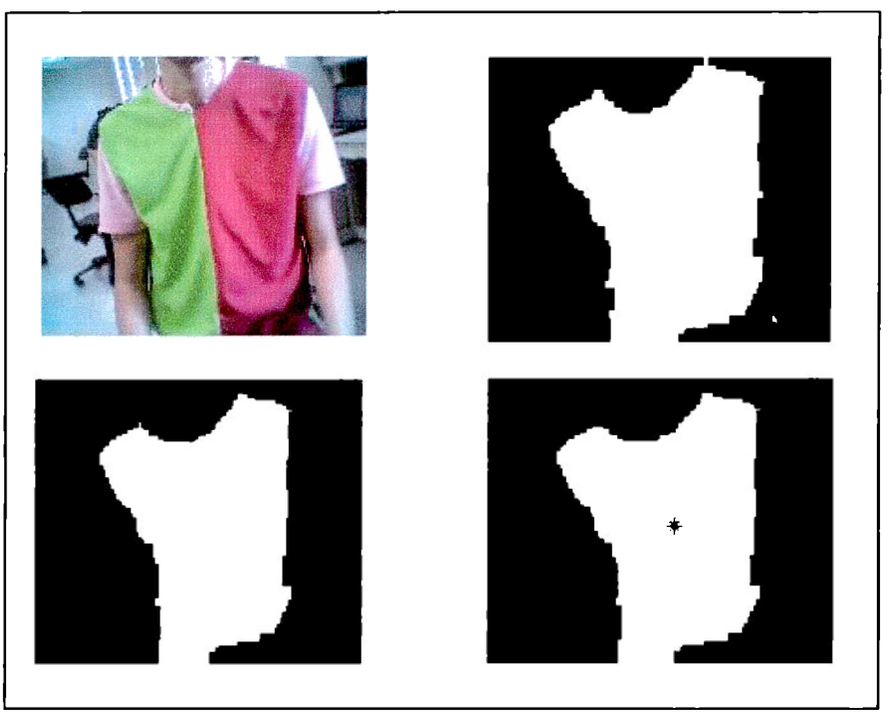


Fig. 3.21 Patrón con una franja de cada color

2do patrón.

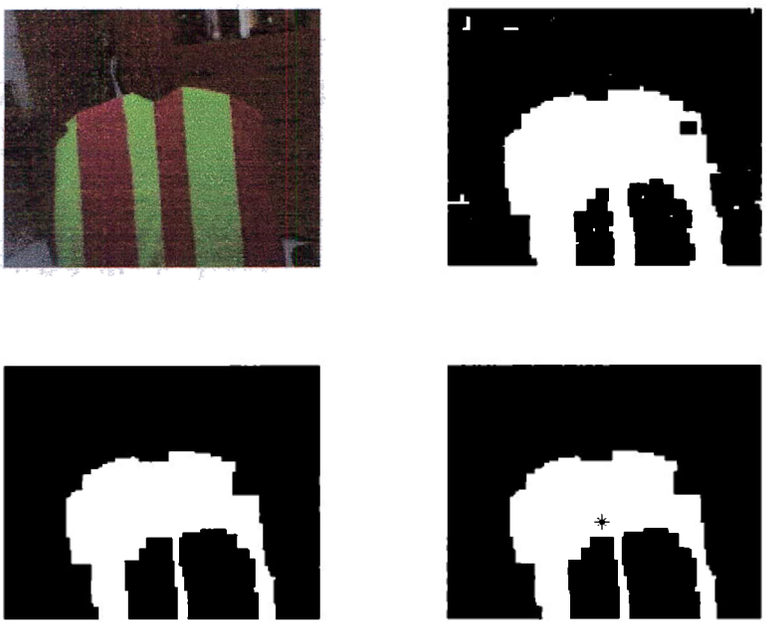


Fig. 3.22 Patrón con dos franjas de cada color

3er patrón

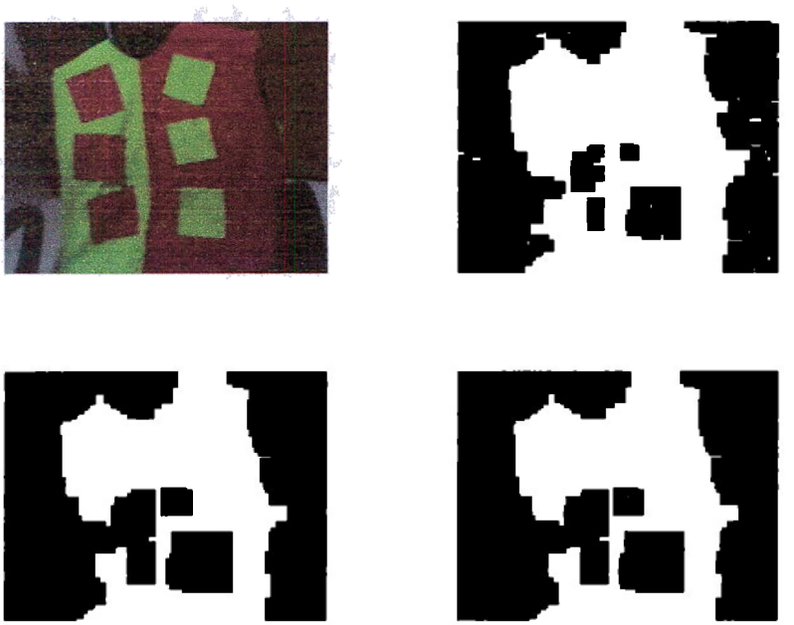


Fig. 3.23 Patrón con una franja de cada color y cuadros.

Se comprobó que en los patrones existe una correcta segmentación de los colores, como se puede observar en cada una de las tres imágenes. En todos los casos se observa que las funciones de *opening* y *closening*, mejoran la imagen del objeto detectado, así como una correcta localización del centroide, por lo que se concluye que cualquiera de los patrones es funcional sin importar la posición y forma de los colores.

Fase II

Esta fase retoma la programación del robot realizada en la quinta etapa. El algoritmo de obtención de de mediciones, cálculo de ángulo de rotación, giro y avance, es retomado de la etapa anterior. No obstante, se realizan las siguientes adecuaciones:

La velocidad de avance del robot se maneja en dos rangos: uno lejano y uno cercano:

- Rango lejano: Este se da cuando el robot recibe la instrucción de avance y la lectura de los sonares es mayor a 5 metros. En este caso la velocidad de es de 380mm/s, esto es para alcanzar al objetivo y con el fin de evitar que la distancia entre ellos crezca.

- Rango cercano: Este se da cuando el robot recibe la instrucción de avance y la lectura de los sonares es menor a 5 metros. En este caso la velocidad del robot es de 300mm/s.

Lo anterior se puede observar en el anexo Xa

De la misma forma el código de de giro en caso de no localizar un objetivo, se modificó para realizar el giro de acuerdo a la localización del último objetivo, es decir si:

- El centroide se encuentra en el rango de 0 a 39 pixeles, el giro se realiza hacia la izquierda.

- El centroide se encuentra en el rango de 78 a 117 pixeles, el giro se realiza a la derecha.
- El centroide se encuentra en el rango de 39 a 78 pixeles, el giro se realiza a la derecha (esta dirección se eligió de manera aleatoria).

Lo anterior se realizó con el fin de disminuir el tiempo de localización del usuario aumentando la probabilidad de encontrarlo al girar a donde posiblemente se dirigió el objetivo. El código correspondiente se encuentra en el anexo VIb.

3.2 Desarrollo de aplicaciones inalámbricas

Se desarrolló una aplicación inalámbrica con el fin de que el usuario final contará con un portal en la red local del supermercado, en el cual se mostrará información muy precisa acerca de los productos con los que contaba el supermercado, además de ofrecer una guía para el traslado y la localización de los mismos. La transmisión de datos como lo son las fotografías, imágenes animadas, datos de base de datos, son relevantes en el desarrollo de esta aplicación.

La tecnología que se adaptó mejor para el transporte de la información, después de un análisis de las tecnologías inalámbricas más recientes en el mercado fue Wi-Fi. Funciona por medio de radio frecuencia y tiene características variadas dependiendo de los dispositivos utilizados para enviar así como para recibir la información. Wi-Fi se basa en los estándares de la IEEE, en la cual se especifican las frecuencias de operación de los dispositivos, las cuales varían desde los 2.4GHz hasta los 5GHz. Regularmente la velocidad de transmisión es de 11Mbps hasta 54Mbps, aunque actualmente se han alcanzado velocidades de transmisión de 108Mbps. El número máximo de usuarios a los cuales se puede dar servicio al mismo tiempo es ilimitado, quedando solo la capacidad del *Access Point* como posible limitante para este caso. El alcance de

la cobertura para esta tecnología se encuentra alrededor de los 60 metros por cada uno de ellos. Los equipos utilizados tienen un consumo de energía de 3W, que comparada con otros dispositivos usados para transmitir información de manera inalámbrica en otras tecnologías, como lo es *Bluetooth,* el consumo de energía es muy elevado, dado que esta última tecnología promedia los 100mW.

Wi-Fi a pesar de que cuenta con algunas desventajas, como lo es su costo, los dispositivos portátiles requeridos para la recepción de datos, demandan una mayor cantidad de energía para su funcionamiento, por lo cual se deberá instalar una batería dentro del dispositivo que permita un funcionamiento adecuado de por lo menos 2 horas, que es el promedio de compras en un supermercado.

Para la creación de la aplicación inalámbrica, la cual utiliza como medio transporte de la información la tecnología Wi-Fi, se requirió de un programa que permitiera el desarrollo de una interfaz gráfica, práctica, agradable, llamativa y fácil de usar, por lo cual se utilizó el software desarrollado por *ADOBE* llamado Flash. Por otra parte el desarrollo de base de datos para el almacenamiento, tanto de los productos que se encuentran en el supermercado, como de los usuarios que pueden tener acceso, así como el fácil acceso y despliegue de la información contenida en cada una de estas bases de datos era parte central del proyecto; es por esto que se utilizó software como PHP y MySQL.

Aplicación Flash

Es probablemente uno de los programas de desarrollo de aplicaciones, diseño e implementación de páginas Web con más éxito. Permite la creación de animaciones vectoriales, los cuales permiten la implementación de animaciones que requieren poca cantidad de información para ser procesadas, son fácilmente redimensionables y alterables por medio de funciones, lo que permite un almacenamiento "inteligente" de las imágenes, sonidos y videos empleados.

Existen dos tipos de animaciones o gráficos que pueden ser desarrollados en este programa:

Las animaciones vectoriales; en las cuales la imagen u objeto utilizado es representado por medio de vectores que poseen determinadas propiedades como los son el color, grosor, textura, etc. Su principal característica es la adaptación de los objetos al tamaño en el cual se desea visualizar, permitiendo una visualización real de la imagen no permitiendo que se expandan los pixeles.

Las animaciones en mapa de bits; estas son parecidas a la cuadrícula en la cual cada uno de los pixeles muestra un color determinado. Toda la información contenida en este tipo de gráficos es dependiente de la variación del tamaño del objeto y de la resolución del dispositivo en el cual se esté desplegando, pudiendo perder calidad conforme se expande la imagen.

La optimización de los objetos, animaciones y eventos que se desarrollaron para visualizarse permiten dar una sensación de vida a una Web sin que para ello el tiempo de carga de la página se prolongue demasiado, provocando la pérdida de tiempo del usuario. Por otra parte a parte de este aspecto estético, Flash permite la interacción con el usuario, por medio de un lenguaje de programación llamado *Action Script*. Este lenguaje está orientado a objetos y tiene mucha influencia de *Javascript,* lo cual permite gestionar los formularios presentados, ejecutar distintas parte de la animación en función de eventos producidos por el usuario, cambiar de página o de aplicación, etc.

De este modo se usó este software pensado para aportar vistosidad a nuestra aplicación al mismo tiempo que permitirá tener una interacción en todo momento con el visitante del supermercado.

PHP

Es un lenguaje de programación diseñado principalmente para la creación de páginas Web dinámicas, es usado comúnmente para la interpretación dentro del servidor pero también puede hacer la función de una interfaz de línea de comandos o interfaz gráfica. Su acrónimo significa *Hypertext Pre-processor* y es producido hoy en día por *The PHP Group*, publicado bajo la PHP license, la Free Software Foundation lo considera como software libre.

Por lo regular PHP se ejecuta dentro de un servidor como código de entrada, creando páginas Web como salida. El gran parecido que posee este lenguaje de programación con lenguajes como C, permite crear aplicaciones complejas en un corto plazo. Su funcionamiento general es el siguiente:

Cuando el usuario hace una petición al servidor para que le mande información, el servido ejecuta el código PHP mediante un intérprete o compilador, éste procesa el script que se genera de manera dinámica, en nuestro caso es una base de datos. El resultado de la interpretación es enviado al cliente mediante extensiones como PDF o Flash, al igual que imágenes.

La conexión a diferentes bases de datos como MySQL, Oracle, ODBC, Microsoft SQL Server y la capacidad de ejecutarlo en la mayoría de los sistemas operativos tales como UNIX, Linux, Mac OS X, y Windows permite que cada vez se vuelva más popular dentro de los servidores Web. En la tabla que se muestra a continuación (Tab. 3.2) se presentan las ventajas y desventajas de este lenguaje de acuerdo a su compatibilidad con otros programas y las funciones permitidas por el mismo.

Ventajas	Desventajas
Es un lenguaje multiplataforma	No posee abstracción de base de datos estándar, sino bibliotecas especializadas
Capacidad de conexión con la mayoría de los manejadores de base de datos en la actualidad y gran conectividad con MySQL	No hay una internacionalización o Unicode
Expansión del potencial utilizando enorme cantidad de módulos o extensiones.	Debido a que es dinámico, no puede ser compilado y es muy difícil de optimizar
Software libre por lo que es una alternativa de acceso común.	Favorece a la creación de código desordenado y complejo de mantener
Técnicas de programación orientada a objetos	Metodología de programación diseñada totalmente por el programador.
Manejo de excepciones	
No requiere definición de tipos de variables	

Tab. 3.2 Ventajas y desventajas de PHP

MySQL

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario desarrollado por una subsidiaría de Sun Microsystems como software libre en un esquema de licenciamiento dual, y estpa desarrollado en su mayoría en ANSI C.

Existen varios programas que permiten a aplicaciones escritas en diversos programas de programación acceder a las bases de datos MySQL, como lo son C, C++, C#, Pascal, Delphi, Java, entre los más importantes, utilizando cada uno de ellos una API específica diferente.

Es muy utilizado en aplicaciones Web que contienen grandes bases de datos, en diferentes plataformas. Su aplicación Web está muy ligada a PHP. MySQL es una base de datos muy rápida en la lectura, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones Web de baja concurrencia en la modificación de datos se comporta de manera totalmente diferente haciéndolo ideal para este tipo de aplicaciones.

Hay tres tipos diferentes de compilación para MySQL:

- Estandar; son recomendados para la mayoría de los usuarios y utiliza un motor de almacenamiento de datos InnoDB.
- Max; se incluyen características adicionales, las cuales no han sido probadas en su totalidad y normalmente no son necesarias salvo en casos específicos.
- MySQL-Debug; tiene información extra de compilación, por lo tanto no puede ser usado en sistemas en producción porque afecta notablemente el rendimiento del equipo.

Empresas y usuarios a través del mundo son usuarios de los servicios y las características que ofrece MySQL para gestionar sus bases de datos debido a la gran adaptabilidad y compatibilidad de los servicios. Algunos de estos usuarios son:

- Pacific Soft, Amazon.com, Cox Communications, Craigslist, CNET Networks, Digg, flickr, Friendster, Google, Joomla!, LiveJournal, NASA, NetQOS, Nokia, Omniture, Palcasa, RightNow, Sabre, Slashdot, Universidad de Piura | Campus Lima, Wikipedia, WordPress, Yahoo!, entre los más importantes.

Aplicación Web

El desarrollo de esta aplicación Web que ofrece sus servicios de una manera inalámbrica tiene como fin el brindar a cada usuario dentro del supermercado, previamente registrado y autorizado para accesar al portal, la información necesaria y suficiente para hacer de su compra una experiencia que le permita obtener la mayor cantidad de información en el menor tiempo posible. El usuario, mediante esta aplicación, es capaz de obtener informes acerca de los productos que más se adapten a sus necesidades así como de la rápida y eficaz localización de los productos que se encuentran a la venta en ese instante. Al mismo tiempo la búsqueda de los productos mediante el nombre, precio, departamento ó código único de producto, dentro de la base de datos, permitirá al usuario organizar de una mejor manera el tiempo destinado a esta actividad.

De la misma manera se programó un evento sobre *ActioScript* de tal manera que cada vez que se cambie de pantalla dentro de la aplicación, el color alrededor de la pantalla que será el mismo que el color del menú, dando más vistosidad a la información publicada.

A continuación se detalla con más precisión las funciones específicas de esta aplicación:

Pantalla de "Registro"

La identificación de cada usuario dentro del supermercado es una opción que se utiliza para brindar atención personalizada. La aplicación cuenta con una pantalla de registro en donde el usuario tendrá que introducir el nombre de usuario así como la contraseña que éste haya seleccionado para la relación, como se muestra en la Fig. 3.24. Una vez que se hayan introducido a los campos correspondientes los datos el sistema verificará la veracidad de los mismos. Este proceso de chequeo se hace mediante la obtención de los

parámetros introducidos por el usuario, una vez hecho esto, el sistema tomará esos datos y los buscará en la base de datos de usuarios en donde deberá encontrar los mismos valores. Si los datos son correctos y son validados por el sistema, entonces se procederá a desplegar la pantalla de bienvenida al usuario. En caso contrario, en el cual los valores introducidos por el usuario no concuerden con los de la base de datos, se mostrará una pantalla en la cual se indique que el acceso fue denegado, no permitiendo el acceso de esta persona al portal.

El chequeo de los datos se obtuvo mediante la función POST en el programa PHP. Cuando el usuario teclea los datos y da click en botón "Enviar" los datos son extraídos por el programa como los valores de usuario y contraseña, respectivamente. Una vez hecho esto cada uno de los datos se asocia con una columna de la base de datos que contenga el mismo campo, de tal manera que para cada usuario se verifique la posición en la base de datos en la cual los dos campos son idénticos.

Si ambos campos coinciden se tomará el valor del renglón en el cual la información coincidió, este valor será entonces diferente de cero, lo que permitirá verificar que el usuario está dado de alta en la base de datos y tiene acceso. Para el caso en el que ninguno de los campos coincide, el valor obtenido como resultado de la búsqueda será igual a cero, identificando que el usuario no está registrado en la base de datos.

Una vez que el usuario es autorizado para tener acceso al portal se buscará, en la misma base de datos de usuarios registrados, el género del cliente. Estos datos permitirán mostrar en la pantalla de bienvenida además de la foto del usuario, información personalizada con respecto a los productos ofrecidos en el supermercado según el género del cliente. El código de esta sección se encuentra en el anexo IXa

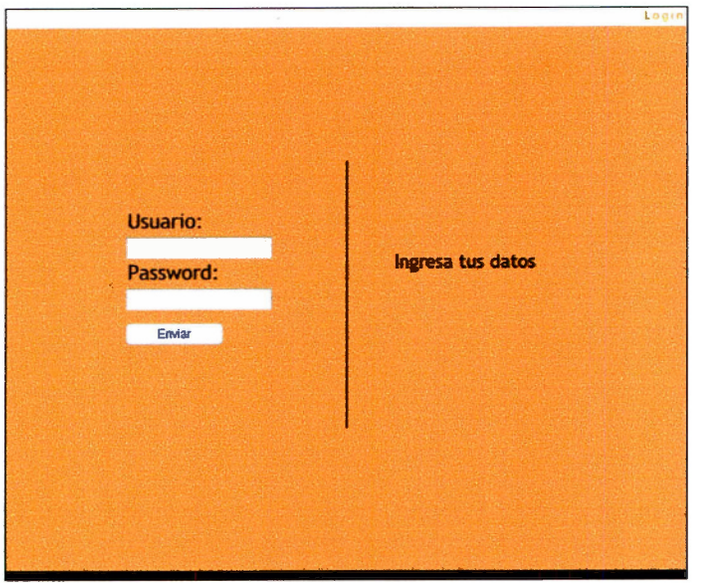


Fig. 3.24 Pantalla de registro de usuario

Pantalla de bienvenida

En esta pantalla se despliega la información obtenida una vez que se ha autorizado el acceso del usuario al portal, como los son la foto y la publicidad personalizada al usuario, esta última dependiendo del género del mismo. Es aquí en donde se mostrara la primera pantalla dentro del portal Web, misma sobre la cual se empezará a mostrar la información relacionada con los productos dentro del supermercado como se puede observar en la Fig. 3.25. En la parte superior derecha se muestra la fotografía del usuario, seguida del lado derecho por un mensaje de bienvenida. En la parte inferior de la pantalla se muestra la publicidad de los productos la cual es mostrada de forma dinámica, permitiendo así que se vayan mostrando uno por uno los artículos.

En la parte inferior derecha se muestra un botón de "Entrar", el cual permite al cliente accesar a la segunda parte de la aplicación Web, en donde se muestra la información relacionada con la localización, precio, cantidad y marca

de cada uno de los productos que previamente se han dado de alta para ser mostrados como posibles alternativas de compra para el consumidor.

La presentación de la imagen del usuario se logró por medio de una conexión entre la base de datos de usuarios con un arreglo de imágenes, en el cual a cada imagen le corresponde un valor, el mismo que estará relacionado con el renglón en el cual se encuentre el usuario dentro de la base de datos. Para desplegar la publicidad de manera dinámica se utilizaron máscaras de diseño, las cuales son una herramienta de Flash. Sobre esta máscara se pasan solo los objetos que se quieren mostrar en determinado instante de tiempo, logrando así un efecto de aparición y desvanecimiento de las imágenes. El anexo IXb.

Fig. 3.25 Pantalla de Bienvenida y muestra de las ofertas en el supermercado.

Pantalla "Departamentos"

La información presentada en esta pantalla está dirigida para que el usuario localice cada uno de los departamentos dentro del supermercado. Consta de un mapa general del supermercado, en el cual se despliegan cada uno de los departamentos en forma de renglones o columnas, los cuales pretender ser una visión superior de cada uno de los estantes contenidos dentro del supermercado. En el momento en que el usuario pasa el mouse sobre cualquiera estos renglones o columnas un texto mostrará el departamento al cual pertenecen y el texto no desaparecerá hasta que el usuario cambie por completo de área, como se muestra en la Fig. 3.26 y 3.27.

El concepto de máscaras de diseño su vuelve a retomar en esta pantalla dado que se cuenta con varios departamentos. Cada vez que el usuario coloque el mouse sobre alguno de los departamentos el texto relacionado a las coordenadas sobre las cuales fue definido el departamento, será mostrado. Cuando no exista ningún elemento seleccionado ninguno de los textos será mostrado y conforme el usuario se desplace a lo largo de la pantalla, las coordenadas serán actualizadas para mostrar en cada momento el texto indicado según el departamento o área en el que se encuentre el mouse. Esto fue programado en el anexo IXc.

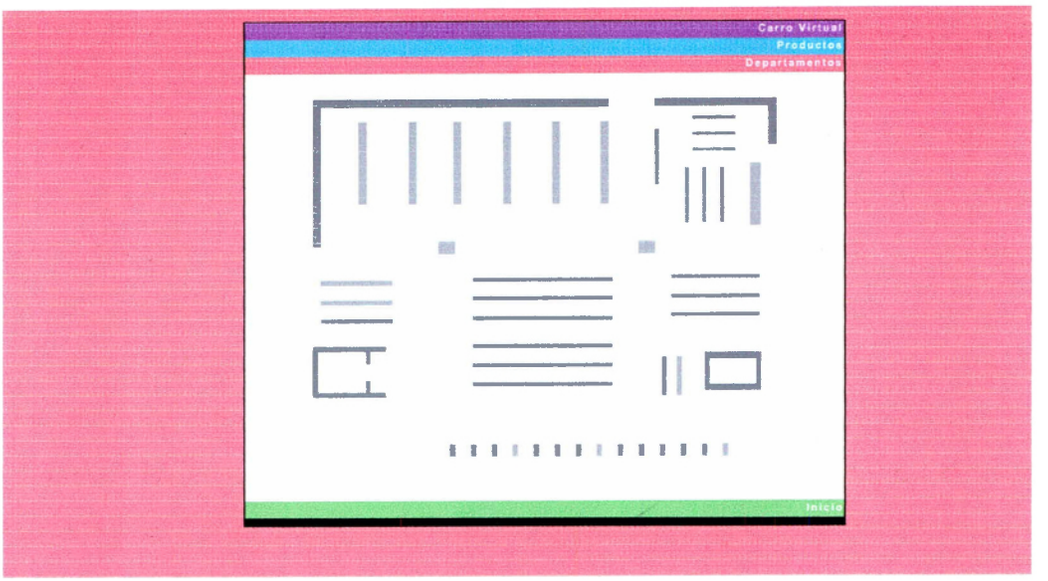


Fig. 3.26 Mapa del supermercado y cada uno de sus departamentos

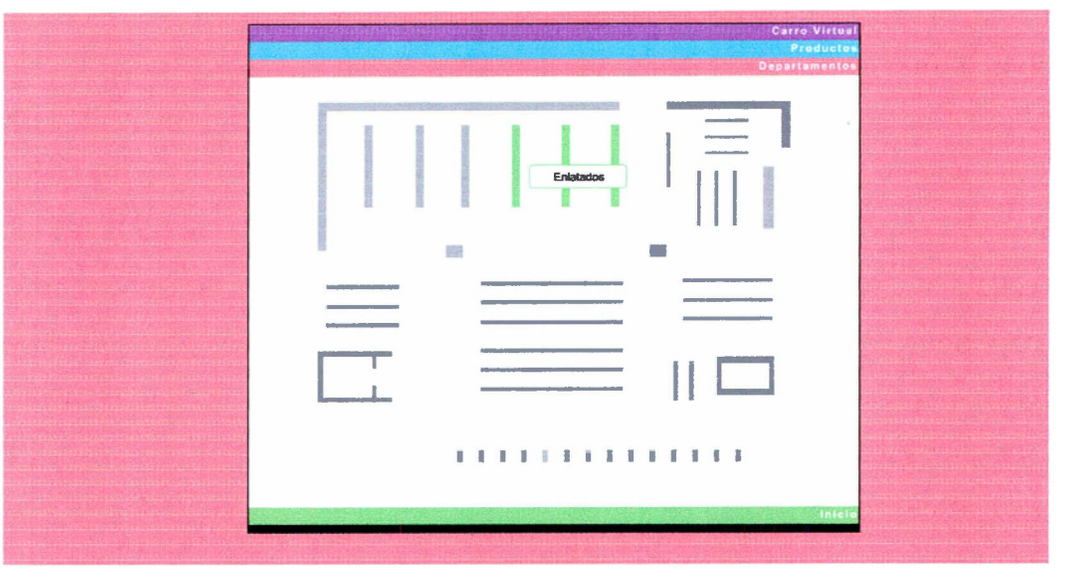


Fig.3.27 mapa del supermercado mostrando la localización del departamento de "Enlatados"

Pantalla "Productos"

Mostrar los productos que se encuentran a la venta, así como la cantidad, precio, marca y departamento en donde se encuentra cada uno de estos es la función principal de esta aplicación. Al usuario se le presenta una base de datos, en la cual puede visualizar todos los productos que estén dados de alta, da tal manera que le permita una decisión más exacta acerca de los productos que

pueden satisfacer sus necesidades, además de que se pretende que ahorre tiempo en la búsqueda innecesaria de los productos sin existencia en ese momento. También el usuario podrá ingresar el código único de producto, el cual deberá estar colocado sobre el mismo y tendrá que estar relacionado con el número de producto que se le asigno al momento de darlo de alta en la base de datos.

Esta aplicación permite que los productos que no sean marcados con el precio puedan ser seleccionados de forma autónoma por el usuario obteniendo las características más actualizadas del producto como se muestra en la Fig. 3.28, en donde la búsqueda se realizó para el producto "Gansito".

La obtención de todas las características de cada uno de los productos se hace por medio de código PHP que está ligado a la tabla construida sobre MySQL. Desde el principio el usuario tiene acceso para visualizar todos los productos contenidos en la base de datos, con cada una de sus características. Cuando el usuario introduce sobre el campo editable algún producto y presiona el botón buscar, la aplicación hace un escaneo del producto que se está solicitando, este a su vez lo procesa y lo compara con los datos que se tienen en la base de datos creada sobre MySQL. Cuando los datos introducidos junto con los datos de la tabla coinciden, el valor que tiene el producto en la tabla se almacena temporalmente.

Con este número se hace una nueva búsqueda en la base de datos pero ahora no para comparar los datos, sino para la obtención de las características del producto encontrado de tal manera que se muestren sobre la pantalla que se está visualizando. En dado caso en que el producto que está solicitando el usuario no se encuentre dado de alta en la base de datos no se mostrará ningún tipo de información en la pantalla. El anexo IXd muestra el código para sta sección.

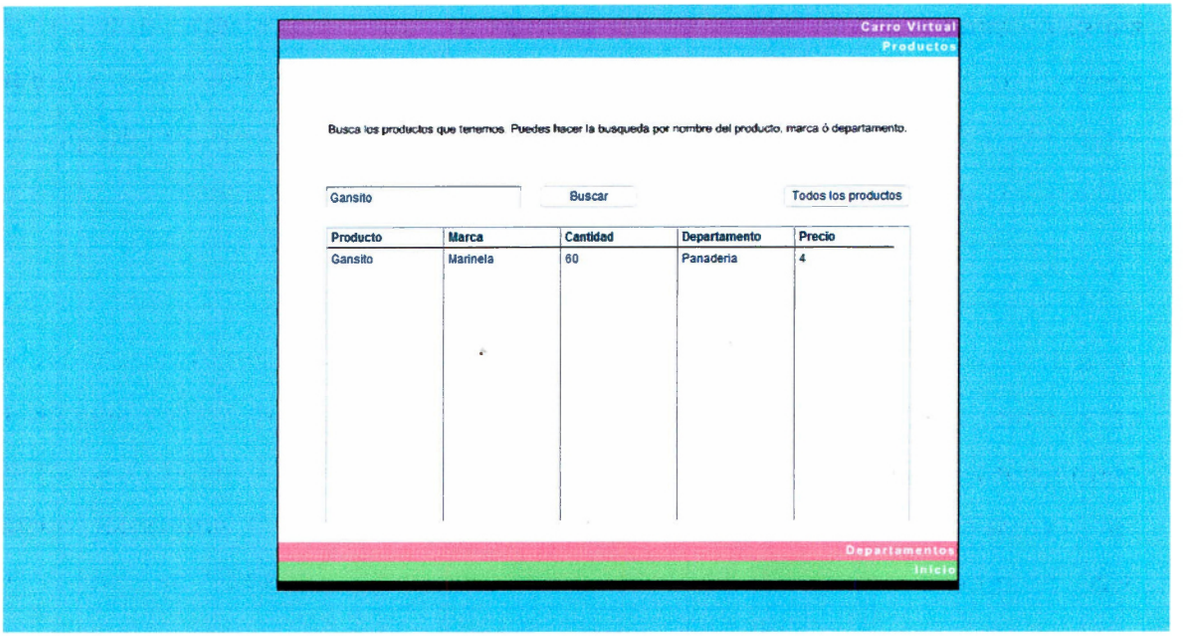


Fig. 3.28 Búsqueda del producto "Gansito"

Aplicación "Carrito Virtual"

En esta aplicación se utilizó el mapa del supermercado en conjunto con una imagen de un carrito de supermercado, para hacer una pequeña simulación que no tendrá ningún efecto real sobre el movimiento del carrito pero queda como trabajo futuro para aquellos que deseen realizar la conexión carrito real – carrito virtual, para facilitar el uso del carrito dentro del supermercado, de tal manera que los usuarios puedan elegir entre manejar al carrito por medio unas teclas de movimiento o preferir que el carrito los pueda seguir.

En esta simulación se presenta al usuario el carrito de supermercado siempre desde el mismo punto de inicio. En la parte superior de la pantalla se muestran 2 botones, uno en color verde que hace referencia a que la aplicación se encuentra encendida y otro en rojo que señala que el carrito no está habilitado para ser manejado por medio de esta aplicación. Si se presiona el botón verde entonces el usuario será capaz de mover dentro de la simulación al carrito por medio de las teclas de dirección que se encuentran en la computadora. El carrito puede ser movido dentro de todo el mapa del

supermercado, dando la impresión de que se mueve en relación al supermercado, como se muestra en la Fig. 3.29. Cuando el usuario presiones el botón rojo el carrito dejará de responder a las acciones que le indique el usuario por medio de las teclas de dirección de la computadora.

Esta aplicación fue realizada por medio de Flash, utilizando parámetros de conexión entre el código de la computadora y el movimiento dentro de la animación. Se basa en el continuo análisis de las teclas de dirección, permitiendo saber cuando alguna de estas fue presionada, la dirección y el tiempo que se mantuvo en ese estado con el fin de permitir a la simulación hacer ese recorrido en forma gráfica, siempre y cuando la aplicación se encuentre encendida. El anexo IXe muestra la programación del carrito virtual.

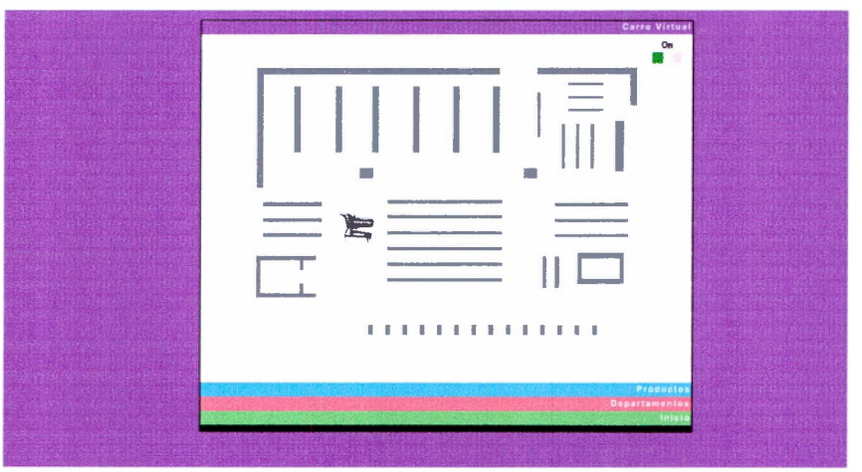


Fig. 3.29 simulación del movimiento del carrito a través del supermercado

Capítulo 4
Conclusiones del proyecto

Conclusiones

Después del trabajo realizado en este proyecto concluimos que se cumplieron los objetivos planteados al inicio del mismo, ya que se logró realizar la programación necesaria que permitiera al robot seguir a un usuario, identificado con dos colores, en un entorno real de supermercado. Esto se logró mediante la implementación de distintas etapas, cada una con un objetivo particular.

Las primeras etapas relativas al seguimiento de rutas preestablecidas por coordenadas con y sin evasión de obstáculos, seguimiento por medio de sonares, fueron desarrolladas con el objetivo de tener dominio de las funciones que serían útiles para lograr el seguimiento de un usuario, concretamente el manejo adecuado de los sonares y movimiento del robot. La etapa de seguimiento por sonar brindó la posibilidad de conocer la manera de avanzar, retroceder, girar, etc. del robot al seguir un objetivo. Mostró las necesidades de control de distancia y velocidad entre el usuario y el robot para evitar colisiones entre estos. Esta etapa fue fundamental para el desarrollo de las ultimas, esto debido a que la navegación del robot, se basa precisamente en las mediciones del sonar para conocer distancias y detectar posibles obstáculos para evitar colisiones, asimismo, da la posibilidad de que el robot sea capaz de detenerse y mantener una distancia con el usuario cuando este se detenga a tomar algún articulo de los estantes del supermercado o cuando se acerque a depositarlo en la canastilla.

El desarrollo propiamente del seguimiento por color se dio en las ultimas dos etapas, donde se logro un seguimiento primero de un color, y finalmente dos colores para lograr mayor confiabilidad en la identificación.

El uso de un segundo color, reduce la probabilidad de que la identificación de color sea errónea debido a los colores del entorno, sobre todo considerando la cantidad de colores existentes en un supermercado, lo que garantiza que en todo momento se estará siguiendo el objetivo adecuado. Aunado a la validación de las matrices de ambos colores, se evita que el seguimiento se lleve a cabo cuando se identifique un solo color de los dos preestablecidos.

Es importante mencionar que el seguimiento del usuario es altamente dependiente a la iluminación del lugar en el cual se implemente este prototipo dado que los rangos de los colores utilizados para la segmentación no son dinámicos ni adaptables una vez que se inicie el programa de segmentación de color, es decir, el color que será segmentado para el seguimiento será elegido únicamente al activar el prototipo, en caso de que la iluminación se vea alterada existe la posibilidad de que los colores presenten tonalidades diferentes, que se encuentren fuera de los rangos establecidos en un inicio; esto derivaría en una no identificación de un objetivo a seguir.

La implementación de las funciones de *opening y closening* en la localización del centroide de un objetivo es otro factor que aumenta considerablemente la confiabilidad del seguimiento, dado que permite la disminución del error eliminando posibles fuentes de color ajenas y aumentando la probabilidad de que el mismo sea el esperado. Esto es de gran ayuda cuando existen variaciones de color poco significativas, dado que por las condiciones de luz, es posible identificar una pequeña parte del objetivo, dichas funciones arrojaran una imagen con un objeto detectado de mayor tamaño ayudando a la identificación y por tanto, al seguimiento.

A pesar del desarrollo logrado en la etapa de seguimiento de coordenadas con evasión de obstáculos, para el prototipo final no se pudo implementar dicha evasión, ya que al manejar un algoritmo de seguimiento de un objetivo móvil dependiente a la visión podría propiciar la pérdida del objetivo. No obstante, la función desarrollada para el giro del robot en su propio eje permite localizar al usuario en caso de que este haya realizado un movimiento súbito como un cambio de dirección o la interposición de otro objeto. Más aún, el hecho de hacer el giro en la dirección del último centroide localizado redujo el tiempo de localización en los casos en los que el objetivo se perdía al realizar una vuelta, sobretodo en los cambios de pasillo.

También se hicieron consideraciones dentro de la identificación del color para reducir el tiempo del procesamiento de las imágenes. Estas consideraciones comprendieron elementos tales como realizar la definición del color o colores a seguir fuera del ciclo de segmentación de la imagen y la reducción de la tasa de muestro al momento de realizar el pixel *labeling*. Otros factores considerados ajenos a la programación, fueron relativos al hardware elegido. La elección de una cámara adecuada fue esencial para obtener de forma rápida imágenes de un tamaño que se procesara rápidamente y que además tuviera la calidad adecuada para llevar a cabo la identificación de los colores. El uso de una computadora con características adecuadas que facilitaran el procesamiento también fue importante, en este caso, se uso una computadora con un procesador AMD Turion 64x2 de 1.94GHz, con una memoria RAM de 3GB y una memoria de video de 512MB. El uso de esta maquina dio tiempos de procesamiento de hasta 0.55 segundos, mientras que el uso de otras computadoras de menores capacidades daban tiempos superiores a un segundo.

Trabajo Futuro

Es importante mencionar que el trabajo futuro que puede ser realizado en posteriores etapas de este proyecto es el desarrollo de un criterio de seguimiento que no requiera línea de vista como podría radio frecuencia ya que así se anularía la necesidad de que el usuario se encuentre en el campo de visión de la cámara siendo más sencilla una evasión de obstáculos.

Por otro lado, para poder ampliar la aplicación del prototipo para que pueda ser usado por cualquier persona, se podrí desarrollar un menú que al activar el dispositivo permita al usuario elegir la velocidad que mejor se adapte al mismo. Esto conllevaría un manejo más detallado del deslizamiento y error que se presente en el arranque y frenado del robot para evitar colisiones.

Otro punto que representaría un avance importante en el desarrollo de este proyecto sería el de programar un sistema dinámico de segmentación de colores, es decir, que pueda cambiar el rango de los tonos a medida que cambie la iluminación en el lugar. Esto se podría realizar conectando una fotoresistencia que capte los niveles de iluminación del entorno y ordene una nueva declaración del matiz (H) y la saturación (S).

Como trabajo futuro para la aplicación inalámbrica, se podría implementar el despliegue de información como publicidad en tiempo real correspondiente al pasillo o zona en la que se encuentre el carrito se supermercado.

Referencias

[1] Instituto Nacional de Estadística, Geografía e Informática, "Censo Nacional de Población" www.inegi.gob.mx.

[2] Universidad Pontificia Bolivariana, "Robótica Móvil Modular: Plataforma de Trabajo en Robótica Móvil y Colectiva", http://espanol.geocities.com/robottotem/Historia_robotica.htm.

[3] Associated Press, "Robot shopping carts follow you around", August 2006, http://www.msnbc.msn.com/id/14282876/.

[3] L. Hauenstein "Ian's Horswill's Polly" Mobile Robotics, Abril 2003.

[4] Autonomous Systems group "Rhino Intelligent Autonomous Systems" University of Bonn, Marzo 2008.

[5] D. Rodriguez-Losada, F. Matia, et al. "Guido, the Robotic SmartWalker for the frail visually impaired" Universidad Politecnica de Madrid. Madrid 2007

[6]Ubergizmo, "Shopping Navis Wagon equipped with LCD screen and RFID reader", Enero 2005, http://www.ubergizmo.com/15/archives/2006/01/shopping_navis_wagon_equipped_with_lcd_screen_and_rfid_reader.html

[7] V. Kulyukin, C Gharpure, John Nichelson, *RoboCart:Toward Robot-Assisted navigation of Grocery Stores by the Visually Impaired.*

[8] D. FU, K. J. Hammond, Michael J. Swain, *Action and percetion in Man-made Envirnoments,* pages 464-469

[9] A. Ollero, *Robótica: Manipuladores y robots móviles.* Madrid: Pearson Prentice Hill, 2006

[10] R. Pallás-Areny, *Sensores y Acondicionadores de Señal.* Barcelona: Marcombo; México: Alfaomega, 2007.

[11] A. F. Orozco *Sensores, Métodos y Técnicas para la Navegación de Robots Móviles Autónomos.* Colombia: Universidad Pontificia Bolivariana.

[12] K.S. FU, R.C. Gonzalez, C.S. G. Lee. *Robótica: Control, detección, visión e inteligencia.*

[13] J. M. Gómez, A. Ollero, A. J. García, *Teleoperación y Telerobótica*. México: Prentice Hall, 2006.

[14]U. Nehmzow, *Mobile Robotics: A practical Introduction*. London; New York: Springer, c2000

[15] S. Ortigoza, J.R. García Sánchez, V.R Barrientos and M.A. Molina. *Una panorámica de los robots móviles*

[16] ANSI/IEEE Std 802.11, 1999 Edition. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications

[17] ActivMedia Robotics. Pioneer 3 & Pioneer 2 H8-Series Operations Manual. p. 6.

[18] ActivMedia Robotics. Pioneer 3 & Pioneer 2 H8-Series Operations Manual. p. 14.

[19] R. Silva, J. R. García, V. R. Barrientos, M. A. Molina, *Una panoramica de los robots moviles*. Universidad Rafael Belloso Chacín, 2007

[20] http://verona.fi-p.unam.mx/fardi/pagina/VISONCOM.htm, junio 2003.

[19] Vijayan, Smitha"An HSV/RGBA colour picker" Noviembre, 2008, http://www.codeproject.com/KB/miscctrl/CPicker.aspx

[22] Central Maryland Photographers' Guild Noviembre 2008, http://www.cmpg.org/site/2005/07/26/an-introduction-to-color-space-and-color-calibration/

[23] Martínez Nistal, Angel. "PROCESO DE IMÁGENES" Noviembre 2008, http://wellpath.uniovi.es/es/contenidos/seminario/tutorialpdi/html/binaria.htm

[24] Kapp, S, 802.11a. More bandwidth without the wires, 2002.

[25]. Josip Lorincz, Dinko Begusic, Physical Layer Analysis of Emerging IEEE 802.11 in WLAN Standard, University ofSplit, FESB-Split, Croatia, 2006.

[26]. LAN/MAN Committee of the IEEE Computer Society, IEEE802.11 Wireless LAN Medium Access (MAC) and Physical Layer (PHY), IEEE Std 802.11™, 2007.

[27]. López Ortiz Francisco, El estándar IEEE 802.11 Wireless LAN, disponible en http://greco.dit.upm.es/~david/TAR/trabajos2002/08-802.11-Francisco-Lopez-Ortiz-res.pdf.

[28]. Redes Wi-Fi, "TELEFÓNICA ONLINE", octubre de 2008, http://www.telefonicaonline.com/on/io/es/atencion/tutoriales_articulos/wifi/escenarios.htm.

[29]. Guido R. Hiertz, Sebastian Max, Rui Zhao, Dee Denteneer, Lars Berlemannt, Principles of IEEE 802.11s, Chair of Communication Networks, Faculty 6, RWTH Aachen University, Aachen, Germany, †Philips, Eindhoven, The Netherlands, Swisscom, Bern, Switzerland, 2006.

[30]. Information Centre, "ISO/IEC Information Centre", octubre de 2008, http://www.standardsinfo.net/info/livelink/fetch/2000/148478/6301438/index.html.

[31]. Radiofrecuencia, "Dreamweaver basics overview", septiembre de 2008, http://www.infoab.uclm.es/labelec/Solar/Comunicacion/Redes_inalambricas/html/02Basics1.html.

[32]. Alexandre V. Garmonov, QoS-Oriented Intersystem Handover Between IEEE 802.11b and Overlay Networks, IEEE Transactions on vehicular technology, Vol. 57, No. 2, 2008.

[33]. WiFi – Propagacion, "Fabio Moreno - WiFi - Propagacion", septiembre de 2008, http://amentis81.googlepages.com/wifi-propagacion.

[34] Basics of color based computer vision implemented in Matlab, Dr.ir. M.J.G. van de Molengraft and prof.dr.ir. M. Steinbuch, Technische Universiteit, June 2005.

[35] H. M. Deitel y P. J. Deitel, *Cómo programar en C/C++ y Java*, México: Pearson Educación, 2004

[36]. Taller de PHP: Envío de datos de un formulario, "Taller de PHP: Programa - elWebmaster.com:", agosto de 2008, http://www.elwebmaster.com/talleres/taller-de-php-envio-de-datos-de-un-formulario

[37]. Taller de PHP: Base de datos, "Taller de PHP: Programa - elWebmaster.com:", agosto de 2008, http://www.elwebmaster.com/talleres/taller-de-php-base-de-datos

[38]. Taller de PHP: MySQL-phpMyAdmin, "Taller de PHP: Programa - elWebmaster.com:", agosto de 2008, http://www.elwebmaster.com/editorial/taller-de-php-mysql-phpmyadmin

Anexos

Anexo I Características de Wi- Fi

Nivel de Acceso al Medio (MAC)

Los diferentes métodos de acceso de IEEE 802.11 están diseñados según el modelo OSI y se encuentran ubicados en el nivel físico y en la parte inferior del nivel de enlace o subnivel MAC. Además, la capa de gestión MAC controlará aspectos como sincronización y los algoritmos del sistema de distribución, que se define como el conjunto de servicios que precisa o propone el modo infraestructura.

- Descripción Funcional MAC.

La arquitectura MAC del estándar 802.11 se compone de dos funciones básicas:

- La función de coordinación puntual (PCF)
- La función de coordinación distribuida.

- DFC Función de Coordinación Distribuida

Definimos *función de coordinación* como la función que determina, dentro de en conjunto básico de servicios (BSS), cuándo una estación puede transmitir y/o recibir tramas de datos de protocolo a nivel MAC a través del medio inalámbrico. En el nivel inferior del subnivel MAC se encuentra la función de coordinación distribuida y su funcionamiento se basa en técnicas de acceso aleatorias de contienda por el medio. El tráfico que se transmite bajo esta funcionalidad es de carácter asíncrono ya que estas técnicas de contienda introducen retardos aleatorios y no predecibles no tolerados por los servicios síncronos. Las características de DFC las podemos resumir en estos puntos:

- Utiliza MACA (CSMA/CA con RTS/CTS) como protocolo de acceso al medio. Necesario reconocimientos ACKs, provocando retransmisiones si no se recibe.
- Usa campo Duration/ID que contiene el tiempo de reserva para transmisión y ACK. Esto quiere decir que todos los nodos conocerán al escuchar cuando el canal volverá a quedar libre.
- Implementa fragmentación de datos.
- Concede prioridad a tramas mediante el espaciado entre tramas (IFS).
- Soporta Broadcast y Multicast sin ACKs.

Protocolo de Acceso al medio CSMA/CA y MACA

El algoritmo básico de acceso a este nivel es muy similar al implementado en el estándar IEEE 802.3 y es el llamado CSMA/CA (Carrier Sense Multiple Access / Collision Advoidance). Este algoritmo funciona de la siguiente manera:

- Antes de transmitir información una estación debe examinar el medio, o canal inalámbrico, para determinar su estado (libre / ocupado).
- Si el medio no está ocupado por ninguna otra trama la estación ejecuta una espera adicional llamada *espaciado entre tramas* (IFS).
- Si durante este intervalo temporal, o bien ya desde el principio, el medio se determina ocupado, entonces la estación debe esperar hasta el final de la transacción actual antes de realizar cualquier acción.
- Una vez finaliza esta espera debida a la ocupación del medio la estación ejecuta el llamado algoritmo de Backoff, según el cual se determina una espera adicional y aleatoria escogida uniformemente en un intervalo llamado *ventana de contienda* (CW). El algoritmo de Backoff nos da un número aleatorio y entero de ranuras temporales (slot time) y su función es la de reducir la probabilidad de colisión que es máxima cuando varias estaciones están esperando a que el medio quede libre para transmitir.

- Mientras se ejecuta la espera marcada por el algoritmo de Backoff se continúa escuchando el medio de tal manera que si el medio se determina libre durante un tiempo de al menos IFS esta espera va avanzando temporalmente hasta que la estación consume todas las ranura temporales asignadas. En cambio, si el medio no permanece libre durante un tiempo igual o superior a IFS el algoritmo de Backoff queda suspendido hasta que se cumpla esta condición.

Cada retransmisión provocará que el valor de CW, que se encontrará entre CWmin y CWmax se duplique hasta llegar al valor máximo. Por otra parte, el valor del slot time es 20seg. Sin embargo, CSMA/CA en un entorno inalámbrico y celular presenta una serie de problemas que intentaremos resolver con alguna modificación. Los dos principales problemas que se pueden detectar son:

- Nodos ocultos. Una estación cree que el canal está libre, pero en realidad está ocupado por otro nodo que no escucha.
- Nodos expuestos. Una estación cree que el canal está ocupado, pero en realidad está libre pues el nodo al que oye no le interferiría para transmitir a otro destino.

La solución que propone 802.11 es MACA o MultiAccess Collision Avoidance. Según este protocolo, antes de transmitir el emisor envía una trama RTS (Request to Send), indicando la longitud de datos que quiere enviar. El receptor le contesta con una trama CTS (Clear to Send), repitiendo la longitud. Al recibir el CTS, el emisor envía sus datos. Los nodos seguirán una serie de normas para evitar los nodos ocultos y expuestos:

- Al escuchar un RTS, hay que esperar un tiempo por el CTS
- Al escuchar un CTS, hay que esperar según la longitud

La solución final de 802.11 utiliza MACA con CSMA/CA para enviar los RTS y CTS.

- Espaciado entre tramas IFS

El tiempo de intervalo entre tramas se llama IFS. Durante este periodo mínimo, una estación estará escuchando el medio antes de transmitir. Se definen cuatro espaciados para dar prioridad de acceso al medio inalámbrico. Enseguida se muestran de los más cortos a los más largos:

- SIFS (Short IFS). Este es el periodo más corto. Se utiliza fundamentalmente para transmitir los reconocimientos. También es utilizado para transmitir cada uno de los fragmentos de una trama. Por último, es usado por el PC o Point Control para enviar testigo a estaciones que quieran transmitir datos síncronos.
- PIFS (PCF). Es utilizado por las estaciones para ganar prioridad de acceso en los periodos libres de contienda. Lo utiliza el PC para ganar la contienda normal, que se produce al esperar DIFS.
- DIFS (DCF). Es el tiempo de espera habitual en las contiendas con mecanismo. MACA. Se utiliza para el envío de tramas MAC MPDUs y tramas de gestión MPDUs.
- EIFS (Extended IFS). Controla la espera en los casos en los que se detecta la llegada de una trama errónea. Espera un tiempo suficiente para que le vuelvan a enviar la trama o espera otra solución [26].

Conocimiento del medio

Las estaciones tienen un conocimiento específico de cuando la estación, que en estos momentos tiene el control del medio porque está transmitiendo o recibiendo, va a finalizar su periodo de reserva del canal. Esto se hace a través de una variable llamada NAV (Network Allocation Vector) que mantendrá una predicción de cuando el medio quedará liberado. Tanto al enviar un RTS como al recibir un CTS, se envía el campo Duration/ID con el valor reservado para la

transmisión y el subsiguiente reconocimiento. Las estaciones que estén a la escucha modificarán su NAV según el valor de este campo Duration/ID. En realidad, hay una serie de normas para modificar el NAV, una de ellas es que el NAV siempre se situará al valor más alto de entre los que se disponga [27].

PFC Función de Coordinación Puntual

Por encima de la funcionalidad DCF se sitúa la función de coordinación puntual, PCF, asociada a las transmisiones libres de contienda que utilizan técnicas de acceso deterministas. El estándar IEEE 802.11, en concreto, define una técnica de interrogación circular desde el punto de acceso para este nivel. Esta funcionalidad está pensada para servicios de tipo síncrono que no toleran retardos aleatorios en el acceso al medio. En la Fig.2.1.24 mostramos la relación entre estos dos modos de operación. Estos dos métodos de acceso pueden operar conjuntamente dentro de una misma celda o conjunto básico de servicios dentro de una estructura llamada *supertrama*. Una parte de esta *supertrama* se asigna al periodo de contienda permitiendo al subconjunto de estaciones que lo requieran transmitir bajo mecanismos aleatorios. Una vez que finaliza este periodo el punto de acceso toma el medio y se inicia un periodo libre de contienda en el que pueden transmitir el resto de estaciones de la celda que utilizan técnicas deterministas [32].

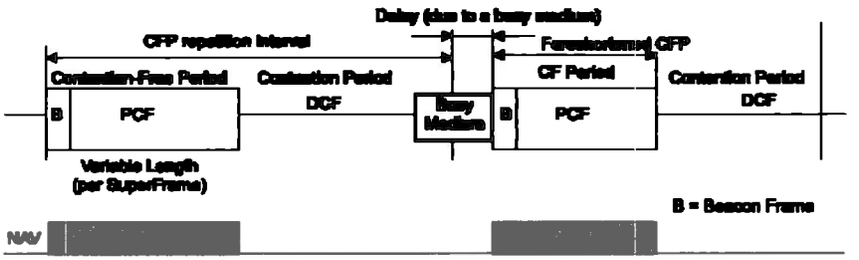


Fig. 2.1.24 Tramas dentro del funcionamiento PFC [27].

Un aspecto previo a comentar el funcionamiento de PFC es que es totalmente compatible con el modo DFC, observándose que el funcionamiento es transparente para las estaciones. De esta manera, una estación se asociará

(se dará de alta en un modo infraestructura) de modo que pueda actuar en el periodo CFP, declarándose como CFPoll habilitada, o por el contrario, se situará su NAV según las indicaciones del punto de coordinación.

Existe un nodo organizador o director, llamado punto de coordinación o PC. Este nodo tomará el control mediante el método PIFS, y enviará un CF-Poll a cada estación que pueda transmitir en CFP, concediéndole poder transmitir una trama MPDU. El PC mantendrá una lista Poll habilitada donde tendrá todos los datos de las estaciones que se han asociado al modo CF-Poll habilitada. La concesión de transmisiones será por riguroso listado y no permitirá que se envíen dos tramas hasta que la lista se haya completado.

El nodo utilizará una trama para la configuración de la supertrama, llamada Beacon, donde establecerá una CFRate o tasa de periodos de contienda. Pese a que el periodo de contienda se puede retrasar por estar el medio ocupado, la tasa se mantendrá en el siguiente periodo con medio libre. Como podemos observar, la transmisión de CF-Polls espera un tiempo SIFS.

También podemos ver que si una estación no aprovecha su CF-Poll se transmite a la siguiente en el listado de Poll habilitada. Las estaciones que no usen el CF, situarán su NAV al valor del final del CF y luego lo resetearán para poder modificarlo en el periodo de contienda en igualdad de condiciones.

Un problema importante que podemos encontrarnos en el espaciado de redes wireless ocurrirá cuando varios sistemas con coordinación puntual compartan una tasa CFRate semejante. Una solución suele ser establecer un periodo de contienda entre PCs para ganar el medio esperando un tiempo DIFS+ BackOff (1-CWmin).

Formato de las tramas MAC

Las tramas MAC contienen los siguientes componentes básicos:

- Una cabecera MAC, que comprende campos de control, duración, direccionamiento y control de secuencia.
- Un cuerpo de trama de longitud variable, que contiene información específica del tipo de trama.
- Un secuencia checksum (FCS) que contiene un código de redundancia CRC de 32 bits.

Además se pueden clasificar en tres diferentes tipos:

- Tramas de datos.
- Tramas de control. Los ejemplos de tramas de este tipo son los reconocimientos o ACKs, las tramas para multi-acceso RTS y CTS, y las tramas libres de contienda.
- Tramas de gestión. Como ejemplo podemos citar los diferentes servicios de distribución, como el servicio de Asociación, las tramas de Beacon o portadora y las tramas TIM o de tráfico pendiente en el punto de acceso.

Todas las tramas de la capa MAC tienen información de manera genérica y por lo regular están compuestas por los siguientes campos:

- Campo de control.
- Duration/ID. En tramas del tipo PS o Power-Save para dispositivos con limitaciones de potencia, contiene el identificador o AID de estación. En el resto, se utiliza para indicar la duración del periodo que se ha reservado una estación.
- Campos address1-4. Contiene direcciones de 48 bits donde se incluirán las direcciones de la estación que transmite, la que recibe, el punto de acceso origen y el punto de acceso destino.

- Campo de control de secuencia. Contiene tanto el número de secuencia como el número de fragmento en la trama que se está enviando.
- Cuerpo de la trama. Varía según el tipo de trama que se quiere enviar.
- FCS. Contiene el checksum.

Los campos de control de trama tienen el formato siguiente:

- Versión.
- Type/Subtype. Mientras tipo identifica si la trama es del tipo de datos, control o gestión, el campo subtipo nos identifica cada uno de los tipos de tramas de cada uno de estos tipos.
- ToDS/FromDS. Identifica si la trama si envía o se recibe al/del sistema de distribución.
- En redes ad-hoc, tanto ToDS como FromDS están a cero. El caso más complejo contempla el envío entre dos estaciones a través del sistema de distribución. Para ello situamos a uno tanto ToDS como FromDS.
- Más fragmentos. Se activa si se usa fragmentación.
- Retry. Se activa si la trama es una retransmisión.
- Power Management. Se activa si la estación utiliza el modo de economía de potencia.
- More Data. Se activa si la estación tiene tramas pendientes en un punto de acceso.
- WEP. Se activa si se usa el mecanismo de autenticación y encriptado.
- Order. Se utiliza con el servicio de ordenamiento estricto [29].

Direccionamiento en modo infraestructura

Como se ha comentado con anterioridad, el caso más complejo de direccionamiento se produce cuando una estación quiere transmitir a otra ubicada en otro BSS o sistema de servicios básicos.

En este caso los campos ToDS=FromDS=1 y las direcciones de cada uno de los componentes por los que pasa la trama toman el siguiente valor en la trama MAC, quedando la dirección 1 como el nodo destino, la dirección 2 será la del punto de acceso final, la dirección 3 sería la del punto de acceso origen y por último, la dirección 4 sería la del nodo origen.

Servicios del Sistema de Distribución. Asociación.

La especificación IEEE802.11 define el sistema de distribución como la arquitectura encargada de interconectar diferentes IBSS o redes inalámbricas independientes. El componente fundamental de este sistema de distribución es el punto de acceso, y además la especificación define lo que se llama "servicios de distribución", los cuales facilitan y posibilitan el funcionamiento en modo infraestructura. Se definirán servicios diferentes para cada componente, según se trate del punto de acceso o estación.

A continuación se mencionan los servicios más básicos que se implementan dentro de un punto de acceso y una estación.

- Distribución. Se encarga de llevar un paquete del punto de acceso de origen al de destino.
- Integración. Se encarga de la función de integración con otros sistemas IEEE802.x. Esto define el componente portal que se encargará de aspectos necesarios como re direccionamiento.
- Asociación. Servicio necesario para que una estación pueda adherirse al modo infraestructura y utilizar sus servicios.
- Re asociación. Consiste en el campo de punto de acceso al que se asocia la estación para adherirse al modo infraestructura. También se utiliza para modificar las características de la asociación.
- Autenticación y Des-autenticación. Proceso necesario para que la estación se pueda conectar a la wireless LAN y consiste en la identificación de la estación. Por lo tanto, el proceso de conexión, pasa por la autenticación previamente a la asociación.

- Privacidad. Este servicio utilizará WEP para el encriptado de los datos en el medio.
- Reparto de MSDUs entre estaciones. Este es el servicio básico de intercambio.

Algoritmo de Asociación Activa.

En el algoritmo de asociación activa se utilizarán las tramas de prueba y respuesta para mantenerse asociada a un punto de acceso que puede variar si tiene la condición de móvil.

El algoritmo consiste en los siguientes pasos:

- El nodo envía una trama de prueba (Probe)
- Los puntos de acceso alcanzados responden con una trama de respuesta (Response)
- El nodo seleccionará generalmente por nivel de señal recibida el punto de acceso al que desea asociarse, enviándole una trama de requerimiento de asociación
- El punto de acceso responderá con una respuesta de asociación afirmativa o negativa

La asociación activa implica que la estación continuará enviando este tipo de tramas y podrá provocar una reasociación en función de los parámetros de selección que él mismo utilice y defina [27], [26].

Subnivel de Gestión MAC

En la subcapa de gestión MAC se implementan las siguientes funciones:

- Sincronización.
- Gestión de potencia
- Asociación - Reasociación
- Utiliza el MIB o Management Information Base

Sincronización

La sincronización se consigue mediante una función de sincronización (TSF) que mantendrá los relojes de las estaciones sincronizados. Según el modo de operación se distinguirá el modo de funcionamiento. En el modo infraestructura, la función de sincronización recaerá en el punto de acceso, de tal manera que el punto de acceso enviará la sincronización en la trama portadora o Beacon y todas las estaciones se sincronizarán según su valor.

En el modo ad-hoc, el funcionamiento es más complejo. Por una parte, la estación que se encuentre en la red establecerá un intervalo de beacon, esto es, una tasa de transferencia de portadoras que permitan la sincronización. Sin embargo, en este caso, el control está distribuido y entre todas las estaciones se intentará mantener la sincronización. Para ello, toda esta estación que no detecte en un determinado tiempo de BackOff una trama de sincronización, enviará ella misma una trama de portadora para intentar que no esté sincronizada a la red.

Gestión de Potencia

Las estaciones en la red pueden adoptar un modo limitado de potencia. Este modo de funcionamiento implicará que la estación se "despertará" sólo en determinados momentos para conectarse a la red. Estas estaciones se denominan PS-STAs (Power Save Station) y estarán a la escucha de determinadas tramas como la de portadora y poco más. El control de este tipo de estaciones se llevará mediante el punto de acceso, que tendrá conocimiento de qué estación se ha asociado en este modo.

El punto de acceso mantendrá almacenados los paquetes que le lleguen con destino a los nodos limitados de potencia. Por tanto, el punto de acceso mantendrá un mapa de paquetes almacenados y los destinos a quienes tendrá que repartirlos o enviarlos. Cuando el punto de acceso decida enviarle el

paquete lo hará enviándole una trama TIM o Traffic Indication Map a la estación para que despierte en el próximo intervalo de portadora. De esta manera, estas estaciones recibirán la información con un desgaste mínimo de potencia [26].

Anexo II: Programa para seguimiento de rutas preestablecidas por medio de coordenadas sin evasión de obstáculos.

Programa adaptado de ejemplos de Aria

```
//Coordenadas sin evasión de obstáculos
#include "Aria.h"
 int main(int argc, char **argv)
 {                                                    //empieza inicialización de servicios, clases, etc
  Aria::init();
   ArArgumentParser parser(&argc, argv);
   parser.loadDefaultArguments();
   ArSimpleConnector simpleConnector(&parser);
   ArRobot robot;
   ArSonarDevice sonar;
   ArAnalogGyro gyro(&robot);
   robot.addRangeDevice(&sonar);

   //  Se agrega respuesta a teclas para usar Esc para salir del programa
   ArKeyHandler keyHandler;
   Aria::setKeyHandler(&keyHandler);
   robot.attachKeyHandler(&keyHandler);
   printf("Presiona Esc para salir \n");

   // Determinar coordenadas para movimiento
   ArActionGoto gotoPoseAction("goto");
   robot.addAction(&gotoPoseAction, 50);

   if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
   {
    Aria::logOptions();
    exit(1);
   }

   // Conectar al robot
   if (!simpleConnector.connectRobot(&robot))

   {
    printf("No se puede conectar al robot\n");
    Aria::exit(1);
   }
   robot.runAsync(true);
   robot.enableMotors();                              // Encender motores
   robot.comInt(ArCommands::SOUNDTOG, 0);
   bool first = true;
   int goalNum = 0;
   while (Aria::getRunning())
   {
    robot.lock();
    printf("x=");
    printf(" %4d", robot.getX(),"\n");
    printf("y=");
    printf(" %4d", robot.getY(),"\n");
```

```
        // Elegir coordenadas para movimiento
    if (first || gotoPoseAction.haveAchievedGoal())
    {
      first = false;
      goalNum++;

            if (goalNum > 8)
            {
                    goalNum = 1; // empezar en el objetivo 1
            }
```

Anexo IIa

```
      // determinar coordenadas para cada movimiento y dirigirse a dicho punto
      if (goalNum == 1)
        gotoPoseAction.setGoal(ArPose(4250, 0));

      else if (goalNum == 2)
        gotoPoseAction.setGoal(ArPose(4250, -1660));

      else if (goalNum == 3)
        gotoPoseAction.setGoal(ArPose(500, -1660));

      else if (goalNum == 4)
        gotoPoseAction.setGoal(ArPose(500, -3320));

      else if (goalNum == 5)
        gotoPoseAction.setGoal(ArPose(4250,-3320));

      else if (goalNum == 6)
        gotoPoseAction.setGoal(ArPose(4250,-4970));

      else if (goalNum == 7)
        gotoPoseAction.setGoal(ArPose(500, -4970));

      else if (goalNum == 8)
        gotoPoseAction.setGoal(ArPose(0, 0));

      else if (goalNum == 9)
        gotoPoseAction.setGoal(ArPose(19300, 13250));

      else if (goalNum == 10)
        gotoPoseAction.setGoal(ArPose(19300, 6500));

      else if (goalNum == 10)
        gotoPoseAction.setGoal(ArPose(0, 0));

          ArLog::log(ArLog::Normal, "Going to next goal at %.0f %.0f",
          gotoPoseAction.getGoal().getX(), gotoPoseAction.getGoal().getY());
    }
    robot.unlock();
  }

  Aria::shutdown();       // Desconectado del robot, apagar
  return 0;
}
```

Anexo III: Programa para seguimiento de rutas preestablecidas por medio de coordenadas y evasión de obstáculos.

Este programa realiza un cuadrado de 3 x 3 metros en base a las coordenadas, también es capaz de evadir obstáculos y seguir con la ruta que se estableció.

```
#include "Aria.h"

int main(int argc, char **argv)
{
  Aria::init();

  ArArgumentParser parser(&argc, argv);
  parser.loadDefaultArguments();
  ArSimpleConnector simpleConnector(&parser);

  //conexión de la computadora al robot
  ArRobot robot;
  ArSonarDevice sonar;

  // Adición de librería del sonar
  ArActionAvoidFront avoidFrontNear("Avoid Front Near", 225, 0);   // Librerías para evasión de
obstaculos cercanos
  ArActionAvoidFront avoidFrontFar;                    //
 Evasión de obstáculos a una distancia mayor

  ArAnalogGyro gyro(&robot);
  robot.addRangeDevice(&sonar);


  // Se agrega respuesta a teclas para usar Esc para salir del programa
  ArKeyHandler keyHandler;
  Aria::setKeyHandler(&keyHandler);
  robot.attachKeyHandler(&keyHandler);
  printf("Se presiona Esc para salir\n");

  // Evitar colisión modificando velocidad
  ArActionLimiterForwards limiterAction("speed limiter near", 300, 600, 250);
  ArActionLimiterForwards limiterFarAction("speed limiter far", 300, 1100, 400);
  ArActionLimiterTableSensor tableLimiterAction;


  robot.addAction(&avoidFrontNear, 100);  // Se agrega la evasión de obstáculos con mayor
prioridad que table limiter action.

  robot.addAction(&avoidFrontFar, 99);
```

Anexo IIIa

```
//Se agregan opciones de evitar colisiones con máxima prioridad
  robot.addAction(&tableLimiterAction, 98);
  robot.addAction(&limiterAction, 95);
  robot.addAction(&limiterFarAction, 90);
  // El llegar a la coordenada deseada tiene menos prioridad que evitar las colisiones
  ArActionGoto gotoPoseAction("goto");
```

```
  robot.addAction(&gotoPoseAction, 50);
   if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
  {
   Aria::logOptions();
   exit(1);
  }

   // Conexión al robot
  if (!simpleConnector.connectRobot(&robot))
  {
   printf("No se pudo conectar al robot\n");
   Aria::exit(1);
  }
  robot.runAsync(true);
  robot.enableMotors();          // Enciende los motores
  bool first = true;
  int goalNum = 0;
  while (Aria::getRunning())
  {
   robot.lock();

   //  Escoge las coordenadas del nuevo destino
   if (first || gotoPoseAction.haveAchievedGoal())
   {
    first = false;
    goalNum++;
    if (goalNum > 4)
     goalNum = 1; // comienza otra vez en el destino 1

    // establecer coordenadas para cada nuevo destino

    if (goalNum == 1)
     gotoPoseAction.setGoal(ArPose(3000, 0));

    else if (goalNum == 2)
     gotoPoseAction.setGoal(ArPose(3000, 3000));

    else if (goalNum == 3)
     gotoPoseAction.setGoal(ArPose(0, 3000));

    else if (goalNum == 4)
     gotoPoseAction.setGoal(ArPose(0, 0));

    ArLog::log(ArLog::Normal, "Moviendose al nuevo destino %.0f %.0f",
                    gotoPoseAction.getGoal().getX(), gotoPoseAction.getGoal().getY());

   }
   robot.unlock();
  }

  // Desconectar del robot.
  Aria::shutdown();
  return 0;
}
```

Anexo IV: Seguimiento por medio de sonares

```
//Seguimiento con sonar

#include "Aria.h"
#include <time.h>
#include <iostream>
#include <string>
using namespace std;

 int main(int argc, char **argv)

{
  Aria::init();
  ArArgumentParser parser(&argc, argv);
  parser.loadDefaultArguments();
  ArSimpleConnector simpleConnector(&parser);
  ArRobot robot;
  ArSonarDevice sonar;

  ArAnalogGyro gyro(&robot);
  robot.addRangeDevice(&sonar);
  ArKeyHandler keyHandler;
  Aria::setKeyHandler(&keyHandler);
  robot.attachKeyHandler(&keyHandler);
  printf("Presiona Esc para desconectar \n");


    ArSensorReading reading;
    ArTcpConnection con;
    Aria::init();


  if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())

  {
    Aria::logOptions();
    exit(1);
  }


  if (!simpleConnector.connectRobot(&robot))
  {
    printf("No se puede conectar al robot \n");
            Aria::exit(1);
  }

  robot.runAsync(true);
  robot.enableMotors();
  bool first = true;
  while (Aria::getRunning())
  {

    robot.lock     ();
          int i;
```

```
        string m;
  printf("\r");
```

Anexo IVa

```
for (i = 0; i < 8; i++)
        {
                printf(" %4d", robot.getClosestSonarNumber(-90,90));
                printf(" %4d", robot.getSonarRange(i));
                printf(" %4d",m);
                if (robot.getSonarRange(3)<1000 && robot.getSonarRange(3)>500)
                {
                robot.setVel(500);
                }
                if (robot.getSonarRange(3)<=300)
                {
                robot.setVel(-200);
                }

// Condiciones para distancias, avanza, para, o se mueve hacia atrás
        if (robot.getSonarRange(3)>300 && robot.getSonarRange(3)<=500)
                {
                robot.stop();
                m="No te alcanzo";
                }
                if (robot.getSonarRange(4)<1000 && robot.getSonarRange(4)>500)
                {
                robot.setVel(500);
                m="Hola";
                }
                if (robot.getSonarRange(4)<=300)
                {
                robot.setVel(-200);
                m="adios";
                }
                if (robot.getSonarRange(4)>300 && robot.getSonarRange(4)<=500)
                {
                robot.stop();
                m="No te alcanzo";
                }
                /*if(robot.getClosestSonarNumber(-90, 90)==0)
                {
                m="no puedo girar";
                }
                if(robot.getClosestSonarNumber(-91, 91)==1 &&
                robot.getSonarRange(0)>1000)
                {
                robot.setHeading(90);
                }
        }

        robot.unlock();
  }
  Aria::shutdown();
  return 0;
}
```

Anexo V: Seguimiento con sonar con vueltas

```
#include "Aria.h"

int main(int argc, char **argv)
 {
//empieza inicialización de servicios, clases, etc
   Aria::init();
   ArArgumentParser parser(&argc, argv);
   parser.loadDefaultArguments();
   ArSimpleConnector simpleConnector(&parser);
   ArRobot robot;
   ArSonarDevice sonar;
   ArAnalogGyro gyro(&robot);
   robot.addRangeDevice(&sonar);
   ArKeyHandler keyHandler;
   Aria::setKeyHandler(&keyHandler);
   robot.attachKeyHandler(&keyHandler);
   printf("You may press escape to exit\n");

//Declaración de sonares
     ArSensorReading reading;
     ArTcpConnection con;
     Aria::init();

//Revisa el analizador de comandos
   if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
   {
     Aria::logOptions();
     exit(1);
   }

//Revisa si el Robot está conectado, si no está conectado sale del //programa
if (!simpleConnector.connectRobot(&robot))
   {
     printf("Could not connect to robot... exiting\n");
            Aria::exit(1);
   }
   robot.runAsync(true);
   robot.enableMotors();  //Habilita los motores del robot
   bool first = true;

 //Inicia la actividad del robot
while (Aria::getRunning())
   {
     robot.lock     ();

//Si la lectura de los sonares forntales está entre .5 y 1m avanza a 500mm/s
          if ((robot.getSonarRange(3)<1000 && robot.getSonarRange(3)>500) ||
(robot.getSonarRange(4)<1000 && robot.getSonarRange(4)>500))
                    {
                    robot.setVel(500);
                    }
```

```
//Si la lectura de los sonares fotnales es menor a 300mm retrocede a 200mm/s
                else if (robot.getSonarRange(3)<=300 || robot.getSonarRange(4)<=300)
                {
                robot.setVel(-200);
                }

//Si la lectura de los sonares frontales está entre .3 y .5m el robot se detiene
                else if ((robot.getSonarRange(3)>300 && robot.getSonarRange(3)<=500) ||
(robot.getSonarRange(4)>300 && robot.getSonarRange(4)<=500))
                {
                robot.stop();
                }
```

Anexo Va

```
//Si el sonar con la lecutra más cercana es el 2 entonces el robot gira 10 gradosa la izquierda,
// pero si el sonar lateral (sonar 0) tiene una lectura menor que 1.2m el robot no gira
else if (robot.getClosestSonarNumber(-90,90)==2)
                {
                        if (robot.getSonarRange(0)<=1200)
                        {

                                robot.stop();
                        }
                        else
                        {
                                robot.setRotVel(100);
                                robot.setDeltaHeading(10);
                        }
                }
//Si el sonar con la lecutra más cercana es el 5 entonces el robot gira 10 grados a la derecha,
//pero si el sonar lateral (sonar 7) tiene una lectura menor que 1.2m el robot no gira
else if (robot.getClosestSonarNumber(-90,90)==5)
                        {
                        if (robot.getSonarRange(7)<=1200)
                        {
                        robot.stop();
                        }
                else
                        {
                        robot.setRotVel(100);
                        robot.setDeltaHeading(-10);
                        }
                }
                else
                {
                        robot.stop();
                }

        robot.unlock();
  }

  Aria::shutdown();
  return 0;
}
```

Anexo VI: Programación del seguimiento de dos colores con adaptación de velocidad y giro según última posición del objetivo (Etapa 6)

```
#include <math.h>
#include "Aria.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char **argv)
 {
 //empieza inicialización de servicios, clases, etc
  Aria::init();
  ArArgumentParser parser(&argc, argv);
  parser.loadDefaultArguments();
  ArSimpleConnector simpleConnector(&parser);
  ArRobot robot;
  ArSonarDevice sonar;
  ArAnalogGyro gyro(&robot);
  robot.addRangeDevice(&sonar);
  ArKeyHandler keyHandler;
  Aria::setKeyHandler(&keyHandler);
  robot.attachKeyHandler(&keyHandler);
  printf("You may press escape to exit\n");
  ArSensorReading reading;
  ArTcpConnection con;
  Aria::init();
 //termina inicialización de servicios, clases, etc

 //Revisa el analizador de comandos
  if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
                {
                Aria::logOptions();
                exit(1);
                }
        //termina la revisón del analizador de comandos


        //Revisa si el Robot está conectado, si no está conectado sale del programa
 if (!simpleConnector.connectRobot(&robot))
                {
                printf("Could not connect to robot... exiting\n");
                Aria::exit(1);
                }
 //Termina revisión de la conexión

 //Declaración de variables de variables
        float cuenta1, cuenta2, bandera;          //variables de entrada
        float band, xmax, ungrado, resta, angulo, valorx, valory, noencuentra;
        xmax=117; // ancho de la imagen (en pixeles)
        noencuentra=1;
        robot.runAsync(true);
```

```
        robot.enableMotors();                          //habilita los motores del robot
        bool first = true;
        int bandex, deroizq;
        printf("\r");
        FILE *ptrCf;
        //puntero para la lectura de archivos de texto
 //Termina declaración de variables

 //Inicia la actividad del robot
        while (Aria::getRunning())
                {
                        //abre el while cuando se activa el robot
//adaptación de rutina de lectura de archivos de texto[35]

                        //lee bandera
                if((ptrCf=fopen("bandera.txt", "r"))== NULL)
                {
                        printf( "El archivo no puede abrirse\n");
                }
                        else
                {
                fscanf( ptrCf, "%15f", &bandera);
                while ( !feof( ptrCf))
                        {
                                fscanf ( ptrCf, "%10f", &bandera);
                        }
                                fclose ( ptrCf);
                }

                if (bandera==1)
                {
        //leer archivo de coordenada x
                if((ptrCf=fopen("coordenada.txt", "r"))== NULL)
                        {
                                printf( "El archivo no puede abrirse\n");
                        }
                        else
                        {
                        printf( "matlab da:\n");
                        fscanf( ptrCf, "%15f", &cuenta1);
                        while ( !feof( ptrCf))
                                {
                                        printf("en x:");
                                        printf( "%5f", cuenta1);
                                        fscanf ( ptrCf, "%10f", &cuenta1);
                                        printf("\n");
                                }
                                fclose ( ptrCf);
                        }

//leer archivo de coordenada en y

                if((ptrCf=fopen("coordenada2.txt", "r"))== NULL)
                        {
                        printf( "El archivo no puede abrirse\n");
                        }
```

```
                else
                        {
                        printf( "matlab da:\n");
                        fscanf( ptrCf, "%15f", &cuenta2);
                        while ( !feof( ptrCf))
                        {
                                printf("en y:");
                                printf( "%5f", cuenta2);
                                fscanf ( ptrCf, "%10f", &cuenta2);
                                printf("\n");
                        }
                        fclose ( ptrCf);
                }
                FILE *ptrCf2;
                if((ptrCf2=fopen("bandera.txt", "w"))== NULL){
                printf( "El archivo no puede abrirse\n");
        }
        else {
                band=0.0;
                        fprintf( ptrCf2, "%10f", band);
                        fclose ( ptrCf2);
                }

        bandex=1;
        valorx=cuenta1;
        valory=cuenta2;
                        }

// +90° es a la izquierda
// -90° es a la derecha

// Si la lectura de los archivos de texto se encuentra
// en el rango de la derecha
if(valorx>=0 && valorx<39 && bandex==1)
{
        if (robot.isHeadingDone())//verifica que no exista un giro previo
        {
        resta=valorx-(xmax/2);          // cálculo del ángulo
        ungrado=46/xmax;                // de
        angulo=-1*resta*ungrado;          // giro
        printf("el angulo a girar es:");
        printf("%f", angulo);
        robot.setRotVel(1000000);
        robot.setDeltaHeading(angulo);
        deroizq=200;
        bandex=0;
        }
        else                                    //en caso de haber un giro previo
        {                                       //avisa que no puede girar
        printf("todavia no termino");
        bandex=0;
        }
}
//Termina giro a la izquierda
// Si la lectura de los archivos de texto se encuentra
// en el rango de la derecha
```

```
if(valorx>=78 && valorx<=117 && bandex==1)
{
        if (robot.isHeadingDone())//verifica que no exista un giro previo
        {
        resta=valorx-(xmax/2);          // cálculo del ángulo
        ungrado=46/xmax;                                    //de
        angulo=-1*resta*ungrado;                    //giro
        printf("el angulo a girar es:");
        printf("%f", angulo);
        robot.setRotVel(10000000);
        robot.setDeltaHeading(angulo);
        bandex=0;
        deroizq=-200;
        }
        else                                                //en caso de haber un giro previo
        {                                                   //avisa que no puede girar
        printf("todavia no termino");
        bandex=0;
        }
}
//Termina giro a la derecha

// Si la lectura de los archivos de texto se encuentra
// en el rango del centro
else if(valorx>=39 && valorx<78)
{
if(robot.getSonarRange(1)<=800 || robot.getSonarRange(2)<=800 ||
robot.getSonarRange(3)<=800 || robot.getSonarRange(4)<=800 || robot.getSonarRange(5)<=800
|| robot.getSonarRange(6)<=800 )
                        {
                        robot.stop();// si la lectura de los sonares es muy
                                        //cercana el robot no avanza
                        }
else if((robot.getSonarRange(1)>800 && robot.getSonarRange(1)<=2000) ||
                (robot.getSonarRange(2)>800 &&robot.getSonarRange(2)<=2000)
                || (robot.getSonarRange(3)>800 && robot.getSonarRange(3)<=2000) ||
        (robot.getSonarRange(4)>800 && robot.getSonarRange(4)<=2000) ||
        (robot.getSonarRange(5)>800 && robot.getSonarRange(5)<=2000) ||
        (robot.getSonarRange(6)>800 && robot.getSonarRange(6)<=2000) )
                {
                robot.setVel(300);// si la lecutra de los sonares está
                                        //entre .8 y 2m el robot avanza a 300mm/s
                }
else if robot.getSonarRange(1)>2000 || robot.getSonarRange(2)>2000 ||
robot.getSonarRange(3)>2000 || robot.getSonarRange(4)>2000 || robot.getSonarRange(5)>2000
|| robot.getSonarRange(6)>2000 )
                {
                robot.setVel(380); // si la lecutra de los sonares es mayor
                                        //a 2m el robot avanza a 380mm/s
                }
deroizq=deroizq*-1;
}


//En caso de no encontrar un objetivo (matlab manda un 400.0)
```

Anexo VIa

```
else if(valorx==400.0 && bandex==1)
{
                                        printf("papa no aparece\n");
                                        robot.stop();
                                        robot.setRotVel(deroizq); //el robot gira en la
                                                        //dirección del último objetivo
                                        while(valorx==400.0)
                                        {
                                                banderita=1;
                                                goto leer;
                                                regreso:
                                                banderita=0;
                                        }
                                        robot.stop();

                                }

}
        Aria::shutdown();
        return 0;
        }
```

Anexo VII: Codigo correspondiente a la quinta etapa, se incluye el seguimiento de color y la programación del robot.

```
function mframes

vidobj = videoinput('winvideo');          %Se inicializa la entrada de video, en este caso se refiere
a la cámara web
```

a. Código usado en Robocup para definir matriz de color y etiquetado de pixeles

```
%Empieza definición de color, se definen las matrices correspondientes para los valores de H, S
y V

colors = [ ] ;

% Rangos de H y S

h = 0.06:0.002:0.13 ;
s = 0.50:0.002:1 ;
ss = [] ;
for i = 1:length(h)
hh((i-1)*length(s)+1:i*length(s),1) = h(i) ;   %Se crean matrices para cada uno de los intervalos de
colores

ss = [ss ; s'] ;
end

%Se define el intervalo de V, para crear una matriz del mismo tamaño que hh y ss

v = repmat([0.65:0.002:1],[length(hh),1],1) ;
colors(:,:,1) = repmat(hh,[1,length(v(1,:))],1) ;
colors(:,:,2) = repmat(ss,[1,length(v(1,:))],1) ;
colors(:,:,3) = v ;

% Se transforman los valores del espacio HSV a RGB, con valores de 1 a 256

colors = hsv2rgb(colors) ;
colors = round(colors*255)+1 ;

% Se crea una matriz en tres dimensiones con los colores definidos, la matriz tiene nombre clrs

clrs = repmat(0,[256,256],256) ;
s = size(colors) ;
for i = 1:s(1)
for j = 1:s(2)
clrs(colors(i,j,1),colors(i,j,2),colors(i,j,3)) = 1 ;
end
end

%Termina definición de color.
```

```
%Se despliega en pantalla, la imagen tomada por la cámara web

for i = 1:10000000000000000   % Se define el ciclo del numero de veces que se tomaran
imagenes
   tic

   img1 = getsnapshot(vidobj);   %Se captura una imagen desde la camara web y se se
despliega en pantalla

   %subplot(2,3,1),imshow(img)

% Empieza Pixeling

pix = 3 ;%Se define el tamaño de muestreo de la imagen para el etiquetado
p = round(pix/2) ;
img = double(img1) ;
h = size(img,1) ;
w = size(img,2) ;

%Se realiza el etiquetado de los pixeles con el tamaño de muestreo definido, comparándolos con
la matriz cnls

segmented_image = repmat(0,[floor((h-pix)/pix+1),floor((w-pix)/pix+1)],1) ;
for y = p : pix : floor((h-pix)/pix)*pix+p ;
for x = p : pix : floor((w-pix)/pix)*pix+p ;
c = clrs(img(y,x,1) + 1,img(y,x,2) + 1,img(y,x,3) + 1) ;
segmented_image((y-p)/pix+1,(x-p)/pix+1) = c ;
end
end

%Termina Pixeling
Todo el código correspondiente a la definición del color, así como el proceso de etiquetado de
pixeles, fue extraído de:
Luijten H.J.C. "Basics of color based computer vision implemented in Matlab" Eindhoven, June,
2005

subplot(1,2,1),imshow(img1);
```

b. Funciones de *opening*, *closening* y centroide

```
%Empieza definición de centroide, así como las operaciones de opening y closening.
%Hacer closening

BW1=segmented_image;
SE=ones(5,5);
BW2=imdilate(BW1,SE);                %Dilatación
BW3=imerode(BW2,SE);                        % Erosión

%Hacer opening

BW2=imerode(BW3,SE);                %Erosion
BW4=imedilate(BW3,SE);              %Dilatacion

%Localizacion del centroide
```

```
siz = sum(sum(BW4>0));
B = double(BW4);
Bx = B .* ( ones(size(B,1),1) * [1:size(B,2)] );
By = B .* ( [1:size(B,1)]' * ones(1,size(B,2)) );
cx = sum(sum(Bx)) / siz;          %Coordenada en x
cy = sum(sum(By)) / siz;          %Coordenada en y


c = [cx,cy];
subplot(1,2,2),imshow(BW4)

hold on
plot(cx,cy,'g*')
hold off
bandera=1.0;

%Termina localizacion de centroide

%Si no se detecto un centroide, a las coordenadas x y se les asigna un valor de 400, esto
también tiene una función de sincronía en el movimiento del robot

if siz==0
cx=400;
cy=400;

else
  if cosa<50
   cx=400;
   cy=400;
  else
     if cosa2<50
      cx=400;
      cy=400;
     else
      cx = sum(sum(Bx)) / siz;
      cy = sum(sum(By)) / siz;
     end
  end
end
```

c. Escritura de coordenadas en archivos txt

```
%Se escribe en archivos txt, las coordenadas x y, así como una bandera en 1, para sincronía

save 'coordenada.txt' cy -ascii;
save 'coordenada2.txt' cx -ascii;
save 'bandera.txt' bandera -ascii;

display(c);
display(cx);

display(cx);
```

```
elapsedTime=toc


stop(vidobj)
end

%Se termina el ciclo de la segmentación y localización de centroide, también se detiene la
captura de video
```

Anexo VII. Código para la sexta etapa (seguimiento de dos colores), incluye código para segmentación de color y código para programación del robot.

a. Código para definición de dos colores

function mframes2

%Se despliega un menu para dar opciones de posibles combinaciones a seguir, la cuarta opción permite ingresar los rangos de H, S y V para colores no definidos en esta programación

```
x=1;
while(x==1)
x=0;
display('Elige el número de la combinación que quieres');
display('1: Rosa y Naranja');
display('2: Rosa y Verde');
display('3: Naranja y Verde');
display('4: Dar nuevos valores');
pregunta=input('');
switch pregunta
   case 1
      inih=.88;
      termih=.98;
      inis=.3;
      termis=.8;
      iniv=.3;
      termiv=.8;
      inih2=.03;
      termih2=.13;
      inis2=.5;
      termis2=1;
      iniv2=.03;
      termiv2=.13;
   case 2
      inih=.90;
      termih=.97;
      inis=.5;
      termis=.8;
      iniv=.5;
      termiv=1;
      inih2=.25;
      termih2=.35;
      inis2=.2;
      termis2=.6;
      iniv2=.5;
      termiv2=.8;
   case 3
      inih=.03;
      termih=.13;
      inis=.5;
      termis=1;
      iniv=.5;
```

```
        termiv=1;
        inih2=.27;
        termih2=.43;
        inis2=.2;
        termis2=.7;
        iniv2=.3;
        termiv2=.7;
    case 4
        inih = input('Dame el valor de inicio de h:     ');
        termih = input('Dame el valor de término de h:  ');
        inis = input('Dame el valor de inicio de s:     ');
        termis = input('Dame el valor de término de s:  ');
        iniv = input('Dame el valor de inicio de v:     ');
        termiv = input('Dame el valor de término de v:  ');
        inih2 = input('Dame el valor de inicio de h 2:  ');
        termih2 = input('Dame el valor de término de h2: ');
        inis2 = input('Dame el valor de inicio de s2:   ');
        termis2 = input('Dame el valor de término de s2: ');
        iniv2 = input('Dame el valor de inicio de v2:   ');
        termiv2 = input('Dame el valor de término de v2: ');
    otherwise
        display ('no es una opción');
        x=1;
    end
end

vidobj = videoinput('winvideo');


%Empieza definición de color, se definen las matrices correspondientes para los valores de H, S
y V

colors = [] ;

% Rangos de H y S

h = inih:0.002:termih ;
s = inis:0.002:termis ;
ss = [] ;
for i = 1:length(h)
hh((i-1)*length(s)+1:i*length(s),1) = h(i) ;
ss = [ss ; s'] ;
end

%Se define el intervalo de V, para crear una matriz del mismo tamaño que hh y ss

v = repmat([iniv:0.002:termiv],[length(hh),1],1) ;
colors(:,:,1) = repmat(hh,[1,length(v(1,:))],1) ;
colors(:,:,2) = repmat(ss,[1,length(v(1,:))],1) ;
colors(:,:,3) = v ;

% Se transforman los valores del espacio HSV a RGB, con valores de 1 a 256

colors = hsv2rgb(colors) ;
colors = round(colors*255)+1 ;
```

```
% Se crea una matriz en tres dimensiones con los colores definidos, la matriz tiene nombre clrs

clrs = repmat(0,[256,256],256) ;
s = size(colors) ;
for i = 1:s(1)
for j = 1:s(2)
clrs(colors(i,j,1),colors(i,j,2),colors(i,j,3)) = 1 ;
end
end

%Termina definición del primer color.

%Se hace el mismo proceso anterior, pero para un segundo color

colores = [] ;

% Rangos de H y S

hdos = inih2:0.002:termih2 ;
sdos = inis2:0.002:termis2;

ssdos = [] ;
for a = 1:length(hdos)
hhdos((a-1)*length(sdos)+1:a*length(sdos),1) = hdos(a) ;
ssdos = [ssdos ; sdos'] ;
end
%Se define el intervalo de V, para crear una matriz del mismo tamaño que hh y ss

vdos = repmat([iniv:0.002:termiv],[length(hhdos),1],1) ;
colores(:,:,1) = repmat(hhdos,[1,length(vdos(1,:))],1) ;
colores(:,:,2) = repmat(ssdos,[1,length(vdos(1,:))],1) ;
colores(:,:,3) = vdos ;

% Se transforman los valores del espacio HSV a RGB, con valores de 1 a 256

colores = hsv2rgb(colores) ;
colores = round(colores*255)+1 ;

% Se crea una matriz en tres dimensiones con los colores definidos, la matriz tiene nombre clrs

clrsdos = repmat(0,[256,256],256) ;
sdos = size(colores) ;
for a = 1:sdos(1)
for b = 1:sdos(2)
clrsdos(colores(a,b,1),colores(a,b,2),colores(a,b,3)) = 1 ;
end
end
%Termina definición de segundo color.

for i =
1:10000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
  % Se define el ciclo del numero de veces que se tomaran imágenes, se busca que sea lo mas
grande posible para que el seguimiento sea en cualquier momento

   tic
```

```
  img1 = getsnapshot(vidobj);
imágenes\color3_2.bmp');

% Empieza Pixeling

pix = 3 ;%Se define el tamaño de muestreo de la imagen para el etiquetado
p = round(pix/2) ;

img = double(img1) ;
h = size(img,1) ;
w = size(img,2) ;
%Se realiza el etiquetado de los pixeles

segmented_image = repmat(0,[floor((h-pix)/pix+1),floor((w-pix)/pix+1)],1) ;
for y = p : pix : floor((h-pix)/pix)*pix+p ;
for x = p : pix : floor((w-pix)/pix)*pix+p ;
c = clrs(img(y,x,1) + 1,img(y,x,2) + 1,img(y,x,3) + 1) ;
cdos= clrsdos(img(y,x,1) + 1,img(y,x,2) + 1,img(y,x,3) + 1) ;

mat1((y-p)/pix+1,(x-p)/pix+1) = c;
mat2((y-p)/pix+1,(x-p)/pix+1) = cdos;
%El etiquetado se hace para cada color, por lo que se generan dos matrices binarias, una para
cada color

end
end
wara=mat1+mat2;        %Se suman las dos matrices binarias para generar una sola con ambos
colores detectados.

segmented_image= wara ;

%Termina Pixeling
Todo el código correspondiente a la definición del color, así como el proceso de etiquetado de
pixeles, fue extraído de:
Luijten H.J.C. "Basics of color based computer vision implemented in Matlab" Eindhoven, June,
2005

%Se obtienen los tamaños de las matrices binarias, para mas adelante hacer una validación

cosa = length(find(mat1(: )));
cosa2 = length(find(mat2(: )));
cosa3=length(find(wara(:)));
cosa
cosa2
cosa3

subplot(1,2,1),imshow(img1);

%Empieza definición de centroide, así como las operaciones de opening y closening.
%Hacer closening

BW1=segmented_image;
SE=ones(5,5);
BW2=imdilate(BW1,SE);
BW3=imerode(BW2,SE);
```

```
%Hacer opening

BW2=imerode(BW3,SE);                %Erosion
BW4=imedilate(BW3,SE);              %Dilatacion

%Localizacion del  centroide

siz = sum(sum(BW4>0));
B = double(BW4);
Bx = B .* ( ones(size(B,1),1) * [1:size(B,2)] );
By = B .* ( [1:size(B,1)]' * ones(1,size(B,2)) );

%Se realiza una validación del tamaño de las matrices, si la matriz es muy pequena, es decir,
menor a 50 pixeles, se considera que no hubo detección correcta del objetivo, y las coordenadas
de x y se escriben en 400

if siz==0
        cx=400;
        cy=400;
else
  if cosa<50
   cx=400;
   cy=400;
  else
     if cosa2<50
      cx=400;
      cy=400;
     else
      cx = sum(sum(Bx)) / siz;
      cy = sum(sum(By)) / siz;
     end
  end
end



c = [cx,cy];
subplot(1,2,2),imshow(BW4)

hold on
plot(cx,cy,'g*')
hold off
bandera=1.0;

%Termina localizacion de centroide

%Si no se detecto un centroide, a las coordenadas x y se les asigna un valor de 400, esto
también tiene una función de sincronía en el movimiento del robot

%Se escribe en archivos txt, las coordenadas x y, así como una bandera en 1, para sincronía

save 'C:\Users\owner\Desktop\New Folder\Aria 2.5.1\bin\coordenada.txt' cx -ascii;
save 'C:\Users\owner\Desktop\New Folder\Aria 2.5.1\bin\coordenada2.txt' cy -ascii;
save 'C:\Users\owner\Desktop\New Folder\Aria 2.5.1\bin\bandera.txt' bandera -ascii;
```

```
display (cosa);
display(c);
display(cx);
display(cy);

elapsedTime=toc

stop(vidobj)
end
```

Anexo IX. Programación de la aplicación inalámbrica

Anexo IXa:

Login

- **Permite el ingreso a la aplicación o no**

```
stop();
userinput.restrict="a-zA-Z0-9";
Selection.setFocus(userinput);
passinput.restrict="a-zA-Z0-9";
status="Ingresa tus datos";
this.onEnterFrame = function () {
        if(_root.checklog == 1){
                _root.gotoAndStop(2);
        }
        if(_root.checklog == 2){
                _root.gotoAndStop(4);
        }
}
```

Anexo IXc: Departamentos

- Permite la visualización de la pantalla de departamentos y cambia de color según la pestaña de la pantalla presentada.

```
onClipEvent (load) {
        this.xO = this._x;
        this.scaleO = this._s;
        _root.tt.start();

        this.onRelease = function() {
                getURL("javascript:changeBgColor('#64C6D7')");
                _root.carro.tween("_y", 0, .3, "linear");
                _root.prods.tween("_y", 20, .3, "linear");
                _root.deptos.tween("_y", 530, .3, "linear");
                _root.home.tween("_y", 550, .3, "linear");
                _root.cargador.tween("_y", 40, .2, "linear");
                _root.cargador.loadMovie("");
                pausafun=function(){
                        clearInterval(pausafunID)
                _root.cargador.loadMovie("productos.swf");
                                }
                        pausafunID=setInterval(pausafun,400)
/*              _root.deptos.onTweenComplete = function(Y) {
          if (Y == "_y") {

                  }
};*/
        };
}
```

Anexo IXb: Código de la pantalla de Inicio

- Permite la visualización de la siguiente imagen de promoción de los productos.

```
function showNext() {
        depth++;
        clone = cage.attachMovie("imagemc", "imagemc"+depth, depth);
        cage["imagemc"+(depth-10)].removeMovieClip();
        curimg++;
        if (curimg>images.length-1) {
                curimg = 0;
        }
        preload(images[curimg], clone);
        trace("curimg = "+curimg);
}
```

- Permite la visualización de la imagen anterior de promoción de los productos.

```
function showPrevious() {
        depth++;
        clone = cage.attachMovie("imagemc", "imagemc"+depth, depth);
        cage["imagemc"+(depth-10)].removeMovieClip();
        curimg--;
        if (curimg<0) {
                curimg = images.length-1;
        }
        preload(images[curimg], clone);
        trace("curimg = "+curimg);
}
```

- Permite cargar las imagenes de promoción de productos para ser mostradas con rápidez.

```
function preload(file, clip) {
        loadMovie(path+file, clip.cage);
        clip.onEnterFrame = function() {
                tkb = this.cage.getBytesTotal();
                lkb = this.cage.getBytesLoaded();
                p = Math.round(lkb/tkb*100);
                if (isNaN(p)) {
                        p = 0;
                }
                if (p>0) {
                        percentage = p;
                }
                if (lkb == tkb && p>99) {
                        percentage = "";
                        this.maskmc.dotween();
                        if (autoplaymc._currentframe == 2) {
                                clearInterval(autoID);
                                autoID = setInterval(showNext, delay);
                        }
                        delete this["onEnterFrame"];
                }
        };
}
```

- Después de 4 segundos se muestra una nueva imagen de promoción de productos, hasta que se muestren todas y se vuelva a empezar el ciclo.

```
function autoPlay() {
        autoID = setInterval(showNext, delay);
        showNext();
```

```
        autoplaymc.gotoAndStop(2);
}
function stopAutoPlay() {
        clearInterval(autoID);
        autoplaymc.gotoAndStop(1);
}
images = [];
images[0] = "m_imgs/img1.jpg";
images[1] = "m_imgs/img2.jpg";
images[2] = "m_imgs/img3.jpg";
images[3] = "m_imgs/img4.jpg";
images[4] = "m_imgs/img5.jpg";

curimg = -1;
depth = 100;
delay = 4000;
autoPlay();
```

Anexo IXc: Código de Departamentos

- **Permite visualizar el nombre de cada uno de los departamentos en la pantalla, colocando sobre cada uno de ellos una etiqueta con el nombre cada vez que se pase el mouse de la computadora sobre el área del supermercado.**

```
MovieClip.prototype.aplicarColorMC = function(colorMC:Number) {
   var newColMC:Color = new Color(this);
   newColMC.setRGB(colorMC);
};
```

- Se muestra la etiqueta de Lacteos cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
lacteos.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        clacteos._alpha += (100-clacteos._alpha)/3;
clacteos._x -= (clacteos._x-_xmouse)/2;
        clacteos._y -= (clacteos._y-_ymouse)/2;
        }
};
lacteos.onRollOut = function() {
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
                clacteos._alpha += (0-clacteos._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Salchichoneria cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
salchichoneria.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        csalchichoneria._alpha += (100-csalchichoneria._alpha)/3;
csalchichoneria._x -= (csalchichoneria._x-_xmouse)/2;
        csalchichoneria._y -= (csalchichoneria._y-_ymouse)/2;
        }
};
```

```
salchichoneria.onRollOut = function() {
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
                csalchichoneria._alpha += (0-csalchichoneria._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Cajas cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
cajas.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        ccajas._alpha += (100-ccajas._alpha)/3;
ccajas._x -= (ccajas._x-_xmouse)/2;
        ccajas._y -= (ccajas._y-_ymouse)/2;
        }
};
cajas.onRollOut = function() {
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
                ccajas._alpha += (0-ccajas._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Farmacia cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
farmacia.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        cfarmacia._alpha += (100-cfarmacia._alpha)/3;
cfarmacia._x -= (cfarmacia._x-_xmouse)/2;
        cfarmacia._y -= (cfarmacia._y-_ymouse)/2;
        }
};
farmacia.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
                cfarmacia._alpha += (0-cfarmacia._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Ofertas cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
ofertas.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        cofertas._alpha += (100-cofertas._alpha)/3;
cofertas._x -= (cofertas._x-_xmouse)/2;
        cofertas._y -= (cofertas._y-_ymouse)/2;
        }
};
ofertas.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
```

```
            cofertas._alpha += (0-cofertas._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Limpieza cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
limpieza.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        climpieza._alpha += (100-climpieza._alpha)/3;
climpieza._x -= (climpieza._x-_xmouse)/2;
        climpieza._y -= (climpieza._y-_ymouse)/2;
        }
};
limpieza.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
            climpieza._alpha += (0-climpieza._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Ropa y Calzado cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
ropacalzado.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        cropacalzado._alpha += (100-cropacalzado._alpha)/3;
cropacalzado._x -= (cropacalzado._x-_xmouse)/2;
        cropacalzado._y -= (cropacalzado._y-_ymouse)/2;
        }
};
ropacalzado.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
            cropacalzado._alpha += (0-cropacalzado._alpha)/3;
        };
};
```

- * Se muestra la etiqueta de Ferretería cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
ferreteria.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        cferreteria._alpha += (100-cferreteria._alpha)/3;
cferreteria._x -= (cferreteria._x-_xmouse)/2;
        cferreteria._y -= (cferreteria._y-_ymouse)/2;
        }
};
ferreteria.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
```

```
                cferreteria._alpha += (0-cferreteria._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Enlatados cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
enlatados.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        cenlatados._alpha += (100-cenlatados._alpha)/3;
cenlatados._x -= (cenlatados._x-_xmouse)/2;
        cenlatados._y -= (cenlatados._y-_ymouse)/2;
        }
};
enlatados.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
                cenlatados._alpha += (0-cenlatados._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Cereales cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
cereales.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        ccereales._alpha += (100-ccereales._alpha)/3;
ccereales._x -= (ccereales._x-_xmouse)/2;
        ccereales._y -= (ccereales._y-_ymouse)/2;
        }
};
cereales.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
                ccereales._alpha += (0-ccereales._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Electrónica cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
electronica.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        celectronica._alpha += (100-celectronica._alpha)/3;
celectronica._x -= (celectronica._x-_xmouse)/2;
        celectronica._y -= (celectronica._y-_ymouse)/2;
        }
};
electronica.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
```

```
        this.onEnterFrame = function() {
                celectronica._alpha += (0-celectronica._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Hogar cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
hogar.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        chogar._alpha += (100-chogar._alpha)/3;
chogar._x -= (chogar._x-_xmouse)/2;
        chogar._y -= (chogar._y-_ymouse)/2;
        }
};
hogar.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
                chogar._alpha += (0-chogar._alpha)/3;
        };
};
```

- Se muestra la etiqueta de Verduras cuando se pasa el mouse sobre el área perteneciente al departamento, además de colorear la misma de un color verde.

```
verduras.onRollOver = function() {
   this.aplicarColorMC(0x84DC6E);
        this.onEnterFrame = function () {
        cverduras._alpha += (100-cverduras._alpha)/3;
cverduras._x -= (cverduras._x-_xmouse)/2;
        cverduras._y -= (cverduras._y-_ymouse)/2;
        }
};
verduras.onRollOut = function() {
   this.aplicarColorMC(0x999999);
   this.aplicarColorMC(0x999999);
        this.onEnterFrame = function() {
                cverduras._alpha += (0-cverduras._alpha)/3;
        };
};
```

Anexo IXd: Productos

- Permite la visualización de la pantalla de úsqueda de los productos y cambia de color según la pestaña de la pantalla presentada.

```
onClipEvent (load) {
        this.xO = this._x;
        this.scaleO = this._s;
        _root.tt.start();

        this.onRelease = function() {
                getURL("javascript:changeBgColor('#FF7890')");
                _root.carro.tween("_y", 0, .3, "linear");
```

```
                _root.prods.tween("_y", 20, .3, "linear");
                _root.deptos.tween("_y", 40, .3, "linear");
                _root.home.tween("_y", 550, .3, "linear");
                _root.cargador.tween("_y", 60, .2, "linear");
                _root.cargador.loadMovie("");
                pausafun=function(){
                        clearInterval(pausafunID)
                _root.cargador.loadMovie("departamentos.swf");
                                }
                        pausafunID=setInterval(pausafun,400)
/*              _root.deptos.onTweenComplete = function(Y) {
        if (Y == "_y") {

                }
};*/
        };
}
```

Boton de Login

- Comunica la interfaz de Flash con PHP para verificar si el usuario es valido o no mediante la busqueda en base de datos de cada uno de los usuarios.

```
on (release, keyPress "<Enter>") {
        if (user != "" && pass != "") {

                status = "Procesando Acceso, espera...";
                loadVariablesNum("login.php", 0, "POST");
        }
}
```

Anexo IXe: Carro

- Permite la visualización de la pantalla del carrito virtual y cambia de color según la pestaña de la pantalla presentada.

```
onClipEvent (load) {
        this.xO = this._x;
        this.scaleO = this._s;
        _root.tt.start();

        this.onRelease = function() {
                getURL("javascript:changeBgColor('#A44FB2')");
                _root.carro.tween("_y", 0, .3, "linear");
                _root.prods.tween("_y", 510, .3, "linear");
                _root.deptos.tween("_y", 530, .3, "linear");
                _root.home.tween("_y", 550, .3, "linear");
                _root.cargador.tween("_y", 20, .2, "linear");
                _root.cargador.loadMovie("");
                pausafun=function(){
                        clearInterval(pausafunID)
                _root.cargador.loadMovie("carro.swf");
                                }
                        pausafunID=setInterval(pausafun,400)
/*              _root.deptos.onTweenComplete = function(Y) {
```

```
            if (Y == "_y") {

                    }
};*/
        };
}
```

Código de la pantalla de Búsqueda de Productos

- **Consulta la base de datos de los productos en el supermercado mediante la obtención de los parámetros necesarios del código PHP**

```
import Conexion;
import mx.remoting.RecordSet;
var db:Conexion = new Conexion();
```

- Hace una consulta a la base de datos de los productos localizados en el supermercado.

```
db.consulta("SELECT precio as Precio,departamento as Departamento,cantidad as
Cantidad,marca as Marca,producto as Producto FROM productos");
db.onConsulta = function(rs) {
        tabla.dataProvider = rs._items;
};
```

- Extrae la información de la base de datos de los productos en el supermercado para ser mostradas en la interfaz Flash.

```
boton.onRelease = function(){
db.consulta("SELECT precio as Precio,departamento as Departamento,cantidad as
Cantidad,marca as Marca,producto as Producto FROM productos where id_producto =
'"+nombre_txt.text+"' or producto = '"+nombre_txt.text+"' or marca = '"+nombre_txt.text+"' or
departamento = '"+nombre_txt.text+"'");
db.onConsulta = function(rs) {
        tabla.dataProvider = rs._items;
};
};

btnall.onRelease = function(){
db.consulta("SELECT precio as Precio,departamento as Departamento,cantidad as
Cantidad,marca as Marca,producto as Producto FROM productos");
db.onConsulta = function(rs) {
        tabla.dataProvider = rs._items;
        nombre_txt.text = "";
};
};
```

Código Carrito

- Determina la velocidad de avance y rotación de la imagen del carrito dentro de la pantalla, además de permitir mover la misma por medio del PAD de la computadora o dispositivo que se esté utilizando.

```
velocidad = 10;
rotacion=1;
bton._alpha = 20;
stcar.text = "Off";

bton.onRelease = function(){
bton._alpha = 100;
```

```
btoff._alpha = 20;
stcar.text = "On";
carrito.onEnterFrame = function() {
with (carrito) {
if (Key.isDown(Key.RIGHT)) {
if (rotacion == 1) {
} else {
rotacion = 1;
_xscale = -_xscale;
}
_x += velocidad;
if (_x>=650) {
_x = 650;
}
}
if (Key.isDown(Key.LEFT)) {
if (rotacion == 2) {
} else {
rotacion = 2;
_xscale = -_xscale;
}
_x -= velocidad;
if (_x<=50) {
_x = 50;
}
}
if (Key.isDown(Key.DOWN)) {
_y += velocidad;
if (_y>=450) {
_y = 450;
}
}
if (Key.isDown(Key.UP)){
_y -= velocidad;
if (_y<=0) {
_y = 0;
}
}
}
}
};

btoff.onRelease = function(){
btoff._alpha = 100;
bton._alpha = 20;
stcar.text = "Off";
        carrito.onEnterFrame = function() {
with (carrito) {
        }
}
}
```

Código PHP

Código Login

- Crea la comunicación entre la base de datos creada en MySQL y la interfaz de Flash mediante el chequeo de los campos contenidos en la base de datos de usuarios para poder hacer la distinción entre un usuario de género masculino y femenino respectivamente. Código adaptado de envío de datos de un formulario. [36]

```
<?PHP

include 'connect.php';

$user=$_POST['user'];
$pass=$_POST['pass'];


$query = ("SELECT * FROM usuarios WHERE username = '$user' AND password = '$pass'");
$result = mysql_query( $query,Connection() ) or die ("didn't query");

$numRows = mysql_numrows( $result );
if ($numRows == 0){
        print "status=No tienes acceso!!&checklog=2";

        } else {

        while($i = mysql_fetch_row($result)) {

                $id = $i[0];
                $nombre = $i[1];
                $sexo = $i[3];


                if ($sexo == "M")
                print "status=Bienvenido $nombre&checklog=1&nnn=$id&sxx=$sexo";
                else
                print "status=Bienvenida $nombre&checklog=1&nnn=$id&sxx=$sexo";

                                                                        }
                        }

Clean();
?>
```

Código Connect

- Crea la conexión entre el servidor en el cual se encuentra la base de datos y la aplicación mediante la validación del servidor, el administrador, la clave y el nombre de la base de datos a la cual se quiere accesar. Código adaptado de base de datos. [37]

```
<?php
  $g_link = false;

  function Connection()
  {
     global $g_link;
     if( $g_link )
        return $g_link;
```

```
    $g_link = mysql_connect( 'mysql3.000webhost.com', 'a2004555_vic', 'vicsan9') or die('Could
not connect to server.' );
    mysql_select_db('a2004555_super', $g_link) or die('Could not select database.');
    return $g_link;
  }

  function Clean()
  {
    global $g_link;
    if( $g_link != false )
      mysql_close($g_link);
    $g_link = false;
  }

?>
```

Código Consultor

- Clase de conexión simple y de consulta a base de datos que devuelve un recordset de una consulta realizada a Flash orientada a objetos. Código adaptado de administración de base de datos MySQL y php MyAdmin. [38]

```
<?php
class Consultor{
        function Consultor(){
                $this->methodTable = array(
                        "consulta" =>array (
                                "description" => "Devuelve un objeto RecordSet a Flash de una
consulta pasada por parametro",
                                "access" => "remote",
                                "arguments" => array("sql")
                        )
                );
        }
        function consulta($sql){
                /*
```

- Se introducen los datos de la administración respectiva a la base de datos como el usuario, el password de acceso, nombre de la base de datos y el nombre del servidor, para que la consulta a la base de datos pueda ser realizada.

```
                */
                $cons_user = "a2004555_vic";
                $cons_pass = "vicsan9";
                $cons_db = "a2004555_super";
                //Conecta a la base de datos
                $dbh=mysql_connect("mysql3.000webhost.com",$cons_user,$cons_pass) or die
('Error conectandose a la base de datos por: ' . mysql_error());
                //Selecciona la base de datos
                mysql_select_db ($cons_db);
                //Guarda el resultado de la consulta en un identificador (Puntero)
                $resultado=mysql_query($sql,$dbh);
                mysql_close($dbh);
                //Retorna lo obtenido
                return $resultado;
        }
}
?>
```

Anexo X: Programación del seguimiento de dos colores con adaptación de velocidad y giro según última posición del objetivo (Etapa 6)

```
#include <math.h>
#include "Aria.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char **argv)
 {
 //empieza inicialización de servicios, clases, etc
  Aria::init();
  ArArgumentParser parser(&argc, argv);
  parser.loadDefaultArguments();
  ArSimpleConnector simpleConnector(&parser);
  ArRobot robot;
  ArSonarDevice sonar;
  ArAnalogGyro gyro(&robot);
  robot.addRangeDevice(&sonar);
  ArKeyHandler keyHandler;
  Aria::setKeyHandler(&keyHandler);
  robot.attachKeyHandler(&keyHandler);
  printf("You may press escape to exit\n");
  ArSensorReading reading;
  ArTcpConnection con;
  Aria::init();
 //termina inicialización de servicios, clases, etc

 //Revisa el analizador de comandos
  if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
                {
                Aria::logOptions();
                exit(1);
                }
        //termina la revisón del analizador de comandos


        //Revisa si el Robot está conectado, si no está conectado sale del programa
 if (!simpleConnector.connectRobot(&robot))
                {
                printf("Could not connect to robot... exiting\n");
                Aria::exit(1);
                }
 //Termina revisión de la conexión

 //Declaración de variables de variables
        float cuenta1, cuenta2, bandera;          //variables de entrada
        float band, xmax, ungrado, resta, angulo, valorx, valory, noencuentra;
        xmax=117; // ancho de la imagen (en pixeles)
        noencuentra=1;
        robot.runAsync(true);
```

```
        robot.enableMotors();                          //habilita los motores del robot
        bool first = true;
        int bandex, deroizq;
        printf("\r");
        FILE *ptrCf;
        //puntero para la lectura de archivos de texto
 //Termina declaración de variables

 //Inicia la actividad del robot
        while (Aria::getRunning())
                {
                        //abre el while cuando se activa el robot
//adaptación de rutina de lectura de archivos de texto[35]

                        //lee bandera
                if((ptrCf=fopen("bandera.txt", "r"))== NULL)
                {
                        printf( "El archivo no puede abrirse\n");
                }
                        else
                {
                fscanf( ptrCf, "%15f", &bandera);
                while ( !feof( ptrCf))
                        {
                                fscanf ( ptrCf, "%10f", &bandera);
                        }
                                fclose ( ptrCf);
                }

                if (bandera==1)
                {
        //leer archivo de coordenada x
                if((ptrCf=fopen("coordenada.txt", "r"))== NULL)
                        {
                                printf( "El archivo no puede abrirse\n");
                        }
                        else
                        {
                        printf( "matlab da:\n");
                        fscanf( ptrCf, "%15f", &cuenta1);
                        while ( !feof( ptrCf))
                                {
                                        printf("en x:");
                                        printf( "%5f", cuenta1);
                                        fscanf ( ptrCf, "%10f", &cuenta1);
                                        printf("\n");
                                }
                                fclose ( ptrCf);
                        }

//leer archivo de coordenada en y

                if((ptrCf=fopen("coordenada2.txt", "r"))== NULL)
                        {
                        printf( "El archivo no puede abrirse\n");
                        }
```

```
                else
                        {
                        printf( "matlab da:\n");
                        fscanf( ptrCf, "%15f", &cuenta2);
                        while ( !feof( ptrCf))
                        {
                                printf("en y:");
                                printf( "%5f", cuenta2);
                                fscanf ( ptrCf, "%10f", &cuenta2);
                                printf("\n");
                        }
                        fclose ( ptrCf);
                }
                FILE *ptrCf2;
                if((ptrCf2=fopen("bandera.txt", "w"))== NULL){
                printf( "El archivo no puede abrirse\n");
        }
        else {
                band=0.0;
                        fprintf( ptrCf2, "%10f", band);
                        fclose ( ptrCf2);
                }

        bandex=1;
        valorx=cuenta1;
        valory=cuenta2;
                        }

// +90° es a la izquierda
// -90° es a la derecha

// Si la lectura de los archivos de texto se encuentra
// en el rango de la derecha
if(valorx>=0 && valorx<39 && bandex==1)
{
        if (robot.isHeadingDone())//verifica que no exista un giro previo
        {
        resta=valorx-(xmax/2);          // cálculo del ángulo
        ungrado=46/xmax;                // de
        angulo=-1*resta*ungrado;          // giro
        printf("el angulo a girar es:");
        printf("%f", angulo);
        robot.setRotVel(1000000);
        robot.setDeltaHeading(angulo);
        bandex=0;
        }
        else                                    //en caso de haber un giro previo
        {                                       //avisa que no puede girar
        printf("todavia no termino");
        bandex=0;
        }
}
//Termina giro a la izquierda
// Si la lectura de los archivos de texto se encuentra
// en el rango de la derecha
if(valorx>=78 && valorx<=117 && bandex==1)
```

```
{
        if (robot.isHeadingDone())//verifica que no exista un giro previo
        {
        resta=valorx-(xmax/2);          // cálculo del ángulo
        ungrado=46/xmax;                                    //de
        angulo=-1*resta*ungrado;                    //giro
        printf("el angulo a girar es:");
        printf("%f", angulo);
        robot.setRotVel(10000000);
        robot.setDeltaHeading(angulo);
        bandex=0;
        }
        else                                                //en caso de haber un giro previo
        {                                                   //avisa que no puede girar
        printf("todavia no termino");
        bandex=0;
        }
}
//Termina giro a la derecha

// Si la lectura de los archivos de texto se encuentra
// en el rango del centro
else if(valorx>=39 && valorx<78)
{
if(robot.getSonarRange(1)<=800 || robot.getSonarRange(2)<=800 ||
robot.getSonarRange(3)<=800 || robot.getSonarRange(4)<=800 || robot.getSonarRange(5)<=800
|| robot.getSonarRange(6)<=800 )
                        {
                        robot.stop();// si la lectura de los sonares es muy
                                        //cercana el robot no avanza
                        }
else
        {
                robot.setVel(300);// si la lecutra de los sonares es mayor a
                                        //0.8m el robot avanza a 300mm/s
        }
}


//En caso de no encontrar un objetivo (matlab manda un 400.0)
```

Anexo Xa

```
else if(valorx==400.0 && bandex==1)
{
                                printf("papa no aparece\n");
                                robot.stop();
                                robot.setRotVel(200); //el robot gira en la

                                while(valorx==400.0)
                                {
                                        banderita=1;
                                        goto leer;
                                        regreso:
                                        banderita=0;
                                }
```

```
                        robot.stop();

                    }

}
    Aria::shutdown();
    return 0;
    }
```